uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Francisco Javier Ovalle Martinez
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Master of Computer Science
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Area Based Beaconless Broadcasting in Ad Hoc and Sensor Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Ivan Stojmenovic
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Thomas Kunz

Amiya Nayak

Gary W. Slater
LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Area Based Beaconless Broadcasting in Ad Hoc and Sensor Networks

by

Francisco Javier Ovalle Martínez

Thesis submitted to the Faculty of Graduate and Post-Doctoral Studies
in partial fulfillment of the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 0-494-11372-3
Our file Notre référence
ISBN: 0-494-11372-3

Acknowledgments and Dedications

First of all I would like to thank my supervisor: Professor Ivan Stojmenović. His advice and guidance taught me to always pursue high quality research. Thanks to him I have develop several abilities that for sure will be fundamental in pursuing my PhD and beyond that, his lessons will transcend in my future research life. Thank you Ivan.

Also I would like to thank Professor Amiya Nayak. His chats and personal support was extremely useful in times when thinks did not look very well.

I thank The National Council for Science and Technology (CONACyT). This Mexican organization funded my Master studies at the University of Ottawa.

Another Mexican institution that helped me during part of my studies with financial support for some living expenses was the Mexican Secretary of Public Education (SEP).

I dedicate this work to my friends, to my family, and to the love of my life: Iris. To my friends, who in spite of being geographically far, always were close in my heart and my mind. They forged to a large extent the person who I am. To my grandmother, who with her unconditional affection helped me to finish this step in my life. To my Mother, who always has supported me in all my dreams. Besides receiving her unconditional love, her tenacity in life has been the best example that somebody has given me to develop the qualities that I have. To my little sister Irma and my beautiful niece Cecilia.

To the love of my life, thank you Iris for the sacrifice of so many things to live together this adventure in Canada. Your affection, your support, your laughter and mainly your love helped me on to continue successfully. You are the energy that moves me and that impels me to continue looking for my personal dreams. Without you, definitively this could not be possible. Thank you very much my "pequeña".

Abstract

We consider the broadcasting problem in ad hoc and sensor networks where no 'Hello' messages are used in order to preserve power and bandwidth. We propose five Area based Beaconless Broadcasting Algorithms (ABBAs). In 2D, each node calculates the ratio P of its perimeter along the circle of transmission radius, which is covered by the transmissions of the same packet. The node then sets or updates its timeout proportionally to P. We also consider an alternative random timeout function. If the perimeter becomes fully covered, the node cancels the retransmission; otherwise, it retransmits. We also describe four 3D ABBAs, these protocols are based on covering three projections, covering particular points on intersection circles, covering intersection points of three spheres, and covering intersection circles. Three ABBAs are the first reliable broadcasting protocols, other than blind flooding. 2D ABBA is compared with two beaconless protocols, OFP and Geoflood, showing its overall superiority.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Background and Motivation

Wireless networks consist of a set of wireless nodes spread over a geographical area. These nodes are able to perform processing tasks and are capable of communicating with each other by means of a wireless ad hoc network. Wireless nodes cooperate to perform data communication tasks.

Broadcasting is an important task used for paging, alarming, location updates, route discoveries or even routing in highly mobile environments. Blind flooding is a classical approach to broadcast messages across a network. It starts with the source node disseminating a message among its neighbors. Whenever a node receives the message for the first time, it transmits one copy to all of its neighbors. The scheme stops after each node, connected to the source, has received (possibly a few times) the message and retransmitted it once. Despite of its broad use and advantages, an increase in the density (average number of neighbors) produces communication overheads that limit the scalability of blind flooding.

Other broadcasting algorithms have been developed. There exist several localized broadcasting algorithms that use intelligent techniques to reduce communication overhead during the broadcast. However, these protocols assume 1-hop or possibly 2-hop knowledge. In order to discover its neighbors, the nodes in the network use beacons. The use of such 'Hello' or 'ping' messages increases dramatically in topologies with great mobility, resulting in critical energy losses. To address this issue, beaconless broadcasting, more intelligent than flooding, has been considered. In these protocols, here are no 'hello' messages, and nodes act based on information gathered only during the broadcasting process. We believe that

beaconless broadcasting protocols can reduce energy consumption and can even be a viable option for routing in networks with great topology changes (great mobility). This thesis studies the design of beaconless broadcasting protocols. The protocols proposed here require position information of each node to be available, which was not necessary to run blind flooding protocol.

1.2 Existing Solutions

The first beaconless broadcasting protocols (other than blind flooding) are probabilistic and location based schemes by Ni, Tseng, Chen and Sheu [NTCS]. However, they do not have good performance, either leaving many nodes uncovered or requiring most nodes to retransmit, as shown experimentally in [SSZ]. We identified two existing recent beaconless broadcasting protocols: Optimal Flooding Protocol and Geoflood. OFP ([PDJ] and [PDDJ]) is based on selecting forwarding nodes closer to the optimal vertex location of a honeycomb network. The message contains the location of the previous and last senders. This enables a receiver to determine the locations of two vertices of a honeycomb network, with side being equal to transmission radius, and with one direction corresponding to the edge direction determined by the last two senders. Upon receiving such message, each node finds its distance to the two locations (vertices of the hexagon), and sets a timeout proportional to the minimum distance. This allows closer nodes to retransmit sooner, which also suppresses other retransmissions. The authors [PDJ, PDDJ] claim that their algorithm minimizes the number of necessary transmissions and that it outperforms all other variations of flooding. However, they did not implement the beaconless version of OFP. Their implementation uses 1-hop knowledge, and they did not discuss OFP's reliability. As we will show in later sections, the beaconless version of OFP lacks reliability (the most important factor in a

broadcasting task). We will also show that, even for highly connected networks, OFP can fail to deliver to all recipients connected to the source. We consider that OFP is not a satisfactory option to perform a beaconless broadcasting.

Geoflood [ADEP] is based in setting timers before retransmitting the message. Each node defines a Cartesian plane with its location as the origin. Upon receiving a first message a node records the quadrant from where the message was sent and sets a timeout. If the receiving node receives more copies of the same message from all the other quadrants before the timeout expires, it does not retransmit. As with OFP, the lack of reliability is the principal problem that Geoflood has. Its authors recognized and showed this problem; however, they claimed that in high-density networks this could not be an issue. In later sections we will show how to modify Geoflood to convert it to a reliable beaconless broadcasting algorithm. However, we will also show that reliable variants are not efficient compared to our new proposals.

1.3 Assumptions and Limitations

The development of our proposals are based on the following assumptions:

- All nodes use omni directional antennas.
- Each node has its own geographic position information available.
- In 2D, the transmission pattern of every node is a circle with radius equal to the transmission radius R.
- In 3D, the transmission pattern of every node is a sphere with radius equal to the transmission radius R.

- Nodes are not aware of its neighbors; in other words, nodes have no k-hop neighbor's knowledge.
- Each message has the coordinates of the sender.

The simulations for this work have the following assumptions:

- Ideal MAC and Physical layers.
- Only one broadcast at a time
- Transmission radius was the same for each node and was fixed.
- All nodes shared the wireless channel; when a node transmitted a message, all its neighbors heard the transmission.
- There were no obstacles between any two nodes; a message was received if and only if the distance between sending and receiving node was less than the transmission radius.
- All nodes were static while a broadcasting was in progress.

1.4 Objectives and Contributions

The principal objective of this work is to propose and develop several beaconless broadcasting algorithms. We want to establish the basis of wireless broadcasting protocols that do not use 'Hello' messages and that are reliable. More precisely we propose five versions of Area Based Broadcasting Algorithms (ABBA).

1.4.1 2D-ABBA

The idea behind 2D-ABBA is very simple. We assume that nodes do not have information about their neighbors. Upon receiving the first copy of a message that is broadcast, each node sets a timer. Before the timeout expires, the node may receive more copies of the same message. Every node has a transmission coverage, which is assumed to be a circle. If a node A receives a message from different sources and these sources cover the transmission circle of A, then node A has no need to retransmit the message. This means that each of its possible neighbors has received the message already from one or more nodes that cover it. If the node is covered before the timeout expires, then the node decides not to transmit and ends the timer. However, as long as the node is uncovered, the timer will continue to run, and when the timer expires the node transmits.

Since all circles are of the same radius R, and each node A is aware only of intersections with circles whose center B is inside its own circle (that is, $|AB| \leq R$), the coverage criterion can be simplified. Instead of covering the whole circle centered at A, only the perimeter of that circle needs to be covered. Further, when this perimeter is intersected by several circles whose centers are inside A's circle, there can be up to two segments on the perimeter that are not yet covered. This property further simplifies the implementation. This broadcasting protocol is reliable independently of the selected timeout function.

We propose two possible timeout functions for 2D-ABBA. The first one is a function that increases when the total length of uncovered portions of the circles with transmission radius around the node decreases. The function chosen was *timeout* = *degreesCovered*. The variable *degreesCovered* refers to the angle in degrees of the circle that has been covered. The second function that we considered is a random function with values between 0 and 1.

We considered the second function to test the viability of 2D-ABBA if used in conjunction with already deployed IEEE 802.11 networks.

1.4.2 3D-ABBA1

In 3D, a similar simplification can be applied. All spheres are of the same radius, and each node A is only aware of transmissions from neighbors at distance $\leq R$. Instead of covering the whole sphere centered at A, one can consider only covering its 3D perimeter (set of nodes at distance $=R$ from A). All versions described here refer to covering that 3D perimeter.

The idea behind our 3D-ABBA1 protocol is to use the three projection planes XY, XZ, YZ. We used these planes to try to observe the intersections of transmission spheres of nodes in the network. By projecting the transmission spheres into the three planes, the nodes are able to apply 2D-ABBA in each plane. Node A will not transmit if its corresponding transmission circles in all three planes have been covered. This in turn is simplified to covering the corresponding 2D perimeters. Again we defined two different timeout functions. The first one was directly proportional to the three perimeters (one for each plane) covered by other nodes. In other words we considered:

$$timeout = \frac{degreesCoveredXY + degreesCoveredXZ + degreesCoveredYZ}{3}$$

The variables *degreesCoveredXY*, *degreesCoveredXZ*, *degreesCoveredYZ* refer to the angles in degrees of the circles that have been covered in three corresponding planes.

The second function considered is a random function with values between 0 and 1. We considered the second function to test the viability of 3D-ABBA1 in IEEE 802.11 networks.

This variant of broadcasting protocol is not reliable. Examples can be constructed to show that some nodes, connected to the source, will not receive the message.

1.4.3 3D-ABBA2

For this version we considered the circles of intersection between two transmission spheres. When a node A receives a transmission, a circumference of intersection is created. If after receiving several transmissions each of the created circumferences is covered by other transmission spheres (defined by nodes that transmitted the same message) then the transmission sphere of node A is completely covered by others and it will not transmit. We further simplify computationally this scheme. Instead of dealing with the whole circles of intersection, we propose to select six points forming a hexagon that resides in the circles of intersection. Thus instead of trying to cover the whole circumference, we propose to cover six points that reside in the circle.

Again we propose two possible timeout functions. The first one is a timeout function that depends on the number of already heard transmissions, number of points to be covered, and another parameter. This extra parameter is directly proportional to the volume covered by the transmission. It is obvious that the volume covered by the transmitters depends directly on the distance between the sender and the receiver. For this reason, we decided to assign a weight of $\frac{d}{R}$ to each point of the hexagon. The parameter d is the distance between the transmitter that created the point and the receiver. The parameter R is the transmission radius. In conclusion, we propose a timeout function:

$$Timeout = (\text{number of received transmissions}) + IntersectFunction$$

$IntersectFunction$ = sum of the weights of the intersection points still to be covered.

The second function that we considered is a random function with values between 0 and 1. We considered the second function to test the viability of 3D-ABBA2 in already deployed IEEE 802.11 networks.

One drawback of 3D-ABBA2 is that reliability cannot be proven. In fact, it is possible to design pathological cases where some nodes will not receive the message despite being connected to the source. However, these instances are rare; in fact we did not find any in our experiments.

1.4.4 3D-ABBA3

For 3D-ABBA3 we considered applying three spheres of intersection instead of two. One of these spheres is the one centered by the node making the decision. Thus the intersection points are on its 3D perimeter. The intersection (if it exists) consists of two points. If each such intersection point is located inside another transmitting sphere, then the considered sphere is covered in full. This property can be proven to be correct.

Therefore each receiving node keeps a list of all intersection points. Whenever a new node transmits a message in the neighborhood then following events happen:

1. Some existing intersection points are inside it. These points are eliminated.
2. A new sphere is considered, with the current fixed sphere, and with all previously transmitting spheres, for triples to find new intersection points. Each such point is tested whether it is inside another existing sphere. Those that are inside are ignored. Those that are outside any existing sphere are entered into a list of intersection points to be covered.

When a transmission from a node B does not generate intersection points, receiving node A adds node B to a list of transmitters that do not generate intersection points. We named this list as list of single spheres. Every time a message is received, node A checks its list of single spheres to see if these nodes now create intersection points with the last received transmission. When an intersection is created, the corresponding node is taken out from the list of single spheres and the generated points are inserted in the list of intersection points.

When a node A has an active timer and both lists (the list of intersection points and the list of single spheres) are empty, then node A is fully covered, ends its timer for this message and decides not to transmit.

We again propose two possible timeout functions. The first one involves using the intersections created by transmitting nodes with receiving nodes (transmission patterns). Each time a receiving node A gets a transmission from a node B, node A applies the following algorithm:

BEGIN 3d-abba3 timeout Algorithm

IF the transmission done by B creates n valid intersection points THEN

$i = 1$;

WHILE $i \leq n$ DO

C_i = node that created intersection point with node A and with node B;

$$\text{timeout} = \text{timeout} + \frac{\text{distance}(A,B) + \text{distance}(A,C_i)}{2R};$$

i++;

ENDWHILE

ELSE IF the transmission done by B does not intersect with other spheres THEN

$$\text{timeout} = \text{timeout} + \frac{\text{distance}(A,B)}{R};$$

ENDIF

END 3d-abba3 timeout Algorithm

The second function that we considered is a random function with values between 0 and 1. We considered the second function to test the viability of 3D-ABBA3 in already deployed IEEE 802.11 networks.

1.4.5 3D-ABBA4

In 3D-ABBA4 we considered the circles of intersection between two transmission spheres. When a node A receives a transmission, a circumference of intersection is created. If after receiving several transmissions all the created circumferences are inside another transmission spheres (from all nodes that transmitted the same message) then the transmission sphere of node A is completely covered by others and it will not transmit. Therefore this is a very close procedure to the 3D-ABBA3. Instead of considering coverage of intersection points only, the whole perimeter is considered for coverage. These procedures are therefore both reliable, and even expected to have the same performance, if the same timeout functions are applied to them.

Again we propose two possible timeout functions. Since we are dealing with circles of intersection, each time a new circle is created by the transmission of node B to receiving node A, and after its perimeter is updated with previous transmissions, then the timeout function results in: $timeout = timeout + \theta \frac{\text{distance}(A,B)}{R}$;

Where θ = angle in degrees of the perimeter yet to be covered of the circle created between nodes *A* and *B*. The second function that we considered is a random function with values between 0 and 1. We considered the second function to test the viability of 3D-ABBA4 in already deployed IEEE 802.11 networks.

1.5 Analysis

We simulated our proposed broadcast algorithms in Matlab, measuring the number of transmissions done for different network densities. We implemented two versions of OFP, three versions of Geoflood and compared them with 2D-ABBA. We showed that OFP and Geoflood are not reliable algorithms. In order to become reliable, the nodes in Geoflood have to divide its transmission area in six angular regions instead of four. In our tests with 500 nodes and different average degrees, OFP failed considerably. In connected networks with average degree 7, OFP failed to complete a broadcast in 97% of the tested networks. Even in connected networks with high average degrees, OFP failed occasionally. This shows that OFP is not a good broadcasting protocol due to its frequent lack of reliability. Although Geoflood was experimentally successful in every test but one, it is shown that the versions with 4 and 5 angles division can be unreliable. In addition, Geoflood, even with 4 angles division, had higher retransmission counts than ABBAs. The reliable version of Geoflood (with division into six angles) leads to further increase in retransmissions. 2D-ABBA with coverage based timeout used on average 40% less messages than the reliable version of Geoflood, while the random 2D-ABBA had on average 28% less messages. Compared to Geoflood with 4 and 5 angles division, the original 2D-ABBA had on average 12% and 34% less messages respectively.

Among our 3D proposals, 3D-ABBA1, based on covering three 2D projections, had the best results in terms of number of messages to complete the broadcast. However, this version lacked reliability. The second best 3D version (3D-ABBA2) performed very close to 3D-ABBA1 and had 100% delivery rate during the tests, although this version in theory is also unreliable. 3D-ABBA2 is based on covering (by other transmitting spheres) six equidistant points on the intersecting perimeter of considered sphere and any transmitting neighboring sphere. 3D-ABBA3 is a reliable broadcasting protocol. It is based on a covering theorem. A sphere A is covered by other spheres if every intersection point of three spheres from the set, one of them being A, is located inside another sphere from the covering set. Any three spheres determine two such common intersection points to be covered. The other reliable algorithm 3D-ABBA4, presented very similar results when compared with 3D-ABBA3. Thus we believe that it is better to use 3D-ABBA3 than 3D-ABBA4 due to its simplicity.

Protocol versions that used the random timeout function were very competitive to the original optimized timeout values. Therefore we believe that our protocols will preserve good performance if they are implemented under a realistic MAC layer such as in IEEE 802.11 based networks. Recall that IEEE 802.11 uses random timeouts in its back off scheme.

1.6 Thesis organization

The remainder of this work is organized as follows: Section 2 presents the related work for the stated (broadcasting) problem. In section 3 we explain, describe and show the five versions of ABBA. Section 4 gives performance evaluation of the proposed broadcasting algorithms. Section 5 compares OFP and Geoflood with 2D-ABBA. Section 6 concludes this work and discusses relevant open ideas.

Chapter 2 Literature Review

Williams and Camp [WC] classified the broadcast protocols into: simple (blind) flooding, probability based, area based, and neighbor knowledge methods. Stojmenović and Wu [SW] reclassified area based methods within other groups while neighbor knowledge methods were divided into clustering based, selecting forwarding neighbors, and internal node based methods. They presented a comprehensive taxonomy of broadcasting schemes with a one-to-all model in mind (the other models can similarly be considered). All schemes can be classified following the taxonomy consisting of five categories: determinism, network information, reliability, 'hello' message content, and broadcast message content.

In the blind flooding scheme, whenever a node receives the message for the first time it forwards that message to its neighbors, except to the node from which it just received it. The method disseminates information quickly in a network with enough bandwidth and no loss prone links [HKB]. Nevertheless, it has disadvantages prohibiting its use in networks with dynamic topology behavior or where requests for information are frequent and must be known by all the participants. The main disadvantage of blind flooding is its extensive use of the bandwidth, if dense underlay networks are used. At the node level, each message retransmission consumes processing time and the available bandwidth on both incoming and outgoing links.

In local broadcasting, a distributed broadcast protocol is based on solely local state information. All protocols that select forward nodes locally (based on 1-hop or 2-hop neighbor set) belong to this category. It has been recognized that *scalability* in wireless networks cannot be achieved by relying on solutions where each node requires global knowledge about the network. To achieve scalability, the concept of *localized* algorithms

was proposed, as distributed algorithms where simple local node behavior, based on local knowledge, achieves a desired global objective.

Ni, Tseng, Chen and Sheu [NTCS] studied the broadcast storm problem. A straightforward broadcasting by flooding is usually very costly and will result in serious redundancy, contention, and collision. They identified this broadcast storm problem by showing how serious it is through analyses and simulations. Several beaconless schemes (probabilistic, counter-based, distance-based and location-based) were proposed in [NTCS] to reduce redundant rebroadcasts and differentiate timing of rebroadcasts to alleviate this problem. In the probabilistic scheme [NTCS], each node rebroadcasts the first copy of a received message with a given probability p. In the counter-based scheme [NTCS], each node rebroadcasts the message if and only if it received the message from less than C neighbors. In the distance-based scheme [NTCS] the message is retransmitted if and only if the distance to each neighbor that already retransmitted the message is bigger than D. In the location-based scheme [NTCS] the message is retransmitted if and only if the additional area that can be covered if the node rebroadcasts the message (divided by the area of circle with transmission radius) is greater than the threshold A. A simplified version of the method is to rebroadcast the message if the node is not located inside the convex hull of neighboring nodes that already retransmitted the message. However, these methods are not reliable. Further, the experimental data in [NTCS, SSZ] indicate low saved rebroadcasts for high reachability, compared to other methods. We therefore did not include them in our experiments.

Cartigny and Simplot [CS] described a distance-based method without using position information. The distance between two neighboring nodes is measured by a formula that depends on the number of common neighbors. The broadcast message is piggybacked with a

list of 1-hop neighbors. Neighbor elimination is also used to enhance the performance. The method is suitable for highly mobile environments, since 'hello' message content is minimized.

Several authors [CMWZ, QVL, LK, SL] proposed independently reliable broadcasting schemes in which the sending node selects adjacent nodes that should relay the packet to complete broadcast. Sun and Lai [SL, SL1, SL2], and Calinescu, Mandoiu, Wan and Zelikovsky [CMWZ] presented heuristics that aimed at covering the whole area where 2-hop neighbors could be located by a minimal set of 1-hop neighbors, and analyzed the performance of their schemes. The problem is equivalent to selecting a minimal set of disks that still cover the same area as the area covered by all disks centered in neighboring points. Their forwarding sets contain, on average, more nodes than the one based on a set cover heuristic, since the forwarding sets are given larger areas to cover, and no 2-hop information is used. Thus only 1-hop position information is used. The solutions are based on the notion of curved convex hull. Each forwarding node in variants [SL, CMWZ] includes the forwarding set as part of message. In variant [SL1, SL2], this is avoided by transferring the overhead to hello messages, which contain position of the sender and the list of its neighbors (without position information). The 2-hop neighbor information and 1-hop position information are used to calculate the local cover set of the sender's node at the receiver's node.

