IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 10, No. 1, March 2021, pp. 244~252
ISSN: 2252-8938, DOI: 10.11591/ijai.v10.i1.pp244-252

Virtual machine migration in MEC based artificial intelligence technique

Ali OUACHA[1], Mohamed EL Ghmary[2]
[1]Department of Computer Science, Mohamed V University, Faculty of Sciences, Rabat, Morocco
[2]Department of Computer Science, SidiMohamed Ben Abdellah University, FSDM, Fez, Morocco

Article Info

Article history:

Received Oct 27, 2020
Revised Jan 27, 2021
Accepted Feb 22, 2021

Keywords:

Ant colonny algorithme
Intelligence artificial
Migration virtual machines
Mobile edge computing
Routing protocole

ABSTRACT

The whole world is inundated with smaller devices equipped with wireless communication interfaces. At the same time, the amount of data generated by these devices is becoming more important. The smaller size of these devices has the disadvantage of being short of processing and storage resources (memory, processes, energy,...), especially when it needs to process larger amounts of data. In order to overcome this weakness and process massive data, devices must help each other. A low-resource node can delegate the execution of a set of computionly heavy tasks to another machine in the network to process them for it. The machine with sufficient computational resources must also deposit the appropriate environment represented by the adapted virtual machine. Thus, in this paper, in order to migrate the virtual machine to an edge server in a mobile edge computing environment, we have proposed an approach based on artificial intelligence. More specifically, the main idea of this paper is to cut a virtual machine into several small pieces and then send them to an appropriate target node (Edge Server) using the ant colony algorithm. In order to test and prove the effectiveness of our approach, several simulations are made by NS3. The obtained results show that our approach is well adapted to mobile environments.

Corresponding Author:

Ali OUACHA
Intelligent Processing Systems & Security Team (IPSS)
Computer Science Department
Mohamed V University, Faculty of Sciences
4 Avenue Ibn Battouta, B.P. 1014 RP, Rabat, Morocco
Email: ali.ouacha@um5.ac.ma

1. INTRODUCTION

The growing development in fields of computing and electronics has given rise to smaller machines with wireless communication interfaces and energy autonomy. The interconnection of these machines, the ability to move freely and the ability to self-organize constitutes a network known by mobile adhoc network abbreviated as MANET [1-2]. In addition, the amounts of data generated by each device and the demands of processing resources have also become increasingly large. However, the smaller size of these sophisticated devices poses the constraint of lack of resources and more particularly, those that are related to computing and storage. One of the solutions consists in placing a more powerful machine (Edge Server), in terms of processing resources, in the periphery of the network. This solution falls within the concept of mobile edge computing (MEC) [3-6], illustrated by Figure 1. However, in order to be able to process information or perform a set of tasks coming from any node in the network, the edge server (ES) must have the virtual machine (VM) representing the environment for processing or executing these tasks. Thus, the main

objective of this work is to find a technique that allows nodes in the network to send their virtual machines to other resource-rich nodes. To this end, we have proposed an approach that is based on an artificial intelligence (AI) technique. Thus, by applying the ant colony algorithm, our principal aim is to migrate a virtual machine, which is considered as large amount of data, by dividing it into small pieces transported through several other nodes to a destination node (edge server). To implement this approach, we will also profit from the advantages the OLSR protocol [7-8]. Following its proactive principle, the control messages exchanged between the nodes serve, on the one hand, to disseminate information on the processing capacities and, on the other hand, to decide on the paths adopted to send the composite pieces of the virtual machine.

The rest of the paper is organized as: section 2 is about a literature review. The problem formulation and its solution are discussed in section 3. The simulation results are discussed in section 4. Finally, the conclusion of this paper is in section 5.

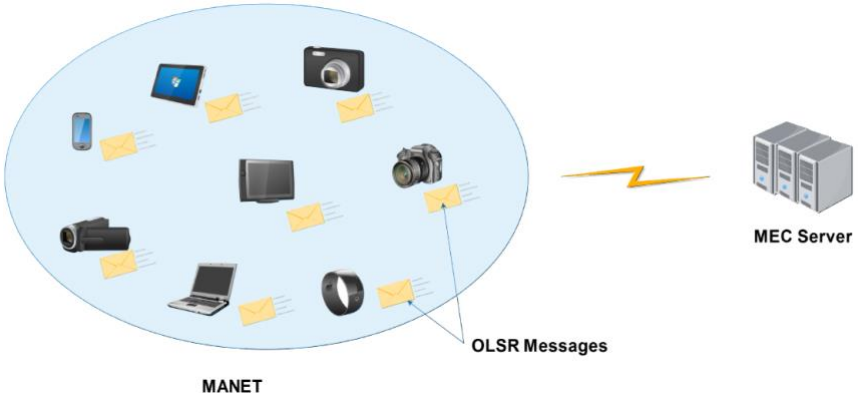


Figure 1. Mobile edge computing (MEC)

