Opportunistic Data Gathering in IoT Networks using Discrete Optimization

Edvar Afonso and Miguel Elias M. Campista
Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ), Rio de Janeiro, Brazil
{edvar,miguel}@gta.ufrj.br

***Abstract*—The Internet of Things (IoT) is based on data collection for future processing and decision making. In multihop Low-Power and Lossy Network (LLN) scenarios, efficient data forwarding in terms of generated traffic and energy consumption is fundamental. This paper revisits the concept of mobile agents to collect data along the agents's itinerary. The idea is to avoid sending requests to the network when non-expired contents that were opportunistically collected are available in the cache of a central element. In the proposed mechanism, the itinerary is composed of devices of interest and intermediate devices in a closed loop at the origin. Knapsack optimization is used to add unsolicited data opportunistically. The reward is calculated according data popularity. Simulations show that it is possible to reduce network traffic and the energy consumed by devices when compared to the traditional mobile agent data gathering model.**

***Index Terms*—IoT, data gathering, mobile agents**

I. Introduction

The Internet of Things (IoT) is a system composed of electronic devices able to interact and share data from their surroundings. This data is typically forwarded to a sink node where it is stored and used as input of upper-level applications. IoT devices are known to their hardware constraints and, therefore, data transmission and reception become a challenge. In multihop scenarios, this is even more critical, as the traffic generated by any node may require multiple wireless transmissions until it finally reaches the destination.

In multihop scenarios, the considerable effort for data gathering and forwarding can lead to a high increase in latency, energy consumption, and congestion over the wireless links. To overcome these issues, the Mobile Agent (MAs) paradigm has been revisited as an efficient alternative for data collection in IoT [1]–[5]. Unlike Internet routers, IoT devices are at the network edges and can consequently take more intelligent decisions. We borrow the traditional definition of MA as a computer code that is dispatched by the sink to the network. This code is used by sensor nodes to perform specific tasks, e.g., data fusion and aggregation. By using MAs, we can change devices behavior by temporarily installing different codes on each node. For instance, rather than performing data aggregation only at the sink node, the MA can save energy by compressing the raw data at each device before forwarding it. To perform the assigned tasks, the MA must cross the network following a predetermined itinerary computed using an optimized algorithm (Itinerary Planning).

The MA itinerary in IoT networks can be computed using either static or dynamic approaches. In the static approach, the sink node is in charge of itinerary computation. Once determined, the itinerary is registered in the MA before being dispatched to the network. Even though the static approach moves the computation task from resource-constrained devices to the sink, it requires these devices to periodically send network updates to the sink. In the dynamic approach, on the other hand, the mobile agent autonomously determines the IoT devices to visit and the order of migration based on current network status. This approach eliminates the need for periodical updates, but requires higher processing power from sensors, besides increasing the MA complexity. Our proposal considers only static itineraries for the sake of simplicity to conduct traditional MA data collection. Traditional MAs traverse many intermediate nodes to migrate from the current source node to the next, ignoring the data available on intermediate devices. These intermediate devices, however, may contain relevant data that can be of interest to network clients in the future. Proactively collecting data from intermediate nodes represents a smart alternative for data gathering.

This paper proposes the Agent-Knap, a network architecture that combines the use of MAs with opportunistic data collection to reduce the network energy consumption. Our proposal relies on static itineraries, which include source and intermediate nodes, and a selection mechanism that prioritizes data acquisition based on freshness and popularity. Unlike previous proposals [1], [3], [6], in which the static itinerary is computed for data collection at source nodes, our proposal also considers opportunistic data collection on intermediate nodes within the MA itinerary. The mechanism to assign priority for opportunistic data is reduced to a 0-1 knapsack problem. This model computes the Knapsack award based on weights proportional to the data content popularity. In addition, the model considers the MA payload size as the maximum backpack capacity. We compare the Agent-Knap with the traditional MA, i.e., without using opportunistic data collection at intermediate nodes. Simulation results reveal a relevant reduction in the network energy consumption.

This paper is organized as follows. Section II overviews the related work. Section III presents the proposed collection mechanism, Agent-Knap. Following, Section IV describes the simulation environment and the results achieved. Finally,

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

Section V concludes this work and presents future directions[1].

