An Energy Efficient Algorithm for Routing in Wireless Sensor Networks

Mr. Vijay B. Mohite[1], Prof. N. D. Kale[2]
[1, 2]Department of Computer Engineering
[1,2] PVPIT, Bavdhan, Pune-21

Abstract- *In wireless sensor networks (WSN), building efficient and scalable protocols is a very challenging task due to the limited resources available and dynamics. Geographic routing protocols, that take information of each node location, are very valuable for wireless sensor networks. The state required to be maintained should be minimum and low overhead, addition to their fast response to dynamics. The routing protocols are in charge of discovering and maintaining the routes in the network. The appropriateness of a particular routing protocol mainly depends on the capabilities of the nodes and on the application requirements. An overview of geographic routing and load balancing protocol is presented in this paper. In this paper we mainly focus on ALBA-R, a protocol for convergecasting in wireless sensor networks. ALBA-R is the cross-layer integration of geographic routing with contention-based MAC for specific relay selection and load balancing (ALBA), also a mechanism to detect and route around connectivity holes (Rainbow). ALBA and Rainbow together solves the problem of routing around a dead end without overhead-intensive techniques such as graph planarization and face routing.*

Keywords- WSN, MAC, Cross-Layer Routing, Connectivity Holes, Geographic Routing, XLP

I. INTRODUCTION

Distributed sensing environment and seamless wireless data gathering are main ingredients of several monitoring applications implemented through the dissemination of wireless sensor networks. There are various routing techniques that have been proposed so far and several of these have been already explicitly implemented and are working good. The sensor nodes perform their data collection duties with the Unattended, and the corresponding packets are then transfer to the sink using multihop wireless routes (WSN routing or convergecasting). The main research on protocol design for WSNs has mainly focused on MAC and routing solutions. Various mobile ad-hoc networks (MANETs) are infrastructure free networks of mobile nodes that communicate with each other in wireless mode. The applications of such networks have been in disaster relief operations, conferencing, military surveillance, and environment capturing. Several ad hoc routing algorithms at present that utilize topology information to make routing decisions at each node in the network. An important class of protocols is mentioned by geographic or location-based routing scheme are present in which path is chosen via greedy approach it provides toward the sink. Due to almost stateless, distributed and localized, geographic routing requires little computation and storage resources at the nodes and is therefore very attractive for WSN some applications.

The wireless sensor network is built of nodes from a few to several hundreds or even more, where each node is connected to one (or sometimes several) sensors present in the network. Cross layer mechanism can be used to make the optimal modulation to improve the transmission performance.

Sensors are acting as a gateway between sensor nodes and the end user; they typically forward data from the WSN on to a server.

Routers are one among the important components of WSNs, which are designed to compute and calculate and distribute the routing tables. Georouting routing (also called geographic or position-based routing) is a routing principle that relies on geographic position information and it is based on the idea that the source sends a message to the geographic location of the destination instead of using the network address. In geographic routing, each node can determine its own location and source node is aware of the location of the destination node. Thus the information can be routed to destination without prior route discovery or knowledge of the network topology.

Greedy forwarding technique adopted by most single path strategies tries to bring the message closer to the sink in each step using only local information. Thus every node forward packets to the neighbor which is most suitable from a local point of view, node which minimizes the distance to the destination is the most suitable neighbor. Whenever there is no neighbor closer to the destination, greedy forwarding can lead into a dead end.

Connectivity holes are inherently related to the way greedy forwarding approach works. In a fully connected topology, there may exist nodes which are known as dead ends that have no neighbors that provide packet transmission toward the destination. SO the dead ends are unable to forward the packets they generate or receive. So these packets will never reach their destination and will eventually be discarded. There are several geographic routing schemes fail to fully address important design challenges includes,

i) Routing around connectivity holes
ii) Resilience to localization errors
iii) Efficient relay selection

So the main objective is of this paper is to provide load balancing among the nodes and to overcome the packet loss and dead-ends. ALBA mechanism performs load balancing. Here the splitting of packets is based on number of inputs.