2. OLSR OVERVIEW

Optimized link state routing protocol (OLSR) [7] is a routing protocol generally used in MANET [1] networks. OLSR is a proactive, link-state optimized routing protocol that creates a routing table containing routes to all nodes in the network ahead of time and even before an actual data transmission request occurs. It uses control messages HELLO and topology control (TC) to propagate information about nodes and link states to the neighborhood and to the entire network. Thanks to this exchange of messages, each node obtains information about topology of the network and feeds its routing table by the shortest paths to all the other nodes.

The OLSR presents an optimized version of the open shortest path first (OSPF) [9] link-state routing protocol. Indeed, instead of using the flooding mechanism to broadcast information about the topology of the network through TC messages, as OSPF does, just nodes from an elected subset among neighbors of the node wishing to broadcast or rebroadcast these messages which are authorized to retransmit them. The elements of this subset are called multipoint relai (MPR) and each node of the network selects its MPRs from its symmetric one hop neighbors so as to construct the most minimal subset which covers all of the two-hop neighbors (Green color nodes in Figure 2(d)). It should also be noted that HELLO messages are never retransmitted. It serves for discovering the neighborhood and also the two-hop neighborhood. In fact, each OLSR node extracts the information on the nodes of the first neighborhood and the nodes of the second neighborhood via received HELLO message. In addition, the node which received the HELLO message calculates the node which provides the best path to the two-hop nodes among the nodes of its first neighborhood and selects it as the MPR. In this way, each node builds a set of MPRs.

In order to further illustrate the OLSR protocol operating, the Figure 2 gives the structure of the main control messages used when all nodes of the network are equiped with a ingle interface and use OLSR as the routing protocol as shown in Figure 2(a-b). This figure also shows the time intervals that separate the sending of two consecutive messages, whether for HELLO messages or for TC as shown in Figure 2(c) messages. HELLO messages are sent every two seconds while TC messages are sent every 5 seconds. It also explains the different categories of nodes as well as the links that exist between them and their relationship with the node (consider as an example) concerned by the execution of the OLSE protocol as shown Figure 2(d). Ended, the operation of the OLSR protocol is based on the periodic exchange of control messages.

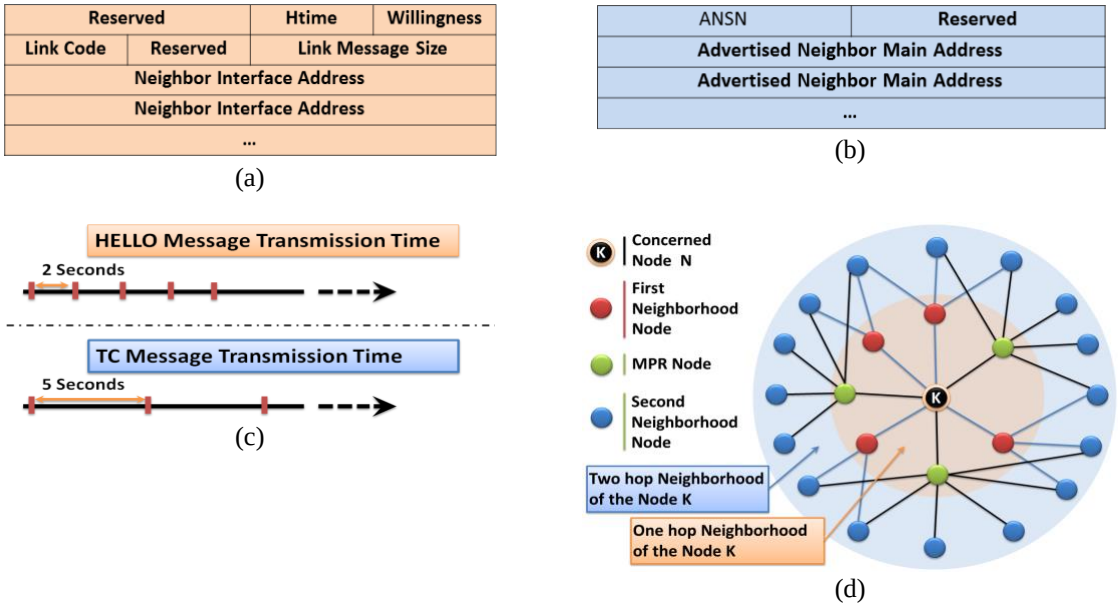


Figure 2. Optimized link state routing protocol (OLSR). (a) HELLO message structure, (b) TC message structure, (c) HELLO and TC messages time trnsmission, (d) OLSR nodes type

