WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. (2008)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.680

Geographic rendezvous-based architectures for emergency data dissemination

Karim Seada[1]*,† and Ahmed Helmy[2]
[1]*Nokia Research Center, Palo Alto, U.S.A.*
[2]*Computer and Information Science and Engineering Department, University of Florida, Gainesville, U.S.A.*

Summary

Communication and information dissemination in times of disasters could have a critical role in aiding the emergency personnel and saving lives. Unfortunately, regular communication becomes more challenging at these times, as the infrastructure might be damaged or overloaded, and the requirements for communication (e.g., response time, capacity) becomes highly critical. Ad-hoc networks forming between wireless devices have always been considered as a promising technology at disaster recovery situations. In this paper, we examine the need for efficient and robust information dissemination mechanisms on top of infrastructure-less ad-hoc networks. We argue that distributed rendezvous-based approaches are more promising for emergency data dissemination than approaches based on network flooding or centralized storage. We present the design of a geographic rendezvous-based architecture and evaluate its performance in detail against other variants and traditional approaches. The study shows its efficiency, high success rate in delivering data, and its robustness to failures and mobility; characteristics highly valuable at emergency situations. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: wireless network architecture; data dissemination; wireless multi-hop networks; mobile ad-hoc networks; location-aware protocols; geographic routing

1. Introduction

Communication and information dissemination in times of disasters are critical and challenging. Several entities such as public authorities, volunteer organizations, and individuals are typically involved and need to organize and exchange information in efficient manners, even when the infrastructure is lacking. The requirements for these responding entities to collaborate are critical as there is a need for saving lives and preserving the infrastructure, in addition to the role they play in passing information to the public and increasing the situation awareness. The information to be managed covers a wide range. For example, timely interactive maps based on the current situation that show rescue workers how to move; these maps are updated based on data coming from both the personnel in the field and the agencies in control. Information about available resources such as medical or fire facilities need to be exchanged between different agencies and the rescue workers at disaster locations. Contextual information gathered by devices carried by the rescue workers could be used by the agencies in evaluating the situation and allocating their resources

*Correspondence to: Karim Seada, Nokia Research Center, 955 Page Mill Road, Palo Alto, CA 94304, U.S.A.
†E-mail: karim.seada@nokia.com

properly. All of these examples and others show the need for a scalable, efficient, and robust architecture to disseminate the information under the unfavorable conditions.

Wireless multi-hop *ad-hoc* networks have always been considered as a potential technology for disaster recovery. These networks form between wireless devices acting as relays and do not depend on the existence of any infrastructure, such as access points or servers, making them a good candidate for instant deployment. There has been a lot of research in *ad-hoc* networks, typically in the problem of routing packets between two mobile nodes. In this paper, we focus on the problem of data dissemination between the various nodes in the network, where the data might need to be delivered to or retrieved from multiple nodes at different times between delivery and retrieval. In an emergency situation, at any moment there will be nodes providing data and nodes seeking data, these nodes might be moving and some of them might fail. These conditions determine several requirements needed by the architecture:

- scalability and efficiency: the architecture needs to be efficient in terms of packet overhead, as the amount of information exchanged could be quite large. It also needs to scale to a large number of nodes covering a wide area.
- robustness: the architecture needs to be robust to node failures and mobility. It should be able to adapt quickly to changes in the network.
- distributed: it should not depend on centralized servers or single-point of failures.
- location-aware: many of the data exchanged make sense based on their location. The architecture should be able to exploit geographic information without requiring very accurate locations to operate.

Based on these requirements we notice that the typical data dissemination approaches such as flooding and centralized storage are not the most appropriate. Flooding is not efficient or scalable as the data or number of nodes increase. Centralized servers could become a bottleneck, besides they are not feasible without an infrastructure to reach these servers. Accordingly, we believe distributed rendezvous-based mechanisms, where different data are stored at different *rendezvous* locations, are suitable for emergency communication and information dissemination. Different rendezvous-based approaches have the commonality that the data should be stored *somewhere* that can be known and reached by both the providers and the seekers of the data. They differ on how the data are stored and accessed. Generally, in infrastructure-based networks (e.g., the Internet), the rendezvous could be a logical address such as a hostname or an IP address. Logical addresses are not a feasible rendezvous in infrastructure-less dynamic wireless networks, either because they do not exist or because their mapping to physical locations will change frequently causing high overhead. Geographic addresses provide a natural rendezvous in wireless networks, and due to the correspondence between the geographic locations of nodes and their network topology, no additional infrastructure is required other than nodes aware of their geographic locations.

Based on the previous arguments, we are in favor of a geographic rendezvous-based architecture and we believe it meets the requirements for efficiency, robustness, and scalability. In this paper, we will show the design of such an architecture, called Rendezvous Regions, which uses geographic regions as the rendezvous for the providers and seekers of information. Another alternative was to use geographic points instead of geographic regions. We chose to use geographic regions, because regions relax the requirements for location accuracy and are more robust to the topology changes caused by dynamics and mobility.

In the rest of this paper we will discuss more about rendezvous-based mechanisms, present the design of this specific architecture, and evaluate its characteristics in detail using extensive simulations and high-level analysis. Section 2 presents various rendezvous-based mechanisms. In Section 3, we explain the context and design of our geographic rendezvous-based architecture. Section 4 contains the detailed evaluation and performance results. Additional design issues are discussed in Section 5. Finally, conclusions are presented in Section 6.

2. Rendezvous-Based Mechanisms

In wireless multi-hop networks, the simplest form of data dissemination and search is global flooding. This scheme does not scale well. Other approaches that address scalability employ hierarchical schemes based on cluster-heads or landmarks [1]. These architectures, however, require complex coordination between nodes, and are susceptible to major re-configuration (e.g., adoption, re-election schemes) due to mobility or failure of the cluster-head or landmark, incurring significant overhead. GLS (Grid Location Service) [2] provides a scalable location service by using a predefined geographic hierarchy and a predefined ordering of node identifiers to map nodes to their locations. GLS is presented for locating nodes and assumes that node identifiers are known. It is not clear that GLS could be extended efficiently to provide a general rendezvous-based mechanism.

Various geographic-based rendezvous mechanisms have been proposed for data-centric storage in sensor networks. GHT (Geographic Hash Table) [3] is a geographic hash table system that hashes keys into geographic *points*, and stores the key-value pair at the sensor node closest to the hash of its key. GHT requires nodes to know their exact geographic location and uses geographic routing to reach the destination. It uses GPSR (Greedy Perimeter Stateless Routing) [4] for geographic routing, where GPSR perimeter routing is used in a novel way to identify a packet home node (the node closest to the geographic destination). Packets enter perimeter mode at the home node (since no neighbor could be closer to the destination), and traverse the perimeter that encloses the destination (home perimeter) before returning back to home node. GHT uses a perimeter refresh protocol to replicate keys at nodes in the home perimeter. The perimeter refresh protocol refreshes keys periodically using also perimeter routing to deal with topology changes after failures or mobility. In Subsection 3.1, we explain geographic routing mechanisms in more detail. ARI (Adaptive Ring-based Index) [5] is another geographic-based rendezvous scheme for data-centric storage in sensor networks. In this scheme data are stored at nodes close to the detecting nodes, and the location information of these storing nodes is pushed to some index nodes. The index nodes for a certain event type form a ring around a rendezvous location for that type. The idea of this scheme is that the nodes in the index ring capture storage and query messages passing through the ring. In order for the index nodes to do that, GAF (Geographical Adaptive Fidelity) [6] is used to divide the network into grids with a single node in each grid responsible for forwarding messages. Since GAF is based on the assumption that each node can only forward messages to the nodes in its neighboring grids, a message sent by a node outside of the ring-encircled region and destined to the index center, must pass some nodes on the ring. There are also other variations of schemes providing data-centric storage in sensor networks. DIFS [7] is a system built on top of GHT to provide range searches for event properties in sensor networks. Another system, DIM [8] allows multidimensional range queries in sensor networks, which is useful in correlating multiple events. DIM uses a geographic embedding of an index data structure (multidimensional search tree) to provide a geographic locality-preserving hash that maps a multi-attribute event to a geographic zone. The sensor field is logically divided into zones such that there is a single node in each zone. DIMENSIONS [9] provides multi-resolution storage in sensor networks by using wavelet summarization and progressive aging of the summaries in order to efficiently utilize the network storage capacity. In References [10,11], geographic curves are used for match-making between producers and consumers of content. The idea is for producers to send their advertisements along four directions (north, south, east, and west) and for consumers to send queries also along these four directions. Nodes where advertisements and queries intersect will reply back to the consumers.

