

Resource Allocation for Energy Efficient k -out-of- n System in Mobile Ad Hoc Networks

Chien-An Chen[†], Myounggyu Won[†], Radu Stoleru[†], Geoffrey G. Xie[‡]

[†]Department of Computer Science and Engineering, Texas A&M University

[‡]Department of Computer Science, Naval Postgraduate School
{jaychen, mgwon, stoleru}@cse.tamu.edu, xie@nps.edu

Abstract—*Resource Allocation* has been widely used for improving various performance metrics in wireless networks. Applying resource allocation to a Mobile Ad Hoc Network (MANET), however, is a challenging problem because of dynamic network topology. In this paper, we develop a novel resource allocation scheme designed for MANETs that minimizes the communication cost for accessing distributed resources while improving the reliability by adopting the k -out-of- n system, a widely used technique for reliability control. Specifically, we propose a scheme that allocates resources to n nodes, called *service centers*, such that the expected energy consumption for nodes to access k service centers out of the n service centers ($k \leq n$) is minimized. Our scheme accounts for dynamic network topology by estimating the failure probabilities of nodes and monitoring the network for significant topology changes. In addition, an *Importance Sampling technique* is used to reduce the computation-overhead. To evaluate the performance, we build a mobile distributed file system based on our resource allocation scheme. Through both extensive simulations and real hardware implementation on Smartphones, we show that our resource allocation scheme effectively reduces energy consumption by up to 45% and increases the successful data retrieval rate by up to 50% in comparison with a greedy algorithm.

I. INTRODUCTION

Resource allocation has been widely used for achieving various objectives in distributed computing research. The main idea is to strategically break a resource-intensive task into pieces (e.g., smaller data fragments or smaller processing units) and allocate the pieces to multiple nodes called *service centers*, so that each service center concurrently processes the pieces and offers services to nodes. This way resource allocation improves the system performance. Resource allocation is also important for improving the reliability of a system. For example, a mobile distributed file system uses resource allocation to distribute file fragments to multiple service centers so that data can be stored more reliably.

A *Mobile AdHoc Network* (MANET) is an infrastructureless network, where mobile nodes can communicate with each other while freely moving. Since a MANET requires no infrastructure, it is useful for applications where no infrastructure is available, e.g., disaster response [1] and military [2]. However, due to the dynamic nature of MANETs (e.g., node mobility, uncertainty in communication links, and the limited energy of nodes), directly applying *resource allocation* to MANETs is very challenging. For example, depending on the network topology, accessing service centers may incur large overhead, or some service centers may become unreachable. Therefore,

resource allocation must estimate topological changes and take the estimation into account in selecting service centers. At the same time, resource allocation must ensure energy efficiency by minimizing the energy consumption of the system for accessing the service centers. In addition, resource allocation must ensure reliability; the services running on the network should be resilient to some degree of node failures, i.e., given n service centers, in situation when some service centers fail, the system should provide full functionality as long as k or more of the service centers ($k \leq n$) are still available. The ability of tolerating partial failures is described as the “ k -out-of- n system”, which is a commonly used technique in reliability control [3].

To the best of our knowledge, we are the first to investigate the k -out-of- n resource allocation problem under the circumstances of dynamically changing topology; more precisely, in this problem, considering the dynamic nature of a MANET, we select a set of n service centers among a total of N nodes in the network such that the expected energy consumption for any node to access k service centers among the selected n service centers is minimized ($k \leq n$). Specifically, we provide a stochastic model for estimating the *failure probability* of nodes and integrate the model into our problem. To reduce the computational overhead for solving our problem, we employ the *Importance Sampling* technique [4]. We also propose a distributed algorithm to monitor the network topology so that our estimated solution can be updated whenever a “significant topology change” is detected. Finally, a novel routing mechanism called the *multi-destination route discovery* is introduced to allow nodes to access k service centers more efficiently.

We implement our *resource allocation scheme* as a middleware. An application determines k and n values based on its reliability requirements and passes the values to the middleware as parameters. The middleware then provides a set of service centers selected by our algorithm to the application and notifies the application whenever the selected service centers need to be updated due to topology changes. We evaluate our resource allocation scheme by implementing a Mobile Distribute File System (MDFS) with our resource allocation scheme on Android phones. We show that the MDFS based on our scheme is more reliable and energy efficient when compared with the state-of-art solution [2].

The rest of this paper is organized as follows. Section II reviews the state of art. We formally define our problem in

Section III and present our proposed solution in Section IV. In Section V, we give an example application which uses our proposed scheme. We then present the hardware implementation of our resource allocation scheme and our example application in Section VI. Simulation results are presented in Section VII. We conclude and present ideas for future work in Section VIII.

II. RELATED WORK

Resource allocation has been a well researched topic in distributed computing systems. Resource allocation achieves diverse objectives depending on applications. Those objectives include optimizing latency, energy, reliability, or quality of service. Since most distributed computing platforms are dedicated for data-intensive and high-computational tasks, most previous works have focused on solving the problems in a static network.

Alicherry and Lakshman proposed a 2-approximation algorithm for the optimal selection of data centers that serve nodes requesting virtual machines [5]. Their algorithm minimizes the maximum distance, or latency between any pair of selected data centers. Beloglazov et. al. solved the similar problem by applying their Modified Best Fit Decreasing algorithm [6]. The algorithm places virtual machines in a set of data centers such that the total power consumption is minimized. Agarwal et. al. proposed a system that automatically migrates data across geographically distributed data centers [7]. It reduces data center capacity skew, inter-data-center traffic and latency by using an iterative optimization algorithm based on the weighted spherical means. The main difference between these solutions and ours is that they focus on resource allocation in a static network consisting of powerful servers, while we emphasize on the dynamic network consisting of mobile devices.