Stojmenović, Seddigh and Zunic [SSZ] applied the concept of dominating sets to reduce the communication overhead of broadcasting a message; they proposed the use of node degrees instead of node id's for primary key. They also proposed the neighbor elimination method to reduce the number of broadcasts in the network. The basic idea of neighbor

elimination is that a node will rebroadcast the message only if it has a neighbor that might need the message. Thus, some of the neighbors are eliminated for re-broadcasting.

Lin and Gerla [LG] proposed a distributed clustering algorithm that is initiated by all nodes whose id is lowest among all neighbors (local lowest id). They broadcast their decision to create clusters to all of their neighbors. Each node may hear the broadcast by its neighbors and select the lowest id among neighboring cluster heads, if any. If all the neighbors that have a lower id send their decisions and none of them declares it as a cluster head, the node decides to create its own cluster head and broadcasts its id as cluster id. Otherwise, it chooses a neighboring cluster head with lowest id and broadcasts such a decision. Thus, each node broadcasts its clustering decision after all its neighbors with lowers id have already done so. Every node determines its cluster (only one) and transmits exactly one message during the algorithm.

The second algorithm evaluated in [SSZ] was proposed in [CS1], the authors proposed that each node is assigned a combined id, using the node id and the node degree. They proposed that a node has cluster head priority over the other if it has higher connectivity or, in case of equal connectivity, has lower id. The algorithm then follows the algorithm proposed in [LG].

Wu and Li [WL] also introduced two rules that considerably reduce the number of internal nodes in the network. Rule 1 [WL] is as follows. Consider two intermediate neighboring nodes v and u. If every neighbor of v is also a neighbor of u, and $id(v) < id(u)$, then node v is not an *inter-gateway* node. We may also say that node v is 'covered' by node u. Observe that retransmission by v, in this case, is covered by retransmission of u, since any node that might receive a message from v will receive it instead from u. Next, let the *gateway* nodes be those inter-gateway nodes that are not eliminated by Rule 2 [WL], defined

as follows. Assume that *u, v* and *w* are three inter-gateway nodes that are mutual neighbors. If each neighbor of *v* is a neighbor of *u* or *w,* where *u* and *w* are two connected neighbors of *v,* and *v* has lowest *id* among the three, then *v* can be eliminated from the list of gateway nodes.

Gerla, Kwon and Pei [GKP] proposed a combined clustering and broadcasting algorithm that has no communication overhead for either maintaining cluster structure or updating neighborhood information. In their passive clustering algorithm, the cluster structure is updated with existing traffic by adding two bits to each ongoing message. The source *S* of a broadcasting task transmits the message to all of its neighbors. *S* declares itself a CH (for the timeout period that is a parameter in the method) if it has no neighboring active CH. Upon receiving the message, each node *A* declares itself a CH using the same criterion as the source *S.* Otherwise, *A* checks the ratio of neighboring CHs and neighboring gateway nodes and declares itself a gateway if that ratio is above certain threshold, which is also a parameter of the method. If *A* decides to be a gateway, it retransmits the message. Otherwise *A* decides to be an ordinary node and does not retransmit the message. The method is not reliable, there are pathological cases of poor delivery ratio, and has global parameters.

Yi, Gerla and Kwon [YGK] modify the passive clustering protocol from [GKP] (note that they do not give reference to [GKP] in [YGK]) to achieve broadcasting without periodic, background control packet exchange. Passive clustering maintains clusters using implicit timeout. A node assumes that the nodes it had previously heard from have died or are out of its locality if they have not sent any data within the timeout duration. Thus instead of using CH to gateway ratio [GKP], the protocol [YGK] uses a cluster timeout interval. A node that belongs to two or more clusters at the same time is eligible to be a gateway node.

Only one gateway for each pair of neighboring gateways is allowed. Distributed gateways are used to connect neighboring cluster heads three hops away. The problem with the protocol [YGK] is that no description is given on how the process works at the very beginning. Assume, for example, that there was no traffic at all in the network for the timeout interval, and therefore complete cluster structure is lost. How then broadcasting proceeds from a node? According to the algorithm [YGK], a source node transmits the message, and all its neighbors then become gateway candidates and simply hold the message without retransmitting it. Another problem is that the selection of a proper value of timeout interval is highly dependent on the mobility speed. High node mobility may 'destroy' the cluster structure before expiration of timeout interval, which of course affects delivery rate for broadcasting. The protocol can be completed along the lines presented in this article, which however does not distinguish between gateways and cluster heads; they are both dominating set members. It also does not use timeout interval to decide lifetime of dominating set structure, but starts a separate timeout upon receiving a message, not before. Since message speed is normally much higher than mobility speed, the delivery rate is not affected as in [YGK]. In our protocol, nodes instead make timely accurate decision based on ongoing traffic, creating the dominating set on the fly.

Bergonovo et al. [BCCFCMR] described applications of broadcasting in inter-vehicle communications on the highway, with large and variable number of mobile terminals, an extremely dynamic network topology, and stringent requirements related to broadcast warning messages. Their proposed multi-hop broadcast protocol is based on reservation ALOHA MAC protocol. The protocol appears to be a variant of a neighbor elimination scheme [SSZ], which requires the maintenance of the neighbors list. This may not be an easy task in highly mobile environment, where ongoing traffic has to be mixed with beacon

messages for maintaining neighborhood information. The solution should be beaconless, and with guaranteed delivery, since missing to warn a single car may be a fatal mistake.

Lin and Shrivastava [LS] considered small-scale auctions and proposed a broadcast counter-based broadcast algorithm from [NTCS] for use in the environment. In this protocol, each node receiving a message for the first time sets a timeout interval, which can be modified if more messages arrive while it waits. If a node receives a certain threshold number of messages (suggested value is 3) the message is not forwarded. Otherwise, it is forwarded upon the expiration of timeout interval. As noted already in [SSZ], the protocol does not guarantee delivery, and therefore is not suitable in applications for auctions, which should provide access to auctioneer to all buyers.

Sun, Huang, Wang, Arora, and Lai [SHWAL] proposed a reliable multicasting, where RTS signals are followed by CTS signals from selected neighbors instead of all neighbors. The set of neighbors for sending back a CTS signal is selected so that they cover the same area as all the neighbors. Two versions are proposed, with and without using geographic information. Therefore [SHWAL] also use area coverage, but for a different purpose. It does not change the broadcast or multicast protocol on the network layer.

Beaconless algorithms were reviewed by Heissenbuttel et al. in [HB1] and [HB2]. Lately, a new class of position-based routing algorithms was introduced (BLR [HB1], CBF [FWKMH], IGF [BHSS]) that avoid having beacons transmitted periodically and, hence, eliminating the drawbacks that come along. The mechanism that allows selecting one neighbor as next hop in a completely distributed manner without having knowledge of the neighboring nodes is achieved in all the papers by applying the concept of DFD. They mainly differ in the investigated aspects. In CBF [FWKMH] and IGF [BHSS], the focus is on the integration of the routing protocol with the MAC-layer, namely the IEEE 802.11

protocol. [HB1] discusses several optimizations to the basic greedy scheme and as well analytically derives properties and limitations of this new class of protocols. The main drawback of these protocols are that greedy forwarding can be applied less often due to some restrictions on the position of the next node. Some of the advantages of these beacon-less algorithms are lost when a recovery procedure needs to be initiated. Therefore, these algorithms perform best in dense networks where they operate in greedy mode most of the time. The authors of [HB1] and [HB2] designed a beaconless routing algorithm, however their algorithm cannot be used for broadcasting tasks. Since their algorithm is based in the knowledge of the destination position, it cannot be considered for comparison.

After doing an extensive literature review we were able to find only two recent beaconless broadcasting protocols: Optimal Flooding Protocol and Geoflood.

2.1 Optimal Flooding Protocol

Paruchuri, Durresi, and Jain in [PDJ] proposed a flooding protocol based on hexagonal tiling of a plane, with transmission radius as the edge length of hexagons. The source chooses six nodes, which are closest to points that best approximate a regular hexagon for retransmitting the message. Designated nodes continue the process similarly. The reliability is not proven. As observed by Kim and Maxemchuk [KM], the protocol may repeat flooding if neighbors do not agree on the choice of a node near the common ideal point. They [KM] introduced a stopping rule to prevent that, and applied this type of flooding for route discovery.

The same authors from [PDJ] proposed a revised version of their protocol in [PDDJ] without citing the first article. In [PDDJ] the authors described a beaconless flooding algorithm named Optimal Flooding Protocol (OFP). OFP was the first 2D beaconless

broadcasting proposal to appear. Their protocol is based on selecting forwarding nodes closer to optimal vertex location of a honeycomb network. The message contains the location of the previous and last senders. This enables a receiver to determine the locations of two vertices of a honeycomb network, with its side being equal to the transmission radius, and with one direction corresponding to the edge direction determined by last two senders. Upon receiving such message, each node finds its distance to the two locations (vertices of the hexagon), and sets a timeout proportional to the minimum distance. This allows closer nodes to retransmit sooner, which also suppresses other retransmissions. They claimed their algorithm minimizes the number of necessary transmissions and that it outperforms all other variations of flooding. Their algorithm can be beaconless as they described in the last section of [PDDJ], however their implementation and results assumed 1-hop knowledge. This 1-hop knowledge gives unfair results compared to a beaconless implementation. OFP is based in three properties found in a hexagonal tiling of a plane, with transmission radius R as the edge length of hexagons. The mentioned properties are

- Property-1: Each vertex v is joined to three other vertices.
- Property-2: The lines joining these three vertices to vertex v make an angle of 120° with each other.
- Property-3: Each vertex is at a distance R from each of its neighboring vertices.

The first beaconless version of OFP is as follows:

At source node S: S calculates six ideal points forming an hexagon in its range R, and transmits them along with the message and an identifier stating that this message was initiated by S.

At an intermediate node B: A receiving node *B* calculates its distance from the locations specified and selects the shorter one. Node B waits for a time interval that is proportional to this distance, in this case we propose $\frac{\text{distance}}{R}$. If *B* receives the same message from another node *C* that is nearer to the specified location before the time interval expires, then node *B* does not retransmit the message. However, if *B* has to retransmit the message, first it checks if the message was received from a source node *S*.

- If yes, then it calculates the next ideal points that have to broadcast the request. Let (X_S, Y_S) be the location of S, (X_B, Y_B) the location of the intermediate node B. Then the coordinates (Xn,Yn) of the ideal location of the next node are given by:

$$Xn = 2X_B - X_S$$
$$Yn = 2Y_B - Y_S$$

  Then, node *B* appends the location (Xn, Yn) to the request and broadcasts it.

- If the request was not directly received from source node *S*, then, the locations of the next ideal points are calculated using Properties 1,2, and 3. More precisely, if node *B* received the message from node *A*, then the next ideal points P_1 and P_2 are the points located at a distance *R* where the line segments BP_1, BP_2 and *BA* make an angle of 120° with each other. Node *B* appends points P_1 and P_2 to the message and transmits it.

The second beaconless OFP version is almost the same; it only changes in how a node makes a coverage decision. Each node M stores the distance *dm* to the nearest node that has already transmitted the packet. A node does not retransmit, if *dm* for that broadcast message is less than a threshold *Th*. The choice of a right threshold will be the key for the success of

the proposed algorithm. The authors [PDJ] proposed a threshold value *Th* of 0.4*R to ensure high delivery ratio while keeping the nearest node to the strategic location.

A serious problem that OFP presents is its lack of reliability. As we will show in our experimental results OFP can fail even in highly connected networks.

2.2 Geoflood

Arango, Degermark, Efrat, and Pink [ADEP] proposed Geoflood. The algorithm is based in setting timers before retransmitting the message. Each node defines a Cartesian plane with its location as the origin. Upon receiving the first copy of a message a node records the quadrant from where the message was sent and sets a timeout. The authors [ADEP] proposed that receiving nodes furthest away from the local sender should select smallest holding times. So holding times increase as the distance to the sender decreases. They also introduced a random component to avoid contention that could arise between nodes located at the same distance from the sender. If the receiving node receives more copies of the same message from the other quadrants before the timeout expires, it does not retransmit. The authors [ADEP] recognized that their algorithm could fail in some situations. They showed the worst-case scenario in which part of the transmission area was not covered even when a node received messages from the four quadrants. They claimed that this area was relatively small and that other nodes in high-density networks would likely cover it. Unfortunately this causes Geoflood to be unreliable. Geoflood divides the transmission area in four quadrants or in four angles. This division causes the unreliability of Geoflood. In order to ensure reliability, Geoflood needs to divide into six angles instead of four. However, six transmissions from neighbors may not be necessary to cancel a retransmission, even in some cases three would be enough. This will bring unnecessary transmissions leading to

unnecessary use of energy. In other words, reliability comes with the price of increased average retransmission cost. Figure 1 shows a case where a node divides its transmission area in four angles, and since the node received a message from the four angles it will not retransmit. However as it can be observed there is still area uncovered. Also in Figure 1 we show the cases where a node divides its transmission area in five and six angles. As the reader may observe, even with five angles the algorithm could fail. The only way that Geoflood can guarantee reliability is by dividing the transmission area in six angles.

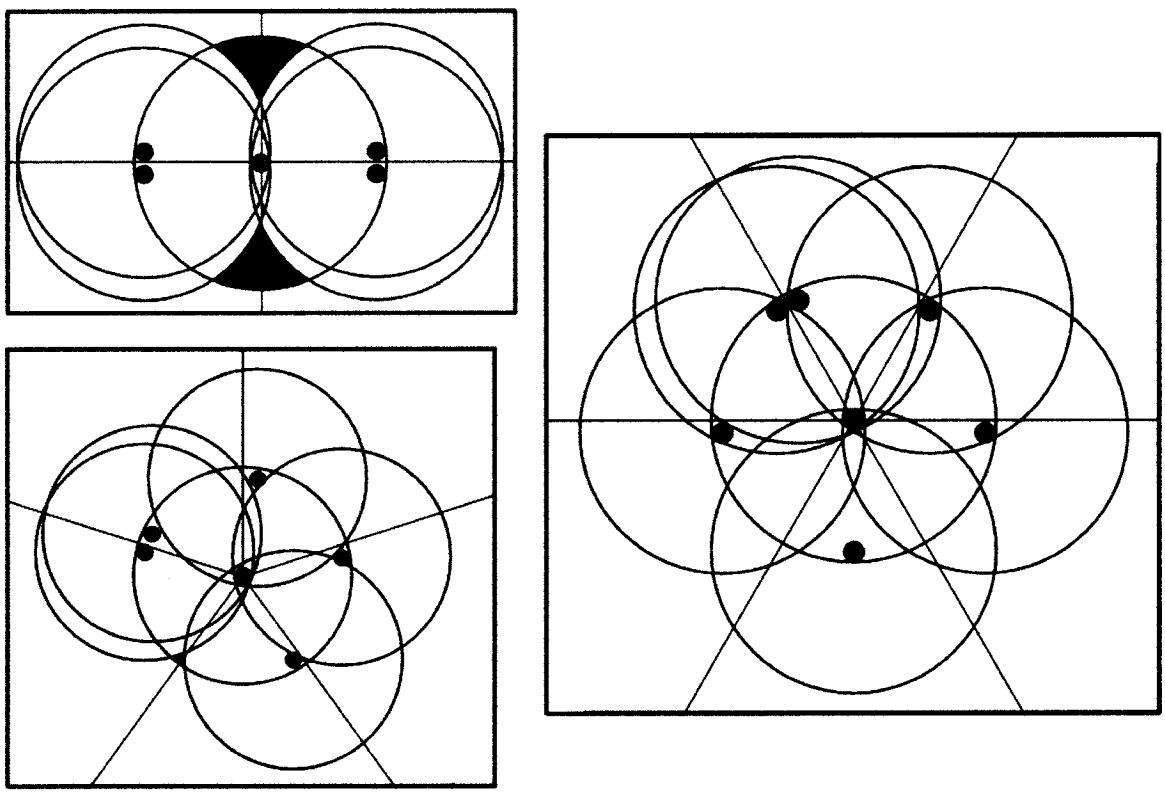

Figure 1. Receiving nodes dividing their transmission area in four, five, and six angles. Only with six divisions Geoflood guarantees reliability.

Chapter 3 Area Based Beaconless Broadcasting Algorithm in 2D

3.1 Assumptions

As in many existing broadcasting and routing protocols for wireless ad hoc networks we assume that every node in the network knows its geographic position. This geographic position can be obtained, for example, by the global positioning system (GPS). Our second assumption is that all nodes have the same transmission radius and use omni-directional antennas; in other words, their transmission patterns are circumferences. Finally, nodes do not know where neighboring nodes are, so no beacons or hello messages are used.

3.2 Description of 2D Area based beaconless broadcasting

In 2D, the omni-directional transmission pattern is a circle formed by the transmission radius. Basically, each node covers an area of transmission defined by this pattern. If each node is aware of the transmissions from other nodes that cover its own area, then it could decide whether or not it is necessary to retransmit. By only appending the transmitter coordinates to the messages, the receiving nodes are able to calculate how much of their transmission area is covered by the senders. However, when do receiving nodes make a decision? Receiving nodes set a timer that is directly proportional to the area covered by other nodes, a decision is made at the end of such a timeout. Afterwards, a node does not consider retransmitting again (although it may receive more copies of the same message). In other words, any node in the network can retransmit at most once a specific message. In

order to identify the messages, the original sender can add a unique identifier to the message, formed for example by its own ID plus a timestamp. Each node that rebroadcasts the message also sends the original identifier (from source) that was received at first. The pseudo code for our 2D algorithm is as follows.

```
BEGIN 2d-abba
    A source node S broadcasts a message M
    Each intermediate node L runs as follows with respect to the message M initiated by S
    {
        BEGIN wait until M is received for the first time
            Set perimeter covered;
            Set timeout;
            REPEAT
                Wait until another copy of M is received or timeout expires;
                IF another copy is received THEN
                    update perimeter covered;
                    update timeout (if required by protocol);
                ENDIF
            UNTIL timeout expires;
            IF still uncovered THEN
                Retransmit;
            ENDIF
            Ignore further copies of M
        END wait until M is received for the first time
    }END 2d-abba
```

We chose a timeout function that increased when the total length of uncovered portions of the circles with transmission radius around the node decreased. The function chosen for the 2D case is *timeout* = *degreesCovered*. The variable *degreesCovered* refers to the angle in degrees of the circle that has been covered. In Figure 2 we show two examples of how the transmission radius of a node could be covered by other transmission nodes. When a node gets completely covered it sets its timeout to 0 so it can decide immediately not to transmit.

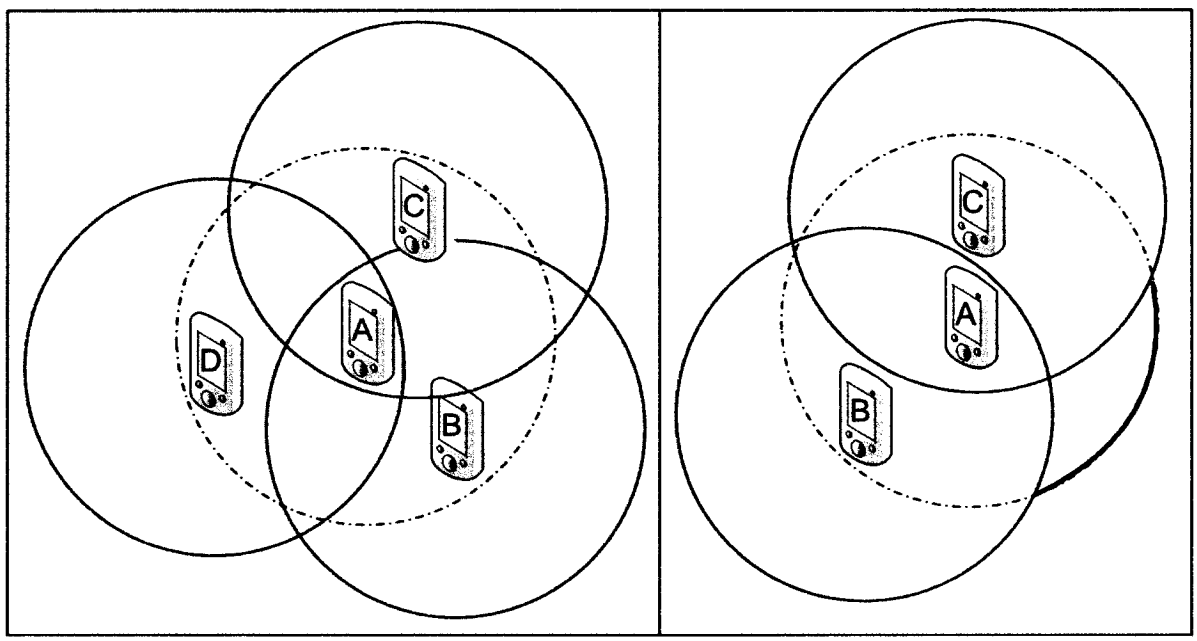

Figure 2. On the left: transmission area of node *A* is covered by transmissions of nodes *B*, *C* and *D*, so node *A* does not transmit. On the right: Nodes *B* and *C* retransmits the same message received by other sources, node *A* processes the messages and updates its perimeter covered by the other two nodes. After its timeout expires, node *A* will transmit.

