Edge Computing for Visual Navigation and Mapping in a UAV Network

Mohamed Ayoub Messous, Hermann Hellwagner, Sidi-Mohammed Senouci, Driton Emini, Dominik Schnieders

Edge Computing for Visual Navigation and Mapping in a UAV Network

Mohamed Ayoub Messous[1], Hermann Hellwagner[2], Sidi-Mohammed Senouci[1], Driton Emini[3], Dominik Schnieders[3]
[1]DRIVE EA1859, Univ. Bourgogne Franche Comté, France
[2]Institute of Information Technology, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
[3]Deutsche Telekom AG, Germany
{ayoub.messous, sidi-mohammed.senouci}@u-bourgogne.fr, hermann.hellwagner@aau.at, {driton.emini,dominik.schnieders}@telekom.de

***Abstract*—This research work presents conceptual considerations and quantitative evaluations into how integrating computation offloading to edge computing servers would offer a paradigm shift for an effective deployment of autonomous drones. The specific mission that has been considered is collaborative autonomous navigation and mapping in a 3D environment of a small drone network. Specifically, in order to achieve this mission, each drone is required to compute a low latency, highly compute intensive task in a timely manner. The proposed model decides for each task, while considering the impact on performance and mission requirements, whether to (i) compute locally, (ii) offload to the edge server, or (iii) to the ground station. Extensive simulation work was performed to assess the effectiveness of the proposed scheme compared to other models.**

Keywords—UAV Network, Edge Computing, Computation Offloading, Visual Navigation and Mapping.

I. Introduction

Unmanned aerial vehicles (UAVs), also commonly named drones, have proliferated extremely fast and continue to have far-reaching effects on today's society, transforming our lives and the way we do business. While big UAV platforms have been initially introduced for strategic and defense applications, relatively small and very flexible new UAV platforms are proving to be very useful for a wide variety of applications such as disaster damage assessment, law enforcement surveillance, and earth science data collection [1-4]. In order to accomplish their missions, UAVs are required to (i) collect data about their environment, (ii) compute and process the collected data for decision making, and (iii) relay specific information to their appropriate users [5-6]. To achieve these three basic requirements, depending of their mission, UAV platforms are equipped with different types of sensors, onboard computing capabilities and dedicated wireless communication interfaces. This would be essential for effective special awareness, realization of locally distributed processing along the communication path where applicable, and finally, for the dissemination of relevant information from producers to consumers while maintaining reliable and efficient air-to-air (UAV-to-UAV) and air-to-ground (UAV-to-ground station) communications [7].

In this research work, we choose to focus on a specific mission where a small fleet of drones is deployed in order to achieve a collaborative autonomous 3D navigation and reconstruction task [8-9]. More precisely, using only inertial information, taken from IMU readings, and visual information, through captured imagery, the drones will be required to collectively map and efficiently navigate in an unknown 3D environment [10]. Technically, a process called visual inertial odometry is used [11]. Similar to other vision-based localization and navigation methods, this process is prone to drift errors while estimating positions and attitudes of drones due to non-perfect sensory reading [12-13]. These drift errors would eventually build up, resulting in degrading flight and mapping accuracy. IMU data readings at a 1 kHz rate and image capturing at 30 images per seconds (30 Hz) would be ideal to keep drift errors as small as possible. However, the constrained computation and limited communication capabilities of drones would not allow such high sampling rates since the preprocessing and fusion of the sensor data in general and the image processing tasks in particular are computationally intensive.

