CFMS: A CLUSTER-BASED CONVERGECAST FRAMEWORK FOR DENSE MULTI-SINK WIRELESS SENSOR NETWORKS

Gokou Hervé Fabrice Diédié[1], Koigny Fabrice Kouassi[2] and Tchimou N'Takpé[2]

[1]Laboratory of Mathematics and Computer Science, Université Peleforo Gon Coulibaly, Korhogo, Ivory Coast
[2]Laboratory of Mathematics and Computer Science, Université Nangui Abrogoua, Abidjan, Ivory Coast

ABSTRACT

Convergecast is one of the most challenging tasks in Wireless Sensor Networks (WSNs). Indeed, this data collection process must be conducted while copying with packet collisions, nodes' congestion or data redundancy. These issues always result in energy waste which is detrimental to network efficiency and lifetime. This paper is aimed to address these problems in large-scale multi-sink WSNs. Inspired by our previous work MSCP, we designed a lightweight protocol stack that seamlessly combines clustering, path-vector routing, sinks' duty cycling, data aggregation and transmission scheduling in order to minimise message overhead and packet losses. Simulation results show that this solution can mitigate delay while significantly increasing packet delivery and network lifetime.

KEYWORDS

Wireless Sensor Networks, Multi-sink, Clustering, Convergecast, Aggregation tree, Scheduling

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are composed of small resource-constrained monitoring devices also called sensor nodes and a sink (i.e. gateway). The latter is in charge of connecting the resulting infrastructure to the outside world. Nowadays, WSNs are pervasive and used for various applications in domains like ecology, health, security, transportation, research or industry [1] - [3].

In such networks, sensor nodes detect physical events and transmit the derived data to the gateway through a multi-hop communication pattern [4]. This process (commonly referred to as convergecast), implies routing a large flow of data to a single point [5][6]. Interference and congestion are inherent to this kind of traffic and lead to packet losses [7]. They always provoke delays and energy waste, hence reduction of network efficiency and lifetime [8].

A common solution especially in dense WSNs, involve increasing the numbers of sinks, clustering the sensor nodes, reducing redundancy, controlling collision or congestion via data aggregation and transmission scheduling [9][10] [11]. Unfortunately, most of such contributions found in the literature are not scalable or make use of assumptions, metrics and strategies that are actually too restrictive for any real-world implementation.

In this paper, we propose to combine clustering, path-vector routing and scheduling schemes in order to dynamically assign a proper sink to each sensor node, duty cycle nodes activity, construct a collision-free data aggregation tree, while balancing network load. The resulting framework helps collect periodical data, mitigate delay, minimize packet losses and prolong network lifetime.

The main contributions of this work are summarized as follows:

- a hop limitation clustering technique that mitigates message overhead and network flooding;
- a route discovery processes that simultaneously consider links' asymmetry, interference, and nodes' congestion ;
- a fully-distributed loop-free path-vector routing strategy that minimizes delays between sensor nodes and sinks;
- a duty-cycling strategy that helps balance loads for both sensor nodes and sinks;
- a tree construction and scheduling scheme for collision-free data aggregation.

The rest of the paper is organized as follows: Section 2 surveys the related contributions; then, the proposed solution is detailed in Section 3; the performance evaluation process, the results, and discussions are presented in Sections 4 and 5 followed by conclusion in Section 6.

2. RELATED WORK

In WSNs, data aggregation and traffic scheduling-aware convergecast schemes are generally classified into tree-based solutions and cluster-based ones [12] - [15]. These works as well can be categorized according to the routing, the aggregation or the traffic scheduling techniques they use. However, regardless the strategy used only cluster-based are actually scalable [16]. This paper is aimed to focus on such solutions.

LEACH by Heinzelman *et al.* [17] is one of the early solutions proposed in this category. It has inspired many contributions for the past two decades [18] [19]. Regrettably, most of them are dedicated to only one sink [20]. In this IoT era, many of these single-sink oriented solutions are still proposed despite their low applicability [21] - [23].

Yu and Li [24] proposed the first hierarchical solution for the minimum-time collision-free aggregation scheduling problem in a multi-sink environment. A Voronoï-based scheduling strategy is suggested to construct a dominating set for all the sensor nodes in forests (i.e. clusters) of trees around each sink. Schedules are then made as well as for dominatee and dominator nodes. Unfortunately, Voronoï partition requires nodes to know their exact positions.

Goyal *et al.* [25] In this paper, an improved data aggregation technique for cluster-based UWSN (Under Water Sensor Network) is proposed where an efficient sleep-wake up algorithm is used for aggregating the sensed data and TDMA based transmission schedule is used to avoid intra and inter cluster collisions to reduce the data redundancy and ensure energy efficient collision-free transmission.

Khan *et al.* [26] suggested a typical multi-layer cluster-based strategy for UWSNs. Nodes are grouped in equal size layers according to their depth. CHs are elected based on their residual energy and distance to sinks. After cluster formation CHs assign a TDMA schedule to other members to help them send data which is aggregated by CHs. This scheme is based on one-hop clusters which could foster interference between CHs.

Rajput and Kumaravelu [27] used a Fuzzy-c-means-based strategy to balance the size of clusters while optimizing the number of sinks and their locations in the deployment area. Regrettably, this solution is limited to applications with a deterministic sink deployment and is not scalable; since the latter deployment and the clustering process are centralized in the distant base station.

Singh and Nagaraju [28] proposed a *Sink-As-CH* strategy [29]-[31] to help reduce number of intermediary hops. Sensor nodes are assigned to clusters according to their Euclidean distance to the sink. Packets are then routed by constructing a Wiener minimum spanning tree based on an ABC (Artificial Bee Colony) optimization scheme. Opportunistic coding is finally used at each node to select the best flows passing through intermediary nodes and to aggregate data of intersecting flows. Regrettably, packet transmission is implicitly scheduled via a queuing process. Daas *et al.* [32] designed a distributed multi-hop cluster-based routing protocol where paths are selected using both hop count and link state via Signal-to-Noise-Ratio. This protocol also leverages to help balance the load of both sensor nodes and sinks. However, this solution requires fixed size clusters thus can be hardly applied to randomly deployed networks, hence to monitor phenomena in harsh environments.

Verma *et al.* [33] proposed a bio-inspired optimisation scheme heterogeneous WSNs. A genetic algorithm-based clustering strategy is used to select CHs via a metric combining residual energy, distance to the sink, and node density. To help mitigate fast energy depletion in sinks' one-hop neighbourhood, they suggest deploying sinks outside the region of interest in order to directly connect CHs to sinks. Unfortunately, this strategy is too restrictive for real-life applications which requiring random deployment.

Recently, another bio-inspired solution was proposed by Chada and Gugulothu [34]. They suggested an optimisation-based scheme using the multi-objective black widow metaheuristics. The latter mimics the cannibalism behaviour of spider black widows. Candidate CHs referred to as spiders are chosen leveraging their residual energy and distance to the sinks via a pheromone-based strategy. Data can be aggregated by CHs before transmission. However, no explicit scheduling scheme is proposed to cope with possible collision.

3. PROPOSED SOLUTION

This section is aimed to present our solution named CFMS (Convergecast Framework for Multi-Sink networks). We discuss our motivations, objectives and assumptions, before detailing the latter solution.

