Rendezvous Regions: A Scalable Architecture for Service Location and Data-Centric Storage in Large-Scale Wireless Networks [*]

Karim Seada, Ahmed Helmy
Electrical Engineering Department, University of Southern California
{seada, helmy}@usc.edu

Abstract

In large-scale wireless networks such as mobile ad hoc and sensor networks, efficient and robust service discovery and data-access mechanisms are both essential and challenging. Rendezvous-based mechanisms provide a valuable solution for provisioning a wide range of services. In this paper, we describe Rendezvous Regions (RRs) - a novel scalable rendezvous-based architecture for wireless networks. RR is a general architecture proposed for service location and bootstrapping in ad hoc networks, in addition to data-centric storage, configuration, and task assignment in sensor networks. In RR the network topology is divided into geographical regions, where each region is responsible for a set of keys representing the services or data of interest. Each key is mapped to a region based on a hash-table-like mapping scheme. A few elected nodes inside each region are responsible for maintaining the mapped information. The service or data provider stores the information in the corresponding region and the seekers retrieve it from there. We run extensive detailed simulations, and high-level simulations and analysis, to investigate the design space, and study the architecture in various environments including node mobility and failures. We evaluate it against other approaches to identify its merits and limitations. The results show high success rate and low overhead even with dynamics. RR scales to large number of nodes and is highly robust and efficient to node failures. It is also robust to node mobility and location inaccuracy with a significant advantage over point-based rendezvous mechanisms.

1. Introduction

Current research in infrastructure-less wireless networks can be categorized into two main categories: mobile ad hoc networks and sensor networks. There are many similarities between the two categories, but the major challenges are typically different. For efficient service provisioning, the challenges in ad hoc networks are the lack of infrastructure and the highly dynamic nature of nodes and their unpredicted mobility patterns. While in sensor networks the challenges are mainly the limited resources and the extremely large number of nodes. Some applications of sensor networks involve also mobility. Communication in sensor networks is typically application-specific and data-centric, and it consists of the tasks sent to nodes and the data recorded by nodes about the environment. Typical approaches for locating resources and data items in these networks rely on either flooding or centralized external storage. Both could suffer from scalability and efficiency problems. In this paper, we describe Rendezvous Regions (RRs) - a novel self-configuring, scalable, efficient and robust rendezvous-based architecture.

In infrastructure-based networks as 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 be frequently changing causing large extra 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. In our architecture, we use 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 mobility.

In our architecture, 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. A key, k_i, is mapped to a 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

[*] This work was supported by grants from NSF CAREER, Intel and Pratt&Whitney.

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 current servers move inside their region and hence the overhead due to mobility updates is quite manageable.

We run extensive detailed simulations to investigate the design space, and study the architecture in various environments including node mobility and failures. In addition, we perform high-level simulations and analysis to analyze RR scalability and evaluate it against other approaches; flooding, centralized storage and GHT [15], to identify its merits and limitations. The results show that RR is scalable to large number of nodes and is highly efficient and robust with node mobility, failures, and location inaccuracy.

The rest of the paper is outlined as follows. In Section 2 we discuss related work. In Section 3 we provide the context and assumptions under which our architecture operates. Section 4 explains the design and section 5 contains the detailed evaluation of the architecture. Conclusions are presented in Section 6.

2. Related work

In wireless networks, the simplest form of data dissemination or resource discovery is global flooding. Flooding does not scale well. Other approaches that address scalability employ hierarchical schemes based on cluster-heads or landmarks [11]. 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 [12] 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.

The original RR idea borrowed from our earlier work on PIM-SM rendezvous mechanism [4] that uses consistent mapping to locate the rendezvous point (RP). However, a rendezvous *point* is insufficient in a highly dynamic environment as wireless networks. We first hinted at the RR idea in [6], in the context of bootstrapping multicast routing in large-scale ad hoc networks, with no protocol details or evaluations. In this work, we present a detailed architecture for RR, with full description of the design and the mechanisms to deal with mobility, failures, and inaccuracies, and generalizing it to deal with resource discovery and data-centric architectures in general.

Our architecture requires nodes to know their approximate locations. Location-awareness is essential for many wireless network applications, so it is expected that wireless nodes will be equipped with localization techniques. In general, many localization systems have been proposed in the literature: GPS, infrastructure-based localization systems [21][14], and ad-hoc localization systems [3][16]. For an extensive survey of localization refer to Hightower *et al.* [7]. 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. [8][10]) have been proposed. GPSR [8] is a geographic routing protocol for wireless networks that works in two modes: 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 neighbors 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 or RNG, to discard a subset of the physical links during perimeter routing, so that the resulting graph is planar.