3. RELATED WORKS

In this section, we will present some previous researchs on the migration of virtual machines (VM) to an edge server (ES) in a mobile edge computing (MEC) environment. Indeed, the authors of [10] propose a methodology based on traffic and geographic locations of data centers to meet customer demands for VM migration called virtual machine migration approach. The algorithm used here is run at regular intervals to check for network traffic. It also checks the distance of the data centers where the customer's request is to be placed. In [11], the authors propose a method of migration of VM, named Ada-Things, which is determined by its workload characteristics. Specifically, based on the variation in the current rate of thirty pages of memory in IoT applications, Ada-Things can adaptively select the most suitable migration method to copy the memory pages. The autors of [12] proposed an adaptive pre-paging to eliminate duplicate page transmission and dynamic self-ballooning to avoid the transfer of free memory pages. The authors of [13] suggest a framework for the allocation and migration of virtual machines that exploits performance-to-power ratios (PPRs) for different types of hosts. By achieving the optimum balance between host usage and power consumption, this framework is able to ensure that hosts are operating at the most energy efficient usage levels. In the research made by [14], the authors propose a virtual machine migration strategy which aims to reduce network congestion in a MEC environment and thus improve QoS. It opts for bandwidth sensitive applications to be offloaded to edge servers to be executed by VM on those edge servers with minimal costs to surrounding users. In order to improve the QoS such as TCP throughput, the authors of [15] propose a VM migration method that routes a VM from one congested node to another node of a mobile device. In this method, users can choose a less congested node even if it is far away instead of a closer but congested one. The choice is made according to an expected TCP speed. The authors of [16] propose a seamless live VM migration between neighboring cloudlets with the goal of eliminating the delay caused by service initiation time after moving away from the cloudlet. A seamless live VM migration is achieved with the prior knowledge of the IP address of the VM being migrated in the destination cloudlet and more importantly with multipath TCP.

For the purpose of optimizing network and computing resources, several studies use artificial intelligence techniques [17-19] to implement VM migration approaches. Indeed, the authors of [17] present a multi-objective virtual machine (VM) placement scheme for ECDC edge cloud data centers which aims to minimize network traffic of interacting VMs. The proposed approach is based on the artificial bee colony optimization algorithm [20-21] to affect VMs on physical machines. The authors of [18] propose a machine learning-based VM migration approach [22]. The main contribution of this article is to use an adaptive live migration approach based on predictive mechanisms that reduce downtime during live migration over wide area networks for standard workloads. The authors of [19] assume that the VM placement problem can be formalized as a bin packing problem, which turns out to be NP-hard. To solve it, they used a genetic

algorithm [23] to develop an algorithm based on clusters and which produces an approximation result of the bin pack problem. Indeed, the proposed algorithm groups the population of the current generation and selects individuals from different groups with reduced crossbreeding operations.

Unlike these researches, our research seeks to take advantage, on the one hand, of the control information exchanged between the nodes of the network; and on the other hand, to use an artificial intelligence technique that can be summed up in the theory of the ant colony in order to find the most suitable paths by considering as a pheromone, a metric calculated from the control messages.

4. PROBLEM FORMULATION AND RESOLUTION

4.1. Problem context

Before we begin to offload any information or set of tasks to be treated, we need to be certain that the node for which those tasks are offloaded contents the execution environment which is represented by the VM appropriate to the execution of those tasks. To send its VM, the node needs at first decide which node in the periphery of the network is privileged to host the VM or the VMs. Indeed, we will take advantage of the proactive concept of the OLSR routing protocol [24] so as to insert in the control messages, periodically exchanged between the nodes, information concerning the processing capacities of the server node or of the server nodes of the network and number of tasks that are in progress or waiting to be processed. Each node can extract from these messages information about the state of the links between nodes. Once the target that will host the VM is known, the virtual machine migration process will be triggered. The technique used in this study is inspired from the ant lifestyle. Indeed, everyone knows that ants work collectively and tirelessly to store enough food to get through the winter. And in the gathering of this food, the ants of a colony always manage to find the shortest route even under the constraints of changing environment [25]. The key and fundamental concept to be drawn here is being able to move a mountain of wheat while working together and carrying only one grain of wheat per ant. Thus, by correspondence to the ant, the general idea of this article is to transfer a very large VM (mountain of wheat) while subdividing it into pieces VM_k (grain of wheat) of smaller size so as not to disturb the function of the entire network.