3.3 Obtaining perimeters of intersection between two transmission circumferences

The first step to obtain these perimeters is to find the intersection points between the two transmission circumferences. One of our assumptions in 2D-ABBA is that all nodes have the same transmission radius R. This assumption simplifies the problem of finding the intersection between two circumferences of transmission. In Figure 3 (left) we show the intersection points $P1$ and $P2$ created by the transmission patterns of nodes A and B. Also from Figure 3 we can obtain the following values:

$$d = \frac{\text{distance}(A,B)}{2}$$

$$h = \sqrt{R^2 - d^2}$$

$$\theta = \cos^{-1}\left(\frac{x2 - x1}{\text{distance}(A,B)}\right)$$

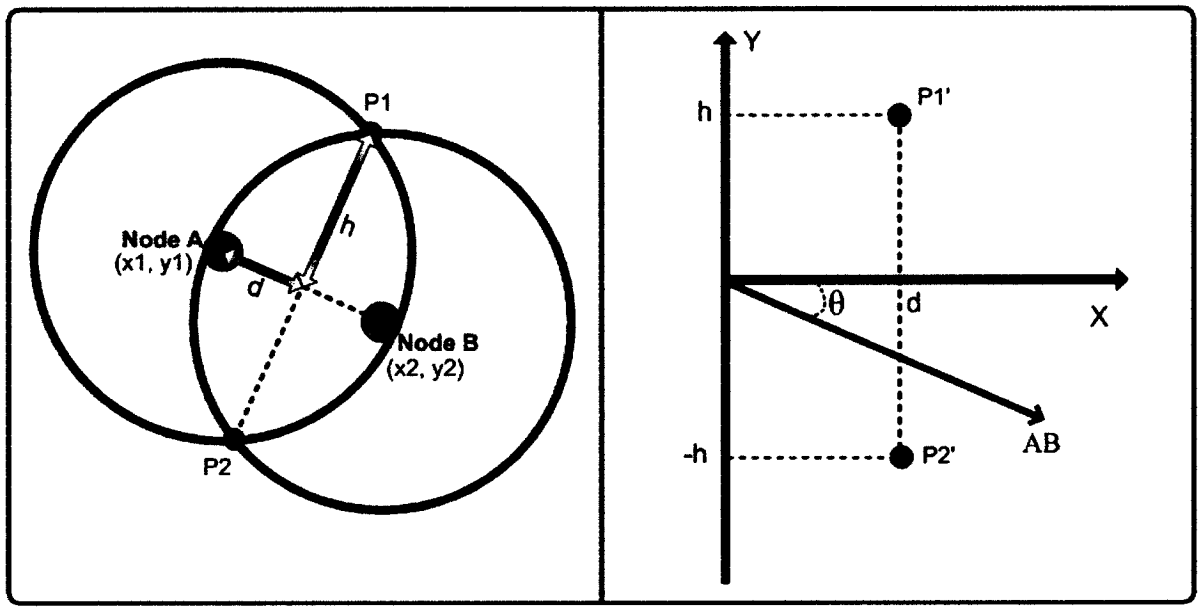

Figure 3. At the left, intersection points P1 and P2 formed by the transmission patterns of nodes A and B. At the right, points P1' and P2' will be rotated an angle θ and translated x1 and y1 units to obtain P1 and P2.

We propose to apply a rotation θ plus a translation (x1, y1) to the points P1' and P2' shown in the right part of Figure 3. In order to obtain points P1 and P2 we apply the following equations (translation and rotation transformations):

$$P1_x = x1 + d\cos(\theta) - h\sin(\theta) \ldots\ldots(1)$$

$$P1_Y = y1 + d\sin(\theta) + h\cos(\theta) \ldots\ldots(2)$$

$$P2_x = x1 + d\cos(\theta) + h\sin(\theta) \ldots\ldots(3)$$

$$P2_Y = y1 + d\sin(\theta) - h\cos(\theta) \ldots\ldots(4)$$

In order to guarantee a correct rotation of points *P1'* and *P2'* we have to verify that the angle of rotation goes in the anti clockwise direction (positive rotation). To guarantee this issue, we can calculate the cross product between the unitary vector of the X-axis and the vector *AB*. The result of this cross product is a vector M that resides in the Z-axis. If the component of *M* is negative, we have an angle in the negative direction. The scalar value of vector M is (*y2 – y1*). So basically we evaluate (*y2 - y1*) and if this value is negative we substitute θ by $-\theta$ in Equations (1) to (4).

After obtaining the points of intersection P1 and P2 we can calculate the perimeter covered by a transmission. Figure 4 shows an example of how a receiver node A calculates the points of intersection P1 and P2 to obtain the angles θstart and θend. By having these angles, node A automatically knows the portion of its perimeter that has been covered by the transmission of node B. We show the procedure getPerimeter, this procedure is used to obtain angles θstart and θend. Also we show the procedures involved in getPerimeter.

```
BEGIN GetPerimeter(P1, P2, x1, x2)
    θstart = getAngle(P1y-y1, P1x-x1);
    θend = getAngle(P2y-y1, P2x-x1);
    θRef = getAngle(y2-y1, x2-x1);
    IF isInInterval(θRef, θstart, θend) = = 0 THEN
        temp = θstart;
        θstart = θend;
        θend = temp;
    ENDIF
END GetPerimeter
```

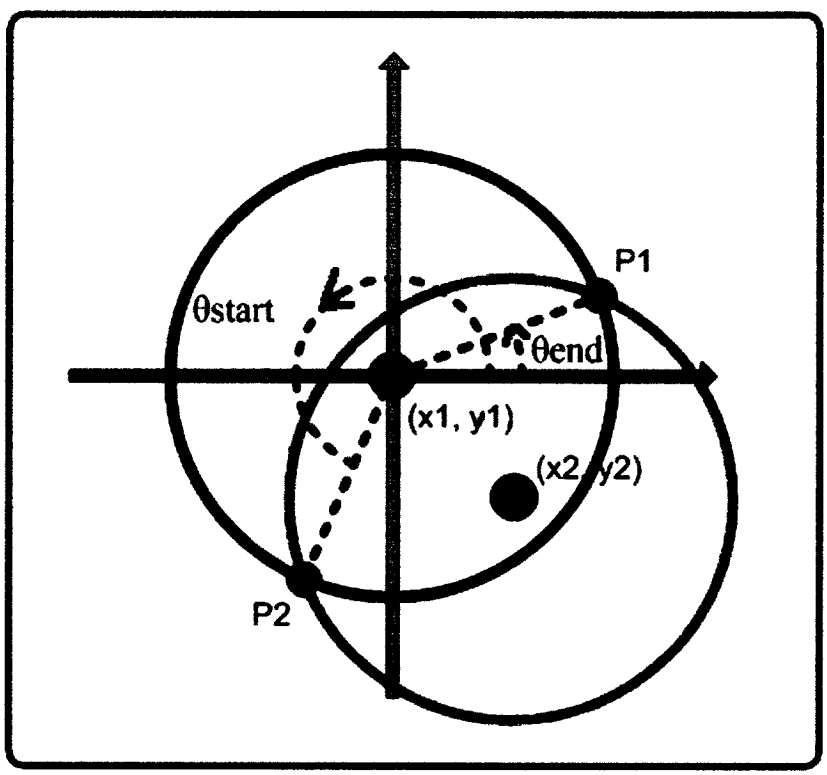

Figure 4. Perimeter intersection of two transmission patterns

BEGIN getAngle(y, x)

IF $x \neq 0$ AND $y \neq 0$ THEN

$$\theta = \frac{180}{\pi} \tan^{-1}\left(\frac{|y|}{|x|}\right);$$

IF $x < 0$ AND $y > 0$ THEN

$\theta = 180 - \theta;$

ELSEIF $x < 0$ AND $y < 0$ THEN

$\theta = 180 + \theta;$

ELSEIF $x > 0$ AND $y < 0$ THEN

$\theta = 360 - \theta;$

ENDIF

ELSEIF $y = = 0$ THEN

IF $x > 0$ THEN

$\theta = 0;$

ELSE

$\theta = 180;$

ENDIF

ELSEIF $x = = 0$ THEN

IF $y > 0$ THEN

$\theta = 90;$

ELSE

$\theta = 270;$

ENDIF

ENDIF

```
END getAngle

BEGIN isInInterval(reference, start, end)
    flag = false;
    IF (start < end AND reference >= start AND reference <= end) OR (start > end AND
    ((reference >= start AND reference <= 360) OR (reference >= 0 AND reference <=
    fin))) THEN
        flag = true;
    ENDIF
END isInInterval
```

3.4 Obtaining intervals of intersection between perimeter's sections of intersection.

We have described how to obtain the perimeter of intersection between two transmission patterns. However each node now has to cope with the objective of finding the intersection between multiple sections of perimeters. To solve this, we propose that each node has a list of independent intervals of intersection. At first, a receiving node *A* will have this list empty. When a first message arrives, node *A* uses getPerimeter to obtain θstart and θend. Node *A* creates its first interval *B* formed by θstart and θend and adds *B* to its list of independent intervals of intersection. In order to obtain the intersection of subsequent transmissions we use the process updateAllIntervals. This process updates the list of independent intervals of intersection (list_intervals) with the last created interval of intersection (new_interval). If after invoking updateAllIntervals, the list of independent intervals of intersection is empty, the corresponding node is covered and do not transmit. We present the pseudocode of updateAllIntervals and of twoIntervalsUpdate. The process twoIntervalsUpdate calculates the

number of intervals created by combining two intervals of intersection. The reader must observe that for this 2D case the maximum number of independent intervals of intersection is 2.

```
BEGIN updateAllIntervals( list_intervals,  new_interval)
    n = number of intervals in list_intervals;
    FOR (i = 1; i <= n; i++)
        actual_interval = list_intervals(i);
        [intervals_created, new_interval] = twoIntervalsUpdate(new_interval,
        actual_interval);
        IF intervals_created = = 1 THEN
            take actual_interval out from list_intervals;
            if (i = = n)
                add new_interval to list_intervals;
        ELSEIF intervals_created = = 0 THEN
            empty list_intervals;
            break;
        ELSEIF intervals_created = = 2 AND i = = n THEN
            add new_interval to list_intervals;
        ENDIF
    ENDFOR
END updateAllIntervals
```

```
BEGIN twoIntervalsUpdate(new_interval, actual_interval)
    θstart1 = new_interval->θstart;
    θend1 = new_interval->θend;
    θstart2 =  actual_interval->θstart;
    θend2 =  actual_interval->θend;
    flag1 = isInInterval(θstart1, θstart2, θend2);
    flag2 = isInInterval(θend1, θstart2, θend2);
    IF !flag1 AND !flag2 THEN
        number_created = 2;
        interval_created->θstart = θstart1;
        interval_created->θend = θend1;
    ELSEIF !flag1 AND flag2 THEN
        number_created = 1;
        interval_created->θstart = θstart1;
        interval_created->θend = θend2;
    ELSEIF flag1 AND !flag2 THEN
        number_created = 1;
        interval_created->θstart = θstart2;
        interval_created->θend = θend1;
    ELSEIF flag1 AND flag2 THEN
        IF isInInterval(θstart2, θstart1, θend1) THEN
            number_created = 0;
            interval_created->θstart = 0;
            interval_created->θend = 360;
```

```
        ELSE
            number_created = 1;
            interval_created->θstart = θstart2;
            interval_created->θend = θend2;
        ENDIF
    ENDIF
END twoIntervalsUpdate
```

Chapter 4 Area Based Beaconless Broadcasting Algorithms in 3D

4.1 Assumptions

Again we assume that every node in the network knows its geographic position. Our second assumption is that all nodes have the same transmission radius and use omni-directional antennas. We consider the transmissions patterns as spheres. Finally, nodes do not know where neighboring nodes are, so no beacons or hello messages are used.

4.2 Description of 3D Area based beaconless broadcasting version 1 (3D-ABBA1)

We have shown how to use the omni-directional pattern for the 2D case in order to have a beaconless algorithm. In 3D we will consider spheres. The idea behind this first version of 3D-ABBA is to use three projections planes, the planes XY, XZ, YZ. We used these planes to try to observe the intersections of the nodes transmission patterns (spheres) in the network. Basically, each node has to cover a transmission surface defined by a spherical pattern. By projecting the transmission spheres into the three planes, the nodes are able to apply 2D-ABBA in each plane. In 3D-ABBA1 the receiving nodes set a timer that is directly proportional to the three areas (one for each plane) covered by other nodes. The pseudo code for 3D-ABBA1 is as follows.

```
BEGIN 3d-abba1
    A source node S broadcasts a message M
    Each intermediate node L runs as follows with respect to the message M initiated by S
    {
        BEGIN wait until M is received for the first time
            Set perimeters covered in planes XY, XZ, and YZ;
            Set timeout;
            REPEAT
                BEGIN wait until another copy of M is received or timeout expires
                    IF another copy is received THEN
                        update perimeters in the planes XY, XZ, and YZ;
                        update timeout (if required by protocol);
                    ENDIF
                END wait until another copy of M is received or timeout expires
            UNTIL timeout expires;
            IF any of the three perimeters are still uncovered THEN
                Retransmit;
            ENDIF
            Ignore further copies of M
        END wait until M is received for the first time
    }
END 3d-abba1
```

For this 3D version the timeout function proposed is

$$timeout = \frac{degreesCoveredXY + degreesCoveredXZ + degreesCoveredYZ}{3}$$

The variables *degreesCoveredXY*, *degreesCoveredXZ*, *degreesCoveredYZ* refer to the angles in degrees of the circles that has been covered in their corresponding plane. In Figure 5 we show a 3D example of how a node receives a transmission from two nodes and how it updates its transmission pattern on the three planes, XY, XZ, and YZ. When a node gets completely covered it sets its timeout to 0 so it can decide immediately not to transmit.

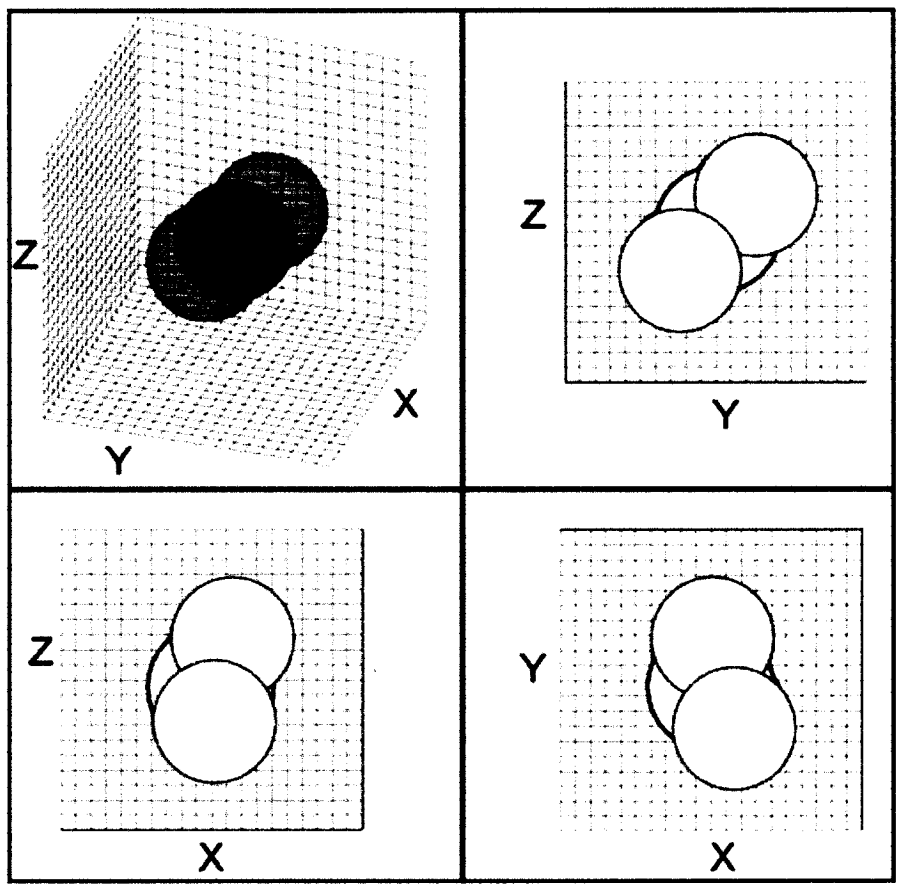

Figure 5. Example of 3D-ABBA1, the transmission perimeters are updated in the three projection planes.

3D-ABBA1 is a simple proposal of a 3D beaconless algorithm; however, the reader must be aware that 3D-ABBA1 can fail in some cases. It is obvious for the 2D case that the algorithm does not fail, however for this case it is not straightforward if it will fail or not. Figures 6 and 7 show a specific example where 3D-ABBA1 fails. In Figure 6 (up-left), the three nodes in red transmit the same message. The blue node receives the transmissions and updates its transmission sphere (up-right of Figure 6). We could think that the blue node is completely covered if we observe the projections shown in Figure 7; however, in the downright part of Figure 6 we can observe that part of the blue transmission sphere is still not covered.

The previous example showed a special case where 3D-ABBA1 fails, denoting its lack of reliability. Unfortunately this can question the viability of 3D-ABBA1. However as we will show in the simulation results section, 3D-ABBA1 was successful in all the tests.

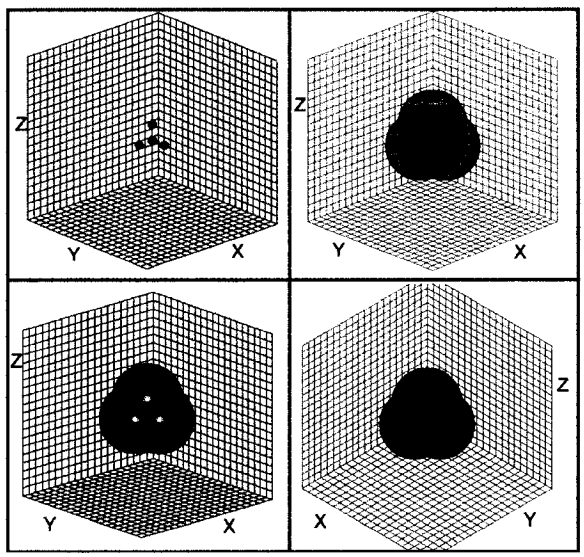

Figure 6. Different views of a receiving node (blue) updating its transmission pattern affected by the transmission nodes (red).

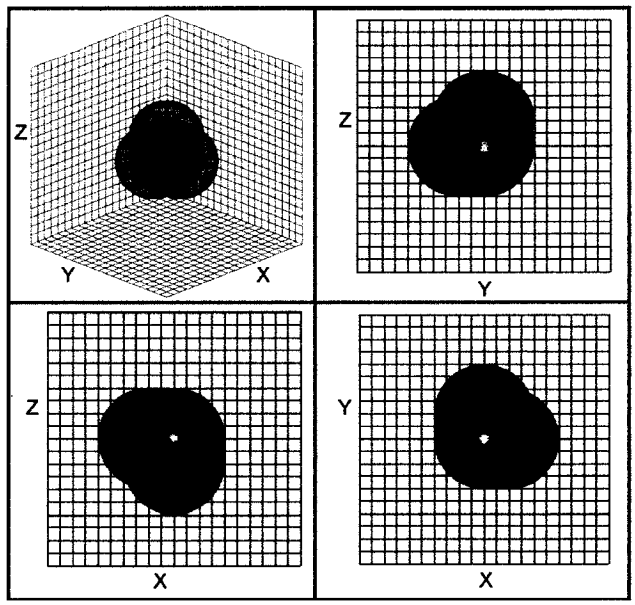

Figure 7. The three plane projections of the transmission patterns. Observe for this special case that the blue node thinks its transmission sphere has been covered.

4.3 Description of 3D Area based beaconless broadcasting version 2 (3D-ABBA2)

Again, the principle of our algorithm is the intersection between two transmission patterns. The basic idea of this second 3D algorithm is to find the intersection of two spheres. As we know, the intersection between two spheres on the surface of both spheres could be a circumference or a single point. In our studied case, the intersection is always a circumference because the receiving nodes are at most *R* (transmission radius) distance away from the transmitting nodes. Figure 8 shows the circumference intersection of two spheres and six selected points that reside in it: *P1, P2, P3, P4, P5,* and *P6*. Also, these points form a hexagon that resides in the same intersection plane of the two spheres. Receiving nodes are located at coordinates (*x1, y1, z1*), and transmitting nodes are located at coordinates (*x2, y2, z2*). Consider the points (hexagon) depicted in Figure 8, if each such point is located inside

another transmitting sphere, then the considered sphere (the one that created the hexagon) is covered in full. More precisely, let (d, e, f) be the coordinates of an intersection point. There must exist another sphere with center coordinates (xi, yi, zi) such that $(d - xi)^2 + (e - yi)^2 + (f - zi)^2 \leq R^2$. The reader must note that the considered sphere (centered at $(x1, y1, z1)$) is not considered for this coverage. It is its volume that needs to be covered by others.

