# An Adaptive Hierarchical Routing Protocol for Wireless Ad-hoc Sensor Networks

Shahram Nourizadeh, Ye-Qiong Song, Jean-Pierre Thomesse

# An Adaptive Hierarchical Routing Protocol for Wireless Ad-hoc Sensor Networks

Shahram Nourizadeh, Y.Q. Song, J.P. Thomesse
LORIA research laboratory – National Polytechnic Institute of Lorraine INPL, France
{Shahram.Nourizadeh, Song, Thomesse}@loria.fr

**Abstract—This paper presents a cluster based, adaptive routing protocol that dynamically adapts to node's failure and mobility. Cluster head election, Mobility management, Failure management and Load balancing are the main parts of the protocol which are controlled by a Fuzzy decision making function. A new load balancing method by using Load tree is presented in this approach. The simulations results show the efficiency of this algorithm to manage the mobility and failure of the nodes and also to balance the load in the network.**

***Keywords: Adaptive routing, Dynamic Clustering, Mobility management, Failure management, Load balancing, Fuzzy Logic***

## I. INTRODUCTION

Nowadays, ad-hoc wireless sensor networks have many applications in the entire world and have recently emerged as a premier research topic. Intelligent homes, smart rooms, interactive virtual worlds are only a few examples. A famous example of these networks is healthcare network. These systems present important research and technical challenges. In a healthcare network, because of mobility of the patients, all the sensors attached to them, are mobile, therefore the mobility management and localization are important points in these networks.

Figure 1, shows a typical architecture of a healthcare system. In this example, the Communication backbone consists of the mobile or fixed sensors and also the sensors attached to the other persons, like doctors and nurses. This system needs an adaptive routing and localization protocol that can support mobility of the sensors and changes of network topology when patients moving.

Dynamic cluster-based routing is one of existing techniques for routing in ad hoc networks. As the membership in each cluster changes over time in response to node mobility and so it can be used to mobility management in mobile networks, and also cluster based routing in ad hoc networks can also make a large network appear smaller, but most importantly it can make a highly dynamic topology appear much less dynamic.

Several dynamic clustering strategies have been proposed in the literature. In [2], the zone routing protocol (ZRP) a hybrid strategy, is proposed by Haas and Pearlman which attempts to balance the trade-off between proactive and reactive routing. The objective of ZRP is to maintain proactive routing within a zone and to use a query–response mechanism to achieve inter-zone routing. In ZRP, each node maintains its own hop-count constrained routing zone; consequently, zones do not reflect a quantitative measure of stability, and the zone topology overlaps arbitrarily.

LEACH [3] is an application-specific data dissemination protocol that uses clustering to prolong the network lifetime. LEACH clustering terminates in a constant number of iterations (like HEED [4]), but it does not guarantee good cluster head distribution and assumes uniform energy consumption for cluster heads. In contrast, HEED makes no assumptions on energy consumption and selects well distributed cluster heads but HEED assumes quasi-stationary nodes.

In [5], a fuzzy logic approach to cluster-head election is proposed based on three descriptors - energy, concentration and centrality. In this approach the cluster-heads are elected by the base station in each round by calculating the chance each node has to become the cluster-head by considering three fuzzy descriptors – node concentration, energy level in each node and its centrality with respect to the entire cluster. This technique is proposed to use in LEACH [3], but it cannot support the mobility of the node and in addition it is centralized algorithm and therefore it cannot be scalable.

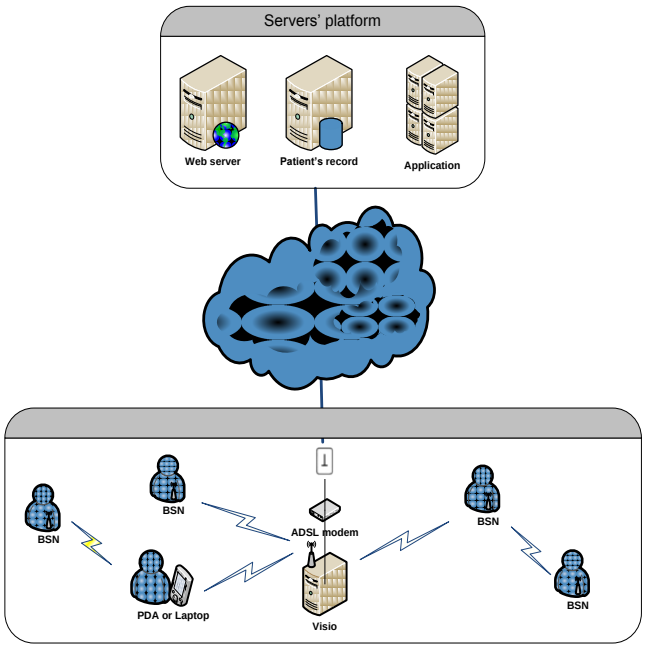


Figure 1. An example of mobile network

Another problem in these networks is failure of the nodes. Generally, the communication in sensor networks, with battery powered nodes, suffers from failure, low transmission power even more so than in regular wireless networks. Despite of many research projects, failure management in sensor networks is an open research challenge. Nodes' mobility, communication problem and link failure may be detected as a failure, on the other hand, in some networks, like healthcare networks, failure of a sensor can be detected as a health anomaly. These 2 cases can cause false alarms. Any failure in a healthcare system can lead patient situation to a critical level. This failure may be occurs in a sensor, computer, router or in the communication link.

