ISSN : 2350-0743

www.ijramr.com

International Journal of Recent Advances in Multidisciplinary Research
Vol. 08, Issue 10, pp. 7248-7252, October, 2021

RESEARCH ARTICLE

GENERALIZED CLUSTER CONSTRUCTION MECHANISM FOR LINEAR WIRELESS SENSOR NETWORKS (G2CMLN)

[1,*]Abdourakhmane Fall, [1]Moussa Dethié Sarr and [1]Cheikh Sarr

1Computer Science Department, Université Iba Der Thiam de Thiès, BP 967 Thiès, Senegal

ARTICLE INFO

Article History:
Received 10th July, 2021
Received in revised form
27th August, 2021
Accepted 20th September, 2021
Published online 30th October, 2021

Key words:

Linear Wireless Sensor Networks, Clustering Algorithm, Automatic Topology Construction.

ABSTRACT

A Linear Wireless Sensor Network (LWSN) is a grouping of sensors arranged in a linear topology with or without junction zones. LWSNs are applied in several areas involving, among others, gas pipelines, road or rail infrastructure, border surveillance infrastructure, waterways, oil, etc. The (LWSN) are subjected to the same constraints as those subjected by the traditional networks (non-linear), energy saving, overload, delays, latency etc. To optimize the energy consumption of the network, among other things, one technique consists in organizing the network into several groups of nodes called clusters. Several algorithms for the construction of clusters have been proposed in the research. The majority of these propositions concern classical networks (non-linear). Recent proposals for clustering algorithms relating exclusively to linear networks have been made in the current research. These last proposals concerned k-redundant networks (number of nodes within transmission range) with a k previously assumed as an assumption (k = 2). On the other hand, these proposals do not take into account the addition, loss and removal of nodes in the sensor network. Therefore, faced with these situations, the clustering algorithms already proposed may be unsuitable. In this article we propose a generalization (extension) of a clustering mechanism for linear wireless sensor networks. The algorithms that we propose in this work will allow the linear sensor network to be able to react to the addition of new nodes as well as to the loss and deletion that the network nodes may experience. The solution that we present also works for all k-redundant networks regardless of the value of k, it is not fixed on a constant value of k as was the case for the solutions previously proposed in current research. The solution was evaluated with the Castalia / Omnet ++ simulator, on a topology with several junction zones. The results obtained showed a complete reconstitution of the clusters with a relatively very fast reaction time, and this as well for the addition of nodes as in the face of the losses and deletions of nodes of the linear wireless sensor network.

INTRODUCTION

The RCSFs are arranged in a given shape depending mainly on their application. We are now seeing more and more new applications of WSNs on linear infrastructures such as railway environments, bridges, gas and oil pipelines, borders, volcanoes, rivers etc. (7) (8) (9) (10) (11), These environments are characterized by one or more portions of lines with branches. This has given rise to the existence of a new type of wireless sensor network topology known as the Linear Wireless Sensor Network (LWSN) (6) (4) (5). Linear wireless sensor networks are a special case of WSN in which the nodes are arranged in a linear fashion. LWSNs are divided into two categories of topologies: Networks with strictly linear topologies. In which we note the existence of a single linear line starting from the sink, and networks with linear topologies with junction zones, in which we note the existence of several linear lines, some of which form crossing points called junctions.

Corresponding author:** ***Mr. Abdourahman Fall **,**
Computer Science Department, Université Iba Der Thiam de Thiès, BP 967 Thiès, Senegal.

In linear wireless sensor networks, the nodes are both data sensors and relay sensors and have at least one neighbor in both linear directions of the network, the number of neighbors is determined by a redundant k factor by general rule. In practice, the number of neighbors of each node depends on its position in the network, on the range of its radio signal in relation to the density of the nodes. In the LWSNs, the junction zones present a significant degree of neighborhood due to the high density therein. Therefore the sensors located there will have a higher number of neighbors compared to other nodes not located in a crossing zone. LWSNs exist according to several architectures, one of them consists in making a topology in which all the nodes located in a strictly linear zone have the same number of neighbors in the direction of the sink as well as in the opposite direction. This topology is said to be k-redundant with k the number of neighbors in each direction starting from the node. In order to optimize the energy consumption of the network, among other things, one technique is to organize the network into several groups of nodes called clusters. Several techniques for building clusters have been proposed (6). The majority of these proposals concern non-linear networks. Recent proposals for clustering algorithms exclusively concerning linear networks have been

