A Comprehensive Review of Self-stabilization Algorithm and its Applications in Wireless Networks

Zargham Heydari[1*], Hamed Gorginpour[2] , Mahdi Shahparasti[3]
1- Department of Electrical Engineering, Bushehr Branch, Islamic Azad University , Bushehr, Iran.
Email: Nariman_azma@yahoo.com (Corresponding author)
2- Assistant Professor Department of Electrical Engineering Persian Gulf University, Bushehr, Iran
Email: Hamedgorginpoor@gmail.com
3- Assistant Professor, Islamic Azad University, East Tehran Branch, Tehran ,Iran.
Email: mshahparasti@yahoo.com

Received: July 2021 Revised: August 2021 Accepted: October 2021

ABSTRACT:
Distributed computing is a field of the vast computer science that deals with distributed systems. These systems have a significant role in computing with high efficiency. One of the important cases with a great role in distributed systems is the self-stabilizing concept. One of the new algorithms with a critical role in engineering and computer science is self-stabilization algorithm. This algorithm is known as a lightweight and convenient property relative to other classic solutions and methods of fault tolerance in obtaining fault tolerance (FT). Moreover, in terms of time and space, the art of this algorithm is that it needs less time and space. These features have made the self-stabilization algorithm highly promising for use in distributed systems that are equipped with low computing and low memory processes. Wireless sensor networks (WSNs) include many sensor nodes that can receive, collect, process and transfer data. These networks are widely used in industrial, military, and civilian uses like industrial facility management, power / engine sources monitoring, target routing, care, healthcare management, and geographic information analysis, and so on. The paper, comprehensively, discussed this algorithm and its uses in wireless networks.

KEYWORDS: Self-Stabilization Algorithm, FT, Convergence, Wireless Networks, Self-Stabilizing Time, Distributed Systems.

1. INTRODUCTION

The concept of “self-stabilizing” was first introduced in 1973 by Dijkstra [1] in the field of distributed systems. Distributed systems refer to the systems that include a finite set of independent processes connected through a network and whose task is to realize a common goal. In these systems, designing self- stable (distributed) algorithms may seem a little complicated as each computing unit (i.e., each process), while being only a part of the system, should coordinate with other processes. In these conditions, the local state and the process information transmitted, usually asynchronous, are connected to the parts of other processes by communication media. However, it should be noted that there are distributed self-stabilization algorithms that are even simpler than non-self-stabilization algorithms [2].

One of the main uses of self-stabilizing is in designing distributed systems that can withstand any (finite) number of transient Faults (TFs). TFs happen unpredictably but do not cause serious hardware damages. Moreover, the frequency of incidence of these faults is small compared to the intermittent ones. Thus, network components (processes or connections) are temporarily affected by transient faults. This effect can be seen in some properties of network components like the number of bits in the local memory of a process; e.g., some messages are lost in a connection, or the order of processes is displaced. Therefore, a TF affects the state of the part on which it occurs.

Thus, after a finite number of TF occurs, the configuration of a distributed system can be random, which means that variables in the process memory may assume random values (in their scope of definition). Communication connections may contain a finite number of random messages whose formatting is completely correct. The time interval before the next transient perturbation is enough to let the system resume a legitimate behavior. This is shown in Fig. 1.

Paper type: Research paper
DOI: https://doi.org/10.52547/mjtd.10.4.147
How to cite this paper: Z. Heydari, H. Gorginpour and M. Shahparasti, “**A Comprehensive Review of Self-stabilization Algorithm and its Applications in Wireless Networks**”, Majlesi Journal of Telecommunication Devices, Vol. 10, No. 4, pp. 147-158, 2021.

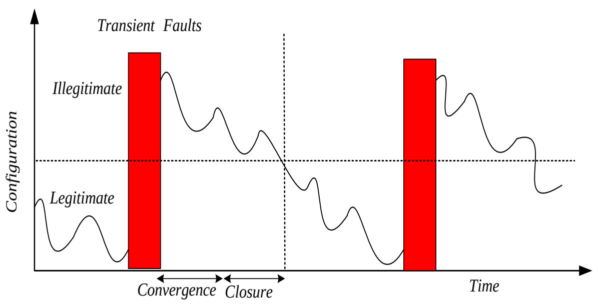

Fig. 1. Tolerance of the systems with self-stability.

Compared to many FT approaches (robust approaches), self-stabilizing is a non-hidden method: it does not hide the effect of faults and allows the system to (temporarily) deviate from its properties. . Hence, faults are not directly dealt with and convergence is guaranteed only when (without any faults) the time window size is large enough. Hence, in "proving accuracy," the starting point of observations is after the occurrence of "last" faults. Thus, the system state starts from a random configuration, as it is affected by transient faults after the transient fault occurs, whereas it is assumed "no fault has occurred" (Fig 2).

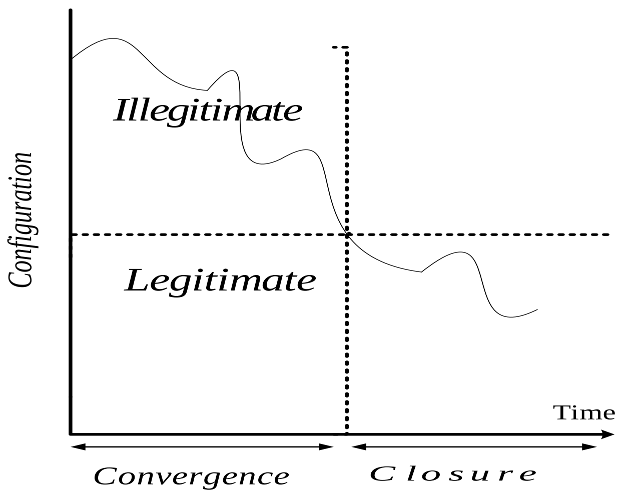

Fig. 2. A fault in a model.

In the papers published, such a configuration is called the "initial configuration". Then, by starting from an arbitrary initial configuration, an algorithm will be self-stabilizing if it ensures that the system automatically (without the intervention of external factors such as humans) converges to a closed set called "optimal configurations" after a finite time passed. Note that this definition indirectly assumes that faults do not change the code of the algorithm. This hypothesis is justified in [3] by two arguments.

The overhead of the self-stabilization algorithms is seen compared to intolerant (fault) algorithms based on the runtime, memory required and the information exchanged. It should be noted that for each problem, the overhead of self-stabilization algorithms can intermittently be ignored. Indeed, the self-stabilizing property is generally regarded as a light-weight method for FT compared to classical robust methods [4]. This is because self-stability assumes an "optimal" approach, whereas robust classical algorithms use "pessimistic" approaches [5].