4.2. Server edge selection (destination node selection)

A MANET [1] network is made up of a set of machines of a heterogeneous nature. To be able to broadcast its processing capacities, each node injects into control messages of the OLSR protocol [7] all the physical characteristics that are related to the execution of tasks. Thus, as illustrated by Figure 3, before sending its first control messages (Hello or TC) the node in question inserts in these messages the number of processors it has at its disposal, the number of cores of each processor, the frequency of execution, the size of the process cache memory, the capacity of the main memory, the access speed of this memory, and the bus frequency. This can be implemented by modifying the structure of the HELLO and TC massages, as shown in Figure 2(a-b) by adding a number of 32 byte lenth lines (one line or two or three depending on the amount of information to be transmitted) before the first "*Neighbor field Interface Address*" in the HELLO message or before the first "*Advertised Neighbor Main Address*" field in the TC message. After that, once the message is received, each node in MANET stores these performances in a local table whose key is the identifier of the sending node. Based on stored information, each node of the network, where the computation performances are limited, decides about the node which will be the target of its virtual machine.

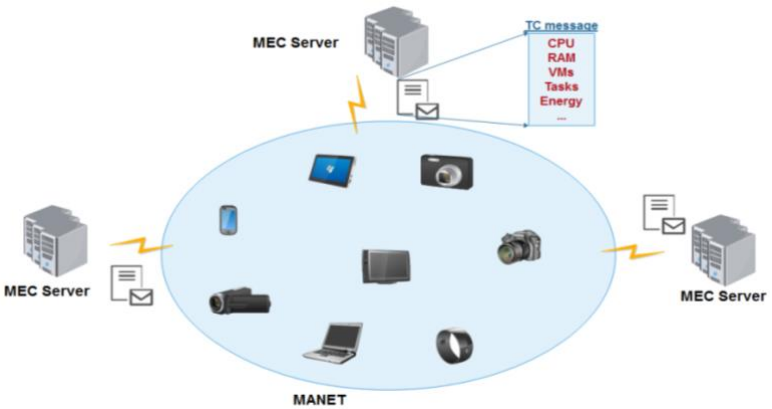


Figure 3. Server edge selection in MEC environment

4.3. Virtual machine splitting and migration

First, the virtual machine VM is divided into equal size pieces $VM_k\ with\ k = 1,2, \dots, m$. The size of each piece is chosen so as to correspond to the greatest possible value of the packets which can be transported by the network. Since we are only interested in the prior sending of VMs, routing, partitioning and reassembling times of VMs are neglected. Once the target node is well defined and the VM is well partitioned, all that remains is to find the path or paths to follow to send the pieces from the VM to the destination node. To do this, an adapted version of the ant colony optimization algorithm [25] is adopted.

4.4. Provided data

In this section, we will define all the concepts that will be useful later. Thus, we define by X the finite set representing the nodes of the network. We also define by $U = \{(i,j)\ with\ i, j \in X\}$ the finite set of links connecting nodes of X. The variables $d_{i,j}$ representing the quality (cost, metric ...) of each link $(i,j) \in U$ are estimated by the (1) as a function of the mobility of the nodes using the $RTTQ$ as a metric. Note that this metric estimates the time remaining for a node to leave the neighborhood of another node. It is deduced from the coordinates exchanged through the HELLO messages [26].

$$d_{ij}(t) = \frac{RTTQ_{i,j}(t)}{RTTQ_{Max}} = \frac{RTTQ_{i,j}(t)}{Range} \quad (1)$$

We aim that pieces of the VM pass once and only once through the nodes of a subset of X (Particularly the nodes from the path to the target node). By using $RTTQ$ as a pheromone, we then want to determine the paths that will be traveled by the different pieces of the VM and that will minimize packets loss (increase the stability of links). Therefore, when an ant moves from one node to another, the quantity of pheromone deposited at time t is denoted by $\tau_{i,j}(t)$. The value of $\tau_{i,j}(t)$ is calculated and updated at the end of each path according to (2). Note that each ant k has traversed the n_k nodes that make this path.

$$\tau_{i,j}(t + n_k) = \rho.\tau_{i,j}(t) + \Delta\tau_{i,j}(t) \quad (2)$$

Where $\rho \in [0,1[$ is a coefficient which will define the speed of evaporation of pheromones on the links between time t and time $t + n_k$, and $\Delta\tau_{i,j}(t)$ represents the quantity of pheromones deposited by the ants on the link (i,j) in this same time interval. It is defined (calculated) by (3).