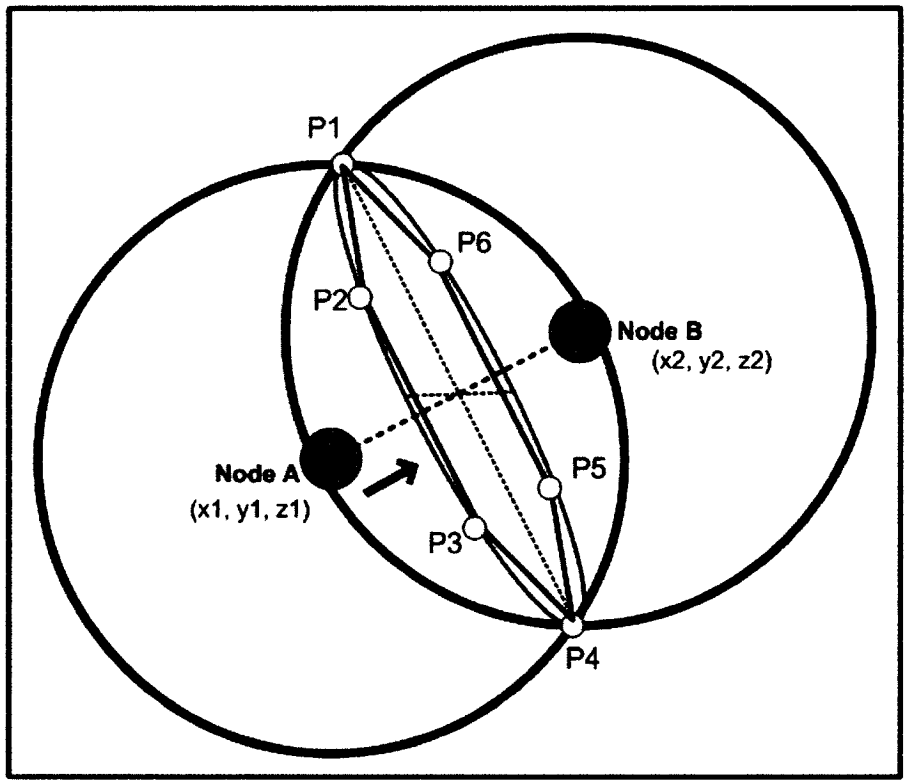

Figure 8. Circumference of intersection formed by the two transmission spheres of nodes *A* and *B*.

Therefore a receiving node keeps a list of all the generated points. Whenever a new transmission is received the following happens:

1 Some existing points are inside the new transmission sphere. These points are eliminated. If no point remains, the sphere is completely covered.

2 The new sphere is considered, precisely the new 6 points of the hexagon. Each such point is tested whether it is inside another existing sphere (spheres of previous transmitters). Those that are inside another one are ignored. Those that are outside any existing ball are entered into the list of points to be covered.

We propose a timeout function that depends on the number of already heard transmissions, number of points to be covered, and an additional parameter. This timeout function is described in a later section.

4.3.1 Obtaining points *P1, P2, P3, P4,* and P6

From Figure 8, we can observe that the medium point (*Pm*) of the line of minimum distance between nodes *A* and *B* lies in the plane of intersection of the two transmitting spheres:

$$Pmx = \frac{x1 + x2}{2};$$

$$Pmy = \frac{y1 + y2}{2};$$

$$Pmz = \frac{z1 + z2}{2};$$

Also, we can obtain the vector *APm*, which is

$$APmx = Pmx - x1;$$

$$APmy = Pmy - y1;$$

$$APmz = Pmz - z1;$$

We propose a vector *Q* of size '*a*'; where '*a*' is the magnitude of vector *APm*, or half the distance between nodes *A* and *B*, to be in the *x* coordinate axis. In other words, vector *Q* has

components [a 0 0]. Also, from Figure 8 we can observe that the distance 'b' between Pm and any of the intersection points is $b = \text{sqrt}(R^2 - a^2)$.

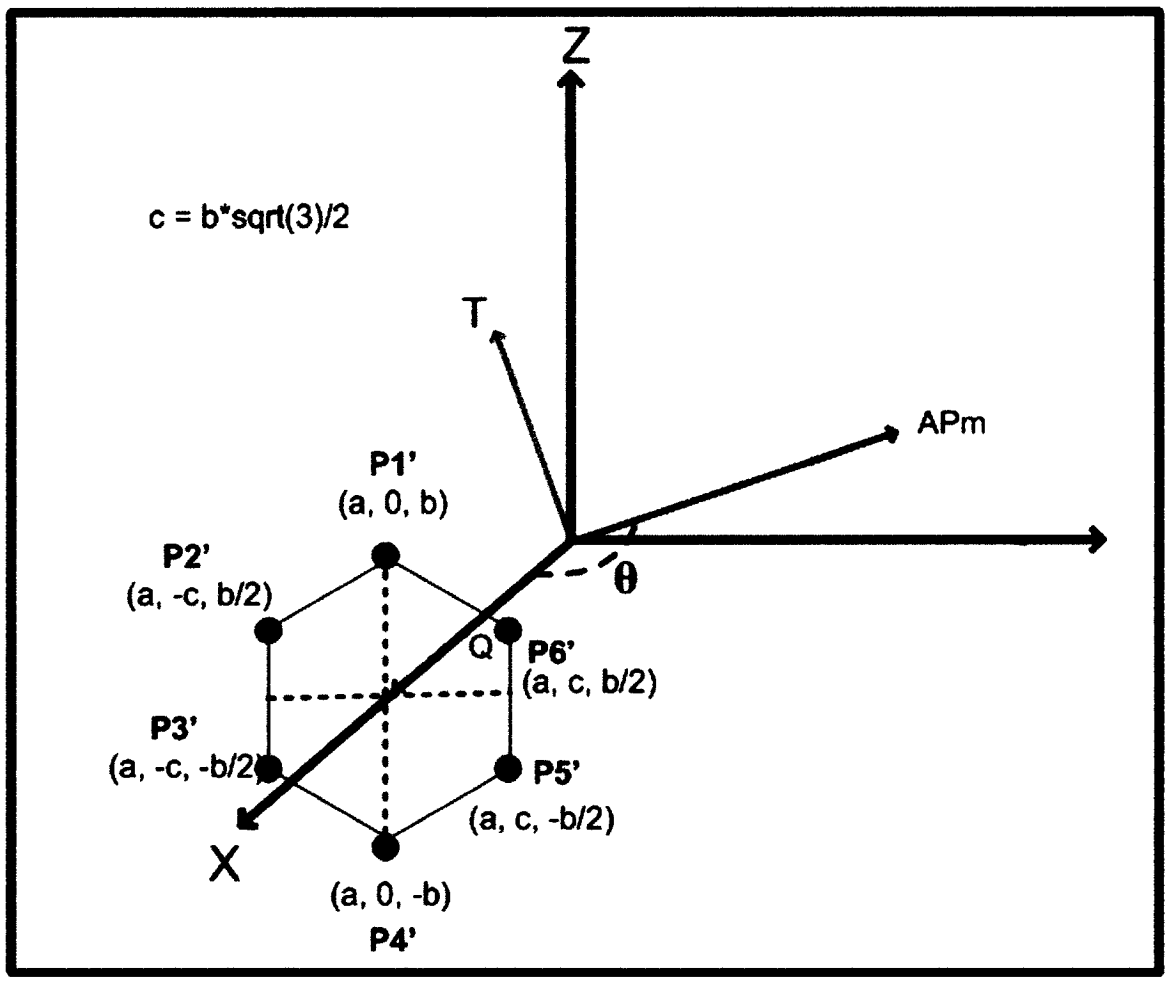

Figure 9. Vectors *APm, Q,* and *T.* Points *P1', P2', P3', P4', P5', and P6'.* Angle θ between vectors APm and Q.

We propose to obtain a vector T (orthogonal to APm and Q), and the angle θ between vector APm and vector Q. The idea is that in order to obtain any point Pi (where i = 1, 2, …, 6), we can apply a rotation θ to Pi' around the vector T and finally a translation (x1, y1, z1). In Figure 9 we show this idea.

In order to apply a rotation around an arbitrary vector we can use quaternions. Quaternions are mathematical entities that help us to represent 3D rotations. They have 4 dimensions, one real dimension and 3 imaginary dimensions. Each of these imaginary dimensions has a unit value of the square root of -1, but they are different square roots of -1 all mutually perpendicular to each other, known as i, j and k. So a quaternion can be represented as: a + i b + j c + k d. It is quite difficult to give a physical meaning to a quaternion, it is just a quantity which represents a rotation. This can be written as:

$$q = \cos\frac{\theta}{2} + i\left(x * \sin\frac{\theta}{2}\right) + j\left(y * \sin\frac{\theta}{2}\right) + k\left(z * \sin\frac{\theta}{2}\right) \ldots\ldots\ldots(5)$$

Where:

- θ = angle of rotation
- x, y, z = vector representing axis of rotation

Knowing this, we can use a quaternion to calculate the rotated point from the original position of the point; this allows us to rotate points without using matrices of rotation. The representation of this is as follows:

$$P\_rot = q*P\_orig*q' \ldots\ldots\ldots(6)$$

Where:

- P_orig = original position of point in the following format 0+ix+jy+kz
- P_rot = resulting position of point after rotation in the following format 0+ix+jy+kz
- q = rotation quaternion = qw + i qx + j qy + k qz
- q' = conjugate of q = qw - i qx - j qy - k qz

In other words, the elements are denoted by x, y and z for the position and qw, qx, qy and qz for the quaternion representing the transform. So putting these elements into formula (6) we obtain the following equations:

$$P\_rot_X = (qw^2 * P\_orig_X) + (2 * qy * qw * P\_orig_Z) - (2 * qz * qw * P\_orig_Y) + (qx^2 * P\_orig_X) + (2 * qy * qx * P\_orig_Y) + (2 * qz * qx * P\_orig_Z) - (qz^2 * P\_orig_X) - (qy^2 * P\_orig_X) \quad ..(7)$$

$$P\_rot_Y = (2 * qx * qy * P\_orig_X) + (qy^2 * P\_orig_Y) + (2 * qz * qy * P\_orig_Z) + (2 * qw * qz * P\_orig_X) - (qz^2 * P\_orig_Y) + (qw^2 * P\_orig_Y) - (2 * qx * qw * P\_orig_Z) - (qx^2 * P\_orig_Y) \quad ..(8)$$

$$P\_rot_Z = (2 * qx * qz * P\_orig_X) + (2 * qy * qz * P\_orig_Y) + (qz^2 * P\_orig_Z) - (2 * qw * qy * P\_orig_X) - (qy^2 * P\_orig_Z) + (2 * qw * qx * P\_orig_Y) - (qx^2 * P\_orig_Z) + (qw^2 * P\_orig_Z) \quad ..(9)$$

To obtain qw, qx, qy, and qz we need to obtain vector T and the angle of rotation θ. Vector T needs to be orthogonal to Q and APm and also it has to be unitary. We can obtain the unitary vector of the cross product between Q and Apm. Also from the dot product between Q and APm we can obtain the angle between them. Equations (10) and (11) show this result.

$$T = \frac{Q * APm}{|Q * APm|} \quad \ldots\ldots (10)$$

$$\theta = \cos^{-1}\left(\frac{Q \cdot APm}{|Q||APm|}\right) \ldots\ldots (11)$$

Finally we substitute the results from Equations (10) and (11) into Equation (5) to obtain the values of qw, qx, qy, and qz. Now we have all the values needed to obtain P_rot, so in order to obtain any point Pn of the hexagon that is in the plane of intersection between two transmission spheres we apply the following operation:

$$P_n = [P\_rot_x, P\_rot_y, P\_rot_z] + [x1, y1, z1] \ldots\ldots (12)$$

It is important to clarify two special cases when we want to obtain T. The two cases are when $\theta = 0$ and when $\theta = 180°$. For the first case, we don't need a rotation so our vector T is [0, 0, 0]. For the case of $\theta = 180°$ we can simply analyze where does vector APm has to be positioned to have an angle of 180° with vector Q. We know in advanced that Q is in the positive direction of X-axis, so if we want APm to have 180° with vector Q, APm has to be in the negative direction of X. For this case vector T can be any unitary vector residing in the plane YZ, the simplest vectors are the ones having the direction of +Y, -Y, +Z, or –Z. For our implementation we used T = [0, 1, 0].

4.3.2 Timeout function for 3D-ABBA2

The timeout function in this version of the algorithm is fundamental. The receiving nodes cannot differentiate a received transmission that covers more volume than another one. When a node receives a transmission, it only adds the number of received transmissions and the number of intersection points left to be covered. For example, when the algorithm begins, a node starts the transmission, and all the neighboring nodes receive the message, all of these nodes have one received transmission and six (hexagon) points, so all these nodes have the same timeout value and all will transmit almost at the same time. In 2D-ABBA and 3D-ABBA1, the timeout function depended on perimeters covered. So how can we evaluate with 3D-ABBA2 the received transmissions in order to have a good timeout function? If we add an extra parameter to each of the points, we will be able to reach better results. This extra parameter has to be directly proportional to the volume covered by the transmission. It is obvious that the volume covered by the transmitters depends directly on the distance between the sender and the receiver. For this reason, we decided that each intersection point has to have a weight $\frac{d}{R}$. The parameter d is the distance between the transmitter that created the point and the receiver. The parameter R is the transmission radius. In conclusion, we propose a timeout function:

$$Timeout = (\text{number of received transmissions}) + IntersectFunction$$

$$IntersectFunction = \text{sum of the weights of the points still to be covered}$$

4.3.3 Reliability of 3D-ABBA2

Until now we have tried to propose reliable broadcasting algorithms. For the 2D case we were able to obtain this objective; however, as we showed in a previous example, 3D-

ABBA1 can fail. Unfortunately 3D-ABBA2 can also fail in very special cases. In Figure 10 we show an example when this occurs. The reader can observe how all the intersection points of the generated hexagons are covered, although, part of the transmission sphere of the receiver is not completely covered. This clearly shows that although it is not frequent, 3D-ABBA2 fails. One way of trying to resolve the problem would be to generate more than six points; however no matter how many points we add, the reliability cannot be guaranteed. The only way to be sure that 3D-ABBA3 will not fail is to consider the whole circumference of intersection. This idea is shown later in 3D-ABBA4.

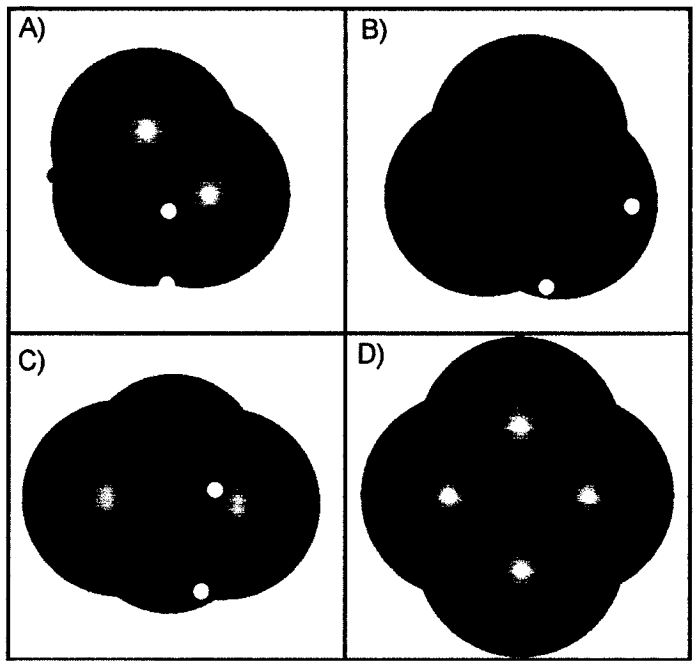

Figure 10. Example when 3D-ABBA2 fails (intersection points of the same color are created from the same sphere). Section A shows how two transmissions cover part of the sphere; the receiving node has some points to be covered from each sphere. Sections B and C shows three and four transmissions respectively with some points still to be covered. Section D shows that all points are covered, however part of the receiving sphere still is not covered.

4.4 Description of 3D Area based beaconless broadcasting version 3 (3D-ABBA3)

For 3D-ABBA3 we consider applying three spheres of intersection instead of two. Figure 11 shows an example of the geometry formed by the intersection of 3 transmission spheres. The result of this intersection (if it exists) is two points. Consider the points of intersection depicted in Figure 11, if each such intersection point is located inside another transmitting sphere, then the considered sphere is covered in full. More precisely, let (d, e, f) be the coordinates of an intersection point. There must exist another sphere with center coordinates (xi, yi, zi) such that $(d - xi)^2 + (e - yi)^2 + (f - zi)^2 \leq R^2$.

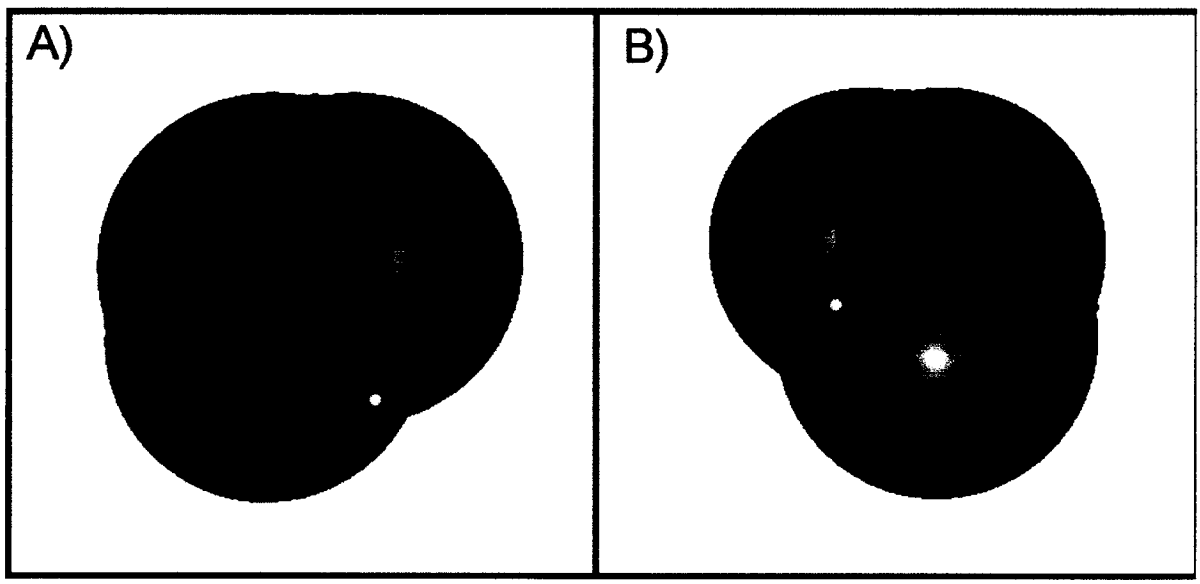

Figure 11. Receiving node updates its transmission pattern (blue sphere), taking into account the two received transmissions (red spheres). Note the two intersection points (yellow) formed by the intersection of the three spheres.

Therefore each receiving node keeps a list of all intersection points. Whenever a new node transmits a message in the neighborhood then two things happen:

1. Some existing intersection points are inside it. These points are eliminated. If no intersection point remains, the sphere is covered.
2. The new sphere is considered, with the current fixed sphere, and with all previously transmitting spheres, for triples to find new intersection points. Each such point is tested whether it is inside another existing sphere. Those that are inside are ignored. Those that are outside any existing sphere are entered into a list of intersection points to be covered.

In this proposal we involve the intersection of three transmission spheres; however, we can have some cases in which the intersection points (on the surface) between the three spheres do not exist. Figure 12 shows an example of this case.

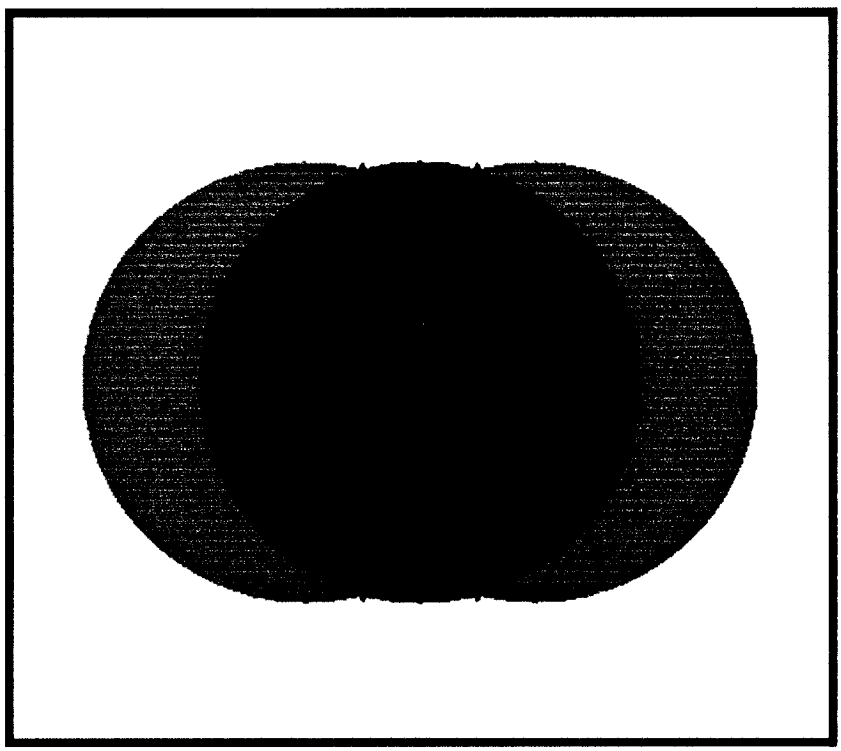

Figure 12. Case where three transmission spheres do not have two intersection points.