Proper and consistent initial configuration in distributed systems is a complex and critical synchronization operation. For instance, it is difficult to create a proper configuration in a high-volume and large-scale network (like the Internet) that involves many processes. Hence, using self-stabilization algorithms for such systems is highly desirable, as the self-stabilizing property does not need initialization. Moreover, studies have shown that self-stability is a good approach for the evolution of the networks, such as networks where there are no communication channels to connect entities [6]. Although self-stabilization algorithms are usually designed for static topologies, the algorithms specific to arbitrary topologies can tolerate some topological changes (such as adding or removing communication links or nodes). More precisely, if topological changes occur locally in the processes involved and if the number of iteration of these events is sufficiently small, then these changes can be considered as Transient faults.

These two advantages make the self-stability property of "autonomic computing" as suitable. The concept of "autonomy" was first introduced by IBM Co. for the first time in 2001 to describe computer systems which are called "self-management" [8, 7]. The nature of autonomic computing (AC) encompasses a broad set of concepts related to "self-ability," including self-organization, self-therapy, self-configuration, self-management or self-optimization. Indeed, AC involves all the methods that allow a distributed system to make unexpected changes. However, the intrinsic complexities of operators and users remain hidden. Thus, self-stability can be defined as an alternative for designing autonomous systems.

As self-stabilization algorithms never end in dead end despite the random configurations of systems, one can easily combine them with each other [9]. For instance, consider A and B self-stabilization algorithms, where B uses the output of the algorithm A as input. A and B can be run in parallel, as A will eventually produce a correct input for B, and thus, the convergence of B algorithm is guaranteed. All the stated benefits make self-stabilization algorithms suitable for real systems. The applications of self-stability have been used in real networks as well [10]. Moreover, classic routing protocols have made self-stabilization algorithms a standard for internet-based routing protocols [11].

In this paper, after the introduction, Part II deals with the concept of self-stabilization algorithm. The third section provides a general overview of the applications of this algorithm. The fourth part deals with the application of this algorithm in wireless networks, and in the last part, the conclusion has been presented.

2. FORMAL DEFINITION OF ALGORITHM [12-15]

Various definitions of self-stabilizing have been presented in different papers and studies. However, two definitions presented by Dolev and Ghosh have been used more commonly. The formal definition of self-stability is expressed using the concept of “S” system, and “P” proposition. S system tends to execute P correctly. We call S system self-stabilizing if it has two conditions of closure and convergence. The concepts of closure and convergence are as follows.

Convergence: If you start from an arbitrary local state, the system is stable if it is guaranteed to reach a desired general state after passing several modes.

Closure: Whenever the proposition P is verified once in the S system, the proposition will no longer be distorted and will not lose its validity.

Beside the above definition, a more complete definition of this algorithm can be given below. Any self-stabilizing system by this definition is also self-stabilizing by the definitions of Dolev and Ghosh. The following assumptions are considered to express this definition of the concept of self-stability.

A: G Distributed Algorithm: network connected graph

D: Destructive factor (daemon)

SP: The problem to be solved under safe and dynamic conditions

$C/L(L \subseteq C)$: Desirable and acceptable series of configurations

A distributed algorithm for solving a problem in a connected graph with a destructive factor (Daemon) is self-stabilizing if there is an infinite set of optimal configurations such that the following three conditions are met under closure, convergence, and accuracy.

A) Closure: L is closed with A in G with D if the following condition is met.

$$(\forall \boldsymbol{\gamma} \in \boldsymbol{L} \ , \forall \boldsymbol{\gamma}' \in \boldsymbol{C} \quad \boldsymbol{if\ \gamma \to \gamma'\ then\ \gamma'} \in \boldsymbol{L} \ ) \qquad (1)$$

B) Convergence

A under D converges to L in G if

$$(\ \forall \ \boldsymbol{e} \in \boldsymbol{\varepsilon} \ \ , \exists \ \boldsymbol{\gamma} \in \boldsymbol{e}\ \mathbf{as} \ \ \boldsymbol{\gamma} \in \boldsymbol{L} \ \ ) \qquad (2)$$

C) Accuracy

If the following conditions are met, SP under D has an acceptable and favorable L state.

$$\forall \ e \in \varepsilon\ (L),\ SP\ (e) \qquad \text{constant} \qquad (3)$$

ε (L) is a subset of starting from the running of ε that starts from structure L

3. DIVIDING THE USES OF THE SELF-STABILIZATION ALGORITHM

Previous sections presented the definitions and features of self-stabilization algorithm. This paper has dealt with the uses and study branches of this algorithm. This algorithm can be classified into four aspects.

A) Synchronization level

Self-stabilizing solutions can be divided into “synchronization level” (consecutive - such as [1,21], synchronized - such as [2] or fully distributed - such as [16]) classified.

B) Level of cognition

Accordingly, self-stabilization algorithm is classified as “anonymous / unnamed” (completely anonymous / unnamed - such as [22], rooted - as [16] or as known [23]).

C) Computational model

Perhaps the most significant criterion is the “computational model” based on which the algorithm is written. Three major models have so far been extensively studied. The three models from the weakest to the strongest are, respectively: “message transmission model” (classic) [24], “registry model” (also called the local shared memory model with read / write atomicity) (3), and “atomic status model” (also called atomic shared memory model) [21, 1]. The last two are, in fact, abstracts from the message transfer model, where “message exchange between neighbors” is replaced by “direct access to the status of the neighbors.” The difference between the registry model and the atomic-status model is in their atomicity. In the atomic-status model, each atomic step contains at least one (or perhaps several) processes that read and update its status and that of all its neighbors. Atomicity is weaker in the registry model. Each process maintains some communication registers and shares them with its neighbors and (perhaps) some internal (non-shared) variables. The atomic step involves a process for performing internal computing, where the reading or writing operation is performed on a communication register.

D) Problem solving

After preliminary studies by Dijkstra [1,21] there are many algorithms for solving various problems like “spanning tree structures” [25], “sign / symbol circulation” [26], “alliance”, [27] and “Information dissemination with Feedback” [27]. These problems can be grouped into the following categories:

1. Calculating the distributed structures (also called “self-organization”) including tree spanning and clustering
2. Routing algorithms [28-30]

3. Wave algorithms including sign / symbol transfer [32, 31] and information dissemination with feedback [27]
4. Synchronization including unity and phase synchronization
5. Problems related to resource allocation, such as "mutual monopoly" and dining philosophers

In these studies, various topologies have been examined including: complete graphs [35], loops [36, 37], directional or non-directional trees [38-40], planar graphs [[42,41] and arbitrary connected graphs [44 43,].