In Reference [12], we examined the design and evaluation of a scalable rendezvous-based architecture for wireless networks, called Rendezvous Regions. This concept was initially presented in Reference [13], in the context of bootstrapping multicast routing in large-scale *ad-hoc* networks, with no protocol details or evaluations. The original Rendezvous Regions idea also borrowed from our earlier work on PIM-SM rendezvous mechanism [14] that uses consistent mapping to locate the rendezvous point. However, a rendezvous *point* is insufficient in a highly dynamic environment as wireless networks. A main goal in our design is to target high mobility environments and this makes Rendezvous Regions more suitable than rendezvous points. It is also based on our objective to design geographic systems that need only approximate location information. The use of regions affects many design details such as the server election, insertion, lookup, and replication. In Section 3, we describe a detailed architecture for data dissemination based on the Rendezvous Regions idea, calling it Rendezvous Regions for Data Dissemination (R2D2). In Section 4, we present extensive performance evaluations with other approaches.

3. Architecture Design

We will start with a brief description of the architecture and a simple scenario. In Rendezvous Regions, the network topology space is divided into rectangular geographical regions, where each region is responsible for a set of keys representing the data or resources of interest (see Figure 1). A key, k_i, is mapped to a

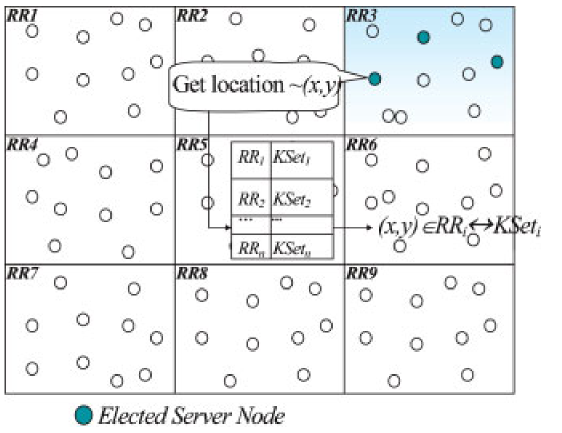

Fig. 1. By knowing their approximate location $\sim(x, y)$, and hence region (RR), nodes in every region (shown in $RR3$) conduct local elections. Only a few nodes (the elected servers) maintain information about the data keys ($KSet_3$) mapping to $RR3$.

region, RR_j, by using a hash-table-like mapping function, $h(k_i) = RR_j$. The mapping is known to all nodes and is used during the insertion and lookup operations. A node wishing to insert or lookup a key obtains the region responsible for that key through the mapping, then uses geographic-aided routing to send a message to the region. Inside a region, a simple local election mechanism dynamically promotes nodes to be servers responsible for maintaining the mapped information. Replication between servers in the region reduces the effects of failures and mobility. By using regions instead of points, our scheme requires only approximate location information and accordingly is more robust to errors and imprecision in location measurement and estimation than schemes depending on exact location information. Regions also provide a dampening factor in reducing the effects of mobility, since no server changes are required as long as servers move inside their region and hence the overhead due to mobility updates is quite manageable.

To give an example, let us assume the following emergency scenario: rescue workers are carrying devices that need to communicate their location and contextual information (e.g., temperature) to the control agencies. The agencies use the location information to determine how the rescue workers are distributed within the disaster field and use the temperatures to determine how the fire is spreading. Based on this information they decide where more help is needed and send requests back to the workers to relocate accordingly. In our architecture, the device will map the information it needs to send, to a specific geographic region. Then it will send a geocast to this region containing the information, which will be stored by a few nodes elected as servers within the region. The control agency, which could also be a mobile personnel in the field, will retrieve this information when needed by sending an anycast request to the same region.

3.1. Context and Assumptions

Our architecture requires nodes to know their approximate locations. Location awareness is valuable for many wireless network applications, so it is expected that wireless nodes will be equipped with localization techniques. Many localization systems exist or have been proposed in the literature: GPS, infrastructure-based localization systems [15,16], and *ad-hoc* localization systems [17,18]. For an extensive survey of localization refer to Hightower *et al*. [19]. In all these localization systems, an estimation error is incurred that depends on the system and the environment in which it is used. In our design we attempt to provide an architecture that requires only approximate location information.

Several geographic routing protocols (e.g., References [4,20–22]) have been proposed for wireless networks that use two modes of operation: greedy mode and perimeter mode. In greedy mode, each node moves the packet closer to the destination at each hop by forwarding to the neighbor closest to the destination. Greedy forwarding fails when reaching a dead-end (local maximum), a node that has no neighbor closer to the destination. Perimeter routing (face routing) is used to route around dead-ends until closer nodes to the destination are found. In perimeter mode, a packet is forwarded using the right-hand rule in a planar embedding of the network graph. Since wireless network connectivity is in general non-planar, each node runs a local planarization algorithm such as GG (Gabriel Graph) or RNG (Relative Neighborhood Graph), to discard a subset of the physical links during perimeter routing and generate a corresponding planar graph.

The Rendezvous Regions scheme can be built on top of any routing protocol that can route packets toward geographic regions. The only requirement of the routing protocol is to maintain approximate geographic information, such that given an insertion or lookup to a certain region, it should be able to obtain enough information to route the packet toward that region. The network space is divided into rectangular equal-sized regions (see Figure 1), where the size of the region is set based on the radio range and how many hops we want the region to cover. The region size we use is covering a few radio hops, to provide adequate relaxation of the inaccuracy and mobility effects, while keeping the region flooding overhead and server load reasonable. In the simulations we study the effect of the region size in more detail. We assume that the geographic space and boundaries of the network are known and that each node has a localization mechanism to detect its approximate geographic location and accordingly its region. Our design also allows us to relax the requirements for exact boundaries by having boundary regions instead of boundary points. Since nodes know the network geographic space boundaries and the region size, they can determine their regions within the space. Under reasonable density, we assume the network is connected and that each region has nodes in it. In case of partitions and empty regions, multiple hash functions can also provide substitute regions. In Section 5, we will discuss empty regions, gaps, and the region size in more detail.

3.2. Rendezvous Regions for Data Dissemination (R2D2)

In this Section we describe the main components of our architecture.

Region Detection: Using a localization mechanism, each node detects its location and accordingly its geographic region. When the node moves, it detects its new location and so it keeps track of its current region. The node uses this information to forward packets toward regions, detect packets forwarded to its region, and potentially participate in server election in its region (if and when needed) (Figure 1).

Server Election: A simple local election mechanism is used inside the region to dynamically promote the servers. The number of servers required in the region is called S. As S increases, the robustness to mobility and failures increases, but also the storage overhead increases. In the simulations we study the effect of S in more detail. Servers are elected on-demand during insertions. When a data insertion operation is issued, the first node in the region that receives the insertion[‡], known as the *flooder*, geocasts the insertion inside the region. Each server receiving the insertion geocast sends an Ack back to the flooder. The flooder keeps track of the servers and if it does not get enough Acks (the minimum number of servers required), it geocasts again and includes a self-election probability, p, in the goecast message. Each node receiving the geocast, elects itself with probability $p \times c$, where c is a factor determined based on certain node capabilities such as stability. If the node becomes a server, it replies to the flooder. If not enough Acks are received, the flooder increases p based on a back-off mechanism until the required number of servers reply or p reaches 1. When servers move out of the region or fail, new servers are elected in the same way. After the new servers are elected, they retrieve the stored keys from other servers.

Insertion: A node inserts a key, K, by first mapping the key to a Rendezvous Region, RR_i, where $K \in KSet_i \leftrightarrow RR_i$. The node generates a packet containing the region identifier, RR_i, in its header. Nodes routing the packet toward the region, check the region identifier to determine whether they are in or out of region. The first node inside RR_i to receive the packet, the *flooder*, geocasts the packet inside the region. Servers inside the region receive the geocast, store the key and data, then send Acks back to the flooder (Figure 2). The flooder collects the Acks and sends an Ack back to original sender. If no Ack is received by the sender, it timeouts and retransmits the insertion up to a fixed number of times.

Lookup: Lookups are similar to insertions except that nodes and previous flooders inside a region cache

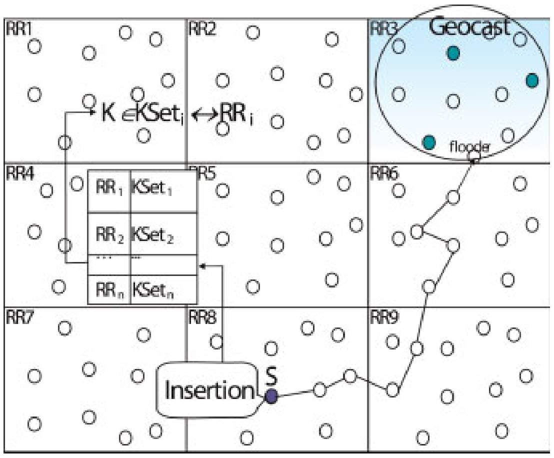

Fig. 2. Insertion (step I): Node S wishing to insert (or store) data key K that belongs to $KSet_i$ gets the corresponding RR (in this case $RR3$) through the mapping ($KSet_i \rightarrow RR_i$). (step II): Node S sends the data toward $RR3$, where it is geocast by the flooder and stored by the servers.

