MULTI-SINK CONVERGECAST PROTOCOL FOR LARGE SCALE 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, Côte d'Ivoire
[2]Laboratory of Mathematics and Computer Science, Université Nangui Abrogoua, Abidjan, Côte d'Ivoire

ABSTRACT

Wireless sensor nodes are designed to collect information about their immediate environment. Once gathered, such data are forwarded via a multi-hop communication pattern to a remote gateway, also known as the sink. This process referred to as the convergecast may often require several sinks in order to improve network efficiency and resilience. Provided that load among the latter nodes are well balanced and packet losses are mitigated. This paper aims to design such a protocol by combining clustering, path-vector routing and sinks' duty cycle scheduling schemes to help balance load and minimize message overhead. Simulation results proved that this solution outperforms DMS-RP (Dynamic Multi-Sink Routing Protocol), a recent state-of-the-art contribution, in terms of delay minimization, packet delivery and network lifetime enhancement.

KEYWORDS

Wireless Sensor Networks, Clustering, Convergecast, Multi-sink, Protocol, Scheduling.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) comprise small detection devices called sensor nodes and a central equipment referred to as the sink. The latter acts like a gateway to a third-party transportation network. WSNs' applications are encountered in domains like ecology, security, transportation, to industry, health etc. [1-3].

One of the most critical operations in such networks is convergecast, namely, the data gathering and forwarding process [4]. Typical WSNs architectures involve usage of a single sink. However, many applications including those deployed in harsh environments require a large number of sensors [5]. Therefore, the latter have to face long routing paths and congestions particularly in sink's neighbourhood [6]. The resulting energy wastes and packet losses quickly become detrimental to network lifetime and efficiency. A common technique to address these issues is to add extra sinks to the network [7]. Nevertheless, to really benefit this strategy, it is mandatory to distribute network load among sinks and to always find the optimal gateway for each sensor.

Unfortunately, most solutions found in the literature leverage assumptions and schemes that are restrictive for real-world applications; e.g. fixed size clusters, interference-free links etc.

In this paper, we propose to seamlessly combine clustering, path-vector routing and scheduling schemes to assign the proper sink to each sensor node while balancing network load. The resulting protocol helps minimize delay, mitigate packet losses and prolong network lifetime.

Our main contributions are as follows:

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

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

Convergecast schemes commonly use routing techniques where shortest paths from sensor nodes to sinks are constructed relying on different metrics (Euclidean distance, number of hops, link quality, nodes' energy level, number of possible retransmissions etc.) [8]. The resulting solutions can be reactive or proactive when the route discovery process is launched respectively on-demand or in advance. From a topological point of view, these protocols are generally classified as flat and hierarchical ones [9]. In the first category, solutions leverage well-known tree construction schemes such as the Shortest Path Tree (SPT), Minimum Spanning Tree (MST), and random tree (RDCT) to find relevant routes between sensor nodes and sinks. However, in large scale wireless sensor networks, where scalability is at stake, hierarchical techniques like clustering are indispensable. This scheme consists of grouping nodes around a leader referred to as the Cluster Head (CH) [10]. The latter may be chosen randomly or not; but often according to different criteria (energy level, degree, location, etc.) [11-15]. LEACH by Heinzelman *et al.* [16] is one of the oldest protocols in this category. This protocol has inspired a huge number of contributions in the past two decades [17]. Unfortunately, these solutions are generally single-sink oriented. Relatively few multi-sink solutions have been recently proposed.

Masdari and Naghiloo[18] suggested a distributed fuzzy logic-based sink selection scheme to cope with congestions. Each sink declares its load to their neighbours. The nearest uncongested sink is then selected. Nevertheless, this solution is dedicated to only one-hop networks. Rajput and Kumaravelu [19] used a similar approach. They applied in contrast, a fuzzy c-means algorithm to balance the size of clusters and to optimize the number of sinks and their locations in the region of interest. However, 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 to the distant base station.

Singh and Nagaraju [20-21] proposed to create routes by constructing a Wiener minimum spanning tree based on an Artificial Bee Colony optimization scheme. Regrettably, this strategy is not really scalable and requires knowing sinks' positions.