II. Related Work

Optimized data collection in LLN networks has been subject of many recent works. The use of mobile agents for data gathering in LLN has been widely studied in the sensor networking context [3]–[5], where authors started looking for new approaches to reduce power consumption and network traffic. Gavalas et al. [7] propose a static itinerary mechanism for multiple mobile agents. They primarily consider energy efficiency influenced by the increase in mobile agent size as it moves forward and aggregates information from network nodes. Unlike Agent-Knap, both studies do not evaluate data collection on intermediate devices while the mobile agent moves through the itinerary. As far as we know, opportunistic data collection using MA is original and particularly enhanced by the utilization of content popularity.

Opportunistic data collection on sensor networking has also been a relevant topic in the recent literature [8]–[10]. Nevertheless, in these works, data gathering mechanisms are applied for mobile networks. The mobile sink or the mobile sensors establish temporary communication links between each other.

III. Data Gathering Using the Agent-Knap

We consider a network composed of a sink node and sensor devices randomly deployed at an Area of Interest (AoI). Each device senses the environment and provides data of one type of content (e.g., temperature, pressure or vibration). Also, each device stores only the last sample of the content collected. The proposed architecture is centralized at the sink node, which also concentrates the role of network gateway. In addition, sensor devices can play the role of source nodes, those providing the requested data; or intermediate nodes, those interconnecting two consecutive source nodes.

The network operation relies on data requests sent to the sink node by external clients. Each request has information about the desired content and the minimum number of source nodes. The sink then decides if a new data gathering round is needed, depending on its cache freshness. In the proposed architecture, data can be collected from any network device. At the system initialization, however, each device must register and inform the type of content it can provide, in addition to the size in Bytes of the content. We assume that, after the system initialization, the sink node has information about the network topology, including the geographical position of all devices, to compute the MA itinerary. This view is periodically updated as a consequence of the static approach for itinerary planning.

A. Data gathering model

The main goal of our proposal is to save energy from IoT devices when a multihop mobile agent is used to collect data. The collected data from each device is stored in the sink cache until an expiration timer remains valid. Each stored data has a timer and, while valid, helps avoiding a new data collection round. Thus, reducing the number of requests sent to the network, we save network resources. The opportunistic data gathering has a key participation into the process, as it brings more data back to the sink, increasing the data freshness.

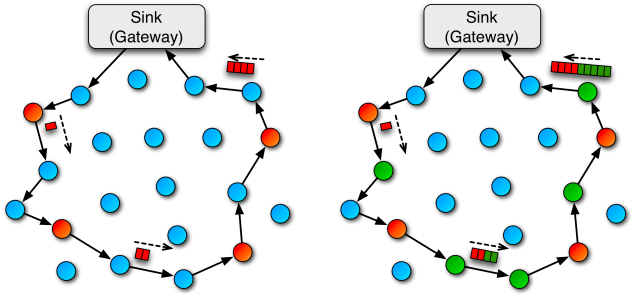


Fig. 1. (a) Traditional data collection using MA. (b) Data collection using the proposed Agent-Knap.

Figure 1(a) illustrates a typical data collection with static itinerary. The figure exemplifies a traditional configuration using a MA on an LLN network. In this example, the agent is programmed by the sink node to perform the collection on four source nodes, which appear in red in the figure. Note that other nodes, i.e., intermediate nodes, are visited anyway. Hence, in addition to the data collected in the source nodes, highlighted in red, the agent could also consider the contents existing in a subset of intermediate nodes, indicated in green. Intermediate nodes are selected with the solution of the Knapsack optimization problem to increase the data collection efficiency. This is the main idea of the Agent-Knap proposal, depicted in Figure 1(b). Note, however, that the proposal introduces a tradeoff on the MA size, which becomes larger as we opportunistically concatenate data.

The proposed MA format has the following reserved fields: processing code, source nodes list, next hop, and payload. The processing code contains the tasks that must be executed at each visited device. The source nodes list contains all the nodes that must be visited to complete a data gathering round. The next hop indicates the next device where the MA must be forwarded. It can be a source or an intermediate node. Finally, the payload stores the data collected at all visited devices. The Agent-Knap has then to deal with two main issues, which are its itinerary (itinerary planning) and the data it collects (payload computation).