II. LITRATURE SURVEY

Ding, Sivalingam, Kashyapa [1] considered the problem of finding a route from a sensor to the single sink in a wireless sensor network. Consisting a reactive route discovery strategy, sink floods the network and sets the routes. Here the difference is that each sensor does not memorize the whole route, but instead it only memorizes its hop count distance to the destination. When a packet is sent toward the destination, any neighbor at one less hop distance can forwards it, Instead of reporting back to the first node that sent task assignment packet to it.

Geographic routing mainly considers transmitting a packet in the direction of its particular sink by giving maximum per-hop advancement. The geographic routing over planarized wireless sensor networks is obtained by employing greedy routing as possible and the resorting to planar routing only when required, to get over around connectivity holes. The spanner graph of the network topology needs to be built and this incurs no negligible overhead. Therefore planar routing may then require the exploration of large spanners before being able to switch back to the more efficient greedy forwarding approach, thus importing higher latencies [2].

Jean-Yves and Le Boude etal. [5] explain an inclusive review on drawbacks on geographical routing where greedy forwarding approach is used, that is destination distance is too long means packet gets discarded because greedy forwarding algorithm works based on shortest path first and tells about various location based routing protocols and their drawbacks are like location based routing is difficult when there are holes in the network topology and nodes are mobile or frequently disconnected to save battery and tells about terminate routing protocols.

Another different approach for handling dead ends is based on embedding the network topology into coordinate spaces that decrease the probability of connectivity holes. Greedy forwarding is mainly performed over the virtual coordinate's space and this reduces the appearance of dead ends, but it does not removing these ends. Various topology wrapping schemes are mainly depends on iteratively updating the coordinates of each node based on the coordinates of its neighbors so that greedy paths are more exist. These approaches are known as "geographic routing without location information," as they do not require accurate initial position estimates [3], [4].

Geographic Routing Techniques

Geographic routing is a technique to deliver packets to anode in a network over multiple hops by means of position information. The routing decisions are not based on network addresses and routing tables instead, packets are routed towards a destination location. With knowledge of the neighbor's location, each node can select the next hop neighbor that is closer to the destination, and thus advance towards the destination in each step. The fact that neither routing tables nor route discovery activities are necessary makes geographic routing attractive for dynamic networks such as wireless ad hoc and sensor networks. In such networks, acquiring and maintaining routing information is costly as it involves additional message transmissions that require energy and bandwidth and frequent updates in mobile and dynamic scenarios.

Geographic routing algorithms use position information for making message forwarding decisions. Not similar to topological routing algorithms they do not need to exchange and maintain routing information and work nearly stateless. This makes geographic routing attractive for wireless adhoc and sensor networks.

Planar Graph

Planar graph routing, which guides the packet around the local minimum and guarantees delivery, required that a planar sub graph of the network graph can be constructed in Figure 4. Therefore, recovery methods have been developed, the most prominent of which are based on planar graph routing, where the message is guided around the local minimum by traversing the edges of a planar sub graph of the network communication graph[14]. Planar graph routing

techniques can provide delivery guarantees under certain assumption. Altogether, greedy forwarding in combination with a recovery can be considered as state-of-the-art technique in geographic routing.

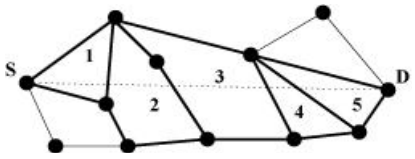


Fig planar graph

Greedy Forwarding

Most geographic routing algorithms use a greedy strategy that tries to approach the destination in each step [13], e.g. by selecting the neighbor closest to the destination as a next hop depicted in Figure 3. However, greedy forwarding fails in local minimum situations, i.e. when reaching a node that is closer to the destination than all its neighbors. A widely adopted approach to solve this situation is planar graph routing.

A simple greedy forwarding by minimizing the distance to the destination location in each step cannot guarantee message delivery. Nodes usually have a limited transmission range and thus there are situations where no neighbor is closer to the destination than the node currently holding the message. Greedy algorithms cannot resolve such dead-end or local minimum situation.