As for Wang and Su [22], they designed an algorithm based on breadth-first search strategy to create 2-disjoint path between each sensor node and the sinks using an edge colouring scheme. Nevertheless, this solution requires estimating Euclidean distance from a node to the sink. Calculation of this distance is costly.

Mukherjee *et al.* [23] used sensor nodes with 3-sector antennas to locate sinks. Based on the Euclidean distance from the sink, the deployment area is divided into 3 regions where nodes can respectively send data directly to the nearest sink (direct routing), via a one-hop (two-hop routing) or using multi-hop routing. Regrettably, this scheme does not cope with load balancing and needs specific sensor nodes.

With protocol GeoM, Leão and Felea [24] considered using a geographical-based solution that linearly combines Euclidean distance and energy level metrics to determine the next hop. Data are sent to available sinks using a two-step multicast communication scheme. The first and the second step consist of selecting respectively target sinks and candidate forwarders. After calculating the weighted metrics of all couples (candidates and sinks), intersecting decisions are detected and eliminated to avoid packets duplications. However, this solution does not scale since it is not really distributed. Additionally, it requires that every node is aware of both its own geographic position and that of all the sinks. Gathering such information is very costly.

Yildiz [25] proposed a Mixed Integer Programming (MIP) model aimed to maximize network lifetime for underwater WSNs. Unfortunately, this solution is not scalable since it is centralized and the proposed model leverages parameters difficult and costly to collect, such as total number of flows generated at a node and transferred over a specific link during the network, otal number of flows collected at a sink etc.

Fu *et al.* [26] advised a field-driven paradigm to make routing decisions. In this scheme network is abstracted into an electrostatic or magnetic field etc. where packets behave like objects that can be"attracted" by the sinks. This solution is the first field-driven protocol that uses multi-path (multi-hop) routing and considers impact of external environment. The path selection leverages different metrics referred to as the potential fields, namely the depth, residual energy and the environmental information of sensor nodes. These metrics help respectively estimate the Euclidean distance while preventing messages to traverse dangerous areas and nodes with poor energy level. This strategy introduces fault-tolerance in the network but struggles to balance sinks' load.

Liu *et al.* [27] developed a solution to schedule traffic and select optimal paths by considering delays and load balancing. Each sensor node tries to find the shortest path to each sink then records the results and the transmitted traffics obtained. The latter are finally scheduled to balance the load of sinks. This solution does not scale since during initial step, it requires each sensor node to construct a path to each sinks.

Onwuegbuzie *et al.* [28] suggested a three-step strategy. Firstly, the amount of traffic or network load/task to assign to each sink is estimated, and then whenever a task is to be executed, the real-time load of each sink is computed. Finally, the real-time load to total weight ratio of each participating sink is calculated. The current task is thus assigned to the sink having the least load to weight ratio. Regrettably, this scheme is centralized hence, not scalable.

Hassani *et al.* [29] presented a RPL-based [30] solution where hierarchical paths to select are evaluated via two linearly combined metrics namely, number of hops and the RSSI (Received Signal Strength Indicator). Each node selects a possible parent then computes its own rank from that of this neighbour using the combined metrics. The proper parent is chosen after comparing the expected transmission count for this neighbour metric to the obtained rank. However, in harsh environments, RSSI-based metrics are often a misleading.

Daas *et al.* [31] designed a distributed multi-hop cluster-based routing protocol where paths are selected using both hop count and link state via its SNR (Signal-to-Noise-Ratio). This protocol formally uses the *Sink-As-Cluster-Head* strategy to help balance the load of both sensor nodes and sinks. However, this solution requires fixed size clusters; thus, can hardly be applied to randomly deployed networks, e.g. to monitor phenomena in harsh environments.

3. PROPOSED SOLUTION

This section aims at presenting our scheme. We discuss our motivations, objectives and assumptions, then detail our solution referred to as MSCP (Multi-Sink Convergecast Protocol).