To address issues related to highly intensive computation tasks with low latency requirements, many previous research studies have resorted to computation offloading to powerful surrogate devices such as mobile edge computing (MEC). This newly emerging computation paradigm extends the capabilities of cloud servers to the edge of an access network. It provides computational power and data storage closer to the end users. Therefore, it can reduce traffic loads for central cloud servers, decrease latency, and improve QoS. Moreover, it supports users' mobility and heterogeneity, provides location awareness, and permits high deployment scalability. MEC is perceived as one of the central building blocks for 5G networks [14-15]. For drone networks, the effective deployment of MEC, on the one hand, holds the promise of a high-throughput communication fabric, and on the other hand, is expected to enable low-latency capabilities and to perform specific computational services for the client devices [16-20]. In our case, drones would consider the offloading of some or all of the computational tasks involved in visual inertial odometry to an edge server. Indeed, MEC can not only bring performance benefits for heavy computational tasks, it can also act as a centralized entity for the nearby deployed swarm of drones. Moreover, the MEC servers deployed at the edge of the access network would help control the drones more accurately compared to decentralized control. They would even be expected to play an important role in synchronizing and ensuring the consistency of the mapping tasks.

The main focus of the current study is to investigate how MEC can conceptually be integrated into a specific mission using a drone network, while evaluating the impact of computation offloading on the flight accuracy and the possible performance benefits of deploying an MEC-based scheme. The specific mission considered is the visual-inertial navigation and 3D mapping in an unknown remote environment. In order to tackle these challenges, the proposed solution aims to decide whether a computational task needs to be offloaded to be executed remotely or not.

The remainder of this paper is organized as follows. In Section II, we present the system model and the mathematical formulation of our problem. Section III gives the details about the main computation offloading schemes. The simulation work and the results obtained are presented and discussed in Section IV. Finally, Section V concludes the paper and gives some future directions.

II. System Model and Problem Formulation

The system model, shown in Fig. 1, is composed of three main entities: (i) drones, (ii) edge server, and (iii) ground station. In the following subsections, we first present the mission requirements and justify our hypothesis. Then, we provide the problem formulation while explaining in detail the different possible computational use case scenarios.

A. Mission Requirements and Scenario Definition

We consider a set of small UAVs deployed in an unknown environment in order to execute a 3D mapping of their environment. Specifically, we consider as a main mission the collaborative autonomous 3D navigation and reconstruction application through a small network of drones, using inertial information (IMU data) and visual information (captured images) only (no GNSS information). The drones will reconstruct (map) an unknown 3D environment and navigate in that space. In order to perform such kind of mission, the UAVs are equipped with an onboard camera for visual imagery and also an inertial measurement unit (IMU) that provides basic information about their spatial orientation. We also consider for our use case scenario that the drones in this mission do not have access to GNSS positioning data. The lack of such information can be explained by the poor quality of GNSS signal or its absence altogether (in indoor environment) or due to problems related to precision (in a city district with a very dense building constellation).

In such a context, the drones are required to perform a process named *visual inertial odometry* (VIO) in order to achieve their mission. Similar to other vision-based localization and navigation methods, this process is prone to drift errors in estimating positions and attitudes of drones in case of non-perfect sensor data; these drift errors will result in degrading flight and mapping accuracy. IMU data readings at a 1 kHz rate and image capturing at 30 images per seconds (30 Hz) would be ideal to keep drift errors as small as possible. However, the constrained computation and communication capabilities of small drones would not be able to handle such high sampling rates since the processing and fusion of the sensor data in general and the image processing tasks in particular are highly compute intensive. Due to these limitations, computation offloading from the drones to an edge server would offer a viable solution.

Computational tasks involved in VIO can be fully or partially offloaded from the drones to an edge server. On the one hand, MEC would achieve better performance compared to local computing. On the other hand, an edge server would eventually act as a nearby deployed central entity for its associated drones, which would allow for an even more accurate control compared to a fully decentralized scheme, and would also ultimately perform the synchronization and mapping tasks more consistently.

B. Computational Models

As previously shown, the kind of missions considered involves performing highly intensive computation tasks. Each task T_i is defined through three basic parameters $\{C_i, D_i, F_i\}$ representing computational complexity, size of data and execution frequency, respectively. The first value C_i corresponds to the number of *CPU* cycles required to perform the task T_i. D_i specifies the amount of data needed for the computation. Finally, F_i denotes the execution rate, i.e., how many times the task T_i is called upon per time unit.