3.1. Motivations and Objectives

As shown in previous section, few state-of-the-art cluster-based multi-sink convergecast protocols have been proposed so far. In our previous work, named MSCP [35] we proposed a proactive loop-free scheme to increase network efficiency and lifetime, besides scalability, by considering interferences and link asymmetry, while duty cycling activity of both sensor nodes and sinks. Unfortunately, MSCP struggles to cope with data redundancy, collisions and node congestion. Furthermore, by allowing each node to individually send its data, MSCP is actually more suitable for event-driven applications. This work is aimed to address these issues while providing a good trade-off between network efficiency and lifetime to periodic (time-driven) application.

3.2. Assumptions

We assume that:

- nodes are equipped with an omni-directional radio;
- each node has a unique identifier (ID);
- nodes are uniformly and randomly deployed in the area of interest;
- nodes' connection is modelled as an UDG (Unit Disk Graph);
- each node can assess distances through the received signal strength or a specific localization protocol;
- the number of sensor nodes is higher than the number of sinks;
- each node has a finite size buffer using a FIFO policy;
- nodes are static.

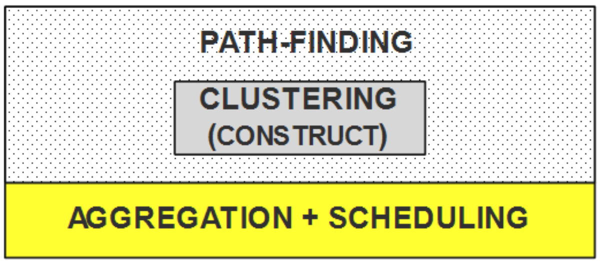


Figure.1 Process stack of CFMS.

3.3. Description

As shown in Figure 1, CFMS is composed of three processes, namely the clustering process, the path-finding process, and the data aggregation-scheduling process. The first two processes are respectively inspired from MSCP presented in our previous work [35].
We detail in the following sub-sections these processes.

3.3.1. Clustering Phase

This is a two-step cyclic phase starting with neighbour discovery followed if necessary by the creation of a new cluster. Each node u must give a unique integer number to each of its neighbours. This number is randomly chosen in interval $\left[1;\left|N(u)\right|\right]$; where $N(u)$ denotes the set of these neighbours.

We use a scheme similar to the one proposed for the clustering process inside the CONSTRUCT protocol [31]. Note that each cluster is identified by its Cluster Head ID (CHID); hence, each node knows the cluster it belongs to. Besides, we use a *Sink-As-CH* strategy [29], i.e. each sink creates a cluster in its *k*-hop neighbourhood; *k* is given as a parameter such as *k* = *max_hop_count*.

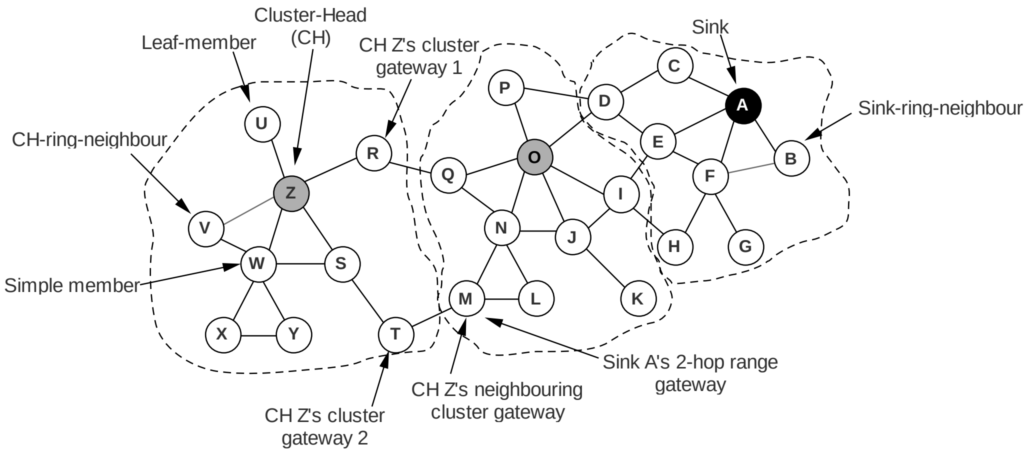


Figure.2 Nodes' statuses after the clustering phase

The duration of CHs' service time is fixed as a parameter. Therefore, after its *mandate* a CH must abandon its status and launch a new election process in its *k*-hop neighbourhood. Moreover, when two CHs move next to each other, the one with less cluster mates will eventually lose its role and become a new cluster mate; ties are randomly broken Figure 2 depicts the different statuses applicable to each node in a section of the network after the cluster formation phase. Equation (1) helps calculate the CH's fitness score; where $n, Ei, Er, CC, S(t)$ respectively denote the number of its neighbours, its residual energy, its initial energy, its clustering coefficient and its stability index (see [31] for details).

$$score = \alpha \times \left(\frac{n-1}{n}\right) + \beta \times \left(\frac{Er}{Ei}\right) + \delta \times (1 - CC) + \gamma \times S(t) \tag{1}$$

3.3.2. Intra-Cluster Convergecast Sub-Tree Construction

After the creation of a new cluster, the CH must construct the intra-cluster data forwarding sub-tree and interconnect it to its inter-cluster counterpart. To do so, CH *u* broadcasts in its *k*-hop neighbourhood a TREE-REQ message containing its ID then triggers a timer (*TREE-timer*) and waits for any response during $max\_hop\_count \times t_{wait}(u)$ seconds. This duration is calculated using Equation (2); where rtt(.) denotes the round-trip-time (in seconds) experienced with node *v* during neighbour discovery. *N(u)* is node *u*'s neighbourhood.

$$t_{wait}(u) = \max\{\text{rtt}(v); v \in N(u)\} \tag{2}$$

When receiving a TREE-REQ message from a neighbour u, a node v must increment the value of the *hop-count* field to 1 then estimates the transmission delay $\delta_{u,v}$ using Equation (3); where $C_{u,v}$, $L_{u,v}$ and S_u respectively denote the capacity (i.e. maximum transmission rate) of link (u,v), the received packet length on this link and the queuing delay (sojourn time) of this message inside neighbour u.
Note that S_u is provided by the latter neighbour via a dedicated field in the message.

$$\delta_{u,v} = \frac{L_{u,v}}{C_{u,v}} + S_u \tag{3}$$

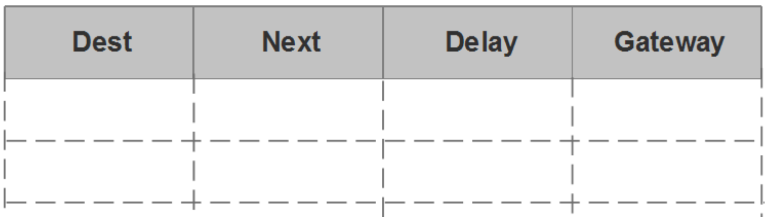
$$C_{u,v} = W_{u,v} \times \log_2(1 + SINR_{u,v}) \qquad (4)$$

$C_{u,v}$ is estimated using Equation (4); where $W_{u,v}$ and $SINR_{u,v}$ respectively denote the bandwidth of link (u,v) and the Signal Interference plus Noise Ratio experienced on this link.

Node v then increments the *total-delay* ield with the estimated delayf $\delta_{u,v}$. Node v must also insert its *ID* into the *first-forwarder* field if the latter node is a neighbour of the TREE-REQ message's sender.

