Optimized Flow Assignment in a Multi-Interface IoT Gateway

Mohamed Ghazi Amor, Kim Khoa Nguyen, Chuan Pham, Mohamed Cheriet
École de Technologie Supérieure, University of Quebec, Canada
Email: {mohamed-ghazi.amor.1, kim-khoa.nguyen, chuan.pham.1, mohamed.cheriet}@etsmtl.ca

***Abstract*—The last few years have witnessed a significant increase in the deployment of heterogeneous Internet of Things (IoT) networks. IoT devices send data with different requirements such as tolerated delay and data rates. Emerging multi-interface IoT devices bring the flexibility of connecting to multiple heterogeneous access networks, which thus improves the network capacity. However, each network interface has its own constraints in terms of network coverage, capacity, packet loss rates, etc. An efficient utilization of the available multiple interfaces in IoT gateways would improve the network performance. Therefore, it is crucial to design a flow assignment mechanism to select the appropriate interface that best satisfies the flow's requirements and maximizes the amount of data accepted by an IoT gateway. In this work, we model and formulate the optimized flow assignment problem (OFAP) in a multi-interface IoT gateway. Then, we develop two heuristic algorithms to find a feasible solution for OFAP. The first algorithm is based on the greedy approach and the second uses dynamic programming to assign flows to interfaces. We provide simulation results that show the effectiveness of our algorithms.**

I. Introduction

Over the past decade, Machine to Machine (M2M) networks have witnessed evolution in their communication schemes [1]. Nowadays, Internet of Things (IoT) devices are often equipped with multiple interfaces. These smart objects send and receive data through different communication protocols such as Zigbee, WiFi, Bluetooth, LoRa and 3G to a multi-interface gateway [2]. This heterogeneity brings the advantage of accessing to multiple network resources which may make an improvement in the network performance and avoid congestion. However, each communication technology has its own specifications. Although available resources are often limited, IoT devices may be called on to provide low-latency guarantees to deadline sensitive applications that are very common in IoT use cases [3]. An example of a deadline sensitive application is an emergency medicine service with low-latency constraints associated with body sensors or patient health devices [4].

As a gateway has multiple network interfaces, IoT devices may not deliver data to a proper interface without an efficient mechanism that assigns the IoT flows to available interfaces while satisfying the application's requirements. The lack of an assignment mechanism may lead to congestion, in particular when the number of devices using the same interface increases. Consider an example of a plant monitoring application [5] which monitors large scale green spaces by collecting information such as soil moisture, humidity, wind, pH, and CO_2. This application sends pictures of leaves to a cloud server using a high-definition camera to allow experts to detect plant diseases. In this scenario, the IoT device is equipped with two different interfaces WiFi and Zigbee. The flows are assigned statically by the providers of end devices to the network interfaces. For this example, the soil humidity, temperature, and CO_2 measures are assigned statically to the Zigbee interface because it consumes less energy than WiFi. As for picture data, it will be assigned to WiFi interface since it requires much more bandwidth. However, the network traffic can vary and more services can be deployed, or new devices can join the network. As a result, Zigbee can be overloaded overtime. Therefore, a static assignment of the flows can cause network congestion and higher delay. Moreover, when traffic is assigned statically to interfaces, this may cause saturation of certain interfaces despite the availability of other interfaces which can handle more traffic. To the best of our knowledge, this issue has been investigated in prior research like [6], [7] and [8], which propose resource allocation and service to interface assignment to ensure the quality of service and to optimize the energy consumption of IoT devices. However, they do not take into account the deadline requirements of applications.

In this paper, we propose a flow assignment mechanism that maximizes the amount of valid data accepted by the gateway, which is the data received within its respective deadline. We perform this maximization by assigning the flows to the suitable network interface that satisfies not only the deadline and data rate requirements of flows but also the energy capacity of devices. Our contributions can be summarized as follows:

- We introduce a mathematical model that characterizes the available resources across multi-interface end devices and models the exchanged flows between the IoT devices and the multi-interface gateway. We define the optimized flow assignment problem (OFAP) to maximize the amount of valid data accepted by the gateway, taking into account the available network resources and satisfying other flow requirements.
- We design a switching mechanism between gateway interfaces that enables a seamless handover and ensures the continuity of the communication while switching.
- We propose two algorithms to solve the optimized flow