4. APPLICATION OF SELF-STABILIZATION ALGORITHM IN WIRELESS SENSOR NETWORKS (WSN).

Wireless Sensor Networks (WSN) is composed of a large number of distributed Ad Hoc sensors. Sensors are very small electronic devices used to collect data from their surroundings and are actually considered as converters [45]. They are equipped with a communication unit (usually a radio transmitter / receiver) that allows them to set up an Ad Hoc network and connect to the central station via a gateway (GW) or sink. WSNs contain many sensor nodes (SNs) capable of receiving, collecting, processing and transmitting data. SNs collect data in the target environment and transmit it to the base station (BS) using wireless transmission methods [46,47]. Unfortunately, these devices with limited resources and low cost are very prone to failure. These sensors communicate with each other via wireless technology, and organize themselves in a multi-hop wireless network. These networks are used in devices like environmental monitoring, smart spaces, medical systems, or robotic exploration.

In spite of all the disadvantages, WSNs have been so welcomed in many areas such as security, defense, research, industry, agriculture or environmental monitoring [12-8]. WSNs are widely used in industrial, military and civilian uses like industrial facility management, power / engine source monitoring, target routing, care, health system management, and geographic information analysis, and so on [48-51].

4.1. Connection control in wireless network

Connection control is a bifuzzy technique commonly used to extend the life of dynamic networks. The production phase of this technique is responsible for improving the current network topology and protecting some network features (node connectivity, directionality, and so on). On the other hand, the purpose of the maintenance phase is to allow the network to react autonomously at breakdowns of nodes and / or junctions, and to reconstruct desirable topological features [52].

A network connection has been considered to address the problem. The assumed network is composed of a large number of nodes where the control is intended to connect the nodes. The following hypotheses have been considered to control the connection.

- The nodes are equipped with a multi-directional radio and have a unique ID.
- The nodes are aware of their approximate position and their neighbors and can estimate the quality of the connection.

The number of packets being retransferred is high, and the connections are weak.

Each node u is able to estimate the transition delay τ (u, v) and the round time rrt (v) of the connections leading to neighbor v, and the displacement of the nodes is only due to the instability of the monitored area.

Imagine V and E show the nodes and the connections between them in area A, respectively. Each connection (u, v) ∈ E shows the ability of the node u to communicate with the node v with the transmission range of rc (u) ∈ Rc. This capability of the connection depends on d (u, v) and lq (u, v), respectively, which show the length (Euclidean distance) and quality of connection (u, v). Rc is also a discrete set of transmission domains. In mathematical terms:

$$E = \{(u, v) \in V: (d(u, v) \leq r_c(u)) \wedge (lq(u, v) \geq \ell)\} \qquad (4)$$

In this equation, ℓ is a limit to the quality of the connection d (u, v) and lq (u, v) are calculated; respectively.

$\hat{\theta}$ shows the maximum number of legitimate transfer attempts determined by the main program. $\varphi_{uv}(t)$ shows the number of failed attempts that node u experienced at time t after transmitting a packet to neighbor v.

At any moment t:

$$[(u, v) \in V] \Leftrightarrow \{[(\theta^{\wedge} - \varphi_{uv}(t)] > 0\} \qquad (5)$$

Thus, the network used creates a spatial directed graph in A, which is shown by G = (V, E).

4.2. Maintaining connections

This process involves identifying topology changes, fault evaluation, and connection reconstruction. In other words, the purpose of the process is to allow each node to react to local topological changes (inputs, outputs, connection quality changes and so on).

- Node and connection failure detection

Broken connections are identified by the development of connected nodes, whereas the broken nodes are identified by their neighbors.

- Reconstructing the connection

A failure in connections can have significant effects on current topology (such as leaf node death, network segmentation, and so on). Hence, when a node cannot communicate with its direct supervisor, it has to find its "useful neighbor" (the most trusted neighbor) to reconstruct its connection.

It has to be noted that during this process, the node tries to join its "useful neighbor" cluster. If all attempts fail, the node must introduce itself as a cluster head (CH) and try to create a new cluster in k-mutation neighborhood.

4.3 Average convergence time

Several networks consisting of a static and population sink of randomly distributed sensor nodes have been implemented to estimate the average estimation time. The implementation parameters have been presented in Table 1. Table 2 also shows the transition status delay.

Table 1: General simulation parameters

Parameter	**Value**
Deployment zone	1000 m × 1000 m
Number of sensors	100–1000
Sink position	(450; 200)
Sensor transmission ranges	{15; 35; 54; 70; 83; 98; 117; 127} m
Sink transmission range	250m
Primary energy of sensors	0.2 J
Self discharge rate in seconds	0.1 µJ
E_{elec}	50 nJ/bit
e_{fs}	10 nJ/bit/m^2
e_{amp}	0.0013 nJ/bit/m^4
d_0	87 m
Message length	2000 bits
CH service time	5 s
nitrmax	200

Table 2: Delay in transfer status

	R_x (µs)	**T_x (µs)**	**Sleep (µs)**	**Idle**
R_x	_	1	194	_
T_x	1	_	194	_
Sleep	5	5	_	_
Idle	_	_	_	_

Weibull and uniform distributions are used for fault injection as presented in Table 3 to make real changes in mean time between failures (MTBF) parameter. Moreover, when it was necessary to change the injected fault of the connection quality, for randomly changing the parameters such as packet reception ratio (PRR), signal-to-noise ratio (SINR), and link quality index (LQI) [53,54] a uniform distribution has been used (Table 4). Table 4presents the qualitative parameters of the connection.

Table 3: Fault injection parameters

Factor	**-**	**+**
Scale	U(1,5)	U(6,10)
MTBF	W[α = W(100, 500), β = 3]	W[α = W(600, 1000), β = 3]
Localization	1	U(2,10)
Network size	U(100, 500)	U(600, 1000)
k	1	U(2, 4)
Type	0 (for the connection)	1 (for the node)
Node level	U(0, 2)	3

Table 4: Link quality parameters

Quality	**PRR**	**SINR, dBm**	**LQI**
Excellent	1	]30:40]	]106:255]
Good	]0.75:1[	]15:30]	]102:106]
Medium	]0.35:075]	]5:15]	]80:102]
Poor	[0:0.35]	[0:5]	[0:80]

Table 5 also compares the analytical functions of the protocols.

Table 5: Comparison of analytical functions.

Protocol	**Method**	**Time**	**Message**	**Space**
CONSTRUCT	Hybrid (k-mutation)	O(n)	O(n)	O(n)
FTS	Hybrid (1-mutation)	O(n)	O(n)	O(n)
SDEAC	Non-hybrid (k-mutation)	O(n)	O(n)	O(n)

Convergence time was measured as the nodes became dependent as soon as the first topology was created. Faults were only injected at nodes where their status was legitimate. During the simulation process, whenever a fault led at least one node to enter the illegitimate state, the timing of its occurrence was recorded and then the location of the affected neighbors was determined (Table 6).