Fault tolerance in measurements by a group of sensors, was first studied by Marzullo [6]. Marzullo proposed a flexible control process program that tolerates individual sensor failures. Issues addressed include modifying specifications in order to accommodate uncertainty in sensor values and averaging sensor values in a fault-tolerant way. The authors in [7] developed an algorithm that guarantees reliable and fairly accurate output from a number of different types of sensors when at most *k* out of *n* sensors are faulty. The results of the scheme are applicable only to certain individual sensor faults and traditional networks. They are not extendable to the reliability needs in complex network levels and most importantly; they do not address the reliability issues that are induced by the ad-hoc nature of the wireless sensor networks. Multi-sensor data fusion is a problem that recently has attracted a great deal of attention in a number of scientific and engineering communities [8][9][10]. Majority of these works are restricted to sensor fusion of the same modality.

One of limits of wireless sensor nodes is their inherent limited energy resource. To maximize the lifetime of the sensor node, it is necessary to distribute the load throughout the wireless sensor network in order to minimize maintenance and message number and maximize system performance. The load balancing averages the energy consumption and this extends the lifetime of the whole sensor network by extending the time until the first node is out of energy. Load balancing is also useful for reducing maintenance time and also message exchange traffic.

In the mobile sensor networks, managing the distribution of the network and also the load balancing is a complex problem, because of large number of the sensors and particularly because of the mobility of the modes. We can reduce this complexity by separating the network in different small parts. Clustering may be one of existing techniques to reduce this complexity, but none of the published clustering protocols consider any load balancing method in their clustering process.

To balance the load in the network, most of the clustering protocols use different parameters to choose cluster-heads. Cluster ID [11], connectivity degree [12, 13] and periodical cluster heads election [3] are used in order to share the load among all the nodes of the network. By applying cluster ID or highest connectivity methods, the same node may be chose as cluster-head every time, and that will result resulting in this sensor to drain its energy very fast. Changing the cluster head in the cluster, connectivity degree or periodical choosing, changes the topology of clusters frequently and this will impose huge overhead since all other cluster-heads have to be notified about the change.

The remainder of this paper is structured as follows: section II discusses some challenges in detail. Our proposal, its load balancing, failure management, clustering and decision making methods are described in section III, while section IV describes its functionality with some examples. Our simulations are resented in section V and section VI provides concluding remarks.

## II. CHALLENGES

As we described, in an ad-hoc wireless network with mobile nodes and therefore dynamic topology, mobility, failure and load balancing are important challenges.

One of the suitable solutions to manage the mobility of the nodes is dynamic clustering, but the existing clustering protocols use many assumptions which make them not able to address the needs of real application. Some algorithms are based on centralized control that makes them not to be scalable. Some algorithms use periodic rounds to change cluster head and elect a new one. The new cluster head will be fixed for one round, but in an ad-hoc network with dynamic topology, cluster topology may change during this period, and in this case a new cluster head must be elected. Therefore this type of algorithms will be good for networks with fixed or very low mobility nodes.

In addition, network dynamics resulted by node mobility and node state transitions due to the use of power management or energy efficient schemes may be detected as node failures or wireless link failures. Such a highly dynamic network greatly increases the complexity of failure management. Also, with bandwidth limitation in a sensor network the failure detector must generate a minimum number of control messages. However, the traditional failure detectors and management systems assume that all of the nodes of the network are synonymous, that means there is no difference between a node that was crashed *n* times in *t* hours, with a node that was crashed *m* times (m>n) in the same period of time. That will cause an unclear decision making and electing between different nodes. In the traditional failure detectors, when a node fails, it will be assumed as a dead node and we don't have a return of the node. That will be a restriction, for example, when a node is in maintenance.

Finally, as we said, none of the published clustering protocols consider any load balancing method in their clustering process. This can cause some clusters with high node density and other clusters with low node density. It is clear that in a cluster with a high node density, energy consumption will be very high and the cluster head will dead rapidly, this can cause network partitioning.

In order to solve these problems we proposed a new cluster based routing which can manage the mobility and failure of the nodes. To make it scalable the protocol is totally distributed, it has also a load balancing part. All event-processing and decision-making processes of this proposal use fuzzy logic. In the next sections we explain how our solution addresses these challenges.

## III. OUR PROPOSAL

As a result of our discussions in the last sections, Mobility and Failure management and Load balancing are main important problems to be addressed in ad-hoc sensor networks. To address these problems, our approach has 5 main parts: Fuzzy logic decision making, Clustering (Cluster-head election), Mobility management, Load balancing and Failure management. Figure 2 shows the main parts of the protocol. As we see in this figure, fuzzy decision making is the basic part of our proposal. That means, the 4 other parts of the protocol, use fuzzy logic to make decision or to process an event.

### A. *Some descriptions:*

Before explain our proposal, in this section we present some definition that we use in our protocol:

- {ALL}: Set of all sensors of the network.
- {Unknown}: the set of the nodes with unknown situation.
- {myNeighbours}: Neighbours of a node.
- {myChild}: Chile nodes of a parent node.
- A→B: Direct communication between A and B
- Base-Station (BS): Central computer that monitor the network or gather the information generated by sensors. (BS in figure 3)
- Zone-Head (ZH): Node (mobile or stationary) that communicate directly with BS (Z1 and Z2 in figure3):

  $\forall n \in \{ALL\} \mid n \rightarrow BS \Rightarrow n \text{ is a ZH}$
- Zone: set of one or more cluster. Each ZH constructs a Zone.