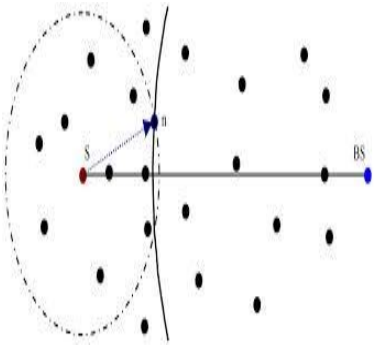


Fig. Greedy forwarding

The current geographic routing schemes fail to fully address some important design challenges, including

i) Routing around connectivity holes
ii) Resilience to localization errors
iii) Efficient relay selection.

III. IMPLEMENTATION

To overcome the current geographic routing problems we proposed ALBA-R, a protocol for converge casting in wireless sensor networks. ALBA-R is the cross-layer integration of geographic routing with contention-based MAC for relay selection and load balancing using adaptive load balancing algorithm (ALBA), also a mechanism to detect and route around connectivity holes (Rainbow).

Load-balancing is needed to effectively use available resources and keep the nodes energy consumption balanced by equally distributing the load to all nodes [6]. The problem is to route data packets avoiding congested path so as to balance traffic load over network and lower end-to-end delay. Distributing the load within the network has two advantages. Firstly the resource of the network is fully utilized through distributing network load. An efficient load-balancing routing protocol is able to improve packet delivery rate and network throughput. Secondly the energy consumption is balanced by equally distributed load due to which the network lifetime could be increases. A dynamic parameter less load-balancing geo routing protocols was proposed. The node holding the packet for delivery compares costs of sending the packet to all available neighbors that are closer to destination and not fully loaded, against the progress made. The cost is then increasing linearly with the consumed bandwidth.

Adaptive Load balancing algorithm

ALBA is a greedy forwarding protocol for Wireless sensor networks. It is designed to take congestion and traffic load balancing into consideration. All the eligible relays of a node compute two values first the Geographic Priority Index (GPI), i.e., the index of the region the node would belong to in the GeRaF (Geographic Random Forwarding) [6] framework, second one is the Queue Priority Index (QPI), which is a measure of forwarding effectiveness as perceived by the relay.

ALBA, is a cross layer solution for convergecasting in Wireless sensor networks that integrates awake/asleep schedules, MAC, routing, traffic load balancing, and back-to-back packet transmissions. As nodes alternate between awake/asleep modes according to independent wake-up schedules with fixed duty cycle d So Packet forwarding is implemented by having the sender polling for availability its awake neighbors by broadcasting an RTS packet for jointly performing channel access and communicating relevant routing information (cross-layer approach). Available neighboring nodes respond with clear-to-send (CTS) packet carrying information through which the sender can choose the best relay so the relay selection is performed by preferring neighbors offering ―good performance‖ in forwarding packets. Positive geographic advancement toward the sink (the main relay selection criterion in many previous solutions) is used to discriminate among relays that have the same

forwarding performance. Every prospective relay is characterized by two parameters: the queue priority index (QPI), and the geographic priority index (GPI).

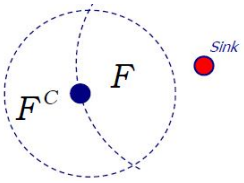
Cross layer protocol: XLP

XLP features the cross-layer integration of geographic routing addition with contention-based MAC for relay selection and load balancing (ALBA), and also a mechanism which detects and route around connectivity holes (Rainbow). ALBA and Rainbow together solve the problem of routing around dead ends without overhead-intensive techniques such as graph planarization and face routing methods.

The Rainbow mechanism allows XLP to efficiently route packets out of and around dead ends. Rainbow is resilient to localization errors and to channel propagation impairments. It does not need the network topology to be planar, unlike previous routing protocols. It is, therefore, more general than face routing-based solutions and is able to guarantee packet delivery in realistic deployments.