[‡]A node can identify that it is the first node in the region to receive the packet by a simple flag set in the packet header.

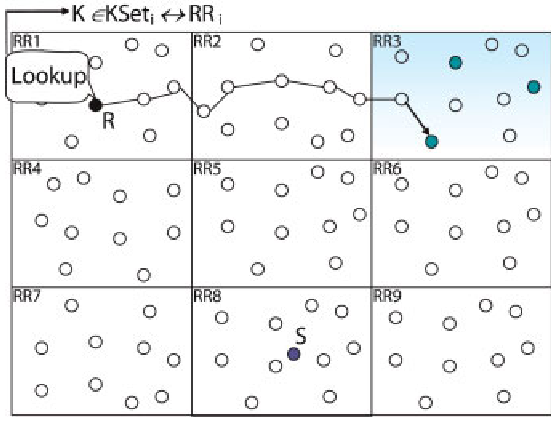

Fig. 3. Lookup: Node *R* looking for data with key *K* that belongs to $KSet_i$ gets the corresponding *RR* (in this case *RR3*) through the mapping ($KSet_i \rightarrow RR_i$). Then sends the data lookup toward *RR3*, where it is anycast to any server holding the information.

locations of the recent servers they hear from, and send the lookups directly to any of the servers (anycast). The server replies to the flooder and the flooder replies back to original sender (Figure 3). If the flooder receives no reply or if it has no cached servers, it geocasts the lookup inside the region.

Replication: Replication is inherent in this architecture, since several servers inside the region store the key and data. This adds extra robustness to failures and mobility. For additional robustness against severe dynamics such as group failures and partitions, multiple hash functions may be used to hash a key to multiple regions.

Mobility: Local movements of nodes and servers have negligible effect and overhead on our architecture as long as servers stay within their regions. The major condition we need to consider is when a server moves out of its region. The server checks its location periodically to detect when it gets out of its region, in order to send an insertion packet containing its stored information toward that region so that new servers are elected. The server then deletes its stored keys and is not a server anymore. It may or may not get elected again later in a new region.

Failures: Since each region contains several servers, and insertions and mobility may invoke new server elections, it is unlikely that independent reasonable failures will cause all servers to vanish. In order to avoid this case, servers use a low-frequency periodic soft-state mechanism during silent (low traffic) periods, to detect failing servers and promote new servers. Each server runs a low-frequency timer, which is reset each time an insertion geocast is received. When the server times out, it geocasts a packet checking for other servers. Other servers reset their timers upon receiving this check and reply back demonstrating their existence. If not enough servers reply back, server election is triggered.

Bootstrap: The last point to consider is how the mapping function itself is obtained. Using the same rendezvous mechanism, we can provide a bootstrap overlay to publish dynamic mappings. Mapping to a *well-known key*, a node sends request to a well-known region to obtain the mapping function of a new type of resource (Figure 4). These mappings however are not expected to change frequently and introduce more flexibility for providing different mappings for different type of resources and changing them when needed. It also allows for incremental expansion of the resources provided.

4. Performance Evaluation

In this section, we evaluate our architecture using detailed NS-2 [23] simulations with detailed models of the wireless MAC and physical layers. We run extensive simulations to investigate the design space, and study the architecture in various environments including node mobility and failures. We also study the effect of inaccurate locations. R2D2 is running as an overlay over the routing layer and we are using GPSR [4] as the routing protocol. We modified GPSR to route to regions instead of specific destinations by

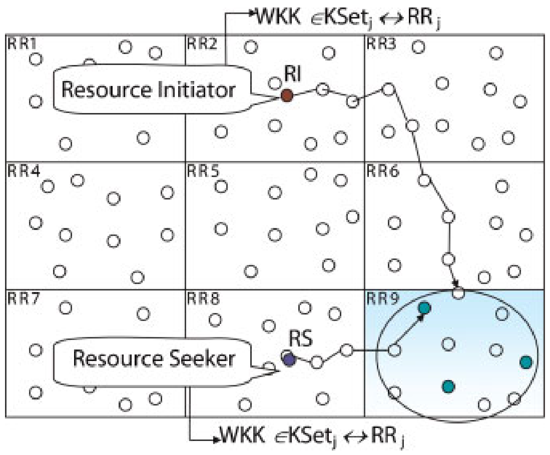

Fig. 4. Using the same mechanism for obtaining the mapping function. A bootstrap mechanism for initiating (and obtaining) new resource mappings using *well-known key*(*s*) (*WKK*).

forwarding the packet toward the center of the region and using geocast or anycast inside the region. We verify the correct operation of R2D2 and evaluate its performance under different scenarios. In addition, we run similar detailed experiments for a rendezvous-based mechanism based on geographic points, GHT [3], in order to compare the geographic region to the geographic point approach. We conducted simulations with up to 400 nodes using the detailed model. To study the scalability of the architecture to higher number of nodes, we perform also higher-level simulations (without the MAC and physical layers) for networks up to 100 000 nodes and compare to GHT, flooding, and centralized storage.

There is a wide range of parameters we consider during the evaluation. The environment parameters include the network dimensions, number of nodes (density), transmission range, the expected number of lookups per second, and the lookup-to-insertion ratio (LIR). The design parameters that we study are mainly the number of regions and the number of servers per region. The performance metrics are the success rate of lookups, message overhead per insertion, message overhead per lookup, and total storage/insertion, in addition to the maximum node overhead of these metrics. We also study the overhead due to mobility and failures.

4.1. Detailed Simulation Results

We implemented R2D2 in NS-2 as an overlay that runs over any wireless routing protocol that can route toward geographic regions. Currently, we are using GPSR, modified to route to a region, with detailed 802.11 MAC and physical layers. The GPSR beacon interval is 1 s and the beacon expiration is 4.5 s. The density is fixed to 1/1024 m^2 and the number of nodes is 100, 200, or 400. We explored several transmission ranges between 60 and 120 m. For convenience (and space limits), in the results shown, we focus on 100–200 nodes topologies and 80 m transmission range. The rate of lookups is 2 per second and the number of retransmissions for both insertions and lookups is 3. The periodic failure check interval is 20 s. Keys are uniformly distributed at random over the space. The results are the average of five random runs over five different random topologies. GHT was already implemented in NS-2. The refresh interval for GHT is set to 10 s. A common problem in GHT that affects its performance, is when a key hashes to a point outside the external perimeter. In this case, perimeter routing may move around the whole external perimeter during insertions, lookups, or refreshes. To reduce the effect of this problem during evaluation, we modified GHT to avoid mapping the keys to points close to the space boundary. We do that by excluding 10 per cent of each side (left, right, top, bottom) of the space during hashing. We will refer to the modified GHT by GHT*.

4.1.1. *Number of servers in region*

First, we study the effect of the number of servers in a region. In the simulations, a value is set for the minimum number of servers, below which flooders ask for new servers. Since the server self-election is probabilistic, the actual number of servers in a region could exceed this value. The experiment is run over a 100 node static topology with four regions and a LIR equal to 10. The number of insertions varies between 10, 30, and 50. The results obtained are the message overhead/insertion, message overhead/lookup, and storage overhead/insertion with different number of servers. From these results we computed the normalized message overhead as $\text{Norm} = \text{Ins} + \text{Lookup} \times \text{LIR}$, where Ins and Lookup are the overhead/insertion and overhead/lookup, respectively. Since, the optimum number of servers depends on LIR, we show in Figure 5, the normalized overhead for different values of LIR. With low LIR, low number of servers causes lower overhead, while high LIR favors more servers. As we notice, 3–4 servers give a good compromise. Another tradeoff, the total storage overhead per insertion increases with the number of servers, but also more servers lead to higher robustness in case of mobility or failures.

In the coming experiments, the number of servers is set to 3. The simulation run is 200 s with 30 insertions at the beginning and 300 lookups at a rate of 2 new lookups per second. Nodes are chosen at random for insertions and lookups. The number of regions varies

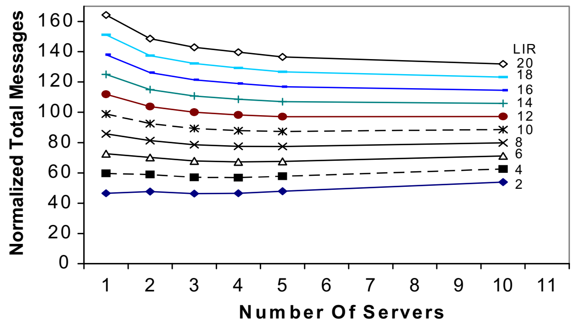

Fig. 5. Normalized overhead for different number of servers.

from 4, 9, 16 to 25. We will start with static networks, then look at failures and mobility.