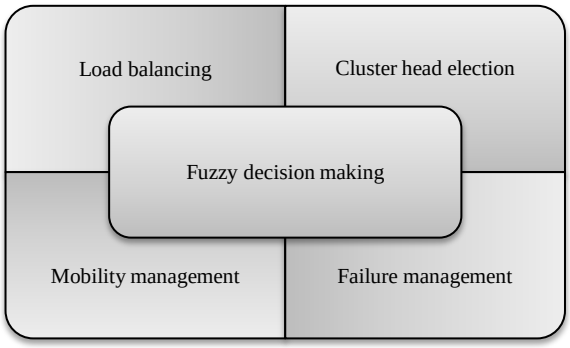


Figure 2. Building blocks of the protocol

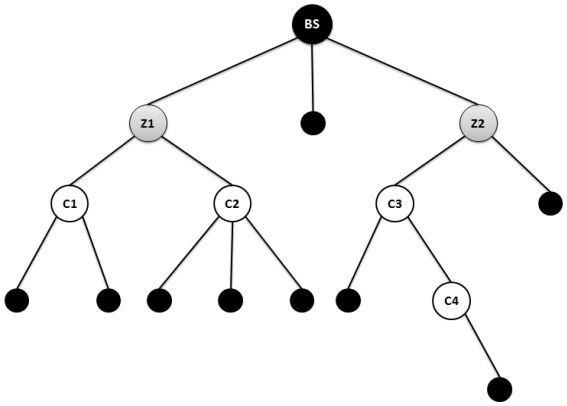


Figure 3. Zone and Cluster example

- Cluster-Head (CH): Node (mobile or stationary) that can communicate with one or more ZH or a node with some children that can communicate with other CH(C1,C2,C3 and C4 in figure 3):

  $\forall n \in \{ALL\}, \exists m, z \in \{ALL\} \mid n \rightarrow m \wedge z \rightarrow n \Rightarrow n \text{ is a CH}$
- Leaf-Node (LN): a node without child (Black nodes in figure 3):

  $\forall n \in \{ALL\}, \nexists m \in \{ALL\} - n \mid m \rightarrow n \Rightarrow n \text{ is a LN}$
- {Unknown}: the set of the nodes with unknown situation.
- Level: Level of a node is the number of hops between the node and BS.
- Mobility: This is a parameter to evaluate the movement of a node. You can find the complete description of this parameter in III.C.
- Quality of Link (QoL): this parameter shows the *Reliability* of the connection between a node and his parent. (See III.F)
- Successful clustering: In this algorithm, we have a successful clustering if and only if:

  $\forall n \in \{ALL\}, \exists m \in \{ALL\} - n \mid n \rightarrow m$

  and also:

  $\forall n,m \in \{ALL\} : (n \rightarrow m) \Rightarrow m \in \{ZH\} \cup \{CH\}$
- *Invite*: it is a message, between the nodes to exchange the information. This message is used by a ZH or CH to invite the other nodes to join them. (See III.F)
- *Hello*: is used by a node to announce a change or event to its neighbors.
- *Find*: it will be used by nodes to find a new parent.
- *Join*: this message is used by a node to answer an *Invite* message. If the node has just one possible candidate to choose as its parent, it will indicate that, in this message.
- *Quit*: a node sends this message to its parent node to advertize leaving it.
- *Join_Other*: It is a message that a ZH or a CH sends to one of its child to ask him to find another parent. This will be when the ZH or CH:
  - received a new request of join from a node with no other possible parent, and the admission condition is not satisfied. Therefore it must reduce its load by reducing number of its child.
  - is in a low level energy state,

### B. *Fuzzy decision making*

In our proposal, we use fuzzy logic because it is capable of making real time decisions, even with incomplete information. Conventional control systems rely on an accurate representation of the environment, which generally does not exist in reality. Fuzzy logic systems, which can manipulate the linguistic rules in a natural way, are hence suitable in this respect. Moreover fuzzy logic can be used for context by blending different parameters – rules combined together to produce the suitable result. In the next sections we will explain role of fuzzy decision making in different parts of our protocols.

In [5], a fuzzy logic approach to cluster-head election is proposed based on three descriptors - energy, concentration and centrality. They improved LEACH routing algorithm [3] by using Fuzzy Logic and the cluster-heads are elected by the base station in each round by calculating the chance each node has to become the cluster-head by considering three fuzzy descriptors – node concentration (Number of neighbors), energy level in each node and its centrality with respect to the entire cluster. As we said, the proposed algorithm in [5] is centralized and therefore is not scalable and also cannot support the mobility of the nodes.

But, which descriptors we can choose in a mobile network? In a dynamic network with mobile nodes, can we use concentration and centrality as factors to decision making in a distributed approach? The answer is that in a distributed approach each node chooses its parent by processing its local information therefore centrality of a node cannot be suitable to its decision, this parameter will be good for a centralized algorithm to reduce the routing hops between CH and nodes in the same cluster, but a node cannot have a general view of network and thus cannot evaluate this parameter for a CH. In addition, in a dynamic network with mobile nodes the number of neighbors is not a good descriptor, because location of the node is not fixed and we cannot be sure about number of neighbors at each moment. In the next sections we explain the main parts of our proposal with more details.

### C. Mobility management

In an ad-hoc sensor network with mobile nodes, we must be able to detect the movement of the nodes in order to have a correct image of network in each time. We know that the nodes, in these networks, are not integrated with systems like GPS, but without GPS how can we detect the movement of a node? Assume that distance between two nodes A and B in time t is X, and in time t+1 the distance is Y where Y>X. By using this information we cannot say which one moved? A, B or both of them? Without a lot of calculations or a GPS system, we cannot answer this question. To solve this problem we proposed a new parameter named *Mobility*. This parameter shows frequency of parent, level or zone change of a node. (Number of CH or level change of a node in his life time). Therefore each time that the node changes its CH or his level, it must increment value of a variable named *Change* and divide it to his lifetime to find the Mobility. It is clear that the mobility of a fix node can be greater than zero, because of the mobility of his parent.