3.1. Motivations and Objectives

As discussed in previous sections, large scale wireless sensor networks often require using both clustering and deployment of several sinks. Surprisingly, few state-of-the-art cluster-based multi-sink convergecast protocols have been proposed so far. *Sink-As-Cluster-Head* is a technique often used in these solutions to enhance energy waste mitigation particularly in sinks' neighbourhood [32]. In this category, DMS-RP by Daas *et al.* [31] is one of the recent contributions that consider both link sate and asymmetry. However, this protocol ignores sinks load balancing, does not cope with network flooding and is only dedicated to deterministically deployed networks.

We believe that to further increase throughput, network lifetime, and its pervasiveness, a convergecast protocol, besides scalability, must consider interferences and link asymmetry, schedule duty cycles of both sensor nodes and sinks, be loop-free, while minimizing message overhead. This work is aimed to address these issues.

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.

3.3. Description

Let V denote the set of nodes in the deployment zone and E the set links between them. Formally, $E=\{(u,v)\in V\times V \mid d(u,v) < (r_u + r_v)\}$ where $d(u,v)$ is the Euclidean distance between nodes u and v ; while r_u and r_v respectively denote the communication ranges of u and v. MSCP consists of two stages, namely the clustering phase and the convergecast tree construction one. This protocol is a distributed asynchronous based scheme that uses message passing communication model.

Clustering-based convergecast protocols require a head node to be elected inside each cluster. This issue referred to as the leader election problem is a well-known topic in distributed systems design. In addition, clustering is a proved NP-complete problem [33].

3.3.1. Cluster Formation

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

We use a scheme similar to the one proposed for the clustering process inside the CONSTRUCT protocol [34]. 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, i.e. each sink creates a cluster in its *k*-hop neighbourhood; *k* is given as a parameter such as *k* = *max_hop_count.*

CHs have a limited service time of which duration 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 (see [34] for details). Figure 1 depicts the different statuses applicable to each node in a section of the network after the cluster formation phase.

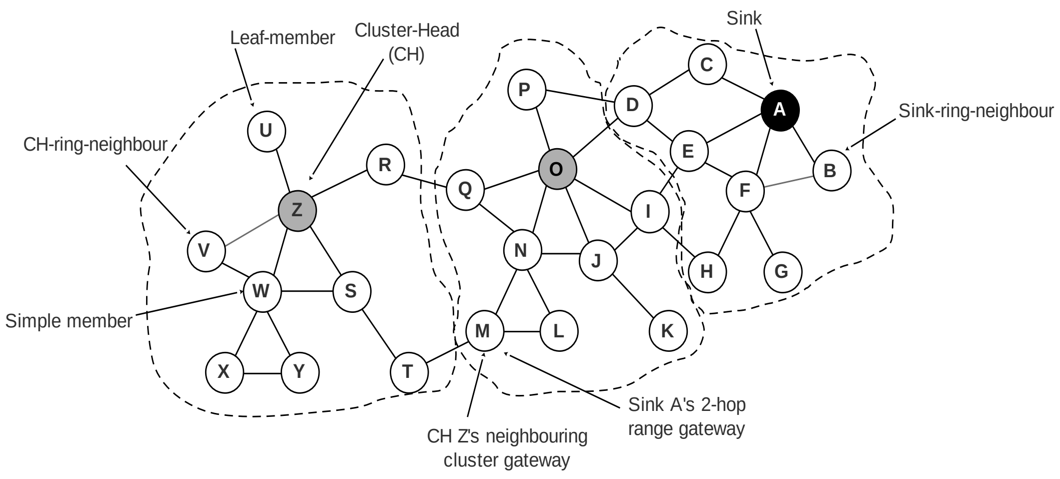


Figure 1. Statuses of nodes used by MSCP's clustering scheme

3.3.2. Intra-cluster convergecast sub-tree construction

After the creation of a new cluster, the CH has to construct the intra-cluster data forwarding sub-tree and interconnect it to the inter-cluster infrastructure. 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 (1); where rtt(.) denotes the round-trip-time (in seconds) experienced with node *v* during neighbour discovery. *N(u)* is node *u*'s neighbourhood.

$$t_{wait}(u) = \max\left\{\mathrm{rtt}(v); v \in N(u)\right\} \tag{1}$$

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 (2); 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.

$$\delta_{u,v} = \frac{L_{u,v}}{C_{u,v}} + S_u \tag{2}$$

$$C_{u,v} = W_{u,v} \times \log_2(1 + SINR_{u,v}) \tag{3}$$

As for, $C_{u,v}$ it is estimated using Equation (3); 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.

