ConSerN: QoS-Aware programmable multitier architecture for dynamic IoT networks

Aurélien Chambon*, Abderrezak Rachedi†, Abderrahim Sahli‡ and Ahmed Mebarki§
*†LIGM (UMR8049), ‡COSYS/GRETTIA, §MSME (UMR8208)
Université Gustave Eiffel, F-77454
Marne-la-Vallée, France
Email: *aurelien.chambon@edu.univ-eiffel.fr, †abderrezak.rachedi@univ-eiffel.fr,
‡abderrahim.sahli@esiee.fr, §ahmed.mebarki@univ-eiffel.fr

***Abstract*—IoT networks allow various new types of services, such as Location-Based Services (LBS), requiring an end-to-end Quality of Service (QoS) without any interruption. However, sustaining these networks is difficult when the devices can move and vanish at any time, as they can be carried around, might run out of battery, or be turned off by users. In this paper, we propose a new programmable multitier architecture and a continuity of service protocol named ConSerN, which are based on the Software Defined Network (SDN) approach and Connected Dominating Sets (CDS) techniques. Unlike the existing solutions, the proposed mechanism considers a dynamic topology where nodes are not static from a mobility and topology point of view. Furthermore, as most of the applications using LBS need to consider the geographic position of the connected objects, we have added it in the selection process of the dynamic gateways. The performance evaluation was conducted using experimentation and emulation tools allowing scenario generations. We used three main metrics i.e. the quality of link, the number of dominant nodes and the amount of control messages. The obtained results show that the proposed solution benefits from a high number of gateways to achieve great performance in terms of overhead and quality of link.**

***Index Terms*—Internet Of Things, Software Defined Networks, Dynamic Topology, Quality of Service, Location-Based Services, multitier Architecture**

I. Introduction

Internet Of Things (IoT) networks have been a major research interest for the past few years. The wide variety of communication technologies available to these devices allow the creation of new types of services, such as Location-Based Services (LBS) [1]. LBS-oriented applications require defining and covering Zones of Interest (ZIs), and collecting and disseminating data through devices. The *Things* are connected devices such as: sensors, smartphones, wearable devices. They can be either mobile or static, turned on or switched off, in active or sleeping mode. In addition, these devices may be able to use only a limited set of the communication technologies available.

Some relevant architectures of IoT networks have been proposed in the literature with focus on redundancy and overload. These papers investigate an appropriate election mechanism on the server side to select specific devices as relayers.

However, the election process must take place often in order to handle the case of a disappearing elected device. Furthermore, routing tables have to be transmitted and updated frequently. This process is time consuming and generates an important overhead which impacts the quality of service.

978-1-6654-3540-6/22 © 2022 IEEE

To overcome this issue, we propose in this paper a new multitier architecture for LBS-oriented IoT networks. This architecture is based on the Software Defined Network (SDN) paradigm and a dynamic gateways selection mechanism. In addition, we provide a protocol to select gateways and maintain continuity of service, named ConSerN.

The SDN paradigm takes advantage of the different ways of transmissions IoT devices can handle. The protocol relies on specific CDS techniques to dynamically select the most relevant devices to act as gateways. The selection algorithm considers the geographic location of the devices according to predefined ZIs.

We evaluate the performance of ConSerN through emulation, using a server hosting different protocols and a scenario generator in which probability distributions for devices to appear or vanish can be configured. We show that our contribution benefits from more gateways to ensure QoS and reduce the amount of control messages to mitigate overhead.

The rest of the paper is organised as follows: section II presents a literature review from wireless sensors networks clustering techniques to connected dominating sets in mobile ad hoc networks, section III introduces the multitier architecture and its modules, section IV presents the ConSerN protocol, section V discusses the performance of the proposed solution. We then conclude in section VI.

II. Related works

A. Clustering Techniques

Clustering techniques are mainly used in Wireless Sensor Networks (WSN) which are a part of IoT networks. The research for network architectures led to the creation of many protocols where the devices are represented by nodes in a graph [2]. The goal of clustering techniques is to elect dominant nodes named cluster heads (CHs) that act as gateways. The cluster head selection protocol can be static (i.e. applied to a static topology of nodes) [3] or dynamic. In this work, we focus on dynamic protocols.

