A Scalable and Mobility-Resilient Data Search System for Large-Scale Mobile Wireless Networks

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

Abstract—This paper addresses the data search problem in large-scale highly mobile and dense wireless networks. Current wireless network data search systems are not suitable for large-scale highly mobile and dense wireless networks. This paper presents a scalable and mobility-resilient LOcality-based distRibuted Data search system (LORD) for large-scale wireless networks with high mobility and density. Taking advantage of the high density, rather than mapping data to a location point, LORD maps file metadata to a geographical region and stores it in multiple nodes in the region, thus enhancing mobility-resilience. LORD has a novel region-based geographic data routing protocol that does not rely on flooding or GPSs for data publishing and querying, and a coloring-based partial replication algorithm to reduce data replicas in a region while maintaining the querying efficiency. LORD also works for unbalanced wireless networks with sparse regions. Simulation results show the superior performance of LORD compared to representative data search systems in terms of scalability, overhead, and mobility resilience in a highly dense and mobile network. The results also show the high scalability and mobility-resilience of LORD in an unbalanced wireless network with sparse regions, and the effectiveness of its coloring-based partial replication algorithm.

Index Terms—Data search, Wireless networks, Geographic routing, Topological routing, Distributed hash tables

1 Introduction

Recent technical advancements have enabled the development of a large-scale wireless network (e.g., wireless sensor network (WSN) and mobile ad hoc network (MANET)) consisting of a vast number of mobile nodes dispersed over a wide area. An important problem in such wireless networks is data search. This paper particularly addresses the data search problem in large-scale wireless networks with high mobility and density.

WSNs are used in various applications such as military sensing and tracking, habitat monitoring, health monitoring, environmental contaminant detection, and wildfire tracking. In a WSN, sensors coordinate to perform distributed sensing of environmental phenomena, and collect and share widely-scattered distributed data in a cooperative manner, which makes data search critical to WSNs. Also, considering the dramatic growth of mobile devices (e.g., laptops, smartphones and communication-enabled vehicles) and the restrictions of wired communication, mobile data search applications that enable ubiquitous data access wherever will proliferate in the near future. It is envisioned that there will be omnipresent wireless devices, and some urban areas will be densely covered by ubiquitous mobile nodes (e.g., WiFi enabled cabs in the Manhattan Area) [1]. Therefore, an efficient data search system for a highly mobile and dense wireless network is needed. However, current wireless network data search systems are not suitable for such an environment.

The flooding and local-broadcasting methods in WSNs [2–7] and in MANETs [8] are not energy-efficient due to a tremendously high volume of transmitted messages. Local-broadcasting also cannot guarantee data discovery. In the topological routing based method in MANETs [9–11], nodes advertise their available data, build content tables for received advertisements, and forward data requests to the nodes with high probability of possessing the data. However, this method generates high overhead for advertising and table maintenance. Also, it cannot guarantee data discovery because of possible expired routes in the content tables owing to node mobility.

Geographic routing based data search systems [12–22] have been proposed for WSNs for high scalability. Specifically, a file is mapped to a geographic location based on the distributed hash table (DHT) data mapping policy (a file is mapped to a location whose ID is closest to the file's ID), and stored in a node closest to the geographic location using geographic routing [23–26]. To search a file, a requester calculates the mapped geographic location and uses geographic routing to send the query to the location. However, in a highly mobile network, a file needs to be frequently transferred to its new mapped file holder, which produces high overhead. Also, a delayed data mapping update may lead to a querying failure. In addition, geographic routing needs exact geographic node localization (e.g., (x,y)) using GPS [20, 12] or virtual coordinates [17, 27, 28], which exacerbates overhead burden and increases energy consumption. GPS consumes nodes' precious energy resources and may not provide location information in some situations (e.g. indoors) [27]. The virtual coordinate methods need periodic coordinator updates, which produces high overhead in a highly mobile network.

In order to build a scalable and mobility-resilient distributed data search system for large-scale highly mobile and dense wireless networks, we propose a LOcality-based distRibuted Data search system (LORD). LORD divides the entire wireless network area into a number

• * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*

• *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634. E-mail: {shenh, zel, kangc}@clemson.edu*

of geographic regions. The metadata[1] of a file is mapped to a region using the DHT data mapping policy, and it is stored in all or a subset of the nodes in the region, thus enhancing mobility resilience. A node needs to update data mapping only when it moves across regions, thus reducing maintenance overhead.

LORD has a novel Region-based Geographic Routing (RGR) protocol for data publishing and querying. RGR only requires nodes to know their located regions and region angle information (i.e., Angle Of Arrival (AOA) [29]) using low-power antenna array or ultrasound receivers rather than exact geographic location, thus greatly reducing overhead and energy of geographic routing algorithms. After a node retrieves its queried metadata, it requests different file segments from physically closest file holders. It determines the sizes of segments requested from different file holders based on their distances to minimize the file fetching latency. After receiving the file, the requester publishes the metadata of the file to its mapped regions. We summarize the contributions of this paper as follows:

(1) An efficient and congestion-resilient region-based data publishing and querying protocol, which generates low overhead for a highly mobile network.
(2) An energy-efficient and mobility-resilient Region-based Geographic Routing protocol (RGR), which only requires region angle information based on low-power devices [29].
(3) A parallel file fetching algorithm, which determines several physically close file servers to send different file segments to minimize file retrieval latency.
(4) A coloring-based partial replication algorithm that replicates metadata to a subset of nodes rather than all nodes in a region while maintaining search efficiency.
(5) Extensive experimental results demonstrate the superior performance of LORD in comparison to previous data search systems and the efficiency of LORD components.

The preliminary version of this paper [30] introduces the region-based data publishing, querying protocol and RGR. This version additionally proposes a method to maintain LORD's efficiency in an unbalanced mobile network with sparse regions and the coloring-based partial replication algorithm. The remainder of this paper is organized as follows. Section 2 presents the design of LORD. Section 3 shows the performance of LORD with comparison to previous representative data search systems in wireless networks. Section 4 concludes the paper with remarks on our plans for future work. The supplemental file presents an extension of LORD for unbalanced networks, additional experimental results and a concise review of representative data search systems in wireless networks.

2 The LORD Data Search System

In LORD, the entire geographical area is divided into a number of physical regions (Section 2.1). Each file's metadata is published to its mapped regions, and the requests for a metadata will be forwarded to its mapped regions (Section 2.2). Metadata's publishing and querying rely on the region-based geographic routing (RGR) that sends a message destined to (x, y) to the region with location (x, y) (Section 2.3). After retrieving the metadata, a file requester uses the parallel file fetching algorithm (Section 2.4) to fetch the file. A back-tracking algorithm ensures that a requester still receives its requested metadata or data even if it moves to another region (Section 2.5).

1. The *metadata* of a file records the file's keywords, its mapped region, and the file holder and its region.

TABLE 1: Notations and definition.

Notation	Description
n_i	Node i
R_i	Region R_i
(x, y)	Metadata ID or location
V	Expected channel propagation rate
W	Expected channel transmission rate
L_i	Length of the file segment of selected file holder i
T_i	Transmission latency from selected file holder i
d_i	Distance from selected file holder i
$\overline{d}$	Expected distance between two routing hops
$meta_i$	The metafile stored in region R_i

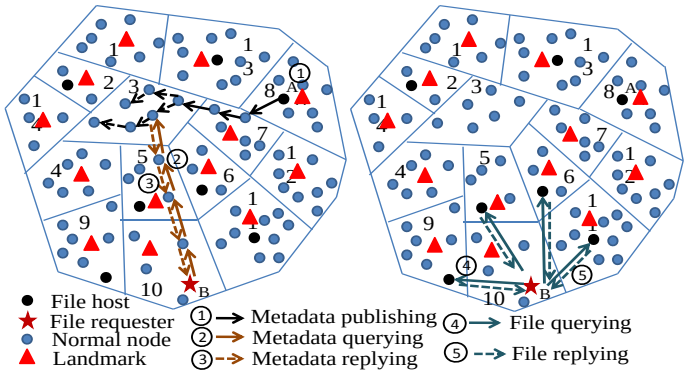


Fig. 1: Steps of file querying in LORD.

Figure 1 shows file querying process in LORD. When file host A in region 8 publishes its file's metadata, it calculates the metadata's ID, say (x, y), and then uses RGR to send it to destination (x, y). The metadata will arrive at region 3, which is located at (x, y), and it is then broadcasted to all (or a subset of) nodes in the region (Step 1). Later on, when node B in region 10 wishes to query the file, using the same process as for data publishing, it first uses RGR to send its metadata query to region 3 (Step 2). If a node's queuing buffer for pending messages to handle is under $k\%$ (k is a threshold), it is lightly loaded; otherwise it is overloaded. A lightly loaded node in region 3 sends the queried metadata to node B by using RGR with the location of node B's region as the destination (Step 3). Node B then chooses multiple physically close file holders and sends queries for file segments (with determined segment sizes) to them (Step 4). Each file holder sends back the requested file segment to node B by using RGR with the location of node B's region as the destination (Step 5). Table 1 lists the main notations used in this paper for easy reference.