As mentioned earlier, we presented a high-level description of the original RR idea in [6]. Later on, in GHT [15] a related idea was proposed for data-centric storage in sensor networks. GHT 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 the key. GHT requires nodes to know their exact geographic location and uses the GPSR [8] protocol, to reach the destination. GHT uses GPSR perimeter routing 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 destination), and traverse the perimeter that enclose 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 uses perimeter routing to refresh keys periodically, in order to detect topology changes after failures or mobility.

The obvious distinction between RR and GHT is using a rendezvous region instead of a rendezvous point. However, the design goals and architectural details are quite different. GHT was designed for sensor networks with low mobility, while a main goal in RR design is to target also high mobility environments. RR 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, as will be explained in the design section.

3. Context

In this section, we present the data model and applications we consider in our study. We then give a brief statement about the geographic requirements of Rendezvous Regions and present our assumptions.

3.1. Data model

RR provides a general architecture for resource discovery and data-centric storage. Data operations could be viewed as general insertions and lookups of keys. Different applications differ in the number and characteristics of their insertions and lookups. The ratio between lookups and insertions affects the performance of our architecture; we call it *LIR* (Lookup-to-Insertion Ratio). In our model, the number of lookups is normally larger than the number of insertions. Hence, this model has large LIR. Stored data are long-lived and queried continuously by large number of nodes. Examples of the applications we consider are:

- Service location & bootstrapping in ad hoc networks.
- Users and object tracking.
- Configuration & task assignment in sensor networks.
- Database querying in sensor networks ([5][1]).

3.2. Geographic requirements

A main objective of our architecture is to *relax the requirements for exact geographic information*. Nodes need only to know their regions and so the exact location may not be required. We design our system as an *overlay* that can run over different routing protocols. The routing protocol need not be a geographic routing protocol that requires the exact geographic positions of nodes. It can be any routing protocol augmented to provide approximate routes toward regions.

In our recent studies, we show how location errors, caused by localization systems and inaccuracy [19], inconsistency of location dissemination [8], or node mobility [20], result in severe performance degradation and correctness problems in geographic routing protocols as GPSR [8] and GOAFR [10], in addition to GHT [15]. It is therefore crucial to provide alternative architectures that relax the assumptions of availability of accurate location information. Our Rendezvous Regions approach provides one such architecture and is more robust to errors and imprecision in location measurement and estimation than schemes depending on exact geographic location. In the results section, we show the effect of location inaccuracy on RR and GHT.

3.3. Assumptions

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 an adequate relaxation of the inaccuracy and mobility effects, while keeping the region flooding overhead and server load reasonable. We studied the effect of the region size in detail using simulations. 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. Using appropriate density, we assume that the network is connected and that each region has nodes in it, which is a valid assumption under a reasonable density. In case of partitions and empty regions, multiple hash functions can provide substitute regions, when the original region is not available.

4. Design overview

The network topology space is divided into geographical regions (*RRs*), where each region (e.g., RR_j) is responsible for a set of resources. The resource key space is divided among these regions, such that each resource key (K_i) is mapped to a region. The key-set to *RR* mapping ($KSet_i \leftrightarrow RR_j$) is known by all nodes.

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. Given that the packet is able to reach the region, there are several design options inside the region itself. These design options affect the operation of insertions, lookups, server election, and replication inside the regions. They also affect the consistency operations for mobility and failures.

The main options for forwarding packets inside the region are (a) Geocast, (b) Anycast, and (c) Unicast.

(a) *Geocast*: By geocast we mean sending the packet to all nodes in a geographical area. Geocasts suffer from a high overhead, but are practical when we want to send the packet to several non-determined nodes (in our case to the servers) within a specific geographic region. It is also robust in the face of dynamics and does not depend on the underlying routing protocol. In [17] we present and study efficient geocast routing protocols that guarantee the delivery of packets to all nodes within the region. This is critical for the consistency of the insertion/lookup operations in RR.

(b) *Anycast*: Anycasts are used when it is sufficient to reach any node of a set of nodes (any server). It has similar advantages to geocasts and can be implemented by using expanding ring search techniques or by caching previously known servers.