LEACH [4] selects CHs which gather, fuse and transmit data to the gateway in order to form clusters. However, this protocol considers that each node is equally relevant to be CH. Thus, CHs are not fixed and are elected randomly in order to distribute the load equally. To counter the fact that neither the location nor the residual energy of the nodes are taken into account in the election, many successors of the initial LEACH protocol were proposed in the literature.

For instance, LEACH-C [5] and HEED [6] are dynamic protocols taking into account many nodes parameters for the election of the CHs. However, centralizing data from every node requires a high number of control messages. Furthermore, ZIs are not taken into consideration to select the CHs. Finally, WSN architectures on which these protocols are applied do not take into account the constraints of a multitier architecture (in which several communication technologies coexist), nodes mobility, and the risk of the nodes to vanish in an unpredictable way.

B. Connected Dominating Sets algorithms

Connected Dominating Sets (CDS) algorithms are largely used in Mobile Ad hoc Networks (MANET), where the mobility of the nodes and their limited battery power are considered. The dynamic topology of the network induces issues in the routing process.

To overcome these problems, some solutions rely on CDS to build a virtual backbone (VB) of dominant nodes in the network. As the problem of finding the minimum CDS has been proven to be NP-hard [7], many algorithms were proposed in the literature to generate the VB [8].

Greedy algorithm is a well known example of CDS construction method, but it does not consider the location of the node as a parameter in the selection process [9]. BadZak algorithm focuses on Vehiculars Ad hoc Networks (VANET) and nodes' location to generate the VB according to predefined ZIs [10]. However, both rely on static nodes as dominant nodes for the VB. Therefore, these algorithms are not relevant for IoT networks since every node can roam or vanish.

III. Programmable multitier architecture

In this section, we present the programmable multitier architecture shown in Figure 1. This architecture is multitier in terms of communication technologies. The devices and the server exchange via cellular (3G/4G/5G) and non-cellular communication technologies (IEEE.802.11). Device-to-device communication is carried out via wireless technologies such as Bluetooth Low Energy (BLE), IEEE 802.11, IEEE 802.15.1 or IEEE 802.15.4. The architecture is programmable thanks to the SDN paradigm allowing centralized network management and distributed service data. Furthermore, the server can dynamically select the relevant communication technologies and parameters for each node. Such an approach allows devices that cannot communicate with the server to still access the network. The proposed architecture has three layers:

1) The server layer: The server is hosted in the cloud and acts as the coordinator of the dynamic network. Its first role is to select the most relevant set of dominant nodes. This selection is based on nodes parameters (battery left, topology connectivity) and their location regarding predefined ZIs.

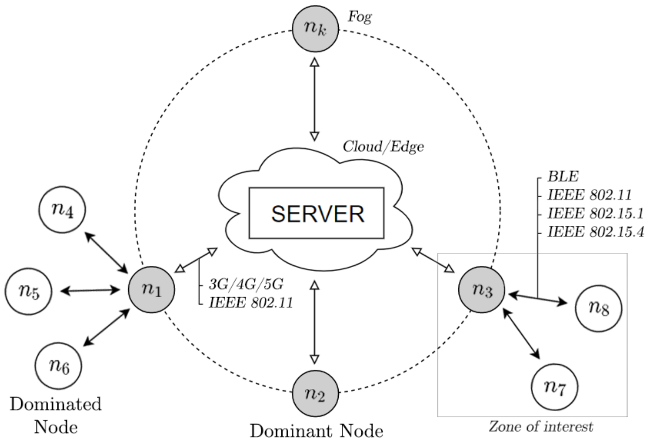

Fig. 1. Proposed architecture diagram

2) The dominant nodes layer: These nodes must be able to communicate directly with the server in order to act as relayers for top-to-bottom communications, and as gateways to dominated nodes for bottom-to-top communications. Each dominant node plays the intermediary role between the set of its dominated nodes and the server. Therefore, the server communicates with all the sets of dominated nodes via the dominant nodes layer.

3) The dominated nodes layer: The third layer is the dominated nodes layer. Each dominated node is assigned to a unique dominant node which can change in real-time. It is worth mentioning that a device that has no access to the Internet is necessarily dominated but can rely on a dominant node to connect to the network and access the service thanks to the SDN paradigm.

The server configures the communication technology used by a dominant node and its dominated nodes. Considering the case of IEEE 802.11 communication, once a dominant node is elected, the server conveys a designated channel of communication. Thus, the communications dominant-to-dominated and dominated-to-dominated are set to one dedicated channel in order to reduce interference with other nodes.