Fig. 1. Overall view of the system model

Furthermore, the defined tasks can be either executed locally in the drone itself or can be eventually fully or partially offloaded to a surrogate, more powerful device if required. In the present study two offloading choices are possible: (i) through a cellular network towards an edge server, or (ii) through a WLAN access towards a nearby ground station. Therefore, three possible choices can be enumerated, namely: (i) local computing, (ii) offloading to edge, and (iii) offloading to ground station. Details for each use case are provided in the following paragraphs.

1) Local Computing

Since computation tasks in this case are executed locally, no actual data ought to be sent via wireless interfaces. Therefore, the overhead of the task is equal to local computation overhead. The latter would only be impacted by the onboard computation power available in the device, i.e., the CPU frequency of the drones which is the number of computation cycles per time unit. So, the execution time for a task T_i if the local CPU frequency is F^{Local}_{CPU} is given as:

$$T_{Local} = C_i / F^{Local}_{CPU} \quad (1)$$

2) Offloading to Edge

In this second case, the drone would send its computation task via its cellular interface towards the edge server. Compared to the previous option, the delay required to obtain results for the task being executed, in addition to the computation time, will incur an extra overhead. This is due to the additional time necessary to transmit data up to the edge server. Therefore, the equation for the execution time is:

$$T_{Server} = C_i / F^{Server}_{CPU} + D_i / R_{Cellular} \quad (2)$$

where F^{Server}_{CPU} represents the frequency of the server CPU, which in practice is very big compared to the frequency of the mobile devices' CPUs, and $R_{Cellular}$ is the effective data rate achieved through the cellular link between the drone and the edge server.

3) *Offloading to Ground Station*

This second offloading choice, and third possible case, considers sending the computational data through a wireless access point to a neighboring ground station. The latter would compute the received task and send back results to the originating drone. In this case, the equation for the execution time is given as:

$$T_{GS}=C_i/F^{GS}_{CPU}+D_i/R_{WLAN} \quad (3)$$

where F^{GS}_{CPU} denotes the CPU's frequency of the ground station and R_{WLAN} is the effective data rate achieved through the wireless local network.

C. Overhead Function

Since computationally intensive tasks are known to necessitate a considerable amount of time to complete their execution, we define the overhead function as the combination of time delays required for data transfer and for data processing. Both, the communication links' effective data rates and the available processors' frequencies will play a significant role in choosing the more suitable computation choice as shown previously. Therefore, it is important to define and implement an appropriate utility function that considers the best possible tradeoff between these two competing parameters. The following equation shows the different parameters required to calculate the overall values for the overhead function for each task.

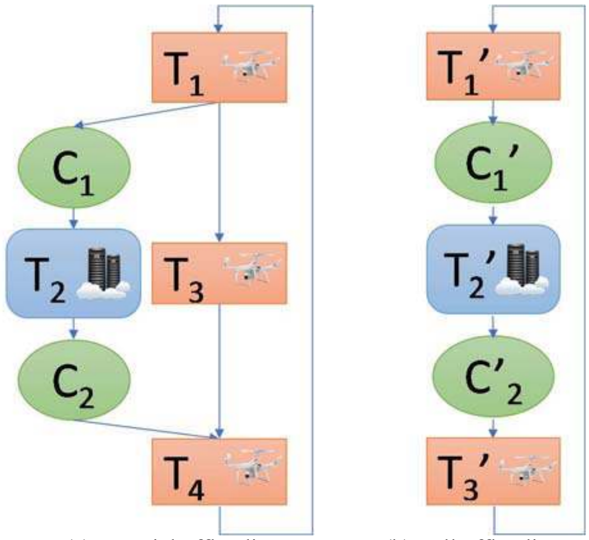
$$\boldsymbol{O_{verhead}} = \boldsymbol{\alpha} \sum_{i=1}^{N} \boldsymbol{T_i^{Computation}} + \boldsymbol{\beta} \sum_{j=0}^{M} \boldsymbol{T_j^{Communication}} \quad (4)$$

where N is the number of computation sub-tasks required to compute the global task; $\boldsymbol{T_i^{Computation}}$ represents the execution time (computational overhead) required for processing each sub-task i; M characterizes the number of communication exchanges required; and $\boldsymbol{T_j^{Communication}}$ stands for time delay (communication overhead) to transfer/receive each message j. Additionally, α and β represent weight parameters of computational and communication delays, respectively, and α+β =1.

