Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Telecommunication Engineering

# Allocation of Computing and Communication Resources for Mobile Edge Computing with Parallel Processing

Bc. Martina Matějková

Supervisor: doc. Ing. Zdeněk Bečvář, Ph.D.
Field of study: Communication Systems and Networks
May 2019

# MASTER'S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Matějková Martina** Personal ID number: **409978**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Telecommunications Engineering**

Study program: **Electronics and Communications**

Branch of study: **Communication Systems and Networks**

## II. Master's thesis details

Master's thesis title in English:

**Allocation of Computing and Communication Resources for Mobile Edge Computing with Parallel Processing**

Master's thesis title in Czech:

**Alokace výpočetních a komunikačních prostředků pro Mobile Edge Computing s paralelním zpracováním**

Guidelines:

Study principles of communication and computing resource allocation for processing of users' applications at the edge of mobile network. Develop an algorithm allocating communication and computing resources to multiple users considering their requirements on an overall latency. Assume possibility of a parallel processing of the users' applications at multiple edge servers. Also, consider a possibility to deliver the offloaded application via any available communication path. Test a performance of the proposed solution by means of simulations.

Bibliography / sources:

[1] J. Cao, L. Yang, and J. Cao, "Revisiting Computation Partitioning in Future 5G based Edge Computing Environments," IEEE Internet of Things journal, July 2018.
[2] J. Liu, Q. Zhang, "Offloading Schemes in Mobile Edge Computing for Ultra-Reliable Low Latency Communications," IEEE Access, February 2018.
[3] X. Chen, L.J Pu, L. Gao, W. Wu, D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," IEEE Wireless Communications, August 2017.
[4] J. Plachy, Z. Becvar, P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," Computer Networks, 2016.

Name and workplace of master's thesis supervisor:

**doc. Ing. Zdeněk Bečvář, Ph.D., Department of Telecommunications Engineering, FEE**

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **04.02.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

| doc. Ing. Zdeněk Bečvář, Ph.D.<br>Supervisor's signature | Head of department's signature | prof. Ing. Pavel Ripka, CSc.<br>Dean's signature |
|---|---|---|

## III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

# Acknowledgements

I would like to thank my supervisor doc. Ing. Zdeněk Bečvář, Ph.D. for his guidance and encouragement. My thanks also belong to Ing. Janu Plachý for his helpful suggestions.

I am also deeply grateful to my family and to Michal Skhut for their support and patience.


# Declaration

I hereby declare that I have completed this thesis on my own and that I have only used the cited sources.

I have no objection to use of this work in compliance with the act "§60 Zákon č. 121/2000 Sb." (copyright law), and with the rights connected with the copyright act including the changes in the act.

Prague, 24. May 2019

# Abstract

In the fifth generation (5G) mobile networks, new use cases and applications with strict requirements for latency emerge. Mobile Edge Computing (MEC) is a novel concept, which supports the offloading of computationally demanding tasks to the edge of the mobile network, and is considered a promising solution to reduce the latencies. The parallel processing of the task in the MEC system aims to further minimize the task's completion delay. Although the problem of parallel processing in the MEC has received attention among researchers, the existing works either assume a single-user scenarios, or focus on partitioning of the computation resources at the edge. In this thesis, a multi-user scenario is considered, with users offloading the partitioned tasks sequentially to the selected clusters of computing eNBs. An algorithm is proposed for the optimal task partitioning and resource allocation. The efficiency of the proposed algorithm is then simulated and compared to other existing approaches. The proposed algorithm decreases the task completion delay by up to 48% when compared to another method exploiting parallel processing and by up to 78% in comparison with a non-partitioning methods.



**Keywords:** Mobile Edge Computing, task partitioning, parallel processing, resource allocation

**Supervisor:** doc. Ing. Zdeněk Bečvář, Ph.D.



# Abstrakt

Mobilní sítě páté generace (5G) přináší množství nových užití a aplikací s přísnými požadavky na latence. "Mobile Edge Computing" (MEC) jakožto nový koncept, který podporuje přenos výpočetně náročných úloh na okraj mobilní sítě, je považován za řešení pro snížení latencí. Paralelní zpracování úloh v MEC systému má za úkol dále snížit celkový čas výpočtu. Přestože problému paralelního zpracování v MEC systémech se dostalo mezi vědci mnoho pozornosti, existující řešení se zaměřují na scénáře s jedním uživatelem, případně na dělení výpočetních prostředků na samotném okraji mobilní sítě. Tato diplomová práce předpokládá systém s více uživateli, kteří sekvenčně odesílají rozdělené úlohy přímo na klastr vybraných základnových stanic s výpočetními prostředky. Je navržen algoritmus pro optimální dělení úloh a alokaci prostředků. Efektivita navrženého algoritmu je pomocí simulací porována s existujícími řešeními. Navržený algoritmus snižuje celkový čas výpočtu až o 48% při porovnání s další metodou využívající paralelního zpracování a až o 78% ve srovnání s metodou bez paralelního zpracování.



**Klíčová slova:** "Mobile Edge Computing", dělení úloh, paralelní zpracování, přidělování prostředků

**Překlad názvu:** Alokace výpočetních a komunikačních prostředků pro Mobile Edge Computing s paralelním zpracováním

# Contents

# Figures

# Chapter 1

# Introduction

In the recent years, the number of mobile devices has grown rapidly, increasing the number of connections and overall mobile data traffic, as well as new use cases and applications. Many real-time applications such as virtual reality, augmented reality, video streaming, or gaming have emerged, creating additional demands on both the computing capabilities and battery lifetime of mobile devices. However, given the restrained size of the mobile devices, limited computation resources and high battery consumption still pose an obstacle. By offloading computationally demanding applications from the user equipment (UE) to the cloud, i.e. Mobile Cloud Computing (MCC) [1], battery lifetime can be prolonged significantly. The offloaded applications are usually composed of a front-end component running on the device and a back-end component that runs on the cloud, with a Virtual Machine (VM) created over the allocated resources [2]. However, this architecture introduces additional network overhead and backhaul load. Moreover, long distances in between the cloud and the device can result in a high latency and an error probability.

To enable superior user experience, 5G aims at improving operational performance, i.e. low latency, high bandwidth, and increased spectral efficiency [3]. One of the fundamental technologies to be deployed in 5G is Multi-Access Edge Computing (MEC) [4], also known as Mobile Edge Computing [5], a new paradigm, which supports the offloading of computationally demanding tasks to the edge of the mobile network, i.e. to base stations (eNBs), small cell eNBs (SCeNB), remote radio heads, etc. Moving computation resources closer to the user (in a sense of network topology) leads to lower communication delays between the UE and its allocated computing resources, as well as to a decrease in the eNB' backhaul load, thus further evolving the concept of MCC [6].

The computation offloading follows two possible models: i) the full offloading model, when the device offloads the whole task to the MEC, and ii) partial offloading, which is conditioned by the application's capability to be split into two or more parts. Depending on its ability to be partitioned, the application can be either offloaded to a single node, or to several nodes. Partitioning of a computationally intensive tasks and their parallel processing has been receiving much attention from reseachers from all over the world in the recent

years, optimizing the whole process of task offloading from the decision on task offloading to user movement enhanced by mobility prediction. Liu *et al.* propose a sequential offloading of partitioned subtasks to a selected cluster of eNBs in order to minimize delay while meeting reliability constraints in [7]. Oueis *et al.* analyze latency and power consumption dependency on the size and topology of the computing cluster in [8]. In [9], the authors propose a method for parallelizing the computations at the MEC servers in order to fully utilize the computational resources.

Along with optimizing network performance and/or improving Quality of Experience (QoE) in order to meet the strict requirements of 5G [10], MEC introduces plenty of new use cases. These are not only the services for consumers and operators, but also third parties applications with possible MEC implementation in Internet of Things (IoT), Intelligent Transport Systems (ITS), mobile big data analytics, software-defined networks (SDNs), etc. [11] [12].

This master's thesis focuses on finding an effective solution for partitioning of the task and allocating the resources for the computation of individual subtasks in MEC in multi-user environment. The main objective is to minimize delay of task completion. A heuristic algorithm is developed, which iteratively allocates available resources, namely bandwidth to the edge and the computation power of the edge servers. The algorithm performs initial partitioning of generated tasks and adjusts it continuously until finding a suitable allocation for all the tasks. The proposed algorithm is compared to other state of the art approaches, improving the performance by reducing the completion delay of the tasks.

The rest of this work is organized as follows. Firstly, related work on task partitioning and offloading, and resource allocation in MEC systems is presented. Chapter 3 introduces the system model. Chapter 4 is dedicated to the problem formulation. In chapter 5, the algorithm for the joint computation partitioning and resource allocation is proposed and explained. Then, in chapter 6, the environment and simulation models are presented, followed by simulation results and discussion. Finally, chapter 7 concludes the thesis and outlines the future work.

# Chapter 2

# Related Works

This chapter focuses on related work in the area of computation offloading, communication and resource allocation for processing of user's applications at the edge of mobile network, as well as task partitioning.