(c) *Unicast* to servers: Direct unicasts can be used when the locations of servers are well-known, either because the servers are fixed or because they coordinate to keep their information up-to-date.

In our current design we use geocasts for insertions and anycasts for lookups. These design choices are simple to implement, robust to dynamics, and do not require tracking of nodes' locations. Following we describe the main components of our architecture.

- **Region detection:** Using a localization mechanism [7], each node detects its location and accordingly its geographic region. When the node moves, it detects its new location and so it can keep track of its region. The node uses this information to forward packets toward regions, to detect packets forwarded to its region, and to potentially participate in server election in its region (if and when needed).

- **Server election:** A simple local election mechanism is used inside the region to dynamically promote the servers. As the number of servers increases, the robustness to mobility and failures increases, but also the storage overhead increases. Servers are elected on-demand during insertions. When a data insertion operation is issued, the first node in the region that receives the insertion[1], 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* and if it 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 1). 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 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 1). If the flooder receives no reply or if it has no cached servers, it geocasts the lookup inside the region.

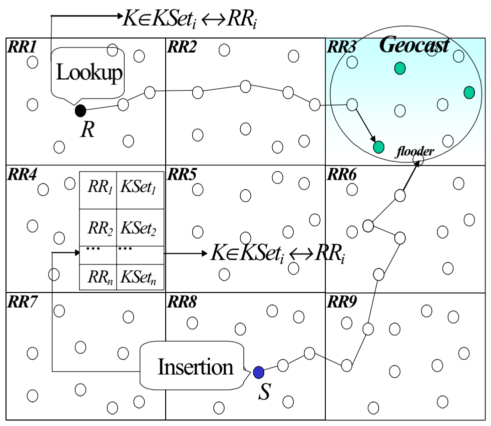

Figure 1: Insertion (step I): Node *S* wishing to insert (or store) resource key *K* that belongs to $KSet_i$ gets the corresponding *RR* (in this case *RR3*) through the mapping ($KSet_i \rightarrow RR_i$). (step II): Node *S* sends the resource information towards *RR3*, where it is geocast by the flooder and stored by the servers.
Lookup: Node *R* looking for a resource with key *K* that belongs to $KSet_i$ gets the corresponding *RR* (in this case *RR3*) through the mapping ($KSet_i \rightarrow RR_i$). *R* then sends the resource lookup towards *RR3*, where it is anycast to any server holding the information.

- **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 only condition we

[1] 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.

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 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 anyway, 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.

5. Performance evaluation

We evaluate the correctness and robustness of our architecture by performing detailed NS-2 [13] 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 study also the effect of inaccurate locations. We verify the correct operation of RR and evaluate its performance under different scenarios. In addition, we run similar detailed experiments for GHT in order to compare the performance of both systems and identify their characteristics and limitations.

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 with up to 100,000 nodes and compare it to GHT, flooding (local storage), and centralized storage (external storage).

RR is running as an overlay over the routing layer and we are using GPSR as the wireless routing protocol. We modified GPSR to route to regions instead of specific destinations by forwarding the packet toward the center of the region and using geocast or anycast inside the region. 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, and the expected number of queries per second. The design parameters that we study are mainly the region size 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.

5.1. Detailed simulation results

We implemented RR in NS-2 as an overlay that runs over any wireless routing protocol that can route towards 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 sec and the beacon expiration is 4.5 sec. We explored several transmission ranges between 60m and 120m and different number of nodes between 100 and 400. For convenience (and space limits), we focus on 100 nodes having 80m transmission range. The density is fixed to 1/1024m^2. 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 seconds. Keys are randomly uniformly distributed over the space. The results are the average of 5 random runs over 5 different random topologies. GHT was already implemented in NS-2. The refresh interval for GHT is set to 10 seconds. 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% of each side (left, right, top, bottom) of the space during hashing. We will refer to the modified GHT by GHT*. Due to space limits, in this paper we show, from our detailed simulations, only the mobility results. For more results including the complete details of RR evaluation, the overhead of its different components, the effect of the number of servers, the effect of the region size, and the node failures effects, please see [18].