Table 6: Transmission status energy consumption

	R_x (mw)	**T_x (mw)**	**Sleep (mw)**	**Idle(mw**
R_x	_	62	62	_
T_x	62	_	62	_
Sleep	1.4	1.4	_	1.4
Idle	_	_	1.4	_

The node retrieval time was calculated as soon as these nodes returned to their legitimate state. The mean of all the recovery times was calculated after the death of the last node.

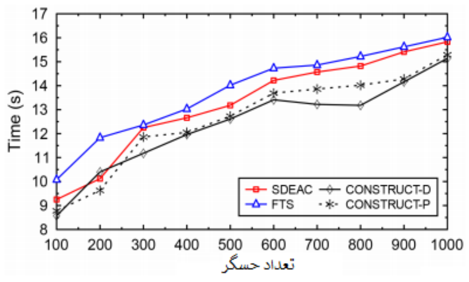

Fig.6. Mean conv The number of sensors rk size

4.4. Energy efficiency

The purpose of these tests is to examine each protocol capability in creating delay in battery finishing point and extending network life. In doing so, we went on with the simulations until the first node death (FND), the first sink neighbor death (FSND), all sink neighbors death (ASNND), and the last node death (without fault injection). We evaluated how the network size affects on its lifespan. Each experiment was iterated 100 times. The results have a confidence interval of 95%.

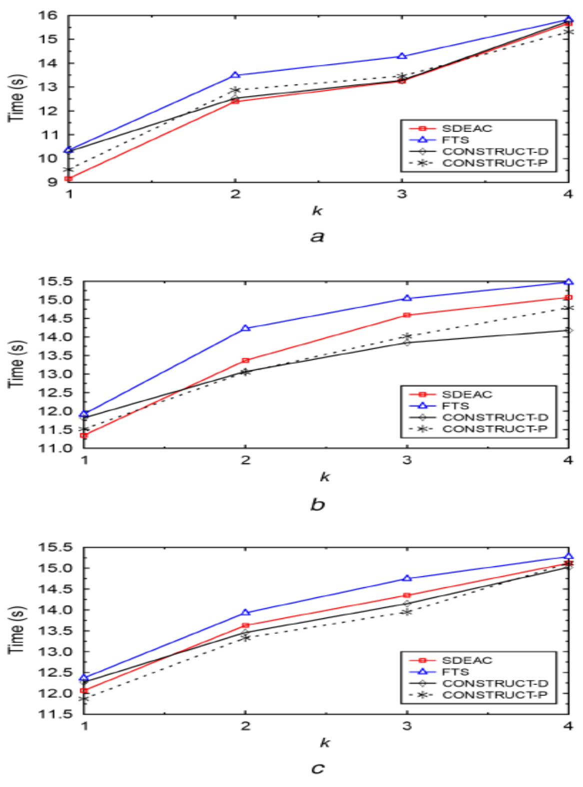

Fig.7. The effect of parameter k on mean convergence time (a), network size n = 100 (b), Network size n = 500 (c), and Network size n = 1000.

4.5. Energy storage in wireless network

Wireless sensor networks life basically depends on the energy storage efficiency. This section has introduced an efficient self-stabilizing topology-control protocol for WSN. Network connection is established along the maximum energy storage in reduction in the transmitting power of each node. The sensors are operated by a battery and through a processor or radio. It is usually hard to recharge or replace sensor batteries. Hence, energy is a rare and great resource in WSN and has to be stored. Energy efficiency is a significant feature in designing these networks. Energy is used due to message processing, transmission and reception in WSN. Moreover, a great part of the energy is lost due to idle listening, eavesdropping, and collisions.

Idle listening is the state where a node awaits a message. The node has to set its radio mode to reception whenever it is not transmitting the message as the node does not know when the message will arrive. This means that a node must listen to all messages from the neighboring nodes. Since most of the times the nodes are in this state, a great part of energy is wasted. Thus, node eavesdropping happens for many packets. This means wasting a great part of energy, particularly when the node congestion is high and the traffic load is heavy. Collisions are another part of the shared nature of wireless media that increases energy loss. Whenever a collision happens, the sensor node has to send its message again, ending in more reception and consequently more power consumption.

The many solutions proposed to extend the network lifetime can be divided into three classes [55].

A) Energy-efficient routes: These solutions intend to minimize the energy consumed due to sending successive messages. Energy-efficient routing minimizes energy consumption by reducing the messages sent and received.

B) Sleep mode: These methods are based on node activity scheduling and switch node states between sleep and wake. This class of methods try to minimize power consumption by ensuring network connectivity and application performance. Hence, this class of methods reduces energy consumption in idle listening.

C) Topology control: In this class of methods, the sensors reduce their transmission range yet the network connection remains established. The purpose of these solutions is to minimize the power consumption due to transmission power, eavesdropping and collision.

A section based on third class and self-stabilization algorithm will be introduced. Self-stabilization is one of the most desirable features of WSNs. Indeed, regardless of the initial state, self-stabilization algorithm reaches proper behavior after a limited number of steps. Then the algorithm is restored without external intervention after any unexpected disturbance. This allows this method to cover classic events in WSN.

4.5.1. The intended model

The model where energy storage is to be examined is a wireless sensor network consisting of n sensors. Each sensor node uses a multi-directional cover, so that the message of a sensor is received by all the sensors in a circle centered on that sensor. This circle is called the “domain of transmission”.

Wireless sensors form a graph G = (V, E), where V means the set of nodes (sensors) and E shows the set of edges that show the relationship between the nodes. Whenever maximum Euclidean distance of two nodes is equal to Rmax, there will be an edge between them. The neighbors of a node u are shown by N (u). It is assumed that graph G = (V, E) is a unique connected graph, and all wireless sensors have distinct identifiers, and can calculate their energy use.

Self-stabilizing system can reach the desired behavior without any external interference and within a limited time, after any temporary failure.

4.5.2. Self-stabilizing energy storage algorithm

The distributed self-stabilization algorithm reduces the nodes transmission power and creates a specific topology to ease routing. Thus, a pervasive tree is first made such that the route between the node u to the root r in the tree equals to the least weighted path from u to r in the graph. Each node stores the weight of its distance from the root. Secondly, this weight is used to construct a Minimum Weight Connected Dominating Sets (MWCDS) that shows the backbone. Ultimately, depending on whether it belongs to the backbone or not, each node determines the scope of its transmission. Any sensors not located in the backbone are called recessive nodes. This sensor selects its nearest neighbor in the backbone (according to the Euclidean distance) as its dominant node. Then it reduces its transmission range to the Euclidean distance between itself and its dominant node. To reach all of its neighbors in the backbone and all of its recessive nodes, the sensor in the backbone reduces its transmission range.