Moreover, no normalization method was required in order to add these two different values since both represent time measurements. Furthermore, using a weighted function provides a much higher flexibility and answers a wide range of applications with specific requirements. Specifically, depending on the envisioned application or even the current system status, different tasks might use different weight parameters. For instance, if the system administrator wishes to put more emphasis on the computational delay part, he or she would give higher values for the weight α, whereas β would be increased in order to highlight the importance of communication delay.

III. Offloading Computation for Visual Navigation

We provide herewith details regarding our implementation for possible choices of computation offloading in a visual navigation mission. Besides the classical approach, where all the computation is executed locally, we consider different task splitting possibilities. Furthermore, each global task can be divided into several elementary sub-tasks. There is always some computation that needs to be treated locally using the embedded processors. Typically, this process might involve the following operation: (i) collecting raw data from different onboard sensors, (ii) compression of raw data into a standard data format, (iii) aggregation of similar data within the same vector, and (iv) multi-sensor fusion. Parts of these low-level operations need to be executed locally and would not be suitable for offloading. However, other high-level operations can be offloaded, especially those including intensive computation routines. Figure 2 provides two examples for a partial-offloading and a full-offloading scenario. Details for each model are given in the following subsections.



(a) Partial offloading (b) Full offloading

Fig. 2. Possible offloading scenarios

A. Partial Offloading

In this first case, a first part of the collected data is computed locally before sending the results of execution along with specific information for further treatment in the edge server. As shown in Fig. 2(a), the partial offloading scheme starts with a locally executed task T_1, then data are sent from the drone to the edge server through the communication link, denoted as communication step C_1. Then, when all the required data are received, the edge server executes task T_2 which ends with communicating results back to the drone (C_2). Meanwhile, other local treatment would continue to be executed on the drone, which is denoted in Fig. 2(a) as task T_3. Finally, when C_2 is completed, local processing can be resumed in T_4 taking into account the newly received data.

For the requirements of our use case scenario, the detailed execution routines for the different tasks are given in the following. First, task T_1 involves (i) IMU data reading, (ii) attitude and acceleration deduction, and (iii) IMU data preintegration. T_3 repeats the same instructions as in T_1 but adds at the end of each cycle a "new pose estimation" routine. Data transferred in C_1 encompasses: (i) the latest key frame image and (ii) up-to-date IMU data. The data sent in C_1 will serve as input for T_3, where the following operations are executed: (i) feature detection, (ii) feature tracking, and (iii) pose estimation and optimization. C_2 will incorporate data about the newly estimated pose. Finally, the drone would focus in T_4 on updating and optimizing the new pose graph based on inputs from C_1 and T_3.

B. Full Offloading

This second case considers that only essential computation, which cannot be offloaded to edge, would be treated locally onboard the drone. All the other actions would be executed remotely on the edge server, then results are sent back to the drone as shown in Fig. 2(b). The details for each elementary action are provided as follows. T_1' includes sensor data reading and compression. Only basic computation is achieved before sending gathered data in C_1'. This step encompasses offloading a continuous flow of images taken by the onboard camera and inertial readings from the IMU. The edge server will do all the computations required for the mission in T_2' before sending back the new commands and instructions for the drone to follow in C_2'. Finally, T_3' represents the control commands and updating the local pose estimate graph for the drone.

IV. Performance Evaluation

Extensive simulation work has been done to evaluate the effectiveness of the proposed computation offloading scheme. In this section, the detailed assumptions regarding simulation work and scenario definition are first introduced. Then, numerical results are presented followed by thorough discussions to validate the feasibility and effectiveness of an edge-based solution for visual navigation missioned by a small set of drones.