IV. Continuity of Service protocol

The goal of ConSerN is to provide continuity of service in a mobile and volatile network via selection of dominant nodes and network management. This process contains two main modules presented in the following sections, the selection algorithm and the continuity of service protocol.

A. Selection Algorithm

Let the undirected graph $G = (S, E)$ be the representation of the network, where $S = \{n_1, n_2, ..., n_N\}$ is the set of nodes, and E the set of links between nodes. Nodes are IoT devices considered graph neighbors if they are within communication range. We consider the graph of all dominated nodes of a dominant node d, denoted $G_d = (C_d, E_d)$. The links in this graph correspond to the signal power value

$S_{n_k \to n_l}, \forall (n_k, n_l) \in C_d^2$. Finally, let S' be the set of all dominant nodes where $S' \subseteq S$.

The selection algorithm (SA) takes a graph as an input and returns the most relevant nodes to be dominants. It is based on the first three phases of the BadZak Algorithm, as it considers the location of the nodes regarding Zones of Interests (ZIs) [10]. The first phase splits up the graph into subgraphs according to the predefined ZIs, and computes the maximal independent set. The second phase corresponds to the assignment of scores to nodes according to different parameters such as the topology (connectivity) and the location of the nodes in relation to the ZIs. Nodes are sorted by score to elect at least one dominant node per ZI. Finally, the third phase rebuilds the graph. In this paper, the fourth phase which generates the connected backbone is not needed.

B. Management protocol

In order to provide continuity of service, the protocol consists in executing the selection algorithm on dominated nodes subgraphs of a leaving dominant node, rather than on the whole graph. In the following sections, we present the protocol from both nodes and server side.

1) Nodes side: Firstly, when a node joins the network, it initializes by discovering near dominant neighbors. If there are one or many dominant nodes, the node hangs on to the one with the highest SINR value, if this value is above a predefined threshold T. Otherwise, the node becomes dominant as shown in Algorithm 1.

In order to maintain communication with the server, each dominant node d periodically sends a keep-alive message containing its location and the graph of its dominated neighbors G_d. If this graph is empty, it means that the dominant node is isolated. It will thus scan again for a nearby dominant node to hang on to. Dominant nodes can also request support from the server according to predefined conditions via $SupportNeeded()$ to balance load (Line 2-9 of Algorithm 2).

2) Server side: The server is continuously waiting for messages from the nodes. When a keep-alive message is received from a dominant node d, its data (i.e. its location and G_d) are updated if it already belongs to S', otherwise it is added to this list (Lines 2-8 of Algorithm 3).

When the server stops receiving the keep-alive message from a node, it is removed from S' and the selection process starts. The SA is executed on the subgraph of eligible nodes of the ousted dominant node. A node is considered eligible if it can exchange directly with the server. Finally, the server conveys the list of the elected dominant nodes. A similar

Algorithm 1 Discovering process for a node n

```
1:  $C_n \leftarrow ScanNeighbors()$          ▷ Scan for near neighbors
2:  $C'_n \leftarrow C_n \cap S'$             ▷ Filtering dominant nodes
3:  $score \leftarrow -\infty$
4:  for $d$ in $C'_n$ do                      ▷ Get highest SINR value
5:      if $max(score, T) < SINR_{n \to d}$ then
6:          $score \leftarrow SINR_{n \to d}$
7:          $dominant \leftarrow d$
8:      end if
9:  end for
10: if $score = -\infty$ then           ▷ No relevant dominant node nearby
11:     Set $n$ as dominant
12: else
13:     Set $dominant$ as the associated dominant node
14: end if
```

Algorithm 2 Listening and update process for a node n

```
1:  while true do
2:      if $n$ is dominant then
3:          Send Keep-Alive message to the server
4:          if $SupportNeeded()$ then
5:              Send support demand to the server
6:          end if
7:          if $C_n$ is empty then
8:              Run Algorithm 1
9:          end if
10:     else if $n$ is not eligible and its dominant left then
11:         Run Algorithm 1
12:     end if
13:     if list of dominants received from the server then
14:         if $n$ in the list of dominants received then
15:             $n$ becomes dominant
16:         else
17:             Run Algorithm 1
18:         end if
19:     end if
20: end while
```