5.1.1. Mobility results. One of the main strengths of RR is its robustness to node mobility. The 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 [2], with a maximum rate of 1m/s, 2m/s, or 5m/s. In RR, the number of servers required in each region is set to 3. The simulation run is 200 seconds 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. Figure 2 shows the low mobility overhead of RR using different number of regions 4, 9, 16, and 25 (since the space size is fixed, increasing the number of regions reduces the region size) compared to GHT. RR has much lower overhead, since only the servers send updates when they move out of region. The mobility update overhead is counted during an interval equivalent to GHT refreshment interval, since both of them reflect the overhead due to mobility. In Figure 3 and Figure 4, we fix the number of regions of RR to 9 and change the pause time of nodes moving with a maximum random-waypoint velocity of 5m/s. We notice the high success rate and the low lookup overhead compared to GHT and GHT*. The success rate of RR is above 99%. GHT success rate drops faster with mobility, since any small movements can cause changes in the key storage and lookup packets may reach home nodes (nodes closest to destination point) that do not yet 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 (to reduce that effect, GHT replanarization timer is set to 2 seconds).

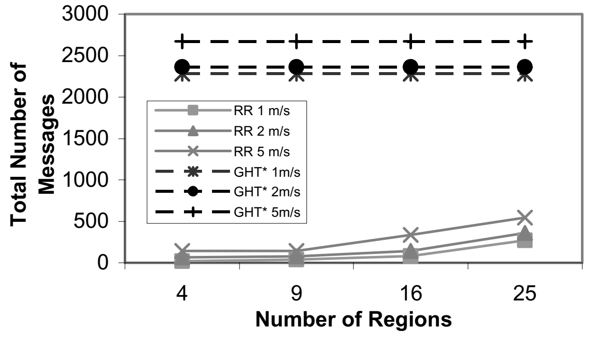

Figure 2: Mobility update (refresh) overhead in RR and GHT

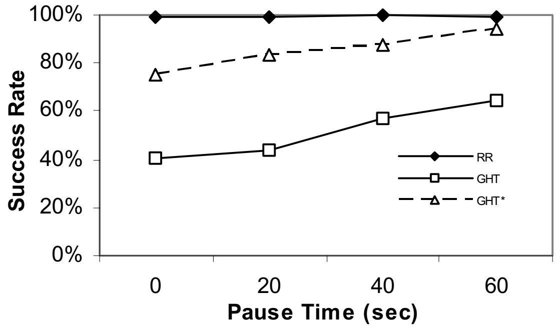

Figure 3: Lookup success rate for different node pause times

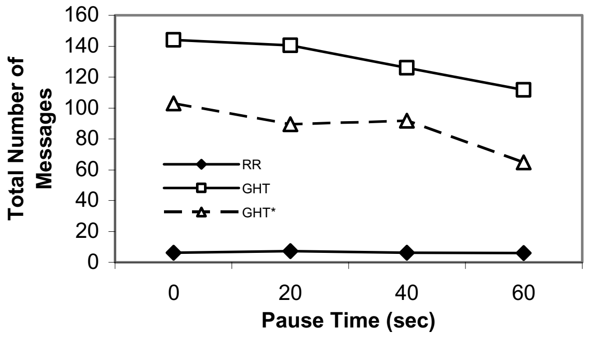

Figure 4: Lookup overhead for different node pause times

5.2. Location inaccuracy

In this section, we study the effect of inaccurate location information on RR. One of the main objectives of RR is to relax the accuracy required by nodes in estimating their location. In this section we consider only the routing behavior in an ideal wireless environment, in order to evaluate the effects of location inaccuracy without the interference from other layers such as MAC collisions or physical layers 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 localization error.

Figure 5 shows the success rate of RR with different number of regions (16, 36, and 64 regions) compared to GHT at low inaccuracy range (2-10% of the radio range). RR is more robust to location inaccuracy than GHT and the effect of inaccuracy is less on larger regions. Figure 6 has a higher inaccuracy range (20-100%), which shows the effects of inaccuracy more clearly. For example, at an inaccuracy equal to 60% of the radio range, GHT success rate goes below 60%, while RR is still above 95%.

