RODENT: A FLEXIBLE TOPSIS BASED ROUTING PROTOCOL FOR MULTI-TECHNOLOGY DEVICES IN WIRELESS SENSOR NETWORKS

Brandon Foubert[1] and Nathalie Mitton[1]
[1]Inria, Lille, France

NOTE: Corresponding author: Brandon Foubert, brandon.foubert@inria.fr

Abstract – *Wireless Sensor Networks (WSN) are efficient tools for many use cases, such as environmental monitoring. However WSN deployment is sometimes limited by the characteristics of the Radio Access Technologies (RATs) they use. To overcome some of these limitations, we propose to leverage the use of a Multiple Technologies Network (MTN). What we refer to as MTN is a network composed of nodes which are able to use several RAT and communicating wirelessly through multi-hop paths. The management of the RAT and routes must be handled by the nodes themselves, in a local and distributed way, with a suitable communication protocol stack. Nodes may reach multiple neighbors over multiple RAT. Therefore, each stack's layer has to take the technologies' heterogeneity of the devices into account. In this article, we introduce our custom Routing Over Different Existing Network Technologies protocol (RODENT), designed for MTN. It enables dynamic (re)selection of the best route and RAT based on the data type and requirements that may evolve over time, potentially mixing each technology over a single path. RODENT relies on a multi-criteria route selection performed with a custom lightweight TOPSIS method. To assess RODENT's performances, we implemented a functional prototype on real WSN hardware, Pycom FiPy devices. Unlike related prototypes, ours has the advantage not to rely on specific infrastructure on the operator's side. Results show that RODENT enables energy savings, an increased coverage as well as multiple data requirements support.*

Keywords – heterogeneous, Pycom FiPy, routing, TOPSIS, WSN

1. INTRODUCTION

Wireless Sensor Networks (WSN) enable a remote monitoring of various metrics and many more use cases [1]. Such networks usually rely either on a medium distance Radio Access Technology (RAT) (*e.g.,* IEEE 802.15.4) and a multi-hop path routing or on a long distance RAT (*e.g.,* LoRaWAN) and a star topology. The latter simplifies the network structure and enables a wider coverage. When deployed, WSN usually use a single RAT shared by all nodes. Deployments are thus constrained by the limits of the chosen RAT, in terms of coverage and performance (throughput, energy consumption, costs, etc). For instance, the network of Sigfox, an operator-based RAT, provides long range communication (up to km) but is not available worldwide. Some RATs are even so constrained that they may not be able to comply with specific data requirements such as delay-intolerant data, high throughput or firmware over-the-air upgrade. Additionally, outdoor nodes have to bear the weather changes (*e.g.,* rain) which greatly impact the wireless links' quality.

Traditional WSN lack flexibility to support multiple use-cases. Many different RATs are available for WSN nowadays [2]. Different RATs come with different performances and capabilities. Multiple Technologies Networks (MTN) could overcome the aforementioned issues [3]. With several RATs built-in, the nodes' range of deployment would be extended, as nodes could switch from one RAT to another at each hop and relay data through multi-hop. An MTN's nodes would be able to select the best technology and route available. The choice would be based on the routes' availability and costs, in terms of energy, money, etc. If the environment changes, and the selected route's quality decreases, a node can dynamically select a better route and RAT. Nodes that support several data requirements (*e.g.,* temperature and video monitoring) can follow several paths accordingly. Network resiliency is increased, as in case of a RAT failure, a node can switch to an alternative technology.

Thus, nodes have to use specific methods to autonomously and dynamically choose which technology is the best suited depending on the data requirements and current context. This issue is known as Network Interface Selection (NIS). Several tools are available in the literature to tackle the NIS problem. Among them are the Multiple Attribute Decision-Making (MADM) methods. MADM methods provide a ranking of different alternatives based on their attributes and their associated weights. One of the most used and studied MADM methods is Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). Said simply, TOPSIS compares candidates based on their mathematical distances to two ideal positive and negative alternatives.

However, TOPSIS suffers from an issue known as rank reversal. A rank reversal happens when the ranking is modified following the removing of one of the alternatives under study. This can alter the quality of the ranking and lead to a sub-optimal NIS. In our case, this could outcome in too many useless and costly technology switches. Moreover, considering hardware constrained WSN nodes, TOPSIS computation is resource-intensive. This would

decrease the devices' lifetimes and may overload the devices' limited memories which leads to hardware failure. We address those issues in this paper, by proposing a lightweight TOPSIS-based NIS method optimized for WSN devices. Furthermore, our method simplifies TOPSIS computations and completely eliminates rank reversal by modifying the TOPSIS normalization algorithm. This results in less complexity and provides time and energy savings.

Currently available routing protocols are not suited for MTN. In this article, we introduce a novel Routing Over Different Existing Network Technologies protocol (RODENT) designed for MTN leveraging our custom TOPSIS method. Our contribution takes every RAT of each node into account for the route selection. Every node has a list of available links between itself and its neighbors. Links have associated costs and performances, in terms of delay, energy consumption etc. A node constructs its routes based on its links' values and the routes' values shared by its neighbors. Criteria for the best route depend on the use case and the requirements data has to meet (*e.g.,* data size, deadline).

RODENT is implemented and its performances are assessed through experimental evaluation. Results show that RODENT increases network flexibility and reliability, decreases energy consumption and enables better consideration of the data requirements while maintaining a good Packet Delivery Ratio (PDR). Compared to related work, RODENT offers a flexible and dynamic way to overcome WSN's limitations without the need of a dedicated infrastructure other than multi-RAT nodes.

The contributions of this paper can be summarized as follows:

- We designed a lightweight selection method for WSN based on TOPSIS, free of rank reversal which shows an improvement in the computation time of around 38%, which in turn results in energy savings, while the technology selection is equivalent to using the classic TOPSIS method in 82% of the experiments.
- We designed a multi-technology routing protocol for WSN based on our custom selection method. It is capable of handling multi-technology devices and selecting the best route and technologies for specific data requirements.
- We designed and developed an MTN prototype composed of Pycom FiPy devices running a custom implementation of RODENT.

The rest of this paper is organized as follows: Section 2 introduces the work related to MTN and TOPSIS. Section 3 presents the background about MADM and TOPSIS. Section 4 explains what issues have to be faced with TOPSIS. Section 5 details our lightweight TOPSIS method. Section 6 presents our experiments on the selection method and the results we have obtained. Section 7 exposes the network model and assumptions RODENT is based on. Section 8 details RODENT's inner workings. Section 9 presents the hardware used and firmware implemented for our MTN prototype. Section 10 introduces the experimental setup and scenario. Section 11 details the experiments' results. Section 12 concludes this article and lists future work.

2. RELATED WORK

Several works have been conducted to mitigate rank reversal in TOPSIS or to apply TOPSIS to NIS. To the best of our knowledge, only few works exist in the literature about multi-technology network. This section presents related work about TOPSIS and multi-RAT devices.