Many recent works have utilized resource allocation specifically for data allocation in distributed systems [8][9]. As cloud storage has become a major backbone for many network services, efficiently allocating data on servers to minimize the communication cost is an important problem. Liu et. al. proposed an Energy-Efficient Scheduling (DEES) algorithm that aims to save energy by integrating the process of scheduling tasks with data placement strategies [8]. Yuan et. al. proposed a matrix based k-means clustering strategy for data placement in scientific cloud work flows [9]. The algorithm groups an existing data set into k data centers based on their dependencies and dynamically clusters newly incoming data. However, these solutions assume a static network; thus, they cannot be directly applied to MANETs.

Resource allocation has also been important for achieving better reliability. Specifically, resources are duplicated to multiple locations so that the service provision is resilient to node failures. In particular, the “k-out-n subsystem” describes a system consisting of n nodes that can function properly as long as k nodes among the n nodes function correctly [3]. Dimakis et. al. proposed several erasure coding algorithms for distributed storage system and described maintenance schemes for repairing the system when nodes fail [10][11][12]. Aguilera

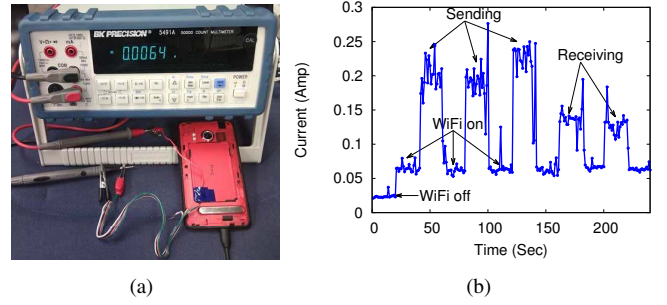


Fig. 1. (a) A setup for energy measurements; (b) Current drawn by a HTC Evo 4G device at idle, sending, and receiving states when Wi-Fi is on.

et. al. proposed a protocol to efficiently adopt erasure code for better reliability [13]. These works, however, focus on only the system reliability. In this paper, we attempt to increase the reliability (i.e., by adopting the k -out-of- n concept) with minimum communication cost.

In dynamic networks, achieving higher retrieval rates from nodes to service centers is important for resource allocation. Leong et. al. proposed an algorithm for optimal resource allocation that increases retrieval rates [14]. Their algorithm determines the amount of data stored on each node to maximize the probability of successfully recovering the original data. Jiang et. al. formulates the fragment distribution scheme as a coloring problem [15]. Their algorithm finds a placement that minimizes the number of hops required for any mobile host to retrieve data from multiple static gateways. Although the nodes in the network are mobile, the gateways storing resources are assumed to be static. In our problem, however, any node can be selected as a service center and all nodes can move freely.

III. PROBLEM FORMULATION

We consider a mobile ad-hoc network consisting of N nodes denoted by a set $V = \{v_1, v_2, \dots, v_N\}$. For convenience, we will use i and v_i interchangeably hereafter. The network is modeled as a graph $G = (V, E)$, where E is a set of edges between nodes. We assume that an edge $e_{ij} \in E$ between two nodes v_i and v_j exists if the Euclidean distance between the two nodes is smaller than R , where R is the communication range of a node. Each node is associated a *failure probability* defined as the probability that the node fails. We denote the failure probability of node v_i by $P[f_i]$, where f_i is the event that node v_i fails.

We assume that, for a mobile device in MANET, data transmission/reception is the major source of energy consumption. To support this assumption, we quantify the consumed energy of a node by measuring the current drawn by an HTC Evo 4G Smartphone in different modes (i.e., transmit, receive, and idle modes). Figure 1(a) shows the setup for our experiments; Figure 1(b) depicts the results. As shown, when the device is in sending and receiving modes, it consumes significantly larger energy than when it is in idle mode. Based on this assumption, we define the following terms to represent the consumed energy in the network when node i sends data to node j .

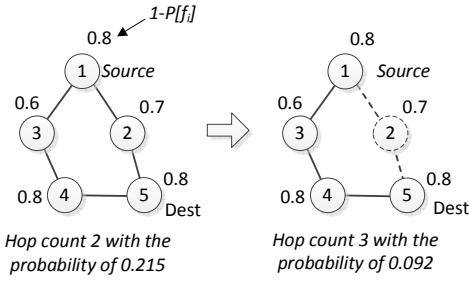


Fig. 2. Two possible graphs for computing the expected distance between node 1 and node 5.

Definition 1: The *Expected Distance* between two nodes i and j denoted by d_{ij} is the expected number of hops for possible paths connecting the two nodes, considering the possible nodes or path failures in a network. ■

Figure 2 illustrates an example. There are five nodes, i.e., nodes 1 to 5. Each node i is annotated with probability $(1 - P[f_i])$. Say that node 1 is a sender and node 5 is a receiver. Depending on which nodes fail (e.g., due to mobility or depleted energy), there are 2^5 possible graphs. For example, if no node fails, as shown in Figure 2(a), which happens with probability $(0.8 \times 0.7 \times 0.6 \times 0.8 \times 0.8 = 0.215)$, the hop count of the shortest path is 2. If we consider a graph where node 2 fails as shown in Figure 2(b), which happens with probability $(0.8 \times 0.3 \times 0.6 \times 0.8 \times 0.8 = 0.092)$, the hop count of the shortest path becomes 3. This way if we consider all possible graphs, the expected distance can be calculated (i.e., $2 \times 0.215 + 3 \times 0.092 + \dots$). Of course, considering all possible 2^N graphs is infeasible. We will address this challenge later.