Dest	Next	Delay

Figure 2. Structure of convergecast table

Node v then increments the *total-delay* field with the estimated delay $\delta_{u,v}$. 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 1. TREE-ACK message contains its *I*D, 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.

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 content of the 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 2) by calculating the delay to each destination (i.e. sink) via Equation (4); 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.

$$\delta_{u,x} = \delta_{u,s} + \delta_{s,x} \tag{4}$$

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

After *TREE-timer* expiration, the CH must reply to each received TREE-ACK message, via the *first forwarder* node, by sending a CAST message where the 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 them. Before, sending this message, CH copies into the *cast_vector* field, content of the corresponding field of the received TREE-ACK.

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 analyzes content of the *cast_vector* field, extracts 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.

$$\delta_{u,x} = \delta_{s,x} - \delta_{s,u} \tag{5}$$

However, when a leaf-member or a CH-ring-neighbour (see Figure 1 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 used for TREE-REQ and TREE-ACK ones.

Each CAST-ACK message contains the list of sinks and delays to reach them with its *cast_vector* field embedding 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 (4) to update its convergecast table by calculating the delay $\delta_{u,x}$ to reach each sink x via the CH s; then extracts 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 (1)). 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 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 (2); 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 embed 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 (6) 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.

$$\xi = \frac{E_i - E_f}{E_i} \tag{6}$$

This idle state will last t_{sleep} seconds. The latter duration is determined using Equation (7) 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 $]0;1[$.

$$t_{sleep} = \frac{1}{\alpha} \times \ln(\frac{1}{R})^{\frac{1}{\beta}} \tag{7}$$

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 (8); where $\overline{t_i}$ denotes the mean time between wake-ups of the ith neighbouring sink while λ is given as a parameter.

$$t_{build} = \frac{\min(\overline{t_i}, \overline{t_{i+1}}, ..., \overline{t_n})}{\lambda}, \quad \lambda \geq 2 \tag{8}$$

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 the convergecast tree is constructed, each cluster mate can send its sensed data to the nearest available sink. To do so, the CH or the sink may send a dedicated schedule via CAST messages.

Note that, to mitigate energy waste, data often needs to be aggregated before being sent out of the cluster. In such a case, CH could information about this aggregation tree along with the possible scheduling scheme. The design of such a process is beyond the scope of this work. Nevertheless, any of the solutions that exist in the literature could be used.

Figure 2 depicts algorithm used by MSCP.