B. Itinerary planning

Before sending the MA to the network, the sink must compute the itinerary, which is the sequence of source nodes to be visited and their intermediate nodes. The itinerary forms a closed loop starting at the sink. The loop must visit the number of source nodes requested by the external client. Considering the loop, the opportunistic content is obtained from possible intermediate nodes connecting two consecutive source nodes. The proposal manages the MA payload.

[1]A preliminary version of this paper was already published in Portuguese in a Brazilian workshop http://www.gta.ufrj.br/ftp/gta/TechReports/FiCa19.pdf.

After receiving a request, the sink node randomly chooses the subset of source nodes that will be used for the current gathering process, i.e., $\mathcal{N}_f \subset \mathcal{N}$. This set is used for itinerary computation. In this step, the algorithms of *Dijkstra* and *Christofides* are used to compute the optimized itinerary. The Christofides algorithm is a heuristic used to solve the Travelling Salesman Problem (TSP), which is the kind of problem to be solved for the MA itinerary computation. The output of the algorithm is a sequence of nodes that must be visited. Since these nodes may not be directly connected, the final itinerary is determined after computing the paths between consecutive source nodes and their respective costs using the *Dijkstra* algorithm. The *Christofides* algorithm is a TSP solution that, in the worst case, guarantees a ratio less than 3/2 between its solution and the optimal one [11]. The computed itinerary is an input for payload computation.

C. Payload computation

Prior to the transmission of the MA to the network, a maximum payload size, in Bytes, should be known by the sink. This payload should be divided into two parts: a fixed space reserved for the collection of source nodes content (G), which is the priority target; and a space (C) used for the concatenation of opportunistic content, collected at intermediate nodes. The Knapsack problem is solved by the sink that considers C as the maximum backpack capacity. Opportunistic content is then added to the MA payload without violating the maximum size C. Figure 2 depicts a payload example populated with four contents collected from source nodes (source node content – G) and four different concatenated contents (opportunistic content – C) to complete P (MA total payload size).

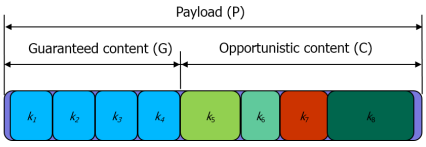


Fig. 2. MA payload P with guaranteed (G) and opportunistic contents (C).

More formally speaking, our proposal considers the existence of m different contents in the network, denoted by the $\mathcal{K} = \{k_1, ..., k_m\}$ set. Figure 3 presents the sink node modules and the relationship between them. The Network Topology, Content Size, and Opportunistic Data Collection modules represent the system input parameters that must be entered at the beginning of the process. The contents are distributed over the n network nodes, denoted by the $\mathcal{N} = \{n_1, ..., n_n\}$ set. For the purpose of simplification, it is considered that each network node provides information of only one content, although the proposed model is easily adaptable for scenarios where devices can provide more than one content type. The sink node receives requests from external clients, R_j ($j = \{1, 2.., p\}$ and $p \rightarrow \infty$). It is assumed that requests arrive sequentially at the sink, on instant t_j. The sink node stores each time instant t_j in a local database for historical recording and computation

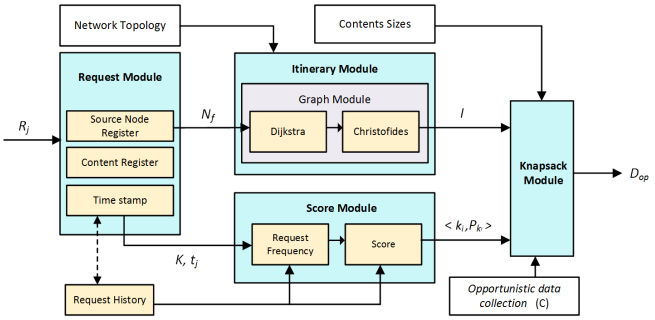


Fig. 3. The modules running at the sink and the relationship between them.

of the *Knapsack* score for each content. This step is processed by the Request Module.