We also propose a coloring-based partial replication algorithm that reduces metadata replicas while maintaining the data querying efficiency (Section 2.6). The density of nodes in the area may change and the network may become an unbalanced wireless network with some dense regions and sparse regions. We will present an extension on LORD to maintain the efficiency of LORD

in an unbalanced wireless network (Section 5). Table 1 lists the main notations used in this paper for easy reference.

2.1 Area Partition

We consider a highly mobile and dense wireless network with nodes spreading over an area and are independently and identically distributed (i.i.d.). LORD is proposed for a wireless network with a number of landmarks. Considering the promising ubiquitous computing environment in the future, such static landmarks (e.g., base stations, WIFI access points) will not be difficult to find. Once the landmarks are determined, LORD divides the entire area into a number of regions. A region is the neighboring zone in the transmission range of a landmark and centered by the landmark. Each region is identified by an assigned integer ID. To make LORD adaptive to general case, the regions can be any shape. We design LORD based on irregular shapes (convex polygons), though the regular shape (a special case of irregular shape) would make the design much easier. Figure 1 shows the divided regions and their centered landmarks in a wireless network. We use (x_i, y_i) $(1 \leq i \leq v)$ to denote a convex polygon with v vertices, where x_i and x_{i+1} are adjacent and x_1 is adjacent with x_v. Two vertices are adjacent means the line connecting them is the border of the region. Assume x_1 and x_s $(1 < s < v)$ are the minimal and maximal x-axis values, respectively, which means $x_1 < x_2 < \cdots < x_s$ and $x_s > x_{s+1} > \cdots > x_v > x_1$. Therefore, each region can be represented as a series of border lines:

$$R = \begin{cases} y = k_1 \cdot x + b_1 & (x_1 \leq x \leq x_2) \\ y = k_2 \cdot x + b_2 & (x_2 \leq x \leq x_3) \\ \cdots & \\ y = k_{s-1} \cdot x + b_{s-1} & (x_{s-1} \leq x \leq x_s) \\ y = k_s \cdot x + b_s & (x_{s+1} \leq x \leq x_s) \\ \cdots & \\ y = k_{v-1} \cdot x + b_{v-1} & (x_v \leq x \leq x_{v-1}) \\ y = k_v \cdot x + b_v & (x_1 \leq x \leq x_v) \end{cases} \quad (1)$$

Similar to the maps in the GPS system, the information of geographic boundaries of regions (called *region map*) is configured into a node when it joins in the system initially. We assume that each node has the ability to sense the direction and signal strength from a landmark [29]. The landmarks periodically emit identification beacon signals [31, 29], so that each node can identify the region it is located by a signal received from a landmark in its region. Like the GPS-based geographic routing algorithm that requires each node to have a GPS, a device with at least the AOA capacity is a requirement for each node in LORD. AOA devices consume much lower energy than GPS, thus greatly reducing the deployment cost of a data search system.

The number of landmarks (i.e., regions) (z) can be determined based on the transmission range r of the nodes, and the size s of the entire area. For example, if we want the basic region to be covered by the transmission range of each node, the diameter d of each basic region should satisfy $d < r$ and $\frac{s}{z} < \pi d^2$; that is $z > \frac{s}{\pi d^2}$. In this paper, we only focus on a certain area, such as a campus, a habitat monitoring area or a wildfire tracking area. We will study the case in which nodes expand to cover new territories in our future work.

2.2 Region-Based Data Publishing and Querying

2.2.1 Metadata Publishing and Querying

Locality sensitive hash function (LSH) [32] hashes two similar keyword groups to close values with high probability. LORD uses LSH to hash a file in order to store the metadata of similar files into the same region for similarity search. A file's keywords can be its file name or the keywords retrieved using information retrieval algorithms [33]. The number of LSH hash values of a file can be one or more than one based on the settings of LSH. We use m' to denote the number of hash values of a file produced by LSH.

When a file host publishes the metadata of its file, it first hashes the keywords of the file, denoted as k, using two different LSH functions, H_1 and H_2. The resultant hash values $(H_1^i(k), H_2^i(k))$ $(i = 1, 2...m')$ are normalized to *virtual coordinates* (x_k^i, y_k^i) $(i = 1, 2...m')$, which are used as the IDs of the file. The metadata is mapped to a region that contains the virtual coordinates (x_k^i, y_k^i) in the region map. Then, the data host publishes the metadata to the mapped regions using the RGR routing protocol. The node in a destination region that firstly receives the metadata broadcasts it to all other nodes in the region.

When a mobile node wishes to query a file, it calculates the coordinate (x_k^i, y_k^i) $(i = 1, 2...m')$, of the file's metadata and uses RGR to send requests with (x_k^i, y_k^i) as the destinations. The requests are forwarded to the destination regions, which are exactly the regions that hold the metadata of the queried file. If the first query receiver in the destination region is lightly loaded, it responds to the requester. Otherwise, the query is forwarded to a randomly selected neighbor continuously until reaching a lightly loaded node in the region, which will respond to the requester.

The requester can specify a similarity threshold. The similarity between the keywords of a file k and the queried keywords k_q is calculated by $\frac{|k_q \cap k|}{|k_q \cup k|}$, where $|k|$ denotes the number of keywords in k. The query receiver responds to the requester with the metadata that has a similarity to the queried keywords greater than the required threshold.

2.2.2 Data Mapping Update and Location Update

In a mobile network, it is important to maintain the mapping between data and regions. LORD uses a reactive data mapping update scheme, in which a node conducts updates only when it moves from one region to another region. During the movement, when a node notices that it moves to a different region based on the signal from the landmarks, it drops the old region's metadata and acquires all metadata in the new region from its new neighbor. It also conducts location updates by sending messages to the mapped regions of its file's metadata to update its current location in the metadata. In this way, when the metadata of a file is retrieved, the current locations of the file's hosts can always be acquired.

The geographic routing based (i.e., location-based) data search systems use a proactive mapping update scheme, in which a file holder periodically sends out messages inquiring the current closest node to the file's mapped location, and transfers the file to such a node

if it is found. GLS [20] updates node locations to certain servers when a node's movement distance reaches a threshold for location tracking. Compared to these systems, LORD has three distinctive features: (1) rather than relying on proactive mapping updates, LORD uses region-based reactive mapping updates to ensure querying success, significantly reducing update overhead; (2) unlike location-based data search systems that store a file in a single node closest to the file's mapped location, LORD maps a file to many nodes in a region, thus providing high resilience to congestion and mobility; and (3) unlike other works that only offer exact data searching, LORD's LSH-based data publishing and querying enables similarity data searching in mobile networks.

To guarantee successful data querying, nodes conduct data mapping updates when moving. The average data mapping update frequency equals the average frequency a node moves across different regions in LORD (denoted by f_{LORD}), and it equals the average frequency a node moves away from a location and causes another node to become the closest node to the location in a location-based data search system (denoted by f_{GHT}). Obviously, $f_{GHT} > f_{LORD}$. Therefore, GHT generates higher mapping update overhead than LORD.

LORD is also characterized by metadata storage instead of data storage. Most current wireless data search systems use data storage. Admittedly, data storage methods avoid an additional file querying step after locating the file hosts in the metadata storage methods. However, in a highly mobile and dense wireless network, metadata storage has the advantage in terms of overhead for data mapping updates determined by message size. High node mobility leads to frequent mapped data (metadata or files) transfer. Metadata storage only produces additional operations of small-size metadata querying and replying.

2.3 Region-Based Geographic Routing

When node n_i searches for a file, it first sends out a metadata query to receive the metadata of the file. Second, n_i sends out a file query, and then the queried file will be sent back to n_i. Finally, n_i publishes the metadata for its received file. These different types of messages (including metadata query, metadata for publishing, a metadata reply, a file query and a file) need a routing algorithm to forward them to their destinations. To achieve scalable, mobility-resilient, and energy-efficient locality-aware routing, we propose RGR. RGR only needs AOA [29] information and conducts routing from a source region to a destination region based on the inter-region direction.

A node senses the directions of its neighbor nodes and always chooses the next hop within a certain direction range towards the destination region, in order to route the message along a comparatively shorter path. Given a pair of source region R_i and destination region R_j, R_i's left-side angle range to R_j (denoted by LR) is defined as the angle between the leftmost vertex of R_i to the leftmost and rightmost vertices of R_j, and R_i's right-side angle range to R_j (denoted by RR) is defined as the the rightmost vertex of R_i to the leftmost and rightmost vertices of R_j. For example, in Figure 2, the source region

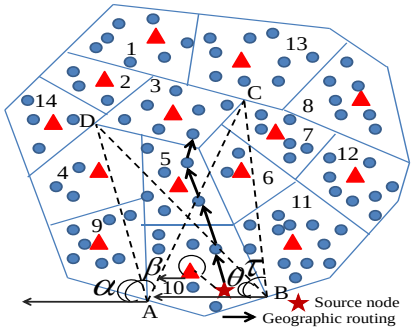


Fig. 2: Region-based geographic routing in LORD.