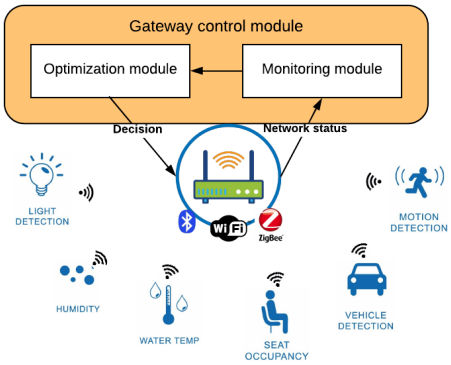


Fig. 1. System architecture.

assignment problem and we show their effectiveness by calculating the acceptance rate of flows while varying different parameters.

The remainder of the paper is organized as follows. In section II, we present the system description. Section III introduces a motivation example and our problem formulation. In section IV, we present the proposed algorithms to solve the OFAP problem. Finally, we present the performance evaluation and draw the conclusion.

II. System description

In this work, we consider a scenario of an IoT system consisting of one gateway, which is equipped with multiple network interfaces such as Bluetooth, WiFi, and Zigbee as described in Fig. 1. This gateway communicates with a set of IoT devices that serve different types of services. IoT devices are equipped with at least one network interface. We assume that the signal strength is strong enough to connect to the gateway and that time is divided into small duration time slots in which devices send discrete data. The flows sent at each time slot are known. In our problem, we consider assigning flows sent within one time slot. Suppose that each flow should be received within its respective deadline which is the tolerated delay for each IoT flow. In this work, we aim at assigning a set of given flows, transmitted in the same time slot, to the available network interfaces while maximizing the amount of valid data received by the gateway.

Regarding a dense IoT-enabled network where the areas are typically overlapped, the network can be overloaded and the traffic can be distributed with an unbalanced way among network interfaces. Hence, the gateway in our system integrates a control module as detailed in Section II-A. This module controls the loads coming to the gateway and migrates flows from a heavily-loaded interface to a lightly-loaded one.

A. Gateway control module

Our solution performs this flow assignment by executing the following modules:

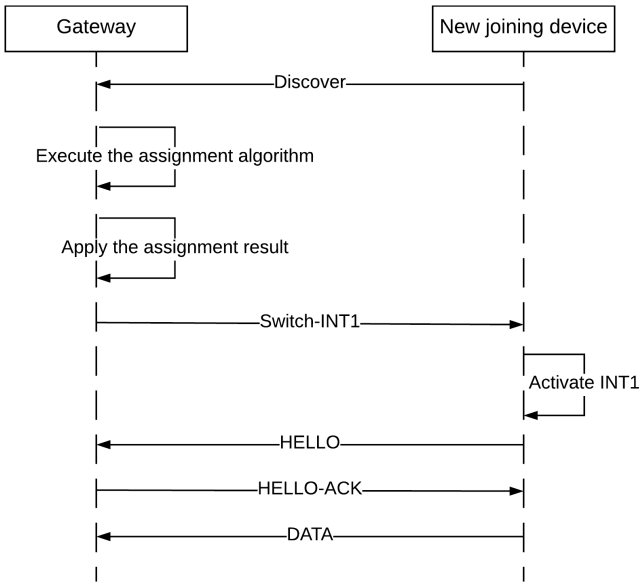


Fig. 2. Switching mechanism.

1) Monitoring module: It collects information about the IoT devices and its services such as the data rates and service delay requirements. This information is used then by the optimization module. Using this information, the gateway can build its global view of the candidates that are initiating communication and willing to join the network.

2) Optimization module: After executing the monitoring module, the gateway control module will then, assign IoT devices to one of its interfaces using the optimization module while meeting the deadline of flows, the capacity of the gateway, and the energy capacity of devices. The objective of this module is to maximize the amount of valid data assigned to the available interfaces of the gateway. It takes as input the total set of devices, their flow sizes, and the set of available network interfaces and it is responsible for selecting the best network interface that satisfies the flows' requirements. The optimization module may require from an IoT device to switch from one interface to another. Therefore, we design a switching mechanism as described in the following section.