$$\Delta\tau_{i,j}(t) = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}(t) \quad (3)$$

In this equation, $T_k(t) = (u_{k_1}, \dots, u_{k_q})$ is the path traveled by the k^{th} ant in the time interval $[t, t + n_k]$, and $L_k(t)$ its length. $T_k(t)$ (therefore $L_k(t)$) is obtained by analyzing the memory of the ant. $\Delta\tau_{i,j}^{k}(t)$, calculated by (4), is the amount of pheromone deposited by this ant on link (i,j) in this same time interval.

$$\Delta\tau_{i,j}^{k}(t) = \begin{cases} \frac{Q}{L_k(t)}\ if\ (u \in T_k(t)\ et\ u = (i,j)) \\ 0\ otherwise \end{cases} \quad (4)$$

Where Q is a constant. The length $L_k(t)$ of a path μ is the sum of the lengths of the links that compose it, that is:

$$L_k(\mu) = \sum_{i=1}^{q-1} d_{u_i,u_{i+1}} \quad (5)$$

4.5. Ant system

Now it is important to clarify the behavior of the entire colony. At any time t, each ant (packet carrying a piece of the VM) chooses a destination node from its neighborhood according to a defined choice. All ants move at time $t + 1$ in another node of their choice. We call an iteration of the Ant System (AS) algorithm, the set of displacements of the entire colony between time t and time $t + 1$. Thus, after n_k iterations each, ant k will have traveled a path and reached an end node (the node which will contain the VM or another node representing an end path).

4.6. Transitions choice

An ant k, placed on node i, at instant t will choose its next hop (neighboring node) j according to the quantity of pheromones $\tau_{i,j}(t)$ deposited on the link connecting these two nodes. This choice will be made randomly, with a probability of choosing the node j given by:

$$p_{i,j}^{k}(t) = \frac{\tau_{i,j}(t)}{\sum_{l \in N_i^k(t)} \tau_{i,l}(t)} \quad (6)$$

Where N_i^k defines the set of nodes belonging to the neighborhood of node i and which ant k, placed on node i, has not yet visited at time t in the current path.

4.7. Global algorithm

- Divide the VM into m of small pieces of equal size; thus, $VM = \sum_{k=1}^{m} VM_k$. each piece will be wrapped in a package which will be the equivalent of a k ant.
- Initialize the value of the Initial pheromone quantity $\tau_{i,j}(t)$ by a constant value c.
- For each ant, we calculate the probability $p_{i,j}^{k}(t)$ of the choice of the next neighboring node according to (6). We will choose as the next hop the node whose probability is greater.
- We update the amount of pheromone according to (2).
- We repeat steps 3 and 4 until the ant k arrives at the destination node or to an unreachebel node.

5. RESULTS AND DISCUSSION

In this section, the simulations are used to prove the efficiency of our approach of sending VM to edge server based on the ant colony algorithm on the MEC. First, the simulations environment and parameters, generated packets as well as their routing are explained. Next, the obtained results present the sending of different parts of a VM are analyzed, including percentage of sent, reached, unreached and lost packets.

5.1. Simulation environment

To test the efficiency of our approach, several simulations are done by NS3 [27]. They took place in two environments which differ in terms of mobility of the nodes. In the first one (static case), the nodes are dispersed according to a grid of five nodes in height and five nodes in width (5×5) spaced by a distance of 500 meters. Consequently, each node has two, three or four fixed neighbors depending on its position in the grid (corner, border or middle node). On the other hand, in the second environment (case with mobility), the simulations consist of a network of 25 mobile nodes moving according to the random waypoint (RWP) mobility model [2] in an area of $1500 \times 1500\ m^2$.