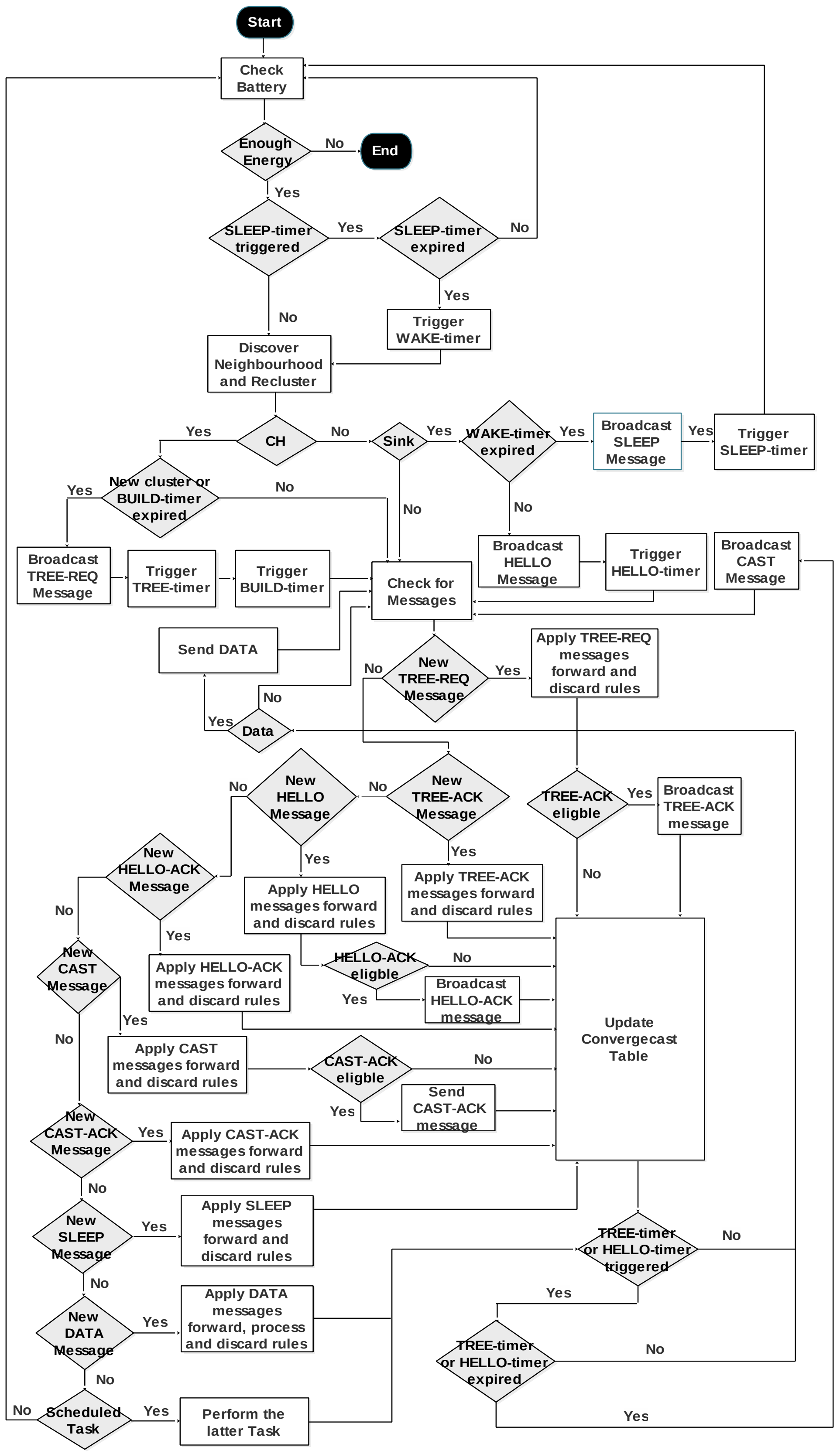


Figure 2. Flowchart of MSCP

4. EXPERIMENTAL SET-UP

In this section, we detail the extensive simulation campaign we carried out. Experiments were conducted using OMNeT++ 6.0 simulator [35]. We used the energy consumption model proposed by Heinzelman *et al.* [16]; the other parameters are summarized in Tables 1 - 4. The results were compared to those we obtained with DMS-RP (Dynamic Multi-Sink Routing Protocol) Multi-parent version + SNR (Signal Noise Ratio) proposed by Daas *et al.* [31].

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) [36]. Tables 2 and 4 are inspired by the specifications of the IEEE 802.15.4 standard [37].

We evaluated the ability of MSCP and DMS-RP to efficiently transfer data the sinks through three metrics, namely, the *average end-to-end delay*, the *packet delivery ratio*, and the *network lifetime* [31]. To this end, 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.

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 the network lifetime definition i.e., when a sensor or 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}	50 nJ/bit
e_{fs}	10 nJ/bit/m^2
e_{amp}	0.0013 nJ/bit/m^4
d_0	87 m
length of data	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

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

This section is aimed to analyse and explain results we have obtained from experiments we have described in the previous sections.

5.1. Average end-to-end delay

Figure 3 depicts the effect of number of nodes on packets delivery delay. Indeed, delays increase according to network size regardless of the evaluated protocol, for networks with less than 500 sensors then decrease.