B. Switching mechanism

For switching between interfaces, we design a protocol that enables a seamless handover between network interfaces. This protocol is described in the sequence diagram as shown in Fig. 2. When a new device joins the gateway, first it sends a *DISCOVER* message to notify the gateway with information such as the device ID, the available network interfaces within the device and the data rate of the service. Then, it executes the assignment algorithm. The result of this algorithm optimally assigns the device to the gateway's interface while taking into account the defined constraints (e.g., deadline of the flow and the gateway's capacity). A device is unable to connect

if gateway's capacity is exceeded. In that case, it will receive *NOT AUTHORIZED* message. The devices which are required to switch from one interface to another will be notified by a *SWITCH-InterfaceID* message. It is followed by a set of *HELLO* and *HELLO-ACK* messages to establish communication using the new assigned interface. After receiving the *HELLO-ACK* packet, the connection is now established and the device can start sending data to the gateway through the selected network interface.

III. Optimized Flow Assignment Problem

In this section, we define, formulate and discuss the Optimized Flow Assignment Problem (OFAP) which is an optimization problem. It aims at assigning flows dynamically to the gateway's network interfaces while satisfying the deadline requirements of services, the capacity of the network interfaces, and the battery's capacity of devices.

A. Motivation example

This section presents a motivation example which highlights the importance of using a mechanism that exploits the available network interfaces of a gateway efficiently. We consider 4 flows sent by 4 different IoT devices. The input of this example is summarized in Table I.

Our architecture is composed of the following components:

TABLE I
Scenario Input

Flows	f_1	f_2	f_3	f_4
Flow size (in KB)	80	20	50	20
Transmission delay using interface 1	8	2	5	2
Transmission delay using interface 2	16	4	10	4
Deadline	10	4	15	9

- An IoT gateway that has 2 interfaces (interface 1 and interface 2) with respectively 10 KB/s and 5 KB/s of transmission rates.
- 4 IoT devices that are equipped with 2 interfaces each. Each device sends a single flow (e.g., Device 1 sends flow f_1 at $t = 0$ using interface 2 as described in Fig. 3).

At $t = 0$, devices 1, 2, 3, and 4 send their respective flows to the gateway. We define the deadline of one flow, D, as the limit time after which data sent by the flow is expired or invalid. In other words, any flow should be sent by the IoT device and received by the gateway within its deadline. When the flows are sent in the same time slot and assigned to the same interface, the flow with the earliest deadline is scheduled to be transmitted first. Let's take the example of the IoT device 1 in Fig. 3 that sends a flow f_1 of 80 KB which takes 8 seconds to send it through interface 1 and 16 seconds to send it through interface 2. This flow should be received before its deadline which is equal to 10th second. If f_1 is assigned to interface 2, its deadline will be missed.

Here, the flows are assigned to the interfaces statically since there is no a dynamic mechanism that will optimize the flow assignment among interfaces. Moreover, as traffic can vary significantly over time, the assignment and reassignment of flows to interfaces should be done dynamically. Therefore, we need an assignment mechanism that optimizes the network resource utilization.

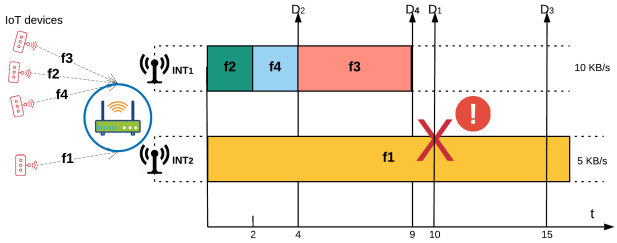


Fig. 3. Scenario of sending 4 flows to a multi-interface IoT gateway.

B. Problem formulation

In this section, we denote the set of IoT devices by $\mathcal{I}$ having multiple interfaces. We denote the set of flows by $\mathcal{F} = \{f_i\}_{\forall i \in \mathcal{I}}$ where f_i is the flow sent by IoT device i. In our work, we assume that each device i sends a single flow f_i. The size of the flow f_i is equal to S_i. Each IoT device has a battery capacity which is equal to C_i. We denote the energy consumption per second for transmitting data through interface j by E_j. Let $\mathcal{J}$ be set of all available network interfaces within the gateway. Each interface j has a transmission rate T_j. The network capacity of interface j is equal to R_j. A is an input matrix with $|\mathcal{I}|$ rows and $|\mathcal{J}|$ columns, representing the available network interfaces within IoT devices. Device i is equipped with interface j if $A_{i,j} = 1$, otherwise $A_{i,j} = 0$.

