Chapter 33

Location Management in Multi-Hop Wireless Sensor Networks

Stefano Basagni* and Maurizio A. Nanni†

Department of Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
**basagni@ece.neu.edu*
†*mnanni@ece.neu.edu*

The chapter concerns solutions about the problems of managing the location information of nodes of ad hoc and wireless sensor networks (WSNs). Once nodes know about their location, solutions for location management take care of disseminating this important information to all or some of the other network nodes that need it for routing or other purposes. The chapter covers first the solutions that have been proposed for mobile ad hoc networks, where research in this area has started along geographic routing over a decade ago. We then describe solutions in the still largely uncharted territory of WSNs. In particular, we survey methods for location management that have been proposed for different application scenarios, emphasizing how location information is managed, i.e., proactively, reactively, or through dedicated nodes (location servers), which nodes move, and the different communication models adopted. Our review of the solutions currently proposed reveals that further research is needed for providing efficient location management protocols, especially for WSNs, and that the evaluation of their effectiveness for location aware applications and geographic routing is needed for truly identify new protocol design guidelines and improved network performance.

33.1. Localization and Location Management in Ad Hoc and Wireless Sensor Networks

Since the renewed interest in *mobile ad hoc networks*[1,2] in the mid 1990s, it has become clear that *geographically enabled ad hoc routing* (or just *geo routing*)[3,4] is a viable and effective solution for data communications in multi-hop networks with node dynamics and mobility. The reasons are multi-fold. First of all, the routes from source to destination do not have to depend on the identity of intermediate nodes, as it happens in both *proactive* and *reactive* approaches to ad hoc routing.[5] For instance, in a proactive ad hoc routing protocol nodes maintain next-hop

information about every possible destination continuously updated in their routing tables. When a packet for a given destination arrives, the table is checked to determine which is the next node to send the packet to in order to progress toward that destination. If the table is not updated and that node is no longer there (e.g., it has moved, a typical ad hoc event), an exception has to be triggered to solve this situation and the packet is either highly likely to be discarded or it is for sure sensibly delayed. Similarly, in a reactive setting, i.e., where routes are sought between source and destination at the time a packet needs to be forwarded, the packet itself carries the exact route it should go through (source routing). If one of the nodes in that route has moved, the packet suffers either discarding or long delays. In the case of geo routing, it is enough for some nodes to be in between source and destination at the time the packet needs to be forwarded. When a node generates or receives a packet for forwarding, it looks for *any* node in the direction of the destination, independently of its identity, which is instead a key information for proactive and reactive routing. Furthermore, in general ad hoc networks, geo-routing is considered more scalable than non-geo routing given that source nodes have to just store or look for the coordinates of the destination location (usually represented by few bytes) instead of keeping large routing tables updated or caching routes. Location awareness enables also a host of new (ad hoc and non ad hoc) applications. Typical examples include automatic device reconfiguration depending on the current location of the user/device, securing application usage/access (e.g., some features are usable only from specified locations), tracking the current position of a user/resource/device, enabling the computation of travel times depending on current position and final destination, providing traffic information based on predetermined routes, and so on.

The use of location information is even more compelling in *wireless sensor networks* (WSNs), where a large number of generally resource constrained devices (the *sensor nodes*) are deployed for providing an interface with the physical environment. These devices either *observe* the environment, or even *interact* with it (in this case they are called *actor nodes*, and the network is also called a WSAN, for wireless sensor and actor network). Specifically, each sensor node has a sensing component that performs monitoring tasks, some computational capabilities (for manipulating the sensed data) and a communication interface (usually wireless, often radio) through which it interconnects to nodes within its own transmission range to form a multi-hop network. Routes are built and maintained from the (many) sensors to few data collectors (the *sinks*), and vice-versa. Communications sensors-to-sinks serve the purpose of gathering the data at the collection points for further forwarding (for instance, over the Internet) or for processing/storage; The second direction of communication is used to communicate to the sensors what they should sense (*interest dissemination*) or, in case of a WSAN, what are the actions that (some of) the actor nodes should perform in response to some specific events. It is clear that whenever an

event of interest is detected by a sensor node, the location where the event happens should be part of the information to be delivered to the sinks. Furthermore, as for ad hoc networks, geographic forwarding has been shown to be a very effective way for data dissemination and gathering in WSNs. For instance, protocols like GeRaF[6,7] or ALBA–R,[8] use nodal location and the knowledge of the current location of the sink to determine the next hop relay for a data packet. These protocols have been shown to be among the most effective routing solutions for WSNs, being able to meet their most crucial requirements, which include routing reliability, scalability, high throughput, low latency and especially energy conservation.

In order to implement geo routing in ad hoc and WSNs, and, more generally, for enabling location aware applications, two fundamental network operations must be efficiently implemented and readily available.

(1) *Localization.* Nodes needs to continuously know where they are. Mechanisms, technologies and protocols must be provided for each node to be able to determine its current position with respect to an absolute, relative or a virtual set of coordinates.
(2) *Location management.* Once a node knows where it is, there should be an efficient way to let the other nodes know its current location every time it is required.

The problem of localizing ad hoc or WSN nodes has been investigated quite widely. The US Global Positioning System (GPS), the Russian GLObal'naya NAvigatsionnaya Sputnikovaya Sistema (GLONASS) and the more recent and still "under construction" EU Galileo Positioning System[9] are nowadays a relatively easy and inexpensive solution for this problem. GPS technology is commonplace in cars, and more recently even in the most advanced cellphones. When, however, satellite-based systems cannot be used because of environmental limitations (e.g., indoor or in "urban canyons") or because of energy constraints (e.g., a tiny battery-powered sensor mote), protocols must be provided that determine and maintain a node aware of where it currently is with respect to the selected reference system. A host of protocols have been presented in the last decade for both ad hoc and WSNs. The reader is referred to Ref. 10 and Refs. 11 and 12, respectively, for further details and references.

Differently from localization, location management has not being extensively investigated, especially for wireless sensor networks. It is the aim of this chapter to provide a description of what has been done for this important field of research. We will start by presenting a brief summary of location management protocols in the general setting of mobile ad hoc networks, where research on the topic started. We then explore location management solutions for WS(A)Ns, and present the different scenarios and protocols that have been investigated and proposed so far in the literature.

33.2. Ad Hoc Location Management

The management of location information in multi-hop ad hoc networks has been mostly investigated for supporting geo routing. The first general solutions for managing location information were presented with the first geo routing protocols specifically designed for ad hoc networking, namely, *DREAM* (Distance Routing Effect Algorithm for Mobility)[13] and Location Aided Routing (*LAR*).[14]

The world according to DREAM is one where, once retrieved its location (e.g., geodetic coordinates obtained through GPS), a node *proactively* spreads it around so that every other node that might want to send it a message knows (approximately) where it is. Flooding location updates, however, can be detrimental for the network performance, especially when nodes are highly mobile (e.g., in a *vehicular ad hoc network*, or VANET[15]). Therefore, DREAM describes an efficient mechanism for location updates dissemination that is based on two simple, but very effective ideas. The first is termed the *distance effect*, and it is inspired by the concept that closer mobile objects seems to "move more" than objects farther away moving at the same speed. In other words, with respect to the same observer, the direction of the closer object changes more than that of the one that is away. In order to take advantage of the distance effect, packet carrying a location update are assigned a time-to-live based on the distance they have traveled (geographic, or in number of hops): Only few of them will actually flood the whole network. Most of them will only reach closer nodes, since these are the nodes that most need to know that the source of the update has changed location. The second idea that allows DREAM to keep location table well updated is based on the even simpler observation that if a node does not move, or moves slowly, it needs to send out location updates less frequently than nodes that are moving often or at higher speed. In other words, the sending of location updates is mobility-dependent. Once a node receives a data packet for a given destination, it checks the location of the destination carried by the packet, and it also checks its own location table to see if it has a more recent information about the destination whereabouts. The packet is then forwarded to those neighbors that are in the direction of the destination, i.e., through *directional routing*. The closer the packet gets to its destination, the more updated and accurate the location of the destination (because of the distance effect), the faster and less overhead-generating the routing.

The LAR concept takes a more reactive approach to geo routing in the sense that location is used in this case to reduce the overhead of route discovery, typical of reactive ad hoc routing protocols (that usually use flooding for route finding[5]). Based on the knowledge of the location of the destination of a data packet at a certain time, and also of its speed (and direction, if available), the source node estimates where the destination can possibly be, i.e., its *expected zone*. Based on this zone, the source sends a route request that is forwarded only by the nodes of a *request zone*. This eventually allows nodes to route request packets to the expected zone of the destination and to the destination itself. Once the routing probe reaches the

destination, the destination sends back a route reply packet, as in non-geo reactive routing protocols. While data routing is of the reactive type, location information for other nodes is maintained at each node in a proactive fashion, although not periodically or based on nodal mobility as in DREAM. When a source node seeks for a route to the destination, it piggybacks its own current location to its route request packet. All the nodes that receive this packet update their entry for that source node. The same is done by the destination when it sends back a route reply packet to the source. In general, location information can be propagated by using any packet. Similarly, a node may propagate its own current speed and, if available, the direction of its movement (or its destination, if known).