Algorithm 3 Dominant node selection and network management process - Server Side

```
1:  while true do
2:      if Keep-Alive message received from node $n$ then
3:          if $n$ in $S'$ then
4:              Update the location of $n$ and the graph $G_n$
5:          else
6:              $S' \leftarrow S' \cup \{n\}$
7:          end if
8:      end if
9:      if Keep-Alive stopped from a node $n$ then
10:         $\mathcal{E}_n \leftarrow$ subgraph of all eligible nodes in $G_n$
11:         Start the SA on subgraph $\mathcal{E}_n$
12:         Send list of dominants to all nodes in $\mathcal{E}_n$
13:         Remove $n$ from $S'$
14:     end if
15:     if support demand received from a node $n$ then
16:         $\mathcal{E}_n \leftarrow$ subgraph of all eligible nodes in $G_n$
17:         $\mathcal{H}_n \leftarrow$ a random selection of half of $\mathcal{E}_n$
18:         Start the SA on subgraph $\mathcal{H}_n$
19:         Send list of dominants to all nodes in $\mathcal{H}_n$
20:     end if
21: end while
```

process is running when a dominant node asks for support: The SA is executed on a random selection of half of its dominated neighbors, and the node is not removed from S' (Lines 9-20 of Algorithm 3).

Eventually, the eligible nodes check whether they are in the list of dominant nodes sent by the server. If they are, they become dominant. Others and non-eligible nodes initialize again in order to hang on to the most relevant dominant node (Lines 7-16 of Algorithm 2).

V. Performance evaluation

A. Emulation and experimentation setup

Smartphones are used as mobile and volatile nodes, communicating with a Java server hosting different protocols. In addition, a python scenario generator emulating devices interacting with the server has been used in order to generate a high amount of nodes.

We consider the case of an outdoor free space, without interference other than that of the nodes. The signal power is calculated following a log-distance path loss model.

$$PL(d) = PL_r + 10\gamma\log_{10}\left(\frac{d}{r}\right) + X_\sigma \tag{1}$$

where $PL(d)$ is the path loss at a distance d, PL_r is the path loss at the range r, γ is the path loss exponent, and X_σ is a centered Gaussian random variable representing random attenuation of the signal. One-way communications are removed to keep the graph undirected.

According to [11], the Signal-to-Interference-plus-Noise Ratio of a node n_i towards a node n_d is set as:

$$SINR_{n_i \to n_d}[dB] = \frac{S_{n_i \to n_d}}{\mathcal{N}_{env} + \alpha I_{\overline{d}} + \beta I_d} \tag{2}$$

with

$$I_{\overline{d}} = \sum_{n \in C_{n_d} \setminus D_{n_d}} S_{n \to n_d}, \quad I_d = \sum_{n \in D_{n_d}} S_{n \to n_d}$$

where $S_{n_i \to n_d}$ is the signal power received by n_d from n_i, C_{n_d} is the set of neighbors of the node n_d and D_{n_d} is the set of dominated nodes of this node. $\mathcal{N}_{env}$ is the environment noise while α and β are both constants representing the impact of interference in the transmission environment. We consider a different impact of interference from the non-dominated neighbors as they are not communicating on the same channel as the dominant.

During the emulation, each node is randomly positioned in a predefined box area and assigned a speed $v \in [2, 20]$ m/s and destination in this same area. Furthermore, two probability distributions are considered that are P_a for nodes to appear and P_{n_i} for each node n_i to vanish.

1) Emulation parameters: They are shown in Table I. The geographic region is a free space area of 500 × 500 m^2 with one 70-metre radius ZI. The initial number of nodes is set to $N = 20$. We consider the case of IEEE 802.11 as a mean of transmission between nodes and thus the range of the nodes is set to 100 m. Poisson distributions are adopted for nodes to appear and to disappear.

TABLE I
EXPERIMENTATION PARAMETERS

Symbol	Parameter	Value
N	Node density	80 to 368 km^{-2}
r	Communication range	100 m
λ_a	Poisson distribution parameter	7
λ_d	Battery proportion parameter	4
n_{max}	Maximum number of dominated nodes	7
T	SINR threshold in discovering process	8 dB
PL_r	Path loss at the range r	-90 dBm
γ	Path loss exponent (free space)	2
σ^2	Path loss Gaussian variable variance	1
$\mathcal{N}_{env}$	Environmental noise	-90 dBm
α	Interference impact of non-dominated nodes	10^{-2}
β	Interference impact of dominated nodes	10^{-4}

$$P_a, P_{n_i} \sim \text{Pois}(\lambda_a), \text{Pois}(\lambda_d.(1 - B_{n_i})) \tag{3}$$

The parameter of the Poisson distribution for nodes to appear λ_a is set to 7. The parameter of the Poisson distribution for nodes to disappear is a function of the simulated battery of the node $B_{n_i} \in [0 - 1]$. λ_d is a proportion parameter set to 4. Thus, the appearance of nodes is more likely than their disappearance, causing an aggressive increase of the number of nodes.