We define the decision binary variable $x_{i,j}$ as the following:

$$x_{i,j} = \begin{cases} A_{i,j}, & \text{if flow } f_i \text{ is assigned to the gateway's} \\ & \text{interface } j, \\ 0, & \text{otherwise.} \end{cases}$$

to represent the assignment of flows to the gateway's interfaces. The notations are summarized in Table II.

Objective function. The objective of OFAP is to maximize the amount of valid data accepted by the gateway while satisfying the deadline, energy, and network capacity constraints. It can be formulated as follows:

$$\max \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} S_i \, x_{i,j} \tag{1}$$

Constraints. The objective function defined in Eq. 1 is subject to the following constraints:

1) Energy capacity: This constraint ensures that when flow f_i is transmitted through interface j, its energy consumption should not exceed the available energy of the device's battery C_i. This energy consumption is calculated by multiplying the transmission delay with the energy consumption per second E_j of interface j.

$$\sum_{j \in \mathcal{J}} \frac{S_i}{T_j} E_j \, x_{i,j} \leq C_i, \quad \forall i \in \mathcal{I}. \tag{2}$$

2) Deadline of flows: Each flow f_i should be received within its respective deadline D_i. We define three delays for each flow f_i: switching delay $\theta_{i,j}$, transmission delay $\beta_{i,j}$ and buffering delay $\alpha_{i,j}$.

Switching delay is the time required to switch from the current connected interface to interface j and the time of the establishment of connection which is described in section II-B. We define $\theta_{i,j}$ as the switching delay of device i to switch to interface j from the current connected interface. As for transmission delay, it is equal to the size of the flow, S_i, divided by the transmission rate T_j. Hence, $\beta_{i,j}$ is defined as follows:

$$\beta_{i,j} = \frac{S_i}{T_j}, \quad i \in \mathcal{I},\ j \in \mathcal{J}. \tag{3}$$

The buffering delay $\alpha_{i,j}$ represents the transmission delay of the flows stored in the buffer. This delay is the sum of two terms: (a) the transmission delay of Q_j which is the amount of the remaining data in buffer that is already transmitted during the previous time slot through interface j. (b) The second term is the delay of transmitting the flows that came at the same time slots as f_i and that are scheduled to be transmitted before f_i. In our work, we assume that flows with the earliest deadline are scheduled to be transmitted first when they are sent in the same time slot and assigned to the same interface. $\mathcal{I}'$ denotes the set of devices that sends flows with a deadline earlier than f_i where $\mathcal{I}' = \{i' \in \mathcal{I} \,/\, \{i' \neq i\}$ and $D_{i'} < D_i\}$. Hence, $\alpha_{i,j}$ is defined as follows:

$$\alpha_{i,j} = \frac{Q_j}{T_j} + \sum_{i' \in \mathcal{I}'} \frac{S_{i'}\, x_{i',j}}{T_j}, \quad i \in \mathcal{I},\ j \in \mathcal{J}. \tag{4}$$

The deadline constraint guarantees that the deadline requirements of the flow is respected. When flow f_i is sent through interface j, the delay of switching $\theta_{i,j}$ plus the delay of transmission $\beta_{i,j}$ using interface j plus the buffering delay $\alpha_{i,j}$ should not exceed the deadline D_i of f_i.

$$\sum_{j \in \mathcal{J}} \theta_{i,j}\, x_{i,j} + \sum_{j \in \mathcal{J}} \beta_{i,j}\, x_{i,j} + \sum_{j \in \mathcal{J}} \alpha_{i,j}\, x_{i,j} \leqslant D_i, \quad \forall i \in \mathcal{I}. \tag{5}$$

3) Flow assignment: We assume that flows are not split over different interfaces. This constraint guarantees that flow f_i is transmitted through a single interface.

$$\sum_{j \in \mathcal{J}} x_{i,j} \leqslant 1, \quad \forall i \in \mathcal{I}. \tag{6}$$

4) Network capacity: This constraint ensures that the amount of data assigned to the interface j does not exceed the network capacity R_j of that interface.

$$\sum_{i \in \mathcal{I}} S_i\, x_{i,j} \leq R_j, \quad \forall j \in \mathcal{J}. \tag{7}$$

OFAP is an Integer Non linear Programming (INLP) problem, due to constraint (5). Since the complexity of INLP solvers is high, we propose two algorithms for solving the OFAP problem in the next section.