10's LR and RR to the destination region 3 are $[\alpha, \beta]$ (i.e., $\angle DAC$) and $[\theta, \tau]$ (i.e., $\angle DBC$), respectively. These two angle ranges serve as the tight bounds of message transmission direction towards the destination region. A landmark is always located in the center of a region. For example, assume a node in region 10 intends to forward a message to region 3. We can see that if the transmitting node is located on the right side of the landmark, its message can reach the destination region faster if it is forwarded within triangle $\angle DBC$. If the node is located on the left side of the landmark, its message can reach the destination region faster if it is forwarded within the triangle $\angle DAC$. When a node exchanges "hello" messages with its neighbors, it can sense the transmission signal strength from neighbor nodes and identify the farthest node.

Therefore, in RGR, when a node initiates a message or receives a message, it calculates its region's LR and RR to the destination region, and then decides the next hop for routing. If the node stays in the left side of its region landmark, it chooses the farthest node within $[\alpha, \beta]$ as the next hop node. If the node stays in the right side of its region landmark, it chooses the farthest node within the direction between $[\theta, \tau]$ as the next hop node. The forwarding process is repeated until the message is forwarded to a node in the destination region. Thus, RGR can always forward messages quickly towards the destination region.

In the destination region, for different types of messages, different routing methods are used. If the message is a metadata query, considering the load balance, the message is continuously forwarded until reaching a lightly loaded node. Thus, the replying workload is evenly distributed among the nodes in one region. If the message is metadata for publishing, its destinations are all nodes in the destination region. Thus, it is broadcasted to all nodes in the region. The metadata reply message only targets the file requester and the file query message only targets the file host. Since these two types of messages have a small size, they can be broadcasted to all nodes in the destination region. Since a file has a large size, the file receiver in the destination region first broadcasts a query to establish a path to the file requester in the region and then sends the file through the path.

2.4 Parallel File Fetching Algorithm

After receiving the metadata of its queried file, a requester can retrieve the region IDs of the file's holders. It then locates the file holders in the region map initially

Algorithm 1 Pseudo-code for metadata replying.

```
// Sending a back-tracking message;
while have not received the metadata reply do
    if it moves to a new region then
        Send a back-tracking message to its old region
    end if
end while
// Receiving a back-tracking message;
if receive a back-tracking message then
    Add the message to its back-tracking message list
    Broadcast the message to its neighbors in the region
end if
//Receiving a metadata reply;
if receive a metadata reply with its region as destination then
    if its back-tracking message list contains the message from the
    requester then
        Forward the reply to the requester's current region
        Flood a message in its region to delete the back-tracking
        message for this reply
    else
        if the requester is its neighbor then
            Send the metadata to the neighbor
        else
            Broadcast the metadata to its neighbors
        end if
    end if
end if
```

configured to itself. In order to reduce file fetching latency, LORD uses a parallel transmission algorithm, in which different file segments are simultaneously transmitted from different file holders to the file requester. Since each segment has a shorter data stream than the whole file, the total time period for transmitting all segments to the file requester is shorter than transmitting the whole file from one file holder. Specifically, the file requester chooses geographically close file holders among the located ones, and asks each file holder to transmit a segment of the file. Different segments destined to the same destination may arrive at the same node in routing. Then, this node can merge these segments before forwarding them out in order to save energy for forwarding. Below, we introduce how to determine the length of a file segment in order to minimize the transmission latency of an entire file.

Channel propagation rate is the rate that messages pass through a channel measured by meters/s, and channel transmission rate is the rate that messages are transmitted from a node to a channel measured by bits/s. We use V to denote the expected channel propagation rate, W to denote the expected channel transmission rate, and $\overline{d}$ to denote the expected distance between two routing hops. These expected values can be calculated based on empirical data. These values may change, and they can be periodically calculated to ensure that they represent the current status of the network. We assume that the total length of a file is L, and the number of selected file hosts is m. We use L_i, T_i and d_i to denote the length of the file segment, transmission latency and the distance from a selected file holder i $(1 \leq i \leq m)$ to the requester. Then, the average number of hops between two nodes with distance d_i is $\frac{d_i}{\overline{d}}$. The expected latency for one data segment transmission is $T_i = \frac{Li}{W} \cdot \frac{d_i}{\overline{d}} + \frac{d_i}{V}$. Since $V \ll W$, $\frac{d_i}{V} \approx 0$, then

$$T_i = \frac{L_i}{W} \cdot \frac{d_i}{\overline{d}}. \tag{2}$$

$T_1 = T_2 = T_3 .... = T_m$ is a condition to minimize the file transmission latency. Therefore, $L_i \cdot d_i = L_j \cdot d_j$ $(1 \leq i \leq m,\ 1 \leq j \leq m)$. Since $L_1 + L_2 + ...L_m = L$, we retrieve

$$L_i = \frac{L}{(\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} + ... + \frac{1}{d_n}) \cdot d_i}. \tag{3}$$

According to Equation (3), a requester can determine the length of a file segment transmitted by a data host based on its distance from the requester. Based on Equation (2) and Equation (3), we retrieve:

Proposition 2.1: In the LORD parallel file fetching algorithm, the shortest average latency for file fetching is

$$\frac{L}{(\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} + ... + \frac{1}{d_n}) \cdot \overline{d} \cdot W}. \tag{4}$$

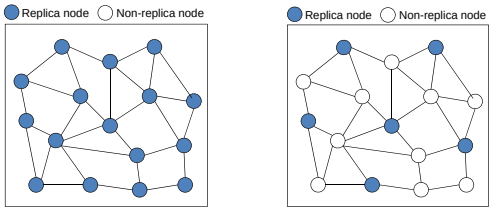
2.5 Back-Tracking Algorithm

A data requester incorporates the ID of its region (i.e. source region) into its request when querying for metadata or data. The required metadata or data will be sent back to the requester based on the RGR protocol. In a highly mobile wireless network, the requester may move out of its region or even travel through a number of regions before the response arrives at the source region. LORD has a back-tracking algorithm to keep track of the requester's movement. In the algorithm, if a requester moves out of its current region before receiving the response, it sends a back-tracking message (including its current region) to the source region. The message is piggybacked on the "hello" messages between neighbor nodes. Thus, each node in the source region keeps a back-tracking message of the requester. Using this message, the response can be forwarded to the requester that moves out of the source region.

For example, when n_i moves from the source region to region R_1 before it receives the response, it sends a back-tracking message to the source region. Then, if n_i moves to region R_2, it sends a back-tracking message to the source region again. When node n_j in the source region receives the response to n_i, n_j forwards it to region R_2. If node n_i doesn't receive a response after a certain period of time, it initiates a new request. Algorithm 1 shows the pseudo-code for metadata request replying.

2.6 Coloring-based Partial Replication

Storing a metafile in every node in a region enables mobility-resilient and fast file retrieval but generates a high overhead for node storage, data mapping updates, and location updates. To handle this problem, we propose a coloring-based partial replication algorithm. The coloring policy in graph theory aims to prohibit two neighboring nodes in a graph from having the same color. Stimulated by this idea, the coloring-based partial replication algorithm aims to guarantee that a node has at least one neighbor holding a metafile while avoiding having the metafile in neighboring nodes. Figure 3 shows the metafile distribution in a region with full replication and with coloring-based partial replication, respectively. In the figure, blue nodes represent replica nodes and white nodes represent non-replica nodes. Compared to full replication, this algorithm reduces the overhead for metafile storage and updates and also



(a) Full replication (b) Coloring-based partial replication

Fig. 3: Metadata replication in a region.

provides high file availability since a non-replica node can easily retrieve a metafile from its neighbors.

In this coloring-based partial replication algorithm, when a node in a region receives the first metadata of the region, it stores the metadata and broadcasts it along with a flip-flop key with an initial value of zero (i.e., K=0) and a TTL (Time to Live). If a node receives metadata with K=0, it changes K to 1, decreases TTL by 1, and further broadcasts the metadata without replicating it. If a node receives metadata with K=1, it replicates the metadata, changes K to 0 and decreases TTL by 1 before broadcasting. A receiver of TTL=0 will not further forward the metadata. We will explain how a node knows its received metadata is the first in its region later on.

Because of high node mobility, the neighboring relationships between the nodes in a region always change. To maintain the coloring status, each node in a region needs to periodically check its neighbors to ensure that it can retrieve the metafile within one hop. Specifically, each node appends a flag bit in the periodic "hello" message to indicate whether it is a replica node. Through the "hello" messages, each node periodically checks whether one of its neighbors is a replica node and records the neighbor's ID. When a non-replica node notices that none of its neighbors has a replica, it sends a metafile request with a TTL to a randomly chosen neighbor. The request is forwarded until meeting a replica node, which sends a metafile to the requester along the original path. If the requester does not receive a response during TTL, it assumes that there is no metafile in its region and its subsequent received metadata is the first metadata in its region. Algorithm 2 shows the pseudocode for the coloring-based partial replication algorithm conducted by each node.