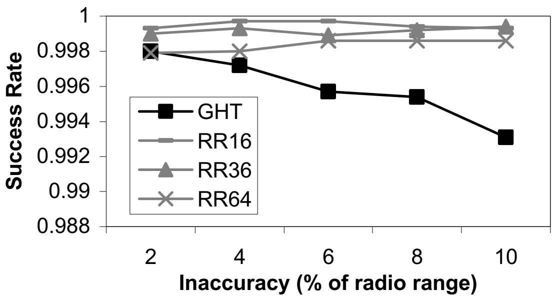

Figure 5: Success rate at low inaccuracy range (2-10% of the radio range)

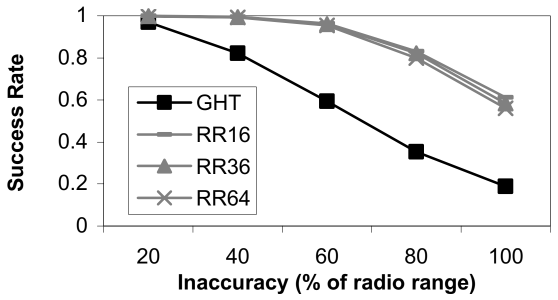

Figure 6: Success rate at high inaccuracy range (20-100% of the radio range)

5.3. High-level simulation & analysis

To evaluate the scalability of RR 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. In a sensor network, the overhead reflects also the energy consumption of a sensor node and the lifetime of the whole network. Without errors or dynamics, the success rate need not be computed (always 100%). We compare RR to GHT, flooding (local storage), and centralized storage (external storage). We count only the insertion and lookup overhead. The periodic refreshment overhead in GHT and the failure check overhead in RR are not included in the high-level simulation results, but we will compute an estimate for them. 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* is the number of insertions, *L* is the number of lookups, *R* is the number of regions in RR, and S is 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	$I\times O(\sqrt{n})+L\times O(\sqrt{n})$	$O(\frac{L}{I})$, {for L > I}
RR	$I\times O(\sqrt{n}+\frac{n}{R})+L\times O(\sqrt{n})$	$O(\frac{I}{R}+\frac{L}{\min(I,R)\times S})$

The total message overhead in RR 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 RR, 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) * average perimeter length'*. In RR, failure check overhead per interval is at most *'n/R * minimum (I,R)'*, since we do not need to geocast in regions that has 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 RR 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 RR 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 100000 with different LIRs. The number of insertions is 10. The density is similar to the detailed simulations and the region size in RR is set to have an average of 100 nodes. The results are the average of 10 random simulations with 10 random topologies. In Figure 7, we see flooding has the highest message overhead since each lookup is flooded. Centralized, GHT, and RR have close total overhead. Figure 8 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 RR. RR has lower hotspot overhead than GHT in these scenarios, because the lookups for a key in RR are distributed over multiple servers in the region, while in GHT the same home node get all lookups for a certain key. As we notice also from the table, the hotspot overhead of RR is low when *I* is small compared to *R*. As *I* increases above *R*, the hotspot overhead will increase.