A. Simulation Setup

In the simulation scenarios, we consider a single cell consisting of a base station (plus edge server) and a small number of drones (between 1 and 10) which also have access to a ground station (GS) via WLAN. The computation tasks considered for visual navigation differ in their computation complexity and also in their size of data required for effective computation. On the one hand, computation complexity is mainly affected by the number of features considered and the type of algorithm used for feature extraction. Since the main focus of the current study is the computation offloading problem and optimized decision making, only the number of features has been considered in representing computational complexity for different tasks. On the other hand, data that would need to be offloaded comes from imagery sensors and the IMU. However, practical estimation shows that data size required for IMU readings are negligible compared to image resolution, even with very high IMU reading rates. For simplicity, as shown in Table 1, three complexity levels are tested (between 50 and 200 features to extract from an image) with three different image resolutions (360p, 480p and 720p). Finally, in order to assess the possible offloading choices explained in Sect. III, different offloading rates have been implemented: ranging from local computing (0%) to full offloading (100%), along with three intermediary offloading rates (25%, 50% and 75%). Furthermore, beside the edge server a second surrogate device, namely the ground station, is considered as a second possible destination choice for offloading computation through a WLAN link (Fig. 1).

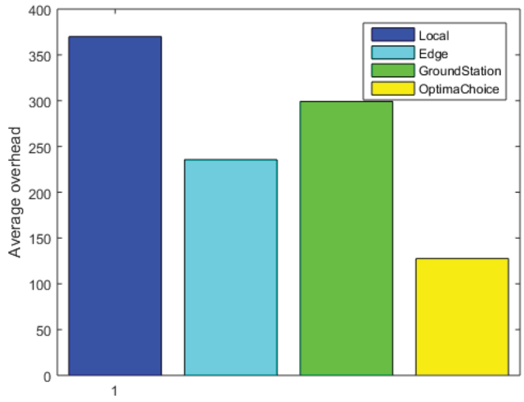


Fig. 3. Evaluation of global average overhead

TABLE I. Test Scenarios

Test Scenarios	
# of UAVs	[1, 5, 10]
# of Features	[50, 100, 200]
Image Resolution	[360p, 480p, 720p]
Edge Offload Rate (%)	[0, 25, 50, 75, 100]
GS Offload Rate (%)	[0, 25, 50, 75, 100]

To evaluate each model, we consider the global utility function presented in Sect. II.C. Furthermore, we give an equal importance to computation and communication delays, i.e., we choose $\alpha = \beta = ½$. For a detailed study of the impact that number of UAVs, number of features and image resolution might have on the overall utility, all the different possible combinations have been considered and tested. Other parameters used in our simulation setup are summarized in Table II. For the sake of simplicity, we consider the processing power of the ground station ($F^{GS}{}_{CPU}$) and the edge server ($F^{Edge}{}_{CPU}$) to be respectively five and ten times the frequency of CPU available onboard the drone ($F^{Local}{}_{CPU}$). As for achievable data rates, the average data rates for wireless links with the ground station and with the edge server (R_{WLAN} and $R_{Cellular}$) are 50 and 20 Mbps, respectively.

TABLE II. Simulation Parameters

Simulation Parameters	*Values*
$F^{Local}{}_{CPU}$	1 GHz
$F^{Edge}{}_{CPU}$	10 GHz
$F^{GS}{}_{CPU}$	5 GHz
$R_{Cellular}$	20 Mbps
R_{WLAN}	50 Mbps

B. Results and Discussion

In this subsection, we first evaluate the global average overhead achieved by the proposed approach compared to the three other models. The diagram shown in Fig. 3 represents the average system-wide overhead achieved through: (i) locally executed tasks, (ii) fully offloaded tasks to the edge, (iii) fully offloaded tasks to the ground station, and (iv) the optimal choice, which uses equation (4) as a premise for