MEC is a new paradigm with the purpose of enabling user to exploit the computation capabilities at the network edge. It is currently being developed by European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG) and is perceived as one of the key technologies for 5G networks and Software-Defined Networking (SDN) [5].

The research in MEC can be divided into several areas [11]. First, a decision on the computation offloading to the MEC based on the determined profit for the UE in terms of execution delay and/or energy consumption. Second, in case the computation is offloaded, an efficient allocation of the computing resources. And finally, mobility management to guarantee service continuity during the UE's movement.

## 2.1 Decision on computation offloading

The decision on computation offloading is based on the determined profit for the UE in terms of execution delay and/or energy consumption, or trade-off between both. Possible outcomes of this decision are:

- local execution on the UE,
- full offloading to the MEC considered in [13], [14], [15],
- and partial offloading, where the task is split into a set of subtasks that are either offloaded to the edge server or computed locally, such as in [9], [16], [17], [18].

Local execution is profitable in case of poor channel quality between the user and its serving base station as in these conditions the energy used for transmission exceeds the energy saved on offloading. It is for this reason the application requiring the offload of a larger amount of data should be computed locally. Moreover, the UE computes its task locally in case of unavailable computing resources at the MEC server.

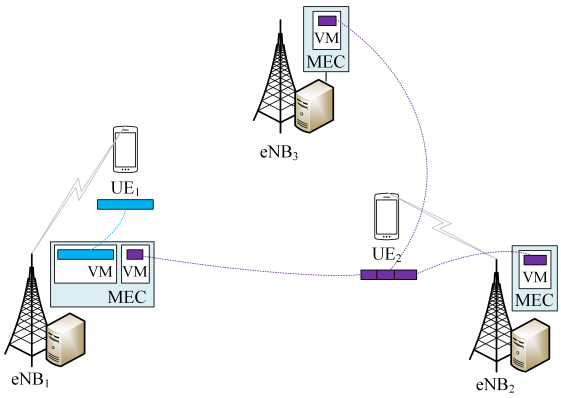



**Figure 2.1:** An example of computing resource allocation in the MEC system

Full offloading on the contrary is beneficial in case of a good channel quality between the UE and the eNB. The offloading is the most efficient in case of computationally demanding application with only a small amount of data to transfer, thanks to the gain between the energy spent on transmission and energy saved on computation [11].

Partial offloading is conditioned by the application's capability to be parallelized. Such applications can be divided into two types: i) app that can split into multiple parts that are all offloadable, ii) app that comprises both offloadable and non-offloadable (e.g. user input, camera, and result display [16]) parts. Depending on its ability to be partitioned, the application can be either offloaded to a single node, or to multiple nodes.

It is important to state that the offloadable parts of a program and their dependencies need to be identified before the offloading decision is made [19].

## 2.2 Allocation of resources

Once the decision on either full or partial offloading is done, communication and computation resources are allocated in the form of VMs. In case the offloaded application enables partitioning, the task can be offloaded to several nodes and processed in parallel (see Figure 2.1, where the $UE_2$ offloads an application that is partitioned and computed at all three eNBs). On the contrary, if the application does not allow parallelization, only one eNB can be used for computing its task (as can be seen in Figure 2.1, where the $UE_1$ offloads only to the $eNB_1$).

For the purpose of offloading to MEC in real-time applications, VM assigned to an UE should be started and prepared for computing when the task is being offloaded, otherwise the delay from establishing and starting the VM would render the application unusable.

In order to design an energy-efficient MEC system, communication and computation resources have to be allocated jointly as proven in [13].

The joint allocation of resources is adopted in [17], where a system is presented with one computing eNB and multiple users generating identical

tasks. The authors connect weighted sum energy consumption minimization problem to a three stage flow-shop scheduling problem and Johnson's algorithm [20]. The objective is to obtain a sub-optimal task operation sequence for processing all tasks on each machine with the minimum total completion time.

Authors in [21] introduce task partitioning in a multi-user system, where the UEs offload their tasks via one eNB to multiple servers in the MEC. They develop a heuristic method, which divides the network bandwidth into several virtual channels. For each user, the algorithm generates an initial partitioning and execution schedule, and then iteratively searches for the first occurence of constraint violation, either during transmission or computation. In order to avoid this violation, the algorithm adjusts the user's original partitioning accordingly.

All the above-mentioned works focus on offloading to one eNB only. Partitioning a task and offloading it to multiple nodes can lead to further decrease in execution delay, and possibly energy and power consumption of computing eNBs. In such a scenario, the selection of computing nodes is vital.

Authors in [8] analyze latency behavior and power consumption in relation to the size and topology of computing cluster. The cluster for particular UE's task is formed from servers deployed at eNBs in the UEs vicinity. The optimal number of computing nodes in the cluster is closely related to the computing data load. In case the offloaded application is computationally demanding, inclusion of more eNBs in the cluster will lead to a decrease in overall delay. On the contrary, with low computation demands of a task, adding extra computing nodes will have little or no affect on the latency, since the computation delay is already low and the main component in this case is the transfer delay.

In [7], authors present a situation, where one UE transmits a partitioned task to multiple nodes in sequence using the full channel bandwidth. In order to balance latency and reliability, algorithms using heuristic search, or linear programming are proposed. However, given the fact that the problem is set in a system with optimal conditions (such as no interference, mobility and therefore handover, sharing of resources with other UEs, etc.), the resulting optimal number of computing eNBs goes to infinity and therefore is conditioned by choice of such number.

In the previouly mentioned papers, the optimal number of computing eNBs is only considered for a single user. Multiple-user system introduces a more difficult optimization problem, as the decision on partitioning of one user affects performance of others. In such a system, UEs compete for either for the network bandwidth to the MEC, computation resources, or the physical layer communication resources.

In [22], authors assume a multi-user system, where the users offload their computation tasks to corresponding serving eNBs. The serving eNBs then form computing clusters of nodes simultaneously for each offloading request.

The clusters are characterized by resource management and adaptive size based on the computation requests' requirements. The optimization method is proposed, with the objective to distribute the computation load between all eNBs for all the requests in such a way that would lead to the minimal power consumption of the clusters while satisfying latency contraints. The proposed solution effectively makes use of computation power of less busy, or idle nodes, however, it puts additional load on the backhaul, which in case of SCeNBs, depending on their connection, could lead to increased latencies.

## 2.3 Users' mobility

None of the previously mentioned articles takes into consideration the mobility of the users. The issue of mobility in MEC system is vital, but it makes the overall solution much more complex. As the UE changes its location, measures need to be taken in order to ensure the service continuity and Quality of Service (QoS). Generally, there exist two approaches to the movement of UEs.

Firstly, the possibility of VM migration in case the gain (lower latencies and less allocated resources on transmitting the results back to the UE) exceeds the cost (time and resources spent on VM migration). Authors in [2] formulate a sequential decision making problem for dynamic service migration. Cost models for transmission and migration are presented, with objective to design a policy that would minimize the expected overall costs long-term. As the cost functions in the defined model depend on distance only, the state space of the Markov Decision Process (MDP) is approximated by the distance between the user and the VM (distance-based MDP) to simplify the search for optimal policy. The solution is obtained by mapping the optimal policy for the approximated distance-based MDP to a policy for two-dimensional MDP.

Secondly, once the offloaded task secures allocated resources in the form of VM at its serving eNB, the VM stays at this particular eNB, even when the UE has already changed location and performed handover to another base station. The computed task then needs to be transferred either via radio or backhaul using alternative communication path to evade degraded performance. This is a better option in case the VM migration would require transmission of a large amount of data that would not only put a load on the backhaul, but would also result in large delay.

For this reason, authors in [23] propose an algorithm that exploits reward function from MDP with the objective of minimizing transmission delay. Proposed algorithm searches for possible paths and forces the UE to perform handover to a new eNB in case it is profitable by means of overall transmission delay. The path-selection algorithm is enhanced in [24] by considering the reception of computation results in downlink, evaluation of its impact on the load of backhaul network, and further decrease in the algorithm's complexity. The authors improve the algorithm by considering mobility prediction and proposing dynamic VM placement in [25]. The partitioning of a task into subtasks and parallel processing is not considered.

# Chapter 3

# System Model

The system model comprises the set $\mathcal{U} = \{u_1, u_2, \ldots, u_U\}$ of $U$ moving UEs and the set $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ of $N$ eNBs, or SCeNBs. Each device is connected to the mobile network by the means of a serving eNB, which is selected based on the highest received signal strength ($RSS$). $RSS$ can be obtained as the difference between an eNB's transmission power $P_t$ and path loss between $n$-th eNB and $u$-th UE, as follows:

$$RSS_{nu} = P_t - PL_{nu}. \tag{3.1}$$

The $RSS$ value changes with the mobility of the user and therefore needs to be carefully monitored in order to ensure quality of communication and meet the QoS requirements. As the $RSS$ degrades, the UE inquires after possible serving eNB candidates (target eNB) in its surroundings, resulting in handover procedure when the $RSS$ from target eNB continues to exceed $RSS$ from the serving eNB raised by a handover hysteresis $\Delta_{HM}$ for a period of $TTT$ (Time-To-Trigger),

$$RSS_{target} > RSS_{serving} + \Delta_{HM}. \tag{3.2}$$

Given the requirement that 5G system should support seamless handover, handover delay throughout this paper is considered equal to zero. The handover procedure is addressed in detail in [26].