Basically we have two cases: three spheres intersection case and no intersection case. In order to cope with both cases we propose the following algorithm.

```
BEGIN 3d-abba3
    A source node S broadcasts a message M
    Each intermediate node L runs as follows with respect to the message M initiated by S
    {
        BEGIN wait until M is received for the first time from any node B
            setTimeout(L, B);
            REPEAT
                BEGIN wait until another copy of M is received or timeout expires
                    IF another copy is received from a node C THEN
                        updateTimeout( L, C);
                    ENDIF
                END wait until another copy of M is received or timeout expires
            UNTIL timeout expires;
            IF list of intersection points of L is not empty OR list of singles of L is not empty
            THEN
                Retransmit;
            ENDIF
            Ignore further copies of M;
        END wait until M is received for the first time
    }
END 3d-abba3
```

BEGIN setTimeout(A, B)

$$timeout = \frac{\text{distance}(A, B)}{R};$$

Node A adds node B to its list of senders;

Node A adds node B to its list of singles;

END setTimeout

BEGIN updateTimeout(J, I) (J = receiver, I = sender)

FOR each point B in the list of intersection points of J

IF distance(B, I) $\leq R$ THEN

Take out B from the list of intersection points of J;

ENDIF

ENDFOR

FOR each node N in the list of senders of J

Calculate points of intersection between spheres N, J and I;

FOR each point of intersection $K_{N,J,I}$

IF N is in the list of singles of J THEN

take N out of the list of singles of J;

ENDIF

IF $K_{N,J,I}$ is inside a sphere P AND $P \neq N$ AND $P \neq J$ AND $P \neq I$ THEN

Point $K_{N,J,I}$ is ignored;

ELSE

J inserts $K_{N,J,I}$ to its list of intersection points;

$$\text{timeout}_J = \text{timeout}_J + \frac{\text{distance}(J,I) + \text{distance}(J,N)}{2R};$$

ENDIF

ENDFOR

ENDFOR

IF node I did not generate any intersection point with any of the N nodes from the list of senders of J THEN

Node J puts node I in its list of singles;

$$\text{timeout}_J = \text{timeout}_J + \frac{\text{distance}(J,I)}{R};$$

ENDIF

IF list of intersection points of J is empty AND list of singles of J is empty THEN

$\text{timeout}_J = 0;$

ENDIF

END updateTimeout

4.4.1 Obtaining intersection points between three transmission spheres

In 3D, two planes π_1 and π_2 are either parallel or they intersect in a single straight line **L** (See Figure 13). Let π_i (i = 1,2) be given by a point V_i plus a normal vector $\mathbf{n}_i$, and have an implicit equation: $\mathbf{n}_i \cdot P + d_i = 0$. The planes π_1 and π_2 intersect in a line if $\mathbf{n}_1 \times \mathbf{n}_2$ is different of zero. For our specific case when we take into account three spheres (1 receiving and 2 transmitting) we are able to obtain the equations of two planes of intersection.

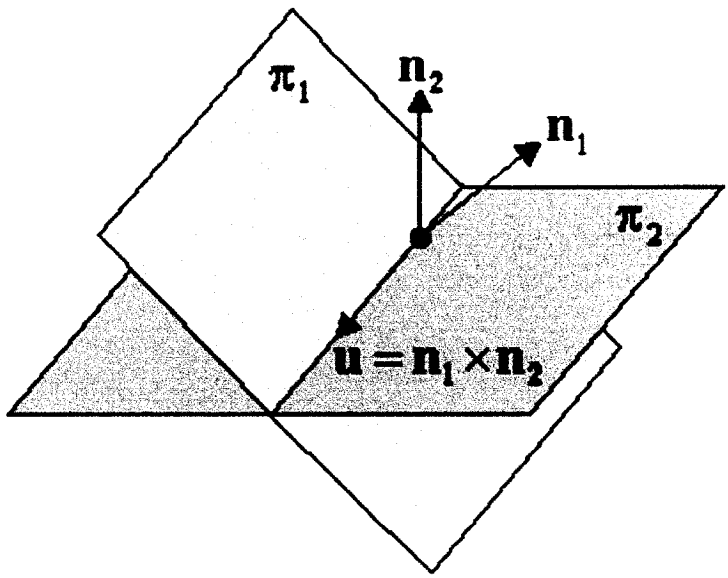

Figure 13. Intersection between two non-parallel planes

We define the two planes with normals n as:

$$n_1 \cdot P = d_1$$

$$n_2 \cdot P = d_2$$

The equation of the line of intersection can be written as:

$$P = c_1 n_1 + c_2 n_2 + u n_1 * n_2$$

Where "*" is the cross product, "·" is the dot product, and u is the parameter of the line. Taking the dot product of the above with each normal gives two equations with unknown variables c_1 and c_2.

$$n_1 \cdot P = d_1 = c_1 n_1 \cdot n_1 + c_2 n_1 \cdot n_2$$

$$n_2 \cdot P = d_2 = c_1 n_1 \cdot n_2 + c_2 n_2 \cdot n_2$$

Solving for c_1 and c_2

$$c_1 = \frac{d_1 n_2 \cdot n_2 - d_2 n_1 \cdot n_2}{\text{determinant}}$$

$$c_2 = \frac{d_2 n_1 \cdot n_1 - d_1 n_1 \cdot n_2}{\text{determinant}}$$

$$\text{determinant} = (n_1 \cdot n_1)(n_2 \cdot n_2) - (n_1 \cdot n_2)^2$$

After solving the equation system we obtain the vector equation of the line of intersection between the two planes formed by the transmission spheres.

$$P = a + ub$$

Or in its parametric form:

$$x = a_x + ub_x$$

$$y = a_y + ub_y$$

$$z = a_z + ub_z$$

Also we know that the two intersection points are at a distance R from any of the three spheres involved. Taking the coordinates (x_A, y_A, z_A) of the receiving node A we have:

$$[x_A - (a_x + ub_x)]^2 + [y_A - (a_y + ub_y)]^2 + [z_A - (a_z + ub_z)]^2 = R^2$$

Developing the equation we have:

$$\begin{aligned}&(b_x^2 + b_y^2 + b_z^2)u^2 + 2[b_x(a_x - x_A) + b_y(a_y - y_A) + b_z(a_z - z_A)]u + \\ &[x_A^2 + y_A^2 + z_A^2 + a_x^2 + a_y^2 + a_z^2 - 2(x_A a_x + y_A a_y + z_A a_z) - R^2] = 0\end{aligned} \quad \ldots (13)$$

Finally, by solving this second-degree equation we obtain the two intersection points of the three transmission spheres. So in summary, receiving nodes perform the following steps.

- Obtain planes of intersection.
- Test if planes are not parallel (cross product between plane's normals is different from zero).
- If the plane is parallel there are no intersection points.
- If not parallel, solve Equation 13 to obtain the intersection points.
- If the solutions are not real the planes do not generate intersection points in the surface of the transmission sphere.

4.4.2 Reliability of 3D-ABBA3

A node in 3D-ABBA3 will not transmit only if it is covered by other transmissions. In order to test coverage, a receiving node *W* looks its list of single nodes (transmitting spheres that do not intersect with other transmitting spheres) and its list of intersection points (points generated by the receiving sphere and two transmitting spheres). If both lists are empty before the timeout expires, node *W* will not transmit. The only way 3D-ABBA3 would fail is if *W* has its singles and its intersection points lists empty, but its transmission sphere is still not completely covered.

At first, both, the single's and intersection point's lists are empty. When the first transmission arrives at a receiver node *W*, the single list of *W* has an entry and the intersection point list remains empty. When *W* receives more transmissions two cases can arise. The first case is where transmission spheres do not intersect with each other (Figure 12); in this case the singles list has some values and the only way to become empty is by the

generation of intersection points. Thus, if no intersection points are generated, the singles list will never reach emptiness and by consequence W will transmit. For the second case, W generates intersection points with other transmission spheres. In order to fail, the list of intersection points of W has to be empty but its transmission sphere still not covered. Figure 14 shows an example of how the intersection points are generated. It is easy to observe from this example that as long as there exists an uncovered surface, there will exist intersection points. In other words if there exists intersection points there exists an uncovered surface, thus in order to cover completely the transmission sphere there has to be no intersection points. This guarantees that 3D-ABBA3 achieves complete reliability. The reliability follows from the following theorem.

Theorem 1. Suppose that node A received transmissions from nodes C_1, C_2,..., C_m. Consider all intersection points X of 3D perimeters of spheres centered at three nodes: A, C_i and C_j. If there exists at least one such intersection point and every such intersection point X is located (strictly) inside at least of one of the remaining spheres, centered at C_k then the sphere centered at A is fully covered by spheres centered at C_1, ..., C_m, and node A does not need to retransmit, without affecting the reliability of broadcasting.

Proof. Note that all the considered intersection points and all intersection circumferences (between sphere centered at A and any of spheres centered at C_i for any i) are located on 3D perimeter of sphere centered at A. The conditions of the theorem are then topologically equivalent to an analogous theorem for the plane, applied, for example, in [26] for sensor area coverage problems. Both theorems use circles as closed curves that are intersected, but the circles are located in the plane and on a 3D perimeter (sphere), respectively. The proof, in its simplified form, is by contradiction. Assume that the condition of the theorem is satisfied, but there is an area (on the considered 3D perimeter) that is still not covered. Let Y

be a point in this area. Find the closest curve (circle) to Y from the set. This curve separates covered and uncovered regions. Follow this curve until another curve is met. The meeting point is an intersection point from the theorem. It is easy to see that this intersection point is not strictly inside other curve (circle), which is a contradiction.♦

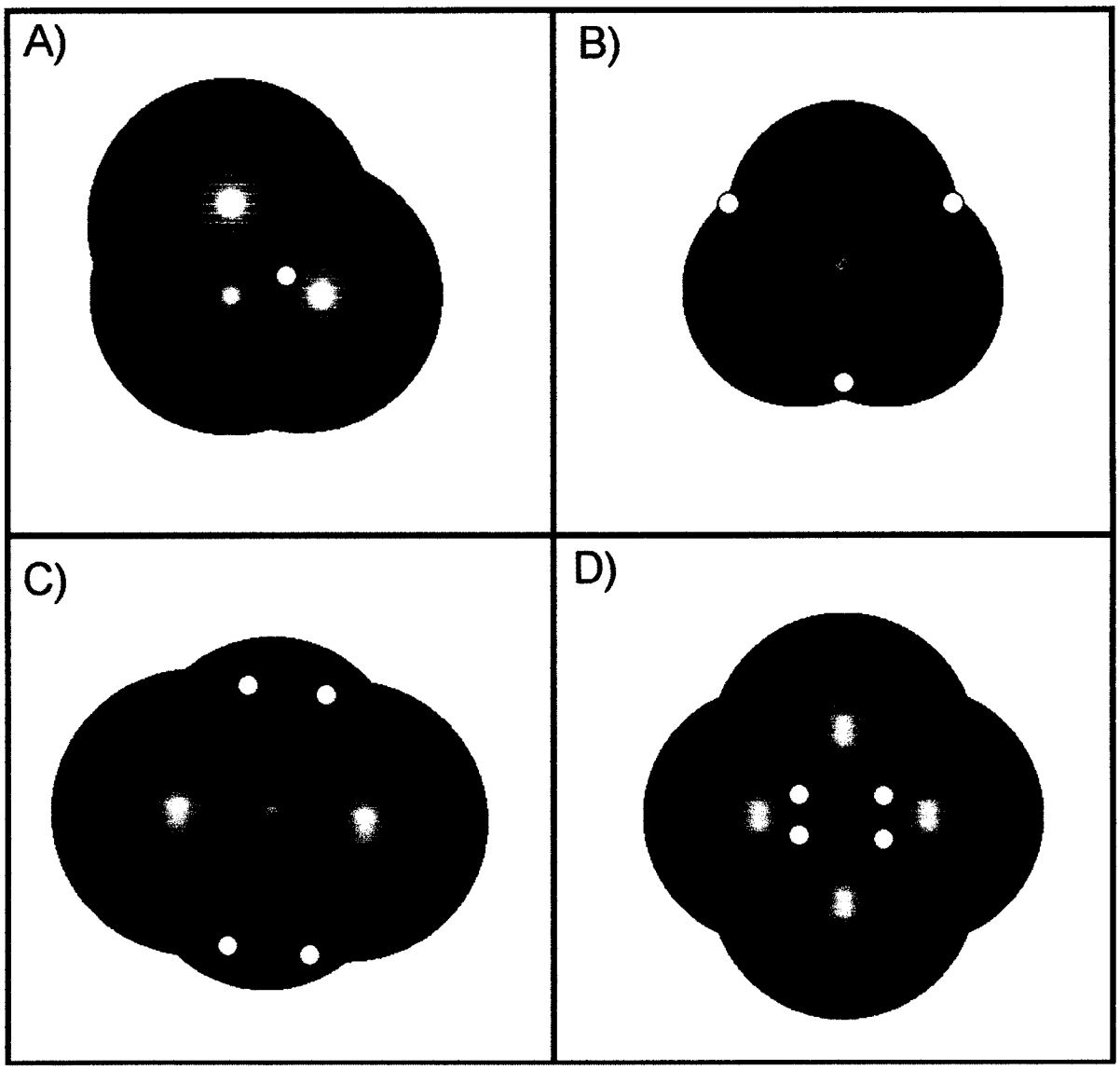

Figure 14. Intersection points created during 3D-ABBA3. The creation of this points guarantees 3D-ABBA3 reliability.

4.5 Description of 3D area based beaconless broadcasting version 4 (3D-ABBA4)

For 3D-ABBA we propose a method that guarantees 100% reliability, as in 3D-ABBA3. Our algorithm is based on covering the circles of intersection generated by transmitters and receivers. The idea is simple; each node *V* upon receiving a message from node *W* calculates the circumference of intersection *VW* between the two transmission patterns. Node *V* calculates the portion of the circle *VW* that is covered by previously received transmissions and also updates the portions of the circles from those transmissions that will be covered by *W*. If there are no portions of the circles uncovered, node *V* is completely covered and will not transmit. As in previous proposed versions of ABBA node *V* has a timer, if the timer expires before being covered it will transmit. We present the pseudo code of the idea behind 3D-ABBA4.

```
BEGIN 3d-abba4
    A source node S broadcasts a message M
    Each intermediate node L runs as follows with respect to the message M initiated by S
    {
        BEGIN wait until M is received for the first time from any node B
            setTimeout(L, B);
            REPEAT
                BEGIN wait until another copy of M is received or timeout expires
                    IF another copy is received from a node C THEN
                        setTimeout( L, C);
                    ENDIF
                END wait until another copy of M is received or timeout expires
            UNTIL timeout expires;
```

IF its list of circles of intersection is not empty THEN

Retransmit;

ENDIF

Ignore further copies of M;

END wait until *M* is received for the first time

}

END 3d-abba4

BEGIN setTimeout(*I, J*) (*J* = receiver node, *I* = sender node)

C = circle of intersection between node *J* and node *I*;

FOR each circle of intersection *B* in the list of *J*

updatePerimeter(*C, B*);

IF *B* is completely covered THEN

Node J takes out *B* from its list of circles of intersection;

ENDIF

ENDFOR

IF *C* is not completely covered THEN

Node J inserts *C* into its list of circles of intersection;

θ = angle of *C* not covered;

$$timeout = timeout + \theta \frac{\text{distance}(I,J)}{R};$$

ENDIF

IF the list of circles of intersection of *J* is empty THEN

$timeout = 0$;

ENDIF

END setTimeout

For the update of the perimeters between two circles of intersection, there exist some cases. These cases are shown in Figure 15.

Case 1: Intersection between the two circles.

Each perimeter is updated

Case 2: No intersection between the two circles

For this case there exist two cases.

Case 2.I: One circle completely covers the other

If circle A is between the receiving node and circle B, then circle B is covered by circle A. In this case A can be the new generated circle by the last transmission and B a circle of previous transmissions, or A can be a circle of a previous transmission and B the new generated circle.

Case 2.II. The circles have no relation

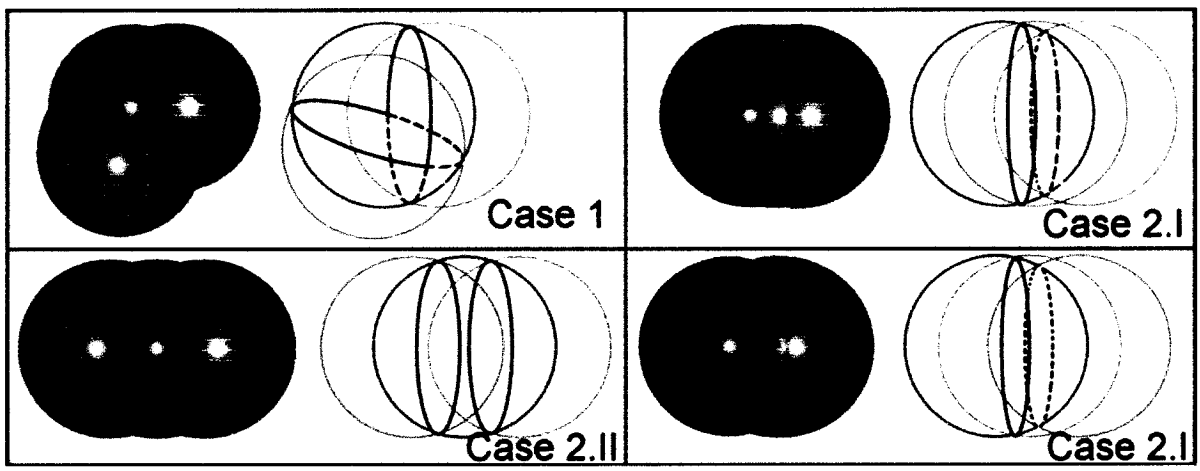

Figure 15. Update of intersection circles between transmission spheres. Blue spheres represent the transmission pattern of the receiving nodes. Green spheres represent previous received transmission patterns. Red Spheres represent last received transmission pattern.

4.5.1 Updating circumferences of intersection in 3D-ABBA4

The updating of perimeters of the circumferences in 3D-ABBA4 is not as trivial as the 2D case. As mentioned in the previous section, there exist different cases for the process of updating the circumferences of intersection. Before showing how to cope with each case, we will explain how we can define a circle of intersection. Since transmitting nodes create the circles of intersection, a receiving node can define each circle with its corresponding transmitter node. When a receiver node *A* gets a transmission from a node *B,* to define the circle of intersection created by *B*, node *A* defines this circle by the coordinates of its creator (node *B*) and by the perimeter still not covered. In Figure 16 we show two cases where node *A* defines the circles of intersection created by transmitting nodes. In Figure 16.A, node *A* receives a transmission from node *B* and defines the circle of intersection. The circle for this first example is composed by the coordinates of node *B* and by the perimeter still not covered. The perimeter still not covered will be defined with a θstart and a θend. For this example θstart = 0 and θend = 360. In Figure 16.B we show a second example. Node *A* now receives a second transmission. This transmission comes from node *C*. Node *A* defines this circle with the coordinates of *C* and by the θstart and the θend. Later we will explain how to obtain these angles.

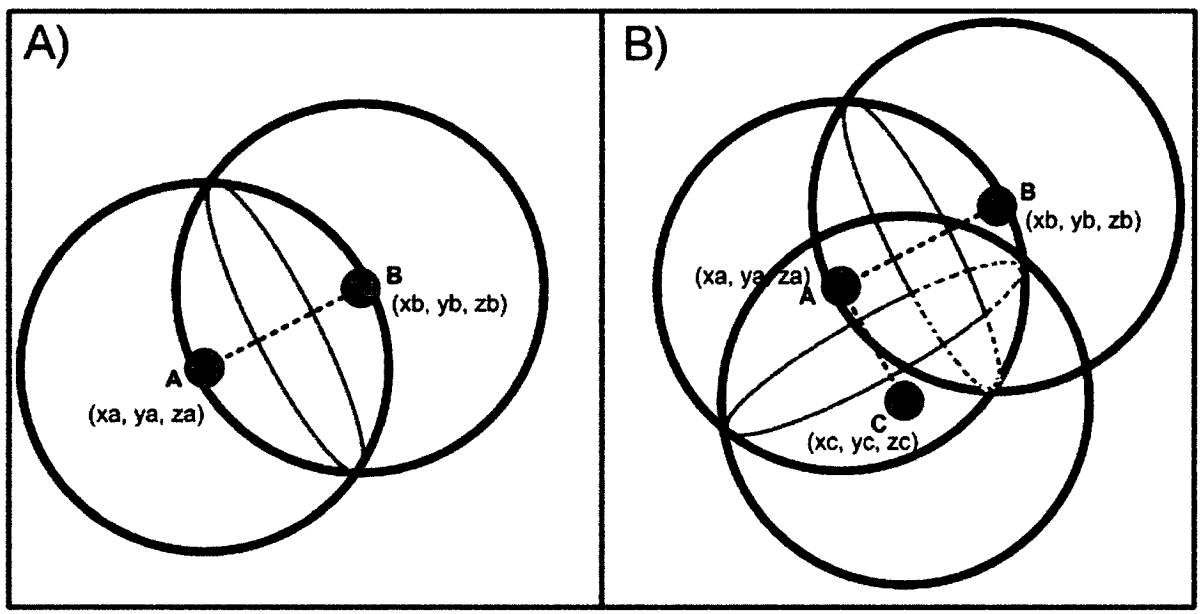

Figure 16. Circles of intersection created by transmitting nodes. Receiving nodes define these circles with the coordinates of the transmitter and with the perimeter still not covered.

The first step that receiving nodes perform in 3D-ABBA4 is to check if there exist intersection points between transmission spheres. Basically each receiving node performs the following steps when a new transmission is received.

- Obtain planes of intersection.
- Test if planes are not parallel (cross product between normals of the planes is different from zero).
- If the plane is parallel there are not intersection points.
- If not parallel, solve Equation 13 to obtain the intersection points.
- If the solutions are not real, the planes do not generate intersection points in the surface of the transmission sphere.