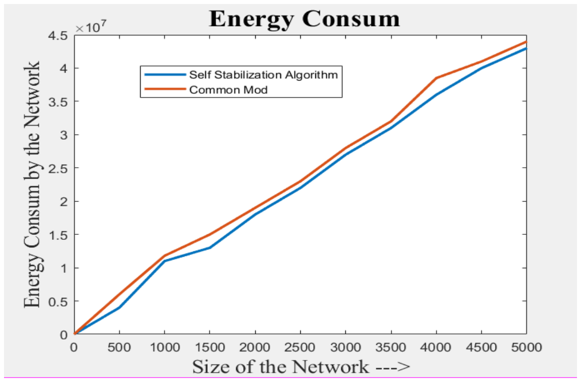

Fig 8: Comparison of energy consumption between self-stabilization algorithm and the conventional model.

Thus, all communication links between recessive and dominant nodes as well as all communication links in the backbone are bidirectional.

The proposed algorithm is self-stabilizing and relies on the energy used in the sensors. As this energy changes over time, there is a danger of ping-pong effects: MWCDS can constantly alter, and thus the algorithm never becomes stable. Instead of using the current energy consumed, a variable that saves the energy consumed in building MWCDS is used to avoid this problem.

The algorithm is divided into six main parts, and each part is introduced as a self-stabilization algorithm. In any algorithm, the sensors can read the variables of the previous algorithms and can read and write the variables of the current algorithm. This assures that if a part is self-stabilizing then the whole algorithm will be self-stabilizing.

These six parts are as follows:

1. Updating the energy used in the sensors: This is done in the light of sending messages to the leader node.
2. Updating the mode: each node sends its local state to all its neighbors
3. Making a spanning tree based on sensor energy
4. Building MWDS: This algorithm selects sensors with minimum local weight
5. Building MWCDS: This algorithm selects those sensors that act as connectors between the sensors in MWDS.

5. REDUCING COMPUTATIONAL DOMAIN. Simulation results

In this section, the simulation is presented to evaluate the performance of the distributed self-stabilization algorithm. In each experiment, 9 various networks with 500 to 5000 size and 500 sensor growth were examined. Moreover, 100 simulations were done, and confidence intervals were calculated at 95% level for each size. Sensor nodes were randomly distributed in 200 by 200 m squares. The maximum transmission range of each node is 25 meters, and the battery energy is random value from 1 to 100%. The simulation results are given in Figs 1 and 2. The purpose is to reduce energy consumption. The average energy consumed in the network is calculated by using the backbone for message routing to show algorithm efficiency. The energy consumed in the network in the optimal routing mode (using the shortest path) is calculated without reducing the transmission range. The energy consumption function is based on the energy consumption model above and the data obtained from the simulations. The average energy consumed in the network to communicate using the shortest path equals the sum of the energy consumed in sending and receiving.

This Fig shows that the self-stabilization algorithm is better than optimal routing with maximum range.

Fig 2 shows the energy percentage of backbone sensors based on the number of network sensors. Moreover, it indicates that the self-stabilization algorithm is efficient and the greater the network size, the better the algorithm will be.

Using the proposed self-stabilization algorithm will reduce the energy consumed in communication. Moreover, only the sensors with the highest battery level will be responsible for routing the message.

.

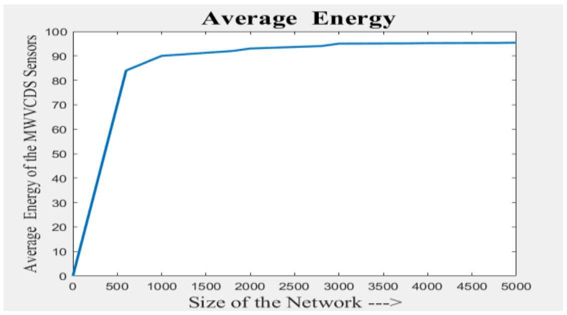

Fig. 9. Average energy of dominant sensors.

5.1. Energy efficient routing using self-stabilization algorithm

The delays resulting from wireless multi-hop communications usually impede time-sensitive applications in WSNs. Thus, these delays have to be controlled to optimize the situation, which is one of the optimal routing methods. Delay in sending data packets can be measured as the number of hops of a sensor to the main station and the acceptable delay of each packet shows the initial value of the lifetime label on the service quality criterion. Whenever a lifetime tag is sent by a node, its value reduces. According to time-division multiple access (TDMA), one can use a self-stabilizing hop-constrained energy-efficient (SHE) protocol to establish energy networks and minimum-hop for routing service. In this method, multi-hop ad hoc routes are first created in a cluster and the number of nodes in the cluster is controlled at the same time to perform acceptable delays of data packets from the member nodes in the cluster.

An SHE protocol is presented in the following that uses acceptable delay to send packets to the network layer in terms of the number of hops [56]. This concept shows adaptation and is a good criterion for various uses of WSNs architectures.

In a large-scale wireless network, a cluster-based routing protocol reduces energy consumption by compressing data in the cluster. Thus, it increases network lifetime and reduces network congestion. Moreover, clustering reduces channel content and increases the network efficiency under heavy loads. Sending each data packet is considered as sending with a predetermined acceptable delay. The purpose is to reach a clustering method and a routing protocol to execute the hop constraints of each data packet received at the central station and to develop network and convergence lifespan.

The energy of sending radio waves is so significant for energy efficiency in WSNs, as the energy increases secondarily by the transmission range.

5.2. The proposed method

Fig 10 shows the framework of the proposed method that includes the offline and online steps and selects multi-hop clustering methods, CHs, and CHs supporters to create a cluster. Routing data packets in the cluster is accomplished by inter-cluster sending. In the offline step, the algorithms determine the location and size of each cluster with multi-sectional messages. The algorithms leave significant traces for making clusters with maximum hop diameter. The cluster information stage is executed only once in network setup stage, which greatly reduces the overhead of the cluster reconfiguration that is the time and power. Multi-hop routing in each cluster reduces the number of clusters in the network. To implement the hop limitations of each message, the intra-cluster routing protocols include clustering and intra-cluster routing that consider the lifetime of the compressed data packets. By applying distributed self-stabilization algorithm, it creates paths with minimum hop number between the CHs and the central station.

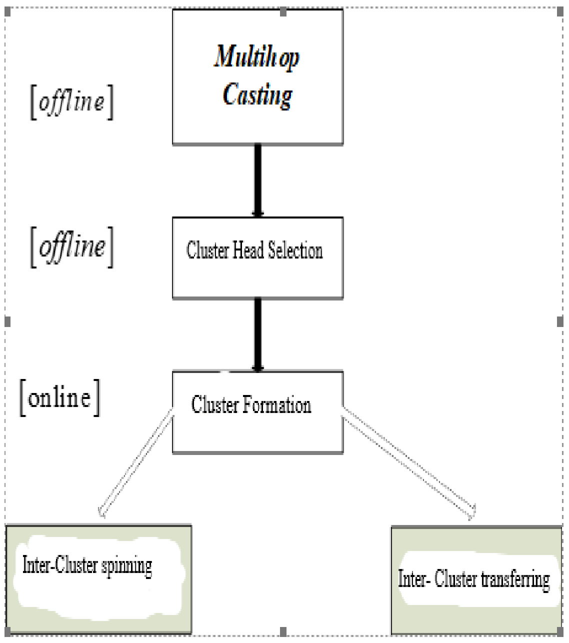

Fig.10. Framework of the proposed method.