## 3.1 Task model

The mobile devices $u \in \mathcal{U}$ randomly generate computationally intensive tasks $J_u$, which are offloaded to and processed in the MEC (further on denoted as offloaded task). Let $\pi_u$ be a binary indicator, which equals 1 if a device has generated such a task, or 0 otherwise.

$$\pi_u \in \{0, 1\}. \tag{3.3}$$

These indicators are grouped into a tuple:

$$\pi = (\pi_1, \pi_2, ..., \pi_U). \tag{3.4}$$

In order to simplify the problem, the UE cannot generate a task while waiting for an already generated task to be allocated, or for an allocated task to finish computing. This condition can be achieved by defining a binary parameter $\phi_u$, $1 \leq u \leq U$, for determining the state of the UE:

$$\phi_u \in \{0, 1\}. \tag{3.5}$$

In case the UE is waiting for a task to be allocated or computed, and therefore cannot generate a new task, $\phi_u$ equals 0. The state of all the UEs is written into a tuple $\phi$:

$$\phi = (\phi_1, \phi_2, ..., \phi_U). \tag{3.6}$$

The time when the UE waits for the task to be computed is described by a tuple $t^{UE}$, which works as a timer, counting down the remaining time $t_u$ of the task completion:

$$t^{UE} = (t_1^{UE}, t_2^{UE}, ..., t_U^{UE}). \tag{3.7}$$

The offloaded task is characterized by a size of input (offloaded) data $L^{input}$, output (collected) data $L^{output}$ and the number of instructions necessary to execute the task, denoted as $L^{comp}$.

As mentioned previously, the tasks consist of both offloadable and unoffloadable parts that need to be identified prior to the offloading itself. The partitioning of tasks depends on several factors, primarily the type of application, but also delay and reliability demands, etc. [7]. For the purpose of reducing the complexity, the task is considered to be offloadable as a whole and individual subtasks are independent after the partitioning.

## 3.2 Parallelization

In order to achieve the minimum delay, partitioning of tasks and parallel processing at the eNBs is employed, allowing the UE to communicate with multiple eNBs. For simplicity, a server with uniform processing capabilities is employed at each eNB in the set $\mathcal{N}$. The quantity of available computing resources of the $n$-th eNB is denoted as $C_n^{VM}$ in Millions of Instructions Per Second (MIPS). Additionally, each UE possesses its own computing resources $C_u^{VM}$, however, for the purpose of this work, it is considered that all generated tasks are offloaded to the MEC.

The tasks are partitioned into several subtasks in correspondence with the number of eNB candidates, i.e. the set $\mathcal{M}_u \subseteq \mathcal{N}$ of $M$ eNBs that are chosen by the $u$-th UE for communication and offloading ($1 \leq M_u \leq N$). The individual subtasks $s_m$ represent ratios of the task itself. This can be summarized as:

$$s_u = \sum_{m=1}^{M} s_m = 1, \tag{3.8}$$

where $1 \leq m \leq M$ and $1 \leq u \leq U$.

The UE then transmits the individual subtasks in sequence to these chosen eNB candidates for computation, as illustrated in Figure 3.1.

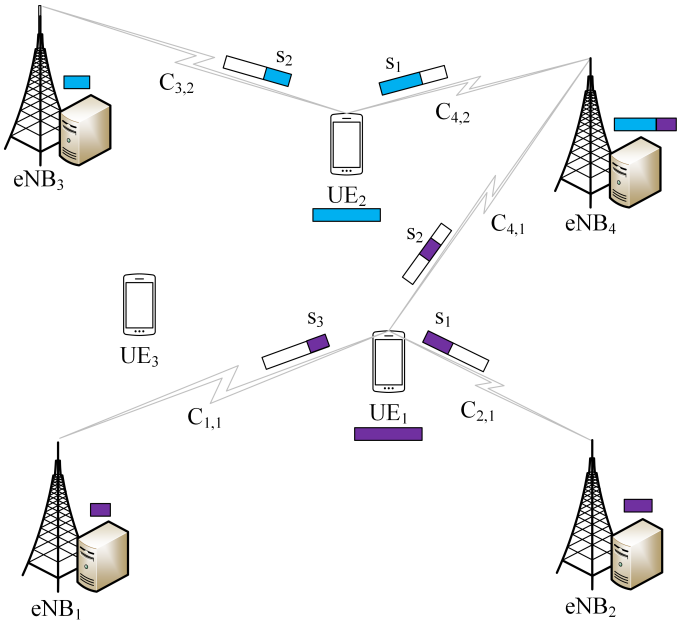



**Figure 3.1:** The system model

## 3.3 Delay

The overall delay of a subtask is defined as:

$$D_m = D_m^{wait} + D_m^{UL} + D_m^{comp} + D_m^{DL}, \tag{3.9}$$

where $D_m^{wait}$ is the waiting period of the subtask caused by sequential offloading, $D_m^{comp}$ is the computing delay, $D_m^{UL}$ and $D_m^{DL}$ are the transmission delays. The individual delays are explained in the following sections.

### 3.3.1 Transmission delays

The $u$-th UE communicates with $n$-th eNB via radio channel with capacity defined by Shannon-Hartley theorem [27]:

$$C_{nu} = B \cdot \log_2(1 + \gamma_{nu}), \tag{3.10}$$

where $B$ is the channel bandwidth and $\gamma_{nu}$ is the Signal to Interference plus Noise Ratio (SINR) calculated from:

$$\gamma_{nu} = \frac{P_t}{P_{noise} + I}, \tag{3.11}$$

where $I$ is the sum of interference from all but the serving eNBs. $P_{noise}$ is the power of thermal noise that is obtained from:

$$P_{noise} = 10 \cdot \log(kT_sB), \tag{3.12}$$

$k$ being the Boltzmann's constant and $T_s$ the effective system noise temperature.

Transmission delays of the $m$-th subtask are computed as the ratio of the transferred data, i.e. the portion $s_m$ from the total size of the task in uplink $L^{input}$, or downlink $L^{output}$, and capacity of the channel. Based on the previous statement, it can be assumed that minimum transmission delay can be achieved with communication over channel with maximum data rate (capacity):

$$D_m^{UL} = \frac{s_m \cdot L^{input}}{C_{mu}^{UL}}, \tag{3.13}$$

$$D_m^{DL} = \frac{s_m \cdot L^{output}}{C_{mu}^{DL}}. \tag{3.14}$$

### 3.3.2 Waiting delay

Because parallel offloading would require e.g. a multiple-input and multiple-output (MIMO) system integrated in each UE, it is assumed that UEs offload the individual subtasks in sequence. This leads to a waiting period defined as a sum of offloading delays of all the previous subtasks:

$$D_m^{wait} = \sum_{i=1}^{m-1} D_i^{UL}. \tag{3.15}$$

### 3.3.3 Computing delay

Computing delay is dependent on available computing capacity ($C^{VM}$) of MEC servers situated at computing eNBs:

$$D_m^{comp} = \frac{s_m \cdot L^{comp}}{C_m^{VM}}. \tag{3.16}$$

# Chapter 4

# Problem Formulation

The objective of this work is to find a strategy for allocation of computing and communication resources that minimizes the overall delay of a task for the $u$-th UE, denoted as $D_u$. The total offloading delay needs to take into account the delay of individual subtasks, as well as possible waiting of the task for the allocation of resources.

The optimal solution to the partitioning of a task is one, where all the subtasks are sequentially offloaded to the edge servers, computed and sent back to the UE, all reaching it at the same time, i.e. when the delays of all the subtasks $D_m$ are equal. Then the delay of the first subtask is also the delay of the entire task's completion and can be calculated using the equation 3.9. For the first subtask to be uploaded to the MEC, there is no waiting for the offload of previous subtasks, therefore $D_1^{wait} = 0$.

As each UE with a task to offload competes for the resource allocation with other UEs, it is possible that the UE's request is not satisfied immediately, as shows the figure 4.1.