When a node receives a message for storing, deleting, or updating a file's metadata, if it is a replica node, it conducts the operation accordingly and forwards the message to its neighbors. If it is a non-replica node, it directly forwards the message to its neighbors. In this way, all replicas in the region are updated. When a node receives a metadata query, if it is a replica node and is lightly loaded, it responds with the queried metadata. Otherwise, it forwards the query to the neighbor that has the replica.

Replica management in node mobility. In order to ensure that a metafile is always stored in a sufficient number of nodes in a region with the partial replication algorithm, nodes need to transfer replicas when they move in and out of a region. Algorithm 3 shows the pseudocode for replica management in node mobility with the partial replication algorithm.

Algorithm 2 Pseudocode for the coloring-based partial replication algorithm conducted by each node.

```
1: //Ensuring that at least one neighbor has metafile;
2: if it does not have metafile then
3:    for each "hello" message from its neighbors n_i do
4:       if n_i is a replica node then
5:          Record n_i in its replica node list
6:       end if
7:    end for
8:    if none of its neighbors is a replica node then
9:       Randomly select a neighbor n_j
10:      Send a metafile request to n_j with TTL
11:   end if
12: end if
13: //Handling received metafile request;
14: if receive a metafile request from n_j then
15:   if it is not a replica node then
16:      if it has no neighbor which is a replica node then
17:         Forward the request to a randomly selected neighbor
18:      else
19:         Forward the request to its neighbor that is a replica node
20:      end if
21:   else
22:      Send a metafile to n_j
23:   end if
24: end if
25: //Handling received metafile;
26: if receive a metafile then
27:   if it is not the metafile requester then
28:      Send metadata to the previous request sender
29:   end if
30: end if
```

Node join. When a node, say n_i, moves into a region, it checks whether it has neighbors and whether any of its current neighbors has a metafile. If node n_i has neighbors but none of them has a replica, n_i sends a metafile request to a randomly chosen neighbor n_k, which asks for a metafile from its neighbors and forwards the metafile to n_i. If at least one of n_i's neighbors has a replica, n_i can just stay in the region without the need to be a replica node. If n_i does not have any neighbors, it sends out a metafile request targeting region R_i using RGR. If R_i is not vacant, the request message will be forwarded to a node n_j in region R_i. If n_j is a metafile node, it sends a metafile back to n_i. Otherwise, it fetches the metafile from its neighbor and sends it to n_i. If R_i is vacant, the request message will enter a region R_j, which is the proxy region of R_i. The first request receiver in R_j notifies all nodes in R_j to update their metadata $meta_{i,j}$ to $meta_j$. If it is a metafile node, it sends $meta_i$ to n_i. Otherwise, it fetches $meta_i$ from its neighbor and sends it to n_i.

Node departure. When a node moves out of its current region, if it is not a replica node, it does not need to notify any nodes before leaving. Otherwise, it transfers its metafile to a node in the region. If a node has a certain number of neighbors, the node sends its metafile to a randomly chosen neighbor before leaving. If a leaving node n_i does not have any neighbors, n_i sends its metafile transfer request to its region R_i using RGR. RGR forwards the metafile around the region until it meets a node n_j, which responds to n_i. If n_j is a metafile node, n_i drops the metafile and leaves R_i. If n_j is not a metafile node, n_i sends the metafile to n_j, and then drops the metafile and leaves R_i.

The coloring-based partial replication algorithm also

Algorithm 3 Pseudocode for metafile replication in node mobility.

```
1: //When node n_i moves into region R_i;
2: if n_i has neighbors in region R_i then
3:    if no neighbor has a metafile then
4:       Request a metafile from a randomly selected neighbor
5:    end if
6: else
7:    Send a metafile request to region R_i using RGR
8: end if
9: //When node n_i moves out of its region R_i;
10: if n_i is a replica node then
11:    if n_i has neighbors in region R_i then
12:       Send its metafile to a randomly selected neighbor
13:       Delete its metafile and leave
14:    else
15:       Send a metafile transfer request to region R_i
16:    end if
17:    if n_i receives a metafile transfer response from n_j then
18:       if n_j is not a replica node in R_i then
19:          Send metafile to n_j
20:       end if
21:       n_i deletes its metafile and leave
22:    end if
23: end if
```

reduces the overhead due to intra-region mobility. When a node moves into a new region, without the algorithm, it needs to acquire metadata and drops its old metadata. With the algorithm, only when the node is a replica node, it needs to move its metadata to its old neighbor; if the node has a neighbor with metadata in the new region, it does not need to acquire metadata.

3 Performance Evaluation

In order to simulate a highly dense network, we conducted simulation on the ONE event-driven simulator [34]. We compared the performance of LORD with GHT [12] and GLS [20], the most representative geographic routing based (locality-based) data search systems. GHT maps a file to a geographic location, and stores it in the node closest to the location. GHT employs geographic routing for file publishing and querying. Each node holder periodically sends out a mapping update message to check for a new closest node to the file's location and transfers the file when needed. GLS is a node location service for geographic routing. We extended it for data search system. In GLS, the entire geographic area is recursively divided into a hierarchy of increasing smaller squares. The metadata of a node's file is mapped to several squares and stored in nodes in the squares that have the closest IDs to the node virtual ID. A message is routed based on node virtual IDs, and geographic routing is employed in each routing step. If a node's moving distance reaches a pre-defined updating distance threshold, it sends its new location to the new mapped nodes of its metadata . As in [20], we set the updating distance of GLS to 50m in the experiments. As in [12], we set the metadata-location mapping update interval to 2s. To make all methods comparable, we mapped file metadata rather than files to nodes in all methods. We set the location updating interval in GHT to 2s in our experiment.

In order to test the performance of a topological routing based data search system, we include LORD with AODV [35] (denoted by AODV) instead of RGR for data transmission in the comparison set. AODV is an on-demand multi-hop routing protocol that builds a route path from the source to the destination using flooding, and then transmits a message along the path. To measure the effectiveness of the coloring-based partial replication algorithm, we also evaluated LORD with this algorithm (denoted by LORD-P) and LORD with full replication (denoted by LORD-F).

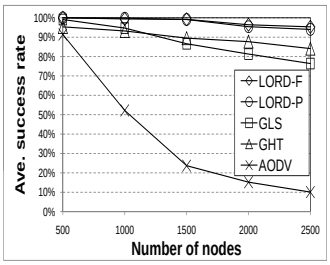
In the simulation, all nodes move within a 2200m*2200m area. We divided the area into 100 regions with a 220m*220m region size. We set the transmission range of each mobile node to 150m. Thus, a node can reach almost half of the nodes in its region. Unless otherwise specified, the number of nodes was set to 1000. In order to reflect a realistic mobile network, we randomly classified the nodes into three groups with moving speeds randomly chosen within [0.5-2.5]m/s, [1-5]m/s and [20-30]m/s to represent the movement of walkers, bikers and cars. The ratio of the number of nodes in the three groups of nodes was set to 4:3:3. The packet transmission speed of nodes was set to 250kbit/s and the size of each file's metadata was set to 2kbits (kb). The buffer size of each node was set to 5Mb. A message arriving at a node with a fully occupied buffer is dropped. All nodes move according to the movement pattern in [36] with 0 pause time. That is, each node randomly selects a region as its destination and moves to the region, and then it randomly selects another region as the destination and moves towards that region. We initially assigned 400 files to randomly chosen nodes. The simulation first warmed up for 100s and then ran for 400s, in which we randomly selected 4 nodes every second to query for randomly selected files. In the experiments, every message was transmitted once without retransmission. The overload threshold in each node was set to 0.8. Each experiment was run for 10 times. We report the results that are within 95% confidence interval.

We used the following metrics in the evaluation:

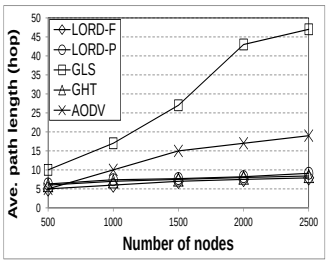
1) *Average success rate*. A node's success rate is the ratio of the number of received files to the number of initiated file queries. This metric represents the performance of successful data querying.
2) *Average path length*. A query's path length is the number of hops for routing the query to the metadata holder. This metric represents routing protocol efficiency.
3) *Overhead*. This is the total number of all traversed hops in metadata responding, mapping updating, and location updating. This metric shows the overhead and reflects the energy-efficiency of a data search system.

3.1 Scalability and Efficiency

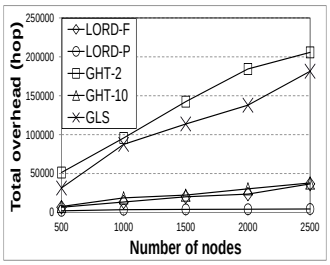
Figure 4(a) shows the average success rate in different data search systems with varied number of nodes in a network. The figure shows that the success rate follows LORD>GHT>GLS$\gg$AODV, which confirms the high success rate of LORD. GHT stores data in the node closest to the data's location, leading to frequent changes of a data's home node in a highly mobile wireless network. If mapping updates are not timely, a file query can arrive



(a) Success rate vs. network size



(b) Path length vs. network size



(c) Overhead vs. network size

Fig. 4: Scalability performance.