In Multi Hop Casting, each sensor node releases the exploratory message with an initial hop counter with a lifespan equal to the lower limit of the counter to create a multi-hop cluster smaller than twice that diameter. Moreover, each sensor node cluster forwards messages to its neighbors and reduces the lifetime of the tag by one unit. In this approach, valid explorer messages are registered and those with zero lifespan or with the same source are eliminated. However, discoverer messages go through a long distance. After all, the discoverer messages have been deleted from the network, each cluster node summarizes the discoverer messages. The header selection algorithm adjusts the location and shape of the clusters by merging smaller clusters. At the end of the clustering time, the algorithm determines the CHs and their backup.

Figs 11 to 14 show the simulation results.[56] Fig 11a compares the network lifetime in the proposed method with LEACH, M-HEED, and PEGASIS [15, 23, 25]. According to the Fig, one can see that SHE performs better than previous approaches considering the first SN completed. Fig 11 shows that SHE provides a longer network lifespan compared to previous approaches.

Fig 12 compares energy loss in the proposed method compared to energy loss in the previous methods. Fig 13 shows the percentages of packets lost by the protocol constraint depending on the number of variable sensor nodes in the network. Closed hop counter simulates the number of sensor nodes and the CHs that packets move through. In other words, it counts the packet from the sensor node source to the central station.

Fig 14 shows that the number of CHs needed is much lower than the previous methods.

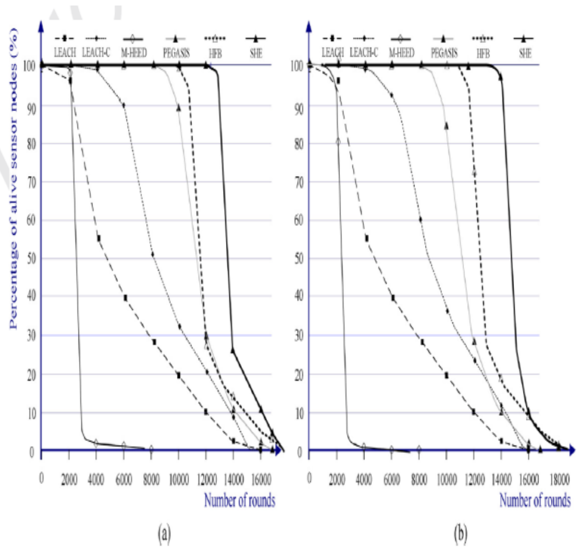

Fig. 11. Network lifetime of various protocols in (a) 50 and (b) 200 SN

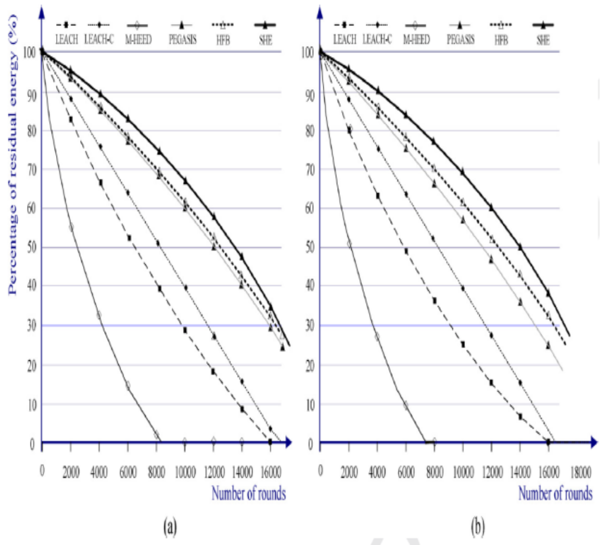

Fig.12. Residual energy of different protocols in (a) 50 and (b) 200 SN.

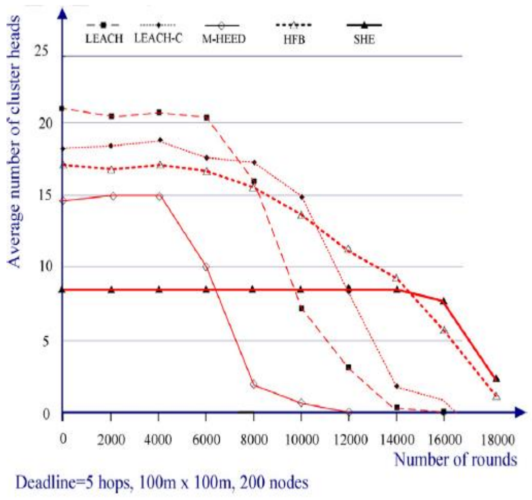

Fig.13. The number of required CHs.

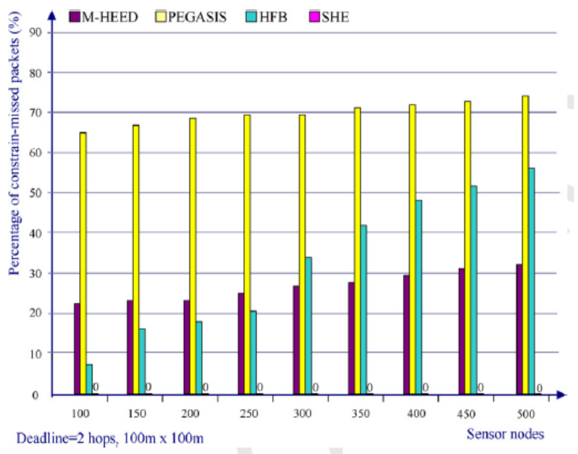

Fig.14. Constrain missed packets in various number of SNs.