The sink node initially checks for updated data in its cache aiming to give a fast and complete reply to the client with the desired number of source nodes. If the request is fully met by the cached data, the sink node sends the response to the requesting client without dispatching the MA to the network. If there is not enough updated data in cache, the sink sends a MA to the network to collect the missing data, which is those needed to complete the desired number of source nodes. For example, if a request to collect temperature information from seven strategic devices in the network is received and the sink node only has updated information from four devices in its cache, the sink node computes the itinerary and dispatches a MA to collect the data from the three missing source nodes.

The output from the Itinerary Module is the itinerary, I, containing all nodes that will be visited by the MA. I, therefore, is one of the Knapsack Module entries. The other entries are: a set of tuples $\langle k_i, P_{k_i} \rangle$, which lists the k_i contents served by the intermediate nodes and their related P_{k_i} scores, the size in Bytes of each intermediate data, and the payload space to be considered, C, in Bytes. The score for each content is computed as follows:

$$P_{k_i} = \alpha \left(\frac{1}{t_j - t_{j-1}} \right) + (1 - \alpha) f_{k_i}, \qquad (1)$$

where f_{k_i} is the frequency of requests received for k_i content. Equation 1 is composed of two components. The first is a component of P_{k_i} scores that is inversely proportional to the time interval between two consecutive requests for a given k_i content. That is, the closer two consecutive requests for k_i, the higher the corresponding P_{k_i} score. The second component is directly proportional to the frequency of requests received for the k_i content. The more frequently requests are made for a certain content k_i, more popular this content becomes and, thus, higher is its score P_{k_i}. The α parameter weights the importance of each component. For α close to one, a burst of requests for a given content results in higher scores. For α close to zero, more popular contents receive higher scores. With P_{k_i}, the sizes of each content, and the maximum size for the opportunistic content, C, the Knapsack problem is solved by the sink, which determines which content and in

which nodes they must be opportunistically collected; this output is D_{op}. This ensures that the payload of the agent is optimally populated by the most valuable content provided by the itinerary nodes without violating P.

IV. Simulations

We developed a simulation environment using Python to evaluate the proposed Agent-Knap. All topologies were implemented with the *NetworkX* package. In order to simplify the analysis, some premises were assumed. The initial power of all devices is the same; the data size for different types of content is the same; network devices have the same number of sensors available, with one sensor for each type of content; The sensing range for each sensor is the same in the entire network. Each request message received by the gateway carries the identification of what content is requested in the current data collection round. The request message always refers to the entire area of interest (AoI).

In all simulations, the devices are randomly distributed in a fixed-size and square-shaped area. The link costs are proportional to the Euclidean distances between the devices. Each device is able to communicate with neighbors that are within the communication radius. The network gateway is fixed at the central point of the AoI. Table 1 shows the parameters used in all simulations.

TABLE I
Simulation parameters.

Parameter	Value
Number of network devices	100, 200 and 300
Number of contents per device	1
Number of contents in the network	4
Device initial energy	20 Joules
Energy Consumption (Transmission)	9.72 μJ/Byte
Energy Consumption (Reception)	8.22 μJ/Byte
Area of Interest	50 m x 50 m
Gateway coordinates	(25, 25)
Device communication range	15 m
Mobile Agent initial size	200 Bytes
Sensed data size	12 Bytes
Expiration time for cached content	30, 60, 90, and 120 seconds

We compare our proposal in all simulations with the baseline approach, i.e., we compare the Agent-Knap with the traditional MA (TMA) which collects data only from the set of requested source nodes. The Agent-Knap, on the other hand, considers gathering data of different content types available at intermediate nodes. Simulations computed three different metrics: Devices Average Energy Consumption, Cache Hits and Content Popularity Sensitivity.

All data gathering cycles are triggered when the sink node receives a request and does not have all the data requested on its cache. Such requests are received with an arrival rate that follows a Poisson distribution with $\lambda = 5$. The probability distribution for request messages arrival is either uniform or Zipf. All experiments are performed with 1,000 requests sent to the sink node at each round. For each point in the simulation results, 10 runs are performed. We plot a confidence level of 95%.

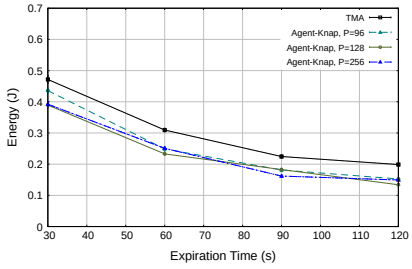
A. Energy consumption

This analysis computes the average energy consumption of all nodes in the network using both data gathering methods: the Traditional MA (TMA) and the Agent-Knap. These energy consumption values are the same as those adopted in [12] to evaluate *Tmote Sky* sensors.