Before we formulate our k -out-of- n resource allocation problem, we define several other terms as follows:

Definition 2: The *Distance Matrix* D is an $N \times N$ matrix, where an element at i -th row and j -th column corresponds to the expected distance d_{ij} , $i, j \in V$. ■

Definition 3: The *Relationship Matrix* R is a $N \times N$ matrix which predefines the relationship between nodes and service centers; more precisely each element R_{ij} is a binary variable indicating that if R_{ij} is 0, node i will never access service center j ; if R_{ij} is 1, node i will always choose service center j . ■

The relationship matrix R is used to add constraints in the relationship between nodes and service centers. For example, a constraint $\{R_{i5} = 0, \forall i = 1, \dots, N\}$ implies that node 5 will never be selected as a service center and a constraint $\{R_{5j} = 0, \forall j = 1, \dots, N\}$ implies that node 5 will never request a service.

Definition 4: The *service center list* X is a vector containing service centers, i.e., its element X_i is a binary variable indicating that if $X_i = 1$, v_i is a service center. ■

Now we are ready to formulate our k -out-of- n resource allocation problem. In this problem, we are interested in finding a set of n service centers denoted by a set $S = \{s_1, s_2, \dots, s_n\}$, $S \subseteq V$ such that the *total expected distance* from every node in the network to k service centers among

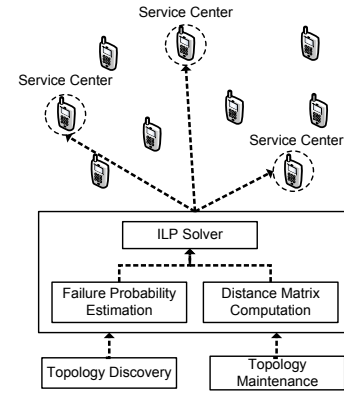


Fig. 3. An overview of our resource allocation scheme.

the n service centers is minimized. We formulate this problem as an Integer Linear Program (ILP) as follows:

$$R_{opt} = \arg \min_R \sum_{i=1}^N \sum_{j=1}^N D_{ij} R_{ij} \quad (1)$$

$$\text{Subject to: } \sum_{j=1}^N X_j = n \quad (2)$$

$$\sum_{j=1}^N R_{ij} \geq k \quad \forall i \quad (3)$$

$$X_j - R_{ij} \geq 0 \quad \forall i \quad (4)$$

$$X_j \in \{0, 1\} \quad \forall j, R_{ij} \in \{0, 1\} \quad \forall i, j \quad (5)$$

The first constraint (Eq 2) means that exactly n service centers will be selected; the second constraint (Eq 3) indicates that each node has access to at least k service centers; the third constraint (Eq 4) makes sure that j^{th} column of R can have none zero values only if X_j is 1; and the constraints in Eq 5 are requirements for the decision variables.

To solve our problem, several research questions need to be answered: i) How to model the *failure probability* of nodes and apply it to our problem? ii) How to compute the *distance matrix* more efficiently without considering all possible 2^N graphs? iii) How to monitor potential changes in the network topology in real time, so that we can maintain the most up-to-date solution? iv) How to allow nodes to access service centers more efficiently? In the rest of the paper, we address the above challenges and derive a near optimal solution for the k -out-of- n resource allocation problem.

IV. RESOURCE ALLOCATION FOR k -OUT-OF- n SYSTEM

In this section, we first present an overview of our resource allocation scheme and then provide the details of the scheme in the following subsections. Our resource allocation scheme consists of four main components as shown in Figure 3: Topology Discovery, Failure Probability Estimation, Distance Matrix Computation, and Topology Maintenance. To allocate resources in the network, a node first runs the Topology Discovery component. The component collects the neighbor

information and the failure probability estimated at each node. Knowing the failure probabilities of all nodes in the network, the node can compute the expected distance matrix. Based on the distance matrix, the node then solves our ILP problem defined in Eq. 1 to locate n service centers. The Topology Maintenance component continuously monitors the network for any significant topology change and updates the solution when necessary. The following subsections describe the details of each component of our scheme.

A. Topology Discovery

For resource allocation, a node first runs Topology Discovery to collect a global view of the network – a node floods a control packet throughout the network. Upon receiving the control packet, nodes reply with their neighbor information and calculated failure probabilities. Based on the information collected from the network, the sender finds an optimal set of service centers. Because the topology discovery process incurs high communication overhead, our Topology Maintenance algorithm in section IV-D tries to minimize the frequency of running topology discovery procedure.

B. Failure Probability Estimation

The failure probability of a node estimated at time t is the probability that the node may fail before time $t + T$, where T is a time interval during which the estimated failure probability will be effective. When we estimate the failure probability of a node, we consider the following events causing the failure: energy depletion, temporary disconnection from a network (e.g., turned off by a user or losing the connection), and application-specific reasons. We assume that these events happen independently. Let f_i be the event that node i fails; specifically, we let f_i^E, f_i^C , and f_i^A be the events that node i fails due to energy depletion, temporary disconnection from a network, and application-specific reasons, respectively. The failure probability of a node can be written as follows:

$$P[f_i] = 1 - (1 - P[f_i^E]) (1 - P[f_i^C]) (1 - P[f_i^A])$$

In the following sections, we present how to estimate $P[f_i^E], P[f_i^C]$, and $P[f_i^A]$ to compute $P[f_i]$.