When there exist no intersection points we have case 2.I or case 2.II shown in Figure 15. Suppose we have three nodes: *A*, *B*, and *C*. Node *A* receives transmissions from nodes *B* and

C. For this example, the two intersection circumferences do not create intersection points. Node *A* obtains two vectors, vector *AB* and vector *AC*. Finally node *A* obtains the angle γ between this two vectors. If $\gamma > \frac{\pi}{2}$ then node *A* knows that the circumferences are independent and have no relation (case 2.II). However, if $\gamma < \frac{\pi}{2}$ then one circle covers the other and we have case 2.I. For this case node *A* will keep the circle whose creator is nearer to it. In other words:

IF distance(*A,B*) < distance(*A,C*) THEN

Node *A* keeps the circle created by *B* and drops the circle created by *C*;

ELSE

Node *A* keeps the circle created by *C* and drops the circle created by *B*;

ENDIF

Now we will show how to solve case 1. When a receiving node *A* wants to update the perimeter of a circumference whose creator was node *B* with another circumference created by node *C,* we propose the following steps. Figure 17 helps to follow these steps.

- Calculate the points of intersection P1(x1, y1, z1) and P2(x2, y2, z2) (between the circumferences created by B with A, and C with A) with Equation 13.
- Calculate the points P1' and P2'. In order to obtain them we apply a negative translation of P1 and P2. This negative translation is formed by node's *A* coordinates (xA, yA, zA). Pi' = (xi-xA, yi-yA, zi-zA).
- Calculate the medium point (*Pm*) of the line of minimum distance between nodes *A* and *B,* in order to obtain vector *APm.*

- Obtain a vector Q of size 'a'; where 'a' is the magnitude of vector APm, or half the distance between nodes A and B, to be in the z coordinate axis. In other words, the vector Q has components [0 0 a].
- Obtain a vector T (orthogonal to APm and Q), and the angle θ between vector APm and vector Q. We use Equations 10 and 11 to obtain T and θ. The idea is that in order to obtain any point Pi'', we can apply a rotation -θ to Pi' around the vector T.

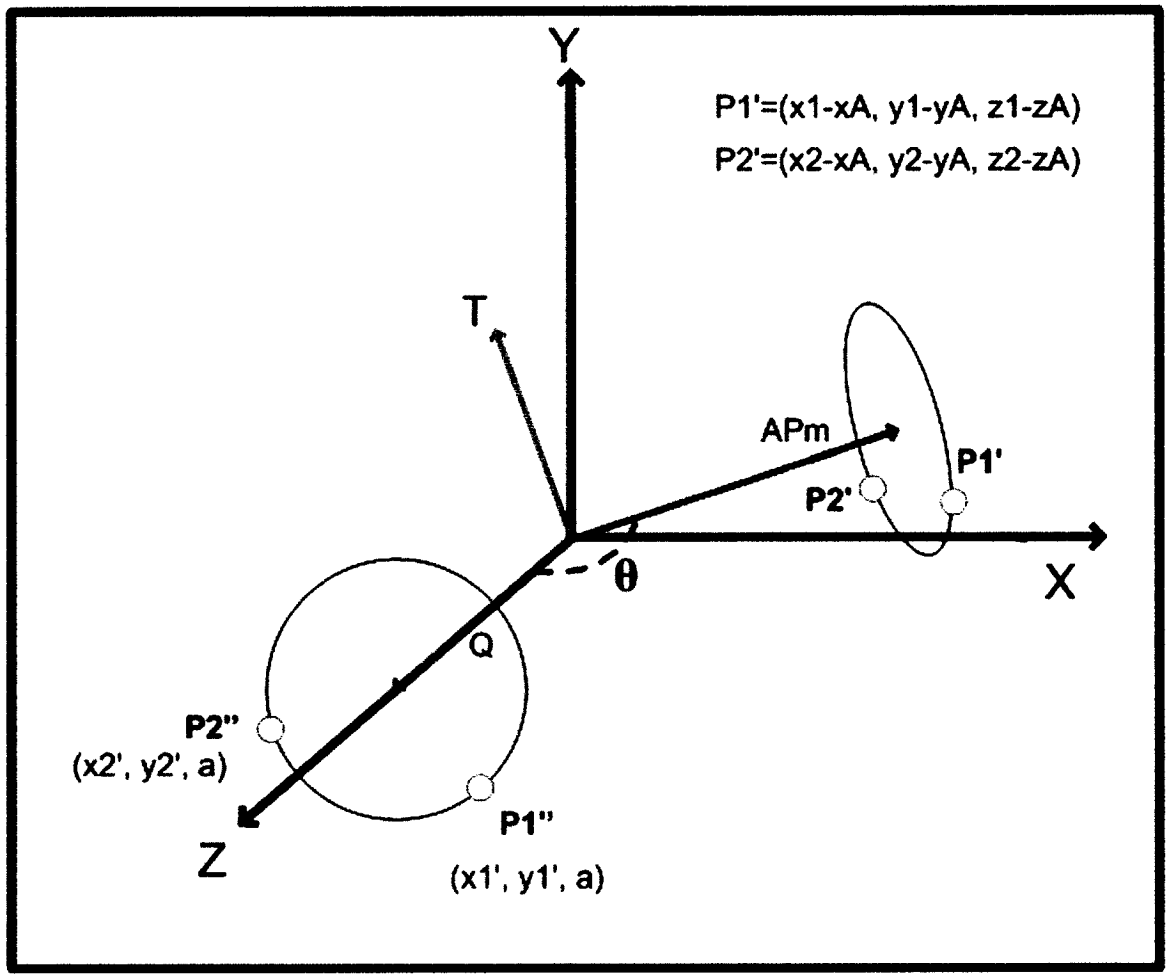

Figure 17. Points P1'' and P2'' obtained by the rotation around vector T of points P1' and P2'.

In order to apply a rotation of $-\theta$ around T we will use again quaternions. We substitute the results of T and $-\theta$ into Equation (5) to obtain the values of qw, qx, qy, and qz. Finally we substitute these values in Equation (7), (8), and (9). So any point Pi'' results in: P_i''= [P_rot_x, P_rot_y, P_rot_z].

It is important to clarify two special cases when we want to obtain T. The two cases are when $\theta = 0$ and when $\theta = 180°$. For the first case, we don't need a rotation so our vector T is [0, 0, 0]. For the case of $\theta = 180°$ we can simply analyze where does vector APm has to be positioned to have an angle of 180° with vector Q. We know in advanced that Q is in the positive direction of axis Z, so if we want APm to have 180° with vector Q, APm has to be in the negative direction of axis Z. For this case vector T can be any unitary vector residing in plane XY, the simplest vectors are the ones having the direction of +X, -X, +Y or -Y. For our implementation we used T = [1, 0, 0].

After obtaining the points of intersection P1'' and P2'' we can calculate the perimeter covered by a transmission. We can obtain two angles with these two points. However in order to get the correct angles, we have to be sure on the correct order of them. Figure 18 shows this problem.

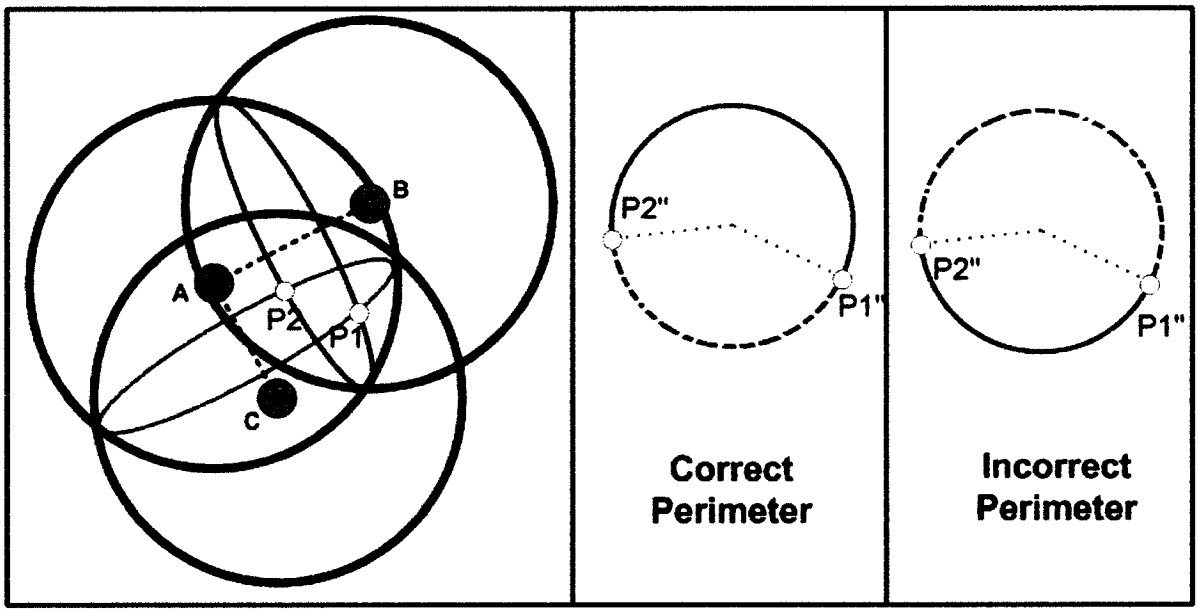

Figure 18. Node *A* needs to update the perimeter of the circle previously created by node B. We show a correct and an incorrect update of the perimeter.

We show the procedure getPerimeter. Receiving nodes that want to obtain angles θstart and θend uses this procedure. For example if node *A* from Figure 18 wants to update the circle generated by node *B* with the circle generated by node *C*, it will call getPerimeter(C_coordinates'', P1x'', P1y'', P2x'', P2y''). If for instance node *A* wants to update the circle generated by node *C* with the circle generated by node *B*, it will call getPerimeter(B_coordinates'', P1x'', P1y'', P2x'', P2y'').

BEGIN getPerimeter(other_node_coord'', P1x'', P1y'', P2x'', P2y'')

θstart = getAngle(P1y'', P1x'');

θend = getAngle(P2y'', P2x'');

IF θend > θstart THEN

$$\theta\_reference = \frac{\theta start + \theta end}{2};$$

ELSE

$$\theta\_reference = 180 + \frac{\theta start + \theta end}{2};$$

ENDIF

$x\_reference = |P1| \cos(\theta\_reference);$

$y\_reference = |P1| \sin(\theta\_reference);$

point_reference = (x_reference, y_reference);

IF distance(point_reference, other_node_coord'') > R THEN

temp = θstart;

θstart = θend;

θend = temp;

ENDIF

END getPerimeter

BEGIN getAngle(y, x)

IF $x \neq 0$ AND $y \neq 0$ THEN

$$\theta = \frac{180}{\pi} \tan^{-1}\left(\frac{|y|}{|x|}\right);$$

IF $x < 0$ AND $y > 0$ THEN

θ = 180 - angle;

ELSEIF $x < 0$ AND $y < 0$ THEN

θ = 180 + angle;

```
        ELSIF x > 0 AND y < 0 THEN
            θ = 360 - angle;
        ENDIF
    ELSEIF y = = 0 THEN
        IF x > 0 THEN
            θ = 0;
        ELSE
            θ = 180;
        ENDIF
    ELSEIF (x = = 0) THEN
        IF y > 0
            θ = 90;
        ELSE
            θ = 270;
        ENDIF
    ENDIF
END getAngle
```

In order to conclude 3D-ABBA4, we now have to find a way to obtain the intervals of intersection between several circles in 3D. Since we have presented a way of obtaining the perimeters in the XY plane after rotating the circles of intersection (Figure 17), we can apply the same procedures and steps presented for 2D-ABBA in Section 3.2.1.2. In 2D-ABBA each node had to check only one transmission circle in the XY plane; however in 3D-ABBA4, each node will have to check several circles in the XY plane.

4.5.2 Reliability of 3D-ABBA4

A node in 3D-ABBA4 will not transmit only if it is covered by other transmissions. In order to test coverage, a receiving node W checks its list of circles of intersection. If the list is empty before the timeout expires, node W will not transmit. The only way that 3D-ABBA4 can fail is if W has its circles list empty, but its transmission sphere is still not completely covered.

When a circle of intersection A is generated, at first its whole perimeter is in the surface of the transmission sphere. When A is updated with the intersection of other circles, if it has segments still uncovered, these segments reside in the uncovered surface of the transmission sphere. It is easy to observe from this example that as long as there exists an uncovered surface, there will exist uncovered sections of circles. In other words if there exists uncovered sections of circles there exists uncovered surfaces, thus in order to cover completely the transmission sphere there has to be no uncovered segments of circles. This guarantees that 3D-ABBA4 achieves complete reliability.

Chapter 5 Performance evaluation of the 3D Area Based Beaconless Broadcasting Algorithms

We considered a network of $n = 500$ static nodes, randomly distributed over a volume of 100 x 100 x 100. In order to control the average node degree d, we sorted all $\frac{n(n-1)}{2}$ (potential) edges in the network by their length, in increasing order. The transmission radius R is equal to the $nd/2$-th edge in the sorted array. We used Dijkstra's shortest path algorithm *(SP)* to test whether a graph was connected. We generated a total of 100 connected graphs for each of the following network degrees: $d = 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100$.

After creating each of the connected graphs we started a transmission from a randomly selected node, and we measured the following characteristics to broadcast a single message to the entire network using 3D-ABBA1, 3D-ABBA2, 3D-ABBA3 and 3D-ABBA-4:

- Average number of transmissions.
- Average percentage of nodes receiving the message during the broadcast.
- Percentage of 'internal' transmitting nodes.
- Percentage of 'external' transmitting nodes.

We define 'internal' node as the node whose transmitting volume is covered by its neighbor nodes. In other words, if before performing a broadcast the transmission pattern of a node A is covered by the transmission patterns of its neighbors, then node A may or may

not retransmit during a broadcast; however, if the transmission pattern of a node B is not covered by the transmissions patterns of its neighbors, then node B must retransmit during a broadcast. Such type of nodes will be called as 'external' nodes.

It is important to clarify that in our simulation we considered ideal MAC and Physical layers. Our simulations were done in Matlab. Besides simulating all ABBA's we considered an alternate version of each of them. We considered a random timeout in each version to see the impact of the original timeout functions. Tables 1 to 4 show the results obtained for the original 3D ABBAs. Tables 5 to 8 show the results obtained for the random versions. The percentage of nodes receiving the message during the broadcast was 100% in every test for all ABBAs, except for 3D-ABBA1 where we found 2 unsuccessful tests, and the average number of nodes that did not get the message during these failures was only one. This shows that although not all versions of ABBA are 100% reliable (as shown in previous sections), all versions performed extremely well on the reliability criterion. Figure 19 shows the average number of messages during a broadcast using all 3D versions of ABBA. We observe that among the 3D versions, 3D-ABBA1 performed the best in terms of number of messages needed to complete a broadcast. However, since its reliability is not 100% we cannot immediately conclude that 3D-ABBA1 is the best protocol. The second best version was 3D-ABBA2; however, this is again, in theory, an unreliable protocol. The difference between 3D-ABBA1 and 3D-ABBA3 was on average about 25% of the messages for the original timeout function and about 20% for the random timeouts. The other reliable algorithm, 3D-ABBA4, had around 1% less messages than 3D-ABBA3.

Between 3D-ABBA3 and 3D-ABBA2 the difference was on average about 3% of the messages for the original timeout function and for the random timeout versions the difference was about 5% of the messages. The original timeout function in 3D-ABBA1 had

about 10% fewer retransmissions than the random timeout function in 3D-ABBA1. However, when we applied random timeouts to 3D-ABBA2, we surprisingly obtained a minor improvement. In case of 3D-ABBA3, the difference was about 2% in favour of the original timeout function.

Figure 20 shows the percentage of transmissions done by the internal nodes in the network, and Figure 21 shows the percentage of transmissions done by the external nodes in 3D-ABBA1 and 3D-ABBA2. It is obvious that for reliable broadcasting algorithms, the percentage of transmissions of external nodes has to be 100 %. This was the case for 3D-ABBA3 and 3D-ABBA4 but not for 3D-ABBA1 and 3D-ABBA2.

Comparing the percentage of transmissions from internal nodes, 3D-ABBA1, 3D-ABBA2, 3D-ABBA3, and 3D-ABBA4 had a difference of 5.68%, 0.34%, 2.32%, and 2.45% between its two timer functions respectively. The difference between 3D-ABBA1 and 3D-ABBA2 was 15.11% when original timers were used and of 9.77% when random timers were tested. 3D-ABBA2 and 3D-ABBA3 had differences of 1.47% and 3.45% respectively. When high degree networks were tested only 30% of the internal nodes transmitted in 3D-ABBA3. This shows that 3D-ABBA3 achieves considerable energy savings while guaranteeing the reliable completion of a broadcast. 3D-ABBA4 presented almost the same results as 3D-ABBA3, and since it is more complex than all 3D ABBA versions, we think it is better to use 3D-ABBA3.

The average difference of the percentage of transmitting external nodes was 10.80% between 3D-ABBA1 and 3D-ABBA2 (original timer functions). When using random timers, the difference was of 12.88%. The average percentage of transmitting nodes (external) shows that although 3D-ABBA1 and 3D-ABBA2 are not reliable, they both guarantee that

almost all external nodes will transmit. In 3D-ABBA1 the minimum percentage of transmitting external nodes was of 77% but in 3D-ABBA2 this minimum value was of 94%.

Table 1. 3D-ABBA1 results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	447.04	59.58	92.56
8	430.72	58.69	90.93
9	417.27	57.04	89.97
10	402.42	55.79	89.07
15	340.82	46.81	87.32
20	300.74	40.38	86.29
25	271.78	33.92	85.99
30	250.84	29.66	84.41
35	232.46	25.69	83.43
40	220.24	23.38	82.26
45	209.58	21.12	81.99
50	199.90	19.13	80.77
60	187.36	16.43	80.57
70	173.76	14.52	78.57
80	164.64	12.96	77.43
90	157.18	11.53	77.66
100	151.50	10.62	76.70

Table 2. 3D-ABBA2 results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	484.42	76.05	99.08
8	476.98	75.37	98.88
9	468.04	73.41	98.59
10	457.44	71.62	98.37
15	407.88	63.51	97.75
20	370.12	56.15	97.62
25	342.24	49.60	97.56
30	320.44	44.36	96.97
35	303.42	40.19	96.99
40	290.88	37.24	96.81
45	278.80	34.60	96.46
50	273.22	33.30	96.33
60	260.46	30.63	95.96
70	248.50	28.80	94.85
80	242.46	27.76	94.58
90	234.22	26.37	94.27
100	228.12	25.15	93.80

Table 3. 3D-ABBA3 results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	488.56	75.97
8	482.42	76.24
9	474.44	74.16
10	465.58	73.23
15	419.70	65.99
20	382.86	58.81
25	353.88	51.85
30	332.04	46.24
35	313.74	41.71
40	302.40	39.05
45	289.94	36.13
50	285.46	35.11
60	270.46	31.62
70	261.46	30.19
80	254.48	28.79
90	247.00	27.54
100	242.24	26.51

Table 4. 3D-ABBA4 results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	487.64	75.48
8	481.63	75.45
9	473.67	73.16
10	464.86	72.30
15	419.46	65.25
20	382.38	57.85
25	352.92	51.46
30	331.68	45.84
35	313.21	41.59
40	301.48	38.77
45	289.79	35.35
50	284.65	35.04
60	270.08	30.82
70	261.29	29.54
80	253.58	28.65
90	246.62	27.19
100	241.51	26.08

Table 5. 3D-ABBA1 with random timeout results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	459.06	68.91	94.23
8	444.06	66.50	92.70
9	429.06	62.77	91.51
10	416.40	62.69	90.44
15	357.96	52.82	88.43
20	318.36	45.03	88.32
25	289.44	38.74	87.53
30	269.48	34.53	86.23
35	254.34	31.02	86.10
40	239.72	28.24	84.37
45	229.48	26.12	84.00
50	221.92	24.37	83.54
60	208.32	21.78	82.40
70	194.10	19.24	81.21
80	187.36	18.00	80.85
90	179.84	17.04	79.90
100	174.70	16.05	79.47

Table 6. 3D-ABBA2 with random timeout results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	486.00	79.20	99.10
8	478.52	76.72	99.00
9	469.88	75.23	98.60
10	459.26	72.81	98.45
15	410.68	64.61	97.83
20	372.04	56.79	97.66
25	343.92	49.99	97.81
30	319.28	43.91	97.09
35	304.12	40.54	96.76
40	291.92	37.56	96.80
45	280.84	35.13	96.62
50	273.54	33.18	96.75
60	259.10	30.34	95.73
70	245.96	28.08	94.79
80	237.14	26.24	94.53
90	230.26	25.37	93.97
100	224.52	24.30	93.37

Table 7. 3D-ABBA3 with random timeout results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	490.56	80.17
8	484.84	79.51
9	478.14	77.90
10	469.92	76.61
15	425.62	68.50
20	389.74	61.23
25	358.94	53.52
30	339.16	48.52
35	320.42	43.80
40	310.04	41.41
45	298.90	38.85
50	291.08	36.81
60	277.38	33.68
70	266.20	31.58
80	256.96	29.51
90	251.30	28.77
100	248.38	28.26

Table 8. 3D-ABBA4 with random timeout results of 100 tests of networks of 100 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	490.70	80.03
8	484.58	79.07
9	478.33	77.47
10	469.91	76.11
15	425.47	67.78
20	389.61	60.97
25	359.30	53.24
30	339.36	47.60
35	320.24	43.53
40	309.61	41.39
45	299.17	37.97
50	291.09	36.79
60	277.77	33.48
70	266.22	31.19
80	257.34	28.55
90	251.16	28.46
100	248.36	27.77

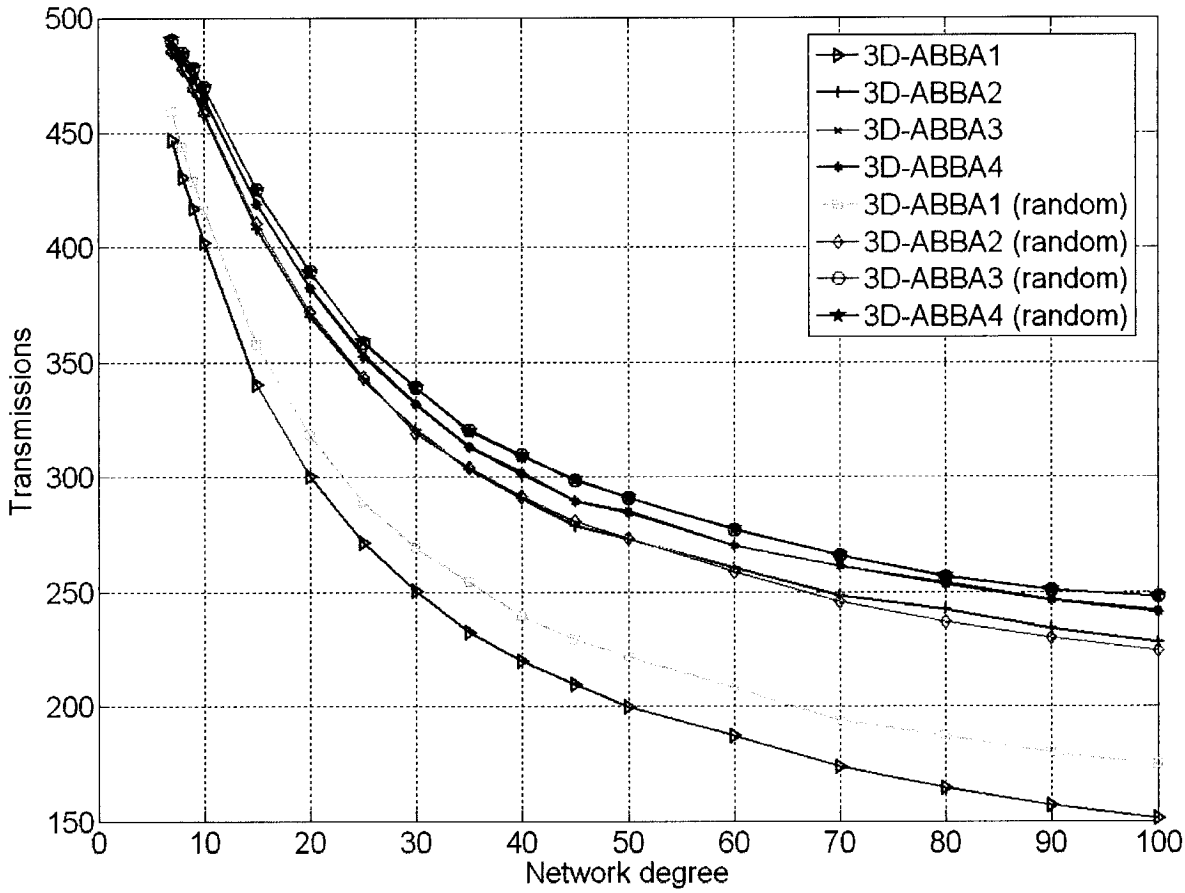

Figure 19. Average percentage of transmitting nodes of all 3D versions of *ABBA*.