2.1 TOPSIS method

[4] proposes an iterative TOPSIS method, where TOPSIS is executed, then the worst alternative is removed from the ranking, and TOPSIS is re-executed, as long as there is more than one alternative in the ranking. The remaining one is selected as a communication technology. [5] combines TOPSIS with fuzzy logic, in order to improve how uncertain attributes are taken into account. [6] introduces alternative methods based on TOPSIS, but with different normalization algorithms using maximum and minimum values of the attributes. [7] compares several NIS methods applied to heterogeneous WSN. [8] introduces an original MADM method along with an in-depth analysis of TOPSIS. [9] proposes a new Service-based Interface Selection Scheme algorithm based on TOPSIS to enable NIS applied to vehicle-to-vehicle communications scenarios. [10] details a fast TOPSIS-based NIS technique for vertical handover in heterogeneous emergency communication systems.

Overall, those propositions reduce the probability of occurrence of rank reversal, but does not nullify it because the euclidean normalization is still used. Furthermore, some of the proposed modifications tend to increase the complexity of the TOPSIS method. This would increase the execution time of TOPSIS and in turn the energy consumption of the nodes, thus reducing their lifetime.

To the extent of the authors knowledge, no works has been conducted to propose a rank reversal free TOPSIS-based method for NIS specifically for energy constrained devices. Thus, in this in paper we introduce a lightweight TOPSIS-based NIS method that aims not only to reduce the complexity and energy consumption of TOPSIS, but also to completely eliminate rank reversal.

2.2 Multi-technology networks

The authors of [11] propose an IoT architecture for multi-RAT devices. This architecture is based on a network convergence layer in charge of the multi-RAT management in nodes, and a heterogeneous network controller on the network operator side. It also proposes a hardware platform for the nodes, a polling scheme as well as a compression scheme based on Static Context Header Compres-

Table 1 – MADM decision matrix.

	P_1 w_1	P_2 w_2	...	P_m w_m
A_1	x_{11}	x_{12}	...	x_{1m}
A_2	x_{21}	x_{22}	...	x_{2m}
...	...	...	...	...
A_n	x_{n1}	x_{n2}	...	x_{nm}

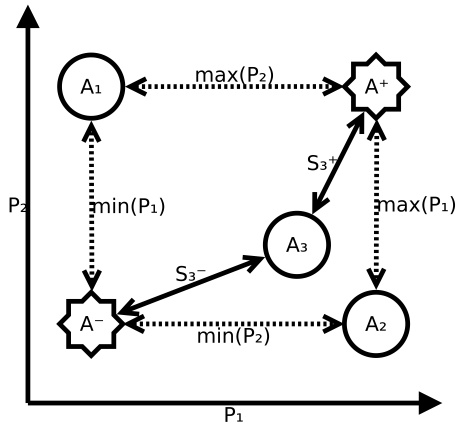

Fig. 1 – Representation of TOPSIS with three alternatives and two attributes.

sion (SCHC). This increases network flexibility, however it requires a specific virtual network operator. While the precedent work focuses on the device side, in [12] the authors propose a cloud-based virtual network operator for multi-modal LPWA networks. This operator takes care of the configuration and management of heterogeneous LPWAN equipment. Again this requires specific infrastructure on the operator side.

In [13], a green path selection inter-MAC selection protocol is detailed. This protocol allows path selection at the MAC layer while focusing on energy consumption and radio frequency minimization. However, it does not give any information about the routing layer. The article [14] presents the ORCHESTRA framework which manages real-time inter-technology handovers. It is based on a virtual MAC layer which coordinates the different layers from different technology with a unique MAC address. This work also focus on the link layer and not on the routing layer.

The aforementioned works increase WSN's flexibility. However several limitations are still present, such as the need of a dedicated infrastructure. In this article we propose a routing protocol adapted to MTN, which greatly increase WSN's capabilities while requesting only multi-RAT nodes.

3. TECHNOLOGY SELECTION BACKGROUND

Multi-technology devices have to autonomously select the best communication technology based on many factors. In the literature, several tools are available to perform this Network Interface Selection (NIS): utility and cost functions, Markov chains, fuzzy logic, game theory, data mining, Dempster-Shafer theory, to name a few. Particularly, Multi-Attribute Decision Making (MADM) methods [15] are commonly used for NIS. MADM methods are interesting as they rank several alternatives, based on their attributes as well as the relative importance associated to those attributes.

The problem can be modelled with a decision matrix as shown in Table 1. It is composed of $A = \{A_i \mid i = 1, 2, ..., n\}$ the set of the alternatives, $P = \{P_j \mid j = 1, 2, ..., m\}$ the set of the attributes and $W = \{w_j \mid j = 1, 2, ..., m\}$ the set of the weights associated to each attribute. Applied to NIS, A is the set of technologies, P the set of attributes associated to those and W the data requirements. The MADM methods take as input a decision matrix and output a ranking of the alternatives. Several MADM methods exist, the most known being: Simple Additive Weighting (SAW), Weighting Product (WP), Analytical Hierarchy Process (AHP) and Gray Relational Analysis (GRA).

One of the most used and studied methods is Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [16]. TOPSIS ranks alternatives depending on their relative mathematical distance to the ideal solution. The TOPSIS method runs the following steps:

1. The values x_{ij} of each attribute from the decision matrix (cf. Table 1) are normalized according to Equation *(1)*.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}} \qquad (1)$$

2. The normalized values r_{ij} are weighted according to Equation *(2)*.

$$v_{ij} = w_j r_{ij}, \ \sum_{j=1}^{m} w_j = 1 \qquad (2)$$

3. The positive and negative ideal alternatives A^+ and A^- are constructed according to Equation *(3)*.

$$\begin{aligned} A^+ &= [v_1^+ ... v_m^+] \\ A^- &= [v_1^- ... v_m^-] \end{aligned} \qquad (3)$$

4. The attribute values of the ideal alternatives are determined according to Equation *(4)* for upward attributes (*e.g.* range) or Equation *(5)* for downward attributes (*e.g.* latency).

$$\begin{aligned} v_j^+ &= Argmax\{v_{ij}, i = 1, ..., n\} \\ v_j^- &= Argmin\{v_{ij}, i = 1, ..., n\} \end{aligned} \qquad (4)$$

$$\begin{aligned} v_j^+ &= Argmin\{v_{ij}, i = 1, ..., n\} \\ v_j^- &= Argmax\{v_{ij}, i = 1, ..., n\} \end{aligned} \qquad (5)$$

Table 2 – Simple decision matrix.

	P_1	P_2	P_3
A_1	1.024537	7.828443	8.650221
A_2	4.226149	0.09865402	4.673396
A_3	8.026353	5.455392	2.536936
A_4	1.700537	1.398855	0.7656412

5. The distances between each alternative and the positive and negative ideal alternatives A^+ and A^- are computed according to Equation *(6)*.

$$S_i^+ = \sqrt{\sum_{j=1}^{m}(v_j^+ - v_{ij})^2} \qquad S_i^- = \sqrt{\sum_{j=1}^{m}(v_j^- - v_{ij})^2} \tag{6}$$

6. Finally, the relative closeness to the ideal solution is computed for each alternative according to Equation *(7)* and a ranking is established based on those values.

$$C_{TOPSIS} = \frac{S_i^-}{S_i^- + S_i^+} \tag{7}$$

When using TOPSIS for NIS, the technology with the highest value of C_{TOPSIS} is selected. A graphical representation of the TOPSIS method with three alternatives and two attributes is depicted in Fig. 1.