4.1.2. Number of regions

The number of regions in the space is an important design parameter that indicates the region size and the average number of nodes in region. Ideally, we like to have regions with small size to reduce the geocast overhead and at the same time to have the region size large enough to relax the geographic accuracy and reduce the effects of mobility. In static networks, the success rate for both R2D2 and GHT is almost 100 per cent, so we consider only the overheads in this section. Figure 6 shows the insertion overhead of R2D2 compared to GHT and GHT*. R2D2 overhead decreases by increasing the number of regions (i.e., reducing the region size, since the network size is fixed), due to the reduction in the region geocast overhead. At low number of regions, GHT has lower insertion overhead, but they become closer as we increase the number of regions.[§] In Figure 7, the region size has less effect on R2D2 lookup overhead, since the main factor here in reducing the overhead is caching the servers' locations and anycasting them. Both GHT and GHT* have a significantly higher lookup overhead than R2D2. GHT* has a lower overhead than GHT, since it excludes hashing points close to the network boundary, which can cause the external perimeter traversal, but it is still higher than R2D2. Insertion and lookup overhead in GHT are similar, since they are using the same mechanism.

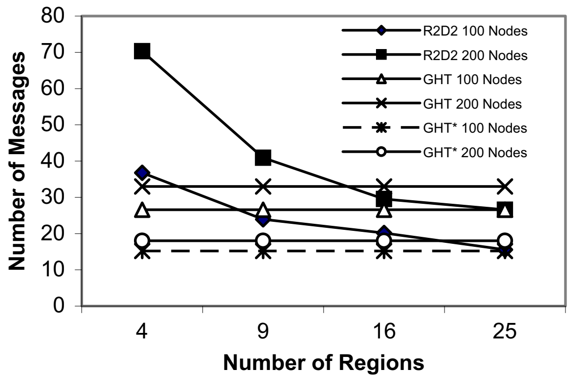

Fig. 6. Insertion overhead for R2D2 *versus* GHT & GHT*.

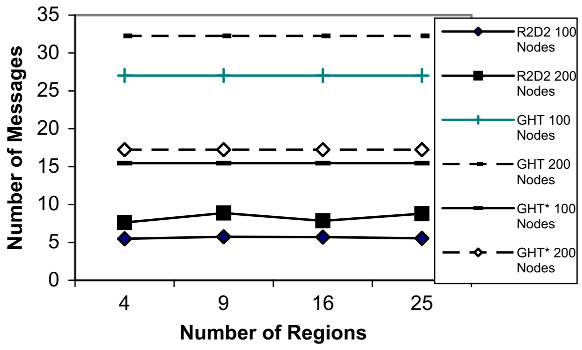

Fig. 7. Lookup overhead for R2D2 *versus* GHT & GHT*.

The high overhead of GHT is due to the home perimeter routing. Each packet in GHT (insertion, lookup, or refreshment) goes to the home node (closest node to destination point), then it traverses the entire perimeter (home perimeter) that encloses the destination, before returning to the home node. The average perimeter length is somewhat long, which can be noticed also from Figure 8, where the average storage overhead for 100 nodes is around 8 in GHT* (18 in GHT), which means that on average 8 nodes are in the home perimeter and store the packet. This is consistent with the insertion and lookup overheads, which are around 15 messages (including the forwards to home node and back replies). By looking at the randomly generated topology in Figure 9, we can understand why average perimeters are so long: (1) even with avoiding the surrounding boundary points in GHT* and using only the internal 64 per cent (.8*.8) of the space for hashing, there are still many areas where a hash point there could cause external perimeter traversal, (2) even with random uniform

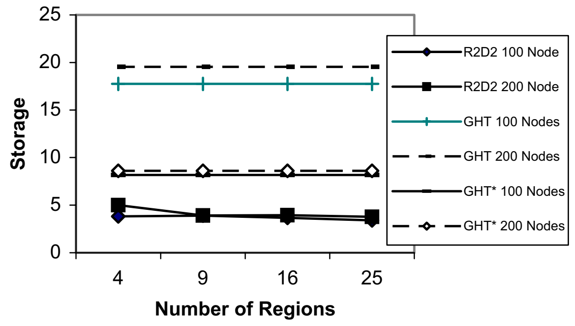

Fig. 8. Storage overhead (per insertion) for R2D2 *versus* GHT & GHT*.

[§]Notice that we are using simple direct flooding, where each node inside the region broadcasts the packet once. Alternatively we can use smart flooding techniques [24] which reduce the geocast overhead.

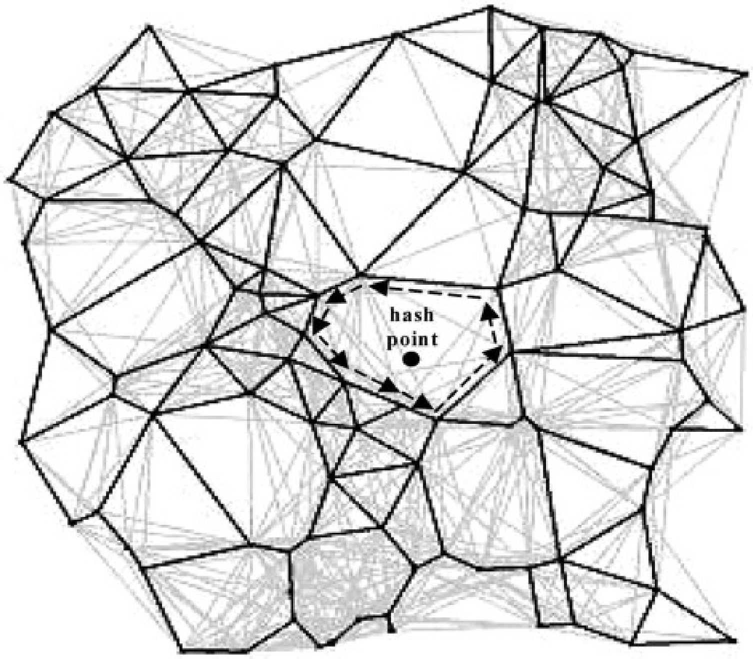

Fig. 9. Example of a random uniform topology where perimeter traversing (in a GG planarized graph) takes many hops. Gray edges are the physical links not included in the planar graph.

distributions there are still somewhat large internal areas without nodes. The keys that hash in these areas have long home perimeters, (3) this may be the most interesting, which is the effect of planarization. Planarization (in this case GG) excludes many links from perimeter routing, such that even in higher dense networks, packets still have to traverse long perimeters because shortcuts between the perimeter nodes are not included in the planar graph.[¶]

Figure 10 shows the periodic failure check overhead in R2D2 compared to the refresh overhead in GHT. In R2D2 it is close to the number of nodes, since in each region, where there are keys stored, one of the servers geocast and so there is only a single geocast in each region independent of the number of keys stored there. In GHT the periodic overhead is much higher, because the refresh is sent for every key around its home perimeter (every key has a different hash point), so the overhead increases with the number of keys stored. For example, in this case we have 30 keys with an average perimeter of 8 nodes in GHT* (18 in GHT) for 100 node topologies, which gives around 240 messages as shown in the figure (GHT* around 250 and GHT around 500).

These results show that the total overhead depends on the data model and the LIR. In order to evaluate the total overhead taking into account the insertions, lookup, and periodic overhead, we compute the normalized total overhead as

$$\text{Norm} = \frac{\text{Ins}}{\text{LIR}} \times \text{iRate} + \text{Lookup} \times \text{lRate} + \frac{\text{Per}}{\text{len}}$$

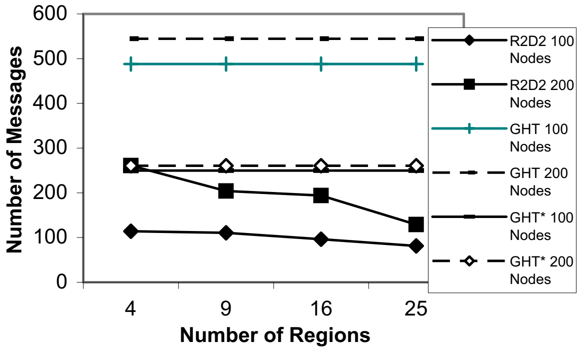

Fig. 10. Periodic (refresh, failure check) overhead for R2D2 *versus* GHT & GHT*.

[¶]We are using a density of 1/1024 m^2 with an 80 m transmission range; in Reference [3] the density was 1/256 m^2 with a 40 m transmission range, which gives the same connectivity. In order to verify that, we run the same experiment with the higher density and lower transmission range of Reference [3] and we got the same results.