XLP: Cross layer protocol algorithm
Steps are as follows

1. XLP assigns to relays nodes based on QPI index and GPI
2. QPI measures the correctness of node for forwarding a packet
3. QPI=d(Q + NB)/Me-1
   Where Q- Queue occupancy
   NB- Number of expected packets of bursts that can be sent by the relay node
4. The GPI value is based only on geographical coordinates. The closer a node to sink the higher will be the GPI
5. Relay Selection is based on QPI values. If QPI value is same in that case GPI value can be used.
6. Rainbow mechanism
7. Each Node is assigned with colors and try to catch the yellow brick route.
8. Initially all nodes are yellow- They look forward for relays in F
   Where F- Positive advancement in the direction of sink
9. No relays in F-The node changes to red and it looks for (yellow, red) relays in Fc
   Where- Fc Negative advancement related to sink
10. No relays in Fc- Node changes to blue and it looks for (red, blue) relays in F.



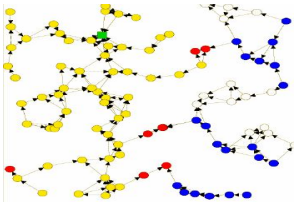


Fig- XLP coloring scheme

IV. RESULTS

This application is a simulation of a wireless sensor network. Such a network is used to detect and report certain events across an expanse of a remote area - e.g., a battlefield sensor network that detects and reports troop movements. The idea behind this network is that it can be deployed simply by scattering sensor units across the area, e.g. by dropping them out of an airplane; the sensors should automatically activate, self-configure as a wireless network with a mesh topology, and determine how to send communications packets toward a data collector (e.g., a satellite uplink.) Thus, one important feature of such a network is that collected data packets are always traveling toward the data collector, and the network can therefore be modeled as a directed graph (and every two connected nodes can be identified as "upstream" and "downstream.")

A primary challenge of such a network is that all of the sensors operate on a finite energy supply, in the form of a battery. (These batteries can be rechargeable, e.g. by embedded solar panels, but the sensors still have a finite maximum power store.) Any node that loses power drops out of the communications network, and may end up partitioning the network (severing the communications link from upstream sensors toward the data collector.) Thus, the maximum useful lifetime of the network, at worst case, is the minimum lifetime of any sensor.

One potential improvement is in making packet-routing decisions that extend the life of the network. The concept is that any node may be connected to more than one downstream node, and it may be more desirable to use one than the other. For instance, if several nodes are connected to downstream bottleneck node that is rapidly exhausted, the lifetime can be extended by reducing the traffic going through it (i.e., upstream nodes preferentially use alternative downstream nodes.)

Of course, given that data generation rates are unpredictable - since it is not known in advance whether any sensor will detect little or much activity - the routing process must be dynamic. Therefore, it is useful for each node to recalculate its routing decisions periodically, based on the energy reserves of each downstream node. An algorithm for doing so was developed by Jae-Hwan Chang and Leandros Tassiulas [8]

This application is a simulation of the wireless sensor network described hereinabove. The network may be deployed based on a wide range of parameters: network size (number of nodes), communications distance, energy costs for transmitting and receiving packets, etc. The network can then be used to simulate the detection of vectors traveling across the sensor network field. In this simulation, when a vector trips the sensor of a network node, the node generates a data packet and sends it to a downstream network node. The packets are routed appropriately until they reach a sensor within the "uplink zone" (the right side of the map, designated with a striped pattern.) Each node also simulates an energy store, which is depleted by sending receiving packets, and by detecting vectors. Since the nodes have finite energy, they will eventually power down and drop out of the communications network, causing network failure.

Use

This simulation consists of two stages: deploying the network and running simulations.

Before deploying the network, the properties of the network should be set using the configuration sliders.