4. TOPSIS PROBLEM STATEMENT

TOPSIS is particularly interesting, as it grades alternatives based not only on the closeness from the best alternative but also on the distance from the worst one. However, TOPSIS suffers from an issue known as rank reversal that can happen when a non-optimal alternative is removed from the ranking. This can alter the quality and pertinence of the ranking. Rank reversal is an issue common to several MADM methods. With an ideal method, the ranking of alternatives should not be altered when another alternative is removed. The cause of rank reversal is the normalization algorithm. Indeed, the TOPSIS normalization (a.k.a. euclidean normalization) computes the normalized values for an attribute based on the values of all the other alternatives for that same attribute. Thus if set A changes, the result of Equation *(1)* also changes, which may modify the final ranking.

To clarify rank reversal let us consider an example. Table 2 represents a simple decision matrix randomly filled. Running TOPSIS on it outputs a ranking order corresponding to $[A_1, A_3, A_2, A_4]$. If the alternative A_4 was to be removed from the ranking (*e.g.* because of a broken link for example), it is expected that the ranking of the remaining alternatives should not be altered and therefore should correspond to $[A_1, A_3, A_2]$. However, running TOPSIS on Table 2 after removal of alternative A_4 outputs a ranking corresponding to $[A_3, A_1, A_2]$. This corresponds to a rank reversal. Applied to NIS, it means that the loss of the wireless link of technology A_4 would change the selected technology from A_1 to A_3. This would cause a technology switch which will require energy and does not bring any overall improvement.

It is to be noted that rank reversal is not a theoretical issue for multi-technology WSN devices. Actually, the wireless technologies' links' quality depends on many factors such as atmospheric and environmental conditions, which vary heavily across the year. This may results in broken links, thus removing a technology from the set of alternatives and potentially resulting in rank reversal, as seen in the previous example. The frequency of such events is entirely dependent on external factors and cannot be anticipated, thus links' quality has to be considered in the NIS process. Rank reversal could lead to the selection of a sub-optimal technology, on top of spending energy for switching between technologies.

A second issue posed by TOPSIS-based NIS on constrained devices is the complex computations that are required. The TOPSIS method as seen in Section 3 is based on computations that use numerous operations and memory accesses. WSN devices are generally hardware constrained, energy-limited and a repetitive execution of the TOPSIS method will have a considerable impact on the energy consumption of nodes. As an example, the Pycom FiPy's CPU [17] holds two cores that can go up to 240 MHz. A classic laptop CPU, *e.g.,* the Intel® Core™ i7-8650U, holds four cores that can go up to 4.20 GHz.

5. LIGHTWEIGHT TOPSIS FOR WSN

As stated in Section 4, the rank reversal issue is due to TOPSIS' normalization which computes normalized values based on all the other alternatives' values. Moreover, this normalization method is rather complex, and may increase the energy consumption of nodes.

Thus, we propose to use a simplified normalization method, which will not cause rank reversal and simplify the computations. Rank reversal happens because other alternatives are taken into account when computing normalized values. Thus, our proposition is to compute those values without taking into account other alternatives' values. Therefore, we need a stable normalization referential to measure our values against. We know that multi-technology devices have a fixed set of technologies available. Those are not supposed to change after deployment, and they have fixed maximum and minimum capabilities. We propose to use those maximum and minimum bounds as referential for our normalization.

5.1 Algorithm

That simplification takes the form of Algorithm 1, which replaces Equation *(1)* in the steps of our lightweight TOPSIS. Each value x_{ij} is normalized by being divided with the upper or the lower bound of its attribute j. Upward attributes' values are divided by their upper bound, while downward attributes divide their lower bound. The set

Algorithm 1 Lightweight normalization

Require: x_{ij} the raw value of each attribute j for each candidate i
  for each attribute P_j **do**
    if P_j is an upward attribute **then**
      B_j^+ is the upper bound of P_j
      $r_{ij} = \frac{x_{ij}}{B_j^+}$
    else if P_j is a downward attribute **then**
      B_j^- is the lower bound of P_j
      $r_{ij} = \frac{B_j^-}{x_{ij}}$
    end if
  end for
  return r_{ij} the normalized value of x_{ij}

$B = \{B_j^+, B_j^- \mid j = 1, 2, ..., m\}$ is composed of the upper and lower bounds of each attribute j, such that $\forall\, x \in B,\ 0 < x < +\infty$. B is stable, thus normalized values from the alternatives will not be altered by the removing of any other alternative. This completely eliminates rank reversal and reduces algorithmic complexity at once. Indeed, Equation *(1)* requires the computation of the denominator $\sqrt{\sum_{i=1}^{n} x_{ij}^2}$ for each value of j (for m attributes). This is not required with our bounded normalization and only the division between the bound and the value is computed. Knowing the fixed bounds allows us to simplify TOPSIS further: Equation *(3)* is used to establish the ideal positive and negative alternatives. Extreme values are found according to Equation *(4)* for upward attributes or Equation *(5)* for downward ones. Those operations require many comparisons. With bounded normalization, we can simplify the determination of the ideal alternatives: determination of A^+ and A^- is trivial, as the normalized maximum and minimum bounds of the attributes are respectively equal to 1 and 0. Thus, Equation *(4)* and Equation *(5)* can be simplified by Equation *(9)*. In turn, determination of the ideal alternatives according Equation *(3)* shows that these are static and shown in Equation *(8)*. Finally, distances computation according to Equation *(6)* can be simplified by Equation *(10)*. Indeed, as the ideal alternatives are known and static, we thus know that $v_j^+ = 1$ and $v_j^- = 0$.

Those simplifications reduce the complexity of the TOPSIS method. Moreover, as the normalization uses a stable referential, rank reversal probability is eliminated. Those modifications thus reduce the time required for execution, as we will see in Section 10.

$$\begin{aligned} A^+ &= [1...1] \\ A^- &= [0...0] \end{aligned} \tag{8}$$

$$\begin{aligned} v_j^+ &= 1 \\ v_j^- &= 0 \end{aligned} \tag{9}$$

Fig. 2 – FiPy board from Pycom [18].

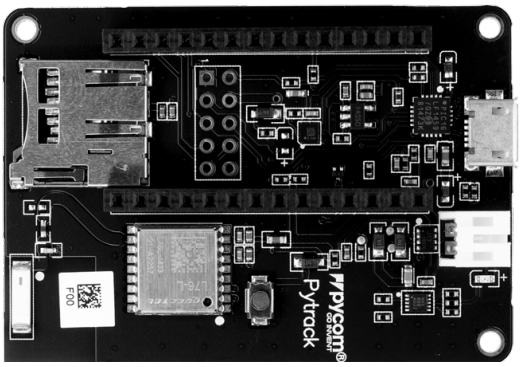

Fig. 3 – Pytrack sensor shield from Pycom [18].

$$\begin{aligned} S_i^+ &= \sqrt{\sum_{j=1}^{m} (1 - v_{ij})^2} \\ S_i^- &= \sqrt{\sum_{j=1}^{m} v_{ij}^2} \end{aligned} \tag{10}$$

5.2 Complexity

The reduced complexity of our algorithm can be assessed with an algorithmic complexity comparison. As big O notation is only pertinent for large inputs, we choose to quantify the number of operations spared with our method instead of classic TOPSIS. We consider one operation as one of the four basic arithmetic operations: addition, subtraction, multiplication and division. We also consider square root and value comparison as a single operation. This is just an estimation and is not exact as a square root is decomposed into multiple simpler operations when computed. However, as the exact decomposition is dependent on the hardware, it is irrelevant to assign a precise operation cost to a square root. Hereafter we consider n and m to be the dimensions of the decision matrix.