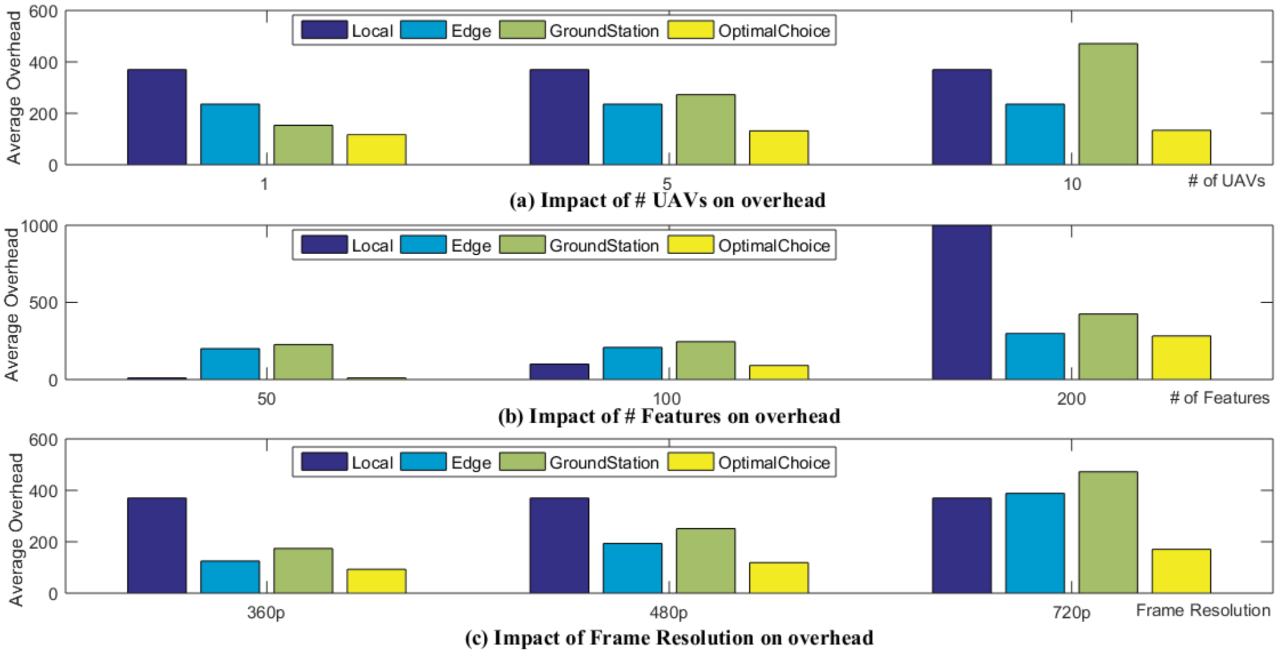


Fig. 4. Detailed evaluation of average overhead

decision-making. It reveals that the optimal selection approach clearly outperforms the three other models in terms of global overhead. This is due to the fact that the proposed model always chooses the most efficient offloading choice in terms of computational and communication delays.

Moreover, the impact that different simulation parameters might have on the performances was thoroughly investigated. First, results shown in Fig. 4(a) represent the average overhead achieved in scenarios with different number of UAVs. We can notice that the growth in UAV swarm size does not impact the performance of locally executed tasks and has a slight impact on offloading tasks to the edge. However, it mostly affects the overhead values when offloading to the ground station. Next, the impact of computation complexity, expressed as the number of image features to be extracted, was tested. Fig. 4(b) shows that adaptive optimal selection outperforms the three other models in terms of average system overhead. The obtained results also show that overhead values increase with higher computation complexity. However, they increase much slower for full offloading approaches compared to locally executed tasks. Even though local computing achieves better performance for tasks with lower numbers of features, edge offloading was better for tasks with a higher number of features. This means that local computing is most suitable for less intensive computation tasks (less than 100 features), whereas offloading to the edge is more appropriate for highly compute intensive tasks (more than 200 features). Finally, Fig. 4(c) shows that the average overhead increases as the data size, expressing the different frame sizes in our case, increases in the two offloading approaches, due to the fact that big data induce high transmission overhead. While for high resolution (720p) local computing achieves comparable results to edge offloading, for lower resolutions (360p and 480p) it is always more interesting to offload. However, adaptive optimal selection, on average, still outperforms all the other models for the three different frame sizes.