made. These proposals concerned k-redundant networks in which the factor k was previously assumed as a starting hypothesis. On the other hand, these proposals did not take into account the addition, loss and removal of nodes in the sensor network (1) (2). In (1) the authors propose (2CMJ) a cluster construction mechanism for linear wireless sensor networks subject to the same constraints as those cited below. In this paper we propose (G2CMLN) a generalization of (2CMJ) in which the addition, loss and removal of nodes are taken into account in the algorithm. The solution works for any k-redundancy network regardless of the value of the k redundancy factor. The article will be organized as follows. In the first part we will talk about the state of the art by studying in detail the related works on which our work is based. Part 2 will be devoted to the presentation of our contribution as a solution to the problems affecting the techniques already proposed in current research. In the third part, we will present the results of the simulations carried out. Finally, the last part will concern the conclusion and some perspectives.

Related work

K-redundant topologies: In linear WSNs, node connectivity is an extremely important parameter for data relaying and especially for network availability. In (12), the authors propose k-redundant architectures. In these architectures, each sensor node of the network has k neighbors in the direction of the sink and as many neighbors in the opposite direction of the sink. In other words, a network is k-redundant if each neighbor node has 2 * k neighbors with equal proportion in both directions. For a 1-redundant LWSN (Figure 1), the loss of a sensor node can cause network partitioning. For k ≥ 2 (Figure 2), the LWSN provides increased availability and reliability versus node loss, for a 3-redundant network and so on. Consequently, the greater the degree of neighborhood, the better the network is in terms of availability because it is more resistant to possible losses of network nodes. This shows the importance of redundancy on the issue of linear wireless sensor network availability.

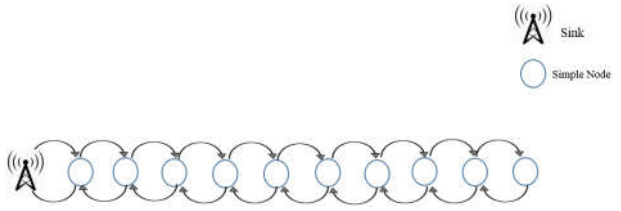

Figure 1. LWSN: 1-redundant

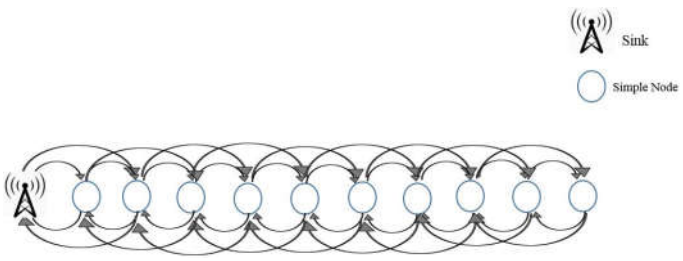

Figure 2. LWSN: 2-redundant

Cluster construction mechanism for LWSNs with junctions (2CMJ)

In (1) the authors propose (2CMJ) a cluster formation technique for wireless sensor networks with linear topologies comprising junction areas. (2CMJ) comprises two phases:

A first phase of discovery of the neighborhood and in which the nature of the nodes is determined, and a second phase of cluster formation in which the construction of the clusters takes place. In the first phase, the nodes determine their area of belonging, once determined, their nature is inferred. Two kinds of nodes are possible with this configuration: simple nodes and junction nodes. Indeed, in an LWSN with junctions, we find two areas: the strictly linear zones in which the nodes have, at most, 2 * k neighbors, with k the redundancy factor. We have, beside, the zones of junctions in which the nodes have at least 2 * k neighbors. So we have the simple nodes (SN), those belonging to strictly linear zones, and the nodes of junctions (JN), that is to say, the nodes being in area of junctions.This first phase takes place as follows:

First all the nodes exchange hello packets, if a node receives a maximum of four Hello packets from different neighbor nodes, Hello_receive ≤ 4, then the node self-elects as a single node (SN). If the number of Hello packets received from different neighboring nodes is strictly greater than 4, Hello_recu> 4, then the node is considered to be a junction node (JN). After defining the natures, each node communicates its nature to its neighbors and records all of its neighbors in a database. At the junction zone level, the authors define another type of node which they will call gate nodes (GN). A gate node is a node connecting a junction zone to a strictly linear zone. The number of GNs existing in a junction area is proportional to the complexity of the area. Simple nodes (SN) having a junction node (JN) as neighbors communicate the list of their broadcast neighbors, the (JN) neighbors receiving this packet check if, among the sending nodes, two of them are one-hop neighbors (close neighbor). If so, then the node is considered a gate node (GN).Following this the second phase is triggered.