Firstly, Equation *(1)* requires at least $3nm$ operations, while using Algorithm 1 instead reduces it to nm operations. Replacing equations *(3)*, *(4)* and *(5)* by Equations *(8)* and *(9)* spares the cost of the min-max algorithm, thus $2(mn - 1)$ operations. Finally, using Equation *(10)* instead of Equation *(6)* spares nm operations. Our proposition thus spares a total of $5mn - 2$ operations.

6. SELECTION EXPERIMENTS & RESULTS

We implemented both algorithms in MicroPython on FiPy modules from Pycom, coupled with Pytrack expansion

Table 3 – Attributes' weights.

	Energy	Delay	Cost
$W_{monitoring}$	0.6	0.1	0.3
W_{alarm}	0.1	0.8	0.1

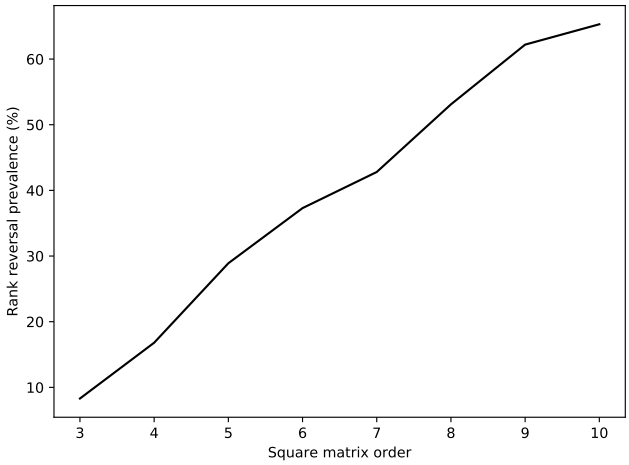

Fig. 4 – Rank reversal prevalence as a function of the decision matrix' size.

boards. Both are depicted in Fig. 2 and Fig. 3. Those devices offer five different wireless communication technologies, and provide hardware close to the one used in WSN. The available technologies on FiPy platform are WiFi, LoRa, Sigfox, LTE-M, NB-IoT and Bluetooth Low Energy. Each one comes with different performances, based on different metrics such as: energy consumption, economical cost, throughput, delay, loss rate, etc. Attributes of each technology are used to fill the decision matrix values x_{ij} used as input for the NIS algorithms. Weights associated to attributes are determined based on the data requirements. Table 3 shows an example set of weights that could be used: for regular monitoring data the weight and thus importance of the energy consumption will be higher. This would probably lead to an NIS of the best energy-efficient technology (*e.g.,* Sigfox). On the contrary, for an alarm the weight of delay will be higher, leading to an NIS of the fastest technology (*e.g.,* WiFi).

6.1 Rank reversal prevalence

We wanted to know how painful can be a rank reversal using TOPSIS for NIS. We ran experiments to quantify the prevalence of rank reversal using TOPSIS. The nodes execute the following steps: *i)* create a random matrix, *ii)* run TOPSIS on it and *iii)* compute the resulting ranking. Then we randomly remove one of the potential alternatives and the new ranking was computed. TOPSIS was run again on the matrix without the alternative removed from the ranking, and the resulting ranking was compared with the previous ranking. If the order of remaining options was different, then a rank reversal happened.
Results are highly dependent on the size of the matrices. Generally, the bigger the decision matrix, the more rank reversals as we can see in Fig. 4. Large matrices are not

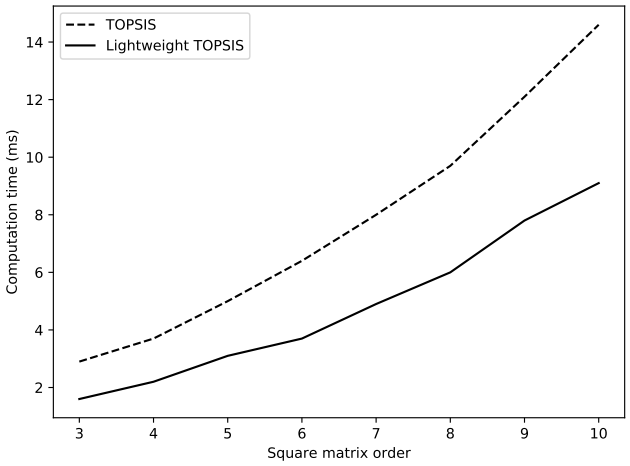

Fig. 5 – Classic and lightweight TOPSIS run times.

a current realistic representation of NIS in WSN. Multi-technology WSN nodes have several technologies available, but it is very unlikely that plain nodes carry hundreds of technologies. Similarly, technologies can have tens of attributes compared, but it is unlikely to be hundreds. Nonetheless, later, hardware will integrate more and more computing resources and communication technologies so our proposition will be able to scale with them. Still, we can see that even with small (5×5) matrices as we can obtain with FiPy modules, rank reversal happens approximately in 30% of the experiments. Rank reversal may cause useless technology switches, that are costly energy-wise. Larger matrices imply more frequent rank reversal, which emphasizes the need for a solution as ours. This is considerable if we assume TOPSIS to be run periodically to select the best technology after attributes or data requirements change.

6.2 Computation time

We compare the performance of a classic TOPSIS with our lightweight TOPSIS. We measure the time needed for the algorithms completion with the Timer library available for the FiPy as well as the similarity between the resulting NIS. It is worth noting that TOPSIS does not embed an objective comparison referential to estimate the quality of a ranking. However, TOPSIS is considered to produce a good quality ranking and is thus commonly used as a point of comparison. The obtained results are visible in Fig. 5.
We obtain a mean speed up of the computing time of 38%. At the same time, we still maintain a similarity with TOPSIS ranking in 82% of the experiments. Note that the ranking in the remaining 18% of the experiments cannot be qualified as worse for all cases since it mainly depends on the application and of what is expected or required. The ranking is only different from TOPSIS' ranking, which we used as a reference, but is not a ground truth. If we look at what we obtained when using a (5×5) matrix for a population of 7000 experiments with the results rounded to two decimal places, the mean execution time of the

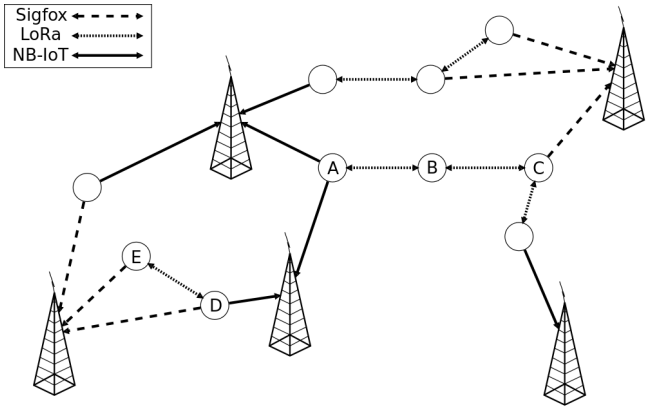

Fig. 6 – MTN example.