Whether to deliver data (DREAM) or to seek a route (LAR) to the intended recipient of a packet, the two protocols employ a reduced (directional) form of flooding, which becomes full flooding when, for some reasons the destination cannot be found. Such situation arises, for instance, when while progressing toward the known direction of the destination, packets get stuck at a so-called *dead end node*, i.e., a node that does not have neighbors in the direction of the final recipient of the packet. Rather than resorting to flooding, Bose *et al.*[16,17] proposed *GEDIR* plus face routing (and then Karp and Kung[18] propose *greedy perimeter stateless routing*, GPSR), for which a packet gets out of a dead end node by routing around the perimeter of the region where greedy forwarding is impossible. In both cases, the protocol assumes that the nodes have access to an ideal, efficient, location database that provides them with the location of the destination. However, in the GPSR paper the increased overhead of accessing an actual location database is discussed, and the authors claim that research has started to solve the location management problem efficiently. In fact, at the same ACM MobiCom 2000, Li *et al.*[19] presented GLS, a distributed scalable location service for geo routing. A small set of the mobile ad hoc nodes act as location servers, i.e., maintain the location of the other nodes. Each node periodically updates the location servers with its current location. Queries for a mobile node location are sent by source nodes to the location servers, that reply back with the latest whereabouts of the destination. Once a source node receives the location information about the destination, it can forward the data packets by following the rules of one of the geo routing protocols seen above.

It is interesting to notice that GLS is a hybrid approach with respect to the proactive/reactive classification of location management schemes. Methods à la GLS are often referred to as *rendezvous* methods, in that the destinations proactively send their current coordinates to the location servers, the sources reactively query the servers for the destination position, and these two operations *meet* "in the middle," i.e., at the servers. Another characteristic of GLS is that a node sends its current position to the servers without knowing their actual identity. This is made possible by a predefined order of node IDs and by a spatial (geographical) hierarchy, also predefined. More specifically, the network deployment area is partitioned into

a hierarchy of *grids* (whence the name GLS, for *grid location service*) with grid cells (squares) of increasing sizes. A precise order among the squares is defined so that a hierarchical organization of them is obtained that avoids overlapping. This order and the induced hierarchy are such that each node belongs to exactly one square of each size. More specifically, four smallest squares (order-1 squares) make up an order-2 square. Four of these make up an order-3 square, and so on, with the restriction that a lower order square is part of only one upper order square, not four. (This creates a tree-kind of hierarchy.) A node X determines which are its location servers according to its own ID and the predetermined grid hierarchy. The strategy is based on choosing for location servers those nodes whose ID is "close" to X's. In particular, once defined the *closest* node to X as the node that has the least ID greater than X's (the IDs are organized circularly), X chooses the three closest nodes from each of the three square siblings of the one that contains it. Similarly, it chooses three location servers for each level of the grid hierarchy. A node Y that needs to find the location of X uses geo routing to send a request to the node whose ID is the least greater than or equal than X's ID of which Y knows the location. That node forwards the location query in the same way and so on. Since the query contains Y's location, X can reply directly to Y (using geo routing). The whole GLS mechanism is based on a bootstrapping technique that takes care of X's initial selection of location servers. (The reader is referred to the GLS paper[19] for further details and the GLS performance analysis.)

As the initiator of rendezvous-based approaches to location management, GLS has inspired many other works. Xue *et al.*[20] deal with the problem of location management by proposing a *Distributed Location Management* (DLM) scheme aimed at addressing some of what they consider GLS shortcomings. Specifically, the authors argue that a GLS node can potentially scan an entire, large higher order grid before finding its closest location server in that area. Furthermore, when the location server of a node moves away from the grid cell it belongs to, another location server should be selected, and this would impose periodic location server selection, with the corresponding non-negligible overhead. Overall, the authors conclude, GLS performs well in more static environments, and some other method should better take care of mobility. DLM is based on a different partition of the network deployment area. The whole area is called the 0-order region. A first partition of this area into squares creates the 1-order regions, which are further partitioned into 2-order regions, and so on, according to a top-down partitioning (GLS implements a bottom-up one) that uses rectangles of equal size instead of squares. As in GLS, a node X should select location servers only based on its unique ID, since that is the only thing that a potential sender Y knows about X. DLM nodes select location servers as follows. The ID of X is mapped into a set of locations. Any node currently at that location can act as location server for X. Therefore, the crucial operation here is the mapping from X's ID to the locations. In DLM this is performed by using a simple hash

function that takes as input the order of a region, the number of rectangles into which that region is partitioned, and X's ID. For each of the regions at a given level of the partition, based on its ID, X can choose any of the nodes in that region to act as its location server. (This process is optimized to avoid the selection of too many servers[20]). A node Y that wants to get ahold of X's current location uses X's ID to get the location server in its own region, and queries that server for the current location of X. Once the server has provided the information requested, Y can reach X directly, and X can provide more accurate information about its whereabouts. Location updates are sent by X to its servers every time it moves from its current region. When a selected server for a node X moves out of a region R, two possible solutions are suggested. Either the server itself selects another node in that region to act as a server for X and passes X's information to the newly selected node, or X itself could check whether its location server for region R is still there, and if not it selects a new server by using the selection algorithm. Other works based on the GLS idea are those by Sucec and Marsic[21] and more recently by Wang and Wang.[22] In the first case the authors address the problem of fair distribution of location management responsibilities to the location servers, so that none of them become too much of a hot spot, degrading the network performance. Furthermore, through their *Virtual Hierarchy Location Management* (VHLM) they also discuss the communication overhead associated to location registration events that happens every time a node moves. In the case of the paper by the two Wang,[22] their *Modified Grid Location Service* (MGLS) focuses on finding a different (in this case binary) grid partitioning scheme. MGLS is shown to reduce the overhead associated with location updates.

Woo and Singh[23] define the *Scalable Location Update-based Routing Protocol* (SLURP). The SLURP[a] location management is conceptually similar to that used for roaming by Mobile IP.[24] Each node is associated with a home region (a subdivision of the network deployment area, a rectangular region in the paper) by a mapping of its own unique ID (the mapping is static and known to all nodes in the network). When a node moves from a region to another it always informs the nodes in its own home region about its new coordinates (mobility triggered updates). This message is broadcast to all nodes in the node home region. When a node Y has a message for node X, it determines the home region of X and then it sends a request to the nodes in that region. The first node in X's home region that receives the request sends back X's current region to Y, and the rest of the communication happens from Y to X directly, via geo routing. In other words, the nodes in the home region of a node X act as the location servers where X's (approximate) location is proactively maintained.

[a] Note that SLURP is an ad hoc geo routing protocol, not just its location management mechanism. In the chapter however we will make no distinction between the two.

Another rendezvous-based method, termed SLALoM (for *ScaLable Ad-hoc Location Management*), has been introduced by Cheng *et al.*[25] The starting points for SLALoM are the location management solution presented for SLURP[23] and the GLS[19] protocol described above. The latter two protocols have different ways for selecting location servers, with clear pros and cons. The SLURP concept of home region makes the servers independent of the current location of the destination node X. This has the advantage that a source node Y always knows where to retrieve X's location. At the same time, if Y is far from X's home region it could wait a long time before receiving X's location, even when X is close. GLS, on the other hand, does not have a static association between a node ID and a particular region of the network deployment area, and selects servers depending on the current location of X. The density of GLS location servers is higher closer to X, which achieves the advantage that the distance traveled by the location query is proportional to the path length between X and Y. Unfortunately, however, every time X moves all the servers have to be re-appointed, which generates mobility dependent overhead (oftentimes non-negligible). The authors of SLALoM set out to define a protocol that joins the advantages of SLURP and GLS without keeping their drawback. The solution is fairly simple: Instead of having one home region, a node has many of them, uniformly distributed in the network deployment area. This renders the Y's waiting for X's location shorter than *SLURP*'s. However, many home regions require higher traffic for updates. In order to keep this traffic at bay, they use the GLS concept that nodes in those among X's home regions that are close to the current location of X should be receiving X's update, while farther home regions should be updated less frequently (à la distance effect of DREAM).

A similar approach (multiple home regions) is described by Dolev *et al.*[26] The paper defines a geo routing protocol based on the *Virtual Stationary Automata* (VSA) programming abstraction. Said abstraction consists of mobile nodes, the VSAs acting as virtual timed machines, and a local broadcast mechanism for communications among mobile nodes and VSAs. The substantial difference between this paper and previous papers on the topic is that the VSAs are statically deployed, and provide a virtual infrastructure for the communications among the nodes, having broadcast capabilities. Each VSA represents a predetermined geographic area. The location management scheme presented in this paper is based on hash tables and on the VSA-to-VSA routing service. More precisely, a mobile node unique ID hashes to one (or more, for increased reliability and fault tolerance) VSA, which represents the mobile node home region and maintain its location up-to-date.

