International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Data Aggregation Mechanism for Linear Wireless Sensor Networks (DAMLN)

Abdourakhmane Fall[1], Moussa Dethié Sarr[1], El hadji Malick Ndoye[2], Cheikh Sarr[1]
[1]Computer Science Department, Université Iba Der Thiam de Thiès, BP 967 Thiès, Senegal
[2]Department of Informatics, University of Assane Seck, Ziguinchor, Senegal

Abstract— *A linear wireless sensor network is a grouping of sensors connected and arranged in a topology made up of one or more axes of linear nodes. These types of wireless networks can be used in several areas such as industry, monitoring of gas or water pipes, road or rail infrastructure etc. The linearity of this type of sensor network constrains the information path and induces a rapid saturation of the links as one approaches the sink. This is because in a linear network, each node is both a data sensor and a relay. Consequently, we end up with a consumption which gradually increases in bandwidth during the routing of data to the sink (s).*
The data aggregation technique consists of consolidating multiple data packets into a single aggregate data packet. This aggregation technique will thus make it possible, for the case of a linear network, to optimize the consumption of bandwidth during data transfer, eliminate redundancies at the data level and also contribute to the network energy optimization.
The solutions made available by new standards such as ZigBee as well as the aggregation techniques studied, so far, to our knowledge, remain unsuitable for linear topologies. Faced with this, mechanisms adapted exclusively to linear networks must be the subject of reflection in order to remedy the many constraints (energy consumption, loss rate, delivery time).
In this paper we propose (DAMLN) an aggregation mechanism based on the clustering technique in which we develop three aggregation techniques. The proposed algorithms perform data aggregation after a self-training phase of nodes in clusters. Using the Castalia / Omnet ++ simulator, the mechanisms were evaluated according to three criteria: latency, packet loss rate and energy consumption of the sensors. A classification (of mechanisms), based on these criteria, was also made in conclusion.

Keywords— *Linear wireless sensors network, data aggregation techniques, cluster construction, energy opitimization*.

I. INTRODUCTION

A wireless sensor network is a set of sensors grouped together according to a specific topology. The roles of the sensors are to collect, process and transmit the data received or measured to a base station. These types of networks are now used in various fields and are the subject of several applications. In the case where these are deployed in a linear topology, we speak of Linear Wireless Sensor Networks (LWSN). These particular types of networks are used in gas, oil and stream pipelines [7] [8] [9] [10] [11] and in the monitoring of road or rail infrastructure.

Wireless sensor networks are characterized by a very limited lifespan caused by their low energy resource. Well aware of this fact, several methods and techniques have been developed in the direction of increasing the lifetime of the sensors, and therefore of the network. One of these techniques consists of aggregating the volume of data transmitted to the sink in order to reduce the number of transmissions at the nodes, and thus reduce the energy consumption at the level of the sensors. The aggregation technique also eliminates redundancy at the given level during the transmission process.

In order to achieve data aggregation, several approaches have been proposed. One of these approaches consists in making an aggregation based on a cluster architecture [4]. In a clustered topology, the nodes are grouped into groups called a Cluster. Within each cluster, a particular node is chosen as cluster leader. The organization of the cluster network as well as the choice of the cluster leader is based on previously defined algorithms. With this approach, the aggregation operation can be done within each cluster. The cluster-Heads will be responsible for aggregating the data sent by the nodes located in the same cluster as him.

However, in linear wireless sensor networks (LWSN) [2], these aggregation techniques become unsuitable due to the linear arrangement of the sensors and the high depth (number of hops) resulting in a very low lifetime of the sensors in the network. Based on this observation, it therefore becomes fundamental to reflect on aggregation mechanisms that are suitable for linear wireless sensor networks.

In this paper we propose (DAMLN) a data aggregation mechanism for linear networks. The mechanism is based on the cluster-based approach.

The article will be organized as follows. In the first part, a state of the art on data aggregation mechanisms in WSNs will be done. Part 2 will be devoted to the presentation of our solution. 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.

II. DATA AGGREGATION MECHANISMS IN WSNS

Several data aggregation techniques have been proposed in the literature. These can be classified according to four approaches namely the centralized approach, the approach based on the tree structure, the one based on the clusters and, finally, the aggregation in network [4].

A. *The cluster-based approach*

In this approach (Figure 1), the network is divided into several groups of sensors called cluster. In each cluster, a Cluster Head is chosen through an algorithm. One of the tasks of the Cluster Head is to aggregate data sent by other nodes belonging to the same cluster. The packet resulting from the aggregation is relayed to the sink through the other nodes.

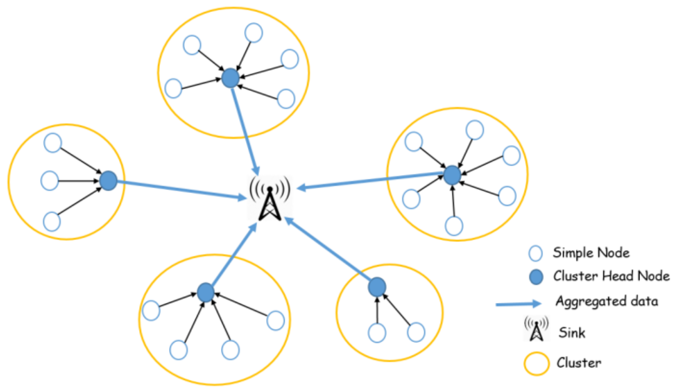


Fig. 1. Data aggregation based on cluster approach.

B. *Aggregation techniques based on the clustering approach*

Several aggregation techniques based on a cluster topology have been proposed in the literature. In [3], Sukhchandan and Sushma make a detailed and comparative study of several data aggregation techniques, based on a large collection of 932 published research articles. Detailed and critical analyzes were carried out covering fifty types of data aggregation techniques. In what follows we will give a presentation of some techniques.