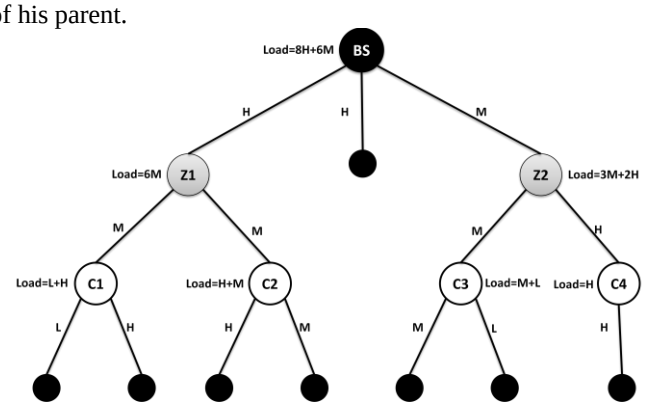


Figure 4. A load tree

### D. Load Balancing

Our load balancing strategy considers the cumulative load of data traffic from child nodes in a load tree on their parent nodes. We use *Load tree* and *admission condition* for load balancing.

#### 1) Load tree

Figure 4 shows a sample *load tree*. The *load tree* is rooted in the base station. The load of child sensor nodes adds to the load of each upstream parent in the tree. Hence, the sensor nodes nearest the base station will be the most heavily loaded. The goal of this load balancing technique is to evenly distribute packet traffic generated by sensor nodes across the different branches of the Load tree. But here Load has a special definition. Load is the sum of the QoL (see next section) between a node and his children. It is a new definition that can be used as a new parameter in QoS. In a load tree, the weight of each link, in load tree is QoL between each node and his parent, and load of each node is the sum of the QoLs between the node and his child.

#### 2) admission condition

In order to balance the load between nodes of the network, we use admission condition, which is the condition of accepting a new child node in node n:

$$(n.Load + NewChild.QoL)/\ n.numberLowLevelNodes + 1 \leq n.QoL$$

In this formula the number of low level nodes means the number of all the nodes that are connected to this node directly or indirectly.

### E. Failure management

In this approach we have a parameter named *failure* shows the failure history of a sensor. This parameter will be computed by using the number of sensor's failures during its lifetime. Like the other used parameters, it is also a Fuzzy variable that has 3 levels: *High*, *Medium* and *Low*. That is not a static parameter, that means, for each node it can change from *Low* to *High* and also *High* to *Low*. To manage the failure of nodes or links, each time that the node detects a failure in a neighbor, it updates failure parameter for this node:

$$failure = fuzzy\ (n / L);$$

where *fuzzy* is a function to convert decimal value to fuzzy value, *n* is number of neighbor's failures and *L* is our life time, therefore the *failure* parameter has different value for each node in the other nodes. In each network, due to the mobility or failure frequency of the nodes, BS will defines a update period, in which each node will update the *failure table*, therefore *failure* and *Reliability* parameters are really dynamic parameters that can change not only from *Low* to *High* but also from *High* to *Low*.

### F. Cluster head election

We use four parameters: *Energy level* of the node (Battery charge), *Mobility*, *Quality of Link - QoL* (*Reliability* between a node and his parent) and the *failure*, to evaluate a node that is candidate to be a ZH or CH. These parameters will be the *Fuzzy Logic Descriptors* and each of them has three possible values: *low*, *medium*, *high*.

Therefore we have 81 rules to evaluate a node. The result of the rules will be *Reliability* with five possible levels: *Very Low, Low, Medium, High and Very High* (See figure 5). In each *Invite* message the node will send necessary information to be evaluated by the other nodes, as like as: *Energy level* and *QoL*, and the node will compute the QoL of the connection between candidate and itself. The QoL of a node is *Reliability* parameter that he was calculated for his parent. This parameter helps us to choose the best parent node, a node with maximum energy, maximum stability, and higher reliability of connection. By finding the Reliability of a candidate we must evaluate the chance of the candidate to be a parent. To restrict depth of network's tree when a node receives more than one Advertisement, it will choose the node with smaller level, therefore we use:

$$Chance = Reliability / Level$$

### G. Different processes in the nodes

This section presents the process in the nodes to answer different messages on them. In Table I, we find action of a ZH or a CH to answer *Join* message from a node. First, the ZH (or CH) will check if *n* is in the {Unknown} or not, if *n* is in this set, it will remove *n* from the set, also if *n* is not his neighbour, *n* will be added into {myNeighbours}. Then the ZH will verify *admission condition*: If OK, it will accept *n* as a new member; If KO, and *n* hasn't any other choice, ZH sends a OK message to *n* and chooses one of his actual members with smallest QoL than *n* and will send a request the join another one and will wait to a response, if the answer is KO it will choose another child node to send *Join_Other*.

Table II shows that when a node revives *Invite* from a CH or a ZH, *c*, it will add the node in its {myNeighbours} set and then if it has not a parent, it will send a *Join* message to *c*. If the node has already a parent, it will run the Fuzzy decision making function to know if c is better than its current parent or not: If c is better it will choose it by sending a Join message to it and by receiving OK from c, it sends also a Quit message to its current parent.