TABLE II
NOTATION LIST

General Inputs	
$\mathcal{I}$	Set of IoT devices
$\mathcal{J}$	Set of the available network interfaces within the gateway
$\mathcal{F}$	Set of flows
T_j	Transmission rate of the network interface j
Q_j	The amount of the remaining data in buffer (in MB)
D_i	Deadline of the flow f_i
C_i	Energy capacity of the device i
E_j	Energy consumption per second for transmitting data through interface j
R_j	Network capacity of interface j
S_i	Size of flow f_i
A	Input matrix representing the available network interfaces within IoT devices.
Flow delays	
$\theta_{i,j}$	Delay to switch flow i from the current connected interface to interface j (in seconds)
$\alpha_{i,j}$	Buffering delay of flow f_i when using interface j
$\beta_{i,j}$	Transmission delay of flow f_i when using interface j
Variables	
$x_{i,j}$	Binary decision variable indicating that flow f_i is assigned to interface j

IV. PROPOSED ALGORITHMS

In this section, we design two algorithms for our optimized flow assignment problem (OFAP).

A. G-OFAP: A greedy approach for OFAP

This approach is based on the knapsack problem. We model each interface as a knapsack. The capacity of each knapsack is equal to the resource capacity of the network interface. We consider the flows as the objects to be added to the knapsack. We model the weight of a flow as the amount of data in MB. In our algorithm, the network interfaces are sorted according to their network capacities and the set of flows is sorted according to their sizes. Then, we apply the best fit strategy. In other words, the algorithm assigns a flow to a network interface which has the smallest sufficient capacity among the available interfaces. Furthermore, the selected interface should satisfy the defined constraints in section III-B. However, if a flow is assigned to no interfaces, it will be dropped. The algorithm iterates on all interfaces and stops when all flows are visited.

B. D-OFAP: A dynamic programming-based algorithm for OFAP

In order to solve our flow assignment problem in polynomial time, we propose a second solution based on dynamic programming which is described in algorithm 1. This solution performs the flow assignment based on a work array W calculated using a dynamic programming algorithm which gives an exact solution for a simple knapsack problem. So the idea is to find an optimal subset of the set of flows. This subset maximizes the amount of data assigned to a particular

interface. Then, we apply the rest of the constraints on this subset to produce a valid result.

Algorithm 1 D-OFAP algorithm

Input: $\mathcal{I}$, $\mathcal{F}$, $\mathcal{J}$, Network capacities R_j, Deadline of flows D_i, Battery capacities C_i, Energy consumption E_j
Output: Assignment matrix X.

1: **sort** set of interfaces by their capacities R_j.
2: $j \leftarrow 0$
3: $X \leftarrow 0$
4: **while** $j \leq |\mathcal{J}|$ and $\mathcal{F} \neq \emptyset$ **do**
5: **compute** the work array W based on flow sizes S_i and R_j.
6: **update** W after applying the constraints on all the subsets associated to each sum of W.
7: **selects** a subset $F \subset \mathcal{F}$ with the biggest sum of flow sizes from W.
8: **if** subset $\neq \emptyset$ **then**
9: $\mathcal{F} \leftarrow \mathcal{F} \setminus F$
10: **update** the assignment matrix X based on the new subset.
11: **end if**
12: $j \leftarrow j + 1$
13: **end while**
14: **return** X

This algorithm takes as input the set of devices $\mathcal{I}$, set of flows $\mathcal{F}$, the set of network interfaces $\mathcal{J}$ and the characteristics of flows and interfaces. The output of the algorithm is a binary matrix X, with $|\mathcal{I}|$ rows and $|\mathcal{J}|$ columns, which represents the flow interface assignment.

First, this algorithm sorts interfaces according to their network capacities R_j. Then, for each interface, it calculates the work array using dynamic programming (line 5). W contains all feasible subsets of flows that can be assigned to the current processed interface. At line 6, we apply the deadline and energy constraints on all the subsets of W. After that, the algorithm selects a subset of devices that has the biggest sum of flow sizes from W (line 7). If there is a feasible subset for the interface j, the generated subset will be subtracted from the set $\mathcal{F}$ (line 9). Then, the assignment matrix will be updated and the subset of flows is assigned to the current processed interface. This algorithm stops when all interfaces are processed or when all devices are assigned.