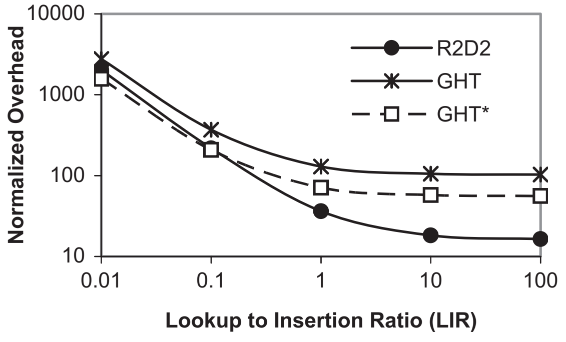

Fig. 11. Normalized total overhead per second for different LIR.

where Norm is the normalized total overhead per second, Ins the overhead/insertion, Lookup the overhead/lookup, iRate and lRate are the insertion and lookup rates (LR), respectively, Per is the periodic overhead in an interval, and len the length of the interval. Plotting this equation at different LIRs, we show in Figure 11 the point where R2D2 and GHT* overhead intersect. This is beneficial in deriving which system is more efficient in a certain situation. The overhead ratio between R2D2 and both GHT and GHT* is shown in Figure 12, for different LR. For LIR > 0.1, R2D2 incurs less overhead than GHT*, with the savings approaching 80 per cent for LIR > 10.

4.1.3. Failures

In this section, we show the robustness of R2D2 with node failures. Robustness is achieved by the replication among servers and the periodic failure checking mechanism. We run simulations of 100 nodes where 10, 30, or 50 per cent of the nodes fails at random times. In Figure 13, the success rate remains high with failures in both GHT and R2D2 (around 97 per cent when 50 per cent of the nodes fail). With 25 regions and 100 nodes in R2D2, the success rate falls down because on average we have only 4 nodes per region, and less than that with failures. By setting the region size appropriately we can avoid this problem. Figure 14 shows the periodic overhead with failures; R2D2 overhead increases slightly but remains around the number of nodes, while GHT overhead increases significantly due to the extra perimeter traversals for key refreshments after node failures.

4.1.4. Mobility

R2D2 is designed to be robust to node mobility. The main reason is that local movements as long as the servers remain in the region, do not require change of servers or any extra overhead. Mobility updates happen only when a server moves out of region. In this experiment, nodes are moving using the random waypoint model [25], with a maximum rate of 1, 2, or 5 m/s. These speeds are selected as representative of movements in an emergency scenario, where rescue

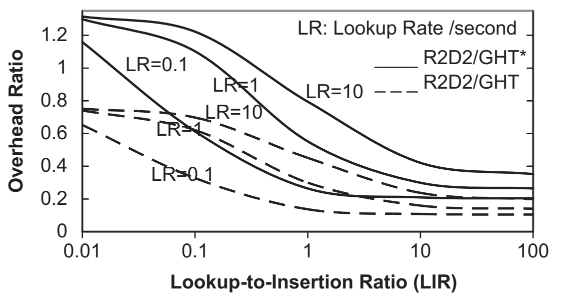

Fig. 12. Overhead ratio for different LIR and different lookup rates (LR).

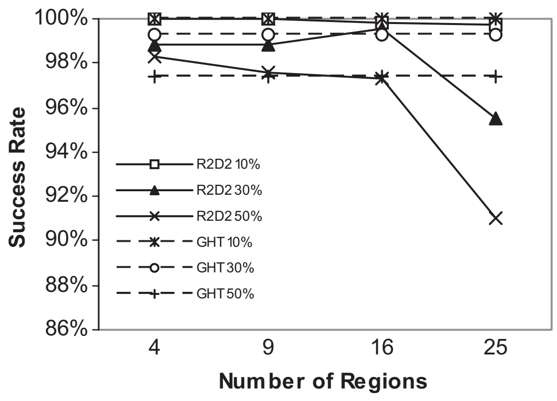

Fig. 13. Lookup success rate for different node failure rates.

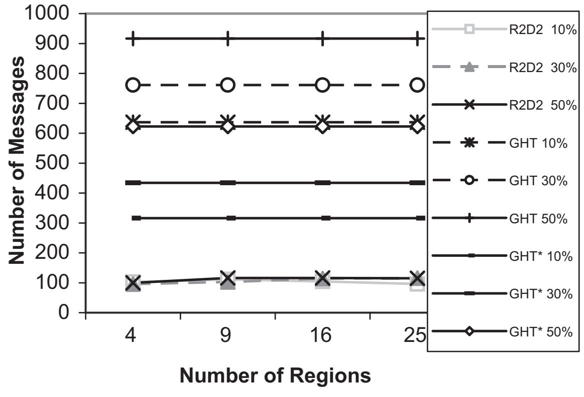

Fig. 14. Periodic overhead for different node failure rates.

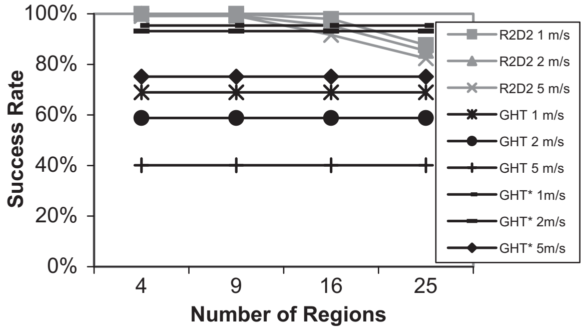

Fig. 15. Lookup success rate for different node mobility rates.

personnel are moving typically faster than regular walking or using small rescue vehicles at low vehicular speeds to reach the disaster locations. There is no pause time, all nodes are moving continuously. Figure 15 shows the high success rate of R2D2 under mobility. It drops only with higher number of regions, because of the low number of nodes per region, which is also the reason for the insertion and lookup overhead increase in Figures 16 and 17. This problem does not exist with higher number of nodes and can be avoided by setting an appropriate region size. For example, with 100 nodes moving at 5 m/s, nine regions give a success rate above 99 per cent. GHT success rate drops faster with mobility, since any small movements can cause changes in the key storage causing lookup packets to reach home nodes (nodes closest to destination point) that do not have the key due to changes in topology. In addition, perimeter traversal with mobility is susceptible to loops, which may cause the packet to exhaust its TTL. We used a 2 s replanarization timer with GHT to reduce this effect.‖ In Figure 18, we show the mobility update overhead during an interval equivalent to GHT refreshment interval, since both of them reflect the overhead due to mobility (in addition, GHT performs refreshes when new node joins or leaves are discovered). R2D2 has much lower overhead, since only the servers send updates when they move out of region. In Figures 19 and 20, we fix the number of regions of R2D2 to 9 and change the pause time of nodes moving with a maximum random-waypoint velocity of 5 m/s. We notice the high success rate and the low overhead compared to GHT and GHT*.

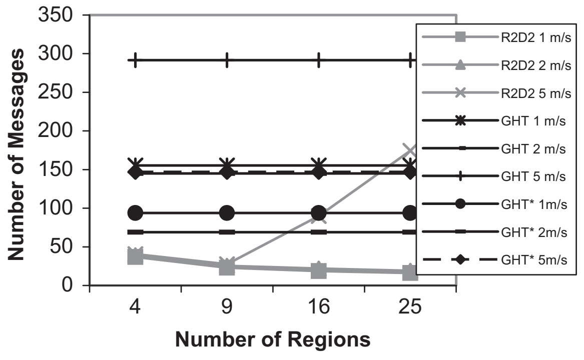

Fig. 16. Insertion overhead for different node mobility rates.

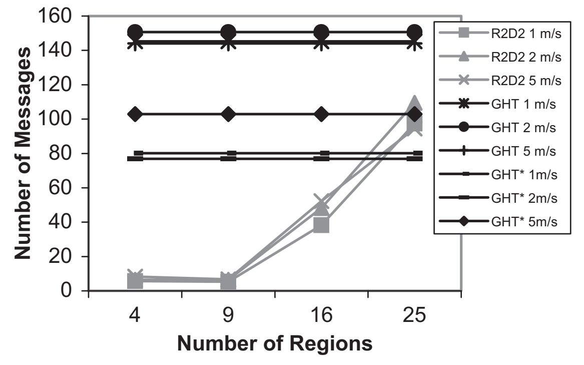

Fig. 17. Lookup overhead for different node mobility rates.

4.2. Location Inaccuracy

In this section, we study the effect of inaccurate location information on R2D2. One of the main objectives of R2D2 is to relax the accuracy required by nodes in estimating their location. In this section we consider only the routing behavior in an ideal

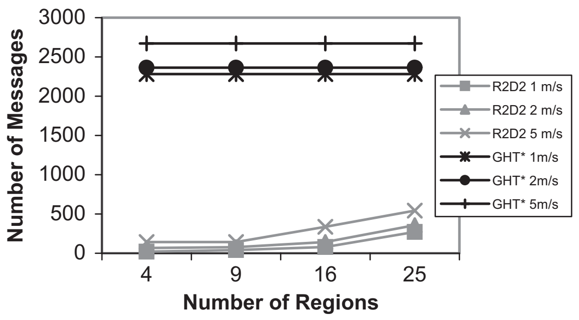

Fig. 18. Mobility update (refresh) overhead in R2D2 and GHT.