In [18], Jung and al propose a technique of hybrid data aggregation called "combined clustering based data Aggregation". They perform dynamic aggregation using multiple clustering techniques simultaneously. Depending on the condition of the sensor network, the appropriate clustering technique is selected. In this technique, initially, a tree topology is created between the sink and the other nodes. The second phase consists in electing the clusters heads statically. This technique considerably increases the lifetime of the network and the energy consumption of the nodes. However, they do not consider latency as an optimization criterion. In [15], Chen and al propose (ADA) an adaptive data aggregation technique based on clustering. In this method, the load of the sensor nodes and the cluster heads is regularly shifted to the part of the network that is richest in terms of resources. The sink determines the transmission frequency of the sensor nodes, it also determines the aggregation frequency of the cluster heads. In [14], Mantri and al propose a two-level aggregation approach based on clustering (TTCDA). In this technique, two aggregation functions are applied to data packets using temporal and spatial correlation. The algorithm works in three steps: A cluster formation phase; an intra-cluster aggregation phase and an inter-cluster aggregation phase. In the first phase, the clusters are formed, the CHs are elected on criteria such as energy and the distance separating them from the well. In the second phase, the data from the sensor nodes are aggregated locally within each cluster by the Local Aggregators (LA). In the third phase, this aggregated data is transmitted to the Aggregators/Gateway (A/G) which perform a second aggregation by combining the received data into a single packet which will be transmitted to the sink. In [16], Zheng and al propose a distributed data aggregation mechanism. Their main objective is to maximize the overall compression gain of the sensor network by solving the Clustered Slepian – Wolf Coding (CSWC) problem. The algorithm they come up with can find optimal throughput within each cluster to minimize the cost of intra-cluster communication. Moreover, the authors do not consider the energy consumption of the nodes.

In [13], Maraiya and al propose (ECHSSDA) a technique of cluster formation which is based on a system of election of clusters heads. This technique makes it possible to reduce the overload of the cluster by electing a cluster head and associated cluster head. In clustering mechanisms, when a CH dies a new CH election is organized. This process of reorganizing clusters can be resource intensive. With ECHSSDA, this process is avoided thanks to the associated CH. In fact, when the energy level of CH reaches a certain critical level, the associated CH takes over and plays the role of the main CH. Their algorithm works in two phases. In the first phase, they elect the CHs and form the clusters. In the second phase the CHs receive the packets, perform data aggregation and transmit the aggregated packet to the sink. This technique dramatically increases the overall energy of the sensor network.

In [5], Xu and al propose hierarchical data aggregation using compressive sensing (HDACS). In this approach, the data compression limit is not static. It varies according to the size of the clusters in order to reduce the amount of data circulating in the network. The central idea is to form clusters following a hierarchy at several levels.

In [19], Yuan and al propose energy efficient and balanced cluster-based data aggregation algorithm (EEBCDA). In their approach, the authors, initially, divide the network into several rectangular zones, each zone is subdivided, also, into several rectangular sub-zones called grid. In each grid, the node with the greatest amount of energy is elected CH. Moreover, the latter is not fixed, it changes regularly in the grid depending on the node with the richest energy in terms of energy. The aggregated data is transmitted to the sink by the CHs directly by a send to a hop. The clusters farther from the sink contain more nodes. This technique optimizes and balances the energy consumption of the nodes and at the CH level. The more energy the CH expends, the greater the number of nodes that can succeed it. However, the latency at remote grids can be high.

In [17], Mantri and al present grouping nodes and clusters for efficient data aggregation (GCEDA). In this method, the nodes are organized in groups according to the nature of their data. The technique is based on three phases: Cluster formation, intra-cluster phase and inter-cluster phase. In the cluster formation phase, clusters are formed. The cluster-heads are elected according to their energy reserve and the Euclidean distance separating them from the well. In intra-cluster phase, nodes with similar data are organized into groups. Within each group, an aggregation function is applied based on the data. In the inter-cluster phase, the aggregated data is transmitted to the sink through the CHs.

In [12], Sinha and Lobiyal propose a data aggregation technique for a network of sensors with diverse energy reserves. The authors propose a data aggregation mechanism based on the entropy of the sensors. The technique is performed in two phases: an initial phase and a final phase. In the initial phase, nodes that are identical in terms of data are grouped into clusters. In the final phase, the other nodes that do not belong to any cluster choose their home cluster based on the degree of divergence between them and their neighbors. The proposed algorithm increases the lifetime of the sensor network, optimizes the transmission cost of nodes and improves the network convergence rate. Also, the algorithm does not take latency into account.

In [6], Mantri and al present bandwidth efficient cluster-based data aggregation (BECDA). The algorithm they propose considers a heterogeneous sensor network in which the sensors have different energy levels. The authors classify sensors into three types based on their energy level: " normal node " (20 J), " advance node " (30 J) and " super node " (40 J). The authors use the principle of packet-level data correlation in their aggregation function. The node with the highest number of one-hop neighbors and with the best amount of power will be chosen as the cluster head. Their proposal optimizes network energy and bandwidth usage.

III. Data Aggregation Mechanism in Linear wireless sensor Networks (DAMLN)

In this paper, we propose (DAMLN), a data aggregation mechanism for wireless sensor networks with linear topologies. We propose three aggregation techniques using the cluster-based approach.

The aggregation techniques based on the notion of clusters studied so far in the literature, in particular those listed in part 2. All become inefficient when they are applied in a network having a linear topology. Even though the approach remains the same, the constraints caused by topological linearity make these aggregation techniques unsuitable for linear topologies.

DAMLN assumes a homogeneous 2-redundant topology at node level, in which all nodes are neighbors at junction areas.

The DAMLN algorithm has two phases: a first phase called the cluster construction phase and a second phase called the data aggregation phase.

A. Cluster construction phase [1]

This cluster construction phase is itself made up of two phases: A first phase called node definition and neighborhood discovery phase and a second phase called cluster formation phase.