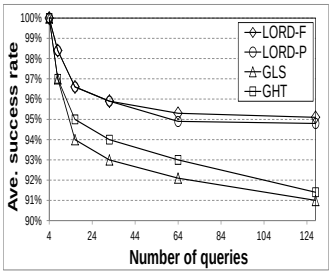


Fig. 5: Congestion resilience.

at the file's new home node before the file is transferred to it. Also, GHT's periodic proactive mapping updates generate many messages and higher channel contention, leading to message drops. In LORD, a node conducts mapping updates only when it changes its region. Its back-tracking algorithm enhances the probability that a response successfully arrives at the requester. Fewer mapping updates, timely mapping updates, fewer location update messages and back-tracking cause LORD to have a higher success rate than GHT. GLS copes with node mobility by requiring a node to update its metadata with its new location in all its home nodes when its movement distance exceeds a threshold. One metadata is mapped to a number of nodes. Also, these update messages travel along the long virtual ID based paths, thus resulting in higher transmission overhead and exacerbating traffic congestion. Therefore, GLS has a lower success rate. We note that the average success rate of AODV drops sharply with the increase of the number of nodes in the system. This is caused by the flooding-based on-demand routing in AODV. In a larger-scale network, it is more likely that a node in the observed route moves away before a message is forwarded to it, resulting in routing failures. Moreover, flooding makes the channel contention more severe, leading to more message drops.

We see that the average success rate of each system decreases as the network size increases. More nodes moving in the network causes more mapping updates. This increases the probability of untimely mapping updates, and hence increases the number of occasions that a node receives a data query it cannot answer. Also, more messages for mapping updates lead to higher channel contention and hence more message drops. We also see that LORD-P generates success rate comparable to LORD-F. This is because the coloring-based replication algorithm ensures that each node can find a metadata item from its neighbors. This result confirms the effectiveness of this algorithm in reducing replicas without compromising data search efficiency.

Figure 4(b) shows the average path length for metadata querying versus the network size in different data search systems. We see that the average path length follows GLS>AODV>GHT≈LORD and that of GLS increases rapidly as the network size grows, which confirms the high data search efficiency of LORD. GLS's routing is based on node virtual IDs. The next hop in the virtual path may not be the geographically closest node, which leads to a longer travel path, especially in a large-scale network. Thus, GLS has low scalability. The geographic routing employed in GHT forwards a message to a node geographically closer to the destination in each step, generating the shortest path. RGR in LORD achieves similar path lengths even without the exact geographic location information. In AODV, a source node broadcasts a packet and the destination detects the shortest-latency path, which is not necessarily the shortest-length path, so AODV produces longer routing path lengths. We also see that LORD-P and LORD-F have approximately the same path lengths, which implies that an occasional extra hop in querying in LORD-P does not greatly affect its overall querying path length. This result again confirms the effectiveness of the coloring-based replication algorithm.

Figure 4(c) shows the total overhead versus the number of nodes in the system. In the figure, GHT-2 and GHT-10 denote GHT with updating interval equals to 2s and 10s, respectively. The result follows GHT-2>GLS≫GHT-10≈LORD-F>LORD-P. Also, as the number of nodes in the system increases, the overheads of GHT-2 and GLS increase rapidly, those of GHT-10 and LORD-F increase slightly, and that of LORD-P stays nearly constant. The results confirm the low overhead and high scalability of LORD and the effectiveness of its coloring-based replication algorithm. The reasons for the different performance between GLS, GHT and LORD are the same as in Figure 4(b). GHT-2 generates significantly higher overhead than GHT-10 because GHT-2 has a higher location update frequency. LORD-P has much lower overhead than LORD-F because it generates fewer data mapping updates. In LORD-F, when a node moves from one region to another region, one metafile in the new region is always transferred to the node. In LORD-P, if one of the node's neighbors has the metafile, no metafile transfer is needed. If the node is a replica node, it also needs to transfer its metafile to its neighbor in the old region. The result shows the effectiveness of the coloring-based replication algorithm in reducing overhead.

In conclusion, the higher success rate, shorter path length and lower overhead of LOAD indicate its high scalability. Also, LORD-P has lower overhead than LORD-F and has comparable success rate and average path length to LORD-F, which confirms the effectiveness of the coloring-based replication algorithm in reducing overhead without compromising LORD's scalability.

3.2 Congestion Resilience

In order to compare the congestion resilience performance of different systems, we randomly selected a number of nodes to send queries on the same file

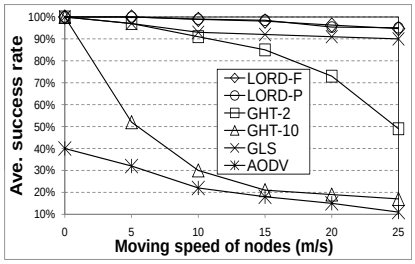
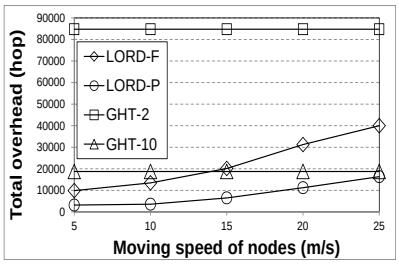
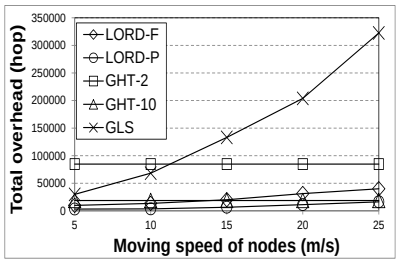


Fig. 6: Success rate vs. mobility rate.



(a) Without GLS



(b) With GLS

Fig. 7: Overhead vs. mobility rate.

simultaneously. We randomly selected 100 files to be queried. Figure 5 shows the average query success rate versus different number of queries sent at one time for a file. We see that as the number of simultaneous queries on a file increases, the average success rate in all systems decreases due to the congestion on the metadata holders. The figure also shows that the decrease rate follows LORD<GHT<GLS≪AODV, which confirms the high congestion resilience of LORD. In LORD, all nodes in a region can provide the queried metadata and only a lightly loaded node responds. Therefore, the nodes are less likely to be congested. In contrast, in GLS, GHT and AODV, as the query can only be resolved by one node in the system, the node is very likely to be congested by many queries. In addition, LORD's back-tracking algorithm increases the probability that a metadata or file response successfully arrives at the requester.

3.3 Mobility Resilience

Figure 6 shows the average success rate versus node moving speed. We see that LORD-F and LORD-P generate the highest success rate and GLS generates higher success rate than others. LORD-F, LORD-P and GLS exhibit slight decreases as node moving speed increases, which demonstrates their high mobility resilience. The success rates of GHT-2 and GHT-10 are much lower and drop sharply when node moving speed increases, and AODV produces the lowest success rate. Mapping updates play an important role in ensuring that metadata is stored in its mapped node in node mobility to guarantee querying success. In both LORD-F and LORD-P, the mapping updates and location updates occur only when a node moves out of a region. Also, LORD's back-tracking algorithm helps forward data to the requester. Further, the redundant replicas in LORD help increase success rate. As a result, LORD-F and LORD-P produce a high success rate. The similar success rates of LORD-F and LORD-P demonstrate the effectiveness of the coloring-based partial replication algorithm in maintaining the success rate while reducing overhead.

GLS's long physical routing path due to virtual ID based routing in mapping updates cannot guarantee timely updates, leading to slightly lower success rate than LORD. In GHT, with increasing node moving speed, the periodic mapping updates cannot guarantee that a file is always stored in the node closest to the file's mapped location, leading to a sharp decrease in the success rate. GHT-2 achieves a higher success rate than GHT-10 due to its higher update rate. In AODV, nodes in the shortest observed path are more likely to be unavailable with faster node mobility. Also, the flooding for AODV path detection and the more frequent mapping updates in faster node mobility exacerbate channel congestion and hence message drops. Consequently, the success rate of AODV is very low and decreases as the node moving speed increases.

Figures 7(a) and 7(b) show the total overhead versus the moving speed of nodes without GLS and with GLS, respectively. The figures show that the overhead of GHT remains constant regardless of the node moving speed. In GHT, each data holder periodically probes whether there is a node located closer to the mapped location of the data. Thus, its location update frequency is fixed and it is not affected by node moving speed. GHT-2 generates higher mapping update overhead than GHT-10 because of its higher update frequency. Recall that GHT also has a low success rate in high node mobility in Figure 6. Hence, GHT is not appropriate in a highly dynamic environment. Figures 7(a) and 7(b) show that the overheads of GLS, LORD-F, and LORD-P increase as the node moving speed increases and GLS increases rapidly. As the node moving speed increases, GLS and LORD produce more location updates and metafile updates. In GLS, a node stores its file's metadata in a number of home nodes and updates the information when it moves a certain distance. Also, the update metadata travels along a virtual path rather than the geographically shortest path. In LORD-F, a node conducts metadata and location updates only when it moves from one region to another, which makes it generate fewer update messages than GLS. LORD-P generates less overhead than LORD-F due to the same reason as in Figure 4(c). These results confirm the high mobility resilience of LORD.