‖In Reference [3], the TTL was also limited during refreshments to reduce the overhead if looping happens, but we have not included this change here, since it is not clear how the TTL can be set dynamically without extra overhead and what effect it has on other functions of the protocol.

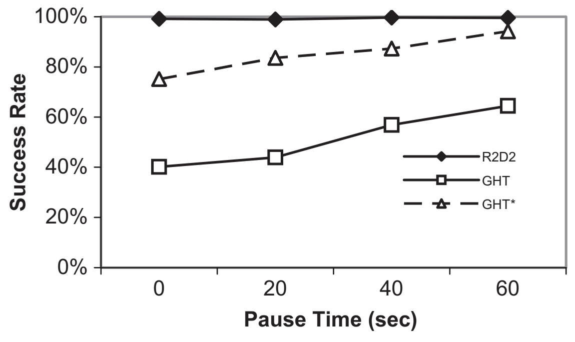

Fig. 19. Lookup success rate for different node pause times.

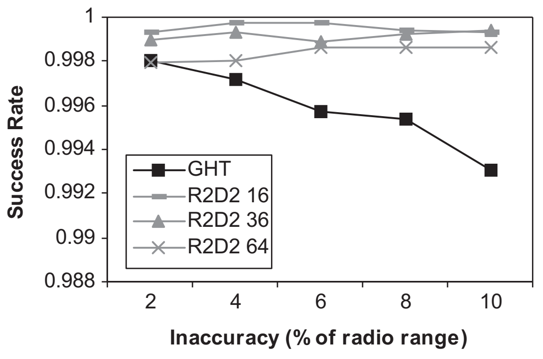

Fig. 21. Success rate at low inaccuracy range (2–10 per cent of radio range).

wireless environment, in order to evaluate the effects of location inaccuracy without the interference from other layers such as MAC collisions or physical layer effects. We use a static and stable network of 1000 nodes having the same radio range and density as in the previous section. Results are computed as the average of 1000 runs, where in each simulation run, nodes are placed at random locations in the topology and 10 insertions and 100 lookups are generated by random nodes. The success rate is the percentage of successful lookups. The maximum localization error is presented as a fraction of the radio range. The estimated node location is picked uniformly from a random location around the node accurate position limited by the maximum error.

Figure 21 shows the success rate of R2D2 with different number of regions (16, 36, and 64 regions) compared to GHT at low inaccuracy range (2–10 per-cent of the radio range). R2D2 is more robust to location inaccuracy than GHT and the effect of inaccuracy is less on larger regions. Figure 22 has a higher inaccuracy range (20–100 per cent), which shows the effects of inaccuracy more clearly. For example, at an inaccuracy equal to 60 per cent of the

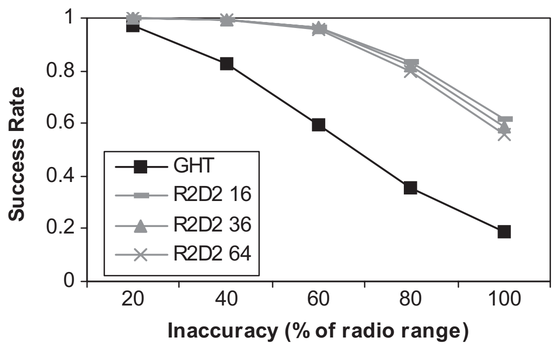

Fig. 22. Success rate at high inaccuracy range 20–100 per cent of radio range).

radio range, GHT success rate goes below 60 per cent, while R2D2 is still above 95 per cent.

By adding the fix we introduced in Reference [26] to GHT[⊥] and the geographic routing in R2D2, the success rate improves significantly. At low inaccuracy range, both R2D2 and GHT have a success rate higher than 99.8 per cent, with larger regions still having better rate, but the differences are small. Figure 23 shows the success rate at high inaccuracy range. With an inaccuracy up to 80 per cent of the radio range, both R2D2 and GHT have above 95 per cent success rate. At an inaccuracy up to the whole radio range, R2D2 is

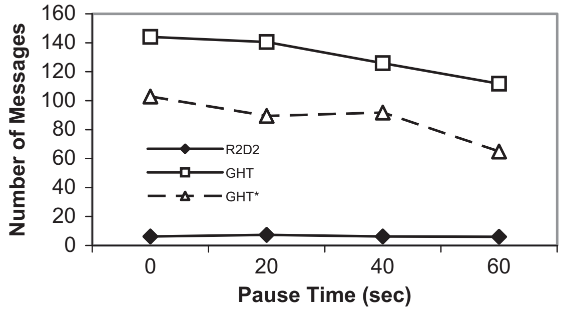

Fig. 20. Lookup overhead for different node pause times.

[⊥]This fix solves the disconnection problem in planarization, which we found to be the major problem that results from location inaccuracy and causes failures during face routing. In this fix, a node does not remove an edge to another node in the planar graph, unless the other node is connected to the same witness [26].

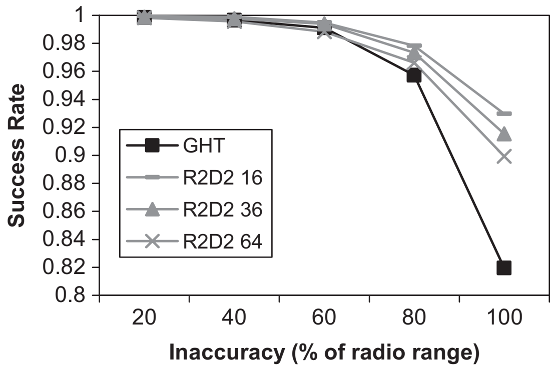

Fig. 23. Success rate at high inaccuracy range (20–100 per cent of radio range) with the fix added.

above 90 per cent and GHT above 80 per cent, which is a significant improvement.

4.3. High-Level Simulation & Analysis

To evaluate the scalability of R2D2 to higher number of nodes, we perform high-level simulations without the wireless MAC and physical details. We compute the total message overhead and the hotspot message overhead (maximum node overhead) in a static network. The overhead reflects also the energy consumption of a wireless device and the lifetime of the whole network. Without errors or dynamics, such as failures and mobility, the success rate need not be computed (always 100 per cent). We compare R2D2 to GHT, flooding, and centralized storage. We count only the insertion and lookup overhead. The periodic refreshment overhead in GHT and the failure check overhead in R2D2 are not included, but we can easily compute an estimate for them based on the detailed simulations results. In GHT, we do not use hash points that lead to long perimeter traversal (more than square root the number of nodes), in order to avoid the high overhead of the external perimeter traversal.

We will first do some approximate analysis for the communication overhead of the mechanisms. We will consider general insertion and lookup operations, where n is the number of nodes, I the number of insertions, L the number of lookups, R the number of regions in R2D2, and S the average number of servers per region. We will use the asymptotic expression of $O(n)$ for flooding the whole network, $O(\sqrt{n})$ for point-to-point routing, and $O(n/R)$ for region geocasts. In flooding, we assume that the nodes store (insert) their keys locally and other nodes flood (lookup) to get them. In centralized storage, we assume a centralized node, storing all the keys, so that all insertions and lookups are forwarded to it. The following table shows the asymptotic insertion and lookup message overhead:

	Total message overhead	Hotspot message overhead
Flooding	$O(n) \times L$	$O(L)$
Centralized	$I \times O(\sqrt{n}) + L \times O(\sqrt{n})$	$O(L + I)$
GHT	$I \times O(\sqrt{n}) + L \times O(\sqrt{n})$	$O(\frac{L}{I})$, $\{\text{for } L > I\}$
R2D2	$I \times O\left(\sqrt{n} + \frac{n}{R}\right) + L \times O(\sqrt{n})$	$O\left(\frac{I}{R} + \frac{L}{\min(I,R) \times S}\right)$

The total message overhead in R2D2 includes the insertion overhead, where an insertion is composed of a point-to-point route to reach the region and a geocast inside the region. The lookup is anycast to a cached server, so it can be considered as a point-to-point route. In the hotspot message overhead of GHT, we assume lookups are uniformly distributed over keys so that for each key inserted, its home node will have $O(L/I)$ lookup. In R2D2, we assume also that keys are uniformly distributed over regions and lookups are uniformly distributed over keys, so that the overhead of a server is the insertions of keys in its region and the lookups (anycasted) are distributed between the S servers in the region. The term min(I, R) takes care of the case when the number of insertions is less than the number of regions, so that lookups are distributed only over those regions that have insertions. In GHT, the refresh overhead per interval is equal to 'the number of keys stored (I) $\times$ average perimeter length.' In R2D2, failure check overhead per interval is at most '$n/R \times$ minimum(I, R),' since we do not need to geocast in regions that have no keys. At a constant node density, increasing the number of nodes will increase the network size, and by increasing the number of regions in R2D2 with the network size and keeping the region size fixed, n/R remains constant. In other way, the average number of nodes in the region is constant, since the density and region size are constant. In this case, the asymptotic overhead at large number of nodes for R2D2 will be similar to centralized storage and GHT. We can also see that in the simulations, where we increase the number of nodes from 100 to 100 000 with different LIRs. The number of insertions is 10. The density is similar to the detailed simulations and the region size in R2D2 is set to have an average of 100 nodes. The results are the average of 10 random simulations with 10 random topologies. In Figure 24, we see flooding has the highest message overhead since each lookup is flooded.