A.1. In the phase definition of nodes and discovery of the neighborhood (Algorithm1 of2CMJ) [1], each node determines in which zone it is located and therefore to deduce its nature (simple node or junction node). Indeed, in an LWSN with junctions, one finds, mainly, two zones: the strictly linear zones and the junction zones. In strictly linear zones, the nodes have at most, $2\times k$ neighbors. k being the degree of redundancy (in our case four neighbors). In the junction zones, the nodes located there have at least $2 * k$ neighbors. Starting from this fact, we define two types of nodes: Simple nodes (SN), that is to say, those belonging to strictly linear areas, and junction nodes (JN), i.e. nodes lying in junction areas.

At the start of the network, each node broadcasts a Hello packet:

a. If a node receives a maximum of four Hello packets from different neighboring nodes, received_Hello ≤ 4, then the node self-elects as a single node (SN)

b. If the number of Hello packets received from different neighbor nodes is strictly greater than 4, received_Hello > 4, then the node elects itself as a junction node (JN).

After defining its nature, the node communicates it to its neighbors and registers all of its neighbors in a list.

At the level of junction areas, we define another type of node, besides junction nodes (JN), these are gate nodes (GN). The gate node is the node connecting a junction area to a strictly linear area, in other words, a packet leaving a junction area will meet a gate node last. If this is an incoming packet, the first node to receive this packet will be the GN. The number of NGs in a junction area is proportional to the complexity of the area. The higher the complexity of the zone, the higher its number of (GN).

Simple nodes (SN) having a neighbor of type (JN) communicate the list of their broadcast neighbors, neighbors (NJ) receiving such a packet check whether, among the sending nodes, there are two which are one-hop neighbors (close neighbor). If so, then the node self-elects as a gate node (GN) (Algorithm 2 of 2CMJ) [1].

Once the (GN) have been chosen, the second phase can begin.

A.2. In the second phase, the clusters are built gradually in a linear fashion. We will distinguish, here, two types of cluster: Simple Clusters (SC) made up only of simple nodes, and Junction Clusters (JC) made up only of junction nodes. This cluster construction phase is based on two stages:

a. Step 1 reuses the M2CRL algorithm [1] for the formation of clusters at the level of strictly linear areas (Algorithm 3 and 4 of 2CMJ) [1]

b. At 2nd step, as in M2CRL, the node closest to the sink will be elected Cluster Head. Cluster Head coordinates the process of creating clusters within the JC as described below (Algorithm 5 of 2CMJ) [1]:

Once defined, the CH triggers a search for gate nodes by sending a Hello-GN broadcast. Only the GN will respond to this packet in unicast to the CH. The CH records the various GNs located in his cluster after receiving GN responses. At this stage, the CH perfectly knows the type of its neighbors, the GNs and the JNs. Following this, the cluster leader sends two types of build packets in the cluster:

1. A classically constructed packet, like those used in (M2CRL), multicast to neighboring non-GN nodes. The latter initializes their Pn to 2, the other variables (Cid, Cp) are equal to those contained in the construction package received from the CH.

2. A second construction packets intended for the different (GN) of the cluster.

At each GN, the CH sends a construction packet containing an index i (unique for each GN) which will be used for the calculation of the Cid of the following Simple Cluster. The (GN) set their Pn to 2, the other variables (Cid, Cp) are equal to those of CH. Each GN calculates the Cid (x) of the next Simple Cluster (CS) connected to it using (1). The following CS sets its Cid to the value (x) contained in the construction package sent by the GN (Algorithm 6 of 2CMJ) [1]. Construction continues in strictly linear areas.

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

After performing the self-construction of the clusters, the data sent to the sink will be aggregated by the cluster-heads during the aggregation phase. We thus propose, in this paper, three aggregation techniques based respectively on three mathematical functions (f_{A1}(Cp) f_{A2}(Cp) f_{A3} (Cp)).

B. *Data aggregation phase*

In this part we have proposed three algorithms based on three aggregation techniques. For each technique, we define an aggregation function which depends on the position of the cluster within which the aggregation is performed. In each cluster, aggregation is performed by the cluster head. The aggregation functions depend on the position of the cluster that performs the aggregation, so the aggregation wait time increases as one moves away from the sink. Indeed, because of the linearity of the sensor network, the activity rate of a sensor depends on its position relative to the sink, the closer the sensor is to the sink, the greater its activity rate. This is also applicable to clusters.

B.1 Notations

- We denote by $\boldsymbol{T_A^{C_p}}$ the time you wait for the aggregation of a cluster with position p defined by the function $\boldsymbol{f_A(C_p)}$. In other words, it also constitutes the time that the cluster head will wait to aggregate the data it receives.
- We denote by $\boldsymbol{\tau}$ the network latency limit: τ constitutes the transmission time limit, not to be exceeded for any packet sent from the source to the destination, defined by the network administration.
- We denote by **ψ** the latency time until the sink of a packet sent from a cluster of position p (propagation time + aggregation time).
- For any transmitted packet its transmission time will always be less than or equal to τ: $\boxed{\boldsymbol{\psi \leq \tau}}$

We denote by $\boldsymbol{T_{C_p \to S}}$ the theoretical propagation time of a packet sent from C_p to the sink (time that the packet would do without an aggregation process). $\boxed{\boldsymbol{T_{C_p \to S} = 0.05 \times C_p}}$

- We denote by $\boldsymbol{p}$ the maximum depth of the network

B.2 Algorithms

The three proposed mechanisms are based on the following five general principles:

- Initially, the Sink evaluates the definition domain of $\boldsymbol{\tau}$, it also defines the function $f_A(C_p)$ to use
- These parameters are included in the build packets during the process of creating the clusters.
- CHs calculate their aggregation wait time using $f_A(C_p)$
- If the aggregate packet size limit is reached, the aggregate packet is transmitted immediately even if the aggregation wait time has not yet expired.
  - Aggregated packets are routed through intermediate nodes to the sink

B.2.1 Mechanism 1

This technique (Algorithm 1) works through an algorithm which is based on the following two principles:

- The aggregation process is triggered upon receipt of a first data packet
- Each cluster-head calculates its aggregation wait time using $f_A(C_p)$. The CH performs the aggregation of packets coming, only, from nodes belonging to its own cluster.
- The aggregated packets will be relayed automatically by the intermediate nodes to the sink.