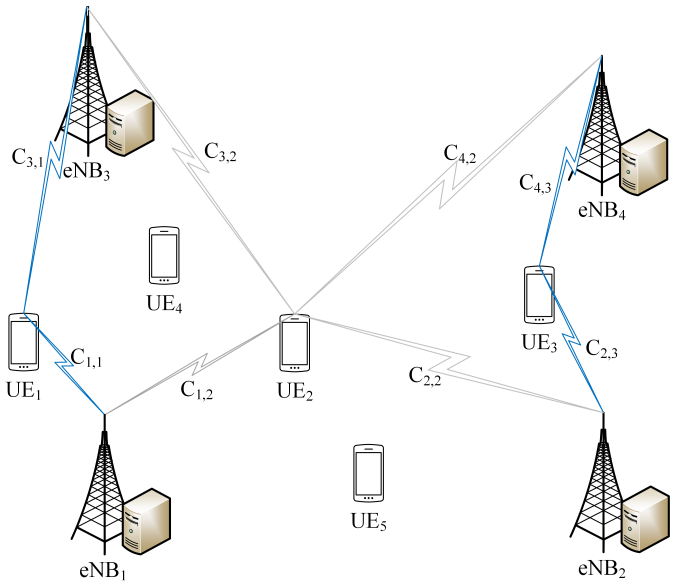



**Figure 4.1:** Exemplary situation with multiple UEs that generated tasks - tasks by $UE_1$ and $UE_3$ are allocated, $UE_2$' task waits for allocation.

In this case, the overall completion time of the task grows by the time it spends waiting. Once the task gets chosen for offloading, the final accumulated value of $D_u^{acc-wait}$ is added to the delay of the first subtask, i.e. transmission and computing delays of the whole task, as can be seen in figure 4.2.

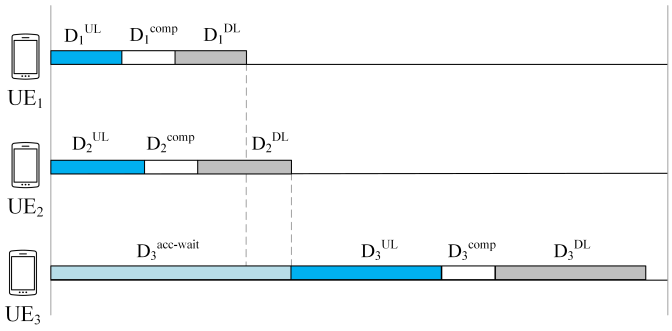



**Figure 4.2:** Timeline for offloaded tasks from situation in Figure 4.1

The resulting total completion delay of the task is:

$$D_u = D_u^{acc-wait} + D_1^{UL} + D_1^{comp} + D_1^{DL}. \tag{4.1}$$

Another problem poses the determination of the VMs placement for the $u$-th UE, denoted as $n_u^*$. Joining both together, the optimization problem can be formulated as:

$$n_u^* = \text{argmin}\{D_u\}. \tag{4.2}$$

As the partitioning of the task assumes transmission at the maximum data rate and an immediate start of the task computation at the selected eNBs, the optimization problem is subject to the availability of both the communication and computation resources at the eNBs. The compliance with the constraint 3.8 is necessary for optimal task partitioning.

# Chapter 5

# Proposed Algorithm

This chapter is focused on the explanation of the proposed algorithm. Its intent is to find an allocation strategy minimizing the average completion of the offloaded task, i.e. the offloading delay, and the optimal way to partition an allocated task.

## 5.1 Task partitioning algorithm

In this section, the algorithm for optimal task partitioning is proposed.

### 5.1.1 Selection of eNB candidates

Each UE randomly generates computationally intensive task $J_u$ that is partitioned into subtasks and offloaded to appropriate eNBs. Given the waiting time defined in 3.15, the order in which the subtasks are offloaded, plays an important role. Each subtask but the first is delayed by the sum of $D_m^{UL}$ of the previous subtasks. In order to minimize the accumulation of waiting times and achieve the optimal solution, subtasks are primarily transmitted to the eNBs with better performance (channel quality). In optimal conditions, the more eNBs the UE would offload to, the lower the overall latency of the task would get. However, as the large number of computing eNBs would result either in collisions or queuing, and, additionally, worse reliability, it is necessary to select appropriate eNB candidates and eliminate those with worse channel quality.

Choice of appropriate eNB candidates for offloading can be conditioned using values of SINR. In order for the $n$-th eNB to participate in the $u$-th UE's task offloading decision, $\gamma_{nu}$ must be higher than the minimum level $\gamma_{min}$ required for communication:

$$\gamma_{nu} \geq \gamma_{min}. \tag{5.1}$$

The eNBs in compliance with this condition are selected for offloading and form subset $\mathcal{M}_u = \{m_1, m_2, ..., m_M\}$, $\mathcal{M}_u \subseteq \mathcal{N}$ of $M$ eNB candidates ($1 \leq M \leq N$) for each $u$-th UE that has generated a task (lines 1 to 3).

Once the initial condition for eNB candidates is met, the quality of both data rate and computing capabilities of each eNB from set $\mathcal{M}_u$ are combined

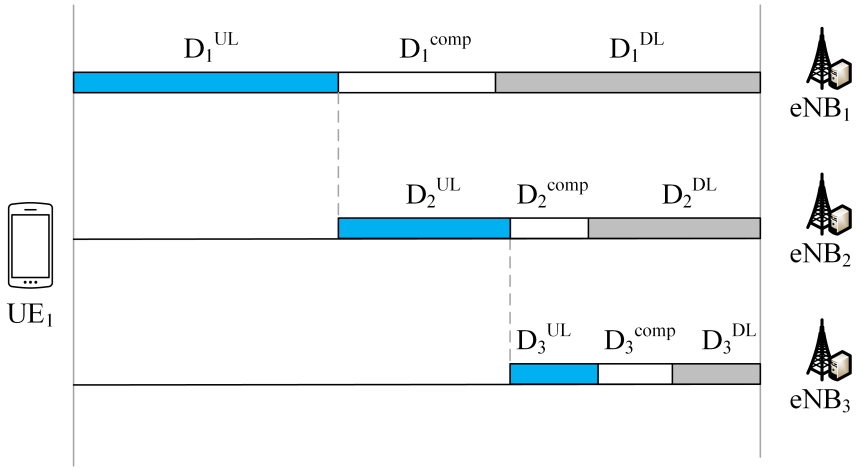



**Figure 5.1:** Optimal task partitioning

into the weighing parameter (line 4):

$$w_{mu} = \frac{L^{input}}{C^{UL}_{mu}} + \frac{L^{comp}}{C^{VM}_{m}} + \frac{L^{output}}{C^{DL}_{mu}}, \tag{5.2}$$

which represents the overall delay in case the entire task of $u$-th UE is offloaded to the $m$-th eNB, and it is composed of delay of UL transmission $D^{UL}_{mu}$, computation delay $D^{comp}_{mu}$, and delay of DL transmission $D^{DL}_{mu}$.

Given the waiting period caused by the sequential offloading, the transmission order of the individual subtasks has a large impact on the task's total completion delay. The waiting time, as a sum of UL delays of all the previous subtasks, is dependent on the data rate to the eNBs and therefore the parameter $w_{mu}$. In order to prevent the accumulation of waiting delay, the eNB with the highest data rate to the given UE, or the least value of $w_{mu}$, should be the first the UE offloads its subtask to. All the eNBs in set $\mathcal{M}_u$ are then sorted by the weighing parameter $w_{mu}$ in ascending order, which also sets the order for the offload of individual subtasks (line 5).

### 5.1.2 Partitioning of tasks

The solution for task partitioning is optimal when the completion times of all subtasks are equal, as pictured in Figure 5.1. Unlike in [7], where the authors assume the same condition, the delay of DL is taken into consideration and the problem of resource allocation is further solved for multiple users. The simultaneous completion of all subtasks can be expressed using the following equation:

$$\begin{aligned} s_1(D^{UL}_1 + D^{comp}_1 + D^{DL}_1) = s_1 D^{DL}_1 + s_2(D^{UL}_2 + D^{comp}_2 + D^{DL}_2) = \\ = \ldots = \sum_{i=1}^{M-1} s_i D^{UL}_i + s_M(D^{UL}_M + D^{comp}_M + D^{DL}_M), \end{aligned} \tag{5.3}$$

When elaborated (see *Appendix*), the following equation is acquired:

$$s_m = s_1 \frac{\prod_{i=1}^{m-1} (D_i^{comp} + D_i^{DL})}{\prod_{i=2}^{m} (D_i^{UL} + D_i^{comp} + D_i^{DL})}, \tag{5.4}$$

where $s_1$ is the portion of the task that is offloaded first and can be expressed from 3.8:

$$s_1 = \left(1 + \sum_{i=2}^{M} \frac{\prod_{j=1}^{i-1} (D_j^{comp} + D_j^{DL})}{\prod_{j=2}^{i} (D_j^{UL} + D_j^{comp} + D_j^{DL})}\right)^{-1}. \tag{5.5}$$

The values $s_m$ are calculated using 5.4 and 5.5 for each of the generated tasks to determine the tasks' initial partitioning (line 6). As $s_1$ represents the first offloaded subtask, its corresponding delay is also the delay of processing the whole task. Using the formula 3.9, the overall latency of each of the generated tasks from $\pi$ is calculated by the corresponding UE and then send to the MEC for comparison, causing only a very small additional overhead (line 7).