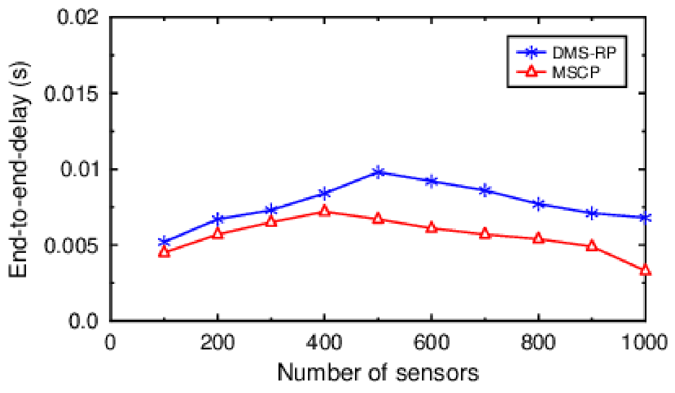


Figure 3. Number of sensors vs. End-to-end delay

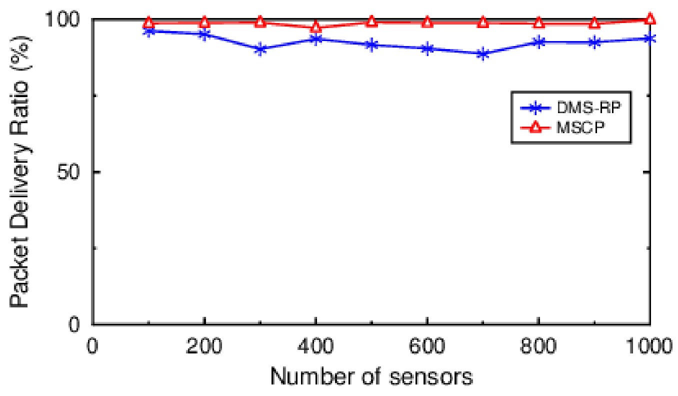


Figure 4. Number of sensors vs. PDR

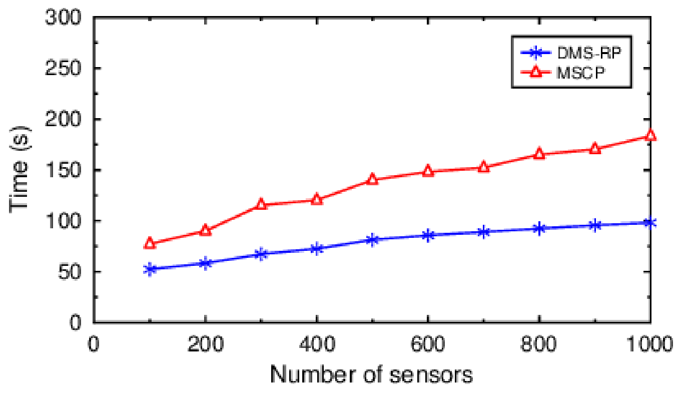
However, one can notice that MSCP provides the best results. This is due to the delay-based route selection process of MSCP. Indeed, sensors always choose the nearest sink considering lowest transmission and queuing delays. Unlike, DMS-RP, to make decisions MSCP leverages the interference (via the SINR metric) and the level of congestion of the intermediary nodes, especially in dense networks (number of sensors > 500) with high levels of interference.

5.2. Packet Delivery Ratio

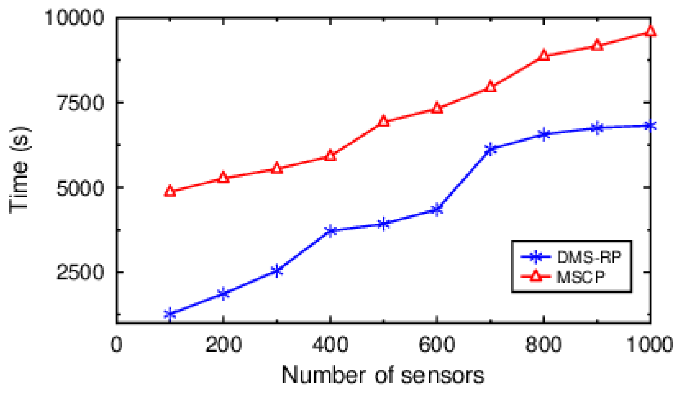
Figure 4 suggests that both protocols yield ratios higher than 90%. However, the values obtained by MSCP are around 98% despite the number of sensors. This is also due to its path update policy and to its SINR + congestion level-based link quality estimation scheme. Indeed, during the route selection phase, MSCP strives to keep only the links with the lowest transmission and queuing delays in both intra and inter-cluster topologies. This results in the minimization of packets losses.