A node must respond to a TREE-REQ message by broadcasting a TREE-ACK message if this node is a neighbouring gateway (i.e. is affiliated to another CH but has at least one neighbour inside the sender's cluster), is a leaf-mate (i.e. a cluster member with only one neighbour) or is a CH-ring-neighbour (i.e. has exactly two neighbours of which one is the latter CH) see Figure 2 for illustration.

TREE-ACK messages contain the sender's ID, its CH*'s* ID, the value of the *first-forwarded* field provided by the received TREE-REQ message, a list of IDs of its neighbouring sinks, the delay to each of these sinks, and the estimated *total delay* of the TREE-REQ message.



Dest	Next	Delay	Gateway

Figure.3 Structure of convergecast table

Note that TREE-REQ messages are forwarded by a node if the sender is a cluster mate (CHID =ID), this message is received for the first time, and its *hop_count* field's value is below *max_hop_count* +1. Moreover, to mitigate the protocol overhead, TREE-REQ messages are never sent back to a forwarder.

When forwarding a TREE-REQ message to its neighbours, a node sends the list of pairs (unique number, neighbour's ID). The recipient neighbour then concatenates the given unique number to the content of this message's *cast_vector* field.

TREE-ACK messages are forwarded following the same rules applied to TREE-REQ except that the CH's ID is not compared to the forwarder ID.

After receiving a TREE-ACK message sent by a neighbouring cluster gateway, the CH node u must update its convergecast table (as shown in Figure 3) by calculating the delay to each destination (i.e. sink) via Equation (5); where $\delta_{u,x}$, $\delta_{u,s}$ and $\delta_{s,x}$ respectively denote delays from CH u to sink x, from u to the sender s and from the sender s to sink x.The *first forwarder* provided by the TREE-REQ message is the next-hop. For the same destination only the lowest delay so far experienced is kept in this table.

$$\delta_{u,x} = \delta_{u,s} + \delta_{s,x} \qquad (5)$$

TREE-ACK messages sent by other eligible nodes help CH to determine total delay on the senders-to-CH paths.

After *TREE-timer* expiration, the CH must reply to each received TREE-ACK message, via the *first forwarder* node, by sending a CAST message of which field *type* is respectively set to 1 if the receiver (i.e. sender of the TREE-ACK message) is a neighbouring cluster gateway and 0 otherwise.

Each CAST message also contains the list of sinks discovered by the CH and the delays to reach them. Before sending this message, CH copies into the *cast_vector* field, content of the corresponding field of the received TREE-ACK message.

When receiving a CAST message with *type* field equals to 1, a node u must update its convergecast table by calculating its transmission delay $\delta_{u,x}$ to each sink x as expressed by Equation (5); where $\delta_{u,x}$, $\delta_{s,x}$ and $\delta_{s,u}$ respectively denote delays from node u to sink x, from the sender s to sink x and from the sender s to node u.
The latter must analyse content of the *cast_vector* field, then extract the unique number of which position equals value of the *hop_count* field. Neighbour that was assigned this unique number is chosen as the next hop.

$$\delta_{u,x} = \delta_{s,x} - \delta_{s,u} \tag{6}$$

However, when a leaf-member or a CH-ring-neighbour (see Figure 2 for illustration) receives a CAST message, it must respond by sending a CAST-ACK. Except that a CH-ring-neighbour must send a CAST-ACK only after receiving a CAST message forwarded by its non-CH neighbour.

Note that CAST and CAST-ACK messages are forwarded following the same rules applied to TREE-REQ and TREE-ACK ones.

A typical CAST-ACK message contains the list of sinks and the delays to reach each of them. The *cast_vector* field embeds the forwarders' unique number. All these information were extracted from the CAST message previously sent by the CH.

After receiving a CAST-ACK message a node u must use Equation (5) to update its convergecast table by calculating the delay $\delta_{u,x}$ to reach each sink x via the CH s; then must extract from the *cast_vector* field, the unique number of which position equals value of the *hop_count* field; so as to choose as next hop, the neighbour that was assigned the latter unique number.

3.3.3. Inter-Cluster Convergecast Sub-Tree Construction

Note that only sinks are allowed to periodically (i.e. at a beginning of a new duty-cycle) broadcast a HELLO message to clusters located at most *max_cluster_hop_count* hops.
After sending such a HELLO message a sink *u* must trigger a timer (*HELLO-timer*) and wait for any response from next-hop gateways during $max\_cluster\_hop\_count \times t_{wait}(u)$ seconds (see Equation (2)). HELLO messages are forwarded like TREE-ACK messages.

Once a node receives a new HELLO message, if the forwarder's CHID is different from its own CHID, the latter node must increment the *cluster_hop_count* field's value to 1 and forward this message only if the value of this field is lower than *max_cluster_hop_count*. This node must also insert its ID into the *first-forwarder* field if it is a neighbour of the message's sender (i.e. the sink). Only a range gateway (i.e. last cluster hop gateway), a leaf-mate (i.e. member with only one neighbour), sink-ring-neighbour (i.e. a node that has exactly two neighbours of which one is the sender sink), must respond by broadcasting a HELLO-ACK. (See Figure 1 for illustration). Note that a HELLO-ACK message is forwarded only by nodes that have previously received a HELLO message from the same sink.

After receiving a new HELLO-ACK from a neighbour u, a node v estimates the transmission delay $\delta_{u,v}$ using Equation (3); then increments the *total-delay* field with the latter estimated delay $\delta_{u,v}$ and finally forwards the message.

After *HELLO-timer* expiration, the sink must reply to each received HELLO-ACK message, via the *first forwarder* node, by sending a CAST message where the field *type* is set to 0. This message also contains the total delay value extracted from the HELLO-ACK. Such CAST messages are forwarded using the same rules applied to the HELLO ones.

When a node receives a CAST message, it must reply by sending a CAST-ACK. Except that sink-ring-neighbours must send a CAST-ACK message only after receiving a CAST message forwarded by its non-sink neighbour. CAST-ACK messages must also contain the total delay value extracted from the CAST ones. Besides, the *cast_vector* field of each CAST-ACK message contains the values provided by its corresponding CAST message.

Therefore, when receiving a CAST-ACK message sent by node s, a node u can update its convergecast table by calculating its transmission delay $\delta_{u,x}$ to the sink x using Equation (5) where $\delta_{u,x}$, $\delta_{s,x}$ and $\delta_{s,u}$ respectively denote delays from node u to sink x, from the sender s to sink x and from the sender s to node u. Then the latter must analyzes content of the *cast_vector* field, in order to extract the unique number of which position equals value of the *hop_count* field. The neighbour that was assigned the latter unique number is chosen as the next hop.

To help balance the load, every sink must shift to sleep mode after t_{wake} seconds, i.e. if its energy consumption ratio ξ reaches a threshold ρ defined as a parameter. ξ is calculated using Equation (7) where E_i and E_f respectively denote sink's energy at the beginning and at the end of its current duty-cycle. Before being inactive, sinks must alert their cluster mates by broadcasting a SLEEP message. The latter will be forwarded like HELLO messages.

$$\xi = \frac{E_i - E_f}{E_i} \tag{7}$$

This idle state will last t_{sleep} seconds. The latter duration is determined using Equation (8) where α denotes the number of times a sink has been active; $\beta > 0$ is the Weibull distribution shape parameter; R is uniformly and randomly chosen in the interval $]0;1[$.

$$t_{sleep} = \frac{1}{\alpha} \times \ln(\frac{1}{R})^{\frac{1}{\beta}} \tag{8}$$

It is also noteworthy to mention that a CH must also trigger a timer (*BUILD-timer*) to help reconstruct the intra-cluster part of the convergecast tree after t_{build} seconds. This duration is determined using Equation (9); where $\overline{t_i}$ denotes the mean time between wake-ups of the i^{th} neighbouring sink while λ is given as a parameter.

$$t_{build} = \frac{\min(\overline{t_i}, \overline{t_{i+1}}, ..., \overline{t_n})}{\lambda}, \quad \lambda \geq 2 \tag{9}$$

Note that $\overline{t_i}$ is calculated then included in HELLO messages sent by sink *i*. Sink's range gateways also spread this information along with the list of active sinks when sending TREE-ACK messages.