In the second phase, the clusters are built gradually. The authors propose two types of cluster: Simple Clusters (SC) made up only of simple nodes, and Junction Clusters (JC) made up only of junction nodes. The phase consists of two stages, in the first stage the authors use the M2CRL algorithm (2) for the formation of clusters at the level of strictly linear zones. In the second step, at the level of the junction clusters (JC), the node closest to the sink is elected as Cluster Head. The cluster head coordinates and organizes the creation of clusters within the (JC) as follows: The CH searches for gate nodes (GN) which are in its cluster, by broadcasting a Hello-GN in the cluster. once received, only the GNs will respond to this packet, the response is transmitted to the CH. Following this, the cluster head registers the various (GNs) that are in the cluster. Then the cluster head sends two types of build packets in the cluster:

- A construction packet of type M2CRL (2), this packet is sent to neighboring non-GN nodes. The latter take position 2 in the cluster (Pn = 2), the other variables (Cid, Cp) will be the same as those contained in the construction packet.
- A second construction packet sent to the different gate nodes (GN) of the cluster. The CH sends a construction packet containing an index ***i***, which will be unique for each (GN). The index will be used for the calculation of the Cid of the following Simple Cluster (cluster following

the (GN). Take position 2 in the cluster (Pn = 2), the other variables (Cid, Cp) will be the same as those contained in the construction packet. The (GN) will be responsible for calculating the Cid (x) of the following Simple Cluster (CS) thanks to the equation (Equation 1). The following CS puts are Cid at the value (x) contained in the construction packet issued by the (GN)

$$x = C_{id} + \left(\frac{p}{P_i}\right)^i \quad with \begin{cases} C_{id} & \text{CJ identifier} \\ p & \text{LWSN depth} \\ P_i & \text{Simple Cluster (SC) cardalite defined by the sink} \\ i \geq 1 & \text{index received by the gate node (GN)} \end{cases} \tag{1}$$

The authors tested and evaluated their proposal, and obtained very good results concerning the formation of clusters. They also measured the construction time of their technique, the results also obtained at this level are more than satisfactory. However, their technique suffers from a few constraints, particularly those of adding new nodes and losing nodes. Indeed in their technique they do not take into account this changing and dynamic aspect of the network, which is also very common in LWSNs

Generalized Cluster Construction Mechanism for Linear wireless sensor Networks (G2CMLN): In this paper we propose (G2CMLN) a generalization of (2CMJ). The (G2CMLN) technique will be applicable for k-redundant network regardless of the value of k, so our solution solves the problem of the fixed (constant) k-redundant factor in (2CMJ). Our solution also takes into account the dynamic aspect of the wireless sensor network. Indeed, wireless sensor networks are very rarely fixed (in the sense that does not change) they are often subject to a change of node, due to the loss of a certain node in the network. The resulting node losses therefore lead to the addition of new nodes in order to replace lost sensors and increase the availability of the wireless sensor network.

Algorithm: (G2CMLN) has two stages in its operation: a first phase called phase of discovery of the factor k,k being the degree of redundancy of the network of sensors. This first phase also includes the phase of discovery of the sink as elaborated in (2CMJ) (1). The second phase concerns the construction of the clusters. This phase reuses the same operation as that of (2CMJ) in the process of creating clusters. When new nodes are added, the network reacts automatically to restore the new settings of the new node, thus introducing it into the cluster according to its position. In the event of a loss of nodes, the network automatically readjusts the parameters of the cluster affected by the loss, and information relating to the loss of the node is transmitted to the sink.

K factor discovery phase (Algorithm 1): First, the sink broadcasts a discovery packet to its neighbors. Then the nodes receiving such a packet will in turn broadcast a Hello packet to their neighbors, these Hello packets will be acknowledged. Following this, each node will count the number of acknowledgments ***a*** it has received, the respondents will also be recorded in the neighborhood database of the node in question. After determining the value of ***a***, the nodes respond in unicast to the sink through a Response_To_Sink (RTS) packet containing the value of ***a***. After receiving all the answers, the sink will proceed as follows:

- The number of responses received will be equal to the network redundancy factor k***k=number_of_RTS***$_{received}$
- Neighboring nodes will be ranked relative to the value of ***a*** they hold. The neighbor with the lowest value of ***a*** will be the closest to the sink, the one with the highest value of ***a*** will be the farthest. In other words, the degree of proximity of a neighboring node is determined by the value of ***a*** that it holds.
- After determining the k factor, the sink initiates the construction phase exactly as in the algorithm (2CMJ). Except that for (G2CMLLN) the k factor will be contained in the construction packages in addition to the other parameters that existed in a construction package (2CMJ) (1).