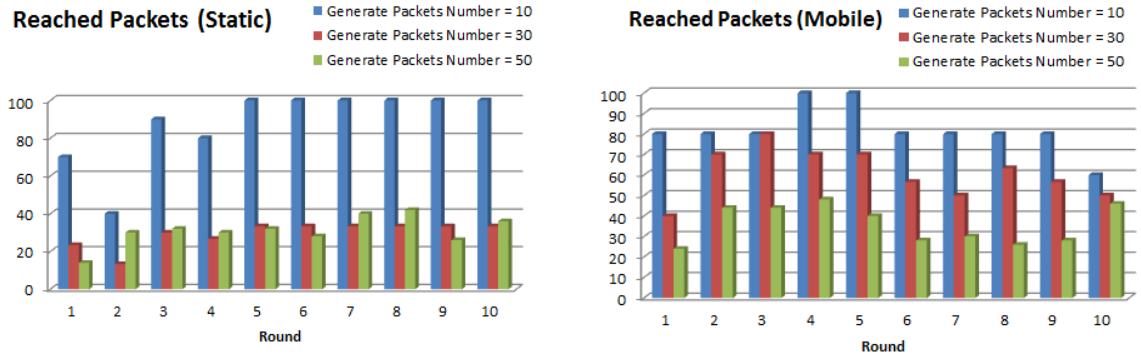
In both cases, all nodes have a single wireless communication interface that also uses OLSR as a routing protocol. The other parameters of the simulation environments are shown in Table 1. To simulate the sending of VM_k parts of a VM to the destination node representing the MEC server, during each time interval (round) lasting of one second, a random node is chosen to send consecutively a series of packets whose size is 1000 Byte. The routing of VM_k is done from node to node using the ant theory algorithm defined in section 3. Thus, each packet will be considered as an ant carrying a load to be routed to a specific destination (Edge Server). To increase the probability that all ants will pursue different paths at the beginning of the send, the number of all VM_k parts is divided equally on nodes in the first neighborhood of the source node. The next hop is defined according to the algorithm 1.

Table 1. The simulations environments

Parameter	Value
Network Simulator Vertion	ns-3.29
Protocol	OLSR/Ant Colonny
Simulation Time	100 s
Simulation Area	1500×1500 m^2
Number of Nodes	25
Transmission Range	500 m
Mobility Model	Random WayPoint
Max Speed	20 m/s
Pause Time	0 s
Part of the VM (VM_k) Size	1000 Byte
WiFi Mac Protocol	802.11b

5.2. Results and discussion

Figure 4 represents the percentage of the number of successfully sent and received packets in both static and mobile case as a function of the round number. Figure 4(a) shows an improvement in the number of packets received in static case over time. In other words, each round has a more improved number of received packets than the previous round. This shows the effectiveness of our approach in terms of learning about finding paths to the target node. If the improvement in the success rate for finding the path is guaranteed in the static case, this is only periodic in the mobile case, as shown in Figure 4(b). Indeed, because of the mobility of nodes, the topology of the network changes over time. Thus, the paths can change from round to round. Therefore, the approach used to find the paths to the destination node must be adapted to the new topology after change. This justifies the periodic improvement in the number of successfully received packets.



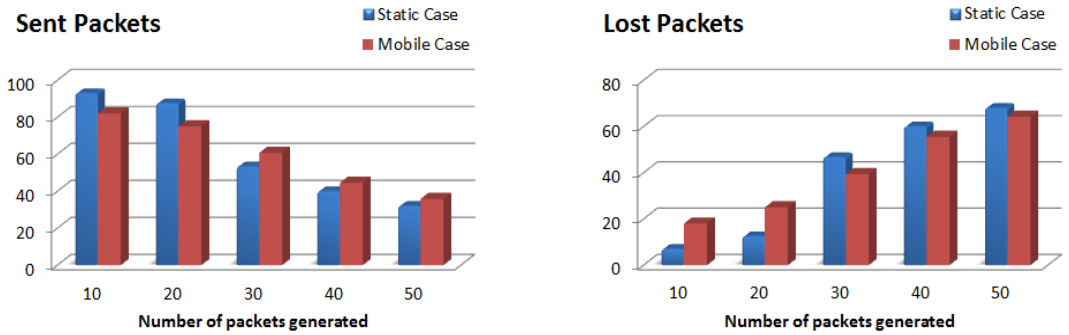
(a) (b)

Figure 4. Percentage of reached packets sent based on the round number in both static and mobile environements. (a) Reached packets in static case, (b) Reached packets in mobile case

In the case of our study, virtual machines are known by large sizes. So to be able to decide on the effectiveness of our approach, we started by splitting each VM into a set of packages of similar size. Then we have to study the behavior of network towards this large quantity of data. Thus, in the Figure 5, we represent the averages of the percentage of packets sent and lost compared to the number of packets generated in static and mobile environments. In fact, Figure 5(a) represents the packets sent and Figure 5(b) represents the lost packets. So, for more details, the Figure 5(a) represents the average of the number of packets that are sent whether in the static or in the mobile case. In the two cases, it is found that the percentage of the number of packets that are successfully sent decreases as the number of generated packets increases. This reduction applies for both cases: static and mobile. Thus, the percentage of packets sent approaches almost one hundred percent when the number of packets generated is smaller. On the other hand, this number is only 35% when the number of packets generated exceeds 40 packets and tends to stabilize when the number of packets sent exceeds 50. Indeed, sending of large quantities of packages is also confronted with a set of constraints related to the nodes of the network. Therefore, when the temporary memories (queue), intended to contain packets waiting to be sent, are full, the other packets that arrive are automatically rejected.