3.3.4. Data Sending Process

After each cluster member has found a path to the nearest sink, the data sending process must be scheduled in order to favour aggregation and mitigate collisions or nodes' congestion. To do this, the data sending coordinator (i.e. the CH or the sink) broadcasts a SCHED message containing the ID of a gateway randomly chosen then triggers a *SCHED-timer*.

When receiving this message, only *periphery-members* whose paths to their nearest sink pass through the latter gateway must reply by sending a SCHED-ACK.
Note that inside a cluster the term "periphery-members" refers to nodes (except gateways) that are located max_hop_count hops from the coordinator.

The sender of a SCHED-ACK message must specify the total delay to reach the designated gateway in addition to the lowest capacity (i.e. forwarded node's buffer length) encountered on the path towards this gateway. This information is recorded during the path-finding process. After expiration of the *SCHED-timer*, the data sending coordinator constructs a schedule leveraging all the received SCHED-ACK messages. To do this, it must infer from $\hat{f}_x$ (i.e. the buffer remaining space) of gateway x the duration t_{data} of the current process.

Equation (10) helps to estimate f_{data} the portion of data that each periphery-member can send to this gateway. $\tilde{N}_x$ denotes the set of periphery-members eager to send data to gateway x.

$$f_{data} = \frac{\hat{f}_x}{\tilde{N}_x} \tag{10}$$

Equation (11) helps estimate $t_{data}(u)$ the time slot assigned to periphery-member u to send its data. $\tilde{\delta}_{u,x}$ denotes the delay between periphery-member u and the chosen gateway x.

$$t_{data}(u) = f_{data} \times \tilde{\delta}_{u,x} \tag{11}$$

After calculating the schedule (i.e. the set of durations $t_{data}(u)$), the coordinator broadcasts an AGGR message containing the chosen gateway's ID and the amount of data f_{data} to be sent by each periphery-member. The coordinator randomly chooses the periphery-member which should start the process. The coordinator then triggers a timer *AGGR-timer* for $\hat{t}_{data}$ seconds. This duration is estimated using Equation (12).

$$\hat{t}_{data} = \sum_{u \in \tilde{N}_x} t_{data}(u) \tag{12}$$

When receiving such a message, only a periphery-member that has previously sent a SCHED-ACK containing the chosen gateway's ID, must start the data sending process. If so, such a periphery-member will send its data through a DATA message to the next neighbour according to its convergecast table.

Upon receiving such a message, a node must aggregate its own data (if destined to the same gateway), before sending the result.

Note that, in this *snowball effect*, gateways act like ultimate aggregators of the data collected by other cluster members.

After expiration of *AGGR-timer*, coordinator ends the current transmission session by randomly choosing among the remaining gateways a new ultimate aggregator before broadcasting a new SCHED message and so on.

Note that all these messages are forwarded based on the same rules applied to those used for the path-finding process.

Figure 4 a) depicts SCHED-ACK messages sent by some periphery-members whose shortest path to sink A goes through gateway T randomly chosen by the CH Z. Figure 3 b) shows the convergecast and aggregation tree constructed by the CH based on all the SCHED-ACK messages received. Gateway T is the ultimate aggregator of data sent by periphery-members U, R, Y and X. Node U is randomly chosen to initiate the data sending process according to durations in the schedule calculated by CH. Figure 5 and 6 depicts the processes used by CFMS.

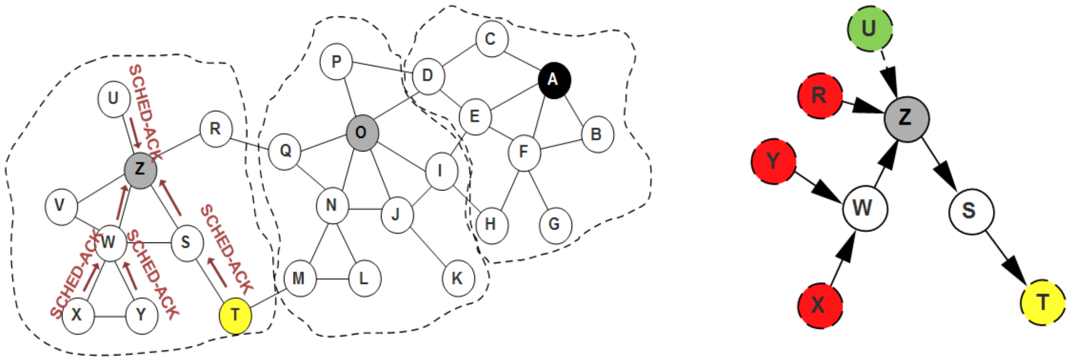


Figure.4 Example of Convergecast tree construction: a) Periphery-members responses. b) The resulting Direct Acyclic Graph (DAG).