4 Conclusion

The advancements in WSNs and the rapid increase of wireless devices necessitate an efficient data search system for a large-scale, highly mobile and dense wireless network. Current decentralized data search systems either rely on topological routing or geographic routing. The former fails to achieve high scalability due to flooding-based routing or routing table maintenance, while the latter is not resilient to high node mobility and generates high update overhead and energy consumption. In this paper, we propose a LOcality-based distRibuted Data search system (LORD) for large-scale, highly mobile and dense wireless networks.

LORD consists of a region-based data publishing and querying protocol and a region-based geographic routing protocol (RGR). It divides the network area into regions, maps the metadata of similar files to the same region for similarity data retrieval, and stores the

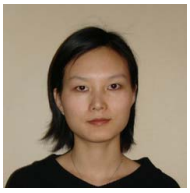
metadata in multiple nodes in the region for mobility-resilience. Unlike the traditional geographic routing, LORD's RGR generates low overhead by forwarding data in the direction of its destination without relying on GPSs. LORD has a coloring-based partial replication algorithm that reduces the number of replicas in a region while maintaining the querying success rate. It further has a parallel file fetching algorithm to minimize the file fetching latency. LORD also works for an unbalanced wireless network with sparse regions. Extensive experimental results show the superiority of LORD over other data search systems in terms of scalability, overhead and mobility resilience. In the future, we will study how to leverage social network properties in LORD and test the system with real human movement trace data.

ACKNOWLEDGEMENTS

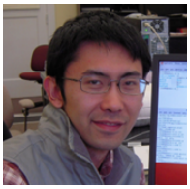
This research was supported in part by U.S. NSF grants CNS-1254006, CNS-1249603, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751. An early version of this work was presented in the Proc. of MASS'09 [30].

REFERENCES

[1] T. Kindberg, M. Chalmers, and E. Paulos. Guest editors' introduction: Urban computing. *Pervasive Computing, IEEE*, 2007.
[2] S. Guo, Y. Gu, B. Jiang, and T. He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. In *Proc. of Mobicom*, pages 133–144, 2009.
[3] H. Yang, F. Ye, and B. Sikdar. A swarm-intelligence-based protocol for data acquisition in networks with mobile sinks. *TMC*, 2008.
[4] K.-W. Fan, S. Liu, and P. Sinha. Dynamic forwarding over tree-on-dag for scalable data aggregation in sensor networks. *TMC*, 2008.
[5] M. Lotfinezhad, B. Liang, and E. S. Sousa. Adaptive cluster-based data collection in sensor networks with direct sink access. *TMC*, 2008.
[6] G. Xing, M. Li, H. Luo, and X. Jia. Dynamic multiresolution data dissemination in wireless sensor networks. *TMC*, 2009.
[7] A. Talari and N. Rahnavard. Cstorage: distributed data storage in wireless sensor networks employing compressive sensing. In *Proc. of GLOBECOM*, 2011.
[8] S. Schnitzer, H. Miranda, and B. Koldehofe. Content routing algorithms to support publish/subscribe in mobile ad hoc networks. In *Proc. of LCN Workshops*, 2012.
[9] C. Hoh and R. Hwang. P2P File Sharing System over MANET based on Swarm Intelligence: A Cross-Layer Design. In *Proc. of WCNC*, pages 2674–2679, 2007.
[10] N. Shah and D. Qian. An efficient unstructured p2p overlay over manet using underlying proactive routing. In *Proc. of MSN*, 2011.
[11] P. Sharma, D. Souza, E. Fiore, J. Gottschalk, and D. Marquis. A case for manet-aware content centric networking of smartphones. In *Proc. of WOWMOM*, 2012.
[12] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Grovindan, L. Yin, and F. Yu. Data-centric storage in sensornet with ght: A geographic hash table. In *Proc. of MONET*, 2003.
[13] C. Tien Ee and S. Ratnasamy. Practical data-centric storage. In *Proc. of NSDI*, 2006.
[14] M. Aly, K. Pruhs, and P. K. Chrysanthis. KDDCS: A load-balanced in-network data-centric storage scheme in sensor network. In *Proc. of CIKM*, 2006.
[15] J. Xu, X. Tang, and W.-C. Lee. A new storage scheme for approximate location queries in object tracking sensor networks. *IEEE TPDS*, 2008.
[16] M. Demirbas, X. Lu, and P. Singla. An in-network querying framework for wireless sensor networks. *TPDS*, 2009.
[17] Y. Zhao, Y. Chen, and S. Ratnasamy. Load balanced and efficient hierarchical data-centric storage in sensor networks. In *Proc. of SECON*, 2008.
[18] P. Desnoyers, D. Ganesan, and P. Shenoy. Tsar: A two tier sensor storage architecture using interval skip graphs. In *Proc. of SenSys*, 2005.
[19] S. M. Das, H. Pucha, and Y. C. Hu. Performance comparison of scalable location services for geographic ad hoc routing. In *Proc. of Infocom*, 2005.
[20] J. Li, J. Decouto, and R. Morris. A scalable location service for geograghic ad-hoc routing. In *Proc. of MobiCom*, 2000.
[21] H. Shen, L. Zhao, and Z. Li. A distributed spatial-temporal similarity data storage scheme in wireless sensor networks. *IEEE Trans. Mob. Comput.*, 10, 2011.
[22] S. Kim. Adaptive ad-hoc network routing scheme by using incentive-based model. *Ad Hoc & Sensor Wireless Networks*, 2012.
[23] H. Frey and I. Stojmenovic. On delivery guarantees and worst case forwarding bounds of elementary face routing components used in ad hoc and sensor networks. *IEEE Transactions on Computers*, 2010.
[24] S. Datta, I. Stojmenovic, and J. Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. *Cluster Computing*, 2002.
[25] B. Karp and H. Kung. Greedy perimeter stateless routing. In *Proc. of MobiCom*, 2000.
[26] Y. Li, Y. Yang, and X. Lu. Rules of designing routing metrics for greedy, face, and combined greedy-face routing. *IEEE TMC*, 2010.
[27] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. of MobiCom*, 2003.
[28] A. Caruso, S. Chessa, S. De, and A. Urpi. GPS free coordinate assignment and routing in wireless sensor networks. In *Proc. of Infocom*, 2005.
[29] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *Proc. of Infocom*, 2003.
[30] Z. Li and H. Shen. A mobility and congestion resilient data management system for mobile distributed networks. In *Proc. of MASS*, 2009.
[31] M. Maroti, P. Volgyesi, S. Dora, B. Kusy, A. Nadas, A. Ledeczi, G. Balogh, and K. Molnar. Radio interferometric geolocation. In *Proc. of SenSys*, 2005.
[32] A. Andoni and P. Indyk. Near-optimal hashing algorithms for near neighbor problem in high dimensions. In *Proc. of FOCS*, 2006.
[33] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices vector spaces, and information retrieval. *SIAM Review*, 1999.
[34] The "ONE" Simulator . http://www.netlab.tkk.fi/tutkimus/dtn/theone/.
[35] C. Perkins, E. Belding-Royer, and S. Das. RFC 3561: Ad hoc on demand distance vector (AODV) routing, 2003.
[36] B. Xie, A. Kumar, and D. Cavalcanti. Mobility and routing management for heterogeneous multi-hop wireless networks. In *Proc. of MASS*, 2005.
[37] K. P. Naveen and A. Kumar. Relay selection for geographical forwarding in sleep-wake cycling wireless sensor networks. *IEEE Trans. Mob. Comput.*, 2013.
[38] Z. Li and H. Shen. A direction based geographic routing scheme for intermittently connected mobile networks. *IJPEDS*, 2013.
[39] M. Li, T. X. Yan, D. Ganesan, E. Lyons, P. Shenoy, A. Venkataramani, and M. Zink. Multi-user data sharing in radar sensor network. In *Proc. of ACM SenSys*, 2007.
[40] Z. Zhang, A. D. Kshemkalyani, and S. M. Shatz. Dynamic multiroot, multiquery processing based on data sharing in sensor networks. *Transactions on Sensor Networks*, 2010.
[41] R. Zeng, Y. Jiang, C. Lin, Y. Fan, and X. Shen. A distributed fault/intrusion-tolerant sensor data storage scheme based on network coding and homomorphic fingerprinting. *IEEE Trans. Parallel Distrib. Syst.*, 2012.
[42] J. Li, C. Blake, D. J. Couto, H. I. Lee, and R. Morris. Capacity of wireless ad hoc networks. In *Proc. of MobiCom*, 2001.
[43] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco. Socially aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE J-SAC*, 2008.
[44] F. Li and J. Wu. MOPS: providing content-based service in disruption-tolerant networks. In *Proc. of ICDCS*, 2009.
[45] W. Lou Z. Jiang, J. Ma and J. Wu. An information model for geographic greedy forwarding in wireless ad-hoc sensor networks. In *Proc. of Infocom*, 2008.
[46] K. Chen, H. Shen, and H. Zhang. Leveraging social networks for p2p content-based file sharing in disconnected manets. *IEEE TMC*, 2012.
[47] G.K.W. Wong and X. Jia. A novel socially-aware opportunistic routing algorithm in mobile social networks. In *Proc. of WCNC*, 2013.