1) *Failure by Energy Depletion*: Estimating the remaining energy of a mobile device is a well researched problem [16][17][18]. We adopt the remaining energy estimation algorithm in [18] because of its simplicity and low overhead. The algorithm uses the history of periodic battery voltage readings to predict the battery remaining time. Considering that the error for estimating the battery remaining time follows the normal distribution [16][19], we find the probability that the battery remaining time is less than T by calculating the cumulative distributed function (CDF) at T . In the following formulation, the predicted battery remaining time x is a random variable with mean μ and standard deviation σ .

$$\begin{aligned} P[f_i^E] &= P[\text{Remaining time} < T \mid \text{Current Energy}] \\ &= \int_{-\infty}^T f(x; \mu; \sigma^2) dx, \quad f(x; \mu; \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \end{aligned}$$

2) *Failure by Temporary Disconnection from a Network*: Nodes can be temporarily disconnected from a network by many causes (e.g., due to the mobility of nodes, or simply when users turn off the devices). The probability of temporary disconnection differs from application to application, but this information can be inferred from the history: a node gradually learns its behavior of disconnection and eventually creates a probability distribution of disconnection. As many researchers have modeled the network connectivity by Poisson distribution, we adopt it in our experiment [19]. Let T_{off_per} be the expected period of time during which a node will be temporarily disconnected from a network; T_{off_per} can be estimated based on the historical records storing the times when the node was disconnected from a network. Let T_{last_off} be the elapsed time since the device was last inaccessible. Then, the expected number of times when the node is inaccessible between times $(t - T_{last_off})$ and $(t + T)$, denoted by λ , is $(T_{last_off} + T)/T_{off_per}$. We can now compute the probability of failure due to temporary disconnection from a network as the following:

$$\begin{aligned} P[f_i^C] &= P[\text{Number of inaccessible times} \geq 1] \\ &= \sum_{k=1}^{\infty} f(k; \lambda), \quad \text{where } f(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}. \end{aligned}$$

3) *Failure by Application-specific Requirements*: Depending on applications, some nodes may have different rules than other nodes. For example, in a military application, some nodes may be exposed to higher danger than others or some nodes may be equipped with better defense capability. Such application-dependent restrictions provide failure probability $P[f_i^A]$, which is usually explicitly known, not requiring any estimation process.

C. Distance Matrix Computation

Given the failure probability of nodes, we can find the distance matrix D by calculating the expected distance between any two nodes. The rationale behind the notion of expected distance instead of simply using a shortest path on a given global topology is that, in mobile environments, the shortest path can easily change because of the change of topology. Each node has certain failure probability and the failure of any node may alter the graph, resulting in a different shortest path. Thus, we must consider shortest paths for all possible graphs when we determine the distance between two nodes.

However, one difficult problem is that given a set of N nodes, there are many possible graphs to be considered (i.e., a total of 2^N possible graphs); thus, it is infeasible to deterministically calculate the expected distance when the network size is large. To account for this computational issue, we adopt the *Importance Sampling technique*, one of the Monte Carlo methods, to approximate the expected distance matrix. The *Importance Sampling* allows us to approximate the value of a function by evaluating multiple samples drawn from a sample space with a known probability distribution function.

The function to be approximated in our problem is the distance matrix D .

A sample graph is obtained by considering each node as an independent Bernoulli trial, where the success probability of the Bernoulli trial for node i can be simply defined as the following:

$$p_{X_i}(x) = \begin{cases} (1 - P[f_i]), & \text{if } x = 1 \\ P[f_i], & \text{if } x = 0 \end{cases}$$

$$= (1 - P[f_i])^x P[f_i]^{1-x} \text{ for } x \in \{0, 1\}$$

Then, a set of sample graphs can be defined as a multivariate Bernoulli random variable B with a probability mass function $p_g(b)$ defined as the following:

$$p_g(b) = p_{X_1, X_2, \dots, X_N}(x_1, x_2, \dots, x_N)$$

$$= P[X_1 = x_1, X_2 = x_2, \dots, X_n = x_n]$$

$$= \prod_{i=1}^N p_{X_i}(x_i)$$

Accordingly, each element of B is a $1 \times N$ vector denoted by b , where the element $b[i]$ of the vector indicates whether node i survives or fails in a sample graph represented by b .

Having defined our sample, we determine the number of required samples by checking the variance of the expected distance matrix denoted by $\text{Var}(E[D(B)])$, where the expected distance matrix $E[D(B)]$ is defined as the following:

$$E[D(B)] = \left(\sum_{i=1}^n D(x) p_g(x) \right)$$

In a Monte Carlo Simulation, the true $E[D(B)]$ is usually unknown, so we use the expected distance matrix estimator, $\tilde{D}(B)$, to calculate the variance estimator, denoted by $\widehat{\text{Var}}(\tilde{D}(B))$. The expected value estimator and variance estimator below are written in a recursive form and can be computed efficiently at each iteration.

$$\tilde{D}(B_K) = \frac{1}{K} \left((K-1) \tilde{D}(B_{K-1}) + D(b_K) \right)$$

$$\widehat{\text{Var}}(\tilde{D}(B_K)) = \frac{1}{K(K-1)} \sum_{i=1}^K \left(D(b_i) - \tilde{D}(B_K) \right)^2$$

$$= \frac{1}{K} \left(\frac{1}{K-1} \sum_{i=1}^K [D(b_i)]^2 - \frac{K}{K-1} \left(\tilde{D}(B_K) \right)^2 \right)$$

Here, the Monte Carlo estimator $\tilde{D}(B)$ is an unbiased estimator of $E[D(B)]$, and K is the number of samples used in the Monte Carlo Simulation. The simulation continues until $\widehat{\text{Var}}(\tilde{D}(B))$ is less than dist_var_{th} , a user defined threshold depending on how accurate the approximation has to be. We chose dist_var_{th} to be $\frac{1}{10}$ of the smallest node-to-node distance in $\tilde{D}(B)$.