Table 4 – Example link matrix LM_D.

	Energy	*Money*	*Bit rate*
Sigfox BS	12	102	22
NB-IoT BS	151	87	174
Node E (LoRa)	37	0	72

classic TOPSIS is 4.79 ms, while the mean execution time of our lightweight TOPSIS is 2.96 ms. This means that a node could benefit from a mean time of 1.83 ms longer sleep periods between two TOPSIS executions. Based on the FiPy CPU data sheet [17], with a maximum CPU consumption of 68 mA and a power supply of 3.6 V, it would save up to approximately 448 µJ per TOPSIS run. Data sheets are notoriously optimistic, so in practice the energy savings could be even more significant. The standard deviation is of 0.05 ms, and the confidence intervals are $+/- 2.76 * 10^{-3}$ ms and $+/- 2.48 * 10^{-3}$ respectively for classic TOPSIS and for our lightweight TOPSIS, with a confidence level of 99.999%. Larger matrices offer similar results.

7. NETWORK MODEL & ASSUMPTIONS

We based the design of RODENT on a specific network model and assumptions made on the lower layers of the communication stack. In this section we describe this model and assumptions.

7.1 Network model

In WSN, the nodes usually follow one or multiple traffic patterns [19]. In this work, we assume that the nodes communicate in a convergecast pattern. Nodes forward packets exclusively to sink nodes. The nodes taking part in an MTN are heterogeneous in terms of RAT. We assume that the network is a connected graph where we consider every link from every node independently of their RAT *i.e.,* there can be several links between a single pair of nodes. Nodes can meet several data requirements (*e.g.,* monitoring, alarm, etc.), as long as those requirements are known by every node in the MTN. An MTN is depicted in Fig. 6. In this example, node B (N_B) measures temperature and is not in range of a Sigfox or NB-IoT base station. However, N_B can forward its data to N_A or N_C using LoRa. The latter can then offload N_B's data to a base station with a different RAT.

Table 5 – Example route matrix RM_D.

	Energy	*Money*	*Bit rate*	*Hops*
Sigfox BS	12	102	22	1
NB-IoT BS	151	87	174	1
Node E (LoRa)	49	102	94	2

Table 6 – Requirements vectors.

	Energy	Money	Bit rate
$RV_{monitoring}$	0.6	0.3	0.1
RV_{alarm}	0.1	0.1	0.8

7.2 Data requirements

RODENT aims to support multiple use cases. Nodes can have multiple purposes (*e.g.,* monitoring temperature, video recording). The data requirements differ depending on the use case. For instance, for video data, we need a RAT with a high bit rate to ensure low delay and jitter. For an alarm, we need a very short delay but not necessary high bandwidth. For regular and small monitoring data, the focus is on saving the nodes' energy. A single node can have multiple data requirements *e.g.,* sending regular monitoring data of a rainfall and an alarm in case of a flood. Thus the route selection must satisfy as best as possible all nodes' data requirements.

7.3 Assumptions on communication stack

This article focuses on the network layer, specifically routing. We assume that the other communication stack's layers are comprised of protocols suited to MTN and that the physical and link layers are able to assess the availability and quality of links toward the nodes' neighbors *i.e.,* nodes or base stations. We assume that this process is possible for every RAT. We consider that those layers are able to gather or estimate information about the cost and performances of each link *i.e.,* energy cost, bit rate, etc. Radio link quality estimation in WSN is a well studied subject [20].

RODENT takes a link matrix as input, to which we refer to as LM_x for node x. LM_x's size depends on multiple factors: the number of characteristics, the number of RAT available, and the number of x's neighbors. For example, N_D in Fig. 6 could have a link matrix LM_D such as the one in Table 4. LM_D is comprised of every available link between N_D and its neighbors, and the characteristics of those links.

We refer to the route matrix of node x as RM_x. For route selection, RM_x is composed of all the routes available for node x. RM_x's attributes are relative to the routes *e.g.,* the number of hops, expected transmission count or the total energy consumption. For example, N_D in Fig. 6 could have a route matrix RM_D such as the one in Table 5. TOPSIS takes as input a set of weights for each attribute. The

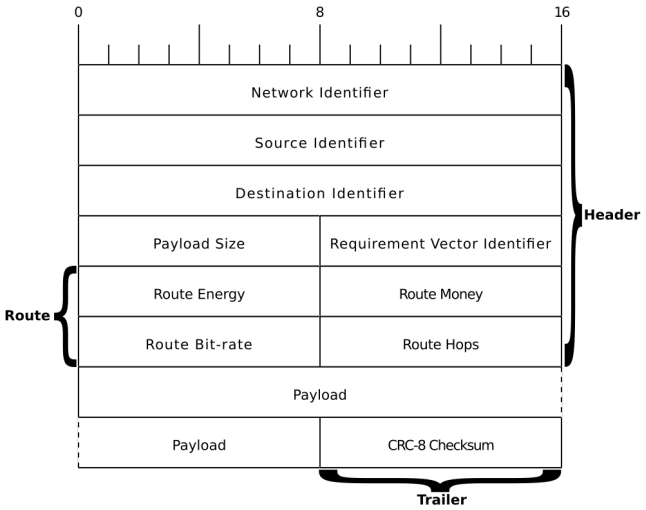

Fig. 7 – RODENT packet structure.

weights represent the importance of each attribute in the ranking process. We refer to a set of weights as a Requirements Vector (RV). RV_x is the requirements vector for use case x *e.g.,* $RV_{monitoring}$. For route selection *RV*'s values are set based on the data requirements that the node have to meet *e.g.,* prioritize speed over energy consumption, and such that $RV\,\{e_n \in RV \mid \sum_{n=1}^{|RV|} e_n = 1\}$. Example requirements vectors are depicted in Table 6.

8. ROUTING OPERATIONS

The distinctive feature of RODENT is to enable multi-RAT routes. Each route offers different cost and performances. In this section we detail RODENT's routing operations. The following notations are used further. Node i is referred to as N_i. Nodes that are in the vicinity of N_i are called neighbors. The set of N_i's neighbors is referred to as $NBR(i)$ and $NBR(i)_j$ is the node j such that $N_j \in NBR(i)$. A neighbor N_j of N_i has at least one link with N_i. For RAT x, such a link is referred to as L_{ij}^x. Consequently, the route from N_i that follows link L_{ij}^x is referred to as R_{ij}^x.