(Note: The properties of the network are set at the time the network is created, so changes to the network configuration and routing parameters will not be effective until a new network is deployed.) The network configuration properties are grouped into two categories:

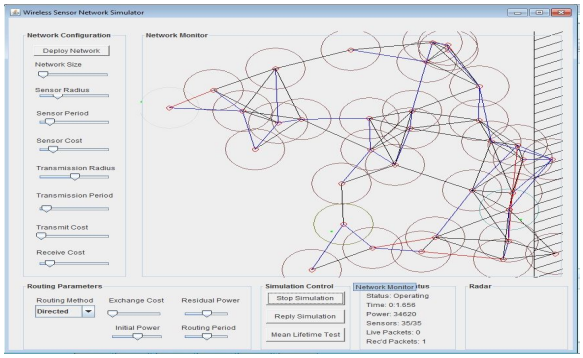


Fig: Main Screen Wireless Network Simulator

1. Network Configuration:

These factors determine the hardware properties of the network. The following variables can be configured:

i. Network Size:

The number of nodes in the network. If set to a high value, the network will have several hundred nodes; and since this will hugely increase the density of the network and the number of network connections, this may bog down the simulation. If a large network is desired, it is recommended to reduce the Transmission Radius.)

ii. Sensor Radius:

The proximity range of the sensors in the network.

iii. Sensor Period:

The delay period between sensor detection events. If set to a low value, a network sensor will fire rapidly as a vector enters its sensor radius (thereby consuming a lot of energy.) If set to a high value, the network sensor will wait a long time between firing a second packet.

iv. Sensor Cost:

The energy cost in detecting a vector and generating a packet.

v. Transmission Radius:

The maximum distance within which two network nodes can communicate. If set to a high value, nodes on opposite sides of the map may be able to reach each other; if set to a low value, nodes must be very close to communicate.

vi. Transmitter Period:

The amount of time required to send a packet. Setting this to a high value will cause each packet transmission to take several seconds. Thus, the data received at the radar will be quite stale, since many seconds will have elapsed since the triggering event. However, the high period allows the user to monitor the packet-exchange process on the network map.

vii. Transmit Cost:

The energy cost in sending a packet. Setting this value very high will cause nodes to be depleted after sending only a few packets; setting this value very low allows the nodes to send many hundred packets. (Note that this is always scaled based on the distance between the nodes; thus, since more distant nodes can only be reached by a more powerful

broadcast, such transmissions more quickly deplete the energy store of the transmitting node.)

viii. Receive Cost:

The energy cost in receiving a packet. (This value is not scaled, as is the transmit cost.)

2. Routing Parameters:

These factors determine the software properties of the network: essentially, the packet-routing method to be used. If routing is set to "Random," each node selects a downstream connection randomly for each packet. If set to "Directed," the network routes packets based on the algorithm described in the Chang and Taissulas article. The directed routing parameters (Exchange Cost, Residual Energy, Initial Energy, and Routing Period) are best understood by reviewing the details of that article.

When the network parameters are set, the network can be deployed by clicking the "Deploy Network" button. The nodes of the network will be randomly scattered and connected, as shown on the main map. The communications of the network are directed from left to right, and nodes in the "uplink zone" (the striped zone at the right side of the map) are presumed to be in direct contact with the data collector. An alternative random scattering of nodes may be created by clicking the "Deploy Network" button again.

Once the network has been deployed, the simulation may be run by clicking "Start Simulation." The map will show vectors moving through the field and triggering sensors. The sensors may run out of power and drop out of the network, and eventually, all nodes will be powered down. The progress of the network can be monitored via the "Simulation Status" box. A new simulation may be run by stopping and restarting the simulation. Alternatively, the previous simulation may be reviewed by clicking the "Replay Simulation"

Display

The network is displayed on the main map as a series of red circles surrounded by gray circles. The red circle represents the sensor/node, and The gray area surrouding the node is the sensor detection range; any vector (the moving green rectangles) that enters this area will trigger the sensor. When this occurs, the gray area will turn a bluish color and gradually fade back to gray (the speed of this fading depends on the sensor delay period - see above.)

Each node is connected to nearby nodes by black lines, which represent communications links. If directed routing is in use, the connection that a node has currently selected is colored blue. When a packet is being exchanged on this connection, it will appear as red. This will likely not be visible unless the transmitter period is set to a rather high value; the lower values, which better reflect reality, cause packets to be transmitted so rapidly that the line will appear red only for a very brief time.