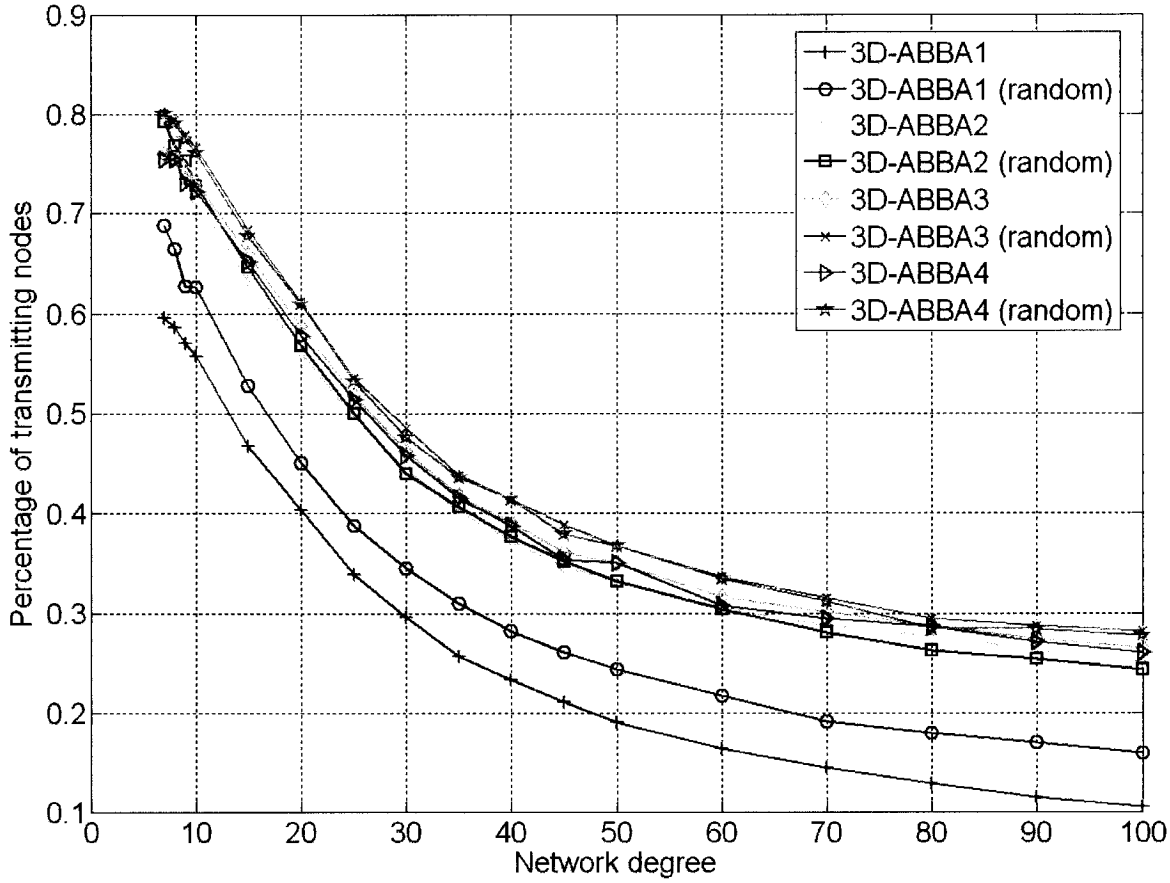

Figure 20. Percentage of transmitting nodes (internal nodes).

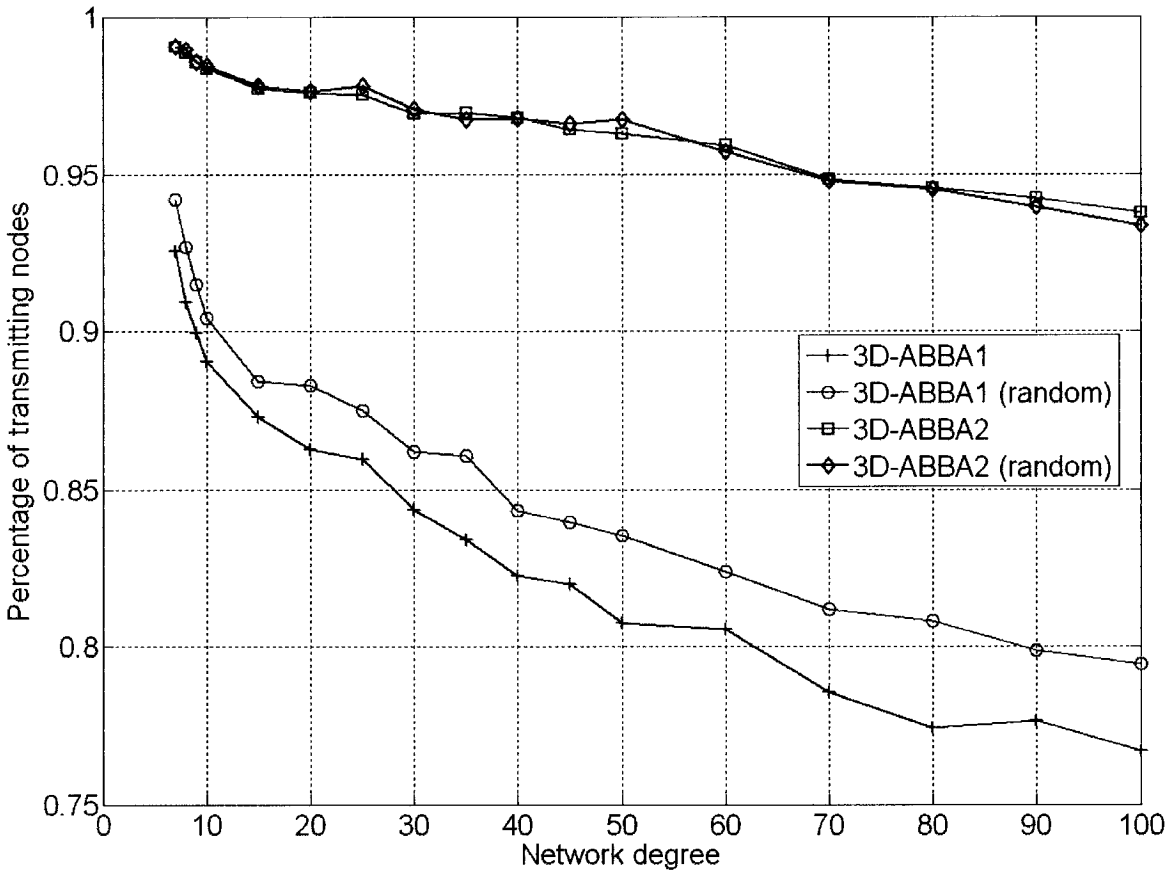

Figure 21. Percentage of transmitting nodes (external nodes).

Chapter 6 Comparison of 2D-ABBA with OFP and Geoflood

As the reader may recall from the literature review section, there exists two recent 2D beaconless broadcasting algorithms beside 2D-ABBA: Optimal Flooding Protocol and Geoflood. The OFP authors explained how to implement a beaconless version of their protocol; however, all of their results were obtained with a 1-hop knowledge implementation. They also claimed that OFP was better than any form of flooding. In order to have a fair comparison between OFP and 2D-ABBA we implemented two beaconless versions of OFP, one with a threshold value of $0.4R$, and the other one following the short description given by the authors, where no threshold value is used.

Geoflood was also implemented. In the literature review section we showed that Geoflood was unreliable. We also showed that in order to be reliable, Geoflood has to divide the transmission area in six angles. For this comparison with 2D-ABBA we implemented three versions of Geoflood; one dividing the transmission area in four angles (as the original proposal of Geoflood), another one with five divisions and finally one with six divisions (reliable version of Geoflood). For the tests, we considered a network of $n = 500$ static nodes, randomly distributed over an area of 100 x 100. In order to control the average node degree d, we sorted all $\frac{n(n-1)}{2}$ (potential) edges in the network by their length, in increasing order. The transmission radius R is equal to the $nd/2$-th edge in the sorted array. We used Dijkstra's shortest path algorithm *(SP)* to test whether a graph was connected. We generated

a total of 100 connected graphs for each of the following network degrees: d = 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 125.

After creating each of the connected graphs we started a transmission from a randomly selected node, and we measured the following characteristics to broadcast a single message to the entire network using the original and the random 2D-ABBA, two beaconless versions of OFP and the three mentioned versions of Geoflood:

- Average number of transmissions.
- Average percentage of nodes receiving the message during the broadcast.
- Percentage of 'internal' transmitting nodes.
- Percentage of 'external' transmitting nodes.

Tables 9 to 17 show the results obtained for all the 2D broadcasting protocols. It is important to remark that OFP is not reliable. Even in high degree networks, OFP can fail as shown in Table 13. For the case of Geoflood, almost all the tests were successful; in other words, the average percentage of nodes receiving the message during the broadcast was 100% in every test but one; only for a network with average degree 7, Geoflood with four angles division failed (once). We can notice how similar in form are all the plots from Figure 22. The threshold version (th = 0.4R) of OFP was the worst among all 2D beaconless versions in terms of the number of messages to complete a broadcast. Since the main purpose of a broadcasting protocol is to deliver a message to all the nodes in the network, we can conclude that neither of the OFP versions is a good option to broadcast a message. We observe that the version of OFP without a threshold value performs better than 2D-ABBA and Geoflood, in terms of number of retransmissions. Also from Figure 22 we show that both versions of 2D-ABBA outperform all versions of Geoflood. Table 17 shows the percentage of the difference in message counts of OFP and Geoflood compared to the

original version of 2D-ABBA. Negative difference means that OFP has smaller retransmission count than 2D-ABBA. In Fig. 23 we show the percentage of transmitting internal nodes; the average percentage difference between the two versions of 2D-ABBA was 4.55%, making us believe that the random version is a viable option to be deployed in IEEE 802.11 networks. The difference between the original 2D-ABBA and OFP was 10.68%. Here again OFP performed better; however as we will show later, OFP also suppress transmissions from external nodes, reaffirming OFP unreliability. The difference between 2D-ABBA and Geoflood of 4, 5, and 6 angles was 6.83%, 20.74%, and 29.22% respectively. 2D-ABBA performed much better than all versions of Geoflood and the difference with OFP was not very considerable.

In Figure 24 we show the percentage of transmitting external nodes. For this case the percentage was 100% for 2D-ABBA and for Geoflood of five and six angles. In this Figure we show in other way how unreliable OFP is. Here we can observe that OFP causes that a big percentage of the external nodes will not transmit, which can cause broadcast failures (if there were more neighbors in areas solely covered by such nodes). In average the OFP version with no threshold caused that 58.28% of the external nodes did not transmit during the broadcast. Even for high connected networks (degree = 125) 73.01% of the external nodes did not transmit. These results support that OFP can fail even in high connected networks. In the case of Geoflood of four angles the percentage of external transmitting nodes remained high. In average, only 10.56% of external nodes did not transmit.

Table 9. 2D-ABBA results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	395.98	62.11
8	373.80	59.42
9	355.40	58.19
10	337.99	56.12
15	268.66	46.15
20	228.37	38.90
25	199.10	32.89
30	179.96	28.92
35	163.27	25.52
40	151.25	23.17
45	142.57	21.35
50	133.90	19.66
60	120.51	16.92
70	112.82	15.52
80	104.40	13.85
90	98.24	12.61
100	93.32	11.69
125	85.07	62.11

Table 10. 2D-ABBA with random timeout results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	419.07	70.58
8	398.60	67.46
9	377.49	64.63
10	360.07	62.18
15	292.29	51.68
20	247.26	43.14
25	217.72	37.04
30	195.68	32.42
35	179.53	29.12
40	168.02	26.87
45	157.81	24.71
50	149.29	23.03
60	136.89	20.50
70	128.29	18.89
80	120.07	17.26
90	114.99	16.25
100	110.21	15.35
125	102.23	13.84

Table 11. Optimal Flooding Protocol (threshold version, $th = 0.4R$) results of 100 tests of networks of 500 nodes when it achieved to complete the broadcast

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	481.93	95.50	97.45
8	477.11	94.56	96.82
9	474.23	94.05	96.60
10	469.70	93.13	96.20
15	450.08	89.20	94.94
20	433.87	85.96	93.33
25	416.21	82.29	91.58
30	402.92	79.60	89.70
35	389.67	76.86	88.26
40	378.54	74.59	86.96
45	365.07	71.75	85.95
50	354.26	69.58	84.21
60	335.66	65.76	81.98
70	317.76	62.21	78.73
80	302.91	59.14	77.36
90	287.61	56.03	75.18
100	273.41	53.10	73.80
125	243.67	47.11	69.04

Table 12. Optimal Flooding Protocol (no threshold version) results of 100 tests of networks of 500 nodes when it achieved to complete the broadcast

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	269.67	45.87	63.92
8	250.57	43.25	61.32
9	233.15	40.61	60.27
10	219.69	38.32	59.43
15	167.87	30.61	51.96
20	137.34	25.01	47.99
25	116.29	21.09	43.06
30	99.52	17.83	39.78
35	89.66	15.93	37.98
40	80.28	13.99	37.59
45	73.78	12.71	36.47
50	67.72	11.65	34.40
60	58.30	9.87	32.07
70	51.39	8.54	31.02
80	46.52	7.57	30.48
90	41.74	6.75	28.37
100	38.45	6.12	27.80
125	32.99	5.06	26.99

Table 13. Percentage of unsuccessful broadcasts and average percentage of nodes receiving the message when the broadcast failed using both versions of OFP.

Degree	*% of failed broadcasts (OFP threshold = 0.4R)*	*Average % of nodes receiving the message when the broadcast was unsuccessful (OFP threshold = 0.4R)*	*% of failed broadcasts (OFP no threshold version)*	*Average % of nodes receiving the message when the broadcast was unsuccessful (OFP no threshold version)*
7	10	98.78	97	91.47
8	9	99.09	93	93.08
9	5	99.76	80	95.49
10	4	99.30	71	98.56
15	1	99.40	30	99.27
20	0	-	17	99.48
25	0	-	9	99.42
30	0	-	14	99.69
35	0	-	7	99.23
40	0	-	10	98.94
45	0	-	3	99.47
50	0	-	2	97.90
60	0	-	3	99.73
70	0	-	1	99.80
80	0	-	2	99.40
90	0	-	1	99.80
100	0	-	4	99.65
125	0	-	2	99.40

Table 14. Geoflood (four angle divisions) results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes	Percentage of external transmitting nodes
7	425.46	74.64	97.39
8	405.80	71.29	96.94
9	386.95	68.78	96.17
10	371.10	66.40	95.76
15	301.91	54.70	94.36
20	255.97	45.95	92.93
25	225.79	39.85	91.14
30	203.43	35.23	90.15
35	184.49	31.39	88.93
40	173.69	29.36	87.81
45	160.65	26.60	87.45
50	151.52	24.86	86.43
60	135.79	21.65	85.50
70	128.07	20.24	84.92
80	118.09	18.28	84.00
90	111.87	17.09	83.02
100	105.79	15.80	83.97
125	96.22	13.98	82.98

Table 15. Geoflood (five angle divisions) results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	468.51	88.60
8	455.60	85.74
9	442.23	83.32
10	429.88	81.02
15	370.27	69.80
20	325.36	60.70
25	292.70	53.75
30	270.43	49.00
35	252.23	45.18
40	237.85	42.23
45	228.40	40.23
50	218.41	38.19
60	203.61	35.10
70	194.75	33.38
80	186.28	31.67
90	181.97	30.81
100	176.46	29.74
125	167.82	28.04

Table 16. Geoflood (six angle divisions) results of 100 tests of networks of 500 nodes

Degree	Mean transmissions	Percentage of internal transmitting nodes
7	488.50	95.81
8	481.17	93.94
9	473.56	92.33
10	464.03	90.23
15	416.12	80.43
20	371.63	71.10
25	338.65	64.00
30	313.84	58.65
35	293.05	54.21
40	280.28	51.58
45	266.41	48.59
50	256.52	46.56
60	238.53	42.74
70	229.35	40.93
80	219.63	38.93
90	214.31	37.85
100	207.56	36.49
125	198.64	34.72

Table 17. Comparison between 2D-ABBA and all the other 2D beaconless protocols

Net Degree	% of less or more messages compared with 2D-ABBA				
	OFP	*Geoflood (4 ang)*	*Geoflood (5 ang)*	*Geoflood (6 ang)*	*2D-ABBA (random timer)*
7	-31.90	7.45	15.48	18.94	5.51
8	-32.97	8.56	17.95	22.31	6.22
9	-34.40	8.88	19.63	24.95	5.85
10	-35.00	9.80	21.38	27.16	6.13
15	-37.52	12.38	27.44	35.44	8.08
20	-39.86	12.09	29.81	38.55	7.64
25	-41.59	13.41	31.98	41.21	8.55
30	-44.70	13.04	33.45	42.66	8.03
35	-45.09	13.00	35.27	44.29	9.06
40	-46.92	14.84	36.41	46.04	9.98
45	-48.25	12.68	37.58	46.48	9.66
50	-49.42	13.16	38.69	47.80	10.31
60	-51.62	12.68	40.81	49.48	11.97
70	-54.45	13.52	42.07	50.81	12.06
80	-55.44	13.11	43.96	52.47	13.05
90	-57.51	13.87	46.01	54.16	14.57
100	-58.80	13.36	47.12	55.04	15.33
125	-61.22	13.11	49.31	57.17	16.79

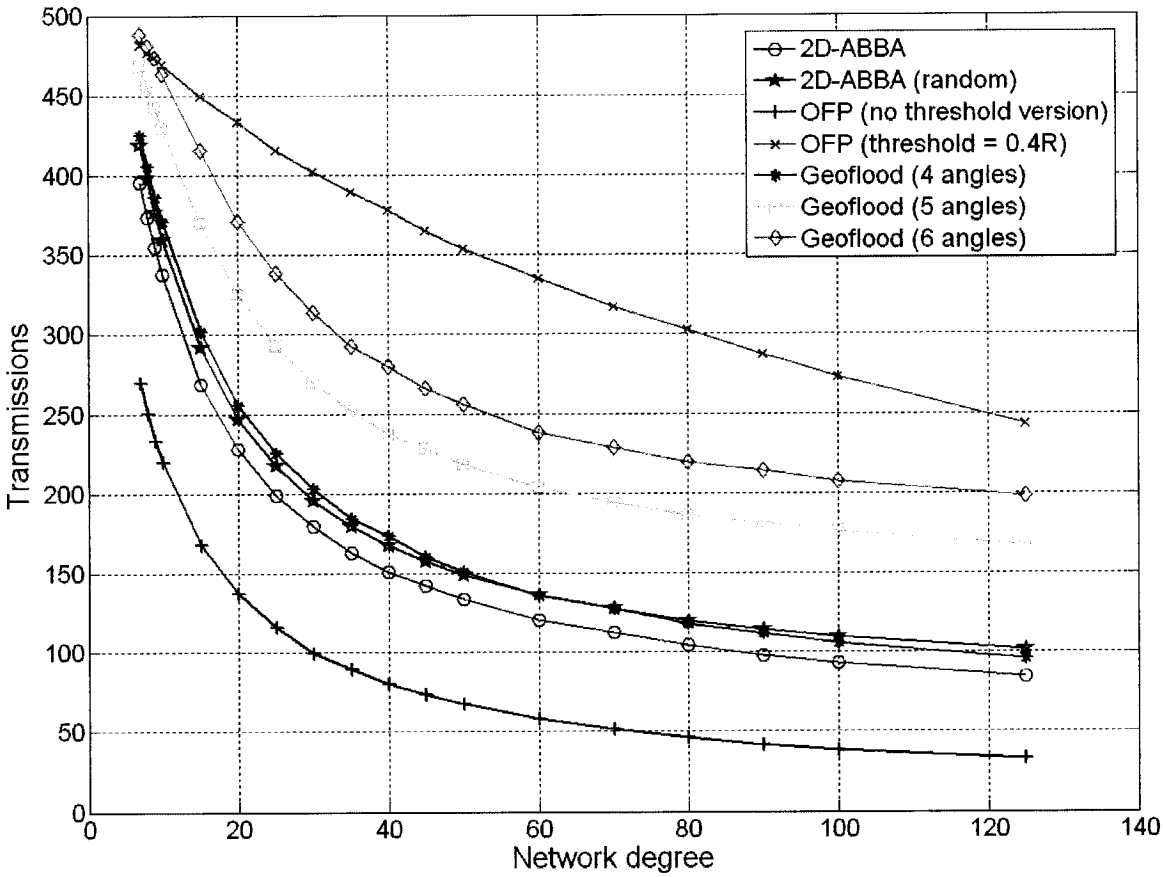

Figure 22. Comparison between 2D-ABBA, beaconless OFP and Geoflood.