V. Performance Evaluation

In this section, we perform simulation experiments to evaluate the performance of the two proposed algorithms *G-OFAP* and *D-OFAP*.

A. Simulation setup

In our simulations, we consider an architecture that consists of one IoT gateway and we vary the number of IoT devices from 10 to 80. Each of these IoT devices sends discrete data. The flow sizes are randomly generated. Each IoT device is equipped with multiple interfaces. We consider four types of communication technologies: WiFi, Bluetooth, Zigbee and Z-Wave, with transmission rates 5 Mbps, 2 Mbps, 240 Kbps and 100 Kbps respectively. In our simulations, we consider two types of IoT smart city applications [9]: (a) low data rates applications that can tolerate deadlines up to 2 min (e.g., Smart Parking, Air quality monitoring) and (b) Critical applications that can tolerate deadlines up to 20 seconds (e.g., alarms for emergency situations, health-care applications).

We compare the results of G-OFAP and D-OFAP with the optimal ones which are obtained by solving the optimization problem by using ILOG CPLEX[10]. These scenarios simulated the flows coming from IoT devices at a certain time slot by varying the following parameters:

1) Amount of generated data (Scenario 1): In this scenario, the number of generated flows is fixed to 80 and the number of interfaces per device is fixed to 3. We vary the total amount of generated data. For each generated set of flows, we calculate the acceptance rate of valid data which is equal to total amount of accepted data divided by the total amount of generated data.

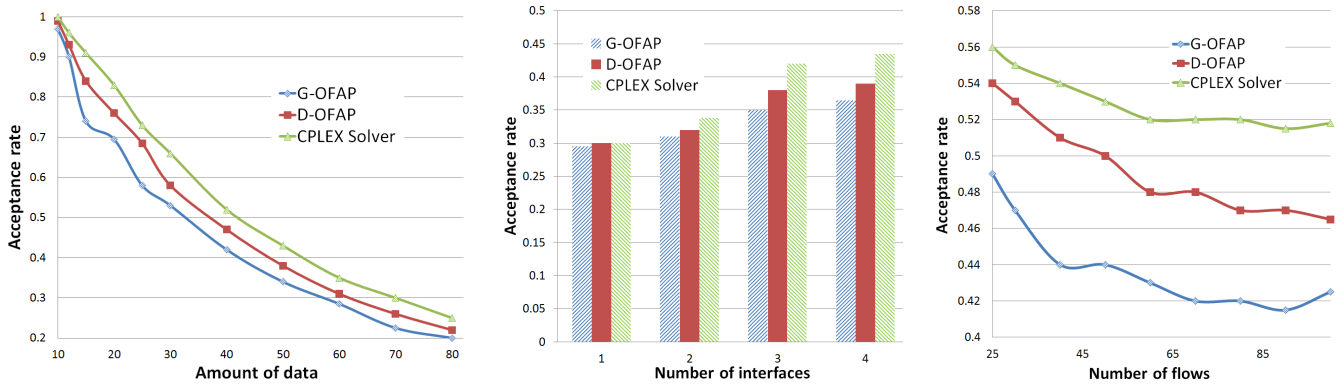
2) Number of interfaces (Scenario 2): In this scenario, we fixed the sum of flows to 75 MB and the number of generated flows is equal to 80. We perform our simulations with different interfaces and we vary the number of interfaces per device.

3) Network size (Scenario 3): Here, we investigate the impact of varying the network size. The number of interfaces per device is fixed to three interfaces and we vary the number of devices that send flows to the gateway. IoT devices generate flows which sum is equal to 40 MB.

B. Simulation results

1) Amount of generated data (Scenario 1): Fig. 4a describes the results of the scenario 1 simulations performed to gain insight into the performance of our proposed solutions. It shows the acceptance rate of valid data when varying the amount of data coming from IoT devices to the gateway. At the beginning, when the amount of generated data is between 10 MB and 15 MB, the acceptance rate of G-OFAP and D-OFAP algorithms varies between 0.9 and 1 which is comparable to the optimal solution. However, when the network load increases, we observe that D-OFAP has better acceptance rates values compared to the greedy algorithm. For example, when the IoT devices are generating data of 25 MB, D-OFAP achieves 0.69 of acceptance rate which is closer to the optimal solution value i.e., 0.73 than the G-OFAP. The reason behind the decreasing acceptance rate values is that the size of flows is getting bigger. For this reason, satisfying the deadline requirements of all flows is not possible. Therefore, the gateway is obliged to drop the invalid flows which cannot satisfy their deadlines.