The algorithm for task partitioning is summarized below:

**Algorithm 1: Task partitioning**

**Input**: $\forall u \in \mathcal{U}, U, n \in \mathcal{N}, \gamma_{min}, \gamma_{nu}, L^{input}, L^{comp}, L^{output}, C_{mu}^{DL}, C_m^{comp}, C_{mu}^{UL}, \pi$

**Output**: Ratios of subtasks $s_m$, task overall delay $D_u$, set $\mathcal{M}_u$

1: **for** $u = 1$ to $U$ **do**
2: **if** $\pi_u = 1$
3: determine set $\mathcal{M}_u$ of $M$ eNBs based on condition 5.1
4: calculate parameter $w_{mu}$ for each eNB $m \in \mathcal{M}$ using 5.2
5: sort eNBs in $\mathcal{M}_u$ in ascending order based on $w_{mu}$
6: calculate $s_m$ for $\forall m \in \mathcal{M}$ from 5.5 and 5.4
7: calculate the overall delay $D_u$ using 3.9
8: **end**
9: **end**

## 5.2 Task allocation algorithm

In order to guarantee the optimal partitioning of the task and the values of total delay, the UE, whose task is chosen for offloading, gets all the resources it has used for the previous calculations, i.e. all the bandwidth in both

UL and DL, and the computing power of the allocated VM. Outside of the optimal conditions assumed in this work (in the real-world application), fast fading and interference may alter the results and should be taken into account during the task partitioning.

For the sake of simplicity, this reservation of the resources will last for the whole duration of the task completion - interlocking more generated and partitioned tasks that are offloaded to and computed at multiple eNBs at the same time would pose an optimization problem with very complex solution. In order to justify this decision, the following example is set.

Let there be a situation with several UEs in the vicinity of one particular eNB, which is also their serving eNB, as displayed in Figure 5.2.

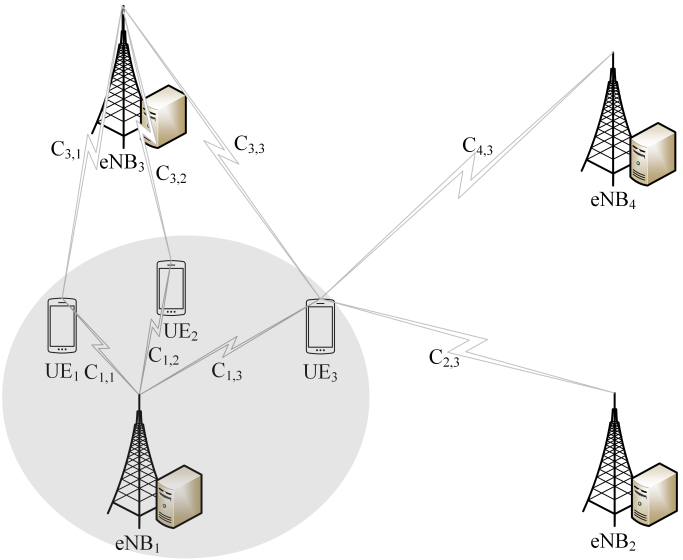



**Figure 5.2:** Exemplary situation with multiple UEs sharing the serving eNB

All of these UEs generate a task at the same moment and demand resources. All the UEs calculated the weighing parameter $w_{mu}$ to decide the order in which the individual subtasks should be offloaded to the selected eNB candidates. Depending on the order and the delays of UL, computing and DL to the individual eNB candidates, the tasks are partitioned into subtasks using 5.4 and 5.5. This partitioning guarantees that all the subtasks are completed simultaneously, despite the sequential offloading, supposing the variables in the equations remain unchanged.

Since the computing capabilities of all the eNBs are the same, the best SINR and therefore the best data rate result in the lowest weight $w_{mu}$ for the serving eNB. Because of that, the serving eNB is the first one all the UEs send their first and largest subtask to. In conventional mobile cellular networks, the UEs communicate with its serving eNB via different channels, sharing the bandwidth. However, sharing the bandwidth would result in lower data rates for the UEs, and in case of simulataneous offload to the same eNB queueing for available computation resources. For the partitioning of a task to be optimal, all the individual subtasks must be completed at the

same time. Fot this reason, sharing the communication and computation resources of the eNB is dismissed.

In order to prevent collisions, as shows the example in Figure 5.3, it is necessary to reserve all the communication and computation resources of the eNB for the duration of task completion. The following competition for resources is proposed.

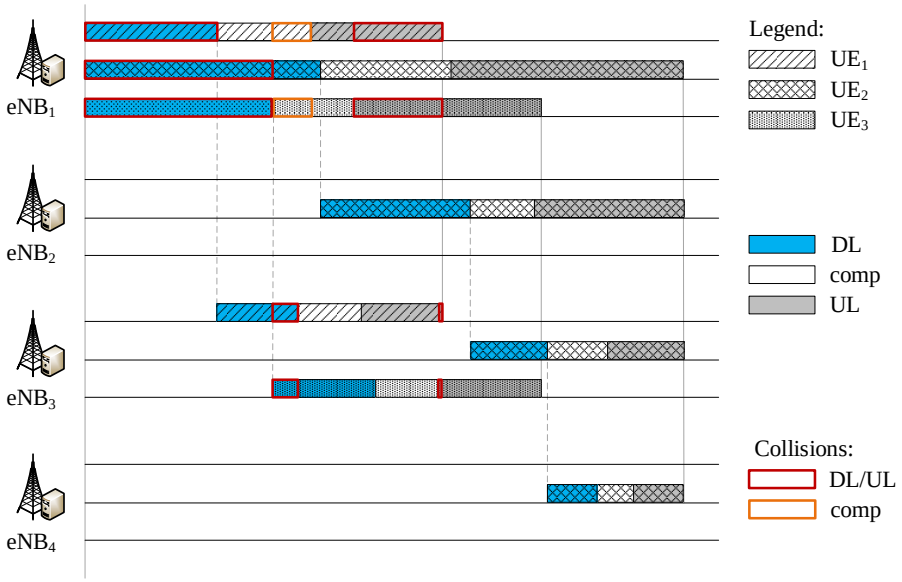



**Figure 5.3:** The collisions in the system from the situation in 5.2

After the task partitioning, the task with the minimum value $D_u$ is selected by the centralized system for the VM allocation (line 1), and denoted as $J_u^*$:

$$J_u^* = \min\{D_1, D_2, ..., D_U\}. \tag{5.6}$$

Once the task gets chosen for offloading, the binary coeffient $\pi_u$ of the corresponding $u$-th UE changes its value to 0 (line 2). As the UE cannot generate another task while waiting for the results (3.5), the value of binary parameter $\phi_u$ is set to 0 (line 3). The duration of this state is monitored by means of $t_u^{UE}$ (line 4).

The chosen UE offloads its task to the eNBs from its set $\mathcal{M}_u$, making them unavailable to other UEs for offloading. As each UE with a task to offload selects its own set $\mathcal{M}_u$ of appropriate eNB candidates, possible overlaps occur. The eNBs computing the task of the chosen UE are dropped from the set $\mathcal{M}_u$ of the rest of the UEs.

Since the reserved resources of the eNB candidates computing the chosen tasks render these eNBs unavailable for the whole duration of the task completion, it is necessary to maintain a timer for the eNBs' unavailability and keep track of its state. Therefore $\psi$ is defined as a tuple composed of

binary parameters $\psi_n$, $1 \leq n \leq N$, for whether the $n$-th eNB is available (equal to 1), or whether is has been assigned a task (equal to 0):

$$\psi_n \in \{0, 1\}, \tag{5.7}$$

$$\psi = (\psi_1, \psi_2, ..., \psi_N). \tag{5.8}$$

The parameter $\psi_n$ for eNBs' state is closely connected to the parameter for eNBs availability $t^{eNB}$ that counts down the time during which the eNB busy with the task processing remains unavailable:

$$t^{eNB} = (t_1^{eNB}, t_2^{eNB}, ..., t_N^{eNB}), \tag{5.9}$$

where $t_n^{eNB}$, $1 \leq n \leq N$ is the remaining time of the task completion. The eNBs that have not been allocated a task have this variable set to 0.

When the task with minimum $D_u$ is chosen for offloading, the values $\psi_n$ of the corresponding computing eNBs (set $\mathcal{M}_u$ of the chosen task) are set to 0 (line 5) and values of $t_n^{eNB}$ are set to $D_u$ (line 6) to keep track of the remaining time of the task completion.

The algorithm for task allocation is summarized below:

**Algorithm 2: Task allocation**

**Input**: $\mathcal{M}_u, D_u, \pi, \psi, \phi, t^{eNB}, t^{UE}$;

**Output**: Tasks allocation at eNBs, $\phi, \psi, t^{eNB}, t^{UE}$

1: select task $J_u^*$ with the minimum $D_u$ for the VM allocation
2: set value $\pi_u$ of selected UE's task to 0
3: set value $\phi_u$ of selected UE's task to 0
4: set value $t^{UE}$ of chosen UE to $D_u$
5: set values $\psi_n$ of corresponding computing eNBs to 0
6: set values $t^{eNB}$ of corresponding computing eNBs to $D_u$

## 5.3 Algorithm

This section summarizes the proposed algorithm.