Cluster construction phase: (G2CMLN) uses the same construction algorithm as (2CMJ). In the construction phase, the k factor found in the packet will help in the process of creating the clusters. Indeed, as the technique (2CMJ) stipulates in its construction phase, a node located in a strictly linear area will wait to receive k packets before applying the rules laid down by the algorithm. The same principle is applied in (G2CMLN). In the junction areas, too, the same operation as (2CMJ) is applied here.

Loss of nodes (Algorithm 2): The objective here is to detect the loss of nodes in the network early. Indeed if no mechanism is put in place to detect node losses, the sensor network will not be able to know that such or such nodes are dead. In the case of a linear sensor network this could have serious consequences, even rendering part of the network unavailable. Let us imagine, for a k-redundant network, that k consecutive nodes of the network are dead, this would immediately make unavailable all the nodes farther from the sink compared to the k lost nodes.

Operation: We denote by d the threshold energy value (in joule) of a given node, from which the node is considered to be a critical node (at low energy level). Indeed if the energy level of a node reaches the value d, the node sends an alert packet to its nearest suppurative neighbor (*Nsup1*). The latter, after receiving an alert packet, will periodically send a hello message to the critical node in order to test its presence. Each Hello message will be acknowledged, if the node does not receive an acknowledgment, the sending node deduces that the node is dead. Information concerning the loss of the node is transmitted to the sink along with information relating to the lost node (Node_Id, Position_Pn in the cluster, and Cluster_Id).

Adding a new node (Algorithm 3): In this phase the sensor network reacts to the addition of a new node in order to assign its parameters to it and allow the nodes of the cluster to update their parameters.

Operation: Once installed in the network, the newly added node in the network broadcasts a packet of type (NNA) (New Node Added) to all its neighbors. After receiving a packet (NNA) all the neighbors respond by sending in unicast a packet of type (R_NNA) (Response to New Node Added), the packet (RNNA) contains the Pn and Cid of the sending node. After receiving all the packets (RNNA), the new node added deduces its position, Pn, in the cluster and its membership cluster Cid through the following process:

The new node added is based on a weighting system to define its parameters. We denote by$A = i(Pn) + Cid$; the complex number defined by the pair (Pn, Cid), A $\in \mathbb{C}$, with Pn the position of the node in its cluster and Cid the identifier of the

cluster (which constitutes its Euclidean position with respect to the sink). Each neighbor of the newly added node is defined by the complex number A given by equation (2). After receipt of all the packets (R_NNA), the new node associates for each neighbor, having responded, a complex number A defined by the parameters of the neighbor itself.

Algorithm 1: Discovery phase

```
Result: This phase allows the sink to determine the degree of
        redundancy as well as its neighbors and their degrees of
        proximity.
Data: neighbor: Sink_neighbors() int: v int: cmpt=1
Initialization: v=0
struct {
    MACAddress source
    int v
} neighbor
begin
    if (isSINK) then
        Pcst=Packet(Discovery)
        Send(Pcst, Broadcast)
        if (Receive_Response_Discovery) then
            neighbor x
            x.source= Address_Packet
            x.v= v_Packet
            Add(x, Sink_neighbors)
            if comp==1 then
                start_Timer(Timer_For_Sink)
            comp=0 if End(Timer_For_Sink then
                k=Sink_neighbors.size();
    else
        if (Receive_Packet_Discovery) then
            Pcst=Packet(Hello)
            Send(Pcst, Broadcast)
            start_Timer(neighborhood_discovery_Timer)
        if (Receive_Hello) then
            Pcst=Packet(Ack_hello)
            Unicast(Pcst, Sender_Packet)
        while (Receive_Ack_Hello) do
            v++
        if (End of neighborhood_discovery_Timer) then
            Pcst=Packet(Response_Discovery)
            Unicast(Pcst, SINK)
```