The color in the center of the red circle represents the battery status of the node, which gradually shifts from white (full power) to black (no power.) When a node loses all power, three changes occur: the node is no longer circled in red, but is totally black; the gray sensor area shrinks and disappears; and all of the communications links vanish.

The radar at the bottom of the screen shows the results of the data transfer. Here, the nodes are shown as green circles, and the vectors are shown as small, white rectangles. If a packet successfully reaches a node in the uplink zone of the network, it is transferred to the radar and displayed as a hit by coloring the circle a bright green. Thus, the speed and accuracy of the network may be viewed, as they pertain to the vectors passing through the field.

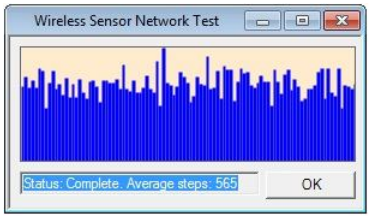


Fig: Average Steps

Programming

This simulator was written in Java language using jdk1.8 version and Integrated Development Environment i.e. IDE as Netbeans8.0.2, which the we finds to be an excellent programming platform.

The application is written in two parts:

One module represents the wireless sensor network, and the other is the simulator that hosts the wireless sensor network objects. The classes that comprise the wireless sensor network are as follows:

i. WirelessSensor:

This class represents a single sensor. The class contains many basic and straightforward parameters, such as iSensorRadius (the radius, in pixels, of sensitivity of the sensor),

iSensorDelay (set to zero if the sensor is ready for detection, or positive if the sensor has recently detected a vector and is momentarily desensitized.), and x and y (integers representing the position of the node on the map.)

Noteworthy member variables include:

ii. Packets:

an ArrayList of Packet objects, each representing a datagram to be forwarded to one of the downstream connections.

iii. Connections:

an ArrayList of Wireless Sensor Connections representing connections to downstream nodes.

iv. connectionCurrent:

if directed routing is in use, the currently selected downstream node.

v. ResidualEnergy:

the amount of battery power remaining in the sensor and node; the sensor is disabled when this falls to or below 0.

vi. WirelessSensorConnection:

This class represents a connection between two network nodes, internally designated as sSender and sReceiver. The connection contains a placeholder for the packet being exchanged, as well as a timer to simulate a non-instantaneous exchange period. These objects fit into the simulation as members of the aConnections list (always hosted by the upstream node.)

vii. Packet:

This class represents the datagram exchanged between nodes. It simply contains variables x, y, lifetime (the maximum amount of time that the sensor will appear on the radar), and timestamp (the amount of time left until the packet expires on the radar.)

viii. WirelessSensorNetwork:

This class represents the entire network. In addition to many parameters that correspond to the sliders in the simulator (Receiver Cost, Sensor Cost, Sensor Delay, etc.) and some data synchronization variables, this class contains two important ArrayLists: aSensors, which holds all of the WirelessSensor objects, and aRadar, which holds all of the packets that have been transmitted out of the network and are appearing on the radar.

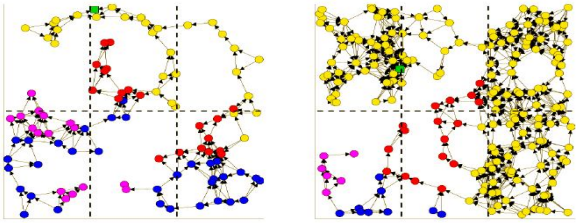
ix. VectorList and Vector:

The VectorList class contains a list of Vector objects, each of which represents an object moving across the map and being detected by the network sensors.