The techniques for location management described so far, from that of DREAM to those based on location servers, involve all the network nodes for spreading out location updates. The involvement of all nodes brings with it a mobility-dependent, sometimes unbearably high, location update overhead. In order to mitigate this problem Sivavakeesar and Pavlou[27] deploy a geographically-based clustering scheme

through which they define a network backbone (the usual *dominating set*[28]) that is used for location updates. More specifically, every node is assigned to a home zone (basically a virtual cluster with a unique ID) and whenever it moves it is required to update its location with any of the nodes in their home zone (by using the backbone).

Hierarchically-based location management is also explored by Philip *et al.*[29] The authors define *HGRID*, a hierarchical grid location management which, as seen so far, is based on a partitioning of the network deployment area into squares, grouped to form the different layers of the hierarchy. The first layer, formed by these unit squares, is called L_0. The grouping is performed recursively so that the leaders of the world at level L_i are chosen among the elements of the hierarchy at level L_{i-1} based on their position. Location management in HGRID is triggered every time a node crosses an L_0 grid boundary. Location updates are broadcast through the new region and also sent to both the old L_1 grid containing the previous L_0 one (if different from the current) and to the new L_1 grid. These location updates are processed at each level of the hierarchy, so that level leaders maintain a consistent vision of where the network nodes are. When a node Y needs to know the location of a node X a location query packet is sent to Y's L_1 leader. The query climbs up the hierarchy until it reaches a level L_i that contains both Y and X. At this time the location server of the ith level returns X location to Y. The location update analysis performed in the paper shows that the average per node signaling overhead grows only logarithmically with the number of nodes in the network. Simulation-based comparison with SLURP and SLALoM shows that HGRID achieves greater scalability with respect to network size and that it also adapts better to node mobility.

A hierarchically organized system of *pointers* is used for location retrieval by Flury and Wattenhofer.[30] This location management system is called MLS (this acronym is not spelled out in the paper). The MLS world is a big square with land (the region where the nodes are densely deployed) and lakes (regions with no nodes), called level-M of the lookup system. Similarly to what seen in many solutions so far, and to geographic hash tables,[31] each node has a designated (home) position on land where its current location is maintained. Instead of its (estimated) location, however, a node stores in which one of the four sub-squares of the world it resides. Recursively, each selected square stores a pointer to its sub-square that contains the node. The lookup terminates when a square is reached with a predetermined side ρ. The pointers to each square are stored at nodes in that area. The selection of the location server in this case is determined by the vicinity of a node to a specific position in the area. (The presence of such a node is "guaranteed" by the assumption of high nodal density.) Any hash function mapping a node unique ID to a position on land is a suitable function for this mechanism. Node updates and node lookup are similar to the rendezvous protocols seen so far.

At the end of this section we want to mention a method that addresses the problem to determine the current location of a node for a purpose different than that of enabling geo routing. This is the case of the joint solution to the problems of mobility management and location management addressed by Haas and Liang.[32] Although geared toward ad hoc networking, this work has a more "cell phone network" flavor, in that it considers location databases and discusses how to extend and distribute the concepts of *home location register* and *visitor location register* to work well in the ad hoc setting. Nodal location is sought for by the network so that calls to a node can be efficiently delivered to it, and the location of a node is actually defined based on its vicinity to a location database (in the specific case illustrated in the paper the location of a node is the ID of its nearest location database, although other location indications are possible). In this sense this work is much more a work on mobility management for ad hoc network than a solution for location management to support geographic routing. Some of the network nodes act as location databases and organize themselves into a virtual backbone that is maintained as the nodes move. Location searches are performed through the backbone, while once a node location is determined, routing happens in the flat network (i.e., the network with no hierarchical organization) and packets do not have to necessarily go through backbone nodes. (The routing itself does not need to be geo routing.) The location databases are organized as a class of *uniform quorum systems*, i.e., sets of databases every two of which intersect at a constant number of databases (and no quorum contains another). When a location update or a call for a mobile node arrives at a database the mobile ID is stored or read from all the databases of a quorum, chosen non-deterministically. The details of the method are conveyed in the paper, which also describes how to partition the ad hoc network so that each node has a database suitably close, even in presence of mobility. As mentioned, although the selection and use of the location databases could be used for location management as intended in this chapter, the actual aim of this paper is to implement efficient mobility management à la cellular networks.

Performance comparisons among different techniques for ad hoc location management as well as among specific location management protocols has been performed by Camp *et al.*[33] as well as by Das *et al.*[34]

Further information and references on location management for ad hoc networks can be found in the surveys by Ivan Stojmenovic[35] and by Mauve *et al.*[36] The reader is also referred to the paper by Friedman and Kliot[37] for a survey on the more general topic of *location services* for (hybrid) ad hoc networks.

33.3. Location Management in Wireless Sensor Networks

As a specialized paradigm of the general ad hoc networking model where nodes are generally heavily resource constrained, wireless sensor networks (WSNs) have been initially investigated with a strong emphasis on energy conservation rather than

mobility. In fact, mobility has not been a problem at all in early WSNs research, since most of the investigated scenarios concerned networks where a large number of *static* wireless sensor nodes would self-configure into a multi-hop network and would deliver data to a small number (usually equal to 1) of *static* data collectors (the *sinks*). A host of protocol stack solutions, new technologies and techniques have been presented in the past decade that focus on how to prolong the lifetime of tiny, resource limited *motes* that are usually battery powered and deployed in regions where recharging their batteries or changing them is not viable, or impossible all-together. Only recently, after so many physical, MAC and network layer solutions have been defined and tested for static wireless sensor networks, an interest in the mobility of some of the network components has arisen. Two are the main research directions. The first and perhaps most investigated concerns the *exploitation* of the mobility of the devices for the betterment of network performance.[38–50] In particular, research have explored ways of moving through the network the least constrained nodes (such as the sinks, or resource-rich relays acting as middle-men between the sensors and the sinks) so to prolong network lifetime. A thorough survey of this kind of mobility is beyond the scope of this chapter, and the interested reader is referred to works by Basagni *et al.*[50–52] and Ekici *et al.*[53] The other direction is similar to that of ad hoc networks, where basically all nodes can move, and there is the need to define protocols that know how to handle routing toward the (possibly mobile) sinks.

With device mobility comes the need to manage their changing location, and therefore the need to define location management protocols that take into account the different requirements of wireless sensor networking applications. In fact, applications for these networks have requirements so specific that each of them defined a different scenario. This is quite different from ad hoc networking. For instance, while in ad hoc networks all nodes are mobile and they have overall the same characteristics and importance, in WSNs devices are very heterogeneous: Sensors are different from sinks, which in turn might be different from relay nodes (e.g., as in the data MULEs scenario[39]). The application mandates which network component is necessary. Ad hoc nodes can all be source and destination of communications (peer-to-peer model). As a consequence, they all query for the location or other nodes, or maintain up-to-date information about it at location servers. The WSN model of communication is greatly application dependent: For environmental monitoring, for instance, data goes from the sensors to one sink (many-to-one model) and from the sink to all sensors (e.g., for interest dissemination; one-to-all model). As a consequence, if a sink moves, the sensors need to query where the sink is, and not the sinks themselves.[50] In tracking applications, the objects to be tracked (humans, robots, resources) can be roaming through the network, and either themselves or the closest sensors should take care of updating their location at the location servers. In case of a WSAN (wireless sensor and actor network), actors can have completely different requirements than those of sensors or sinks. They might need to retrieve

the location of the sensor that has sensed the occurrence of a given event. Or they might make their own location known to the sensors or to the sinks, so that they can be reached in case they are needed.

Because of this variety, we present the mechanisms for WSN location management one by one, and for each one we focus on the description of following characteristics:

- The application scenario, with a particular emphasis on which network components move.
- How location information is managed: Proactively, reactively or through location servers (rendezvous).
- Which network components need to be aware of their location.
- The communication model.

The description of the protocols follows. The section concludes with a table that lists all the major protocol characteristics (Table 33.1).