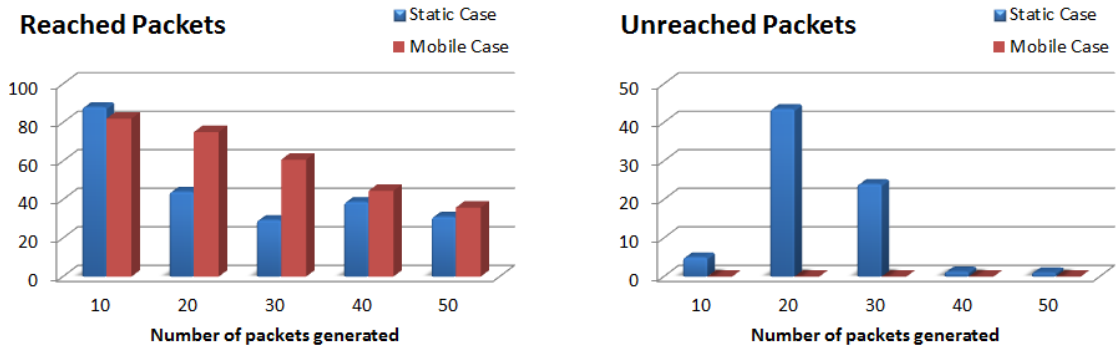
In addition, and since the operating of the VM essentially depends on the total reception of all pieces, we studied the number of reached and unreached packets in the network for the modifier version of the OLSR protocol in two types of environments. Indeed, the Figure 6 represents the average of the percentage of reached packets and unreached packets compared to the number of generated packets in static and mobile environments. In one hand, the Figure 6(a) concerns the unreached packets. On the other hand, the Figure 6(b) concerns the unreached packets. So, regardless the percentage of packets that are successfully received versus the total number of packets generated, here too, we see that this number decreases when the number of packets generated increases. This is justified by the presence of control traffic. So because the network also uses the OLSR as a routing protocol, and due to its proactive nature, control messages (HELLO, TC and others) are generated periodically. Packets containing HELLO messages are generated by each node in the network every two seconds and further packets containing TC messages are generated by all nodes and retransmitted by a subset of nodes every five seconds. When the sending of the packets that represent the parts of the VM coincides with the sending of the control packets, it causes collisions and also queue

congestion, resulting in the loss of a set of packets. In addition, sending large quantities of packets also faces a set of constraints related to links between network nodes as well as network protocols.



(a) (b)

Figure 5. Averages of the percentage of sent and lost packets versus the number of generated packets in both static and mobile environements. (a) Sent packets, (b) Lost packets



(a) (b)

Figure 6. Averages of the percentage of reached and unreached packets versus the number of generated packets in bouth static and mobile environements. (a) Reached packets, (b) Unreached packets

6. CONCLUSION

The objective of this article is to show that the case of a node whose computational performance is very limited. To function well, the node must delegate a set of tasks to the egde server. Performing these tasks requires the presence of the VM representing the simulation environment of the node in question. Thus, a new approach has been proposed in this research. It is based on an artificial intelligence technique to send the whole VM to the edge server. Indeed, it is decomposed into a set of small parts that are encapsulated in packets each of which is considered an ant. Therefore, we used the ant theory algorithm to route these different pieces to a destination node. Considering the RTTQ metric as a pheromone, the more stable paths are founded. The results of the simulations performed by NS3 show that the whole packets find stable paths to the destination node. However, these results show that the percentage of packets that are received also depends on the performance of the nodes and the network. Thus, as a perspective, a future study aims to also take into account the waiting queue of the MAC layer and the bandwidth of the links. Another perspective aims to find a perfect synchronization between the VM_k parts and the control packets. In addition, a final perspective is to make the VM_k parts size variable. It will be set dynamically depending on the network condition.

REFERENCES

[1] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," *RFC2501, MANET Performance Issues*, 1999.