The system comprises the set $\mathcal{U}$ of $U$ UEs, which randomly generate tasks in compliance with the assumption 3.6 (line 2). The binary indicator $\pi_u$ shows whether the UE has generated a task for offload ($\pi_u = 1$), or not ($\pi_u = 0$). The UEs are connected to the cellular network via their serving eNB, but they can also communicate wirelessly with any other eNB from the $N$ eNBs in the set $\mathcal{N}$, as long as the condition 5.1 for the SINR above the minimum is met. Each UE first obtains values of SINR for the eNBs in its vicinity through standard measurements [28] without introducing any additional overhead (line 3). The state of the eNBs is tracked by the binary indicator $\psi_n$, which either equals 1 if the $n$-th eNB is available, or 0 when it is has been assigned a task.

While there are UEs with generated tasks and appropriate eNB candidates available for computing, the following cycle starts (lines 4 to 6).

In algorithm 1 (line 5), each UE compares the measured values of $\gamma_{nu}$ with the minimum $\gamma_{min}$, based on which it selects its own subset $\mathcal{M}_u$. The weighing parameter $w_{mu}$ is calculated for each of the selected eNB candidates using 5.2. The eNB candidates are sorted in the ascending order based on the weighing parameter, i.e. the eNBs with the least values of $w_{mu}$ are the first the UE offloads to. The UE then partitions its task among all the $M$ eNBs from the set $\mathcal{M}_u$ using 5.4 and 5.5. The overall completion times of all the generated and partitioned tasks are calculated from 3.9.

In algorithm 2 (line 6), the task $J_u^*$ with the least delay is selected for the VM allocation and the corresponding value of $\pi_u$ is set to 0. The time it will take to complete the task is $D_u$. During this period, the eNB candidates from the chosen UE's set $\mathcal{M}_u$ are unavailable for task allocation and their corresponding value of $\psi_n = 0$. Given the assumption 3.6, the UE needs to wait for the results of the offloaded tasks before it can generate a new task, so its value of $\phi_u = 0$. The value of $D_u$ is tracked by the variables $t^{eNB}$ and $t^{UE}$, which work as timers for determining the remaining time of the task completion, and therefore the unavailability of eNBs and UEs respectively.

With passing time, the values of $t^{eNB}$ and $t^{UE}$ diminish. Once they reach zero, the task is completed and the values of $\phi_u = 1$ and $\psi_n = 1$, meaning the UE can generate tasks and the eNBs are available for offloading (lines 8 to 9).

The UEs, whose tasks have not been chosen for offloading, wait until the resources become available again. The time the UEs spend waiting for the VM allocation accumulates in the tuple (line 10):

$$D^{acc-wait} = (D_1^{acc-wait}, D_2^{acc-wait}, ..., D_U^{acc-wait}). \tag{5.10}$$

The remaining tasks still have their value $\pi_u = 1$ and participate in the competition for resources until they are selected for the VM allocation. The value of $D_u^{acc-wait}$ is added to the overall task completion delay once it is chosen for offload, but does not play part in the competition for resources. Considering this delay as a metric favouring the UEs that had spent a long time waiting for the VM allocation is a possible topic for future enhancements of this work.

The proposed algorithm is summarized below.

**Algorithm 3: Offloading to Multiple Nodes with Optimal task partitioning (MNO)**

**Input**: $\forall u \in \mathcal{U}, U, n \in \mathcal{N}, \gamma_{min}, L^{input}, L^{comp}, L^{output}, C_{mu}^{DL}, C_{m}^{comp}, C_{mu}^{UL}, \phi, \psi$

**Output**: Offloading tasks for execution

**loop** (continuously repeated in time)
  UEs generate tasks for offload $\pi_u$ based on $\phi$ values
  calculate values of $\gamma_{nu}$ from 3.11
  **while** $\pi_u = 1$ and $\psi_n = 1$ **do**
    partition the tasks using **Algorithm 1**
    allocate task and computing eNBs using **Algorithm 2**
  **end**
  update $t^{eNB} = t^{eNB} - t^{loop}$ and $\psi$
  update $t^{UE} = t^{UE} - t^{loop}$ and $\phi$
  update $D_u^{acc-wait} = D_u^{acc-wait} + t^{loop}$
**end loop**

# Chapter 6

# Simulations

In this section, models for simulations carried out in MATLAB are presented for performance evaluation and compared with several other existing solutions.

## 6.1 Simulation scenarios and models

The main simulation parameters are presented in Tab 6.1.

| Parameter | Value |
|---|---|
| Simulation time | 60 s |
| Simulation step | 10 ms |
| Number of simulation drops | 5 |
| Simulation area | 650 m x 370 m |
| Bandwidth of uplink/downlink | 10/10 MHz |
| Minimum SINR value $\gamma_{min}$ | -10 dB |
| Carrier Frequency | 2 GHz |
| Number of eNBs/SCeNBs | 4/30 |
| Tx power of eNB/SCeNB | 27/15 dBm |
| eNB/SCeNB computing power $C_n^{VM}$ | 30 or 300 or 1000 or 3000 MIPS |
| Number of UEs | 30 or 60 |
| UE computing power | 40 MIPS |
| Maximum speed of users ($v_{max}$) | 2 m/s |
| Values of Poisson coefficient $\lambda$ | 0.05 or 0.10 or 0.15 or 0.20 or 0.25 |
| Input/output data size | 200/200 kB |
| Offloaded task number of instructions | 10e6 instructions |

**Table 6.1:** Simulation parameters

The parameters of the network are in line with recommendations defined by 3GPP in [29]. The size of tasks, both offloaded and collected, is considered 200 kB, based on [30], where the authors assume similar task sizes for real-time applications.

The UEs generate tasks arriving according to the Poisson distribution, a common assumption on the arrival of tasks in MEC systems [31]. The Poisson

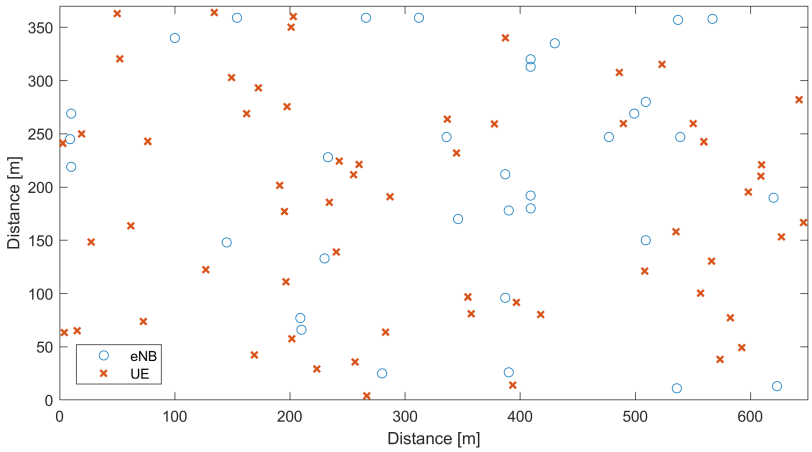



**Figure 6.1:** Initial positions of UEs within the simulation area for one simulation drop

distribution is defined as follows:

$$pr(k) = e^{-\lambda}\frac{\lambda^k}{k!}, \tag{6.1}$$

where $\lambda$ is the Poisson coefficient stating the average number of generated tasks in one step of the simulation, $k$ is the number of generated tasks in a simulation step. The simulations are run for different values of Poisson $\lambda$ coefficient, observing the impact it has on the performance.

The simulation area contains 4 eNBs, 30 SCeNBs and either 30 or 60 UEs. Initially, each UE and eNB are assigned a position, which for eNBs is fixed, whereas for UEs it changes as they move. The initial location of the UEs and the eNBs in simulation area for one drop can be seen in Figure 6.1

The UEs move at constant speeds that are chosen randomly from (0, $v_{max}$) following the two-dimensional (2D) probabilistic random walk [32] mobility model, where the movement is described by means of Markov chain. The chain has a finite number of states for both the x and y coordinates. For each pair of states $x_i$ and $x_j$ a transition probability $p_{ij}$ is defined for going from $x_i$ to $x_j$. In its initial position (state), each UE chooses a random direction of their movement and follows it. In a randomly chosen simulation step, if the UE is the state $x_i$, it selects the following state $x_j$ with probability $p_{ij}$. The y-coordinate follows the same principle. The Markov chain is defined by the transition probability matrix containing the values of $p_{ij}$.