33.3.1. *GMRE: The greedy maximum residual energy protocol and its proactive location management*

GMRE[50,54] is a distributed heuristic designed for WSNs where a large number of sensors are deployed either on a grid or randomly and uniformly, and one sink roams through the network and sojourns at specified *sink sites*. The sink moves in order to drain energy from the nodes evenly, which obtains longer network lifetime. Balanced energy consumption is achieved by making the sink aware that somewhere else in the network there are nodes with a higher level of energy. At that time the sink greedily moves to the neighborhood with the maximum residual energy. Since when the sink is in transit from a site to the next one nodes cannot send their data packets to it, GMRE mandates that before moving the sink floods the networks with a "stop" packet, which makes the nodes buffer all their data packets, and that upon arriving to the next site the sink sends a "resume" packet that has the twofold effect of building new routes to the current position of the sink, and to resume packet transmission. Therefore, the location of the sink (which is its coordinates only if the routing from the sensor to the sink is geo routing, but that in general is just a route to it) is proactively maintained at the sensor nodes, without them having to query for it. However, sensors and sink need to be aware of their own position,[50] which can be obtained via GPS, or, when not possible, by executing a WSN localization algorithm.[11] Once routes are set up, and the sensors know the new position of the sink, the communication flow is from the sensors to the sink.

Basagni *et al.*[52] also describe a solution for a similar scenario with multiple sinks where the GMRE proactive location management is used for route management as described above. As noticed for DREAM and LAR (Sec. 33.2), using flooding can be too expensive in terms of energy consumption and nodal overhead, and therefore,

methods based on intermediate nodes that can provide cheaper mechanism have been developed.

33.3.2. *ACT: Actors location management in Wireless Sensor and Actors Networks (WSANs)*

Wireless Sensor and Actor Networks (WSANs)[55] are distributed wireless systems of heterogeneous devices referred to as *sensors* and *actors*. Actors collect and process data produced and sent by the sensors and consequently perform actions. In general, actors are considered devices with high battery life, good processing capabilities and also high transmission power. It is also assumed that they can easily communicate among themselves, e.g., through wireless access points or the like. In the scenario depicted by Akyildiz *et al.*[55] actors are the only mobile nodes in the network, and because of this mobility, sensors need a mechanism to retrieve their location. Based on the sensor data, actors identify an *action area*, for instance, a place where an event has happened and there might be the need of their presence. Based on the event and on the recognized action area, actors coordinate and some of them move to the area. In such a scenario, both *sensor-actor* and *actor-actor* coordination needs to be efficiently handled. To this purpose, Melodia *et al.*[56] introduce a hybrid location management scheme to handle the mobility of actors requiring minimal energy expenditure to the sensors. This solution is proactive given that it is based on update packets sent by mobile actors to sensors to keep them updated about where they are (à la GMRE). The network deployment area is partitioned according to the Voronoi diagram technique,[57] a well known method for partitioning the plane according to predetermined points. The Voronoi tessellation is performed with respect to the actors, and therefore there is an actor in each Voronoi cell (Fig. 33.1). Each actor node is responsible for the nodes located in the cell where it resides and these nodes are made aware to the presence of their actor based on a periodic broadcast from

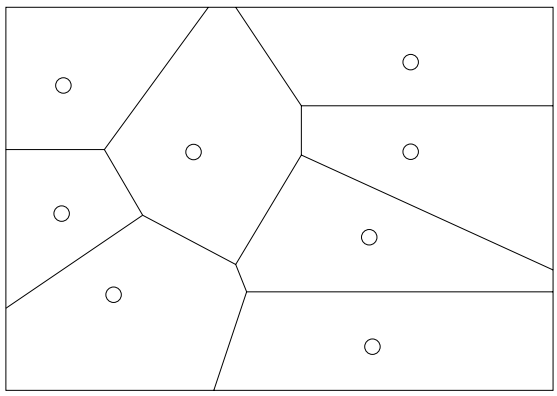

Fig. 33.1. Example of a Voronoi diagram generated by actor nodes (the empty bullets).

it, carrying its position. Location updates are sent by the actors to the sensors locally. By using Kalman filters[58] both at the actors and at the sensor nodes it is also possible to reduce the number of location updates. At each step, by using the filters each actor emulates the prediction procedure performed at the sensor nodes in its cell. It calculates its new position and broadcasts it if and only if a sensor in its cell is not able to estimate it within a maximum error e_{max}. A sensor that does not receive a location update assumes that the predicted position is a good estimation for the position of the actor. Therefore, when a sensor node needs to send some data to the actor node, it either uses the location update it has received (if any) or it uses the estimated position. A mechanism for congestion control is also introduced according to which when an actor a_k detects low reliability (e.g., high packet delays or high packet drop rate), it redirects half of the sensors-actor traffic to another actor. The selection of the alternative actor is performed considering congestion of the other actors and their current position. For further details about the congestion control mechanism and the actor-actor coordination mechanism (for which no location management service is needed), the reader is referred to Ref. 55.

33.3.3. *MT: Object tracking and position retrieving with a multi-tree structure*

Let us consider a WSN scenario in which users know the presence of some objects in the network and may request the location of some of them by querying the sensor nodes. To this purpose, the sensors need to send out a location query in order to retrieve the object position and possibly start a communication with it. For such a scenario Lin *et al.*[59] propose a location management scheme that falls in the flooding-based reactive protocols category. In order to limit the "broadcast storm"[60] which is typical of flooding, the extent of the broadcast of the query packet is contained by defining a multi-tree structure over the flat organization of the network nodes. In order to track the objects, a simple *nearest-sensor tracking* model is adopted. The space is partitioned by using Voronoi diagrams (see above), based on the sensors, i.e., there is a Voronoi cell for each sensor in the network. Each sensor is in charge for the cell containing it. Given a WSN with n sensor nodes, $m \leq n$ are designed as sinks $(\sigma_1, \ldots, \sigma_m)$. A multi-tree construction protocol is then initiated for building m trees each routed at one of the designated sinks. The trees are build with the precise aim of minimizing the cost of location updates. The paper explores two different multi-tree construction protocols: A first one, termed MT-HW (multi-tree construction with high-weight-first property), selects the tree edges based on the weight of the links. In this case the weight is given by the event rate between a node and each of its neighbors. A second algorithm, termed MT-EO (multi-tree construction with edge-overlap-first property), is designed to increase the level of overlap among links of different trees. Both solutions have the property that each of the constructed tree is a shortest-path tree.

Concerning the location update mechanism, every node X maintains a *Detected_List* DL_x that contains an entry for the objects detected by the sensor X itself, and an entry for each neighbor Y that contains all the objects detected by the nodes of the subtrees rooted at Y of some T_{σ_i} (the tree rooted at the sink σ_i). This implies that if a node Z is tracking a certain object o, and Y is an ancestor of Z, then a third node X can find out the position of o by asking Y. It follows that for each sink σ_i, the list DL_{σ_i} contains all the objects tracked by the network. The goal of the location update is to ensure that the *Detected_Lists* of the sensors are updated. The idea behind the location update is that when an object o moves from a sensor X to a sensor Y, for each sink σ_i, the update messages should be sent from X to Y up along T_{σ_i}. This is because the DL of the common ancestor nodes that are two hops away in T_{σ_i} are not affected by this movement. Furthermore, since in a network with m trees the number of neighbors of X may be smaller than m, some of the neighbor nodes are parents in more than one tree and as such they can be updated together. Let us consider the multi-tree structure in Fig. 33.2, where one tree is represented by solid links and the other by dashed ones. The figure shows two trees, T_A and T_B, rooted at the two sink nodes A and B, respectively. The two objects O_1 and O_2 are tracked respectively by the nodes X and F, respectively. If the object O_1 moves from the Voronoi cell of node X to that of node Y, the only nodes that need to be updated are the nodes in the path from X to Y in the trees. Nodes B, D, E, and F do not contain the object O_1 in their DL, and they do not contain it after it moves from X to Y. Node A does not need to update its tables, since object O_1 is still tracked by one of the nodes in the subtree rooted at its descendant node C. The only node that needs to perform the update is node C, which has to move the object O_1 from the subtree rooted in X to the subtree rooted in Y. In other words, the update can be limited to the first common ancestor. For

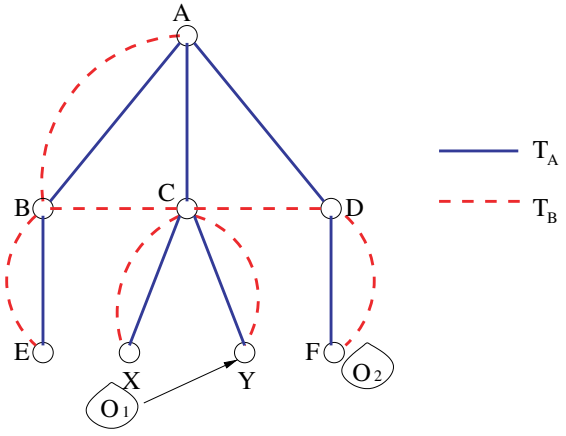

Fig. 33.2. The location update limited to the closest common ancestor in the tree structure.