(2) gives us the function $f_{A1}(C_p)$ which defines the aggregation wait time.

$$f_{A1}(C_p) = \frac{\tau \times C_p}{p+1} - T_{Cp \to S} \qquad (2)$$

(Table I) gives a description of the variables and functions used in the proposed algorithms.

TABLE I. Table of variables and functions

Variables and functions	Description
$\boldsymbol{T_A}$	waiting time for aggregation in the CH
t	packet size limit to aggregate
$\boldsymbol{\tau}$	maximum latency
$\boldsymbol{T_{C_p \to S}}$	Theoretical propagation time
$\boldsymbol{P_{Aggr}}$	aggregation packet
Temporary_Data_List()	Contains local data to be aggregated
Packet_Is_Data_packet	test if received packet is a data packet
Packet_Is_Aggregated_packet	test if received packet is a Aggregated packet
Packet_Destination	destination address
Packet_received→dup()	duplicate the received packet
Received_Packet_List.size()	the number of aggregated data contained in the aggregation packet

```
Algorithm 1: Data aggregation technique: Mechanism 1
  Result: ..........
  Data: Float:$T_A$, $t$, $\tau$, $T_{C_p \to S}$
  Data: Packet: $P_{Aggr}$
  Input: Packet: Received_Packet
  Input: List: Temporary_Data_List
begin
    if (Aggregation_Timer == 0) then
        Send_Aggregate_Data(Temporary_Data_List);
    if (Packet_Is_Data_Packet && IsCH) then
        if (Received_Packet $\notin$ Temporary_Data_List) then
                                        // This condition eliminates redundancy
            Add(Temporary_Data_List, Received_Packet)
        if (Temporary_Data_List.size()==1) then
            /* this is the first received packet                            */
            $T_{C_p \to S} = 10^{-2} \times C_p$
            $T_A = \frac{\tau \times C_p}{p+1} - T_{C_p \to S}$
            Start_Timer(Aggregation_Timer, $T_A$)
        if (Temporary_Data_List.size()== t) then
            Send_Aggregate_Data(Temporary_Data_List);
             End_Timer(Aggregation_Timer, $T_A$)
    if (Packet_Is_Aggregated_Packet && Received_Packet−>destination != self) then
        $P_{Aggr}$ = Received_Packet−>dup()
        Send($P_{Aggr}$, parent_node)
        /* immediate retransmission                                         */

```

B.2.2 mechanism 2

In this technique (Algorithm 2), each CH performs the aggregation of packets coming from its child nodes (nodes belonging to the same cluster as the CH).

Once the aggregation has been carried out by the CH, two possible cases arise:

1. If the size limit of an aggregated packet is reached, then the aggregated packet will be relayed automatically by the intermediate nodes to the sink
2. If the size limit of an aggregate packet is not reached, then the aggregate packet continues to receive data at intermediate CHs until its size limit is reached. The addition of data at the intermediate CHs is done by following the FIFO rule, in other words the packets waiting the longest at the level of a packet being aggregated, will be added first.

We note T'_A the waiting time for adding data at the level of intermediate clusters.

(3) gives us the function $f_{A2}(C_p)$ which defines the wait time for the aggregation of a CH.

$$f_{A2}(C_p) = \frac{\tau \times C_p}{p+1} - [T_{Cp\to S} + (C_p * T'_A)] \ \ with \ \ \{T'_A = 10^{-3}s) \quad (3)$$

Algorithm 2: Data aggregation technique: **Mechanism 2**

```
Result: ..........
Data: Float: $T_A$, $T'_A = 10^{-3}$, $\tau$, $T_{C_p\to S}$
Data: Packet: $P_{Aggr}$
Input: Packet: Received_Packet
Input: List: Temporary_Data_List
begin
    if (Aggregation_Timer == 0) then
        Send_Aggregate_Data(Temporary_Data_List);
    if (Packet_Is_Data_Packet && IsCH) then
        if (Received_Packet ∉ Temporary_Data_List) then
            ;                                   // This condition eliminates redundancy
            Add(Temporary_Data_List, Received_Packet);
        if (Temporary_Data_List.size()==1) then
            /* this is the first received packet                              */
            $T_{C_p\to S} = 10^{-2} \times C_p$;
            $T_A = \frac{\tau \times C_p}{p+1} - (T_{C_p\to S} + C_p \times T'_A)$;
            Start_Timer(Aggregation_Timer, $T_A$);
        if (Temporary_Data_List.size()== t) then
            Send_Aggregate_Data(Temporary_Data_List);
             End_Timer(Aggregation_Timer, $T_A$);
    if (Packet_Is_Aggregated_packet && IsCH && Received_Packet->destination != self )
     then
        if (Temporary_Data_List.size() > 0 && Received_Packet− > aggr.size() < t) then
            int a = Received_Packet− > aggr.size();
            ;                          // we proceed to an inter-cluster aggregation
            for (int i=a; i<=t; i++) do
                Add(Received_Packet− > aggr(i), Temporary_Data_List.front()
                 Temporary_Data_List.pop_front();
                /* this instruction removes the elements already added in the temporal
                   list of data to be aggregated                                   */
                if (Temporary_Data.size() == 0) then
                    break;
        $P_{Aggr}$ = Packet_received− > dup();
        Send($P_{Aggr}$, parent_node);
    if (Packet_Is_Aggregated_packet && !IsCH && Received_Packet->destination != self )
     then
        $P_{Aggr}$ = Received_Packet− > dup();
        Send($P_{Aggr}$, parent_node);
        /* immediate retransmission                                            */
```

B.2.3 mechanism 3

In this technique, first, the Sink evaluates the domain of definition of τ in order to choose the function $f_{A3}(C_p)$ to use. Each CH performs the aggregation of the data sent by the child nodes. We propose a hierarchical aggregation at two levels:

- An intra-cluster aggregation is first performed by the CH containing the data coming from the child nodes.
- An inter-cluster aggregation will then possibly be carried out at the level of the intermediate clusters.