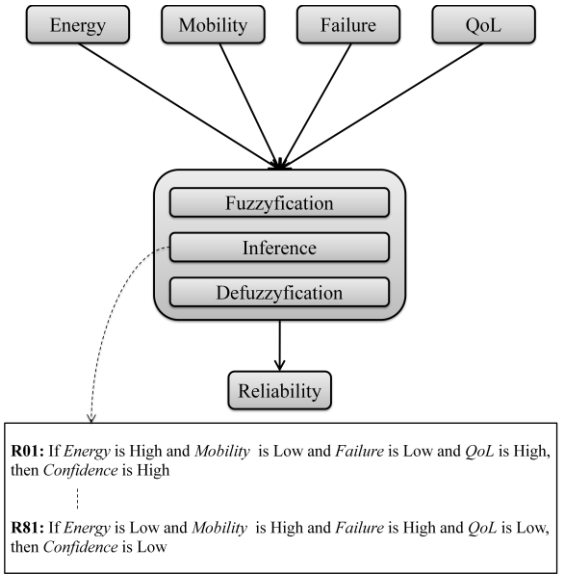


Figure 5. Cluster head election

Table III shows that in a node, by receiving Join-Other message, the Node will send Join message to its neighbours and if it receives OK, from one of them, it will send OK to its parent node, if no, it will send KO.

```
Receive (Join from n)
 accepted = False;
 If   n∈ {Unknown} then {Unknown}={Unknown} – n;
 If  n∉{myNeighbours} then
        {myNeighbours}={myNeighbours}+ n;
 If (admission condition is OK) then  accepted = true;
 Else // admission condition is KO
    If (Join.NumberOfCandidates is1) then  accepted = true;
     For  c ∈ {myChild} |
                (c.QoL is Min) & (n.QoL > c.QoL)
          Send (c,  Join_Other );
          Wait (receive (msg, c));
          If (OK or Timeout)
             accepted=true;
             {myChild}={myChild}-c;
             If (Timeout)
                {Unknown}={Unknown} – c;
                {myNeighbours}={myNeighbours}- c;
If ( accepted == true)
     {myChild}={myChild}+ n;
     Send (n, OK);
else
    Send (n,  Join_Other );
```

TABLE I. RECEIVE JOIN IN A ZH OR A CH

```
Receive (Invite from c)
        If (this.CH==null)
            Send (c, Join)
            Wait (receive (OK from c);
        Else // this.CH ≠ null
             Best = FuzzyFunction (this.CH, c);
             If (Best == c)
                  Send (c, Join);
                  Wait (receive (OK from c)
                  If (OK)
                     Send (this.CH, Quit);
                     Join c;
```

TABLE II. RECEIVE INVITE A NODE

```
Receive (Join-Other from n)
        Found=false;
        For all m ∈ {myNeighbours}
             Send (m, Join);
             Wait (receive (OK from m)
             If (OK)
                  Found=True;
                  Break;
        If (Found = true) then
             Send(n, OK)
             For all c ∈ {myNeighbours} - m
                       Send (c, Hello);
        Else
             Send(n, KO)
```

TABLE III. RECEIVE *JOIN_OTHER* IN A NODE

## IV. SOME EXAMPLES

In this section we will present our protocol with some examples. Let's start with a simple example. Let's take a parent selection scenario in the network presented in figure 3. Figure 5 shows the different state of the scenario. In this network node *n* searches a parent's node. Here are the different steps of the procedure:

a. *n* diffuses *Find* message.

b. Its neighbors, y (a leaf node), C2 and C3 (Cluster heads) who have received the *Find* message send a *Invite* message as answer. By the *Invite* message, they send their *Energy*, *QoL*. Node *n* then will search their *Failure* and *Mobility* in its *neighbor* table. If there is no value the default value for *Failure* and *Mobility* is *Low*.

c. By using four parameters, *Energy*, *QoL*, *Failure* and *Mobility*, *n* will run a *Fuzzy* function to evaluate the candidates.

d. In this example *y* was the best node to be parent of *n*. therefore, *n* send a *Join* message to *y*.

e. By receiving Join message, y verifies *Admission condition*.

f. *Admission condition* was ok, therefore y send a *OK* message to *n*.

g. *n* joins *y*, and the state of *y* will be changed to a *Zone head* (Z3)

h. As state of the *y* is changed (Z3), *it* diffuses a *Hello* message to announce this change.