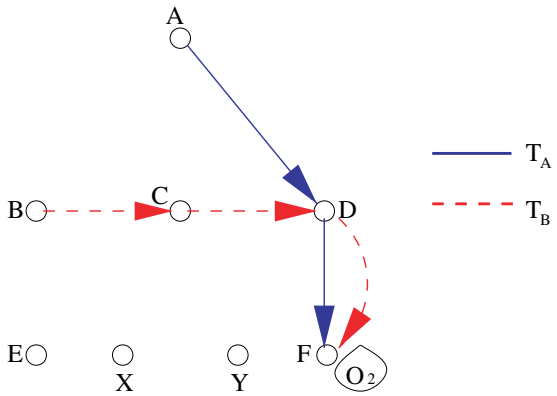

Fig. 33.3. Query graph, and O_2 retrieval.

what concerns the location query mechanism, a user can issue a query from any sensor. When a node X receives a query about an object O, we have two possible cases: (1) O does not appear in any of the entries in DL_X. In this case the query is forwarded to the closest sink. (2) The object O appears in at least one of the entries of DL_X. In this case, given an object O we can define the directed *query graph* $G_O = (V_O, E_O)$ where a directed edge $(u, v) \in E_O$ if and only if $O \in DL_u(v)$. If X forwards the query along the query graph G_O the object O is always reachable. In Fig. 33.3 we show the query graph for the object O_2 of the multi-tree of Fig. 33.2. When the query reaches the sensor that is currently tracking the object, the tracking node replies to the source sensor with the position of the object through the graph. The source sensor is then able to communicate to the object by sending the packets via geo routing.

33.3.4. *ALS: Anchor location service for mobile sinks scenarios*

The WSNs and scenarios considered by the *Anchor Location Service* (ALS)[61] are similar to those of GMRE. In this case, however, multiple sinks are allowed to move throughout the network deployment area. The substantial difference with GMRE is that the ALS protocol is a rendezvous kind of location management: There are nodes (sensors) that keep the location of the closer sink. The *ALS* sensing field is a plane divided into cells of equal size (a grid), similar to that used by GLS[19] for ad hoc networks. The crossing points of the grid are called *grid points*. For each grid point, the nearest sensor node to such grid point is referred to as *grid node*. (For each grid point we have exactly one grid node.) Grid nodes are elected in a distributed manner during the *global grid setup process*. Grid points and grid nodes are depicted in Fig. 33.4. Once grid nodes are selected, each sink selects the nearest grid node as its *sink agent*. The sink agent selects some grid nodes as *anchors* and builds an anchor system, i.e., a set of grid nodes that act as location

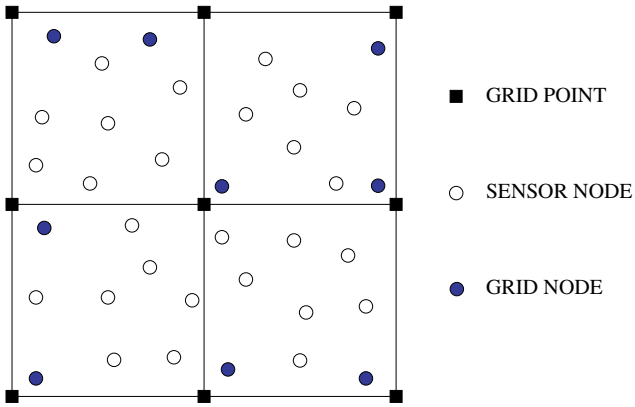

Fig. 33.4. The ALS grid, with grid points and grid nodes.

servers for that sink. Anchors are selected in a distributed manner in such a way that they will be distributed along the entire network and on the border of the void areas (i.e., area where there are no sensors). In general, a sink agent starts the *anchor selection process* sending four *first stage anchor setup packets* toward four orthogonal directions, say, North, East, West, South. All the grid nodes along the path are recruited as anchors. When one of these messages arrives at the border of the deployment area or at that of a void, two *second stage anchor setup messages* are created and routed following the right-hand rule and the left-hand rule[18] with the aim of routing the packets around the void. This mechanism is better explained with an example (Fig. 33.5). The sink agent is located on the bottom border of the grid. It sends out four first stage packets. The packet heading North reaches the node located at point P, in the proximity of a void (dark solid area). The node checks its neighborhood list and realizes that the packet can not be further forwarded North. Therefore, two second stage packets are generated and forwarded East and West according to the the left-hand and to right-hand rules. When a second stage packet reaches the line $x = 5$ in the point P' the packet is divided again into two new second phase messages. In particular, one message keep heading East until it reaches the point P'', while the other is routed North along the line $x = 5$. If some grid node in the network receives some anchor setup messages that it has already received, the node stops propagating that message (as it is the case at points P'' and P''').

Once every sink agent has performed the anchor setup phase, normal communication sensor-to-sink may start. In particular, when a sensor node needs to send out some data to a sink, it sends a request to the nearest grid node (its source agent). The source agent sends out four query packets along four orthogonal directions, as in the anchor system setup process, to find the location of the sink agent. Once the source agent gets back the location of the sink agent from one of the anchor nodes,

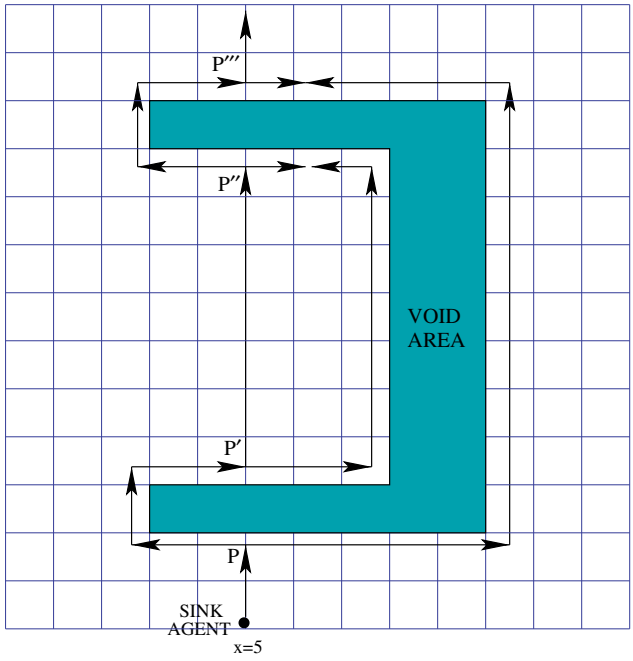

Fig. 33.5. Dealing with voids.

it sends it to the source node. The source node can then send out data packets to the sink agent using any existing geo routing.

33.3.5. *MANY: Manycast communications from a fixed source S*

Zhu and Gupta[62] consider a scenario in which a static source node S wants to perform a manycast communication to a subset D of mobile destination nodes. The authors introduce a solution, termed MANY, which falls into the rendezvous-based category. The deployment area is divided into grids (similar to GLS[19]), and a distributed protocol is introduced for selecting location servers. For each of the equally-sized cell of the grid, a certain number of nodes are elected to be *grid leaders*. The leaders are in charge of maintaining the location information of the destinations in their cell. Destination nodes know their own position and report it to the grid leaders when they switch from a cell to a neighboring one. This update requires the cell size to be fine-tuned. In fact, on one hand, the grid size should be large enough to avoid too many location updates. On the other hand, very large cells may lead to inefficient routing between the grid leaders and the destinations. Furthermore, the few grid leaders in the cell may result to be bottlenecks in the network. Grid leaders are elected in such a way that central nodes in the cell are more likely to be selected. In particular, a node X select itself to be grid leader if

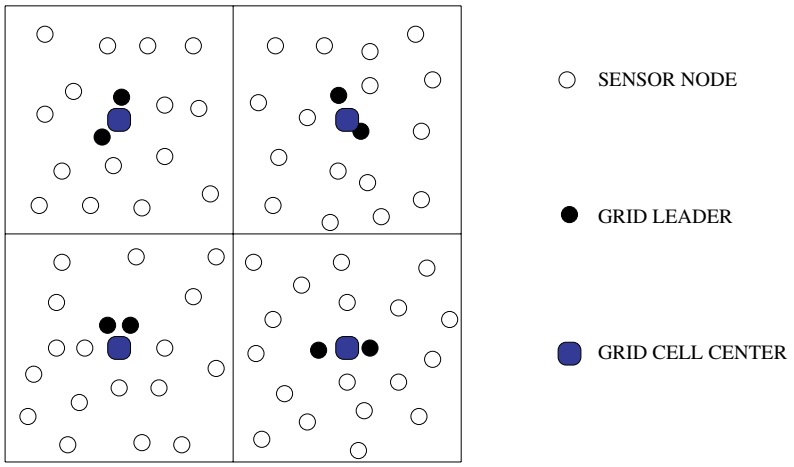

Fig. 33.6. The MANY grid scenario ($l = 2$).

at most $l - 1$ of its neighbors are closer to the geometric center of the cell than X. The value l is chosen based on the network density and it determines the number of grid leaders in each cell. In Fig. 33.6 we show an example of a sensor network organized following the MANY grid structure. In the example, $l = 2$, i.e., the two nodes that are the closest to the centroid of the grid cell are selected as grid leaders. When a destination walks into a new cell, it sends out a location update toward the center of the cell, using geo routing. The destination also includes in the packet its speed and movement direction. Assuming that the destination will move on a straight line for a short period of time, a grid leader may estimate when the destination will leave its cell. Each grid leader uses this information to estimate the number of destinations in its cell. Leaders periodically update the source node S about the number of destinations in their cell. The actual manycast communication happens via a protocol defined in the paper and called *GridTree*. The idea of GridTree is that of optimizing the overall transmission cost by using grid leaders as backbone and routing the data packets through a tree formed based on the grid leaders.