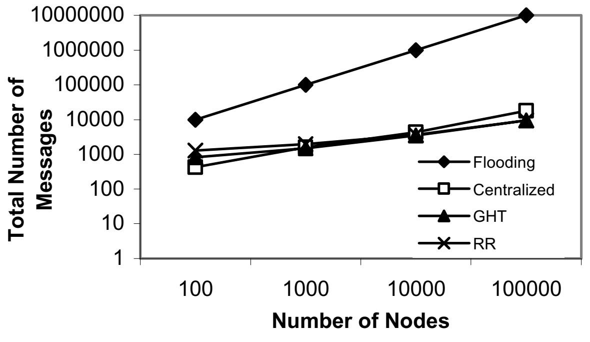

Figure 7: Total message overhead with increasing number of nodes, LIR=10

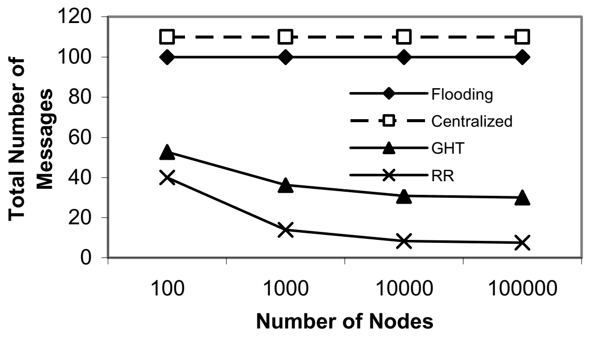

Figure 8: Hotspot message overhead, LIR=10

6. Conclusions & future work

This paper presents the design and evaluation of RR, a scalable rendezvous-based architecture for wireless networks. RR facilitates service location and bootstrapping in ad hoc networks, in addition to data-centric storage, configuration, and task assignment in sensor networks. We evaluated RR using detailed simulations of a realistic wireless environment including the physical details and node dynamics, and compared its performance and robustness to GHT. We studied also the scaling properties of RR using high level-simulation and analysis, and compared its scalability to GHT, flooding, and a centralized approach. The results show that RR is scalable to large number of nodes and is highly efficient, especially in applications with high lookup-to-insertion ratios. It is also robust to node failures and mobility, and it relaxes the requirements for the geographic accuracy of node positions and network boundaries. In mobile networks, RR has a significant advantage over GHT with higher success rate and much lower overhead, since regions provide a dampening factor to the effects of mobility. Other than the mobility advantages, RR is more robust to location inaccuracy than GHT and it requires lower periodic overhead, since periodic refreshments need to be sent per region and not per key. In addition, RR is more flexible in selecting which nodes to become servers and store the keys. It can choose nodes with certain capabilities from within the region, while in GHT the home node of a key is determined solely by the geography. Selecting more stable nodes with more power and memory can have a significant advantage in networks, where nodes have limited power and resources.

References

[1] P. Bonnet, J. E. Gehrke, and P. Seshadri. "Querying the Physical World". *IEEE Personal Communications, October* 2000.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols". *ACM MOBICOM* 1998.

[3] N. Bulusu, J. Heidemann, D. Estrin, and T. Tran. "Self-configuring Localization Systems: Design and Experimental Evaluation". *ACM TECS* 2003.

[4] D. Estrin, M. Handley, A. Helmy, P. Huang, and D. Thaler. "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing". *IEEE INFOCOM 1999*.

[5] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. "The Sensor Network as a Database". *UCLA-TR*, September 2002.

[6] A. Helmy. "Architectural Framework for Large-Scale Multicast in Mobile Ad Hoc Networks". *IEEE ICC*l 2002.

[7] J. Hightower and G. Borriello. "Location Systems for Ubiquitous Computing". *IEEE Computer,* August 2001.

[8] B. Karp and H.T. Kung. "GPSR: greedy perimeter stateless routing for wireless networks". *ACM MOBICOM* 2000.

[9] Y. Kim, J. Lee, and A. Helmy. "Impact of Location Inconsistencies on Geographic Routing in Wireless Networks". *ACM MSWIM* 2003.

[10] F. Kuhn, R. Wattenhofer, and A. Zollinger. "Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing". *ACM Mobihoc* 2003.

[11] S. Kumar, C. Alaettinoglu, and D. Estrin. "Scalable Object-tracking through Unattended Techniques (SCOUT)". *IEEE ICNP* 2000.

[12] J. Li, J. Jannotti, D. Couto, D. Karger, and R. Morris. "A Scalable Location Service for Geographic Ad Hoc Routing (GLS/Grid)". *ACM Mobicom* 2000.

[13] NS Network Simulator. http://www.isi.edu/nsnam/ns.

[14] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. "The Cricket Location-Support System". *ACM MOBICOM* 2000.

[15] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. "GHT: A Geographic Hash Table for Data-Centric Storage". *ACM WSNA* 2002.

[16] A. Savvides, C.-C. Han, and M. B. Srivastava. "Dynamic Fine-Grain Localization in Ad-Hoc Networks of Sensors". *ACM MOBICOM* 2001.

[17] Karim Seada and Ahmed Helmy. "Efficient Geocasting with Perfect Delivery in Wireless Networks". *IEEE Wireless Communications and Networking Conference (WCNC), Atlanta, Georgia,* March 2004.

[18] Karim Seada and Ahmed Helmy. "*Rendezvous Regions*: A Scalable Architecture for Service Location and Data-Centric Storage in Large-Scale Wireless Networks". *USC Technical Report*, June 2003.

[19] Karim Seada, Ahmed Helmy, and Ramesh Govindan. "On the Effect of Location Inaccuracy on Geographic Face Routing in Wireless Networks". *USC Technical Report*, June 2003. Extended Abstract in *ACM Mobile Computer and Communications Review (MC2R)*, October 2003.

[20] D. Son, J. Park, and A. Helmy. "The Effect of Mobility-induced Location Errors on Geographic Routing in Ad Hoc Networks: Analysis and Improvement using Mobility Prediction". *IEEE WCNC* 2004.

[21] A. Ward, A. Jones, and A. Hopper. "A New Location Technique for the Active Office". *IEEE Personal Communications,* October 1997.