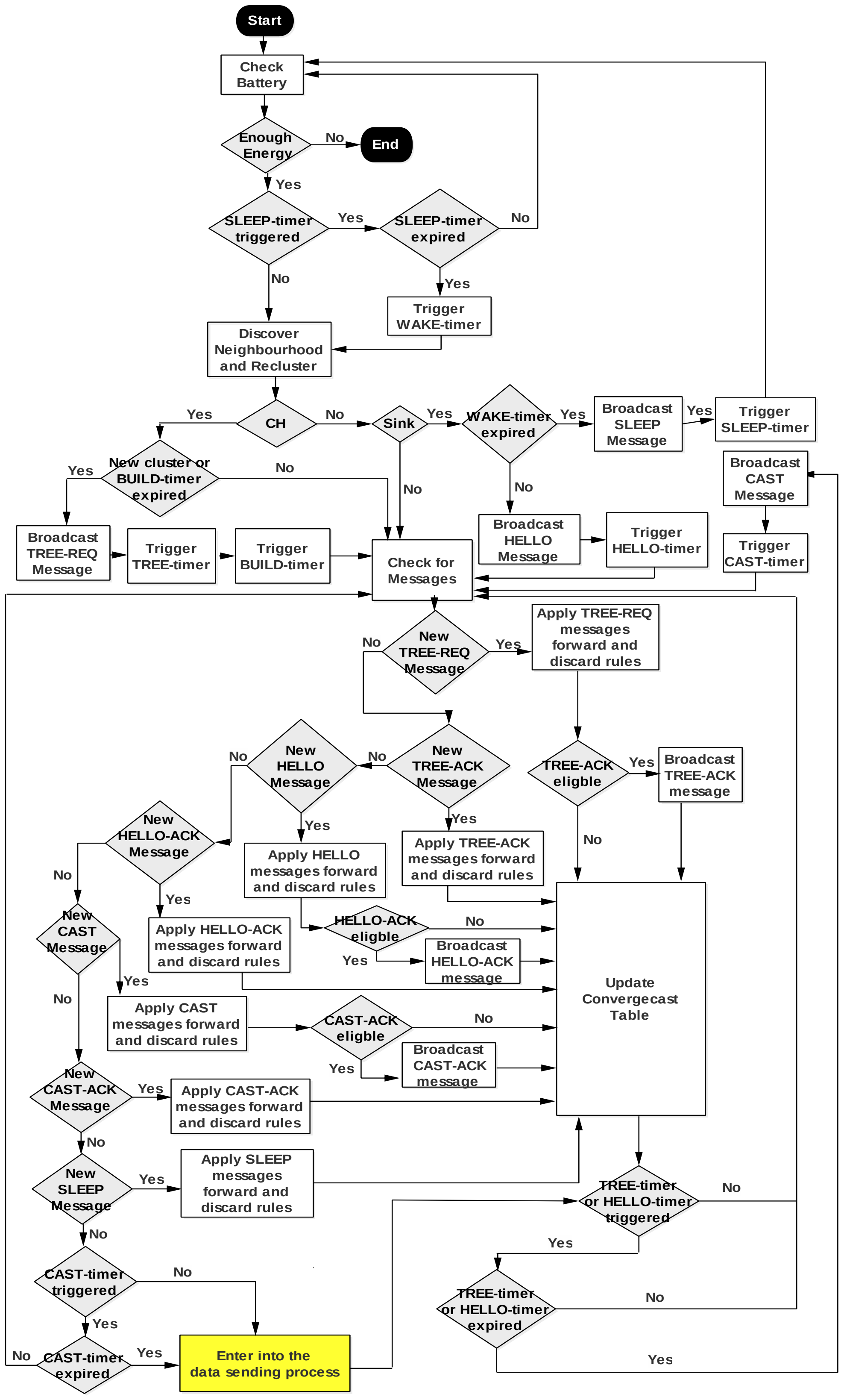

Start
Check Battery
Enough Energy
No
End
Yes
SLEEP-timer triggered
Yes
SLEEP-timer expired
No
No
Yes
Trigger WAKE-timer
Discover Neighbourhood and Recluster
Yes
CH
No
Sink
Yes
WAKE-timer expired
Yes
Broadcast SLEEP Message
Yes
Trigger SLEEP-timer
Broadcast CAST Message
Yes
New cluster or BUILD-timer expired
No
No
No
Broadcast HELLO Message
Trigger HELLO-timer
Trigger CAST-timer
Broadcast TREE-REQ Message
Trigger TREE-timer
Trigger BUILD-timer
Check for Messages
Apply TREE-REQ messages forward and discard rules
No
New TREE-REQ Message
Yes
No
New HELLO Message
No
New TREE-ACK Message
TREE-ACK eligble
Yes
Broadcast TREE-ACK message
Yes
Yes
No
New HELLO-ACK Message
Apply HELLO messages forward and discard rules
Apply TREE-ACK messages forward and discard rules
No
Yes
New CAST Message
Apply HELLO-ACK messages forward and discard rules
HELLO-ACK eligble
No
Yes
Broadcast HELLO-ACK message
Yes
No
Apply CAST messages forward and discard rules
CAST-ACK eligble
No
Update Convergecast Table
Yes
Broadcast CAST-ACK message
New CAST-ACK Message
Yes
Apply CAST-ACK messages forward and discard rules
No
New SLEEP Message
Yes
Apply SLEEP messages forward and discard rules
TREE-timer or HELLO-timer triggered
No
No
Yes
CAST-timer triggered
No
TREE-timer or HELLO-timer expired
No
Yes
No
CAST-timer expired
Yes
Enter into the data sending process
Yes

Figure.5 Flowchart of the path-finding process of CFMS

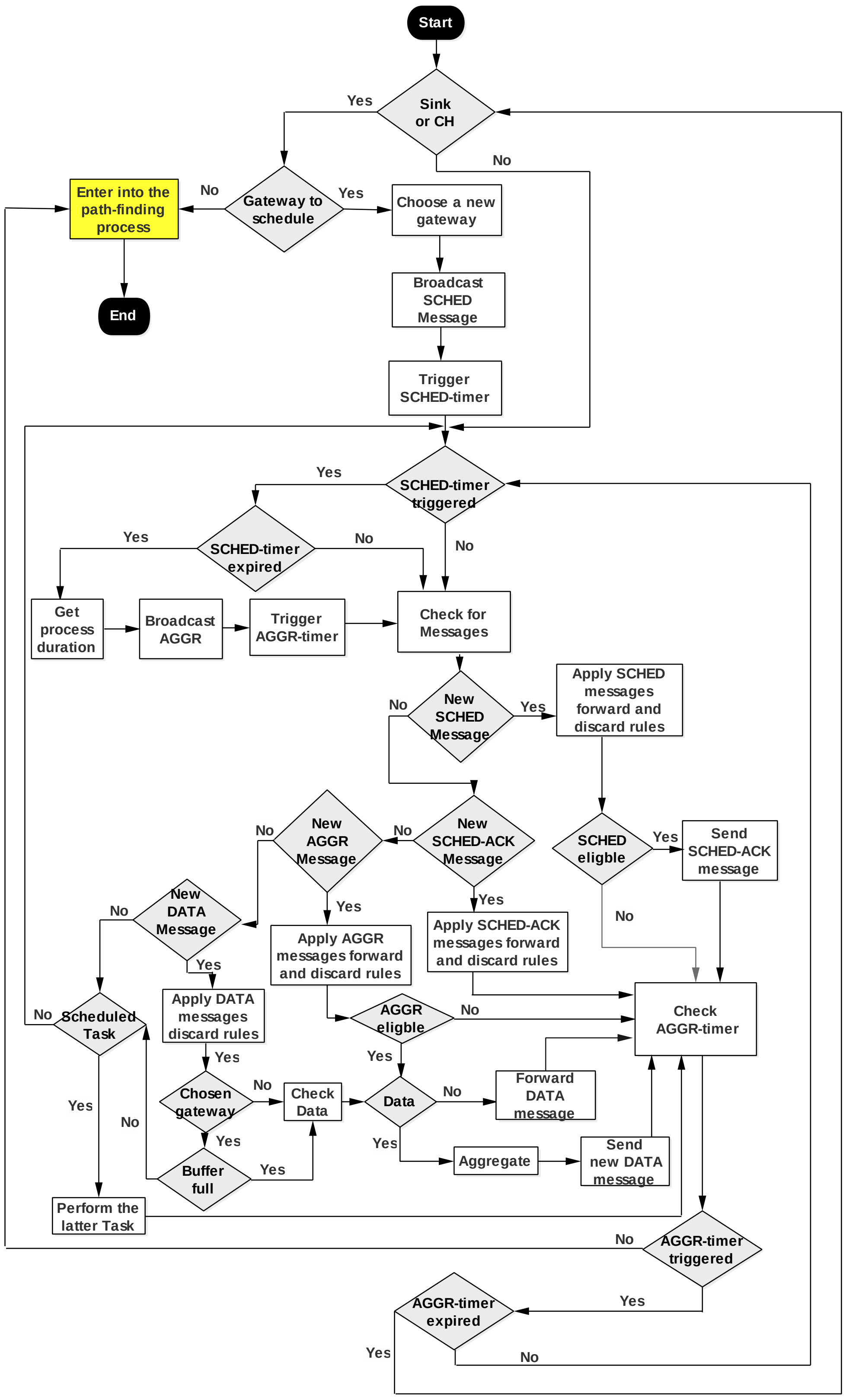


Figure.6 Flowchart of the path-finding process of CFMS

4. EXPERIMENTAL SET-UP

In this section, we detail the extensive simulation campaign we carried out. Experiments were made with OMNeT++ simulator version 6.0 [36]. We used the energy consumption model proposed by Heinzelman *et al.* [17]; the other parameters are summarized in Tables 1 - 4. The results were compared to those we obtained with our previous work named MSCP (Multi-Sink Convergecast Protocol) and the solution proposed by Singh and Nagaraju [28] that we will refer to as S&N.