33.3.6. *RNNS: Reverse nearest neighbor and objects-to-objects communications*

Tseng *et al.*[63] investigate the in-network processing of *reverse nearest neighbor* (RNN) queries for moving objects in sensor networks. Let us consider a sensor network in which sensor nodes are employed for detecting the presence of objects in the sensing field. Given a set of objects and a query object, a RNN query retrieves all the objects whose nearest neighbor is the query object. To this purpose, a rendezvous-based location management scheme has been developed. The sensing field is divided

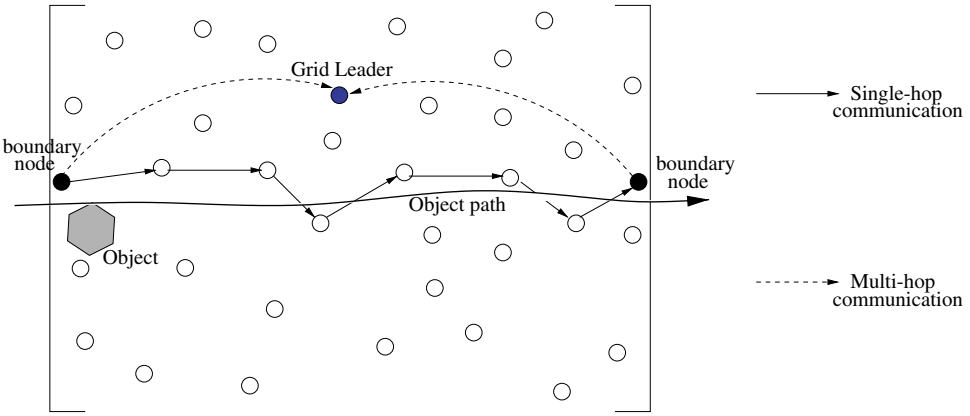

Fig. 33.7. A grid cell, boundary nodes and the grid leader.

into grid cells. The sensor node closest to the centroid of a grid cell is referred to as the *grid leader* and it serves as the location server for the objects tracked in the grid cell. Nodes close to the boundary of the grid cell are called *boundary nodes*: These nodes are responsible of detecting any object entering or leaving the grid cell and of reporting any change to the grid leader. In order to distinguish whether an object is entering or leaving the grid cell, the sensor nodes in the grid cell track the path followed by the object inside the cell, broadcasting a packet every time a sensor node detects the object. The packet carries the ID of the node and all the information collected about it. In so doing, when the object reaches a boundary node and leaves the grid cell, the boundary node has already received information about that object, therefore realizing that the object is leaving the cell. This event is signaled to the grid leader. This mechanism is shown in Fig. 33.7. In the example we can see an object entering and leaving the grid cell. As soon as the object gets into the grid cell, the boundary node signals its presence to the grid leader. (This phase is called *link initialization*.) The intermediate sensor nodes keep tracking the presence of the object, broadcasting information as they detect it. (This phase is called *link construction* and the link chain is referred to as *forwarding link*.) In the picture the forwarding link is represented by the arrow chain from the left boundary node to the right boundary node. When the object reaches the boundary of the grid cell, the boundary node is aware of the fact the object is actually leaving the grid cell, and this is what it communicates to the grid leader. (This phase is called *link reset*.)

In order to perform a reverse nearest neighbor (RNN) query, two phases are defined: A *filtering phase* in which objects that cannot be RNNs are pruned, and a *verification phase*, in which the remaining objects are verified to be RNNs. When a query object needs to inject a query in the network, i.e., when it needs to know which objects are "close" to it, a query packet is geographically routed toward the

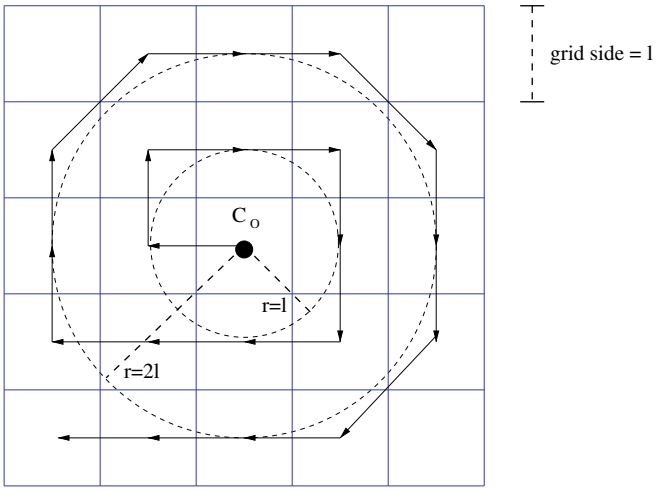

Fig. 33.8. Query forwarding: Spiral routing.

centroid of the actual grid cell C_o, i.e., toward the grid leader of C_o. The grid leader is responsible for performing the RNN query and for returning the query result to the query object. The cells surrounding C_o are sequentially visited in a spiral routing fashion, as shown in Fig. 33.8. In particular, at each round i, the grid cells intersecting the circle centered in C_o and with radius $l * i$, where l is the side of a grid cell, are visited in clockwise order. Each grid leader knows exactly who is the next grid leader to which the message has to be forwarded. When a grid leader receives a query packet, it searches for candidate objects in its own grid cell. In particular, if a candidate object appears in its tables, a packet is sent through the forwarding link until the last sensor node in the chain is encountered. Then, a packet with the location of this sensor is sent back to the grid leader, returning the current position of the inquired object. After the grid node has retrieved the position of all the candidate objects in its grid cell, some objects are discarded via the filtering phase, based on the distance of the objects in the grid cell from the query object. In particular, an object o_i in the grid cell is discarded if $dist(o_i, o_q) > dist(o_i, o_j)$, where o_q is the query object and o_j is another object in the grid cell. If this is the case, o_i cannot be an object such that the query object is the nearest one. After the filtering phase, the query packet is forwarded to the next grid cell. When a cell leader has no more neighbor cells to which to forward the packet, a packet containing the set of the candidate objects is sent back to the grid leader of the cell C_o for the verification phase. During the verification phase, the node analyzes each candidate object o_c. In particular, an object o_c is discarded if, considering a circle of radius $dist(o_c, o_q)$, there exists at least one candidate object inside that circle. The objects that survive the verification phase are RNN objects and such list is sent to the query object.

33.3.7. *SERV: Service localization in static WSNs*

Let us consider a static WSN scenario in which some of the nodes are capable to provide some services, such as providing the locations of other nodes or resources in the network, to other sensor nodes. In order to use one of these services, a sensor node needs to retrieve the location of the node that offers it, i.e., a services localization scheme is required. This is the problem tackled by Nidito *et al.*[64] They propose a rendezvous-based service management scheme, in which each node providing one or more services (also referred as *server nodes*) are mapped to some special nodes (*server locators*) that know the location of server nodes and the services they provide. Each server node registers its services to some server locators. The location of the locator is obtained employing some *Geographic Hash Tables* (GHT).[31] This hash tables perform two basic operations, namely, *put* and *get*. The hash function takes as input a *key* and return a pair of coordinates (x, y). The actual GHT algorithm employed in this solution is called Q-NiGHT,[65] which includes mechanisms for providing some QoS, fault-tolerance and a better balancing of the service requests among the nodes. In particular, in Q-NiGHT the information is stored at the Q nodes that are closest to the point returned by the GHT. Therefore, the information can survive $Q - 1$ faults. Furthermore, Q-NiGHT uses a hash function that considers the distribution of the sensors on the sensing area, and distribute data according to such distribution. Therefore, dense regions store a higher amount of data, while sparse regions store only few entries. This approach guarantees a better balancing of the request traffic among the sensor nodes. The algorithm works as follows. At the beginning each server node registers its service(s) to the server locators using the hashing function provided by Q-NiGHT. In particular, each server hashes the name of each of its services obtaining a pair of coordinates (x, y). Then, the node sends a PUT packet to each of the server locators routing the packet via geo routing. In this way, servers make server locators aware of their position. When a sensor node needs a service s_i, it hashes the service name obtaining the spatial location (x, y) of s_i. Then, the node sends a GET packet to the server locator in order to retrieve the position of the server of the service s_i. The node also caches the location for that service for future uses and maintains this information for a period τ. If the same service is offered by several server nodes, the server locator chooses one of the servers according to some policy (for instance in a round robin fashion) with the aim of balancing the query load among the different servers. In order to obtain a certain level of fault-tolerance, server nodes keep providing their position periodically (every τ seconds) to the server locators. The server locator removes the service from its tables if no PUT packet has been received for 2τ seconds. If a sensor node contacts a dead server using the cached information, the previous neighbor of the server responds to the querying sensor with an error packet. Then, the node remove the information from the cache and perform a GET to the service

Table 33.1. A bird's eye view of the characteristics of location management protocols for WSNs.