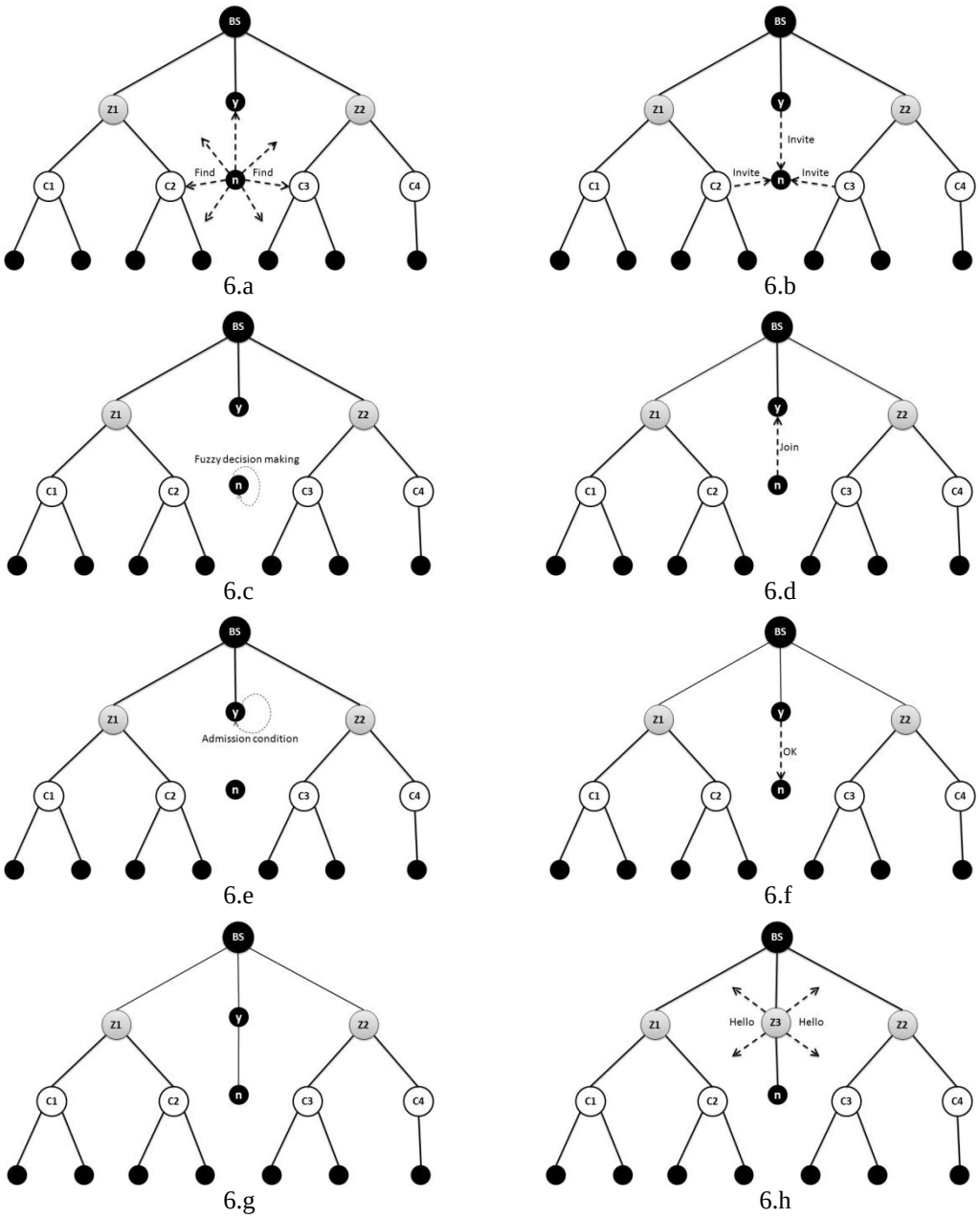


Figure 6. Example 3

As second example figure 7 shows a scenario with a node that has one possible choice:

a. *n* diffuses *Find* message.

b. Its neighbor, Z1 (a zone head) receives the *Find* message send an *invite* message as answer.

c. *n* received just one *invite* message, that means it has no other choice. It runs the fuzzy function to compute QoL. Therefore it sends a join message to Z1.

d. *n sends a Join message to Z1.*

e. By receiving Join message, Z1 verifies *Admission condition*. But it is not OK.

f. *As n hasn't any other choice, Z1 send a OK message to n and a Join_Other message to C2, one its child.*

g. *n* joins *Z1*, and C2 sends a Join message to Z2.

h. *Z2 verifies Admission condition which is OK in this example.*

i. Z2 sends a OK message to C2.

j. C2 joins Z2 and sends a Quit message to Z1. By receiving Quit message, Z1 delete C2 from its Child set.

k. This figure shows final cluster of the network.