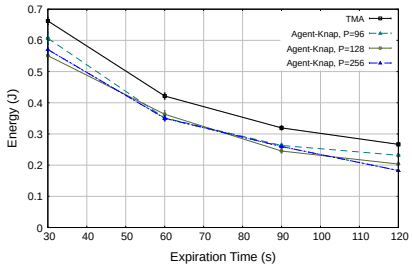
The analysis is performed for sets of five and seven source nodes. In both cases, we use P values of 96, 128, and 256-Byte. Figures 4(a) and 4(b) show the relationship between the average network energy consumption and the expiration time of contents stored in the sink cache, for a network with 100 devices.

For both experiments, the obtained result shows that the Agent-Knap has lower average energy consumption for all cache expiration times. With five source nodes, the Agent-Knap presented, for payload size of 256 Bytes and expiration time of 90 seconds, an average energy consumption of 161 mJ; whereas the TMA consumes 224 mJ, a reduction of 28.12%. With seven source nodes, the highest gain of the Agent-Knap is achieved with 256-Byte payload and 120 seconds cache expiration interval. At this point, the average consumption of the TMA was 266 mJ; whereas the Agent-Knap consumes 182 mJ, which is a 31.57% reduction on the energy consumed.

We observe a tradeoff between the MA payload size and the data freshness in the sink cache. Even though using larger payload sizes leads to higher transmission expenditures in terms of energy consumption, it reduces the number of times a MA is dispatched to the network. Consequently, we do not observe a relevant impact using different payload sizes.



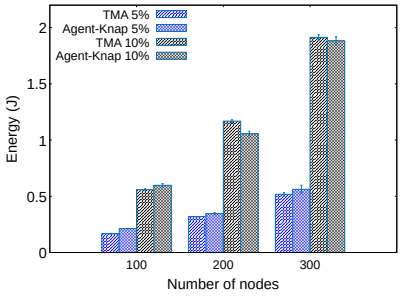
(a) 5 source nodes.



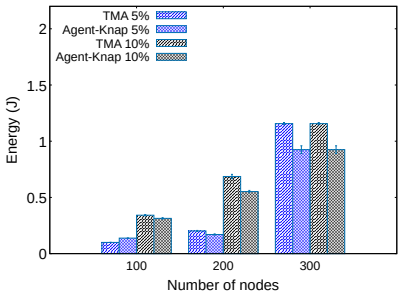
(b) 7 source nodes.

Fig. 4. Network average energy consumption for different cache expiration times.

Figures 5(a) and 5(b) show the average energy consumed in the network when we change the number of devices. The analysis is performed for scenarios with different number



(a) 60 seconds cache expiration time.



(b) 120 seconds cache expiration time.

Fig. 5. Network average energy consumption for different cache expiration times and network sizes.

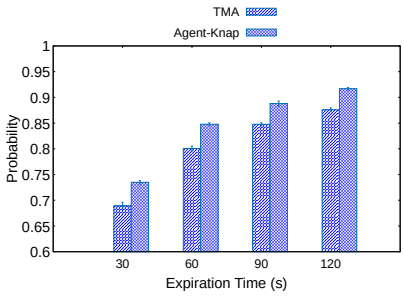
of source nodes. We use 5% and 10% of source nodes in proportion to the total number of nodes in the network.

Figures 5(a) and 5(b) show results for simulations with cache expiration time of, respectively, 60 and 120 seconds. In both Figures, energy expenditure increases with network size because we have more source nodes for data collection. For the 60-second cache expiration time, the Agent-Knap proposal performs better on 200 and 300 node topologies for 10% source node groups. For the 120 second cache expiration time, Agent-Knap performs better in all simulations except the 100 node topology with 5% source node groups. These results confirm a better efficiency of the Agent-Knap proposal in scenarios with longer cache expiration time, where the opportunistically collected data can be used to answer a larger number of requests sent by the network clients.