8.1 Overview

Let's consider the operations of N_D and N_E from Fig. 6 as an example. N_D boots without any knowledge of its surroundings. N_D's link layer scans the environment for every RAT and builds its link matrix LM_D as in Table 4. Based on LM_D, the network layer starts to build the route matrix RM_D. The direct links between N_D and the base stations are registered in RM_D as single-hop routes. N_E meanwhile does the same, and selects its only available route toward the Sigfox base station. N_E advertises its route which is received by N_D through their LoRa link L_{ED}^{LoRa}. N_D constructs its third route by adding the route's and link's costs. Here, we assume that the links' values between the Sigfox base station and N_D and N_E are similar. RM_D is then similar to Table 5. N_D then selects a best route for each of its *RV*. The selection is made independently of the RAT and based only on the routes' costs and performances. In this case and considering Table 6, the best route for $RV_{monitoring}$ is the one toward the Sigfox base station because low energy consumption is favored. The best route for RV_{alarm} is the one toward the NB-IoT base station because high bit rate is favored. N_D then starts to advertise and use its best routes.

8.2 Packet structure

RODENT packets' structure is depicted in Fig. 7. A packet is composed of three parts: *(i)* the header *(ii)* the payload *(iii)* the trailer. The header contains the required control data for RODENT. The *Network Identifier* is a two byte value shared by all nodes and is used to differentiate RODENT's communication. The *Source Identifier* is a two byte value corresponding to the packet's source node's unique ID. The *Destination Identifier* is a two byte value corresponding to the packet's destination node's unique ID. The *Payload Size* is a one byte value equal to the payload's size in bytes. The *Requirement Vector Identifier* is a one byte value which indicates the type (*i.e.,* use case) of the payload's data. The *Route* is a four byte array with $N_{Source\ Identifier}$'s best route's values *i.e.,* energy, money, bit rate and number of hops. The payload contains the data shared by the source. It is a series of n bytes with n equal to the header's *Payload Size* field. The trailer is a single byte carrying the *CRC8 Checksum* of the header and payload parts.

8.3 Route construction

Let's consider the operations of node i. N_i boots up and starts the construction of its route matrix RM_i. RODENT accesses two sets of data: the link matrix LM_i and the set of route shared by N_i's neighbors. N_i's first step is to check LM_i for any link toward a base station *e.g.,* a Sigfox antenna or a LoRaWAN gateway. Such links are turned to single hop routes based on the links values from LM_i. Routes are stored in RM_i. N_i's second step is to construct the routes passing through $NBR(i)$'s nodes. Let's consider the reception of a route from $NBR(i)_j$. N_i adds the received route's attributes to the attributes' values of the link L_{ij}^x. The resulting route R_{ij}^x is stored in RM_i.

8.4 Route selection

In classic WSN, route selection is trivial as the route with the lowest cost or rank is selected. In MTN, a route is a set of successive links, where each link may use a different RAT. Different RATs offer various performances and route selection in MTN has to take account of multiple criteria. We aim to support multiple use cases with different data requirements. Section 5 introduced RODENT's selection method. For node N_i, our lightweight TOPSIS takes as input the route matrix RM_i and a requirement vector RV_x relative to use case x. The selection outputs a ranking of the routes. The route coming out on top best fulfills the data requirements of use case x. For N_i, a best route BR_i^x

is selected for every use case x.

8.5 Route propagation

Route propagation occurs through two mechanisms: piggybacking and control packets. Piggybacking allows routes to be shared without dedicated transmissions. Considering a RODENT packet carrying a data payload of use case x, the header contains RV_x's id number and the best route BR_i^x. Wireless communications share a common medium. Thus, N_i overhears every packet from $NBR(i)$, which allows N_i to update RM_i opportunistically. If N_i stops overhearing route R_{ij}^x from its neighbor $NBR(i)_j$ *e.g.,* because N_j is down, R_{ij}^x will time out and will be removed from RM_i. To keep alive unused routes, $NBR(i)_j$ will send dedicated control packets. Control packets are regular packets with an empty payload.

9. PROTOCOL IMPLEMENTATION

Our implementation of RODENT is performed on Pycom FiPy devices [21]. The specificity of FiPy devices is that they offer five different RATs. These nodes take part in the MTN and offload data to WiFi and LoRa base stations (BS). The hardware and firmware used are detailed in this section.

9.1 Hardware

Pycom FiPy nodes are composed of WSN hardware: wireless RAT, ESP32 CPU, little memory available which allows ultra–low power usage. The available RATs are WiFi, LoRa, Sigfox, LTE-M, NB-IoT and Bluetooth Low Energy (BLE). Each RAT comes with different performances in terms of energy consumption, economical cost, bit rate, etc. RODENT performs route selection based on these characteristics. FiPy nodes are coupled with Pytrack sensor shields which provide an accelerometer, a GPS and a micro-USB port.
The LoRa BS is a B-L072Z-LRWAN1 board [22]. The WiFi BS is an Edimax EW-7811Un dongle [23] connected to the main computer. A Trip Lite U223-007 (7-Port USB Hub) is used to connect every device. The main computer is a Dell Latitude 5590. It powers devices, collects and analyses results.

9.2 Firmware

A port of MicroPython available as firmware for the FiPy allowed us to implement RODENT in Python. Upon boot, a node N_i computes its unique ID. Based on LM_i it boots up the needed RAT and constructs routes. The node is then locked up in the main loop: *i)* select best route for each RV_x, *ii)* add next payload to transmission buffer *iii)* send every payload in buffer. Neighbor's routes are added in RM_i upon reception. Neighbor's payloads are appended in the transmission buffer. Nodes print on the serial port the characteristics of the packets sent. Upon the Pytrack's button press, nodes switch between the two *RV* implemented: $RV_{monitoring}$ and RV_{alarm}.

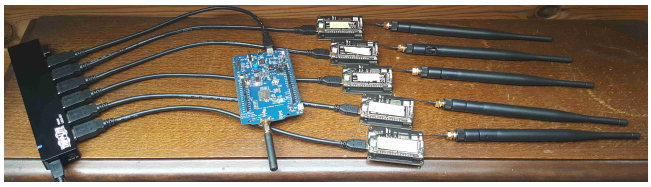

Fig. 8 – Experimental setup.

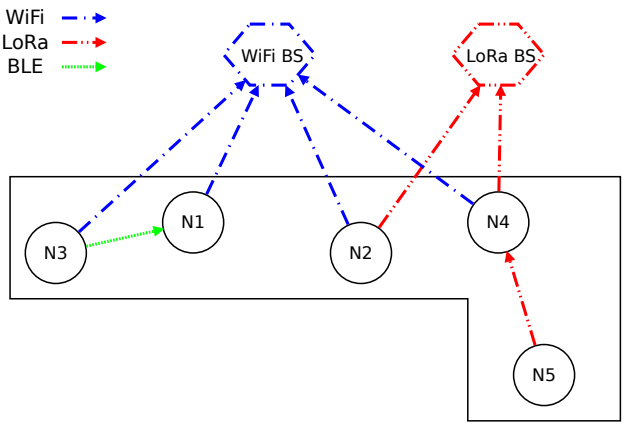

Fig. 9 – Farm monitoring scenario.

The LoRa BS's firmware is implemented in C. It listens constantly for LoRa transmissions. Upon reception of a RODENT packet, it is unpacked and its characteristics are printed on the serial port. The WiFi BS is coded in Python. It listens for RODENT WiFi transmissions, unpacks them and prints characteristics on stdout.