Table 3 presents parameters we used to randomly and uniformly vary link quality; namely, PRR (Packet Reception Ratio), SNR (Signal to Noise Ratio), SINR (Signal Interference plus Noise Ratio) and LQI (Link Quality Indicator) [37]. Tables 2 and 4 are inspired by the specifications of the IEEE 802.15.4 standard [38]. We evaluated the ability of three protocols to efficiently transfer data to the sinks through three metrics, namely, the *average end-to-end delay*, the *packet delivery ratio*, and the *network lifetime* [32]. To do this, we randomly and uniformly deployed sensors and sinks varying their population as described in Table 1, respectively using, a 100 and a 2 steps scale so that the sink-to-sensor ratio is 0.02. We specifically investigated how the latter populations influenced these metrics. To vary link quality, we used the uniform distribution to randomly change parameters described in Table 3. Each 2.5s, 30% to 50% of nodes were randomly chosen to send data. The latter was queued in a buffer in order to wait for a transmission schedule when we applied CFMS. We used in CFMS the same data aggregation scheme proposed for S&N. The reason for that is twofold; first, providing a specific data aggregation scheme is beyond the scope of this paper and secondly, we wanted to evaluate CFMS and S&N on the same basis.

This experiment was replicated 50 times for each variation of the number of nodes. Results were averaged with a 95% confidence interval. The experiment started after all the nodes were deployed and was ended according to two network lifetime definitions respectively when a sensor and a sink depleted is energy.

Table.1 Simulation general parameters

Parameter	**Value**
deployment zone	1000 m X 1000 m
number of sensors	100 - 1000
number of sinks	2 - 20
sensors' transmission ranges	127 m
sinks' transmission ranges	250 m
max_hop_count	2
max_cluster_hop_count	2
sensors' initial energy	0.2 J
sinks' initial energy	20 J
self-discharge per second	0.1 μJ
E_{elec} (see [17])	50 nJ/bit
e_{fs} (see [17])	10 pJ/bit/m^2
e_{amp} (see [17])	0.0013 pJ/bit/m^4
d_0 (see [17])	87 m
length of a data packet	2000 bits
β Weibull distribution shape	3
$W_{u,v}$ bandwith of a link (u,v)	2.4 GHz
ρ Threshold of energy consumption ratio	0.01
Nodes' buffer size	100 data packets

Table.2 Transition state delay

	R_x (μs)	T_x (μs)	Sleep (μs)	Idle
R_x	-	1	194	-
T_x	1	-	194	-
Sleep	5	5	-	-
Idle	-	-	-	-

Table.3 Link quality parameters

	PRR	SNR (dBm)	SINR(dBm)	LQI
Excellent	1	]40; 60]	]30; 40]	]106; 255]
Good	]0.75; 1[	]25; 40]	]15; 30]	]102; 106]
Medium	]0.35; 0.75]	]15; 25]	]5; 15]	]80; 102]
Poor	[0; 0.35]	[0; 15]	[0; 5]	[0; 80]

Table.4 Transition state energy consumption

	R_x (mW)	T_x (mW)	Sleep (mW)	Idle
R_x	-	62	62	-
T_x	62	-	62	-
Sleep	1.4	1.4	-	1.4
Idle	-	-	1.4	-

5. Results and Discussions

In the following, we analyse and explain results of the experiments described in the previous section.

5.1. Average End-To-End Delay

Figure 7 shows the effect of number of nodes on packets delivery delay. These delays appear to increase according to network size when using CFMS and MSCP. In contrast, when MSCP is applied, delays increase only for networks with less than 500 sensors then decrease.

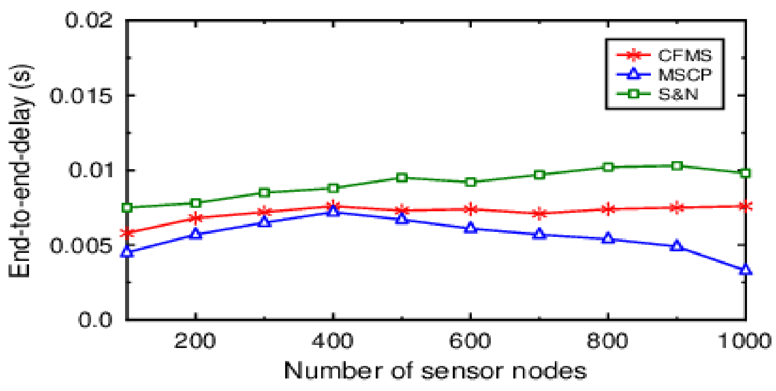


Figure.7 Number of sensor nodes vs. End-to-end delay

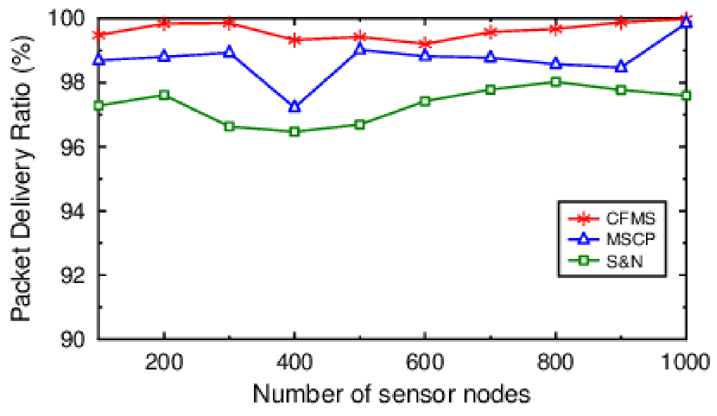


Figure.8 Number of sensor nodes vs. PDR

This is mainly due to the delay caused by the data aggregation process. However, it is noteworthy that when using CFMS delays slightly increase for sparse networks (with number of sensors > 500); but actually remain constant for dense ones. In other words, CFMS helps to better mitigate additional aggregation delay than S&N. Indeed, CFMS use the delay-based route selection scheme inspired from MSCP during its path-finding phase. Decisions implicitly consider both links state (via the SINR metric) and level of congestion of the intermediary nodes. Therefore, sensor nodes always choose the nearest sink considering lowest transmission and queuing delays. In contrast, S&N leverages only the Euclidean distance to the sinks.

5.2. Packet Delivery Ratio

Figure 8 suggests that the three evaluated protocols yield ratios higher than 90%. However, the values obtained with MSCP are around 98% while CFMS reaches values above 99% irrespective of the number of sensor nodes. This is also due to the path selection metric (SINR + congestion level) used by both CFMS and MSCP. Except that the latter protocol helps mitigate packet losses due to both collisions and nodes' congestion by scheduling data transmissions. Hence, the ratios close to 100%. Whereas the data delivery process of S&N is aimed to prevent congestions.