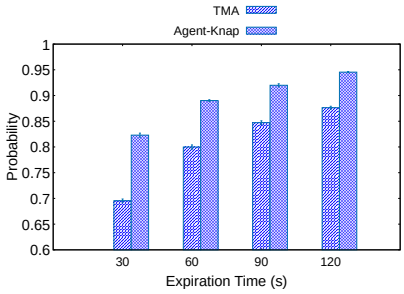
B. Cache hits

The goal of the cache-hit evaluation is to verify the performance of the proactive strategy used by the sink cache. A cache hit is obtained when a requested data is answered by the sink with the information available in its cache. For example, if a request for temperature readings from seven source nodes is received and the sink cache has updated information of only four devices, four cache hits are counted. Note that, in this case, the partial cache hit will still require the MA dispatch to the network to collect data from three source nodes. The analysis is conducted using 96- and 256-Byte payload sizes.

Figures 6(a) and 6(b) present the results for, respectively, 96 and 256-Byte payload sizes. We can observe that the Agent-Knap performance improves with larger payload (P) sizes. With 96-Byte P, the best result with Agent-Knap is 4.72% better than with TMA with a 60-second cache expiration time. With 256-Byte P, the best result with Agent-Knap is 12.74% better than with TMA with 30-second cache expiration time.



(a) 7 source nodes, P = 96 Bytes.



(b) 7 source nodes, P = 256 Bytes.

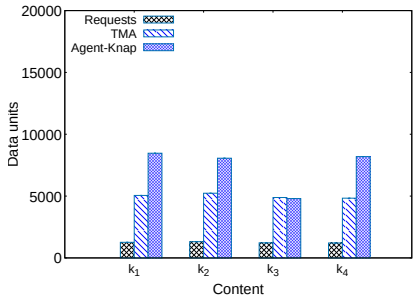
Fig. 6. Cache-hits: Comparison between TMA and Agent-Knap data collection mechanisms for different expiration times for cache content.

C. Content popularity sensitivity

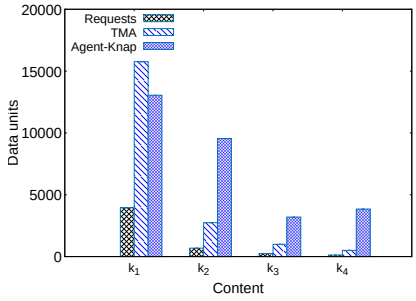
This experiment verifies the performance of the proposed system in a scenario where the contents distributed in the IoT network have different popularity. We consider that the popularity of a content is directly related to the past requests for that content. Hence, we conduct simulations for uniform and Zipf distributions. The metric used to verify the content popularity is the amount of information units collected in the network. Each information unit corresponds to a 12-Byte data sent by each device to the sink in response to a request. For small values of α, a higher correlation is expected between the content popularity and the amount of information units collected from the network for the same content. For all experiments with Zipf distribution, the $s_{zipf} = 2.0$ parameter is used. Since the goal is to analyze the sensitivity of the data collection strategy, the cache is not considered, i.e., data is not stored in cache. The Agent-Knap performance is compared with TMA for two values of α, 1 and 0.001. In all experiments, four source nodes are considered. For the Agent-Knap, C is simulated with the size of 60 Bytes.

Figures 7(a) and 7(b) present the results for $\alpha = 1$. For each content, the bars represent the amount of requests received and the information units received in both TMA and Agent-Knap. When α has a value closer to 1, there is a small correlation between the content popularity and the amount of information

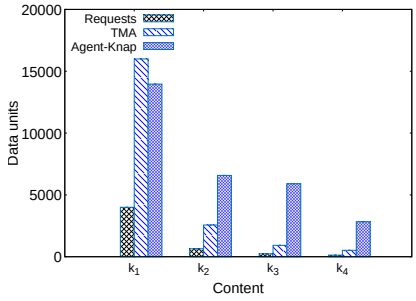
units collected for the same content. This behavior changes to lower values of α. In Figure 7(c) a higher correlation can be seen, when the Zipf distribution is used and an α of 0.001. Results also show that the amount of information collected is higher for TMA when content is more popular. This is explained by the distribution of the contents in the itineraries. If a calculated itinerary has ten contents, 2 k_1, 2 k_2, and 6 k_4, even if k_1 content is the most popular, the Knapsack optimization will fill the C space with many units of k_4. That is, the fact that k_1 is not the most densely distributed content in the network, despite being the most requested, results in a more balanced collection of content in the Agent-Knap proposal. Depending on the application requirements and desired caching strategy, this feature can be positive as it makes the collection process more diverse without violating the calculated ordering of priorities for the content.