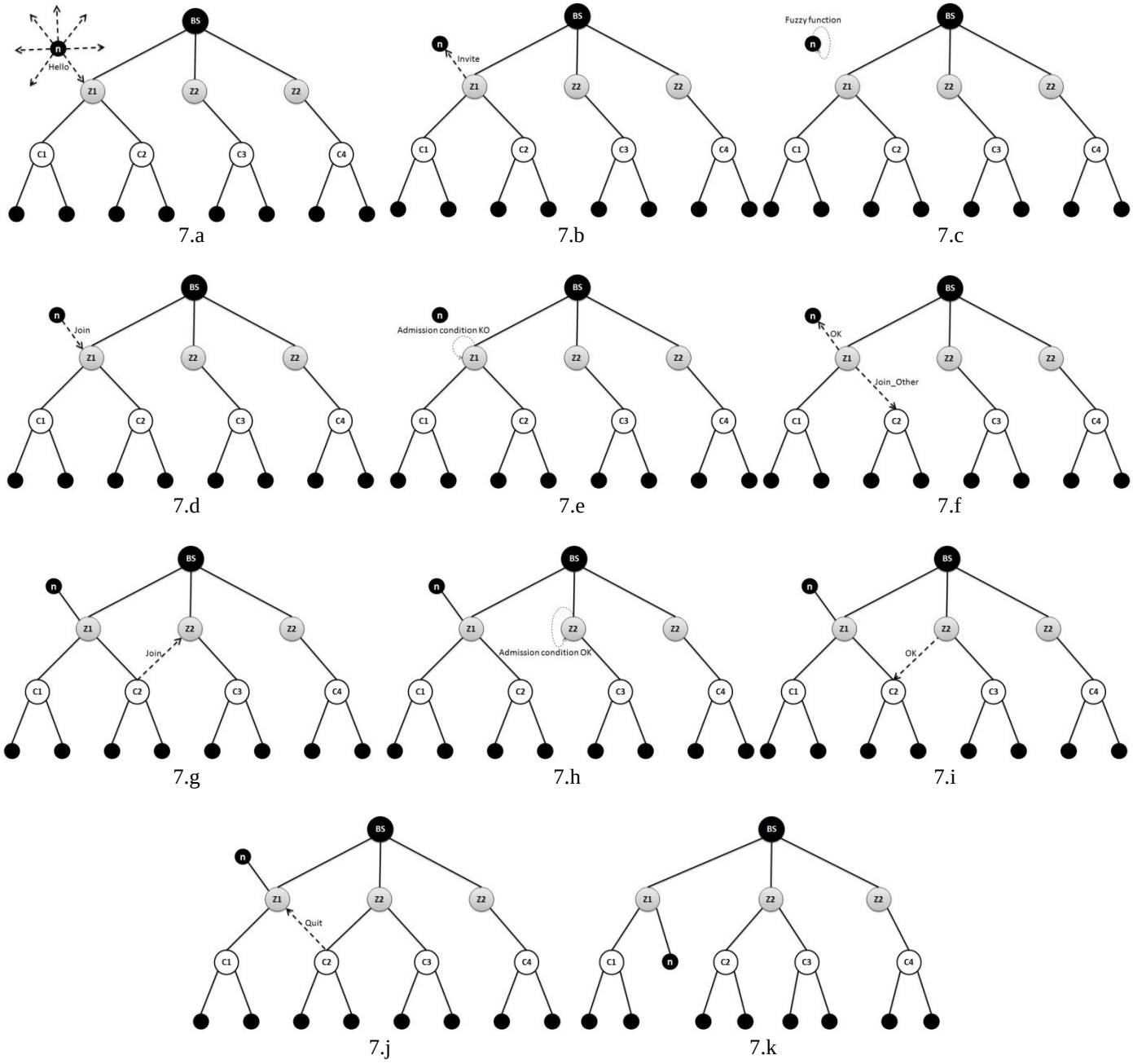


Figure 7. Example 2

| | |
|---|---|
| Node Number | 50 |
| Surface | 100m x 100m |
| Transmission range | 15m |
| Data transmission rate | 15 packet/sec |
| Failure model | Random |
| Packet size | 128 bytes |
| Initial Energy | 5J |
| Energy consumption (Calculation, receive and send) | 10 nJ/bit |

TABLE IV. SIMULATION PARAMETERS

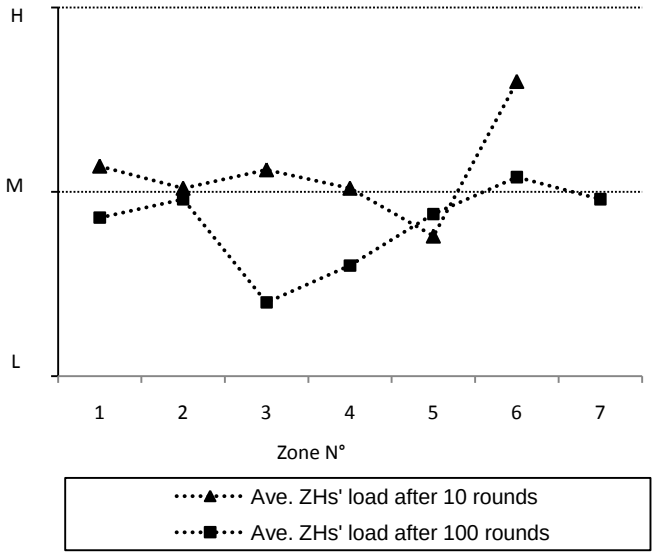


Figure 8. Network's ZHs' load after 10 and 100 rounds

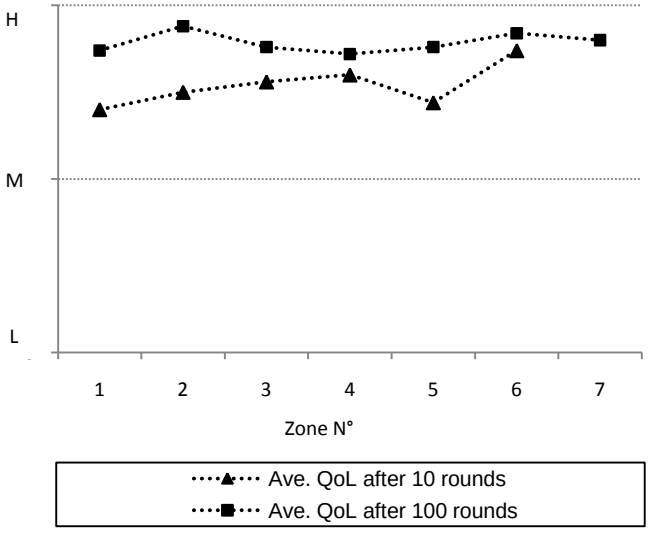


Figure 9. Average QoL in zones

## V. EVALUATION

In this section evaluated performance of our proposition will be presented and will be compared with ZRP routing protocol. We integrated OPNET simulators to implement the physical and MAC layers, with an application developed in C# to implement Fuzzy rules.

In our simulation we focus in load of the zone heads and average QoL in each zone and network's data delivery ratio as performance metrics. Table IV, shows our simulation parameters.

### *A. Some assumptions*

To evaluate our protocol, and to reduce simulation complexity, we have these assumptions:

- We used random waypoint model [14] in our simulations. The Network consists of several low mobility wireless nodes, just 20% of the nodes are mobile and their speed is 0.5 m/s.
- Each node is initially placed at a random position within in the simulation area.
- Round: the period of time in which all the mobile nodes change their zone.
- To focus on the assessment of the performance of the proposed algorithm, we do not generate any user data traffic during a simulation.
- All the nodes are able to detect correctly the failure of the other nodes.
- When a node failed or crashed, it is not dead, it will be return to the network after a variable time, $t \neq \infty$.

### *B. Simulation results*

#### *1) ZHs' load*

Figure 8 shows load in ZHs of the simulated network. We find in this figure that after 10 rounds, network has 6 zones and load of 4 ZHs are medium, one between medium and high, and one between medium and low.

After 100 rounds network has 7 zones and 6 ZHs have a load between medium and low and load of one of them is medium. The average of load in ZHs after 10 rounds is medium and after 100 rounds is between medium and low. These results show that our protocol can balance correctly the load between ZHs and CHs.

#### *2) Average QoL in the zones*

As we explained before, QoL in a zone shows the connectivity of the nodes. A QoL with value of high shows a good connectivity between the nodes and a low QoL shows unstable connection between the nodes.