10. RODENT EXPERIMENTS

To assess the performances of RODENT, we run experiments on real hardware. We configured the nodes to follow a specific scenario and measured the results. The experimental setup and scenario are presented in this section.

10.1 Setup

The aforementioned devices in Section 9 are connected to the main computer through the USB hub. Every node and BS are powered at the same time and boot up immediately. As we can see in Fig. 8, every device is laying very close to each other. The main computer reads the stdout of the WiFi BS and the serial ports of the nodes and LoRa BS. Results are then computed offline, post-experiment.

10.2 Scenario

We simulate a farm monitoring use case. Smart agriculture can help farmers in their everyday life, but farms are often an unfriendly environment for wireless sensors (large rural areas, tall crops...). MTN eases the technical difficulties by offering nodes multiple possibilities of communication (operator based networks, personal net-

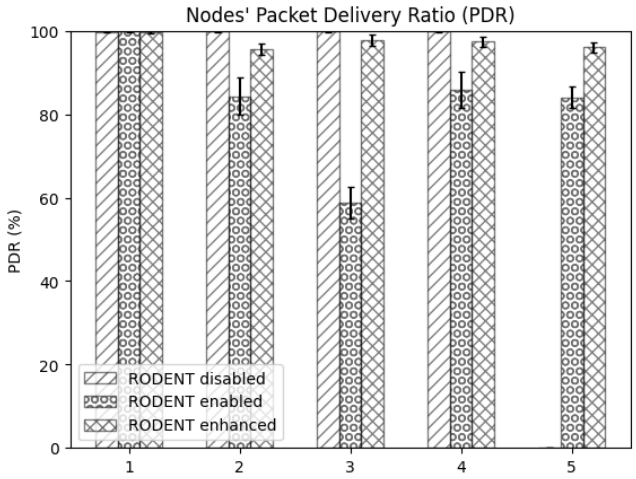

Fig. 10 – Packet Delivery Ratio per node.

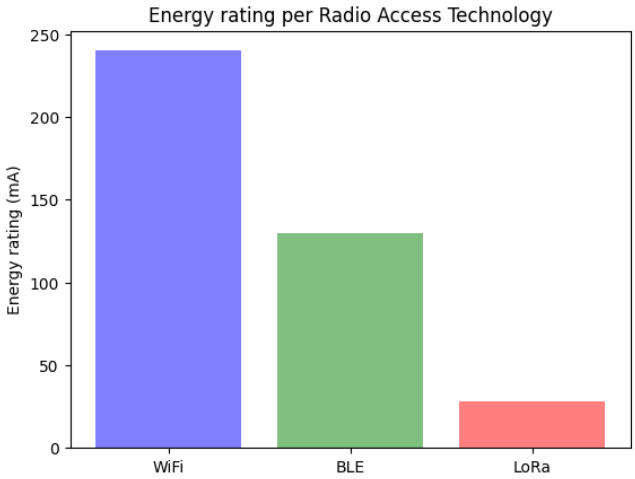

Fig. 11 – Energy consumption per RAT.

works, multi-hop networks...). In our scenario, nodes are deployed throughout a field used for cultivating crops. The simulated setup is illustrated in Fig. 9. Five nodes monitor environmental metrics useful for farmers. Nodes have to offload numerical data on a regular basis while saving up power. They may have to send an alarm if a metric becomes off chart, putting the crops at risk (*e.g.,* temperature).

Out of the five RATs available on FiPy, we are using WiFi, LoRa and BLE in this scenario. Sigfox and LTE-M/NB-IoT are not open technologies, so we could not use them directly. LoRa and BLE links are more interesting in terms of energetic savings than WiFi. Each node (Nx) is in a different situation. $N1$ is the control node, it only has a WiFi link with the WiFi BS. $N2$ can reach the WiFi BS and benefits from the LoRa link when RODENT is active. $N3$ has to choose between reaching the WiFi BS directly at a high energy cost or forwarding its data to its neighbor $N1$ via a BLE link. $N4$ needs to be able to send regular monitoring data as well as alarms, via WiFi or LoRa. $N5$ is an isolated node, deployed too far away to directly communicate with the WiFi BS. Farms are usually located in wide rural environments, unfriendly to wireless waves because of tall crops (*e.g.,* corn). Thus white zones and isolated nodes are common. Using RODENT, $N5$ can forward its data to its neighbor $N4$ using LoRa.

We run three types of experiments. First, RODENT is not active and nodes only use WiFi links, depicted in blue in Fig. 9. Second, RODENT is active, which allows nodes to switch to LoRa and BLE links, depicted in red and green in Fig. 9. Third, RODENT is active, each LoRa message is sent two times and each BLE message is sent three times, which increases the network's reliability. A video of an experiment running is available online[1].

[1] `http://chercheurs.lille.inria.fr/bfoubert/ressources/rodent.mp4`

11. EXPERIMENTAL RESULTS

Topology and Packet Delivery Ratio (PDR) are measured. Nodes transmit at an interval randomly picked in $[2; 4]$ seconds. We consider a population of 20 experiments lasting 10 minutes each. A small population is sufficient because of the low standard deviation. Longer experiments are not relevant because the network stabilizes after a few messages have been exchanged. We do not directly compare RODENT's results to related works as the heavy difference between proposals makes it irrelevant, and increased flexibility cannot be measured. In this section we present the results obtained.

11.1 Topology

With the use of RODENT, the MTN's topology changes. $N1$ does not change its link because it can only reach the WiFi BS. $N2$ uses the LoRa link instead of the WiFi link, because it costs less energy. $N3$ decides to use the BLE link to offload its data to $N1$, which in turn forwards it to the WiFi BS. $N4$ offloads its monitoring data to the LoRa BS, to reduce energy consumption compared to WiFi. It can still use the WiFi link to forward alarms that needs a quicker RAT at the expense of a higher energy cost. $N5$ is not isolated anymore, as it forwards its data to $N4$ through LoRa which will offload it to the LoRa BS in turn.

11.2 Packet Delivery Ratio

The Packet Delivery Ratio (PDR) is the ratio between the total packets received and the total packets sent. The PDR of every node taking part in the MTN is depicted in Fig. 10 along with its standard deviation. $N1$'s PDR does not change, as its route remains the same. Without RODENT, $N5$'s PDR is null as the node is isolated and cannot offload a single data packet. The PDR of $N2$, $N4$ and $N5$ is around 80% with RODENT which allows them to use LoRa. It is not the same as WiFi because of frequent collisions, as nodes do not use a proper MAC. $N3$'s PDR is around 60% with RODENT. The node forwards its data

through BLE to $N1$. We achieved BLE raw transmissions through the use of single BLE advertisements, hence the packet losses. With the enhanced RODENT, we can see a better PDR for all nodes, close to the one obtained with only WiFi.

11.3 Energy consumption

Physical measurement of the Pycom FiPy's energy consumption is hazardous since it suffers from design problems which lead to erroneous measurements [24]. We choose to stick to the energy ratings given in the components data sheets [17, 21] to get a general idea; these are shown in Fig. 11. Compared to WiFi, BLE needs approximately half-less current and LoRa a tenth. With the Pycom FiPy's CPU, WiFi and BLE offers the same bit rate. LoRa's bit rate is much slower leading to longer transmission for the same amount of data. WiFi and BLE require a heavier traffic control than LoRa does, which allows LoRa to consume less energy. Thus, we can assume that RODENT enables significant energy savings.