Indeed, after performing the intra-cluster aggregation two possible cases arise.

1. The size limit of an aggregated packet has been reached, in this case the aggregated packet is automatically relayed by the intermediate nodes to the Sink
2. The size limit of an aggregated packet is not reached, in which case the aggregated packet will remain for some time in the intermediate clusters in order to undergo inter-cluster aggregation. This second scenario differs from that of mechanism 2 by the fact that, here, even if the intermediate cluster has no data awaiting aggregation, the aggregation packet, received, will remain there for a waiting time maximum equal to $f_{A3}(C_p)$ hoping to undergo inter-cluster aggregation. In other words, each aggregation packet will remain a hold time equal to at most $f_{A3}(C_p)$, at the level of each intermediate cluster.

Theorem 1 gives us the defined aggregation function $f_{A3}(C_p)$.

Therorem 1:

$$\boldsymbol{A})\ \ if\ \ \tau > \frac{p^2}{2}:\ \boldsymbol{f_{A3}(C_p) = T_A^p + (C_p - p)}$$

$$with \begin{cases} T_A^p \in\ \ ](p-1); (\frac{\tau}{p} + \frac{p-1}{2)})[\\ \\ p - 1 < T_A^p < \frac{\tau}{p} + \frac{p-1}{2)} \end{cases}$$

$$\boldsymbol{B})\ if\ \tau \geq \frac{p^2}{2}:\ \boldsymbol{f_{A3}(C_p)} = \frac{\boldsymbol{\tau}}{\boldsymbol{p+2}} + (\boldsymbol{C_p \times \theta})$$

$$with\ \{\theta = \frac{2\tau}{p(p+1)^2}$$

Proof of $\boldsymbol{A})\ \ if\ \ \tau > \frac{p^2}{2}:\ \ f_{A3}(C_p) = T_A^p + (C_p - p)$

$f_{A3}(C_p) = T_A^p - (p - C_p)\ \ with\ T_A^p > p - 1\ \ \boxed{1}$

$We\ set\ f_{A3}(C_p) = U_n = T_A^p + (n - p)$

$we\ have\ U_1 = T_A^p + (1 - p)\ ;\ U_2 = T_A^p + (2 - p)\ ;\ U_3 = T_A^p + (3 - p);\ \ldots..\ U_p = T_A^p + (p - p) = T_A^p$

$\boldsymbol{U_n = U_{n+1} - 1}$

$U_n\ is\ arithmetic\ sequence\ of\ reason\ (-1) converging\ to\ T_A^p$

$$we\ set\ \psi = \sum_{n=1}^{p-1} (T_A^p - n)$$

ψ is the sum of the aggregation wait times performed in all the clusters for a packet leaving a cluster of position p

$$\psi = U_1 + U_2 + U_3 + \cdots \ldots \ldots \ldots \ldots .. + U_p$$

$$\psi = T_A^p + (T_A^p - 1) + (T_A^p - 2) + (T_A^p - 3) + \cdots \ldots \ldots + [T_A^p - (p-1)]$$

$$\psi = T_A^p + \sum_{n=1}^{p-1} (T_A^p - n)$$

$$\psi = T_A^p + \sum_{n=1}^{p-1} T_A^p - \sum_{n=1}^{p-1} n$$

$$\psi = T_A^p + (p-1)T_A^p - \frac{p(p-1)}{2}$$

$$\psi = p \times T_A^p - \frac{p(p-1)}{2}$$
$$\psi = p\left(T_A^p - \frac{p-1}{2}\right)$$
$$we\ have\ \psi < \tau \Rightarrow p\left(T_A^p - \frac{p-1}{2}\right) < \tau$$
$$T_A^p - \frac{p-1}{2} < \frac{\tau}{p}$$
$$T_A^p < \frac{p-1}{2} + \frac{\tau}{p} \quad \boxed{2}$$
$$\boxed{1}\ and\ \boxed{2}\ gives\ \boxed{(p-1) < T_A^p < \frac{p-1}{2} + \frac{\tau}{p}}$$

Proof of $\boldsymbol{B}$)

$$if\ \ \tau \geq \frac{p^2}{2}:\ f_{A3}(C_p) = \frac{\tau}{p+2} + (C_p \times \theta)$$
$$with\ \{\theta = \frac{2\tau}{p(p+1)^2}$$