Algorithm 2: phase of detection of the loss of a node

```
Result: this phase makes it possible to prematurely detect the
        deactivation of a node
Data: Int:k< max ≤ N/2 (N is the number of nodes in WNS) ack,
      max, Pi, Pn, Cid, critical_value
Data: MACAddress : neighbors_List[]
Data: Booleen : HAVE_DEAD_NEIGHBOR = false
begin
    if (Node.(Energie)==critical_value) then
        Pcst=Packet(Alert_Packet,Pn,Cid)
        Unicast(Pcst,neighbors_list[0])
    if (Receive_Alert_Packet) then
        start_Timer(alert_Timer)
    if (End_Timer(alert_Timer) then
        Pcst=Packet(Hello)
        Unicast(Pcst,critical_neighbor)
        if (Receive_Ack) then
            start_Timer(alert_Timer)
        else
            HAVE_DEAD_NEIGHBOR=true
            Pcst=Packet(DEAD_NODE_ALERT)
            Sent(Pcst,neighbor_list[0])      // packet sent to the
             nearest neighbor to the sink
              // the packet contains the information relating
             to the dead node, i.e.  its position and its
             membership cluster (Pn and Cid)
    if (Receive_Packet(DEAD_NODE_ALERT)) then
        Pcst=Packet(DEAD_NODE_ALERT)
        Sent(Pcst,neighbor_list[0]) // the packet will be relayed
         to the sink
```

The new node then classifies all the complex numbers in ascending order, that is, from closest to farthest from the sink. After performing the classification, the node determines the missing element $A_{(missing)}$ by skipping 1 from the list containing the numbers A received. Finally the node is self-defined with the Pn and Cid parameters of $A_{(missing)}$

$$A(node) = i(Pn_{node}) + Cid_{node} \qquad (2)$$

Tab 1. Characteristics of topologies

Topologie	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Facteur k	1	2	3	4	5	6	7	8	9	10
Cardinalité	55	70	85	100	120	150	200	250	300	500
Orphan nodes	0	0	0	0	0	0	0	0	0	0

Algorithm 3: network behavior after adding a new node

```
Result: this phase makes it possible to prematurely detect the
        deactivation of a node
Data: Int:k< max ≤ N/2 (N is the number of nodes in WNS) ack,
      max, Pi, Pn, Cid, critical_value
struct {
    MACAddress source
    int Pn, Cid, Id
} neighbor
Data: neighbor[] : my_neighbors[]
begin
    if (Node_Is_New_Added) then
        Pcst=Packet(NEW_NODE_ADDED)
        Broadcast(Pcst)
        Start_Timer(wait_For_response)
    if (Receive_Packet(NEW_NODE_ADDED) then
        if HAVE_DEAD_NEIGHBOR then
            Pcst=Packet(Cons_For_New,Pi,Pn,Cid)    // Pn and Cid
             are those of the dead node
        else
            Pcst=Packet(Response_For_New,Pi,Pn,Cid)
             // neighbors respond by sending their positions
             and cluster id
    if (Receive_Packet(Response_For_New) then
        neighbor x x.source=Packet_receive(Id)
        x.Pn= Packet_receive(Pn)
        x.Cid= Packet_receive(Cid)
        Add(my_neighbors, x)
    if (End_Timer(wait_For_response) then
        my_neighbors(sort_in_descending_order)
             // here the neighboring nodes are classified in
         decreasing order from the closest to the farthest
         from the sink
        my_neighbors(search_the_missing_element)
        neighbor me= my_neighbors(missing_element)
        Pn= me.Pn
        Cid= me.Cid
        Pcst=Packet(Announce_New_Node,Pn,Cid)      // the node
         announces its parameters to its neighbors so that
         they update their neighborhood database
```

Figure 3. Percentage of construction success for each redundancy factor k

Simulation and Results: In order to test and evaluate our solution, we performed a series of simulations on the Castalia environment which is a wireless sensor network simulator based on the OMNet ++ platform. In order to examine the generality regarding the factor k, we have run the algorithm on several topologies with different cardinality. The algorithm is executed on ten topologies (T1, T2, T3, ……., T10) with respective k-redundant factors ranging from 1 to 10 (k (T1) = 1, k (T2) = 2, ……., k (T10) = 10), the cardinality (number of nodes in the network) is chosen so as to increase each time the factor k increases for the different topologies. The table tab (1) gives us the characteristics relating to each topology. The results of Figure 3 obtained show a complete and perfect construction on all topologies. This highlighting allowed us to verify the generality of our solution. The technique performs an excellent construction on all topologies, 100% of the nodes have been placed in their respective clusters. this allowed us to have zero orphan nodes on all the topologies used