[2] R. R. Roy, "Handbook of Mobile Ad Hoc Networks for Mobility Models," 1 ed.: Springer US, 2011.
[3] Q. Z. Jie Cao, Weisong Shi, "Edge Computing: A Primer," 1st ed. ed.: Springer, Cham, 2018.
[4] S. Nunna and K. Ganesan, "Mobile Edge Computing," in *Health 4.0: How Virtualization and Big Data are Revolutionizing Healthcare*, C. Thuemmler and C. Bai, Eds., ed Cham: Springer International Publishing, pp. 187-203, 2017.
[5] R. Torre, T. Doan, and H. Salah, "Chapter 4-Mobile edge cloud," in *Computing in Communication Networks*, F. H. P. Fitzek, F. Granelli, and P. Seeling, Eds., ed: Academic Press, pp. 77-91, 2020.
[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, pp. 450-465, 2018.
[7] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," vol. RFC3626: RFC Editor, 2003.
[8] C. Dearlove and T. Clausen, "Routing multipoint relay optimization for the optimized link state routing protocol version 2 (olsrv2)," *RFC 7187 (Proposed Standard), The Internet Engineering Task Force (IETF)*, 2014.
[9] J. Moy, "OSPF Version 2," *IETF Tools*, p. April 1998, 1998.
[10] N. H. Shahapure and P. Jayarekha, "Distance and Traffic Based Virtual Machine Migration for Scalability in Cloud Computing," *Procedia Computer Science*, vol. 132, pp. 728-737, 2018, doi: 10.1016/j.procs.2018.05.083.
[11] Z. Wang, D. Sun, G. Xue, S. Qian, G. Li, and M. Li, "Ada-Things: An adaptive virtual machine monitoring and migration strategy for internet of things applications," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 164-176, 2019, doi: 10.1016/j.jpdc.2018.06.009.
[12] K. Z. Ibrahim, S. Hofmeyr, C. Iancu, and E. Roman, "Optimized pre-copy live migration for memory intensive applications," presented at the *Proceedings of 2011 International Conference for High Performance Computing*, Networking, Storage and Analysis, Seattle, Washington, 2011, DOI: 10.1145/2063384.2063437.
[13] X. Ruan, H. Chen, Y. Tian, and S. Yin, "Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds," *Future Generation Computer Systems*, vol. 100, pp. 380-394, 2019, doi:10.1016/j.future.2019.05.036.
[14] B. Yang, J. Li, and Y. Li, "Research on QoS-Oriented Virtual Machine Migration Strategy in Mobile Edge Computing," in *2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, pp. 227-231, 2019, doi: 10.1109/ICICTA49267.2019.00055.
[15] J. Kikuchi, C. Wu, Y. Ji, and T. Murase, "Mobile edge computing based VM migration for QoS improvement," in *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*, pp. 1-5, 2017, doi: 10.1109/GCCE.2017.8229344.
[16] F. Teka, C.-H. Lung, and S. A. Ajila, "Nearby live virtual machine migration using cloudlets and multipath TCP," *Journal of Cloud Computing*, vol. 5, no. 12, 2016, DOI 10.1186/s13677-016-0061-0.
[17] S. S. Nabavi, S. S. Gill, M. Xu, M. Masdari, and P. Garraghan, "TRACTOR: Traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization," *International Journal of Communication Systems*, vol. 34, no. 5, 2021, DOI: 10.1002/dac.4747.
[18] M. Arif, A. K. Kiani, and J. Qadir, "Machine learning based optimized live virtual machine migration over WAN links," *Telecommunication Systems*, vol. 64, no. 2, pp. 245-257, 2017, DOI: 10.1007/s11235-016-0173-3.
[19] B. Zhang, X. Wang, and H. Wang, "Virtual machine placement strategy using cluster-based genetic algorithm," *Neurocomputing*, vol. 428, pp. 310-316, 2021, doi: 10.1016/j.neucom.2020.06.120.
[20] G. Lindfield and J. Penny, "Chapter 7-Artificial Bee and Ant Colony Optimization," in *Introduction to Nature-Inspired Optimization*, G. Lindfield and J. Penny, Eds., ed Boston: Academic Press, pp. 119-140, 2017.
[21] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Citeseer2005.
[22] J. R. Anderson, "Machine learning: An artificial intelligence approach," vol. 3: Morgan Kaufmann, 1990.
[23] X.-S. Yang, "Chapter 5-Genetic Algorithms," in *Nature-Inspired Optimization Algorithms*, X.-S. Yang, Ed., ed Oxford: Elsevier, pp. 77-87, 2014.
[24] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC3626: RFC Editor, 2003.
[25] M. Dorigo and T. Stützle, "Ant Colony Optimization," Bradford Company, 2004.
[26] A. Ouacha, N. Lakki, J. E. Abbadi, A. Habbani, B. Bouamoud, and M. Elkoutbi, "Reliable MPR selection based on link lifetime-prediction method," in *2013 10th IEEE International Conference On Networking, Sensing and Control (ICNSC)*, 2013, pp. 11-16.
[27] nsnam. *ns-3 Manual*.