The interface class is a typical Java JFrame event-driven interface. Only two aspects of this system are noteworthy:

The actual network simulation runs on a separate thread from the thread controlling the user interface. This simulator supports completely gigantic networks - 400 nodes x 400 nodes, which, fully interconnected (maximum network transmission radius), include 160,000 network connections. Simulating transmissions across all of these connections, and then drawing the whole network several times per second, requires a nontrivial deal of processing. The CPU of a typical 3.2GHz machine simply can't juggle this task with the message pump that handles window events; hence, cramming both functions into one thread results in a completely unresponsive simulator window. Instead, the simulation thread is started in a separate thread, which runs continuously until the user stops it (by clicking Stop) or closes the window; as a result, the window remains responsive during the simulation.

Isolating the drawing and simulation functions to separate threads creates a synchronization problem: If the main thread is tracing the ArrayList of vectors at the same time that the processing thread updates the list (by inserting or, worse, removing an item), there exists the likelihood of an exception. To preclude this occurrence, the threads synchronize access to the vector list by using a mutex.



100 nodes 200 nodes

V. CONCLUSION AND FUTURE WORK

Wireless sensor networks applications can be found in every field of life. One of the exigent problems occurring in such environment is the formation of network holes. It occurs when a group of nodes stop operating due to some reasons.

Hole degrades the general performance of the networks. It destroys a major part of the network and leads to problems in data reliability and data routing.

The routing techniques are most relevant in the networking based methodologies. Various routing strategies are allowed different types of routing protocols based on topology planarization and greedy forwarding approach. In this paper a cross-layer relay selection mechanism favoring nodes that can forward traffic more effectively and reliably, depending on traffic and link quality. The proposed scheme designed to handle dead ends, Rainbow, is fully distributed and has low overhead also makes it possible to route packets around connectivity holes without resorting to the creation and maintenance of planar topology graphs. Rainbow is a mechanism which guarantees packet delivery under arbitrary localization errors that is at the sole cost of a limited increase of the route length.

This paper gave an idea about connectivity holes in wireless sensor networks and some routing techniques that route packets around these holes. The cross-layer routing named XLP gives the best performance in case of routing around connectivity holes.

REFERENCES

[1] Chiara Petrioli, Michele Zorzi. "ALBA-R: Load-Balancing Geographic Routing Around Connectivity Holes in Wireless Sensor Networks" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 3, MARCH 2014.

[2] J. Ding, K. M. Sivalingam, R. Kashyapa. "A multi-layered architecture and protocols for large-scale wireless sensor networks,"Proc. IEEEVehicular Technology Conference (VCT2003), Orlando, USA, October 2003

[3] S. Datta , I. Stojmenovic "Internal node and shortcut based routing with guaranteed delivery in wireless networks,"Cluster Computing,vol.5, no. 2, pp. 169-178, 2002.

[4] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, and A. Zhu,"Geometric Spanners for Mobile Networks," IEEE J. Selected Areas in Comm., vol. 23, no. 1, pp. 174-185, Jan. 2005.

[5] H. Frey, S. Ru¨ hrup, and I. Stojmenovic, "Routing in Wireless Sensor Networks," Guide to Wireless Sensor Networks, S. Misra, I.Woungang, and S. C. Misra, eds., ch. 4, pp. 81-112, Springer-Verlag, May 2009.

[6] IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 4, NO.2, MARCH/APRIL 2005A Location-Based Routing Method for Mobile Ad Hoc Networks Ljubica Blaze Vic, Member, IEEE, Jean-Yves Le Boudec, Fellow, IEEE, and Silvia Giordano, Member, IEEE

[7] M. Zorzi, ―A New Contention-Based MAC Protocol for Geographic Forwarding in Ad Hoc and Sensor Networks,‖ Proc. IEEE Int'l Conf. Comm. (ICC '04),vol.6, pp. 3481-3485, June 2004.

[8] D. Son, A. H., And Krishnamachari, B. "The effect of mobility-induced location errors on geographic routing in mobile ad hoc and sensor networks: analysis and improvement using mobility prediction".IEEE Trans. Mobile Comput. 3, 3 July 2004

[9] Jae-Hwan Chang and Leandros Tassiulas, IEEE/ACM Transactions of Networking, Vol. 12, No. 4, August 2004.