The previous results shown in Fig. 3 and Fig. 4 only consider binary offloading choice; either all the task is computed locally or fully offloaded. For in-depth analysis, Fig. 5 shows the different possible intermediate offloading rates. As explained in Sect. III, a global task would be divided into several elementary sub-tasks that can be executed on different devices. Three intermediate offloading possibilities have been considered, namely 25%, 50% and 75%. As shown in Fig. 5, on average, higher offloading rates towards the edge server always achieve better performances compared to lower offloading rates in terms of average overhead. This is mostly true for very compute intensive tasks, since performing even a small part of the computation on the drone would penalize the overall performance. However, this statement is not true for the ground station, where offloading 50% and 75% of the computational task is better than full offloading. This later case (100% offload rate to the ground station) produced a similar average overhead compared to 25% offload rate. This observation proves that computation offloading is not always better compared to local execution. It also shows that parallel execution, even on slightly powerful devices, can really achieve better performance. It should be finally noted that sending more data would require also more time, which would eventually have more effect on average overhead in the 100% offload to ground station case compared to the 75% and 50% cases.

V. Conclusion

The rapid emergence of UAV-related technologies attracted the focus of many research groups, which paved the way for new possible use case applications. Throughout this study, we consider the deployment of a small fleet of UAVs in a navigation and 3D mapping mission based on visual and inertial data, which is time sensitive, requires high refresh rates, especially for IMU readings, yet entails highly intensive computations (image preprocessing, feature extraction, etc.). Therefore, a new framework based on the

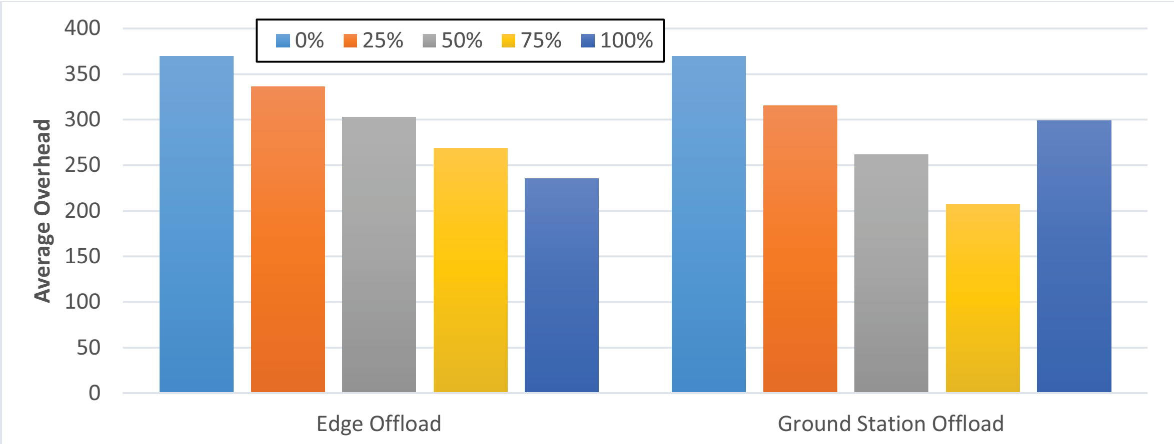


Fig. 5. Impact of different offloading rates on average overhead

MEC paradigm was introduced to better address these requirements. One of the main challenges we addressed is to conceptualize the effective integration of MEC into a specific flight mission. Then, we characterized different types of computational tasks to be offloaded from the drones to the edge servers. Moreover, we evaluated the impact of the proposed scheme on performance in different simulation settings and showed the possible benefits of the deployment of a drone network in a MEC environment for this specific mission. Simulation results show the effectiveness of the proposed model compared to other models.

As future work, we plan to further evaluate the impact that our computation offloading solution might have on the flight accuracy. We also plan to integrate in our model environment reconstruction and 3D mapping as a second use case.

Acknowledgments

This work was partially supported by Magenta Telekom (T-Mobile Austria GmbH) and Deutsche Telekom AG as well as by Alpen-Adria-Universität Klagenfurt (scholarship for visiting researcher M.A. Messous).