5.3. Network lifetime

Figures 5 show that irrespective of the protocol used, network lifetime increases with its size; since high node degrees help to provide more alternative routes. However, MSCP yields the best results. This is mainly due to packet losses hence the retransmissions reduction scheme used during link selection phase (as discussed above); then to CHs' re-elections and sinks' duty-cycle scheduling. Furthermore, unlike DMS-RP, MSCP significantly mitigates message overhead hence energy waste, by reducing the range of its signaling messages and their number thanks to the forwarding rules applied. Additionally, the path-vectors piggybacking scheme used for CAST-ACK and TREE-REQ messages, helps avoid energy-consuming broadcasts.



a)



b)

Figure 5. Number of sensors vs. Network lifetime:
a) Until a node dies b) Until a sink dies

Using MSCP, only TREE-REQ and HELLO messages are respectively used to update the intra-cluster and inter-cluster routes. DMS-RP instead, uses messages that can unnecessary flood the network and are unable to prevent loops.

6. Conclusions

In this paper we presented MSCP, a convergecast protocol for large-scale multi-sink WSNs. We combined clustering technique to path vector-based routing and both sensor and sink scheduling schemes. The *Sink-As-CH* scheme is applied to save sinks' neighbours from being involved clustering elections. The resulting fully-distributed and proactive strategy helped balance the load between sinks, and mitigate message complexity. Unlike many state-of-the-art contributions, during route selection phase, transmission and queuing delays are estimated, considering interferences through a SINR (Signal-to-Interference-plus-Noise-Ratio)-based metric and nodes' level of congestion. Simulations show that MSCP provides high packet delivery ratios, lowest latencies and enhances network lifetime.

As a future work, we will formally focus on the impact of mobility of both sensors and sinks on the performance of MSCP. We also plan to provide this solution with security, data aggregation tree construction and traffic scheduling schemes.

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] 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.

[5] Essam H. Houssein, Mohammed R. Saad, Kashif Hussain, William Zhu, Hassan Shaban, and M. Hassaballah, (2020) “Optimal sink node placement in large scale wireless sensor networks based on harris hawk optimization algorithm”. IEEE Access, Vol.8 pp. 19381–19397.

[6] Jobish John, Gaurav S. Kasbekar, and Maryam Shojaei Baghini (202), Maximum lifetime convergecast tree in wireless sensor networks. Ad Hoc Networks, Vol.120, No.102564.

[7] 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, 2009 International Symposium on Autonomous Decentralized Systems, IEEE.

[8] 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.

[9] Louie Chan, Karina Gomez Chavez, Heiko Rudolph, and Akram Hourani, (2020) “Hierarchical routing protocols for wireless sensor network: a compressive survey”. Wireless Networks, Vol. 26, No.5, pp.:3291–3314.

[10] Piyush Rawat and Siddhartha Chauhan, (2021), “A survey on clustering protocols in wireless sensor network: taxonomy, comparison, and future scope”, Journal of Ambient Intelligence and Humanized Computing, Vol. 12, No. 7, pp. 181-190.

[11] Kowsalya. R and B. Rosiline Jeetha, (2019), “A survey on distributed clustering techniques for wireless sensor networks”, International Journal of Computer Sciences and Engineering, Vol.7, No. 3, pp. 404–409.

[12] Muhammad Noman Riaz, (2018), “Clustering algorithms of wireless sensor networks: A survey”, International Journal of Wireless and Microwave Technologies, Vol. 8, No. 4, pp.40–53.