Finally, dominant nodes will request support when they have more than n_{max} dominated neighbors. The parameter n_{max} is set to 7, as it is the limit allowed by devices in Bluetooth piconets.

2) Metrics: ConSerN was compared to other protocols according to several indicators.

- *Quality of link*: The average over all dominant nodes of their mean Signal-to-Interference-plus-Noise Ratio from their dominated nodes.

$$\mu_{SINR} = \frac{1}{|S'|} \sum_{n \in S'} \left[\frac{1}{|D_n|} \sum_{n_i \in D_n} SINR_{n_i \to n} \right] \tag{4}$$

- *Proportion of dominant nodes (scalability)*: We compute the proportion of dominant nodes in the network. Knowing that an isolated node has to be dominant, this metric is compared to the number of nodes being necessarily dominant.
- *Amount of control messages (overhead)*: Each message from the server to the nodes or from the nodes to the server is counted. This is an indicator of the overhead of the network.
- *Number of calls to the selection algorithm*: We monitor the number of calls and the number of nodes given as parameters, because this algorithm is expensive when executed on a large amount of nodes.

B. Analysis and discussions

1) Compared Protocols: Three protocols are considered and compared with ConSerN. The first one is referred as SA-T. This protocol consists in starting the SA each time the topology of the network changes, and communicating the list of dominant nodes to all nodes.

In the literature, the Trickle-Timer protocol is a well-known mechanism for optimizing control messages and quickly adapt to an unstable network [12]. The two remaining protocols are based on its updating process (RFC 6206). In these protocols, timers of variable lengths are considered. Their minimum duration is set to $I_{min} = 1$ (timestamp) while the maximum is $I_{max} = 2^4.I_{min}$. We set a *transmission* as the list of dominant nodes which should be the same for all modules. Thus, an *inconsistency* is when a dominant node is removed from this list. When a timer ends or an inconsistency is acknowledged, the SA is triggered.

In Trickle-A protocol, the server communicates with all the nodes whatever their role may be and the SA is executed on the whole graph. However, to be close to the ConSerN protocol strategy, Trickle-D protocol consists in the server communicating only with the dominant nodes, and the SA being executed on subgraphs of dominant nodes.

2) Results analysis: Figure 2 shows the context of the emulation, in which a specific seed has been set for the random generation to be the same for each protocol. We emulate an agressive appearance of nodes and evaluate our protocol on a dynamic topology with frequent arrivals and departures. Their number increases rapidly from 20 to 92, then decreases slowly to end at 66 nodes.

The computational cost of the SA is expensive when the number of nodes increases. In Figure 3, we present the average number of SA calls and the number of nodes considered for each protocol. The obtained results show that the Trickle-A and the SA-T call the SA around one time per timestamp on the whole network. Whereas Trickle-D calls it in average 0.28 times. As for ConSerN, it calls the SA up to 5 times per timestamp, with a mean at 1.35. But, in average, these selections are made on 1.9 nodes whereas they are made on 5.4 nodes using Trickle-D. Therefore, ConSerN optimizes the number of calls to the SA and the number of nodes considered. It reacts very quickly to topology changes and thus, guarantees service continuity for the rest of the network while the selection algorithm computes.