Conclusion and perspectives

In this work we have presented (G2CMLN), a generalized cluster construction mechanism for wireless sensor networks with linear topologies. This work comes in extension to the protocol (2CMJ) which proposed a technique of construction of clusters for linear networks with junction zones. This latter technique had some flaws, in particular at the level of the redundancy factor k which was chosen statically. This fixed choice of the factor k made their solution non-applicable for any network which was based on a k differ from their choice. Our solution was able to solve this problem by presenting a technique that would be applicable for any redundancy k factor. (2CMJ) also did not present a reaction mechanism to the addition of new nodes and to the loss of nodes in the network. These phenomena are very common in a wireless sensor network. (G2CMLN) was able to solve this problem, by including in its algorithm mechanisms for updating the network in the face of losses or the addition of new nodes. The solution has been tested on several topologies with different cardinalities. The results obtained showed a complete construction with a success rate of 100% at the nodes In perspective, we plan to set up a position generation system for network sensors. This system will propose a topology with a k factor which varies and which increases as one approaches the sink which would increase the availability of the network especially for the nodes which are relatively close to the sink.

REFERENCES

1. Fall, A., Sarr, M. D., & Sarr, C. (2021) clusters construction mechanism for junction linear wireless sensor networks (2cmj), Journal of Scientific and Engineering Research, 2021, 8 (8): 15-29
2. Fall, A., Sarr, M. D., & Sarr, C. (2020, March). Clusters Construction Mechanism for Strictly Linear Wireless Sensor Networks. In International Conference on Innovations and Interdisciplinary Solutions for Underserved Areas (pp. 224-237). Springer, Cham. Amin Shahraki, AmirTaherkordi, Øystein Haugenb, Frank Eliassen, Clustering objectives in wireless sensor networks: A survey and research direction analysis. Computer Networks180 (2020) 107376
3. Meng-Shiuan PAN, Hua-Wei FANG, Yung-Chih LIU et Yu-Chee TSENG. Address Assignment and Routing Schemes for ZigBee-Based Long-Thin Wireless Sensor Networks“. In:IEEE Vehicular Technology Conference ’08, Spring 2008. 2008, p. 173–177 (cf. p. 1, 3–5, 35, 40, 56, 66, 67).
4. Marco ZIMMERLING, Waltenegus DARGIE and Johnathan M. REASON. Localized power-aware routing in linear wireless sensor networks“. In: Proceedings of the 2nd ACM international conference on Context-awareness for self-managing systems. CASEMANS ’08. New York, NY, USA: ACM, 2008, 24–33 (cf. p. 1, 3, 35).
5. Imad JAWHAR, Nader MOHAMED and Dharma P. AGRAWAL. „Linear wireless sensor networks: Classification and applications“. In: Journal of Network and Computer Applications 34.5 (2011), p. 1671–1682 (cf.p. 3, 5, 35, 38, 39, 59, 70).
6. Ian F. AKYILDIZ, Weilian SU, Yogesh SANKARASUBRAMANIAM et Erdal CAYIRCI. A survey on sensor networks“. In: IEEE Communications magazine40.8 (2002), p. 102–114 (cf. p. 1)
7. Konrad LORINCZ, Matt WEL SH, Omar MARCILLO et al. Deploying a wireless sensor network on an active volcano“. In: IEEE Internet Computing. 2006, p. 18–25 (cf. p. 1, 11).
8. Imad JAWHAR, Nader MOHAMED et Khaled SHUAIB. „A framework for pipeline infrastructure monitoring using wireless sensor networks“. In: Wireless Telecommunications Symposium, 2007. WTS 2007. IEEE, 2007, 1–7 (cf. p. 1, 12).
9. Sukun KIM, Shamim PAKZAD, David CULLER et al. „Health monitoring of civil infrastructures using wireless sensor networks“. In: Proceedings of the 6th international conference on Information processing in sensor networks. ACM Press, 2007, p. 254–263 (cf. p. 1, 3, 12).
10. BAKER, C. R. K. ARMIJO, S. BELKA et al. „Wireless sensor networks for home health care“. In:Advanced Information Networking and Applications Workshops, 2007, AINAW’07. 21st International Conference on. T. 2. 2007, 832–837 (cf. p. 1, 10)
11. NDOYE, E. H. M. F.JACQU, M. MISSON et I. NIANG.„Using a token approach for the MAC layer of linear sensor networks: Impact of thenode position on the packet delivery“. In:2014 IFIP Wireless Days(WD). 2014, p. 1–4 (cf. p. 38, 70).

****