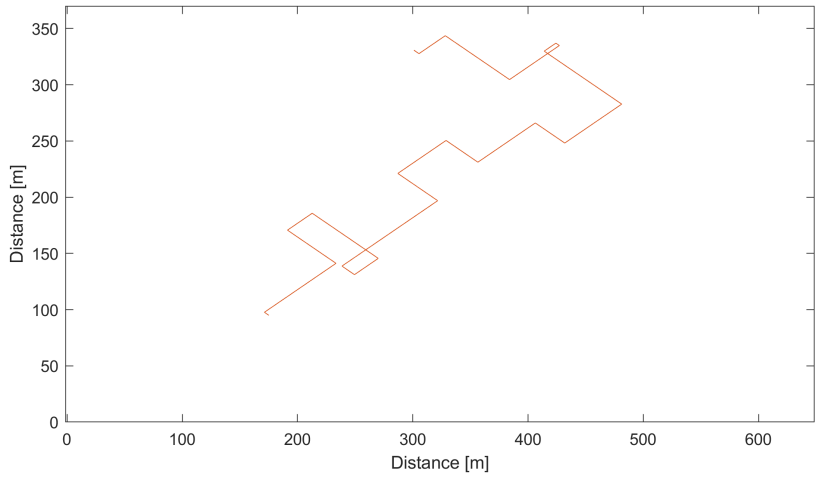



**Figure 6.2:** Example of UE's movement following the random walk mobility model

The signal propagation over the radio channel follows the free-space attenuation [33]. The path loss between the $u$-th UE and the $n$-th eNB can be calculated as follows:

$$PL_{nu} = 32.4 + 20\log_{10} f + 20log_{10} d_{nu}, \tag{6.2}$$

where $f$ is the carrier frequency and $d_{nu}$ is the distance in between them and can be obtained using difference between the coordinates of both:

$$d_{nu} = \sqrt{(x_u - x_n)^2 + (y_u - y_n)^2 + (z_u - z_n)^2}. \tag{6.3}$$

## 6.2 Performance analysis of the proposed algorithm

The performance of the proposed algorithm offloading to Multiple Nodes with Optimal task partitioning (further on denoted as MNO) is evaluated and analyzed in this subsection. The primary metric observed is the task completion latency. In order to observe the efficiency of the algorithm for different loads, the simulations are run for 30 and 60 UEs in the system, for multiple values of Poisson $\lambda$ coefficient, as well as several values of eNBs' computation power.

In order to confirm the efficiency of the task partitioning, dependency of the overall task completion on the number of computing eNBs is observed. The minimum value of delay helps the discovery of the optimal size of the set $\mathcal{M}_u$.

Figures 6.3 and 6.4 show the average delay based on the number of computing eNBs for different values of the eNBs' computation power. The value of Poisson coefficient is fixed at $\lambda = 0.1$. The average overall delay shows

steep decreasing tendency for a low computation power of 30 MIPS. For simulations with higher values of computing power, the overall delay shows minimum for certain numbers of computing eNBs. This is due to the fact that for low computation power, the computing delay presents the largest component in the overall task completion delay. Therefore, the more eNBs participate in the computation of a task, the lower the total task completion delay gets. On the contrary, with higher computation power, computing delays become insignificant compared to transmission delays and adding more eNBs eventually leads to higher overall delays. The simulation for 30 UEs in Figure 6.3 shows lower average delay compared to the one with 60 UEs in Figure 6.4, however, the overall characteristics of both are similar.

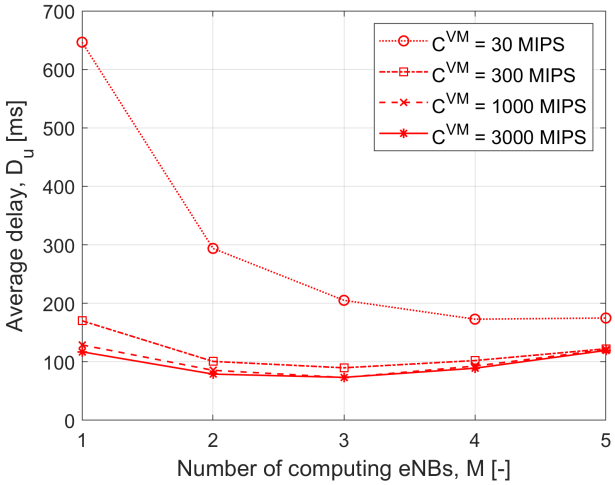



**Figure 6.3:** Average delay based on the number of computing eNBs for different values of $C^{VM}$ for in a system with UEs, $\lambda = 0.1$

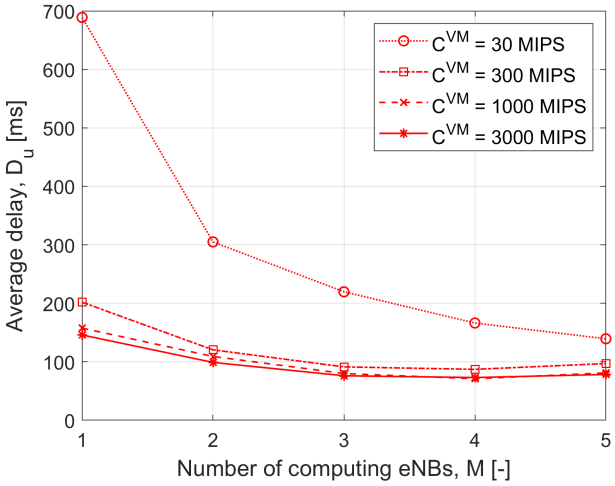
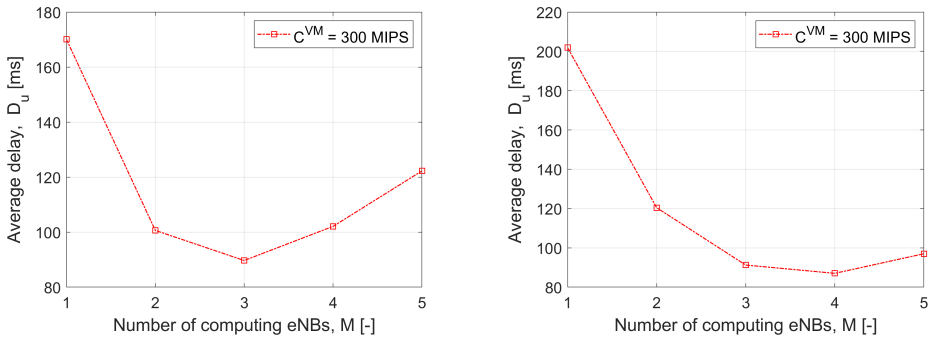



**Figure 6.4:** Average delay based on the number of computing eNBs for different values of $C^{VM}$ in a system with 60 UEs, $\lambda = 0.1$

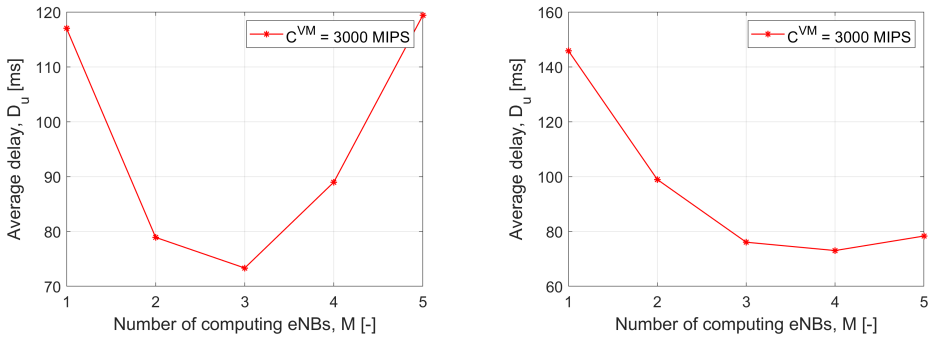
The minimum delay is more obvious when shown individually. Figure 6.5 shows the average delay based on the number of computing eNBs with computing power $C^{VM} = 300$ MIPS for 30 UEs (a) and for 60 UEs (b). In order to better show the behavior of the algorithm for higher values of eNBs' computing power, 6.6 follows with caption of the same dependency for $C^{VM} = 3000$ MIPS, again for 30 UEs (a) and for 60 UEs (b). From both Figures 6.5 and 6.6, it is obvious that for 30 UEs in the system the average latency is lower than for 60. The optimal sizes of computing cluster differ, for system with 30 UEs and 60 UEs reaching 3 eNBs and 4 eNBs, respectively. Adding more than the optimal number of computing eNBs results in increasing delay. In both figures, the system with less UEs shows steeper growth of average delay than the system with 60 UEs. This is caused by the parameter for minimum SINR $\gamma_{min}$, which was chosen with regard to the formation of larger computing clusters. With smaller competition for resources, these clusters are formed more often, resulting in more links with worse channel quality and therefore higher delays and different optimal numbers of computing eNBs.




**(a) :** System with 30 UEs

**(b) :** System with 60 UEs

**Figure 6.5:** Average delay based on the number of computing eNBs for $C^{VM} = 300$ MIPS, $\lambda = 0.1$, subplots for Figure 6.3 and Figure 6.4




**(a) :** System with 30 UEs **(b) :** System with 60 UEs

**Figure 6.6:** Average delay based on the number of computing eNBs for $C^{VM} = 3000$ MIPS, $\lambda = 0.1$, subplots for Figure 6.3 and Figure 6.4

The efficiency of the proposed algorithm also depends on the algorithm's capability to handle heavier load on the system. For this reason, the simulations were run for different sizes of Poisson $\lambda$ coefficient. Figures 6.7 and 6.8 show the dependency of average delay on the size of $\lambda$. The figures display clear increasing tendency of average delay with growing task arrival for all simulated cases. For computing power of eNBs $C^{VM} = 30$ MIPS, the increase in delay is more evident, rising by 22% and 31.5% for 30 UEs and 60 UEs, respectively. For computing powers of 300 MIPS and higher, the delays are significantly lower, the increase in delay when comparing $\lambda = 0.05$ and $\lambda = 0.25$ is from 15% to 19%. Though the delay is generally higher for more UEs in the system, it is not very different for smaller values of $\lambda$. This is due to the low amount of tasks generated by the UEs resulting in lesser competition for the resources.