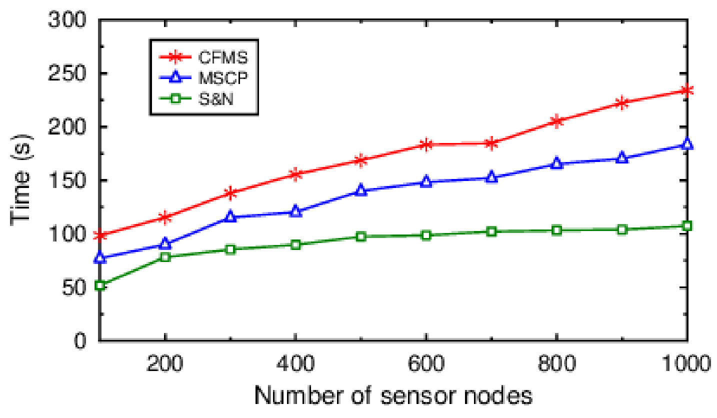


Figure.9 Number of sensor nodes vs. Network lifetime (Until a node dies)

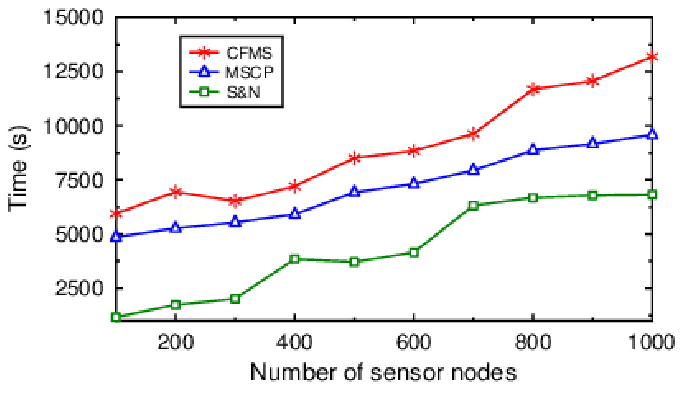


Figure.10 Number of sensor nodes vs. Network lifetime (Until a sink dies)

5.3. Network Lifetime

Figures 9 and 10 show that irrespective of the protocol used, network lifetime increases with its size; since high node degrees help provide more alternative routes. However, MSCP and CFMS yield the best durations. This is mainly due to their route selection metric that considers both link state and nodes' congestion; as discussed above, this scheme help mitigate packet losses hence energy consuming retransmissions. Additionally, these protocols favour CHs' rotation and sinks' sleep/wake scheduling. Furthermore, MSCP and CFMS significantly mitigates energy waste, by reducing the range and the number of signaling messages. However, the data aggregation and transmission scheduling process of CFMS helps mitigate congestions and collisions while further reducing the number of packets.

On the contrary, S&N yields the worst results since this protocol only leverages Euclidean distance and link node's congestion level to route packets to the sinks. No policy is explicitly applied in this protocol to mitigate energy wastes due to collisions or links' state variation.

6. CONCLUSIONS

In this paper we presented CFMS, a framework for convergecast in large-scale multi-sink WSNs. We combined clustering technique to path vector-based routing and duty cycling schemes for both sensor nodes and sinks. This solution was inspired from our previous work MSCP (Multi-Sink Convergecast Protocol). During route selection phase, transmission and queuing delays are still estimated by considering nodes' level of congestion and link interferences via a SINR (Signal-to-Interference-plus-Noise-Ratio)-based metric. However, we replaced nodes 'ability to individually transmit data by a collaborative scheme. This implied the design of two processes inside clusters as well as in sinks' one-hop neighbourhood. The first one is dedicated to data aggregation tree construction and the second one to transmissions scheduling. The duration of each transmission session is calculated according to gateways' congestion level in order to minimise collisions. The resulting fully-distributed and proactive strategy helped balance the load between sinks, mitigate message overhead and packet losses. Simulation results showed that CFMS enhances packet delivery ratio and network lifetime; though slightly increasing latency obtained with MSCP. CFMS outperforms S&N a related protocol recently proposed in the literature.

As a future work, we will focus on scenarios with mobile sinks and sensor nodes. We also plan to provide this solution with security, a specific data fusion module and an explicit congestion control scheme that considers packet priority as required by real time applications.

REFERENCES

[1] Kayhan Erciyes, (2019) *Case study: Environment monitoring by a wireless sensor network*, Computer Communications and Networks,.Springer International Publishing.

[2] Dionisis Kandris, Christos Nakas, Dimitrios Vomvas, and Grigorios Koulouras, (2020) "Applications of wireless sensor networks: An up-to-date survey", *Applied System Innovation*, Vol. 3, No.1, pp.14.

[3] Kamal Gulati, Raja Sarath Kumar Boddu, Dhiraj Kapila, Sunil L. Bangare, Neeraj Chandnani, and G. Saravanan, (2022) "A review paper on wireless sensor network techniques in internet of things (IoT)", *Materials Today: Proceedings*, Vol. 51, pp. 161–165.

[4] Asside Djedouboum, Ado Abba Ari, Abdelhak Gueroui, Alidou Mohamadou, and Zibouda Aliouat. (2018) "Big data collection in large-scale wireless sensor networks", *Sensors*, Vol 18, No 12:4474.

[5] Deepak Sharma, Amritesh Ojha, and Amol P. Bhondekar. "Heterogeneity consideration in wireless sensor networks routing algorithms: a review", *The Journal of Supercomputing*, Vol.75, No.5, pp. 2341–2394.

[6] Filipe Araujo, André Gomes, and Rui P. Rocha, (2020) "Towards optimal convergecastin wireless ad hoc networks", *Ad Hoc Networks,* Vol.107, No 102214.

[7] Jobish John, Gaurav S. Kasbekar, and Maryam Shojaei Baghini, (2021) "Maximum lifetime convergecast tree in wireless sensor networks. *Ad Hoc Networks*, Vol.120, No.102564.

[8] Arnab Roy, Joseph Lalnunfela Pachuau, and Anish Kumar Saha, (2021) "An overview of queuing delay and various delay based algorithms in networks", *Computing*, Vol.103, No.10, pp.2361–2399.

[9] Aysegul Tuysuz Erman, Thijs Mutter, Lodewijk van Hoesel, and Paul Havinga, (2009) *A cross-Layered communication protocol for load balancing in large scale multi-sink wireless sensor Networks*, International Symposium on Autonomous Decentralized Systems, IEEE.

[10] Tarunpreet Kaur and Dilip Kumar, (2019) "Computational intelligence-based energy efficient routing protocols with QoS assurance for wireless sensor networks: a survey", *International Journal of Wireless and Mobile Computing*, Vol.16, No.2 172.

[11] Yuan Chai and Xiao-Jun Zeng,"Delay-and interference-aware routing for wireless mesh network", (2020) *IEEE Systems Journal*, Vol. 14, No. 3, pp. 4119–4130.

[12] Yunquan Gao, Xiaoyong Li, Jirui Li, and Yali Gao, (2019) "Distributed and efficient minimum-data aggregation scheduling for multichannel wireless sensor networks", *IEEE Internet of Things Journ*al, Vol. 6, No.5, pp. 8482–8495.

[13] Louie Chan, Karina Gomez Chavez, Heiko Rudolph, and Akram Hourani, (2020) "Hierarchical routing protocols for wireless sensor network: a compressive survey", *Wireless Networks*, Vol. 6, No.5, pp.3291–3314.