In Figure 4, we compare the expected distance found by using Importance Sampling with the true expected distance found based on a exhaustive search in a network of 16 nodes. The *Root Mean Square Error* (RMSE) on the Y -axis is

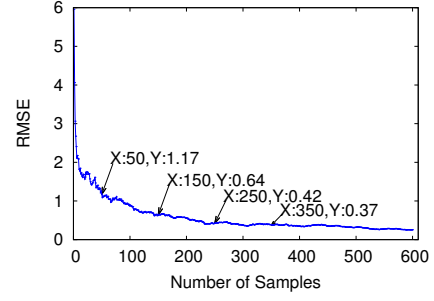


Fig. 4. The Root Mean Square Error (RMSE) between the true expected distance and the approximation from Importance Sampling.

computed between the true expected distance matrix and the approximated expected distance matrix at every iteration. It can be seen that the approximated expected distance has a difference of only 0.5 hop compared with the true expected distance when 200 samples are used.

After the expected distance matrix is computed, we are ready to solve our ILP, which outputs n service centers in the network. A remaining issue is that, although we account for the stochastic nature of MANET, when the network topology has significantly changed, the previous allocation may deviate too much from the optimal one currently and require updates. In the following section, we present a distributed algorithm that monitors the network topology for any significant change.

D. Topology Maintenance

This section presents the details of our distributed algorithm designed for efficiently monitoring the network topology so that we can update the solution if a significant topology change is detected. To describe the algorithm, we first define several notations. A term s refers to a state of nodes which can be either U and NU , where U means that the local topology has changed significantly (i.e., significant changes in its neighbor table entries), while NU means not much changes in the local topology. Node ID is denoted by ID. We denote the number of changed neighbor table entries by p . Parameters τ_1 and τ_2 determine the thresholds for topology changes: a set \mathcal{ID} contains the IDs of nodes with p greater than τ_1 ; τ_2 is an application-dependent threshold indicating significant global topology changes.

Our algorithm first selects one node as *topology_delegate* V_{del} who is responsible for maintaining the global topology information. To minimize the message propagation delay from nodes to V_{del} , V_{del} is assigned to the node that is closest to the geometric center of the network; if the current V_{del} moves too far from the geometric center, V_{del} is reassigned. When V_{del} is notified that the global topology has changed significantly, it initiates topology discovery. Any other node who needs global topology information later can simply query from V_{del} . An important aspect of our algorithm is that it does not incur additional communication overhead because it runs based on beacon messages. The algorithm is depicted in Algorithm 1. If p of a node is greater than the threshold τ_1 , the node

Algorithm 1 Distributed Topology Monitoring

```

1: At each beacon interval:
2: if  $p > \tau_1$  and  $s \neq U$  then
3:    $s \leftarrow U$ 
4:   Put  $+ID$  to a beacon message.
5: end if
6: if  $p \leq \tau_1$  and  $s = U$  then
7:    $s \leftarrow NU$ 
8:   Put  $-ID$  to a beacon message.
9: end if
10: Upon receiving a beacon message on  $V_i$ :
11: for each ID in the received beacon message do
12:   if  $ID > 0$  then
13:      $ID \leftarrow ID \cup \{ID\}$ .
14:   else
15:      $ID \leftarrow ID \setminus \{ID\}$ .
16:   end if
17: end for
18: if  $|\{ID\}| > \tau_2$  then
19:   Notify  $V_{del}$  and  $V_{del}$  initiate topology discovery
20: end if
21: Add the ID in its beacon message.

```

changes its state to U and puts its ID in a beacon message. Whenever a node with state U finds that its p becomes smaller than τ_1 , it changes its state back to NU and puts $-ID$ in a beacon message. Upon each node receiving a beacon message, it checks IDs in a beacon message. For each ID in the received beacon message, it adds the ID to set ID if the ID is positive and deletes the ID from set ID if the ID is negative. Naturally, the size of set ID maintained in each node represents the degree of the topology change. In this way, nodes in the network maintain up-to-date information about the topology change in a distributed manner. In particular, when V_{del} finds that the size of ID is greater than τ_2 , it initiates topology discovery. After V_{del} completes the topology update, all nodes reset their status variables back to NU and update P to zero.

E. Multi-Destination Route Discovery

Once we compute the optimal service centers, routing paths to these service centers can be efficiently found by using a slightly modified routing protocol for MANETs. One simple solution would be to send one Route Request for each service center; but this method is costly and causes broadcast storm problem; thus, we propose a more efficient multi-destination route discovery procedure, specifically designed for our resource allocation scheme.

Let's take AODV as an example, which is a widely used reactive routing protocol for MANETs. In AODV, a source node starts a route discovery by broadcasting a RREQ packet containing the address of a single destination node. Any intermediate node that knows the path to the destination node unicasts a RREP packet back to the source node immediately. The source node keeps only the latest routing path or the routing path with the shortest hop-count. One simple tweak we

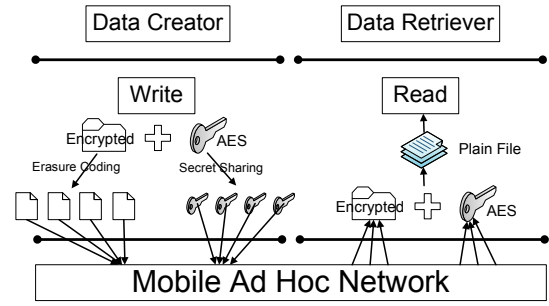


Fig. 5. An overview of MDFS.

add to AODV is to include “multiple” destinations in a RREQ packet. If any intermediate node knows a routing path to one or more destination nodes in a RREQ packet, it puts those paths in a RREP packet and unicasts back to the source node. The intermediate node removes the destination nodes that it has replied to and continues broadcasting the RREQ packet containing only the destination nodes for which routing paths are still unknown. Unlike original AODV, which may have multiple RREQ and RREP packets flooded throughout the network simultaneously, our modification reduces the number of the control packet transmissions significantly. Similar to AODV, when the source node receives information about known routing paths, it selects the routing paths with the latest information or the smallest hop-count.