Algorithm 3: Data aggregation technique: **Mechanism 3**

```
Result: ..........
Data: Float:$T_A$, $T'_A = 10^{-3}$, $\tau$, $T_{C_p \to S}$
Data: Packet: $P_{Aggr}$
Input: Packet: Received_Packet
Input: List: Temporary_Data_List
begin
    if ($\tau > \frac{p^2}{2}$) then
    begin
        $T_A = rand\left((p-1);(\frac{\tau}{p} + \frac{p-1}{2})\right) + (C_p - p)$
    end
    else
        $T_A = \frac{\tau}{p+2} + \left(C_p \times \frac{2\tau}{p(p+1)^2}\right)$;
    if (Aggregation_Timer == 0) then
        Send_Aggregate_Data(Temporary_Data_List);
    if (Packet_Is_Data_Packet && IsCH) then
        if (Received_Packet) ∉ Temporary_Data_List) then
            Add(Temporary_Data_List(), Received_Packet);
        if (Temporary_Data_List.size()==1) then
            Start_Timer(Aggregation_Timer, $T_A$);
        if (Temporary_Data_List.size()== t) then
            Send_Aggregate_Data(Temporary_Data_List);
             End_Timer(Aggregation_Timer, $T_A$);
    if (Aggregation_Timer_Two == 0 && Temporary_Data_List.size() > 0) then
        int a = Received_Packet− > aggr.size();
        for (int i=a; i<=t; i++) do
            Add(Received_Packet− > aggr(i), Temporary_Data_List.front)
             Temporary_Data_List.pop_front();
            if (Temporary_Data_List.size() == 0) then
                break;
        $P_{Aggr}$ = Received_Packet− > dup();
        Send($P_{Aggr}$, parent_node);
    if (Packet_Is_Aggregated_Packet && IsCH && Received_Packet->destination != self )
     then
        if (Temporary_Data_List.size() > 0 && Received_Packet_List.size() < t) then
        begin
            /* Same process with mechanism 2                                    */
        end
        else
            Start_Timer(Aggregation_Timer_Two, $T_A$);
    if (Packet_Is_Aggregated_Packet && !IsCH && Received_Packet->destination != self )
     then
        $P_{Aggr}$ = Received_Packet− > dup();
        Send($P_{Aggr}$, parent_node);
        /* immediate retransmission                                         */
```

calculate ψ et compare it to τ

$$\psi = \sum f_{A3}(C_p) = \left[\frac{\tau}{p+2} + (1 \times \theta)\right] + \left[\frac{\tau}{p+2} + (2 \times \theta)\right] + \left[\frac{\tau}{p+2} + (2 \times \theta)\right] + \ldots\ldots + \left[\frac{\tau}{p+2} + (p \times \theta)\right]$$

$$\psi = \frac{p \times \tau}{p+2} + \theta\left[\sum_{n=1}^{p}(n)\right]$$
$$\psi = \frac{p \times \tau}{p+2} + \theta\left[\frac{p(p+1)}{2}\right]\ replace\ \theta\ by\ its\ expression$$
$$\psi = \frac{p \times \tau}{p+2} + \left[\frac{2\tau}{p(p+1)^2} \times \frac{p(p+1)}{2}\right]$$
$$\psi = \frac{p \times \tau}{p+2} + \frac{\tau}{p+1}$$
$$\psi = \frac{(p\tau)(p+1)+\tau(p+2)}{(p+2)(p+1)}$$
$$\psi = \frac{\tau[p(p+1)+(p+2)]}{(p+2)(p+1)}$$
$$we\ set\ \psi = \frac{\tau \times A}{B}\ with \begin{cases} A = p(p+1)+(p+2) \\ B = (p+1)(p+2) \end{cases}$$
$$B > A\ then\ \frac{\tau A}{B} < \tau \qquad \boxed{\psi < \tau}$$

IV. Simulations and Resultats

In order to highlight the three mechanisms proposed in this paper, we performed simulations using the Castalia environment which is based on the OMNet ++ platform.

The mechanisms are executed on a topology of 52 nodes (51 sensor nodes and the sink) with three junction zones of respective complexity 2, 3 and 4. The topology used is 2-redundant at the level of strictly linear zones. After execution of the phase of creation of the clusters, the topology gives a perfect construction of the clusters. The cardinality at the level of simple clusters is equal to 3 and that of the junction zones depends on the degree of complexity of the latter.

Figure 2 gives us the average latency time of packets sent to the sink over several series of transmissions performed in targeted areas of the sensor network. On rounds, 1, 2, 3, and 4 we made several series of transmissions from four clusters of position equal to 8. On rounds 5, 6, and 7 the transmissions were made from three clusters of position equal to 6. Finally on rounds 8 and 9 the transmissions were made from two clusters of position equal to 4. The maximum latency time τ not to be exceeded (we recall that this time must be defined by the network administrator) has been set to 7 seconds for mechanism 1 and 2, and 18 seconds for mechanism 3.

Mechanism 1 obviously offers a more optimal latency time with a value approaching half that of τ. This is because the packets, once transmitted by the source, do not experience additional latency at the level of intermediate clusters (automatic retransmissions). For mechanism 2, the latency time remains slightly higher than that of mechanism 1 with a value quite close to that of τ for transmissions made from clusters of position 8 (rounds 1 2 3 4). The reason for this increase is that non-full, transmitted aggregate packets may possibly wait some time at intermediate clusters in order to undergo further aggregation. Mechanism 3 records a much greater latency time than the other two mechanisms. This is explained by the fact that each non-full aggregate packet transmitted. Obviously waits for a time less than or equal to $f_A(C_p)$.

In conclusion, the results of figure 2 clearly show us that mechanism 1 offers a better latency time of the aggregated packets transmitted.

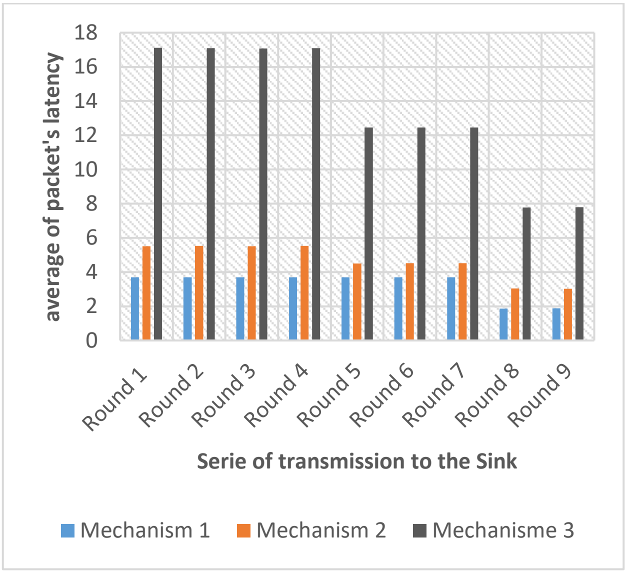


Fig. 2. Average of packet's latency vs Serie of transmission

Figures 3 and 4 show us the packet loss rate as a percentage of the three mechanisms on all the sensors in the network except the node of id 1 (because the one transmits directly to the sink).