Figure 9 shows average QoL in the zones of the simulated network. We find in this figure that after 10 rounds, network has 6 zones and average QoL in the zones in between medium and high, and after 100 rounds network has 7 zones with average QoL near to high. These results show the efficiency of our protocol to establish reliable and stable connections between the nodes.

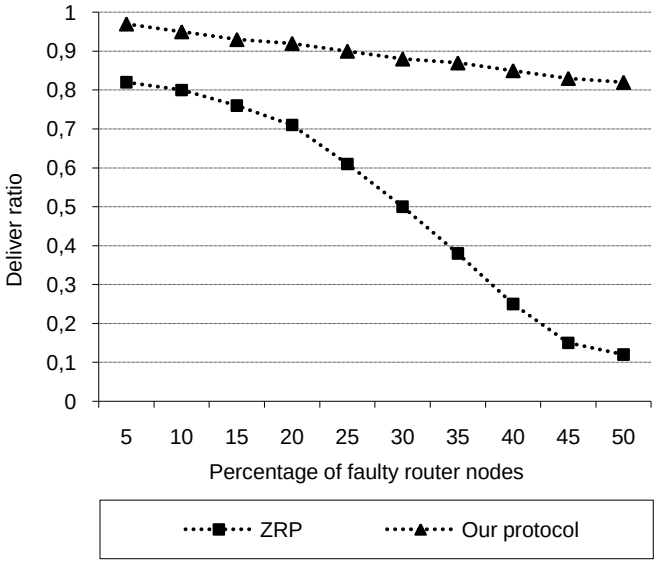


Figure 10. Delivery ratio

*3) Data delivery ratio*

In this step of simulation, we focus on the network delivery ratio. Network delivery ratio defines as the total received packets to the total sent packets in the sensor network. We compared this metrics in ZRP routing protocol and our protocol.

As we said 20% of the nodes of the network are mobile and for the simulation, in each step, we change the number of faulty nodes from 5 to 50 percent. We can find the simulation results in figure 10. The simulation shows that our protocol increases the data delivery in the network and greatly adapts mobility and failure of the nodes.

## VI. CONCLUSION

A fuzzy logic based routing protocol is proposed in this paper. Stable route recovery, high data delivery ratio and good load balancing are the main characteristics that the proposed protocol adds to the ad-hoc sensor networks.

The performance of the protocol can be optimized through finding the best values for the used factors in neighbors table for different network topologies and sizes and it adapts also with mobility and failure of the nodes.

This protocol is especially effective in networks that use sensor nodes to data aggregation and in which the data delivery ratio is important and the nodes are mobile, like health monitoring sensor networks. In such networks health events and information is sensed by several nodes and therefore, this protocol can help the network to deliver sensed events and avoid of data loss in the network.

In all of the data acquisition networks, like health monitoring systems, either the data is collected from the network periodically or on an occurrence of an event, in such systems, the data are highly vital to have a stable monitoring and have a minimum number of faulty alerts.

Hence, none of them adapts completely themselves to the failure of the nodes and the temporal variations in data delivered by the sensor network. This necessitates the use of a routing protocol that readily adapts to the failures of the nodes and changes in the data delivery rate. The simulation results show that the proposed protocol is well suited for such applications.

## REFERENCES

[1] C. R. Lin and M. Gerla, 'Adaptive clustering for mobile wireless networks', IEEE Journal on Selected Areas in Communications 15(7), pp 1265-1275, 1997

[2] Z. J. Haas and M. Pearlman, “The zone routing protocol (ZRP) for ad hoc networks,” IETF Internet Draft, draft-ietf-manet-zone-zrp-04.txt, July 2002

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An Application-Specific Protocol Architecture for Wireless Microsensor Networks,” IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 660–670, Oct 2002

[4] O. Younis and S. Fahmy, “HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks,” IEEE Trans. Mobile Computing, vol. 3, no. 4, pp. 366-379, Oct.-Dec. 2004.

[5] I. Gupta, D. Riordan and S. Sampalli: Cluster-head Election using Fuzzy Logic for Wireless Sensor Networks. Faculty of Computer Science, Dalhousie University. IEEE Communication Networks and Services Research Conference, 2005

[6] K. Marzullo, “Tolerating failures of continuousvalued sensors,” ACM Transactions on Computer Systems, vol. 8, no.4, pp. 284-304, November 1990.

[7] D.N. Jayasimha, “Fault tolerance in multi-sensor networks,” IEEE Transactions on Reliability, vol. 45, no.2, pp. 308-15, June 1996.

[8] R. R. Brooks and S. S. Iyengar, “Multi-Sensor Fusion: Fundamentals and Applications With Software,” Prentice- Hall, 1997.

[9] P. K. Varshney, “Distributed Detection and Data Fusion,” N.Y.: Springer-Verlag, 1997.

[10] G.D. Hager, “Task-Directed Sensor Fusion and Planning - A Computational Approach,” Kluwer Academic Publishers, 1990.

[11] D.J Baker and A. Ephremides, "A Distributed algorithm for Organizing Mobile Radio Telecommunication Networks", in the Proceedings of the 2nd International Conference in Distributed Computer Systems, April 1981.

[12] M. Gerla and J.T.C Tsai, .Multicluster, mobile, multimedia radio network,. ACM/Baltzer Journal of Wireless networks, Vol. 1, No. 3, pp. 255-265, 1995.

[13] A.K. Parekh, "Selecting Routers in Ad-Hoc Wireless Networks", Proceedings of the SBT/IEEE InternationalTelecommunications Symposium, August 1994

[14] Guolong Lin et al. "Mobility Models for Ad hoc Network Simulation",IEEE INFOCOM 2004