6. CONCLUSION

One of the new algorithms with a significant role in engineering and computing sciences is the self-stabilization algorithm. This algorithm is known for achieving FT as a lightweight and convenient property relative to other classic FT solutions and approaches. The paper presented the structure and the concepts of this algorithm and its uses and examined its application in a wireless network. Proper and consistent initial configuration in distributed systems is a complex and critical synchronization operation. Using self-stabilization algorithms is desirable for large systems, as the self-stabilizing property does not require initialization type. Moreover, self-stability is considered a good approach for the evolution of networks, including networks where there are no communication channels. Self-stabilization algorithms can bear some topological changes. These two advantages make the self-stabilizing property suitable for "automatic computation." This algorithm can be classified in terms of aspects of synchronization level, cognition level, computational model and problem solving. To establish an efficient communication topology in a wireless network, one can use self-stabilization algorithm to maintain the connection of active nodes at the time of the fault. The algorithm minimizes the convergence time and reduces signaling and message overhead and energy consumption. In message transmission using self-stabilization algorithm, the energy consumed reduces in establishing communication. Moreover, only the sensors with the highest battery level will be responsible for routing the message. The paper presented a self-stabilizing energy-efficient clustering and routing protocol as one of the uses of self-stabilization algorithm. The simulation results show that this algorithm provides less energy loss and longer lifespan compared to the previous methods.

REFERENCES

[1] Edsger W. Dijkstra., **"Self-stabilization in spite of distributed control."** , Technical report EWD 391, University of Texas, 1973. Published in 1982 as Selected Writings on Computing: A Personal Perspective, Springer-Verlag, OPT.

[2] *Gerard Tel,* **"Introduction to Distributed Algorithms"** , *2nd ed., Cambridge University* Press, 2001.

[3] Shlomi Dolev, **"Self-Stabilization"** , MIT Press, 2000.

[4] Sébastien Tixeuil, **"Toward Self-Stabilizing Large-Scale Systems"**, Habilitation a diriger des recherches, Université Paris Sud - Paris XI, 2006. 4

[5] Chi-Hung Tzeng, Jehn-Ruey Jiang, and Shing-Tsaan Huang, **"Size-independent self-stabilizing asynchronous phase synchronization in general graphs"**, Journal of Information Science and Engineering, 26(4):1307–1322, 2010

[6] Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry, **"Computing shortest, fastest, and foremost journeys in dynamic networks",** International Journal of Foundations of Computer Science, 14(2):267–285, 2003

[7] Jeffrey O. Kephart and David M. Chess, **"The vision of autonomic computing"** ,Computer, 36(1):41-50, 2003

[8] Markus C. Huebscher and Julie A. McCann. **"A survey of autonomic computing:Degrees, models, and applications."** , ACM Computing Surveys, 40(3):1–28, 2008.

[9] Ted Richard Herman, **" Adaptivity through distributed convergence"** , Ph.D. thesis,University of Texas at Austin, 1992.

[10] Ajoy K. Datta, Eugene Outley, Visalakshi Thiagarajan, and Mitchell Flatebo**," Stabilization of the x.25 connection management protocol"** , International Conference on Computing and Information (ICCI), pages 1637-1654, 1994

[11] Yu Chen, Ajoy K. Datta, and Sébastien Tixeuil, **"Stabilizing inter-domain routing in the internet",** Journal of High Speed Networks, 14(1):21-37, 2005

[12] Shlomi Dolev, Mohamed G. Gouda, and Marco Schneider., **"Memory requirements for silent stabilization."**, Acta Informatica, 36(6):447--462, 1999

[13] *Sukumar Ghosh.,* ***"Distributed Systems: An Algorithmic Approach.",*** *2nd ed., Chapman*& Hall/CRC, 2014

[14] Shlomi Dolev, Amos Israeli, and Shlomo Moran., **"Self-stabilization of dynamic systems assuming only Read/Write atomicity."**, Distributed Computing, 7(1):3–16, 1993

[15] Alain Cournier, Ajoy K. Datta, Franck Petit, and Vincent Villain., **"Snap *stabilizing PIF algorithm in arbitrary networks. ",*** *Proc. of the 22nd International Conference on Distributed Computing Systems (ICDCS), pages 199–206, 2002.*

[16] Alain Bui, Ajoy K. Datta, Franck Petit, and Vincent Villain**., " State-optimal snap-stabilizing PIF in tree networks.",** Anish Arora, Ed., Workshop on Self-stabilizing Systems, pages 78–85, IEEE Computer Society, 1999

[17] Ted Richard Herman., **" Adaptivity through distributed convergence."** , Ph.D. thesis, University of Texas at Austin, 1992.

[18] Colette Johnen., **" Memory efficient, self-stabilizing algorithm to construct BFS *spanning trees."*** , *James E. Burns and Hagit Attiya, Eds., Proc. of the 16th Annual ACM Symposium on Principles of Distributed Computing, page 288, 1997*

[19] Mohamed G. Gouda and Nicholas J. Multari., **" Stabilizing communica tion protocols."**, IEEE Transactions on Computers, 40(4):448-458, 1991

[20] [20] Yehuda Afek and Anat Bremler-Barr., **" Self-stabilizing unidirectional network al*gorithms by power supply*** *.", Chicago Journal of Theoretical Computer Science, 1998.*

[21] Edsger W. Dijkstra., **" Self-stabilizing systems in spite of distributed control."**, Communications of the ACM, 17(11):643–644, 1974.

[22] Mohamed G. Gouda and Ted Herman., “ **Stabilizing unison.”**, Information Processing Letters, 35(4):171-175, 1990

[23] Alain Cournier, Ajoy K. Datta, Stéphane Devismes, Franck Petit, and Vincent Villain., **“ The expressive power of snap-stabilization.”**, Theoretical Computer Science, 626:40–66, 2016

[24] Shmuel Katz and Kenneth J. Perry., **“ Self-stabilizing extensions for message-passing systems**.”, Distributed Computing, 7(1):17–26, 1993

[25] Lélia Blin, Maria Potop-Butucaru, Stéphane Rovedakis, and Sébastien Tixeuil., **“*Loop-free super-stabilizing spanning tree construction.”*** *,In Proc. of the 12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), pages 50–64, Springer LNCS 6366, 2010*

[26] Shing-Tsaan Huang and Nian-Shing Chen**., “ Self-stabilizing depth-first token circulation on networks.”,** Distributed Computing, 7(1):61-66, 1993

[27] Alain Bui, Ajoy K. Datta, Franck Petit, and Vincent Villain., **“ Optimal PIF in tree *networks*.”,** *The 2nd International Meeting on Distributed Data and Structures 2 (WDAS), pages 1-16, 1999. 8, 47*

[28] Alain Cournier, Swan Dubois, Anissa Lamani, Franck Petit, and Vincent Villain., **“The snap-stabilizing message forwarding algorithm on tree topologies.”**, Theoretical Computer Science, 496:89–112, 2013

[29] Bertrand Ducourthial and Sébastien Tixeuil., **“ Self-stabilization with r-operators.”** ,Distributed Computing, 14(3):147-162, 2001

[30] Jorge A. Cobb and Mohamed G. Gouda., “ **Stabilization of routing in directed networks.”**,The 5th International Workshop on Self-Stabilizing Systems (WSS), pages 51–66, Springer LNCS 2194, Springer Berlin Heidelberg, 2001