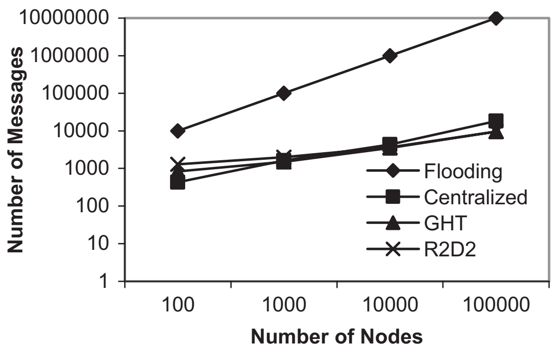

Fig. 24. Total message overhead with increasing number of nodes LIR = 10.

Centralized, GHT, and R2D2 have close total overhead. Figure 25 shows the hotspot message overhead, which is computed as the maximum message overhead at a single node. Centralized and flooding have a high hotspot overhead compared to GHT and R2D2. R2D2 has lower hotspot overhead than GHT in these scenarios, because the lookups for a key in R2D2 are distributed over multiple servers in the region, while in GHT the same home node get all lookups for a certain key. As we also notice from the table, the hotspot overhead of R2D2 is low when *I* is small compared to *R*. As *I* increases above *R*, the hotspot overhead will increase. For GHT lower LIR is preferred.

In Figures 26 and 27, we perform evaluations for low LIR with the following parameters: 100 event types, 10 detected events per type, and 50 queries. This gives 0.05 LIR. Events are detected at random nodes, while all queries are from a single node representing the network gateway. We assume aggregation is performed at all of them, such that a lookup query returns results as a single reply. We do not consider the structured replication version of GHT.

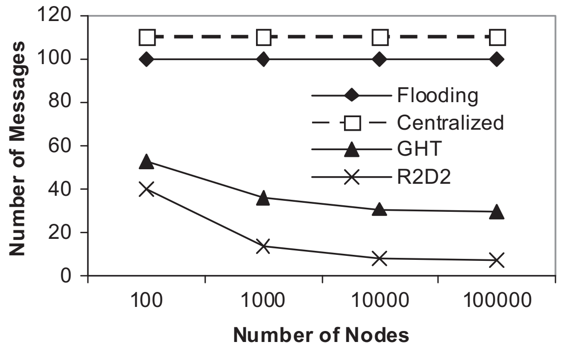

Fig. 25. Hotspot message overhead with increasing number of nodes LIR = 10.

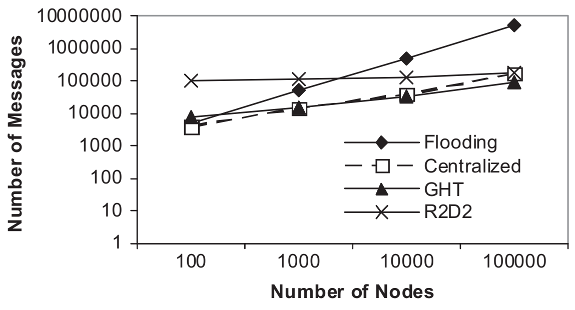

Fig. 26. Total message overhead at low LIR, LIR = 0.05.

This technique could be used similarly in both GHT and R2D2. The figures show that R2D2 will have higher overhead at low number of nodes, because of the insertion geocast overhead with large number of insertions. As the number of nodes increases, R2D2 will be similar to centralized and GHT, since the unicast hops will dominate the overhead in large networks, which confirms the asymptotic overhead shown in the table.

5. Additional Design Considerations

In this section, we consider additional design issues: the existence of gaps in the region, empty regions and region resizing.

5.1. Region Gaps

As shown in Section 3, when a key is inserted in a region, it is stored by all the elected servers in the region. We use geocasting in order to reach all the nodes in the region including the servers. In order to achieve consistency between insertions and lookups, we need a geocasting mechanism that can reach all

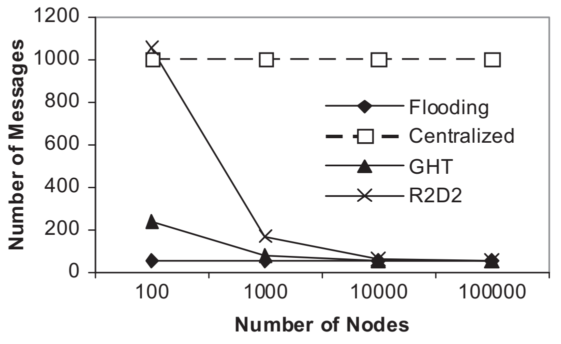

Fig. 27. Hotspot message overhead at low LIR, LIR = 0.05.

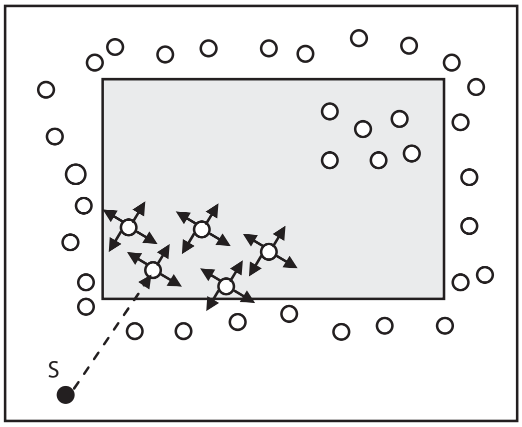

Fig. 28. A gap (disconnection) in the geocast region. A packet flooded in the region cannot reach all nodes without going out of the region.

nodes in the region, otherwise lookups may query servers that are not reached by the insertion geocast. A simple geocasting mechanism is to forward the packet to the region and then flood inside the region boundaries. This mechanism is sufficient at high density networks when all nodes inside the region are connected. But if a gap exists inside the region such that nodes in the region are not connected without going outside the region, this simple mechanism may fail as shown in Figure 28. To solve this problem we proposed a novel geocasting mechanism, Geographic-Forwarding-Perimeter-Geocast (GFPG) [27] that provides guaranteed delivery to all nodes in the region without global flooding or global network information even at low densities and with the existence of region gaps or obstacles. GFPG uses a combination of region flooding and face routing to reach all nodes in the region as shown in Figure 29. In Reference [27], we evaluated this mechanism in detail.

5.2. Empty Regions

One aspect of R2D2 we need to consider is how to deal with empty regions that have no nodes inside. With the relatively large size of regions, this is unlikely to happen when the nodes are uniformly distributed. But for a more general solution that can deal with irregular distributions and obstacles covering a region, we need to consider that. The solution we provide for this problem is to store keys belonging to the empty region in the region of the node closest to that empty region. In both insertions and lookups, a packet forwarded to the empty region will reach the node closest to the region. The node closest to the region will send the packet in perimeter mode, and since there are no nodes closer to the region, it will receive the packet back and it will know that it is the closest node. The node then will insert or lookup the key in its own region similar to regular keys. A server storing keys belonging to other region will have to check periodically that the region is still empty and that no node from other region has become closer to the empty region. If this happens, the key (or a pointer to it) has to move to the other region.

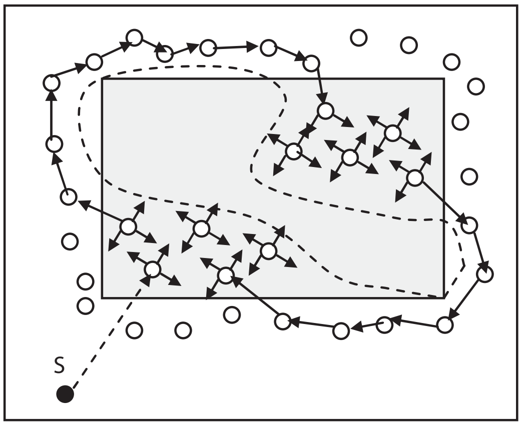

Fig. 29. A mix of region flooding and face routing to reach all nodes in the region. Nodes around the gap are part of the same face. For clarity, here we are showing only the perimeter packet sent around the empty face, but notice that all region border nodes will send perimeter packets to their neighbors that are outside of the region.