V. AN EXAMPLE APPLICATION: MDFS

The energy-efficient resource allocation scheme can be useful for many applications in MANETs. As an example, we consider a mobile distributed file system. Specifically, we create a more reliable and energy-efficient mobile distributed file system compared with the state-of-art MDFS [2]. We first review the state-of-art and explain how our resource allocation scheme is used to improve MDFS.

MDFS [2] is an application originally designed for military; thus, it is designed to promote reliability and security. To ensure the reliability and security, MDFS combines three well-established cryptographic primitives: Advanced Encryption Standard (AES), Shamir's Secret Sharing, and Reed-Solomon code. Assume the network size is N . Each file is encrypted by AES and partitioned into N file fragments by Reed-Solomon code. Meanwhile, the key of AES is partitioned into N key fragments by Shamir's Secret Sharing algorithm. All file fragments and key fragments are distributed uniformly throughout the network such that each node can store at most one key fragment and one file fragment for each file. When a node needs to access a file, it has to download at least k file fragments and k key fragments in order to recover the file. This way, a file is recoverable even if several devices are lost (i.e., reliability), and adversaries are not able to access the file even if they compromise multiple devices (i.e., security).

Our resource allocation scheme improves the reliability and energy efficiency of the state-of-art MDFS [2]. See Figure 5 for an overview of our MDFS. The main idea is that, rather

than distributing file and key fragments uniformly in the entire network, we use the resource allocation scheme to allocate fragments to n service centers ($n \leq N$). By doing so, we can achieve the following: First, since our resource allocation scheme takes into account the possible failure of nodes in assigning service centers, each node can retrieve fragments from more reliable service centers; Second, since service centers are assigned such that the energy and latency are minimized for retrieving services, our MDFS achieves higher energy efficiency; Third, by using our distributed topology monitoring algorithm, nodes can still access services even if the topology changes continuously.

VI. HARDWARE IMPLEMENTATION

We implement our resource allocation scheme and MDFS on a HTC Evo 4G Smartphone, which has a 1G Scorpion CPU, 512MB RAM, and a Wi-Fi 802.11 b/g interface; it runs Android 2.3 operating system. Because the Wi-Fi Ad Hoc mode is disabled on Android OS by default, we had to root the device and modify the *wpa_supplicant.conf* file to enable the Ad Hoc mode. The range of the Wi-Fi transmitter is around 100 – 120 meters in an open space without obstacles.

Our resource allocation scheme consists of 4,000 lines of Java code and is completely modularized as a library. Our MDFS simply accesses the service provided by the middleware as an external library. Our MDFS application consists of 5,000 lines of Java code. In particular, we use an Integer Linear Programming Solver [20] for our optimization problem and use [21] for implementing Shamir Secret Sharing and [22] for implementing the Reed-Solomon code. To support multi-hop communication in a MANET, we implement our routing protocol based on the *adhoc-onandroid* [23]. In particular, to allow nodes to retrieve data from multiple destinations more efficiently, we add the Multi-Destination Route Request feature (See Section IV-E) to the original implementation [23].

In order to test both the middleware and MDFS, we design an interface for users to share photos taken by a built-in camera. The size of a picture is approximately 1.5 to 2.5 MB. The system is tested by 10 students in the Disaster City [24] which spans about a $225 \times 225 \text{m}^2$ open space. Each individual is allowed to move freely and takes pictures. The pictures taken are distributed to the network based on our middleware and MDFS. Students then download others' pictures using our MDFS.

The node-to-node distance in our experiment ranges from 1 to 3 hops. The topology discovery takes less than a second. The file distribution time and file retrieval time vary greatly due to the unreliable links or mobility. The retrieval time can be as short as 3 seconds or as long as 12 seconds. The computation times for Monte Carlo Simulation and ILP solver are around 400 millisecond and 2 seconds respectively. Overall, 75% of file retrievals are successful. After carefully inspecting the log file, we discover two major causes of the failures. First, the network may have partitions when a node tries to distribute a file to the network. If not enough nodes are found, the file creation may fail. Second, when retrieving

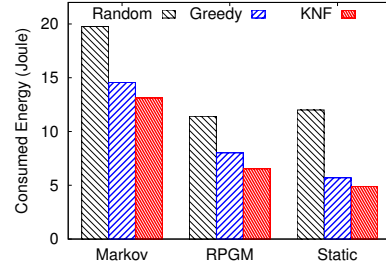


Fig. 6. Effect of mobility.

a file, senders and receivers may be moving, and thus the wireless connection becomes unstable. We believe the first problem can be solved by temporarily caching the fragments and distributing them later when more nodes are available. Packet retransmission should ease the second problem.

VII. SIMULATION RESULTS

For a more comprehensive evaluation of our resource allocation scheme, we conducted a simulation study to evaluate our resource allocation scheme (RAS in short for ease of presentation). We consider a network of $400 \times 400 \text{m}^2$ where different numbers of mobile nodes are randomly deployed. The communication range of nodes is 130m. Mobile nodes move around the network with an average speed of 1m/s based on different mobility models: Markovian Waypoint Model and Reference Point Group Mobility (RPGM). Markovian Waypoint is similar to Random Waypoint Model in which it randomly selects the waypoint of nodes; also it accounts for the current waypoint when it determines the next waypoint. RPGM is a group mobility model where a set of leaders are selected; leaders move based on Markovian Waypoint; and other nodes follow their closest leaders. The performance in a static network is also measured. We employ a modified version of AODV to implement our multi-destination route discovery. We assume that nodes consume energy according to our experimental measurements shown in Figure 1(b). Nodes beacon every 30 seconds.