We can see that the three mechanisms all have an average loss rate that is strictly less than 1.6%, which makes them very efficient in terms of packet loss. Mechanism 2 has a better packet loss rate with a value slightly lower than that of mechanism 3. However, both remain more optimal than mechanism 1 in terms of packet loss.

Figure 3 shows the distribution of this rate of packet loss per node. We can see that the evolution of the loss rate is quite identical for the three mechanisms with a succession of spikes for the eight nodes closest to the sink. This is explained by the fact that, in this area, the traffic is quite high, this is caused by the linear structure of the network, all the data sent to the sink will necessarily pass on this part of the network. This reasoning will apply, of course, to the nodes next to the latter, hence the succession of spikes observed at nodes 26, 27 and 28.

In conclusion, what should be remembered is that regarding the three mechanisms, the second technique is more optimal in terms of packet loss.

Figure 5 gives us the average energy consumption of the sensors, in joules, for the three mechanisms. The simulations performed use a sensor model running on two AA batteries whose typical energy is equal to 18,720 joules. Figure 8 gives us the average consumption of nodes after 10 rounds of transmissions. Aggregation mechanisms are applied during transmissions.

The results show a more optimal energy use at the level of the mechanism 3, the mechanism 2 presents a slightly lower use compared to the mechanism 1. In summary we can affirm that in terms of energy consumption of the nodes, the mechanism 3 remains better compared to the other mechanisms

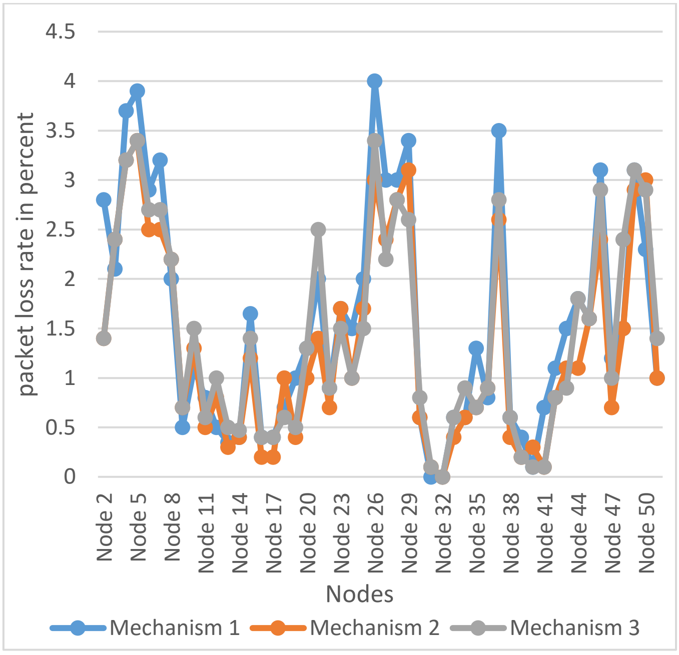
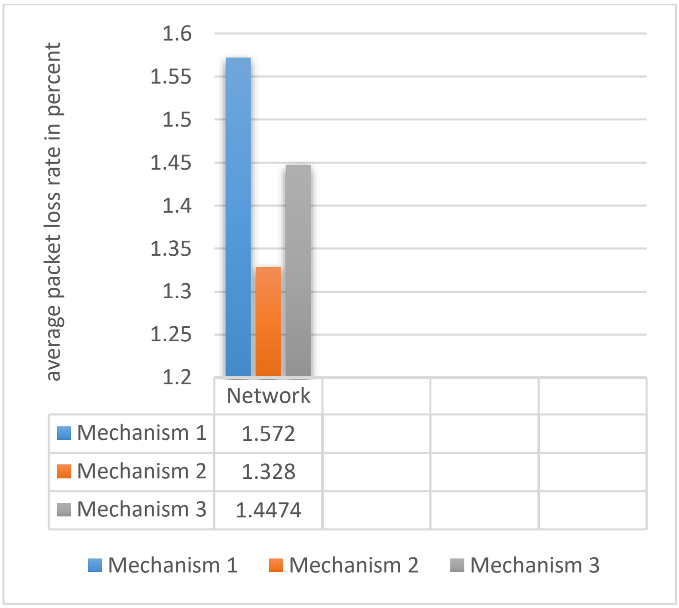


Fig. 3. Packet loss rate per node in percent



	Network
Mechanism 1	1.572
Mechanism 2	1.328
Mechanism 3	1.4474

Fig. 4. Average packet loss rate per mechanism in percent

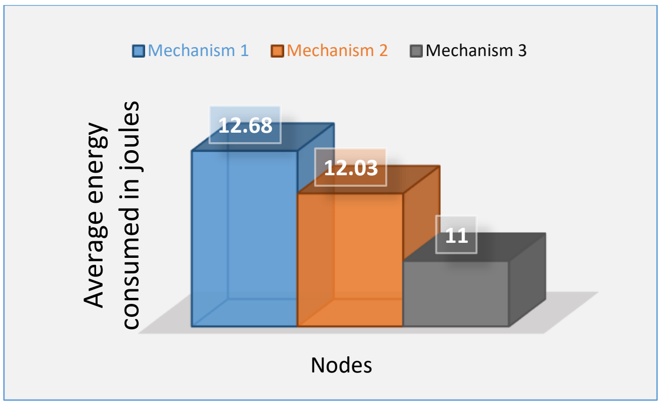


Fig. 5. Average energy consumed by the network for each mechanism

(Table II) gives us a classification of the three mechanisms based on the criteria of power consumption, packet loss and latency.