[14] Shashi Bhushan, Manoj Kumar, Pramod Kumar, Thompson Stephan, Achyut Shankar, and Peide Liu. (2021) "FAJIT: a fuzzy-based data aggregation technique for energy efficiency in wireless Sensor Network", *Complex & Intelligent Systems*, Vol. 7, No.2, pp.997–1007.

[15] Shamim Yousefi, Hadis Karimipour, and Farnaz Derakhshan, (2021) "Data aggregation mechanisms on the internet of things: A systematic literature review", *Internet of Things*, Vol. 15, No.100427.

[16] Mandeep Kaur and Amit Munjal, (2020) "Data aggregation algorithms for wireless sensor network: A review", *Ad Hoc Networks*, Vol.10, No.102083.

[17] Wendi Beth Rabiner Heinzelman, Anathan.P. Chandrakasan, and Hari Balakrishnan (2002) "An application-specific protocol architecture for wireless microsensor networks", *IEEE Transactions on Wireless Communications*, Vol. 1, No. 4, pp. 660–670.

[18] Ikram Daanoune, Baghdad Abdennaceur, and Abdelhakim Ballouk. (2021) "A comprehensive survey on LEACH-based clustering routing protocols in wireless sensor networks". *Ad Hoc Networks*, Vol. 114 No.102409.

[19] Fanpyn Liu, (2021) "Majority decision aggregation with binarized data in wireless sensor networks", *Symmetry*, Vol. 13, No.9:1671.

[20] Amin Shahraki, Amir Taherkordi, Oystein Haugen, and Frank Eliassen, (2021) "A survey and future directions on clustering: From WSNs to IoT and modern networking paradigms", *IEEE Transactions on Network and Service Management,* Vol.18, No.2, pp. 2242–2274.

[21] G. Saranraj, K. Selvamani, and P. Malathi, (2021), "A novel data aggregation using multi objective based male lion optimization algorithm (DA-MOMLOA) in wireless sensor network", *Journal of Ambient Intelligence and Humanized Computing*, Vol.13, No.12, pp. 5645–5653.

[22] Sedigheh Sadat Sharifi and Hamid Barati, (2021) "A method for routing and data aggregating in cluster-based wireless sensor networks", *International Journal of Communication Systems*, Vol. 34, No. 7.

[23] Vani S. Badiger and T.S Ganashree, (2022) "Data aggregation scheme for IOT based wireless sensor network through optimal clustering method", *Measurement: Sensors*, Vol.24, No.100538.

[24] Bo Yu and Jianzhong Li, (2011) *Minimum-time aggregation scheduling in multisink sensor networks*, 2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, IEEE.

[25] Nitin Goyal, Mayank Dave, and Anil Kumar Verma, (2017) *Improved data aggregation for cluster based underwater wireless sensor networks*. Proceedings of the National Academy of Sciences, India Section A: Physical Sciences.

[26] Wahab Khan, Hua Wang, Muhammad Shahid Anwar, Muhammad Ayaz,Sadique Ahmad, and Inam Ullah, (2019) "A multi-layer cluster based energy efficient routing scheme for UWSNs", *IEEE Access*, Vol.7, pp. 77398–77410.

[27] Anagha Rajput and Vinoth Babu Kumaravelu , (2020) "Fuzzy-based clustering scheme with sink selection algorithm for monitoring applications of wireless sensor networks", *Arabian Journal for Science and Engineering*, Vol. 45, No.8, pp. 6601–6623.

[28] Amit Singh and A. Nagaraju, (2020) "Low latency and energy efficient routing aware network coding-based data transmission in multi-hop and multi-sink WSN", *Ad Hoc Networks*, Vol.107, No.102182.

[29] Aarti Jain and B.V.R. Reddy. (2014) *Sink as cluster head: An energy efficient clustering method for wireless sensor networks*, International Conference on Data Mining and Intelligent Computing (ICDMIC), IEEE.

[30] Gokou Hervé Fabrice Diédié, Boko Aka, and Michel Babri, (2018), "Energy-efficient ant colony-based k-hop clustering and transmission range assignment protocol for connectivity construction in dense wireless sensor networks", *Journal of Computer Science*, Vol.14, No. 3, pp. 376–395.

[31] Gokou Hervé Fabrice Diédié, Boko Aka, and Michel Babri, (2019) "Self-stabilising hybrid ,connectivity control protocol for WSNs", *IET Wireless Sensor Systems*, Vol. 9, No. 1, pp. 6–24.

[32] Mohamed Skander Daas, Salim Chikhi, and El-Bay Bourennane, (2021) "A dynamic multi-sink routing protocol for static and mobile self-organizing wireless networks: A routing protocol for internet of things". *Ad Hoc Networks*, Vol.117, No.102495.

[33] Sandeep Verma, Neetu Sood, and Ajay Kumar Sharma, (2019) "Genetic algorithm based optimized cluster head selection for single and multiple data sinks in heterogeneous wireless sensor network", Applied *Soft Computing*, Vol. 85, No.105788.

[34] Sampath Reddy Chada and Narsimha Gugulothu, (2022) "Secure and energy aware cluster based routing using trust centric-multiobjective black widow optimization for large scale WSN", *International Journal of Electrical and Computer Engineering Systems*, Vol. 13, No.7, pp. 525–532.

[35] Gokou Hervé Fabrice Diédié, Koigny Fabrice Kouassi, and Tchimou N'Takpé, (2022) *Multi-sink convergecast protocol for large scale wireless sensor networks.* In Signal, Image Processing and Embedded Systems Trends, Academy and Industry Research Collaboration Center (AIRCC).

[36] Varga, A.: 2022. Available at 'https://omnetpp.org', accessed on August 2022.

[37] Carlo Alberto Boano, Marco Antonio Zuniga, Thiemo Voigt, Andreas Willig, and Kay Romer, (2010) *The triangle metric: Fast link quality estimation for mobile wireless sensor networks*, 19th International Conference on Computer Communications and Networks, IEEE.

[38] Bruno Bougard, Francky Catthoor, Denis Clarke Daly, Anantha Chandrakasan, and Wim Dehaene, (2005) *Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives*, Design, Automation and Test in Europe, IEEE.

AUTHORS

Gokou Hervé Fabrice Diédié received his Ph.D. in Computer Science at Université Nangui Abrogoua (UNA) in Abidjan, Ivory Coast in September 2018. He received his M.S. in Computer Science from the same university in 2012. His research interests include distributed algorithms, self-organization in ad hoc networks and intelligent transportation systems. He is currently a Professor and Researcher in Computer Science and a member of Laboratory of Mathematics and Computer Science at Université Peleforo Gon Coulibaly of Korhogo, Ivory Coast.

Koigny Fabrice Kouassi received his M.S. in Computer Science at Université Nangui Abrogoua (UNA) in Abidjan, Ivory Coast in November 2021. He received his BS in Computer Science from the same university in 2018. His research interests include distributed algorithms and hoc networks. He is a member of and Researcher in Computer Science and a member of Laboratory of Mathematics and Computer Science at UNA.

Tchimou N'Takpé received his Ph.D. in Computer Science at Nancy University, France, in January 2009. He received his MS in Engineering from Ecole Nationale d'Electricité et de Mécanique at Nancy, France, in 2005. His research interests include distributed and parallel algorithms and Networking. He is currently a Professor and Researcher in Computer Science and a member of Laboratory of Mathematics and Computer Science at Université Niangui Abrogoua (UNA) in Abidjan, Ivory Coast.