References

[1] L. Ma, M. Li, L. Tong, Y. Wang, and L. Cheng, "Using unmanned aerial vehicle for remote sensing application", 21st International Conference on Geoinformatics, pp. 1-5, 2013.

[2] M.A. Messous, H. Sedjelmaci, and S-M. Senouci, "Implementing an emerging mobility model for a fleet of UAVs based on a fuzzy logic inference system", Elsevier Pervasive and Mobile Computing, vol. 42, pp. 393-410, 2017.

[3] M.A. Messous, S.-M. Senouci, and H. Sedjelmaci, "Network connectivity and area coverage for UAV fleet mobility model with energy constraint," IEEE Wireless Communications and Networking Conference, pp. 1-6, 2016.

[4] N. Hossein Motlagh, T. Taleb, and O. Arouk, "Low-Altitude Unmanned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives", IEEE Internet of Things Journal, vol. 3, no. 6, pp. 899-922, 2016.

[5] G. Pajares, "Overview and Current Status of Remote Sensing Applications Based on Unmanned Aerial Vehicles (UAVs)", Photogrammetric Engineering & Remote Sensing, vol. 81, iss. 4, pp. 281–329, 2015.

[6] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, and M. B. Yagoubi, "A survey on position-based routing protocols for Flying Ad hoc Networks (FANETs)," Elsevier Vehicular Communications, vol. 10, pp. 29–56, 2017.

[7] J. Liu, Y. Shi, Z. M. Fadlullah and N. Kato, "Space-Air-Ground Integrated Network: A Survey," in IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 2714-2741, 2018.

[8] A.A. Ravankar, A. Ravankar, Y. Kobayashi, and T. Emaru, "Autonomous Mapping and Exploration with Unmanned Aerial Vehicles Using Low Cost Sensors". International Electronic Conference on Sensors and Applications, 2018.

[9] Q. Wang, "Towards Real-time 3D Reconstruction using Consumer UAVs", 28th Workshop on Information Technologies and Systems, 2018.

[10] Y. Lu, Z. Xue, G-S. Xia, and L. Zhang, "A survey on vision-based UAV navigation", Geo-spatial Information Science journal, 21:1, 21-32, 2018.

[11] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-Based Visual Inertial Odometry," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pp. 5816-5824, 2017.

[12] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A Fully Integrated Energy-Efficient Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," IEEE Symposium on VLSI Circuits, pp. 133-134, 2018.

[13] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," IEEE Journal of Solid-State Circuits, vol. 54, no. 4, pp. 1106-1119, 2019.

[14] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3098-3130, 2018.

[15] X. Zheng, M. Li, M. Tahir, Y. Chen, and M. Alam, "Stochastic Computation Offloading and Scheduling Based on Mobile Edge Computing," IEEE Access, vol. 7, pp. 72247-72256, 2019.

[16] M.A. Messous, S.-M. Senouci, H. Sedjelmaci, and S. Cherkaoui, "A Game Theory Based Efficient Computation Offloading in an UAV Network," IEEE Transactions on Vehicular Technology, vol. 68, no. 5, pp. 4964-4974, 2019.

[17] J. Zhang et al., "Stochastic Computation Offloading and Trajectory Scheduling for UAV-Assisted Mobile Edge Computing," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3688-3699, 2019.

[18] T. Bai, J. Wang, Y. Ren, and L. Hanzo, "Energy-Efficient Computation Offloading for Secure UAV-Edge-Computing Systems," IEEE Transactions on Vehicular Technology, vol. 68, no. 6, pp. 6074-6087, 2019.

[19] M.A. Messous, A. Arfaoui, A. Alioua, and S.-M. Senouci, "A Sequential Game Approach for Computation-Offloading in an UAV Network", IEEE Global Communications Conference, 2017.

[20] M.A. Messous, S.-M. Senouci, and H. Sedjelmaci, "Computation Offloading Game for an UAV Network in Mobile Edge Computing", IEEE International Conference on Communications, 2017.