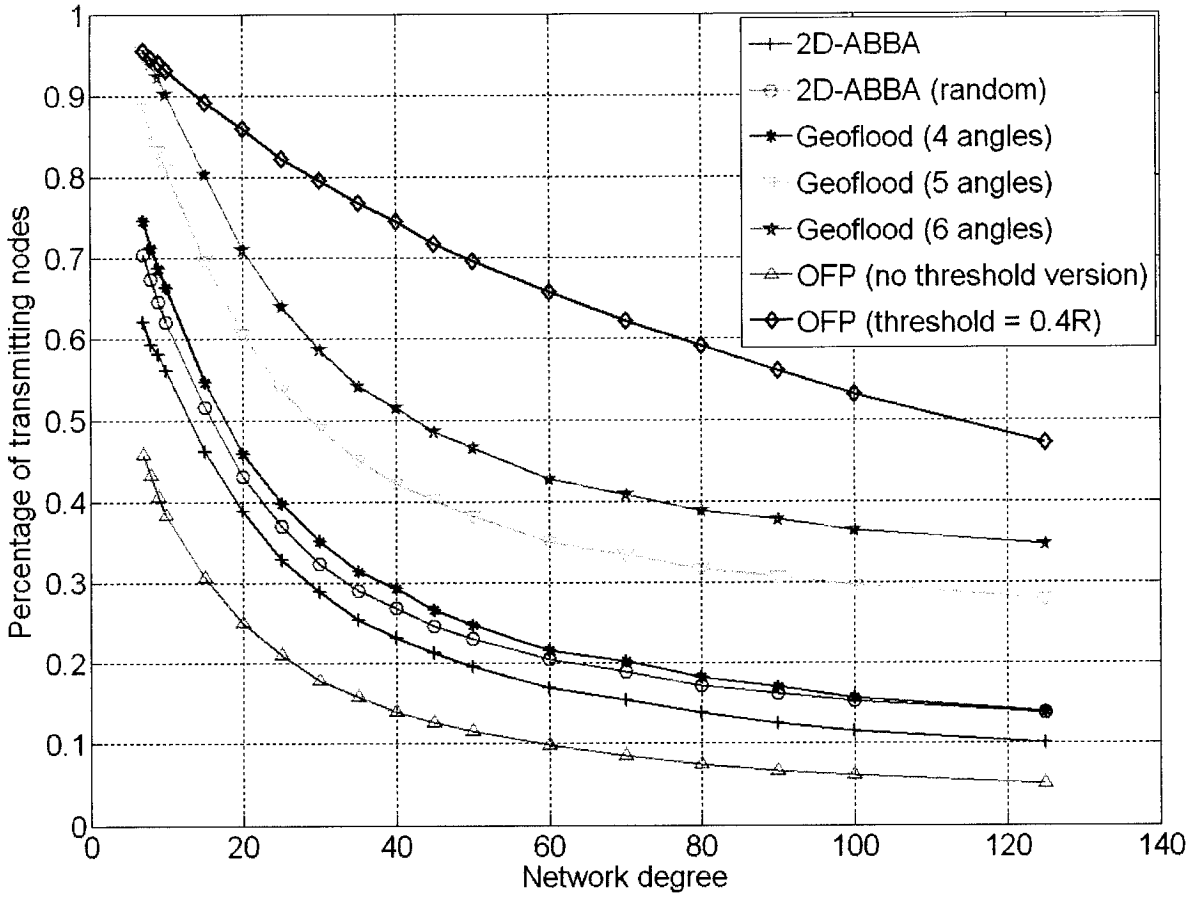

Figure 23. Percentage of transmitting nodes (internal nodes).

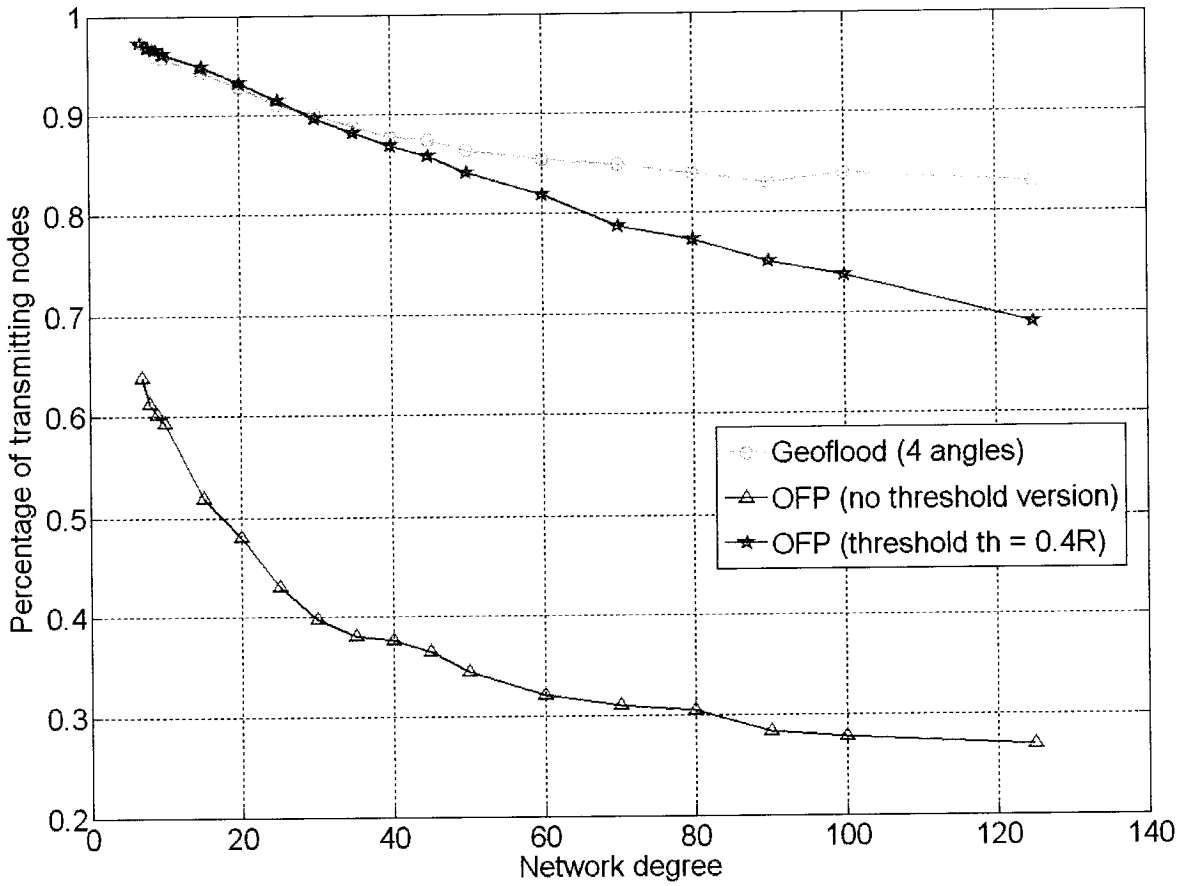

Figure 24. Percentage of transmitting nodes (external nodes).

Chapter 7 Conclusions, Future work, and Open ideas

Area based beaconless broadcasting algorithm (*ABBA*) was proposed as a broadcasting protocol for ad hoc networks. This algorithm has fewer assumptions than almost all broadcasting algorithms. Localized broadcasting algorithms normally assume the use of 'hello' messages to provide the list of *k*-hop neighbors. The use of such beacons can cause large communications overhead and delivery failures due to outdated information. Our protocols, however, find the neighbors only when they are really needed, and rely on the geographic location of the nodes.

We presented five beaconless broadcasting algorithms, one for networks that reside in the plane (2D), and four for networks that reside in space (3D). All versions were based in setting timeouts before retransmitting; this timeouts depended in several things like the transmission area still to be covered. We also tested the algorithms by setting its timeouts with a random function just like IEEE 802.11 does. We did that to test its viability in such already deployed networks.

Several beaconless routing protocols were proposed in [NTCS] but they were not competitive in comparisons made in other articles. We propose 2D-ABBA as a new beaconless broadcasting algorithm in 2D, and compare it with two recent such protocols, OFP and Geoflood. We showed that OFP is not the best broadcasting protocol since it frequently lacks reliability. We showed that Geoflood protocol, in order to be reliable, has to divide the transmission area into six angles instead of four. We implemented two versions of OFP and three versions of Geoflood in order to compare them with 2D-ABBA. When OFP

with no threshold had a successful flooding it had fewer messages than 2D-ABBA; however, 2D-ABBA outperformed OFP in terms of reliability, and since the principal characteristic of a broadcasting task is delivery, it is easy to see that our algorithm is a much better option to perform a broadcasting. 2D-ABBA had fewer message count than all versions of Geoflood. Compared to the reliable version of Geoflood, 2D-ABBA presented on average 40% less messages than the reliable version of Geoflood, while the random 2D-ABBA had on average 38% less messages. Compared to Geoflood with 4 and 5 angles division the original and the random 2D-ABBA had on average 19%, 17%, 34% and 31% less messages respectively.

The 3D-ABBA versions presented in this work are the first beaconless broadcasting algorithms proposed, other than blind flooding, to work in 3D networks. Several 3D applications are envisioned for the very near future; even now, exist 3D sensor networks to measure engineering structures like buildings and bridges; this makes us believe that several broadcasting protocols are needed for environments in 3D. Although only two 3D-ABBA versions (3D-ABBA3 and 3D-ABBA4) are fully reliable in all scenarios, the other two 3D versions performed almost perfectly in the experimental tests. More precisely, 3D-ABBA1 failed in only 2 tests, while 3D-ABBA2 was always reliable in our experiments. 3D-ABBA1 performed better than 3D-ABBA2, 3D-ABBA3, and 3D-ABBA4 in terms of the number of messages to complete a broadcast.

We also measured the percentage of external and internal transmitting nodes during a broadcast. We obtained that both versions of OFP suppressed a considerable percentage of the external transmitting nodes, while in fact, the percentage of the external transmitting nodes has to be 100% to guarantee reliability. This 100% was the case for 2D-ABBA, for Geoflood of 5 and 6 angles, and for both versions of 3D-ABBA3 and 3D-ABBA4.

It is an interesting extension of this research to evaluate how MAC and physical layers affect the performance of all the beaconless algorithms. Recall that we used only ideal MAC and physical layers. That is, all the neighbors located within the transmission radius correctly receive the message sent by a node. In reality, however, it is impossible for any antenna to have a perfect sphere as its transmission pattern. We believe that our 3D proposals set the basis for future research that takes into account the different shapes found in real 3D patterns. Another important open issue is the development of beaconless broadcasting protocols for networks whose nodes have directional antennas. Perhaps, ABBA versions could be modified to work with another transmission patterns, instead of circumferences and spheres or to work in networks where nodes have different transmission radii. Figure 25 presents the most basic examples of antennas (monopole and dipole) and its radiation patterns. By looking at these patterns we can conclude that the 3D beaconless versions require further and extensive research, while the 2D beaconless version can be easily deployed in existing IEEE 802.11 networks.

We were able to obtain excellent results with random timeout variants. We believe that these show a very good promise to deploy our beaconless algorithms in IEEE 802.11 networks. The reason is that IEEE 802.11 uses discrete random timeouts, and the implementation of the random timeout ABBAs would be straightforward.

Finally, we note that, besides serving for broadcasting purposes, our algorithms could also be a viable option for routing purposes. Almost every routing protocol requires the interchange of routing tables or at least the interchange of neighbor information. When great mobility exists, a lot of energy is wasted only for these tasks. For this type of environments with significant node mobility, and small or perhaps medium number of nodes, one can apply ABBA as a routing algorithm.

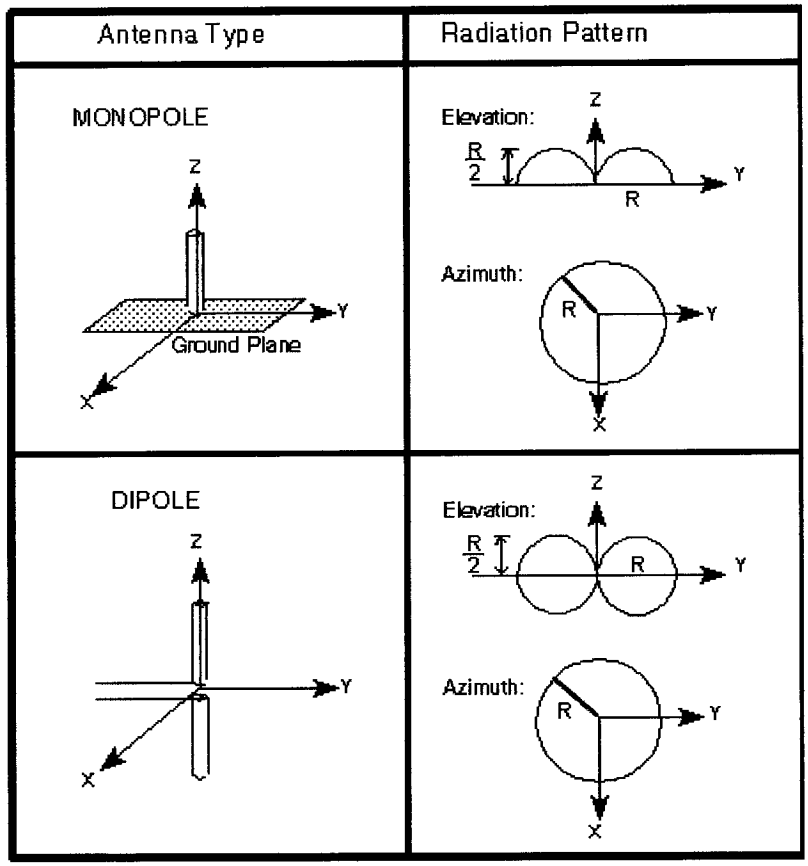

Figure 25. Radiation patterns of a monopole and a dipole.

References

[ADEP] J. Arango, M. Degermark, A. Efrat, S. Pink, "An Efficient Flooding Algorithm for Mobile Ad-Hoc Networks," Proceedings of the 2nd Workshop on Modeling and Optimizations in Mobile Ad Hoc and Wireless Networks (WiOpt 2004), March 2004.

[BCCFCMR] F. Bergonovo, A. Capone, M. Cesana, L. Fratta, L. Coletti, L. Moretti, N. Riato, "Inter-vehicles communication: A new frontier of ad hoc networking, Proc. Medhoc," Tunisia, June 2003.

[BHSS] B. Blum, T. He, S. Son, and J. Stankovic, "IGF: A state-free robust communication protocol for wireless sensor networks," Department of Computer Science, University of Virginia, USA, Tech. Rep. CS-2003-11, 2003.

[CMWZ] G. Calinescu, I. Mandoiu, P.J. Wan and A. Zelikovsky, "Selecting forwarding neighbors in wireless ad hoc networks," Proc. DIAL M, 2001.

[CS] J. Cartigny and D. Simplot, "Border node retransmission based probabilistic broadcast protocols in ad hoc networks," System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on 6-9 Jan. 2003 Page(s):10 pp.

[CS1] G. Chen and I. Stojmenovic, "Clustering and Routing in Wireless Ad Hoc Networks," Technical Report TR-99-05, Dept. of Computer Science, SITE, Univ. of Ottawa, June 1999.

[FWKMH] H. Fussler, J. Widmer, M. Kasemann, M. Mauve, and H. Hartenstein, "A novel forwarding paradigm for position-based routing (with implicit addressing)," Proceedings 2003 IEEE 18th Annual Workshop on Computer Communications, 2003. CCW 2003. 20-21 Oct. 2003 pp.194 – 200.

[GKP] M. Gerla, T.J. Kwon and G. Pei, "On demand routing in large ad hoc wireless networks with passive clustering," Proc. IEEE WCNC, September 2000.

[HB1] M. Heissenbuttel, T. Braun, "A novel position based and beaconless routing for mobile ad hoc networks," 3rd IEEE Workshop on Applications and Services in Wireless Networks ASWN03, Bern, Switzerland, July 2-4, 2003.

[HB2] Marc Heissenbuttel, Torsten Braun, "BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks," Computer Communications, Vol. 27, Issue 11, pp. 1076-1086.

[HKB] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in MOBICOM, Seattle, 1999, pp. 174–185.

[KM] D. Kim, N.F. Maxemchuk, "A comparison of flooding and random routing in mobile ad hoc network," 3rd New York Metro Area Networking Workshop, Sept. 2003.

[LG] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," IEEE J. Selected Areas in Comm., vol. 15, no. 7, pp. 1265-1275, 1997.

[LK] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," Proc. ACM MSWiM Workshop at MOBICOM, Aug. 2000; Computer Communication J., Vol. 24, Issue 3-4, pp.353-363, Feb. 2001.

[LS] N. Lin and S. Shrivastava, "System support for small-scale auctions," Proc. IFIP Medhoc, Tunisia, June 2003.

[NTCS] S.Y. Ni, Y.C. Tseng, Y.S. Chen, J.P. Sheu, "The broadcast storm problem in a mobile ad hoc network," Proc. MOBICOM, Seattle, Aug. 1999, pp. 151-162.

[PDJ] V.M. Paruchuri, A. Durresi, D.S. Dash, R. Jain, "Optimal flooding protocol for routing in ad hoc networks," TR, CS Department, Ohio State University, 2002; IEEE Wireless Communications and Networking Conference, March 2003.

[PDDJ] Vamsi M. Paruchuri, Arjan Durresi, Raj Jain, "Optimized Flooding Protocol for Ad hoc Networks," Ohio State University, Nov. 2003.

[QVL] A. Qayyum, L. Viennot, A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," Proc. Hawaii Int. Conf. System Sciences, January 2002.

[SHWAL] M. Sun, L. Huang, S. Wang, A. Arora, T.H. Lai, "Reliable MAC layer multicast in IEEE 802.11 wireless networks," Wireless Communications and Mobile Computing, 3, 2003, pp. 439-453.

[SL] M.T. Sun and T.H. Lai, "Location aided broadcast in wireless ad hoc network systems," Proc. IEEE Symposium on ad hoc wireless networks, at GLOBECOM, November 2001.

[SL1] M.T. Sun and T.H. Lai, "Computing optimal local cover set for broadcasting in ad hoc networks," Proc. IEEE ICC, 2002, pp. 3291-3295.

[SL2] M.T. Sun and T.H. Lai, "Location aided broadcast in wireless ad hoc network systems," Proc. IEEE WCMC, 2002.

[SSZ] I. Stojmenovic, M. Seddigh, J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," IEEE Trans. on Parallel and Distributed Systems, Vol. 13, No. 1, January 2002, pp. 14-25.

[SW] Stojmenovic I. and J. Wu, "Broadcasting and activity scheduling in ad hoc networks," in: Ad Hoc Networking (S. Basagni, M. Conti, S. Giordano and I. Stojmenovic, eds., IEEE/Wiley, 2004, pp. 205-229.

[WC] B. Williams, T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," Proc. MobiHoc, Lausanne, Switzerland, June 2002.

[WL] J. Wu and H. Li, "A dominating set based routing scheme in ad hoc wireless networks," Proc. DIAL M, Seattle, Aug. 1999, pp. 7-14; Telecommunication Systems, Vol. 18, Issue 1-2, 2001, pp. 13-36.

[YGK] Y. Yi, M. Gerla, T.J. Kwon, "Efficient flooding in ad hoc networks using on-demand (passive) cluster formation," Proc. IFIP Medhoc, Tunisia, June 2003.