We compare RAS with two resource allocation schemes: a greedy algorithm and a random placement algorithm. The greedy algorithm greedily selects nodes with the largest number of neighbors as service centers. The random placement algorithm simply randomly selects nodes as service centers. To compare the performance of different schemes, we implement the same MDFS on top of the three schemes. Specifically, we measure the following metrics: 1) average energy consumption for retrieving data (Joule); and 2) successful rate (percent) of retrieving files. We are interested in the effect of the following parameters: 1) mobility model; 2) network density; 3) node speed; 4) k/n ratio; and 5) τ_2 .

A. Effect of Mobility

In this section, we investigate how mobility models affect the performance of different resource allocation schemes. Figure 6 depicts the results. An immediate observation is that, regardless of the mobility models, mobility results in

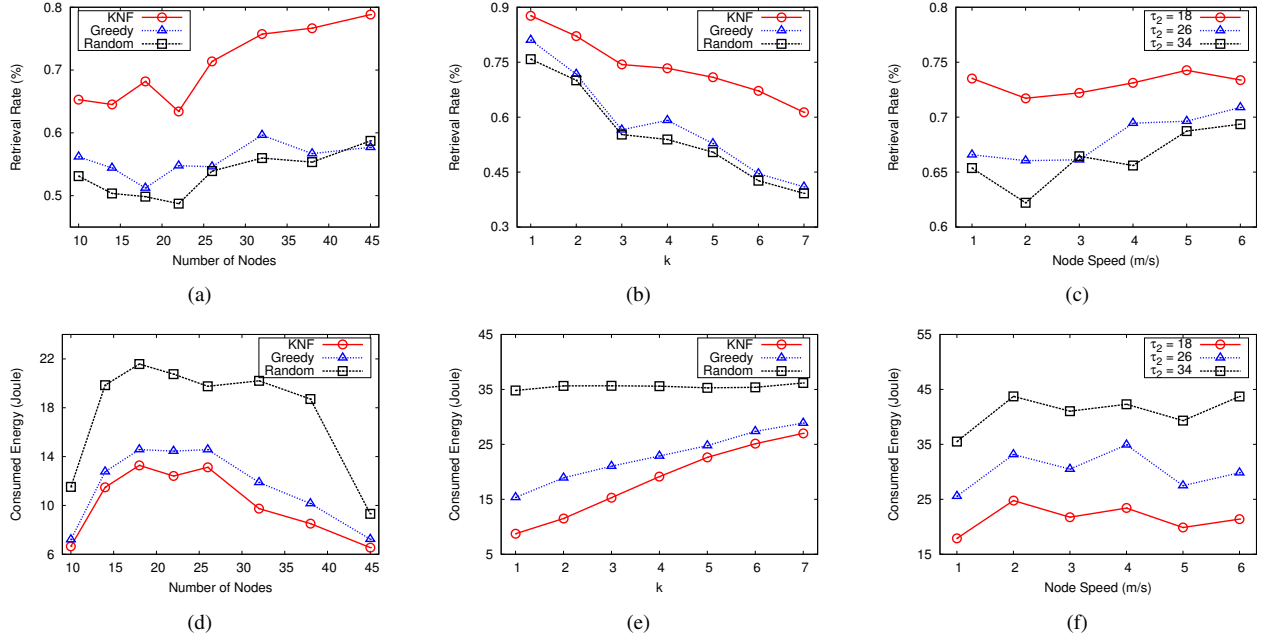


Fig. 7. (a) Effect of node density on data retrieval rate; (b) Effect of k on data retrieval rate; (c) Effect of node speed and τ_2 on data retrieval rate. (d) Effect of node density on energy efficiency; (e) Effect of k on energy efficiency; (f) Effect of node speed and τ_2 on energy efficiency.

higher energy consumption for data retrieval. The reason is straightforward: mobility changes the topology, thereby generating inefficient (i.e., longer) paths from nodes to the service centers. Another interesting observation is that, the difference in energy consumption between the greedy algorithm and RAS when the network is mobile (i.e., Markov and RPGM) is larger than the difference when the network is static. Because RAS includes topology changes in the failure probability estimation, the result indicates that it performs better compared to other schemes in mobile networks. Lastly, we find that the energy consumption for RPGM is smaller than that for Markov. We believe that it is because a service center usually serve nodes in its proximity; thus when nodes move in groups, the impact of mobility is less significant than when all nodes randomly move.

B. Effect of Network Density

In the following sections, all results are based on the Markov Mobility Model. In this section, we study how network density affects the performance of different algorithms. We first measure the average file retrieval rates by varying network density (i.e., by varying the total number of nodes in the network). Figure 7(a) shows the results. As expected, RAS has the highest retrieval rates regardless the network density. It is because RAS takes the dynamic nature of the network into account when selecting service centers, while the other two schemes do not. Another important observation is that the retrieval rates increase with higher network density. There are two possible explanations for this trend: first, with higher network density, it is more likely that shorter paths from nodes to service centers are formed; and second, particularly for RAS, with higher network density (i.e., with more nodes),

there are more choices for selecting service centers, thereby allowing for higher energy efficiency.

We then measure average energy consumption by varying network density. Figure 7(d) depicts the results. As shown, clearly RAS is the most energy efficient, because service centers are selected such that the expected distances from nodes to service centers are minimized. Interestingly, the energy consumption of all three schemes increase for the first few network density (i.e., from 10 to 18). We find the reason is that when the network density is low, inserting more nodes only contributes to forming paths with more hops, leading to higher energy consumption. However, when the network becomes too dense (i.e., starting from 18), the energy consumption of all three schemes starts to decrease. We believe that there are two reasons: first, as the network is saturated with nodes, it is more likely that nodes will find better paths to service centers; second, for RAS, more nodes means more choices for selecting better service centers, leading to higher energy efficiency.