[31] Ajoy K. Datta, Shivashankar Gurumurthy, Franck Petit, and Vincent Villain., **“Self-stabilizing network orientation algorithms in arbitrary rooted networks.”** ,Studia Informatica Universalis, 1(1):1-22, 2001

[32] Shing-Tsaan Huang and Nian-Shing Chen., **“ Self-stabilizing depth-first to*ken circulation on networks*.”** *, Distributed Computing, 7(1):61-66, 1993*

[33] Shing-Tsaan Huang, Tzong-Jye Liu, and Su-Shen Hung., **“ Asynchronous phase synchronization in uniform unidirectional rings.”**, IEEE Transactions on Parallel and Distributed Systems, 15(4):378–384, 2004

[34] Shing-Tsaan Huang and Nian-Shing Chen., **“ Self-stabilizing depth-first to*ken circulation on networks*.”** *, Distributed Computing, 7(1):61–66, 1993*

[35] Carole Delporte-Gallet, Stéphane Devismes, and Hugues Fauconnier., **“Stabilizing leader election in partial synchronous systems with crash failures.”,** Journal of Paralla! and Distributed Computing, 70(1):45–58, 2010

[36] Toshimitsu Masuzawa and Hirotsugu Kakugawa., “ **Self-stabilization in spite of frequent changes of networks: Case study of mutual exclusion on dynamic rings.”** ,In Ted Herman and Sébastien Tixeuil, Eds., Self-Stabilizing Systems, 7th International Symposium, (SSS), Proceedings, volume 3764 of Lecture Notes in Computer Science, pages 183–197, Springer, Barcelona, Spain, October 26–27, 2005

[37] Lélia Blin and Sébastien Tixeuil., **“ Compact deterministic self-stabilizing leader election on a ring: the exponential advantage of being talkative.”**, Distributed Computing, 31(2):139-166, 2018

[38] Pranay Chaudhuri and Hussein Thompson., “ **Improved self-stabilizing algorithms for 1(2, 1)-labeling tree net urks.”**, Mathematics in Computer Science, 5(1):27–39, 2011

[39] Volker Turau and Sven Köhler., **“ A distributed algorithm for minimum distance-k *domination in trees*.”** *, Journal of Graph Algorithms and Applications, 19(1):223–242*

[40] Ajoy K. Datta, Stéphane Devismes, Lawrence L. Larmore, and Vincent Villain., **“ Self-stabilizing weak leader election in anonymous trees using constant memory per edge.”**, Parallel Processing Letters, 27(2):1-18, 2017

[41] Sukumar Ghosh and Mehmet Hakan Karaata., **“ A self-stabilizing algorithm *for coloring planar graphs*.”**, *Distributed Computing, 7(1):55-59, 1993*

[42] Ji-Cherng Lin and Ming-Yi Chiu., **“ A fault-containing self-stabilizing algorithm *for 6-coloring planar graphs*.”**, *Journal of Information Science and Engineering, 26(1):163-181, 2010*

[43] Ajoy K. Datta, Colette Johnen, Franck Petit, and Vincent Villain., **“ Self-stabilizing depth-first token circulation in arbitrary rooted networks.”**,Distributed Computing, 13(4):207–218, 2000

[44] Stéphane Devismes, Swan Dubois, and Franck Petit., **“ Introduction to Distributed Self-Stabilizing Algorithms.”**, Karine Altisen,Copyright © 2019 by Morgan & Claypool

[45] *Rahul C. Shah and Jan M. Rabaey*., **“ *Energy aware routing for low energy ad hoc sensor networks*.”**, *IEEE Wireless Communications and Networking Conference (WCNC), March 2002*

[46] F. Wang, J. Liu., **“ Networked wireless sensor data collection: issues, challenges, and approaches.”** , Commun. Surv. Tutor. IEEE 13 (4) (2011) 673 -687

[47] J.Y. Chang, P.H. Ju., **“ An efficient cluster-based power saving scheme for wireless sensor networks.”** ,EURASIP J. Wirel. Commun. Netw. 2012 (1) (2012) 1-10

[48] D. Li, K.D. Wong, Y.H. Hu, A.M. Sayeed., “ **Detection, classification, and tracking of targets, Signal.”**, Proc. Mag. IEEE 19(2)(2002) 17-29

[49] Y. Durmus, A. Ozgovde, C. Ersoy., **“ Distributed and online fair resource management in video surveillance sensor networks.”,** Mobile Comput. IEEE Transac. 11 (5) (2012) 835-848.

[50] H. Modares, R. Salleh, A. Moravejosharieh., “ **Overview of security is sues in wireless sensor networks.”** **,**Third International Conference on

Computational Intelligence, Modelling and Simulation (CIMSIM), 2011, IEEE, 2011, September, pp. 308-311

[51] P.N. Reddy, P.I. Basarkod, S.S. Manvi., “ **Wireless sensor network based fire monitoring and extinguishing system in real time environment.”**, Int. J. Adv. Netw. Appl. 3 (02) (2011) 1070-1075.

[52] *Gokou Hervé Fabrice Diédié :, Boko Aka', Michel Babric*., **“Self-stabilising hybrid connectivity control protocol for WSNS.”**, IET Journal , 2018,July.

[53] Baccour, N., Koubằa, A., Noda. C., et al., **“Radio link quality estimation in low-power wireless networks”** (Springer International Publishing, London, 2013)

[54] Bas, C.U.. Ergen, S.C., **“Spatio-temporal characteristics of link quality in wireless sensor networks.”**,Proc. IEEE Wireless Communications and Networking Conf. (WCNC), Shanghai, China, April 2012, pp. 1152–1157

[55] Jalel Ben-Othman, Karim Bessaoud, Alain Bui and Laurence Pilard., “ **Self-stabilizing algorithm for energy saving in Wireless Sensor Networks.”**, IEEE Conference proce, pp.68-73,2011

[56] Jalel Ben-Othman, Karim Bessaoud, Alain Bui and Laurence Pilard.,” **An energy-efficient QoS routing for wireless sensor networks using self-stabilizing algorithm.”**,ELSEVIER Ad Hoc Networks.may 2015

[57] P. Suriyachai, J. Brown, U. Roedig., “ **Time-critical data delivery in wireless sensor networks, Distributed Computing in Sensor Systems.”**, Springer, Berlin Heidelberg, 2010, pp. 216-229.

[58] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan., “ **Energy- ficientcommunication protocol for wireless microsensor networks.”**,International Conference on System Sciences, 2000. Proceedings of the 33rd Annual Hawaii, IEEE, 2000, January, p. 10.

[59] Stephanie Lindsey, Cauligi Raghavendra, KrishnaM. Sivalingam., “ **Datagathering algorithms in sensor networks using energy metrics.”**, Parall. Distrib. Syst. IEEE Transac. 13 (9) (2002)924-935.