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



Ze Li Ze Li received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2007, and the Ph.D. degree in Computer Engineering from Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks. He is currently a data scientist in the MicroStrategy Incorporation.

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

5 Data Sharing in Unbalanced Networks

Definition 1: A network in which nodes are distributed in balance and every region is a dense region is called a *balanced network*. A network in which the distribution of nodes is unbalanced and some regions are sparse regions is called an *unbalanced network*.

RGR works well in a balanced network, in which a node can always forward a message to the farthest node that is located within the angle range between the source region and the destination region. However, such a next hop may not be found in an unbalanced network. To solve this problem, when a node cannot find a node in the specified angle range, it releases the angle constraint in the routing.

As shown in Figure 8, if a node meets a low-density region (e.g., region 3), it measures its angles ω with its neighbors. Specifically, if the node is on the right side of the landmark, it chooses the neighbor that has the angle ω larger but closest to angle τ; if the node is on the left side of the landmark, it chooses the neighbor that has the angle ω larger but closest to angle β. Thus, the message moves around the region and gradually moves into the angle ranges of LR and RR. In this way, when a sparse region is not vacant (Figure 8(a)), a message is forwarded around the region until arriving at a node in the region; when a sparse region is vacant (Figure 8(b)), a message is forwarded around the vacant region until arriving at region 4. Then, region 4 will store, delete and update metadata for region 3. We define region 4 as region 3's *proxy region*.

Definition 2: The proxy region of region R_i is the region that stores the metadata of R_i when R_i is vacant.

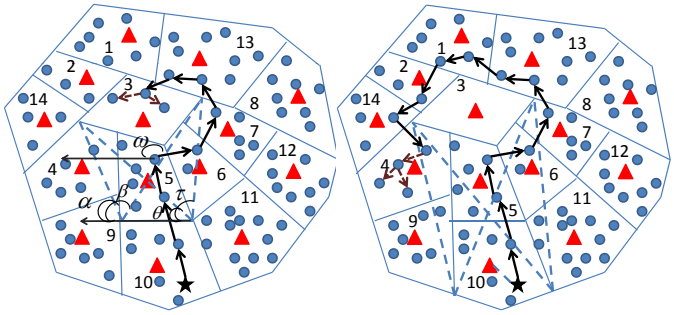
We use a *metadata file* (metafile in short) to represent the collection of the metadata of all files mapped to a region. We use $meta_i$ to denote the metafile stored in region R_i.

Definition 3: A metafile $meta_{i,j} = meta_i + meta_j$ is generated by merging metafile $meta_i$ and metafile $meta_j$; A metafile $meta_i = meta_{i,j} - meta_j$ is generated by separating metafile $meta_j$ from $meta_{i,j}$.

Below, we explain how to conduct mapping updates to maintain data-region mapping in unbalanced networks.

Node join. Recall that node n_i needs to request a metafile from its neighbor when it moves into a new region. If a node does not have a neighbor when it moves into a new region R_i, it sends a request using RGR with R_i as the destination. If R_i was vacant before n_i moves in, the request will arrive at R_i's proxy region R_j. Then, the first request receiver in R_j will notify all nodes in R_j to update their metafile $meta_{i,j}$ to $meta_j$ and sends $meta_i$ to n_i. Thus, the $meta_i$ temporarily stored in the proxy region R_j is moved back to R_i. If R_i was not vacant before n_i moves in, the request receiver sends $meta_i$ to n_i using RGR.

Node departure. When node n_i in region R_i moves out of its region, it sends a message to R_i using RGR if it does not have any neighbors. The message receiver n_j sends a response with its region to n_i. If n_j is in R_i, n_i drops the metafile and leaves R_i. If n_j is in R_j rather than R_i, which means that n_i is the only node in R_i and R_j is R_i's proxy region, n_i forwards the metafile to R_j.



(a) Non-vacant sparse region (b) Vacant sparse region

Fig. 8: RGR routing in sparse networks.

The node in R_j that first receives the metafile broadcasts it as it does for new metadata. Then, $meta_j$ stored in the nodes in R_j is updated to $meta_{i,j}$. That is, R_i's proxy region R_j temporarily manages the metafile for R_i.

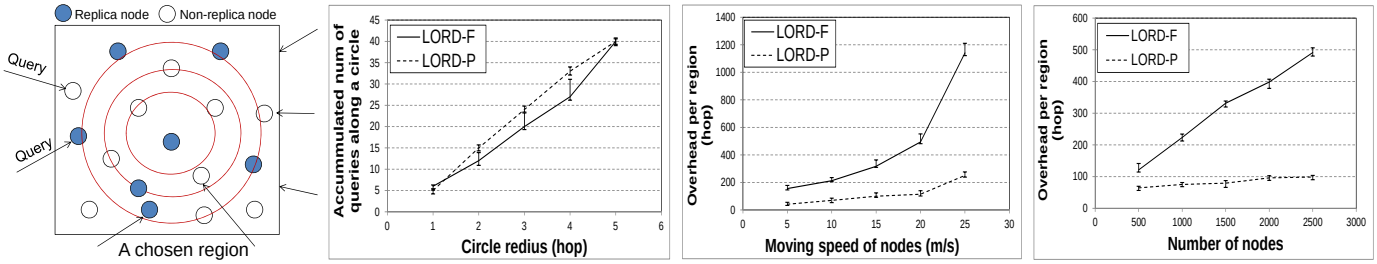
Node failure. A node may fail due to a software or hardware problem. Because mobile nodes exchange beacon messages periodically, the neighbors of failed node n_i can notice that node n_i fails and will avoid forwarding messages to n_i in routing or data retrieval. Because the stored data is replicated in multiple nodes in a region, as long as one node is alive in the region, the data mapped to this region can be successfully retrieved. If the failed node is the only node in a region, then the stored data is lost. However, this case is very rare.

6 Additional Experimental Results

6.1 Comparison of Full Replication and Partial Replication

To test the performance of LORD-F and LORD-P in achieving load balance, as shown in Figure 9(a), we regarded a randomly chosen tested geographic region in the network consisting of a number of concentric circles with different radii, and recorded the total number of queries received by the nodes along each circle. In the experiment, 40 query messages were sent to the chosen region at the same time from other randomly chosen regions. In this experiment, the upper 95%, lower 95% and median of results within the 95% confidence interval are reported. Figure 9(b) shows the accumulated total number of queries received by nodes along different circles versus different radii. The accumulated result increases linearly with the increase of the radius in both LORD-F and LORD-P with a small variance. The results indicate that load distribution is almost balanced in different circles in the region. The figure also shows that the line of LORD-P is above the line of LORD-F, which means that LORD-P has fewer queries along circles with larger radii and more queries along circles with smaller radii. Recall that a node in the destination region may not have the required metadata in LORD-P, and then a node in the circle with a smaller radius is further queried, leading to more queries along the circle with small radius. These experimental results demonstrate the advantage of maintaining redundant metadata replicas. Though it brings about a certain additional overhead, it enables queries to be distributed to different nodes in one region, thus avoiding overloading nodes and enhancing mobility-resilience.

Figure 9(c) shows the overhead incurred per region of LORD-F and LORD-P versus the node moving speed.



(a) Sending queries to the region (b) Load balance (c) Overhead per region vs. mobility rate (d) Overhead per region vs. network size

Fig. 9: Comparison of full replication and partial replication.

It shows that the overhead in LORD-F increases much faster than that in LORD-P and the variance is small. As the node mobility increases, the nodes in the system have a higher frequency of moving in and out of a region, generating higher updating overhead. LORD-F generates more messages than LORD-P for the same reason as in Figure 7(a).

Figure 9(d) shows the overhead incurred per region of LORD-F and LORD-P versus network size. As the network size increases, the overhead per region in LORD-F increases dramatically, while in LORD-P, it increases slightly and the variance is small. As the number of nodes increase, more nodes need to update their locations and conduct mapping updates. Therefore, the overheads in LORD-F and LORD-P grow. As explained for Figure 4(c), in most cases, a node does not need to transfer its metafile to another node when it leaves its region and does not need to get the new metafile when it moves into a new region. More nodes in the system generate higher-density regions, which increases the number of nodes that are not replica nodes and reduces the number of metafile transfers. As a result, as the number of nodes increases, LORD-P only has a slight increase in overhead while LORD-F produces more metafile transfers and hence higher overhead.

6.2 Performance in Unbalanced Networks

In this test, we used the coloring-based partial replication algorithm and randomly chose 10 regions out of the total 100 regions as sparse regions. A node has 99% probability of moving to the 90 high-density regions and 1% probability of moving to the 10 sparse regions. We use balanced-10 and unbalanced-10 to denote a balanced network and an unbalanced network in which the moving speed of nodes was set to 10 m/s. The same applies to other methods.