C. Effect of k/n ratio

The parameters k and n are an application-dependent values determining the number of service centers that each node would access for services. Depending on the applications, varying k/n ratio influences the performance (e.g., reliability or energy efficiency) of an application. In this section, we investigate how the k/n ratio influences different resource allocation schemes—we vary k while fixing n . Figure 7(b) shows the results. As expected, the data retrieval rates decreases for all three schemes when we increase k . The reason is because, with larger k , nodes have to access more service centers, eventually increasing the chances of failing to contact

all k service centers. However, since our solution copes with dynamic topology changes, it still yields better retrieval rate in comparison to the other two schemes.

We then show that increasing k degrades the energy efficiency. Figure 7(e) shows the results. Conforming to our expectation, when we increase k , all three schemes consume more energy. One interesting observation is that the consumed energy for the Random scheme does not increase much compared with the other two schemes. Unlike RAS and Greedy, for the Random scheme, all nodes are equally likely to be selected as service centers; therefore, when we run the experiments multiple times with different random selections of service centers, we eventually obtain a similar average energy consumption. In contrast, RAS and Greedy select service centers based on their specific rules; thus, when k becomes larger, nodes may have to communicate with some data centers farther away, leading to higher energy consumption. In particular, it is important to note that since RAS selects service centers more efficiently, it consumes less energy compared with the Greedy scheme.

D. Effect of τ_2 and Node Speed

In this section, we investigate the impact of node speed and parameter τ_2 . Recall that τ_2 is the threshold determining how often RAS should run to cope with topology changes. The parameter indicates the number of nodes with significant local topology changes. Figure 7(d) shows the average retrieval rates of RAS for different τ_2 . We can immediately see that smaller τ_2 allows for higher retrieval rates. The main reason is that smaller τ_2 causes RAS to update the placement more frequently. We are aware that smaller τ_2 incurs overhead for relocating service centers, but as shown in Figure 7(f), energy consumption for smaller τ_2 is still lower than that for larger τ_2 . The reasons are: first, energy consumed for relocating service centers is much smaller than energy consumed for data retrieval; second, not all service centers are relocated - only part of them. Another interesting observation is that, despite higher node speed, both retrieval rates and consumed energy do not increase much. We believe that the results confirm that our maintenance algorithm works correctly: although nodes move with different speeds, our maintenance algorithm reallocates the service centers accordingly such that the performance does not degrade much.

VIII. CONCLUSIONS

In this paper, we investigate the k -out-of- n resource allocation problem in MANETs. Specifically, we propose a resource allocation scheme that finds n service centers in a network with dynamically changing topology, such that the energy consumption for nodes to access k out of the n service centers is minimized. Through extensive simulations and implementation on real hardware, we show the effectiveness of our scheme. As future works, we plan to explore the resource allocation problem in a delay tolerant network which is characterized by its long duration of network disconnection. We also plan to

investigate how to efficiently reallocate the resources and how to reduce the overhead of the maintenance algorithm.

REFERENCES

- [1] S. M. George, W. Zhou, H. Chenji, M. Won, Y. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, "DistressNet: a wireless AdHoc and sensor network architecture for situation management in disaster response," *IEEE Communications Magazine*, vol. 48, no. 3, Mar. 2010.
- [2] S. Huchton, G. Xie, and R. Beverly, "Building and evaluating a k-resilient mobile distributed file system resistant to device compromise," in *Proc. of MILCOM*, 2011.
- [3] D. W. Coit and J. Liu, "System reliability optimization with k-out-of-n subsystems," *International Journal of Reliability, Quality and Safety Engineering*, vol. 7, no. 2, pp. 129–142, 2000.
- [4] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.
- [5] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *Proc. of INFOCOM*, 2012.
- [6] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012.
- [7] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: automated data placement for geo-distributed cloud services," in *Proc. of NSDI*, 2010.
- [8] C. Liu, X. Qin, S. Kulkarni, C. Wang, S. Li, A. Manzanares, and S. Baskiyar, "Distributed energy-efficient scheduling for data-intensive applications with deadline constraints on data grids," in *Proc. of IPCCC*, 2008.
- [9] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1200 – 1214, 2010.
- [10] A. Dimakis and K. Ramchandran, "Network coding for distributed storage in wireless networks," in *Networked Sensing Information and Control*. Springer, 2008.
- [11] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2809 – 2816, 2006.
- [12] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Su, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2010.
- [13] M. Aguilera, R. Janakiraman, and L. Xu, "Using erasure codes efficiently for storage in a distributed system," in *Proc. of DSN*, 2005.
- [14] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocation for high reliability," in *Proc. of ICC*, 2010.
- [15] A. Jiang and J. Bruck, "Diversity coloring for information storage in networks," in *Proc. of ISIT*, 2002.
- [16] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. of CODES*, 2010.
- [17] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *Proc. of MobiSys*, 2011.
- [18] Y. Wen, R. Wolski, and C. Krintz, "Online prediction of battery lifetime for embedded and mobile devices," in *Power-Aware Computer Systems*. Springer Berlin Heidelberg, 2005.
- [19] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson/Prentice Hall, 2008.
- [20] D. L. Berre and A. Parrain, "The sat4j library, release 2.2," <http://www.sat4j.org/>, 2010.
- [21] T. Tiemens, "Shamir secret sharing in java," <http://sourceforge.net/projects/secretsharejava/>, 2012.
- [22] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in C/C++ facilitating erasure coding for storage applications," <https://github.com/tsuraan/Jerasure>, 2011.
- [23] R. K. Jradi and L. S. Reedtz, "Ad-hoc network on android," <https://code.google.com/p/adhoc-on-android/>, 2010.
- [24] "Disaster preparedness and response," <http://www.teex.com>, 2013.