(a) $alpha$ = 1, Uniform Distribution.



(b) $alpha$ = 1, Zipf Distribution.



(c) $alpha$ = 0.001, Zipf Distribution.

Fig. 7. Content popularity sensitivity: $alpha$ = 1 and $alpha$ = 0.001.

V. Conclusion and Future Work

We proposed the Agent-Knap, a new mechanism for data collection in LLN networks using mobile agents with static itineraries. Agent-Knap improves the efficiency of network communication by reducing the number of requests sent to the network by using opportunistic data collection for proactive caching update. Our simulation results show the potential of the collection mechanism by reducing the energy consumption in the network. As far as we know, this is the first work to propose the use of a greedy Knapsack algorithm to perform opportunistic data collection in a LLN network with the use of mobile agents. The association of content priority with their popularity allowed the implementation of an intelligent data gathering process, prioritizing the items of major interest. As future work, we intend to dynamically change the payload space reserved for opportunistic data, apply data aggregation to reduce the payload size, and include a selection process for source nodes considering upper-layer QoS requirements. We also plan to implement the Agent-Knap in a real testbed and evaluate its performance in a lossy scenario.

Acknowledgement

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. It was also supported by CNPq, FAPERJ, and FAPESP Grants 15/24494-8 and 15/24490-2.

References

[1] J. Lu, L. Feng, J. Yang, M. M. Hassan, A. Alelaiwi, and I. Humar, "Artificial agent: The fusion of artificial intelligence and a mobile agent for energy-efficient traffic control in wireless sensor networks," *Future Generation Computer Systems*, vol. 95, pp. 45–51, 2019.

[2] S. Yousefi, F. Derakhshan, and A. Bokani, "Mobile agents for route planning in internet of things using markov decision process," in *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*. IEEE, 2018, pp. 303–307.

[3] Y.-C. Chou and M. Nakajima, "A clonal selection algorithm for energy-efficient mobile agent itinerary planning in wireless sensor networks," *Mobile Networks and Applications*, vol. 23, no. 5, pp. 1233–1246, 2018.

[4] H. Qadori, Z. Zulkarnain, Z. Hanapi, and S. Subramaniam, "A spawn mobile agent itinerary planning approach for energy-efficient data gathering in wireless sensor networks," *Sensors*, vol. 17, no. 6, p. 1280, 2017.

[5] Y. Zuo and J. Liu, "A reputation-based model for mobile agent migration for information search and retrieval," *International Journal of Information Management*, vol. 37, no. 5, pp. 357–366, 2017.

[6] B. Liu, J. Cao, J. Yin, W. Yu, B. Liu, and X. Fu, "Disjoint multi mobile agent itinerary planning for big data analytics," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 99, 2016.

[7] D. Gavalas, I. E. Venetis, C. Konstantopoulos, and G. Pantziou, "Mobile agent itinerary planning for WSN data fusion: considering multiple sinks and heterogeneous networks," *International Journal of Communication Systems*, vol. 30, no. 8, p. e3184, 2017.

[8] S. Aguilar, R. Vidal, and C. Gomez, "Opportunistic sensor data collection with bluetooth low energy," *Sensors*, vol. 17, no. 1, p. 159, 2017.

[9] Y. Zhan, Y. Xia, J. Zhang, and Y. Wang, "Incentive mechanism design in mobile opportunistic data collection with time sensitivity," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 246–256, 2018.

[10] G. Xu, E. C.-H. Ngai, and J. Liu, "Ubiquitous transmission of multimedia sensor data in internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 403–414, 2018.

[11] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon University, Pittsburgh, Tech. Rep., Feb. 1976.

[12] Y. Jin, S. Gormus, P. Kulkarni, and M. Sooriyabandara, "Content centric routing in IoT networks and its integration in RPL," *Computer Communications*, vol. 89, pp. 87–104, 2016.