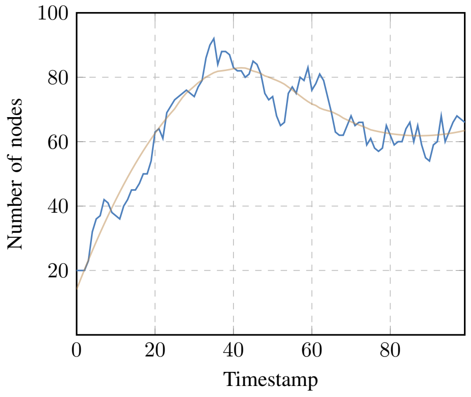

Fig. 2. Evolution of the number of nodes over time and its tendency using a Savitzky-Golay filter

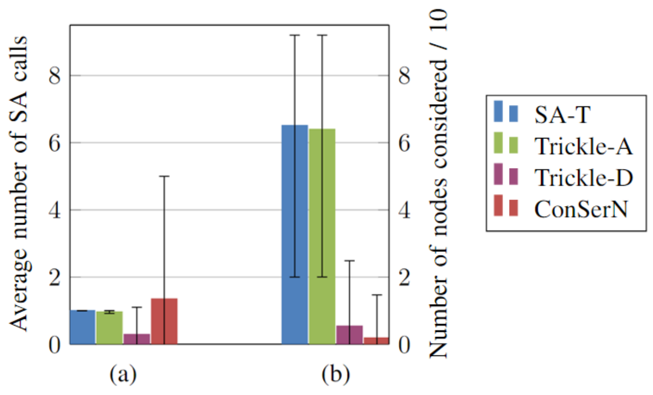

Fig. 3. Average number of calls to the selection algorithm during the emulation (a) and the number of nodes considered (b) according to the protocol

In Figure 4(a), we plot the proportion of dominant nodes according to the timestamp for different protocols. We show that the number of dominant nodes is approximately two times higher for ConSerN compared to other protocols. Since the selection algorithm is less called by the ConSerN method, it necessarily induces a higher proportion of dominant nodes.

In Figure 4(b), we present the amount of control messages according to the timestamps for each studied protocol. We notice that ConSerN divides by ~3 the number of control messages throughout the emulation, compared to SA-T and Trickle-A. Furthermore, ConSerN is more stable than Trickle-D regarding the amount of control messages. This avoids broadcast storms from the server where many nodes respond at once, inducing packet loss.

In Figure 4(c), we plot the quality of link according to the timestamp. Using ConSerN, it is improved by ~200% compared to SA-T and Trickle-A, and by ~75% against Trickle-D in average. Indeed, the protocol takes advantage of the greater amount of dominant nodes to optimize quality of link, and lower the number of control messages in the network.

In Figure 5, we show the network structure at the end of the emulation (100 timestamp) for each protocol. We can see that using ConSerN, a dominant node has one to four dominated nodes in order to keep a high quality of link. Moreover, the load balance per support request is clearly visible in the ZI.

VI. Conclusion

In this paper, we have proposed a new programmable multitier architecture based on the Software Defined Network (SDN) approach with a continuity of service protocol named ConSerN. The SDN architecture investigated is based on three layers: the server, dominant nodes, and dominated nodes. It is managed by a protocol adapted to a dynamic network topology where the nodes can appear, disappear and move.

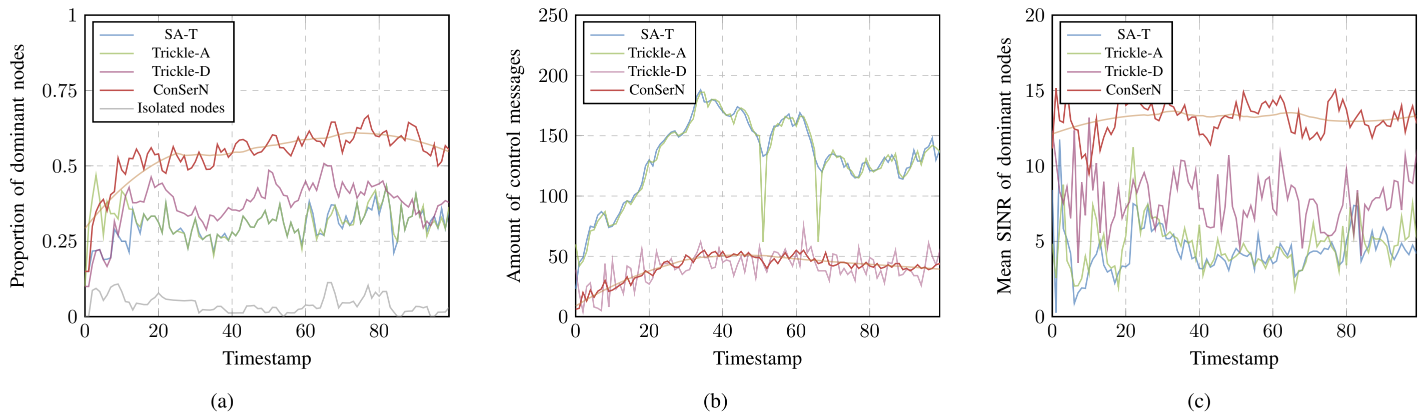

Fig. 4. Proportion of dominant nodes (a), amount of control messages (b) and quality of signal (c) over time according to the protocol. The tendencies of the ConSerN results are added using a Savitzky-Golay filter