Protocol	Type	Who moves	Who queries	Who is located	Communication flow	Location awareness
GMRE[50]	proactive	sink	no query	sink	sensors to sink	sensors and sinks
ACT[56]	proactive	actors	no query	actors	sensors to actors actors to actors	sensors and actors
MT[59]	reactive	objects	users through any sensor node	objects	sensors to objects	sensors, sinks and objects
ALS[61]	rendezvous	sinks	sensors	sinks	sensors to sinks	sensors and sinks
MANY[62]	rendezvous	a set D of destination nodes	fixed source S	destination nodes	node S to a set $D' \subseteq D$ of destinations	sensors and destinations
RNNS[63]	rendezvous	objects	objects	objects	objects to objects	sensors
SERV[64]	rendezvous	none	sensors	servers	sensors to servers	servers

locator. Once the querying node obtain the server location, it can start using the service routing packets via some geographic routing algorithm.

33.3.8. *Table summary*

The main characteristics of the protocols described in this section are summed up in Table 33.1.

33.4. Conclusions

In this chapter we have discussed the importance of mechanisms for location management in multi-hop networks and especially in wireless sensor networks. We have briefly reviews location management schemes proposed for mobile ad hoc networks, with an emphasis on those that enable geo routing. We have then described recent solutions for wireless sensor networking, where location management is a crucial tools for the requirements of very specific and very different applications. Our survey of the proposed methods reveal that for both ad hoc and WSNs research is still needed about the impact that the location management schemes have on the performance of geo routing, and in general of all location aware or location based application. Further insights are also necessary about the interaction of mobility and location management. These insights are expected to provide new and more efficient design guideline for location management as well as for more efficient WSN protocol stacks.

Acknowledgments

This work was supported in part by the NSF grant #CNS-0737720. The first author is also thankful to Michael J. Roncarati for useful comments.

References

1. S. Basagni, M. Conti, S. Giordano and I. Stojmenovic, (eds.), *Mobile Ad Hoc Networking*, IEEE Press and John Wiley & Sons, Inc., Piscataway, NJ and Hoboken, NJ, (April 2004).
2. O. K. Tonguz and G. Ferrari, *Ad Hoc Wireless Networks*, John Wiley & Son., LTD, West Sussex, UK (2006).
3. I. Stojmenovic, Position based routing in ad hoc networks, *IEEE Communication Magazine*, **40**(7), 128–134 (July 2002).
4. S. Giordano and I. Stojmenovic, Position-based routing algorithms for ad hoc networks: A taxonomy, (eds.) X. Cheng, X. Huang and D.-Z. Du, *Ad Hoc Wireless Networking*, *Network Theory and Applications*, chapter 4, 103–136, Springer, New York, NY (2004).
5. E. M. Belding-Royer, Routing approaches in mobile ad hoc networks, (eds.) S. Basagni, M. Conti, S. Giordano and I. Stojmenovic, *Mobile Ad Hoc Networking*, chapter 10, 275–300. IEEE Press and John Wiley & Sons, Inc., Piscataway, NJ and Hoboken, NJ (April 2004).

6. M. Zorzi and R. R. Rao, Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance, *IEEE Transactions on Mobile Computing*, **2**(4), 337–348 (October–December 2003).
7. M. Zorzi and R. R. Rao, Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Energy and latency performance, *IEEE Transactions on Mobile Computing*, **2**(4), 349–365 (October–December 2003).
8. P. Casari, M. Nati, C. Petrioli and M. Zorzi, Efficient non-planar routing around dead ends in sparse topologies using random forwarding, *Proceedings of the IEEE International Conference on Communications, ICC 2007*, Glasgow, Scotland (June 24–28, 2007).
9. N. Samama, *Global Positioning: Technologies and Performance*, Wiley Survival Guides in Engineering and Science, Wiley and Sons, Inc., Hoboken, NJ (2008).
10. A. Savvides and M. B. Srivastava, Location discovery, (eds.) S. Basagni, M. Conti, S. Giordano and I. Stojmenovic, *Mobile Ad Hoc Networking*, chapter 8, 231–254. IEEE Press and John Wiley and Sons, Inc., Piscataway, NJ and New York, NY (April 2004).
11. M. Battelli and S. Basagni, Localization for wireless sensor networks: Protocols and perspectives, *Proceedings of IEEE CCECE 2007*, 1074–1077, Vancouver, Canada (April 22–26, 2007).
12. V. Chandrasekhar, W. K. G. Seah, Y. S. Choo and H. Ee. Localization in underwater sensor networks — Survey and challenges, *Proceedings of the 1st ACM International Workshop on UnderWater Networks, WuWNet 2006*, Los Angeles, CA (September 25, 2006).
13. S. Basagni, I. Chlamtac, V. R. Syrotiuk and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM) *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom 98*, 76–84, Dallas, TX (October 25–30, 1998).
14. Y.-B. Ko and N. H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, *ACM/Kluwer Wireless Networks*, **6**(4), 307–321 (July 2000).
15. H. Hartenstein and K. P. Laberteaux, Topics in ad hoc and sensor networks — A tutorial survey on vehicular ad hoc networks, *IEEE Communications Magazine*, **46**(6), 164–171 (June 6 2008).
16. P. Bose, P. Morin, I. Stojmenovic and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks, *Proceedings of the* 3*rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, DIAL M '99*, 48–55, Seattle, WA (1999).
17. P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, *ACM/Kluwer Wireless Networks*, **7**(6), 609–616 (November 2001).
18. B. K. and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks, *Proceedings of the* 6*th ACM Annual International Conference on Mobile Computing and Networking, MobiCom 2000*, 243–254, Boston, Massachusetts, (2000), ACM. ISBN 1-58113-197-6. doi: http://doi.acm.org/10.1145/345910.345953.
19. J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger and R. Morris. A scalable location service for geographic ad hoc networks, *Proceedings of the* 6*th ACM Annual International Conference on Mobile Computing and Networking, MobiCom 2000*, 120–130, Boston, MA (August 6–11, 2000).
20. Y. Xue, B. Li and K. Nahrstedt, A scalable location management scheme in mobile ad-hoc networks, *Proceedings of the* 26*th Annual IEEE Conference on Local Computer Networks, LCN 2001*, 102–111, Tampa, FL (November 14–16, 2001).

21. J. Sucec and I. Marsic, Location management for hierarchically organized mobile ad hoc networks, *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC 2002*, **2**, 603–607, Orlando, FL (March 17–21, 2002).
22. H.-H. Wang and S.-D. Wang, The modified grid location service for mobile ad-hoc networks, *Proceedings of the 2nd International Conference on Grid and Pervasive Computing, GPC 2007*, **4459**, *LNCS*, 334–347, Paris, France (May 2–4, 2007).
23. S.-C. M. Woo and S. Singh, Scalable routing protocol for ad hoc networks, *ACM/Kluwer Wireless Networks*, **7**(5), 513–529 (September 2001) ISSN 1022-0038, doi: http://dx.doi.org/10.1023/A:1016726711167.
24. C. Perkins and I. P. Mobile, *IEEE Communications Magazine*, **35**(5), 84–99 (May 1997).
25. C. T. Cheng, H. L. Lamberg, S. J. Philip, E. van der Berg and T. Zhang. SLALoM: A scalable location management scheme for large mobile ad-hoc networks. In *Proceedings of the 2002 IEEE Wireless Communication and Networking Conference, WCNC 2002*, **2**, 574–578, Orlando, FL (March 17–21, 2002).
26. S. Dolev, L. Lahiani, N. Lynch and T. Nolte. Self-stabilizing mobile node location management and message routing, *Proceedings of the 7th International Symposium on Self-stabilizing Systems, SSS 2005*, LNCS 3764, 96–112, Barcelona, Spain (October 26–27, 2005).
27. S. Sivavakeesar and G. Pavlou, Scalable location services for hierarchically organized mobile ad hoc networks, *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2005*, 217–228, Urbana-Champaign, IL (May 25–28, 2005).
28. S. Basagni, M. Mastrogiovanni, A. Panconesi and C. Petrioli, Localized protocols for ad hoc clustering and backbone formation: A performance comparison, *IEEE Transactions on Parallel and Distributed Systems, Special Issue on Localized Communication and Topology Protocols for Ad Hoc Networks* (*S. Olariu, D. Simplot-Ryl, and I. Stojmenovic, editors*). **17**(4), 292–306 (April 2006).
29. S. J. Philip, J. Ghosh and C. Qiao, Performance evaluation of a hierarchical location management protocol for ad hoc networks, *Computer Communications, Special issue on Performance Issues of Wireless LANs, PANs, and Ad Hoc Networks*, **8**(10), 1110–1122 (2005).
30. R. Flury and R. Wattenhofer, MLS: An efficient location service for mobile ad hoc networks, *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2006*, 226–237, Florence, Italy (May 22–25, 2006). ISBN 1-59593-368-9. doi: http://doi.acm.org/10.1145/1132905.1132931.
31. S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin and F. Yu, Data-centric storage in sensornets with GHT, a geographic hash table, *ACM/Kluwer Mobile Networks and Applications*, **8**(4), 427–442 (August 2003).
32. Z. J. Haas and B. Liang, Ad hoc mobility management with uniform quorum systems, *IEEE/ACM Transactions on Networking*, **7**(2), 228–240 (April 1999).
33. T. Camp, J. Boleng and L. Wilcox, Location information services in mobile ad hoc networks, *Proceedings of the IEEE International Conference on Communications, ICC 2002*, **5**, 3318–3324, New York, NY (April 28–May 2, 2002).
34. S. M. Das, H. Pucha and Y. C. Hu, Performance comparison of scalable location services for geographic ad hoc routing, *Proceeding of the 24th IEEE Infocom 2005*, **2**, 1228–1239, Miami, FL (March 13–17, 2005).