5.3. Dynamic Region Resizing

One option we examined is the dynamic resizing of the regions based on the network conditions. We decided not to perform dynamic resizing due to the complexity it will cause and the extra overhead, while the benefits are not clear. We believe the region size should be set as a multiple of the radio range and since the radio range is fixed, then the region size is also fixed. The tradeoff in region size is between the flexibility to mobility and inaccuracy larger regions provide and the low geocast overhead of smaller regions. Typically, a region size covering of a few radio hops is adequate to overcome the effects of inaccuracy and mobility, while our simulations show that reducing the region size too much (closer to the radio range) does not have significant effect on the geocast overhead, since most nodes become border nodes of the region and affect neighbor regions.

Accordingly, increasing or reducing the region size beyond a few hops (e.g., 3–5 hops) of the radio range is not beneficial and it is preferable to keep it fixed as a factor of the radio range.

6. Conclusions

This paper presents geographic rendezvous-based mechanisms as a reliable and efficient approach for emergency communication and data dissemination. An architecture based on geographic regions is explained and evaluated using detailed simulations of a realistic wireless environment, including the physical details and node dynamics. In addition, high-level simulations and analysis are used to study its scaling properties. The results show this approach as scalable to large number of nodes and highly efficient. It is also robust to node failures and mobility, and it relaxes the requirements for the geographic accuracy of node positions and network boundaries compared to an approach based on geographic points instead of regions. In addition, the approach is flexible in selecting which nodes within the region to become servers and store the keys. Selecting stable nodes with higher power and memory can have a significant advantage in heterogeneous networks, where nodes have different capabilities and resources. The architecture is flexible enough for building a wide range of emergency applications and services and can accommodate data with different semantics.

Acknowledgement

This work has been supported in part by NSF under grants number 0347621 and 0134650 and a gift from Intel.

References

1. Kumar S, Alaettinoglu C, Estrin D. Scalable object-tracking through unattended techniques (SCOUT). *IEEE ICNP*, 2000.
2. Li J, Jannotti J, Couto D, Karger D, Morris R. A scalable location service for geographic ad hoc routing (GLS/Grid). *ACM MOBICOM*, 2000.
3. Ratnasamy S, Karp B, Yin L, *et al*. GHT: A geographic hash table for data-centric storage. *Proceedings of the ACM Workshop on Sensor Networks and Applications*, September 2002 and *ACM MONET Journal*, 2003.
4. Karp B, Kung HT. GPSR: greedy perimeter stateless routing for wireless networks. *ACM MOBICOM*, 2000.
5. Zhang W, Cao G, La Porta T. Data dissemination with ring-based index for sensor networks. *IEEE ICNP*, 2003.
6. Xu Y, Heidemann J, Estrin D. Geography-informed energy conservation for ad-hoc routing. *ACM MOBICOM*, 2001.
7. Greenstein B, Estrin D, Govindan R, Ratnasamy S, Shenker S. DIFS: a distributed index for features in sensor networks. *IEEE SNPA*, 2003.
8. Li X, Kim Y, Govindan R, Hong W. Multi-dimensional range queries in sensor networks. *ACM Sensys*, November 2003.
9. Ganesan D, Greenstein B, Perelyubskiy D, Estrin D, Heidemann J. An evaluation of multi-resolution storage for sensor networks. *ACM SenSys*, 2003.
10. Aydin I, Shen CC. Facilitating match-making service in ad hoc and sensor networks using pseudo quorum. *IEEE ICCCN*, 2002.
11. Tchakarov J, Vaidya N. Efficient content location in wireless ad hoc networks. *IEEE International Conference on Mobile Data Management (MDM)*, January 2004.
12. Seada K, Helmy A. Rendezvous Regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks. *IEEE/ACM IPDPS 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, New Mexico, April 2004.
13. Helmy A. Architectural framework for large-scale multicast in mobile ad hoc networks. *IEEE ICC*, April 2002.
14. Estrin D, Handley M, Helmy A, Huang P, Thaler D. A dynamic bootstrap mechanism for rendezvous-based multicast routing. *IEEE INFOCOM*, 1999.
15. Ward A, Jones A, Hopper A. A new location technique for the active office. *IEEE Personal Communications*, October 1997.
16. Priyantha NB, Chakraborty A, Balakrishnan H. The cricket location-support system. *ACM MOBICOM*, 2000.
17. Bulusu N, Heidemann J, Estrin D, Tran T. Self-configuring localization systems: design and experimental evaluation. *ACM Transactions on Embedded Computing Systems (TECS)*, 2003.
18. Savvides A, Han CC, Srivastava MB. Dynamic fine-grain localization in ad-hoc networks of sensors. *ACM MOBICOM*, 2001.
19. Hightower J, Borriello G. Location systems for ubiquitous computing. *IEEE Computer*, August 2001.
20. Bose P, Morin P, Stojmenovic I, Urrutia J. Routing with guaranteed delivery in ad hoc wireless networks. *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM)*, 1999.
21. Kuhn F, Wattenhofer R, Zollinger A. Worst-case optimal and average-case efficient geometric ad-hoc routing. *Mobihoc*, 2003.
22. Seada K, Zuniga M, Helmy A, Krishnamachari B. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. *ACM 2nd Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, November 2004. Extended version accepted for publication in *ACM Transactions on Sensor Networks*.
23. NS Network Simulator. http://www.isi.edu/nsnam/ns
24. Ni S, Tseng Y, Chen Y, Sheu J. The broadcast storm problem in a mobile ad hoc network. *ACM MOBICOM*, 1999.
25. Broch J, Maltz DA, Johnson DB, Hu YC, Jetcheva J. A performance comparison of multi-hop wireless ad-hoc network routing protocols. *ACM MOBICOM*, 1998.
26. Seada K, Helmy A, Govindan R. On the effect of localization errors on geographic face routing in sensor networks. *IEEE/ACM 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, April 2004.
27. Seada K, Helmy A. Efficient and robust geocasting protocols for sensor networks. *Computer Communications Journal*, Special Issue on Dependable Wireless Sensor Networks 2006; **29**(2): 151–161.

Authors' Biographies

Karim Seada received his Ph.D. in Electrical Engineering and M.S. in Computer Science from the University of Southern California (USC), and his M.S. and B.S. in Computer Engineering from Cairo University. He is currently a Member of Research Staff at Nokia Research Center in Palo Alto, California, working on experimental research for building innovative mobile services and systems. His research interests include wireless mesh, *ad-hoc*, and sensor networks, in addition to geographic services and location-based protocols. He held previous internships at Intel Network Architecture Lab and Dust Networks. URL: http://research.nokia.com/people/karim_seada/index.html

Ahmed Helmy received the B.S. degree in electronics and communications engineering with highest honors and the M.S. Eng. Math. degree from Cairo University, Egypt, in 1992 and 1994, respectively, and the M.S. degree in electrical engineering and the Ph.D. in computer science from the University of Southern California (USC) in 1995 and 1999, respectively. He is an Associate Professor and the Founder and Director of the wireless networking laboratory in the Computer and Information Science and Engineering (CISE) Department, University of Florida, Gainesville. From 1999 to 2006, he was an Assistant Professor of electrical engineering (EE) at the University of Southern California. He was also the Founder and Director of the wireless networking laboratory at USC. He was a key researcher in the network simulator (NS-2) and the protocol independent multicast (PIM-SM) projects in the Information Sciences Institute (ISI), USC. His research interests lie in the areas of network protocol design and analysis for mobile *ad-hoc* and sensor networks, mobility modeling, design and testing of multicast protocols, IP micromobility, and network simulation. In 2002, he received the US National Science Foundation (NSF) CAREER Award. In 2000, he received the USC Zumberge Research Award, and in 2002, he received the best paper award from the IEEE/IFIP International Conference on Management of Multimedia and Mobile Networks and Services (MMNS). In 2003, he was the EE nominee for the USC Engineering Jr. Faculty Research Award and a nominee for the Sloan Fellowship. In 2004 and 2005, he got the best merit ranking in the EE-USC faculty. In 2007, he was a winner in the ACM MobiCom SRC research competition. He has been an area editor of the *Ad-hoc* Networks Journal, published by Elsevier, since 2007 (editor since 2004). He is the co-chair for the IFIP/IEEE MMNS 2006 and IEEE INFOCOM Global Internet Workshop 2008 and the vice chair for IEEE ICPADS 2006 and HiPC 2007. He has been the ACM SIGMOBILE workshop coordination chair (for ACM MobiCom, MobiHoc, MobiSys, and SenSys) since 2006. He served on the program committees for numerous IEEE and ACM conferences in the areas of computer and wireless networks. URL: http://www.cise.ufl.edu/~helmy