2) Number of interfaces (Scenario 2): Fig. 4b shows the acceptance rate while varying the number of interfaces. We can observe that D-OFAP ensures an optimal solution in the case where the gateway and devices are equipped with only one interface. This is because our algorithm is based on dynamic programming which gives an exact solution for the knapsack problem. When we increase the number of interfaces, we can



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 4. Effects of varying the amount of generated data, the number of interfaces and the number of flows on acceptance rate values.

observe that the acceptance rate increases since the capacity of handling more traffic increases as well. The proposed algorithms closely approximate the optimal solution. However, the optimal solution is costly in terms of time. The elapsed time for solving the problem using CPLEX solver is much higher than the D-OFAP and G-OFAP algorithms.

3) Network size (Scenario 3): Fig. 4c shows that D-OFAP outperforms the greedy approach. Note that the acceptance rate of valid data decreases when we increase the number of flows. This is because more and more flows are allocating the network resources and no resources are left for the unassigned flows. The algorithm cannot find a free network resource that can satisfy the deadline requirement of the device's flow. However, we can observe the acceptance rate doesn't vary too much when increasing the number of flows since the amount of generated data is fixed.

In summary, as observed in the results of the presented scenarios, whereas the D-OFAP has significantly better performance than the greedy algorithm, about 5%, it is outperformed by the optimal solution. It must be noted that despite that the difference between acceptance rate of D-OFAP and optimal solution varies between 4% and 7%, complexity of the optimal solution is much higher than D-OFAP.

VI. Conclusion

In this paper, we address the optimized flow assignment problem in a multi-interface IoT gateway. We introduced a mathematical formulation that formally defines our problem and we developed two algorithms that exploit the available interfaces of the gateway efficiently while satisfying the applications' requirements. Experimental results show that the proposed algorithms G-OFAP and D-OFAP produce promising results for different scenarios in terms of the acceptance rate of the gateway.

In future work, we will extend our research to study the flow assignment problem, where multiple gateways are considered and flows of IoT devices can switch between multi-interface gateways. The gateway control module will be implemented within a centralized server which will fetch information about network status from all gateways before sending the flow assignment decision.

Acknowledgment

The authors thank NSERC and Ericsson for funding the project CRDPJ 469977. This research also receives support from the Canada Research Chair, Tier 1, held by professor Mohamed Cheriet. We also thank SARA at ÉTS for their writing support with this article.

References

[1] T. Taleb and A. Kunz, "Machine type communications in 3gpp networks: potential, challenges, and solutions," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 178–184, March 2012.

[2] C.-T. Chang, C.-Y. Chang, R. Dario Borja Martinez, P.-T. Chen, and Y. Chen, "An iot multi-interface gateway for building a smart space," *Open Journal of Social Sciences*, vol. 03, pp. 56–60, 01 2015.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[4] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadline-aware task scheduling in a tiered iot infrastructure," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[5] M. Stoes, J. Vank, J. Masner, and J. Pavlk, "Internet of things (iot) in agriculture - selected aspects," *AGRIS on-line Papers in Economics and Informatics*, no. 665-2016-45107, p. 6, Mar 2016. [Online]. Available: http://ageconsearch.umn.edu/record/233969

[6] V. Angelakis, I. Avgouleas, N. Pappas, E. Fitzgerald, and D. Yuan, "Allocation of heterogeneous resources of an iot device to flexible services," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 691–700, Oct 2016.

[7] C. Tsai and S. Liu, "An effective iot service-to-interface assignment algorithm via search economics," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1708–1718, June 2018.

[8] M. Ismail and W. Zhuang, "A distributed multi-service resource allocation algorithm in heterogeneous wireless access medium," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 2, pp. 425–432, February 2012.

[9] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[10] "Cplex: Ibm's linear programming solver." [Online]. Available: http://www.ilog.com/product/cplex/