12. CONCLUSION

WSN deployments are limited by the coverage and performances of the devices' RAT. The use of several RATs in an MTN allows these shortcomings to be overcome. In this article, we introduced the novel Routing Over Different Existing Network Technologies protocol (RODENT). RODENT is based on a lightweight TOPSIS method that reduces the complexity of the computations and eliminates rank reversal issues, lessening computation time of about 38%. The resulting network interface selection is still similar to the one obtained using classic TOPSIS in 82% of the experiments.

RODENT is designed for routing in MTN and enables the use of multi-technology routes. We demonstrate the feasibility and utility of MTN with a prototype network based on a custom implementation of RODENT. Results show that RODENT increase flexibility, reliability, energy savings and maintains a good PDR.

For future work, we plan to precisely measure energy savings and extend RODENT to support downlink communications. This will further increase nodes' flexibility and possibilities (*e.g.,* firmware over the air upgrade). We plan to conceive an efficient link layer protocol for precise link costs and performance assessment for multi-RAT. In addition, we plan to combine RODENT with an efficient data reduction scheme to reduce even more energy consumption. We intend to run a larger simulation and experimentation, where end-to-end delay will be measured to assure that RODENT's usage does not slow down the network, and we will show how this satisfies QoS in terms of delay.

ACKNOWLEDGMENTS

This work was partially supported by a grant from CPER DATA, Sencrop, FEDER, I-SITE and Lirima Agrinet project.

REFERENCES

[1] D. Ye, D. Gong, and W. Wang. Application of wireless sensor networks in environmental monitoring. In *International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, 2009.

[2] B. Foubert and N. Mitton. Long-range wireless radio technologies: a survey. *Future Internet*, 12(1), 2020.

[3] B. Foubert and N. Mitton. Autonomous collaborative wireless weather stations: a helping hand for farmers. *ERCIM News*, (119):37–38, 2019.

[4] F. Bari and V. C. M. Leung. Multi-attribute network selection by iterative topsis for heterogeneous wireless access. In *IEEE Consumer Communications and Networking Conference (CCNC)*, 2007.

[5] W. Zhang. Handover decision using fuzzy madm in heterogeneous networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2004.

[6] M. A. Senouci, M. S. Mushtaq, S. Hoceini, and A. Mellouk. Topsis-based dynamic approach for mobile network interface selection. *Computer Networks*, 107:304 – 314, 2016.

[7] E. Stevens-Navarro and V. W. S. Wong. Comparison between vertical handoff decision algorithms for heterogeneous wireless networks. In *IEEE Vehicular Technology Conference (VTC)*, 2006.

[8] F. Bari and V. C. M. Leung. Automated network selection in a heterogeneous wireless network environment. *IEEE Network*, 21(1):34–40, 2007.

[9] L. H. Teixeira and A. Huszak. Preemptive network selection for v2v communication. In *International Conference on Telecommunications and Signal Processing (TSP)*, 2019.

[10] I. Bisio and A. Sciarrone. Fast multiattribute network selection technique for vertical handover in heterogeneous emergency communication systems. *Wireless Communications and Mobile Computing*, 2019:1–17, 2019.

[11] J. Famaey, R. Berkvens, G. Ergeerts, E. D. Poorter, F. V. d. Abeele, T. Bolckmans, J. Hoebeke, and M. Weyn. Flexible multimodal sub-gigahertz communication for heterogeneous internet of things applications. *IEEE Communications Magazine*, 56(7):146–153, 2018.

[12] J. Hoebeke, J. Haxhibeqiri, B. Moons, M. Van Eeghem, J. Rossey, A. Karagaac, and J. Famaey. A cloud-based virtual network operator for managing multimodal lpwa networks and devices. In *Cloudification of the Internet of Things (CIoT)*, 2018.

[13] O. Bouchet, A. Kortebi, and M. Boucher. Inter-mac green path selection for heterogeneous networks. In *IEEE Globecom Workshops*, 2012.

[14] T. De Schepper, P. Bosch, E. Zeljković, F. Mahfoudhi, J. Haxhibeqiri, J. Hoebeke, J. Famaey, and S. Latré. Orchestra: enabling inter-technology network management in heterogeneous wireless networks. *IEEE Transactions on Network and Service Management*, 15(4):1733–1746, 2018.

[15] R. V. Rao. *Decision making in the manufacturing environment: using graph theory and fuzzy multiple attribute decision making methods*, chapter Introduction to multiple attribute decision-making (madm) methods, pages 27–41. Springer, 2007.

[16] C. L. Hwang and K. Yoon. *Multiple attribute decision making: methods and applications*. New York: Springer-Verlag, 1981.

[17] Espressif Systems. *ESP32 series datasheet*, 2020. Version 3.3.

[18] Pycom website. `https://pycom.io/`. [Online; accessed 22-Jan-2021].

[19] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados. Energy-efficient routing protocols in wireless sensor networks: a survey. *IEEE Communications Surveys Tutorials*, 15(2), 2013.

[20] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves. Radio link quality estimation in wireless sensor networks: a survey. *ACM Trans. Sen. Netw.*, 8(4), 2012.

[21] Pycom. *FiPy specsheet*, 2018. Version 1.0.

[22] STMicroelectronics. *UM2115 user manual*, 2018. Rev 5.

[23] Edimax. *EW-7811Un datasheet*, 2015.

[24] T. Andersén. Energy-efficient adaptive sensing in low power wide area networks. Master's thesis, Norwegian University of Science and Technology, 2018.

AUTHORS

Brandon Foubert received a Master's degree in Computer Networks and Embedded Systems from the University of Strasbourg in 2018. He investigated "Cooperation between multiple RPL networks" under the direction of Julien Montavont in the Network research group from the ICube laboratory. He is currently pursuing a PhD degree under the supervision of Nathalie Mitton in the FUN team at Inria Lille - Nord Europe since September 2018 and until September 2021. He is studying "Polymorphic wireless communication for connected agriculture".

Nathalie Mitton received the MSc and PhD. degrees in Computer Science from INSA Lyon in 2003 and 2006 respectively. She received her Habilitation à diriger des recherches (HDR) in 2011 from Université Lille 1. She is currently an Inria full researcher since 2006 and from 2012, she is the scientific head of the Inria FUN team . Her research interests focus on self-organization from PHY to routing for wireless constrained networks. She has published her research in more than 50 international journals and more than 120 international conferences. She is involved in the set up of the FIT IoT LAB platform (`http://fit-equipex.fr`, `https://www.iot-lab.info`), the H2020 CyberSANE and VESSEDIA projects and in several program and organization committees such as Infocom (since 2019), PerCom 2020&2019, DCOSS (since 2019), Adhocnow (since 2015), ICC (since 2015), Globecom (since 2017), Pe-Wasun 2017, VTC (since 2016), etc. She also supervises several PhD students and engineers.