TABLE II. Mechanisms classification table

	Latency	**Loss rate**	**Network lifetime**
Mechanism 1	Very high	Low	Low
Mechanism 2	High	Very High	High
Mechanism 3	Low	High	Very High

V. Conclusion and perspectives

In this paper we have proposed three data aggregation mechanisms for linear topology wireless sensor networks. The mechanisms are all based on the technique based on clustering. The clustering-based aggregation techniques studied in the literature all relied on non-linear topology networks both on the aggregation process and on the cluster formation phase itself. The mechanisms thus proposed were able to resolve this incompatibility problem. The comparison criteria taken are: the latency, the rate of packet loss and energy consumption of the sensors. This allowed us to study for each criteria, what would be the most suitable technique. For example, for emergency and alert networks where latency is of major importance, mechanism 1 is more recommended, etc.

As a perspective, we plan to improve the mechanisms by adding an energy optimization policy based on an election of nodes which will be responsible for performing the aggregation tasks, the choice may be based on the nodes having a large energy reserve and not always the CH. Regarding energy optimization, we can apply a routing technique in the sense of using the most optimal nodes (in terms of energy) as relays to route the aggregated data to the sink. Finally, a generalization of the proposed techniques is possible which would allow them to be applied to k-redundant topologies whatever the value of k.

References

[1] Fall, A., Sarr, M. D., & Sarr, C., “Clusters construction mechanism for junction linear wireless sensor networks (2cmj),” *Journal of Scientific and Engineering Research*, vol. *8*, issue *8*, pages *15-29*, 2021.

[2] Fall A., Sarr M.D., Sarr C. “Clusters Construction Mechanism for Strictly Linear Wireless Sensor Networks” *presented at the Thorn J., Gueye A., Hejnowicz A. (eds) Innovations and Interdisciplinary Solutions for Underserved Areas.*, 2020.

[3] Randhawa, S., Jain, S., “Data Aggregation in Wireless Sensor Networks” *Previous Research, Current Status and Future Directions. Wireless Pers Commun*, 97, 3355–3425 2017.

[4] Nandini. S. Patil, Prof. P. R. Patil, “Data Aggregation in wireless sensor Network,” *IEEE International Conference on Computational Intelligence and Computing Research,* ISBN: 97881 8371 3627.

[5] Xu, X., Ansari, R., Khokhar, A., & Vasilakos ‘‘Hierarchical data aggregation using compressive sensing (HDACS) in WSNs’’ *ACM Transactions on Sensor Networks (TOSN)*, 11(3), 1–25. 2015

[6] Mantri, D. S., Prasad, N. R., & Prasad, R. ‘‘Bandwidth efficient cluster-based data aggregation for wireless sensor network. *Computers & Electrical Engineering,* 41,256–264. 2015

[7] Ian F. Akyildiz, Weilian SU, Yogesh Sankara Subramaniam and Erdal CAYIRCI. A survey on sensor networks”. *In: IEEE Communications magazine* 40.8, p. 102–114 (cf. p. 1) 2002

[8] Konrad Lorincz, Matt Welsh, Omar Marcillo. ‘‘Deploying a wireless sensor network on an active volcano’’. *In: IEEE Internet Computing*, p. 18–25 (cf. p. 1, 11). 2006

[9] Imad Jawhar, Nader Mohamed and Khaled Shuaib. “A framework for pipeline infrastructure monitoring using wireless sensor networks”. *In: Wireless Telecommunications Symposium*. *WTS. IEEE,* 1–7 (cf. p. 1, 12). 2007

[10] Sukun KIM, Shamim Pakzad, David Culler ‘‘Health monitoring of civil infrastructures using wireless sensor networks”. *In: Proceedings of the 6th international conference on Information processing in sensor networks.* ACM Press, p. 254–263 (cf. p. 1, 3, 12). 2007

[11] C. R. Baker, K. Armijo, S. Belka ‘‘Wireless sensor networks for home health care”. *In: Advanced Information Networking and Applications Workshops, AINAW’07. 21st International Conference* on. T. 2. 2007, 832–837 (cf. p. 1, 10) 2007

[12] Sinha, A., & Lobiyal, D. K. *‘‘Performance evaluation of data aggregation for cluster-based wireless sensor network*’’ .Human-Centric Computing and Information Sciences, 3(1), 1–13. 2013

[13] Maraiya, K., Kant, K., & Gupta, N. “Efficient cluster head selection scheme for data aggregation in wireless sensor network”. *International Journal Computer Applications,* 23(9), 10–18. 2011

[14] Mantri, D., Prasad, N. R., Prasad, R., & Ohmori, S. “Two tier cluster based data aggregation (TTCDA) in wireless sensor network”. *IEEE International Conference Advanced Networks Telecommunciations Systems,* 117–122. 2012

[15] Chen, H., Mineno, H., & Mizuno, T. “Adaptive data aggregation scheme in clustered wireless sensor networks”. *Computer Communications,* 31(15), 3579–3585. 2008

[16] Zheng, J., Member, S., Wang, P., & Li, C. “Distributed data aggregation using Slepian–Wolf coding in cluster-based wireless sensor networks.” *IEEE Transactions on Vehicular Technology,* 59(5), 2564–2574. 2010

[17] Mantri, D., Prasad, N. R., & Prasad, R. “Grouping of clusters for efficient data aggregation (GCEDA) in wireless sensor network.” *In 3rd IEEE International advance computing conference IACC* (pp. 132–137).2013

[18] Jung, W. S., Lim, K. W., Ko, Y. B., & Park, S. J. “Efficient clustering-based data aggregation techniques for wireless sensor networks.” *Wireless Networks,* 17(5), 1387–1400. 2011

[19] Yuea, J., Zhang, W., Xiao, W., Tang, D., & Tang, J. “Energy efficient and balanced cluster based data aggregation algorithm for wireless sensor networks.” *Procedia Engineering,* 29, 2009–2015. 2012