[13] Anagha Rajput and Vinoth Babu Kumaravelu (2020), “A cluster leader selection algorithm to enhance the lifetime of scalable wireless sensor networks”, .Journal of Circuits, Systems and Computers, Vol.30, No. 03, 2150056.

[14] Amin Shahraki, Amir Taherkordi, Oystein Haugen, and Frank Eliassen, (2020) “Clustering objectives in wireless sensor networks: A survey and research direction analysis”, Computer Networks, Vol. 180, No. 107376.

[15] Jain Amutha, Sandeep Sharma, and Sanjay Kumar Sharma, (2021) “Strategies based on various aspects of clustering in wireless sensor networks using classical, optimization and machine learning techniques: Review, taxonomy, research findings, challenges and future directions”. Computer Science Review,Vol. 40, No. 100376, pp 1-43.

[16] 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.

[17] 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.

[18] Mohammad Masdari and Farah Naghiloo. (2017) “Fuzzy logic-based sink selection and load balancing in multi-sink wireless sensor networks”, Wireless Personal Communications, Vol. 97, No.2, pp.2713–2739.

[19] 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.

[20] Amit Singh and Aitha Nagaraju.(2018). Energy efficient optimal path based coded transmission for multi-sink and multi-hop WSN, Second International Conference on Green Computing and Internet of Things (ICGCIoT).IEEE.

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

[22] Fu Wang and Qinggang Su. (2018) A nonintersecting multipath routing protocol for wireless sensor networks, 5th International Conference on Systems and Informatics (ICSAI), IEEE.

[23] Sankar Mukherjee, Ruhul Amin, and G. P. Biswas , (2019) " Design of routing protocol for multisink based wireless sensor networks", Wireless Networks, Vol. 25, No. 7, pp. 4331–4347.

[24] Lucas Leão and Violeta Felea. (2019) Latency and network lifetime trade-off in geographic multicast routing for multi-sink wireless sensor networks. Mobile, Secure, and Programmable Networking, Springer International Publishing.

[25] Huseyin Ugur Yildiz (2019) Utilization of multi-sink architectures for lifetime maximization in underwater sensor networks, 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), IEEE.

[26] Xiuwen Fu, Yongsheng Yang, and Octavian Postolache, (2021) "Sustainable multipath routing protocol for multi-sink wireless sensor networks in harsh environments", .IEEE Transactions on Sustainable Computing, Vol. 6, No.1, pp. 168–181.

[27] Yang Liu, Lei Liu, Zhongmin Yan, and Jia Hu, (2021) The algorithm of multisource to multi-sink traffic scheduling. 17th International Conference on Mobility, Sensing and Networking (MSN). IEEE.

[28] Innocent Uzougbo Onwuegbuzie, Shukor Abd Razak, and Arafat Al-Dhaqm, (2021) Multi-sink load-balancing mechanism for wireless sensor networks, IEEE International Conference on Computing (ICOCO). IEEE.

[29] Abdelhadi Eloudrhiri Hassani, Aicha Sahel, and Abdelmajid Badri. (2021).Amalgamation of novel objective function and multi-sink solution for a reliable RPL in high traffic monitoring applications, Digital Technologies and Applications, Springer International Publishing.

[30] Muhammad Omer Farooq, Cormac J. Sreenan, Kenneth N. Brown, and Thomas Kunz, (2017) "Design and analysis of RPL objective functions for multigateway ad-hoc low-power and lossy networks", Ad Hoc Networks, Vol. 65, pp.78–90.

[31] 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.

[32] 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.

[33] Michael Randolph Garey, David Stifler Johnson, (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman,CA, USA.

[34] 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.

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

[36] 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, Proceedings of 19th International Conference on Computer Communications and Networks, IEEE.

[37] 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, Côte d'Ivoire in September 2018. He received his MS in Computer Science from the same university in 2012. His research interests include distributed algorithms, self-organization in ad hoc networks, 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, Korhogo Côte d'Ivoire.

Koigny Fabrice Kouassi received his MS in Computer Science at Université Nangui Abrogoua (UNA) in Abidjan, Côte d'Ivoire 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 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 in Abidjan, Côte d'Ivoire.