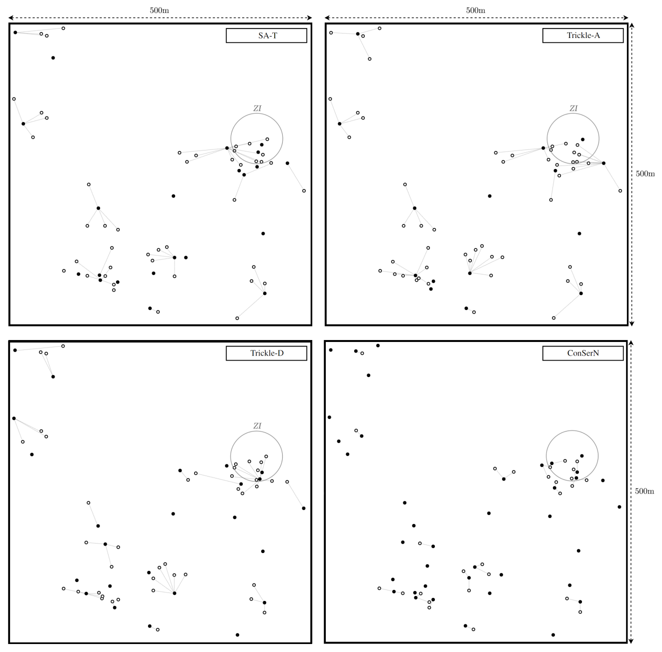

Fig. 5. Top view of the network structure at the end of the emulation for each protocol, solid and hollow circles are respectively dominant and dominated nodes

It elects a set of dominant nodes which act as gateways using a selection algorithm (SA) based on Connected Dominating Set (CDS) techniques. Unlike the existing solutions, the CDS algorithm considers zones of interest and is applied to a non-static topology. Furthermore, this protocol executes the SA on subgraphs of dominant nodes rather than on the whole network. We have performed a sensitivity analysis to compare our proposition with three other protocols based on the SA or inspired by the Trickle-Timer protocol. The obtained results show that ConSerN benefits from calling the SA several times on a reduced set of nodes. It can be drawn that even if ConSerN requires more dominant nodes as it relies less on the SA than other methods, it increases quality of link and reduces the amount of control messages. What's more, the proposed method allows devices that cannot communicate with the server to still access the network. Extension of the proposed work will introduce machine learning techniques to improve decision making on the resignation of dominant nodes.

References

[1] J. Fox, A. Donnellan, and L. Doumen, "The deployment of an iot network infrastructure, as a localised regional service," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pp. 319–324, 2019.

[2] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[3] S. K. Chaurasiya, T. Pal, and S. Das Bit, "An enhanced energy-efficient protocol with static clustering for wsn," in *The International Conference on Information Networking 2011 (ICOIN2011)*, pp. 58–63, 2011.

[4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pp. 10 pp. vol.2–, 2000.

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[6] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[7] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1, pp. 165–177, 1990.

[8] D. S. John Deva Prasanna, D. J. Aravindhar, and A. G. Amalanathan, "A survey on routing algorithms based on connected dominating sets in manets," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1206–1209, 2017.

[9] X. Cheng, M. Ding, D. Du, and X. Jia, "Virtual backbone construction in multihop ad hoc wireless networks," *Wireless Communications and Mobile Computing*, vol. 6, pp. 183 – 190, 03 2006.

[10] A. Rachedi and H. Badis, "Badzak: An hybrid architecture based on virtual backbone and software defined network for internet of vehicles," pp. 1–7, 05 2018.

[11] M. Haenggi, J. G. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti, "Stochastic geometry and random graphs for the analysis and design of wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 7, pp. 1029–1046, 2009.

[12] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A selfregulating algorithm for code propagation and maintenance in wireless sensor networks," *In Proceedings of the first USENIX/ACM symposium on networked systems design and implementation (NSDI)*, pp. 15–28, 01 2004.