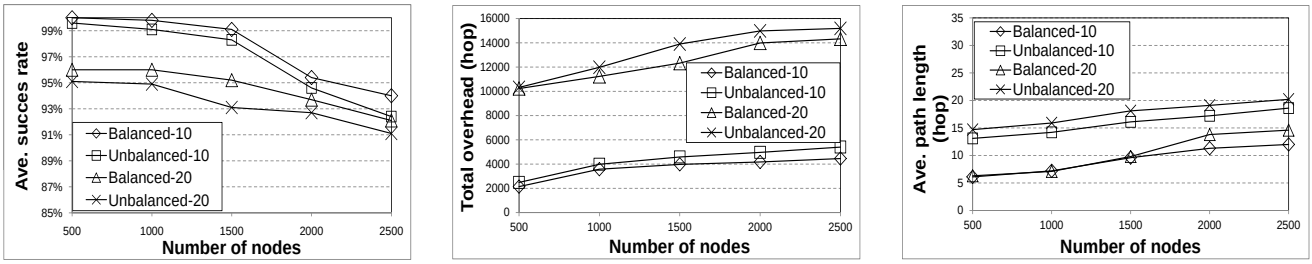
Figure 10(a) shows the average success rates of LORD in the balanced network and the unbalanced network versus the network size. It shows that LORD generates a slightly lower success rate in the unbalanced network than in the balanced network, which indicates that LORD still performs well in sparse networks. In an unbalanced network, if a metadata's mapped region has no node inside, it is stored in a proxy region. RGR can guarantee that a query can always reach the region of the queried metadata. A few nodes in the sparse regions may drop queries when they are overloaded and cannot find a lightly loaded node nearby to transfer the queries. The figure also shows that as the network size increases, the success rates of both the balanced and unbalanced networks decrease. This is mainly due to untimely mapping updates and channel congestion as explained in Figure 4(a). We also see that as the node mobility increases, the query success rates decrease in both balanced and unbalanced networks. High mobility leads to a high mapping update frequency. Therefore, some queries fail to be resolved due to the metadata mapping update delay.

Figure 10(b) shows the total overhead versus the network size. It shows that the overhead of the unbalanced network is slightly higher than that of the balanced network. RGR generates longer path lengths due to sparse regions in responding metadata or transferring metadata in mapping updates. However, only slightly higher overhead indicates that RGR still performs well in sparse networks. We see that as the network size increases, the overhead increases gradually because more nodes generate messages for mapping updates. We also see that higher mobility leads to more messages because higher mobility leads to more location updates and more metafile transfers in mapping updates.

Figure 10(c) shows the average path length versus the number of nodes in the system. It shows that the average path length of the nodes in an unbalanced network is higher than that in a balanced network. This is because in the unbalanced network, a query message must move around the sparse regions, which takes more hops for the message to reach the region of the queried metadata. The figure also shows that as the network size increases, the average path length increases marginally. In RGR, a node forwards a message to the node that is the farthest in an angle range from itself. Since more nodes within a confined angle range generate a more zigzagged path, the path length increases slightly. We also see that as the mobility of the nodes increases, the average path length increases slightly. This is because an identified message receiver may leave its location when the message arrives at the location when node mobility rate is high, leading to a longer path length.

6.3 Resilience of RGR to Measurement Errors

We further validate the performance of RGR with inaccurate measurement of distances and angles. In this test, we set the number of nodes to 1,000 and followed the experiment settings in Figure 4 but purposely added errors to the actual distances and angles. For each measurement, the errors were randomly selected from the range $[-\delta * V, \delta * V]$, in which δ is the inaccuracy rate and V is the actual value of the distance or angle.



(a) Success rate (b) Overhead (c) Path length

Fig. 10: Performance in an unbalanced network.

TABLE 2: Test with moderate inaccurate measurement.

Inaccuracy rate	0	5	10	15	20	25
Success rate	0.986	0.970	0.965	0.963	0.958	0.951
Ave. path length	8.68	9.07	9.43	9.64	9.79	9.95

Table 2 shows the test results when δ was varied from 0 to 25% with 5% increase in each step. We see that as the error increases, the success rate decreases and the average path length increases gradually. This is because when nodes fail to measure the distances and angles accurately, they may fail to choose the optimal forwarders that can lead to the destination most efficiently, thus generating a degraded success rate and an increased path length. However, we see that when δ increases to 25%, the success rate only slightly decreases and still keeps high, while the average path length increases for about 15%. The reason is that packets are still forwarded in the direction of the destination area even with inaccurate measurement. Therefore, even though inaccurate measurement leads to failures of finding the optimal message forwarding paths, most packets can still reach their destinations with slightly longer path lengths.

TABLE 3: Test with severe inaccurate measurement.

Inaccuracy rate	30	40	50	60	70	80	90
Success rate	0.94	0.93	0.91	0.90	0.89	0.87	0.85
Ave. path length	9.98	10.06	10.21	10.26	10.31	10.36	10.45

To further evaluate the performance of RGR with severe measurement inaccuracy, we increased δ from 30% to 90% with 10% increase in each step. The test results are shown in Table 3. We see that even with very high inaccuracy, RGR still achieves relatively high success rate. This is due to the reason that RGR always tries to forward messages in the direction of their destination areas and then the measurement inaccuracy only causes more detours in most cases, thus leading to increased average path lengths, as shown in the table. Above results demonstrate the high resilience of RGR on the inaccurate measurement of distances and angles in routing.

6.4 Performance of the Parallel File Fetching

In this test, we use "ParaFileFetch" to denote the parallel file fetching algorithm, use "SeqFileFetch" to denote the algorithm where a node fetches a file from only one file host, and use "ParaFileFetch-M" to denote the algorithm where an intermediate node merges different file segments of a file when receiving them. We set the size of each file segment to 100 kb and created files with sizes randomly chosen from the range [100kb, 400kb]. Therefore, each file can be divided to 1 to 4 segments. We then created 1500 file requests. The test results are shown in Table 4.

TABLE 4: Performance of different file fetching algorithms.

Method	Success rate	Ave. delay	Ave. path length
SeqFileFetch	0.905	14.92	20.8
ParaFileFetch	0.912	5.27	21.1
ParaFileFetch-M	0.911	5.21	18.1

We see that the parallel file fetching algorithm slightly increases the success rate since it distributes the load more evenly in the network, thereby reducing congestion. We also find that the algorithm greatly reduces the average delay to obtain a file. This is because "ParaFileFetch" and "ParaFileFetch-M" transmit different file segments to the requester simultaneously, while "SeqFileFetch" transmits a whole file sequentially. We further see that "ParaFileFetch-M" has shorter average delay and path length than "ParaFileFetch". "ParaFileFetch-M" merges file segments on intermediate nodes, thus further reducing congestion and the number of forwarding hops needed to retrieve all file segments of a file.

7 Related Work

Data search systems in WSNs. Previous data search systems in WSNs can be classified to categories of flooding [2–7] and geographic routing based method [12–22]. In the flooding method, each node retains the data it senses locally, and a query must be flooded through the network to retrieve data, which generates a high overhead. The geographic routing based method maps a file to a geographical location and stores it in the node closest to the location. The method uses the geographic routing [23–26, 37, 38] to store and query data, which consumes high energy to obtain location information. In a highly mobile network, this method needs to update the file holders of files and transfers data frequently, which generates high overhead and cannot guarantee successful querying. There are other data search works in WSNs focusing on other aspects such as those in [39–41]. Due to space limitation, we only present the works most relevant to our work.

Data search systems in MANETs. Current data search systems in MANETs are either flooding/local-broadcasting based [8] or topological routing based [9–11]. The former relies on flooding or local-broadcasting for data search. However, flooding is not energy-efficient due to a tremendously high volume of transmitted messages, and local broadcasting cannot guarantee data discovery. Also, maintaining many large routing tables in the system consumes extremely large amounts of resources. The work in [42] proved that the topological routing is not applicable in a highly mobile and dense environment. In the topological routing based methods,

nodes advertise their available data, build content tables for received advertisements, and forward data requests to the nodes with high probability of possessing the data. However, they cannot guarantee data discovery because of possible expired routes in the content tables caused by node mobility. Further, advertising generates high overhead. Our proposed LORD system does not rely on exact location information, which saves substantial amounts of energy. It also overcomes the shortcomings of current data search systems in WSNs and MANETs. Datta *et al.* [24] proposed connected dominating set routing, in which a number of special nodes (i.e. dominating nodes) connect to each other for packet transmission. However, it requires GPS and its clusters change as the network topology changes, which increases the cluster maintenance overhead. LORD's regions are geographically fixed and thus do not need maintenance. Also, its RGR routing does not rely on high-overhead GPSs or topological routing.

Recently, social networks are used in routing or content-based service in MANETs [43–47]. Most of the methods utilize node movement pattern by forwarding a message to a node with the highest meeting frequency to the destinations. These methods cannot be simply used for data search because they assume destinations are known. Also, unlike these works, LORD does not focus on a wireless network with an obvious movement pattern in a social network, and it can be applied to general wireless networks. We leave the consideration of social network properties in LORD as future work.