35. I. Stojmenovic, Location updates for efficient routing in ad hoc wireless networks, (ed.) I. Stojmenovic, *Handbook of Wireless Networks and Mobile Computing*, chapter 21, 451–471. John Wiley and Sons, Inc., Hoboken, NJ (April 2002).
36. M. Mauve, A. Widmer and H. Hartenstein, A survey on position-based routing in mobile ad hoc networks, *IEEE Network*, **15**(6), 30–39 (November–December, 2001).
37. R. Friedman and G. Kliot. Location services in wireless ad hoc and hybrid networks: A survey. Technical Report CS-2006-10, Department of Computer Science, Technion, Haifa, Israel (April 24, 2006).
38. A. Howard, M. J. Matarić and G. S. Sukhatme, An incremental deployment algorithm for mobile sensor networks, *Kluwer Autonomous Robots, Special Issue on Intelligent Embedded Systems* (*G. S. Sukhatme, editor*). **13**(2), 113–126 (September 2002).
39. R. C. Shah, S. Roy, S. Jain and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks, *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, SNPA 2003*, 30–41, Anchorage, AK (May 11, 2003).
40. R. Rao and G. Kesidis, Purposeful mobility for relaying and surveillance in mobile ad hoc sensor networks, *IEEE Transactions on Mobile Computing*, **3**(3), 225–232 (July–September, 2004).
41. G. Wang, G. Cao and T. F. La Porta, Movement-assisted sensor deployment. *Proceedings of IEEE INFOCOM 2004*, **4**, 2469–2479, Hong Kong (March 7–11, 2004).
42. G. Wang, G. Cao and T. F. La Porta Proxy-based sensor deployment for mobile sensor networks, *Proceedings of the First IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2004*, 493–502, Fort Lauderdale, FL (October 25–27, 2004).
43. A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava and D. Estrin, Intelligent fluid infrastructure for embedded networks, *Proceedings of the* 2*nd ACM/SIGMOBILE International Conference on Mobile Systems, Applications, and Services, MobySys 2004*, 111–124, Boston, MA (June 6–9, 2004).
44. W. Wang, V. Srinivasan and K.-C. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks, *Proceedings of the* 11*th Annual ACM/SIGMOBILE International Conference on Mobile Computing and Networking, MobiCom 2005*, 270–283, Cologne, Germany (August 28–September 2, 2005).
45. S. R. Gandham, M. Dawande, R. Prakash and S. Venkatesan, Energy efficient schemes for wireless sensor networks with multiple mobile base stations, *Proceedings of IEEE Globecom 2003*, **1**, 377–381, San Francisco, CA (December 1–5, 2003).
46. I. Papadimitriou and L. Georgiadis, Maximum lifetime routing to mobile sink in wireless sensor networks, *Proceedings of the 2005 International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2005*, Split, Croatia (September 15–17, 2005).
47. J. Luo and J.-P. Hubaux, Joint mobility and routing for lifetime elongation in wireless sensor networks, *Proceedings of IEEE Infocom 2005*, **3**, 1735–1746, Miami, FL (March 13–17, 2005).
48. A. P. Azad and A. Chockalingam, Mobile base station placement and energy aware routing in wireless sensor networks, *Proceeding of the IEEE Wireless Communications and Networking Conference, WCNC 2006*, **1**, 264–269, Las Vegas, NV (April 3–6, 2006).

49. S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli and M. Z. Wang, A new MILP formulation and distributed protocols for wireless sensor networks lifetime maximization, *Proceedings of the IEEE International Conference on Communications, ICC 2006*, **8**, 3517–3524, Istanbul, Turkey (June 11–15, 2006).
50. S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli and Z. M. Wang, Controlled sink mobility for prolonging wireless sensor networks lifetime, *ACM/Springer Wireless Networks*. (2008). Published online on February 7, 2007.
51. S. Basagni, A. Carosi and C. Petrioli, Mobility in wireless sensor networks. In ed. A. Boukerche, *Algorithms and Protocols for Ad Hoc and Sensor Networks*, chapter 10, 267–305, John Wiley & Sons, Inc., Hoboken, NJ (2008).
52. S. Basagni, C. Carosi, C. Petrioli and C. A. Phillips, Moving multiple sinks through wireless sensor networks for lifetime maximization, *Proceedings of the* 5*th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2008*, Atlanta, GA (September 29–October 2, 2008).
53. E. Ekici, Y. Gu and D. Bozdag, Mobility-based communications in wireless sensor networks, *IEEE Communications Magazine*, **44**(7), 56–62 (July 2006).
54. S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli and Z. M. Wang, Protocols and model for sink mobility in wireless sensor networks, *ACM Mobile Computing and Communication Review, MC*^{2}*R*, **10**(4), 28–30 (October 2006).
55. I. Akyildiz and I. Kasimoglu, Wireless sensor and actor networks: Research challenges, *Ad Hoc Networks Journal*, **2**(4), 351–367 (October 2004).
56. T. Melodia, D. Pompili and I. F. Akyldiz, A communication architecture for mobile wireless sensor and actor networks, *Proceedings of the* 3*rd Annual EEE International Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2006*, 109–118, Reston, VA, USA (September 25–28, 2006).
57. F. Aurenhammer, Voronoi diagrams — a survey of a fundamental geometric data structure, *ACM Computing Surveys*, **23**(3), 345–405 (1991).
58. R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, Inc., Hoboken, NJ (1996), 3rd edition.
59. C. Y. Lin, Y. C. Tseng and T. H. Lai, Message-efficient in-network location management in a multi-sink wireless sensor network, *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, SUTC 2006*, 496–505, Taichung, Taiwan (June 5–7, 2006).
60. Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen and J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, *ACM/Kluwer Wireless Networks*, **8**(2/3), 153–167 (March–May, 2002).
61. R. Zhang, H. Zhao and M. A. Labrador, The anchor location service (ALS) protocol for large-scale wireless sensor networks, *Proceedings of the* 1*st International Conference on Integrated Internet Ad Hoc and Sensor Networks, InterSense 2006*, 18, Nice, France (May, 30–31, 2006).
62. X. Zhu and H. Gupta, Fault-tolerant manycast to mobile destinations in sensor networks, *Proceedings of the IEEE International Conference on Communications, ICC 2007*, 3596–3603, Glasgow, Scotland (June 24–28, 2007).
63. Y. C. Tseng, C. C. Chen, C. Lee and Y. K. Huang. Incremental in-network RNN search in wireless sensor networks, *Proceedings of the 2007 International Conference on Parallel Processing Workshops, ICPPW 2007*, 64, Xi'an, China (September 10–14, 2007).
64. F. Nidito, M. Battelli and S. Basagni, Fault-tolerant and load balancing localization of services in wireless sensor networks. In *Proceedings of the* 67*th Semi-Annual IEEE*

Vehicular Technology Conference, VTC 2007 Fall, 382–386, Baltimore, MD (September 30–October 3, 2007).

65. M. Albano, S. Chessa, F. Nidito and S. Pelagatti, Q-NiGHT: Adding QoS to data centric storage in non-uniform sensor networks, *Proceedings of the 8th International Conference on Mobile Data Management, MDM 2007)*, 166–173, Mannheim, Germany (May 7–11, 2007).