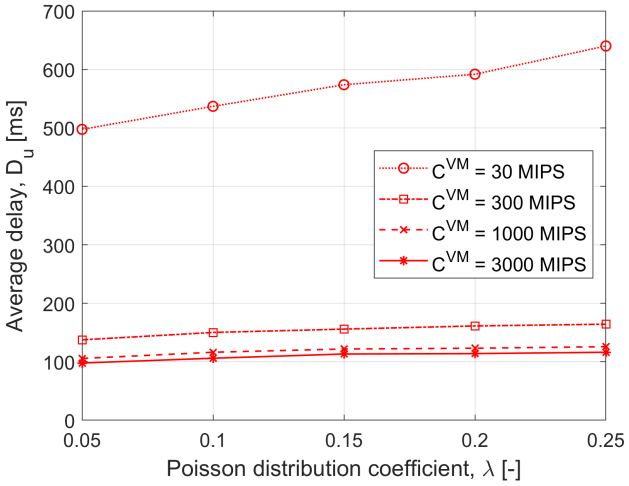



**Figure 6.7:** Average delay based on the Poisson $\lambda$ coefficient for different values of $C^{VM}$ in a system with 30 UEs

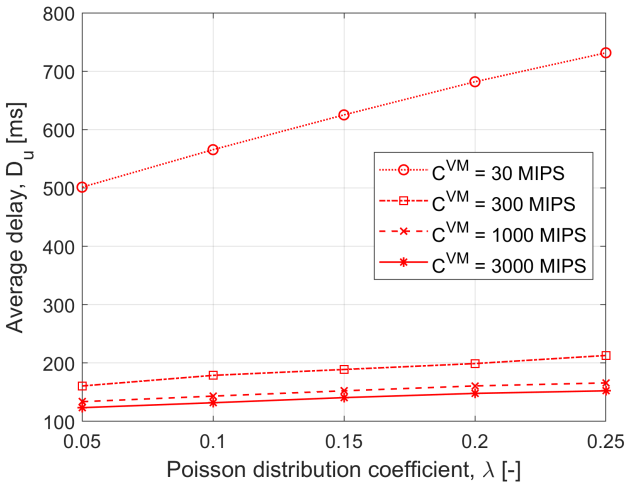



**Figure 6.8:** Average delay based on the Poisson $\lambda$ coefficient for different values of $C^{VM}$ in a system with 60 UEs

The overall delay comprises partial delays of the accumulated waiting period $D_u^{acc-wait}$, delays of transmission $D_u^{UL}$ and $D_u^{DL}$ and computation delay $D_u^{comp}$. The average values of these individual components are shown in Figures 6.9 and 6.10. For low computation power of eNBs, the computing delay is by far the most significant. Given the long period during which the eNBs compute their subtasks and therefore are unavailable to other UEs, the average waiting delay $D_u^{acc-wait}$ grows as well. The average overall delay is therefore composed mainly of waiting and computing, making the offload inefficient. On the contrary, the computing delay for eNBs with high computation power becomes almost insignificant, since the major part of the overall delay is the transmission. In this case, the overall latency is conditioned by the quality of transmission and the parallel processing, though it decreases the overall delay, is not very efficient. The proposed algorithm achieves the optimal performance for computationally demanding applications with a small amount of data to transfer, or when the transmission and computing delays of the tasks are comparable.

With more UEs in the system and more generated tasks, the system gets flooded with more requests for offloading. While some of there requests are satisfied and tasks offloaded, others need to wait until the resources (network bandwidth to the MEC, computing resources) become available. Thus, their overall task completion delay and correspondingly, the average delay in the whole system, is increased. The increase in $D_u^{acc-wait}$ is the most apparent in the figures. When comparing the values of accumulated waiting period of 30 and 60 UEs in the system, the latter shows up to 35% higher $D_u^{acc-wait}$ for different values of $C^{VM}$.

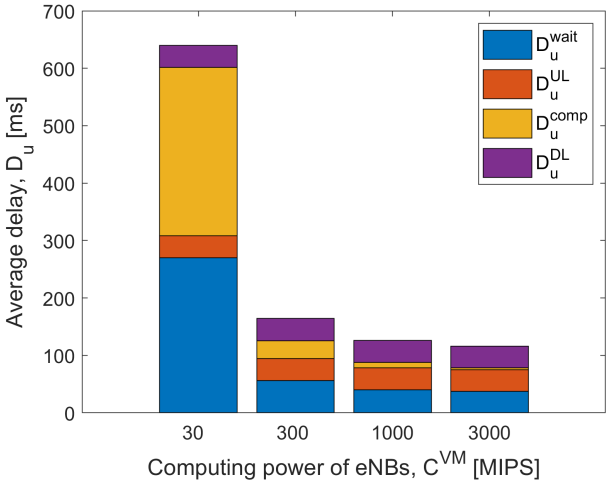



**Figure 6.9:** Average delay based on the computing power of eNBs for different values of $C^{VM}$ in a system with 30 UEs, $\lambda = 0.25$

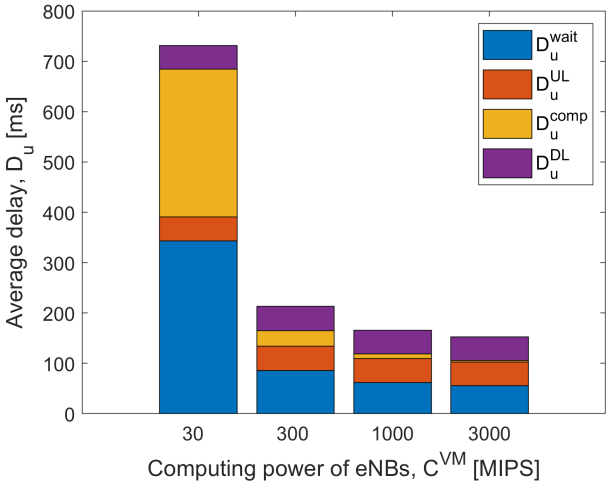



**Figure 6.10:** Average delay based on the computing power of eNBs for different values of $C^{VM}$ in a system with 60 UEs, $\lambda = 0.25$

In order to compare the behavior of the overall task completion delay for different values of Poisson $\lambda$ coefficient and eNBs' computing power $C^{VM}$, the cumulative distribution function (CDF) is used in Figures 6.11 and 6.11. Both the figures show the increasing tendencies of average delay with growing $\lambda$ and smaller computing power. The dash-dot lines represent $C^{VM} = 30$ MIPS. Given the low computing power and therefore long period of waiting for computation, the overall delays increase significantly, as display both figures. The full and dashed lines representing the $C^{VM} = 3000$ MIPS and $C^{VM} = 300$ MIPS, respectively, show that increasing computating power improves the overall delay (compared to $C^{VM} = 30$ MIPS). However, once the transmission delay outweighs the computing delay, increasing computing

power shows little impact on the average delay. In both figures, the blue lines representing the values of $\lambda = 0.1$ are always to the right of the red lines representing $\lambda = 0.25$, showing the average delay grows with Poisson coefficient. Comparing the results for the system with 30 and 60 UEs, the steepness and the ends of the lines indicate the higher values of the average delay for the latter.

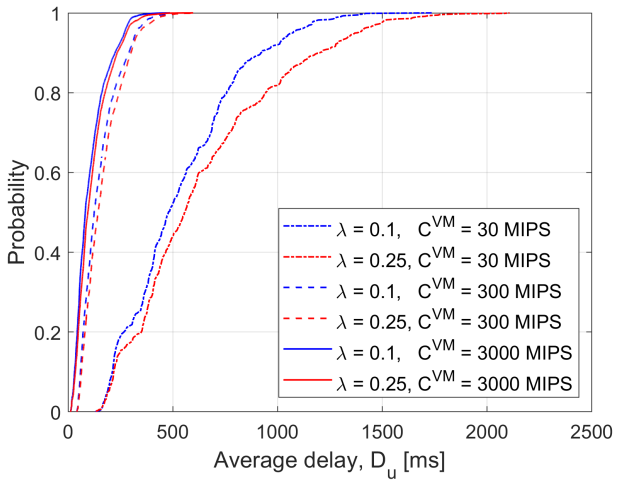



**Figure 6.11:** The CDF of the average delay based on different values of Poisson $\lambda$ coefficient and eNBs' computing power $C^{VM}$ in a system with 30 UEs

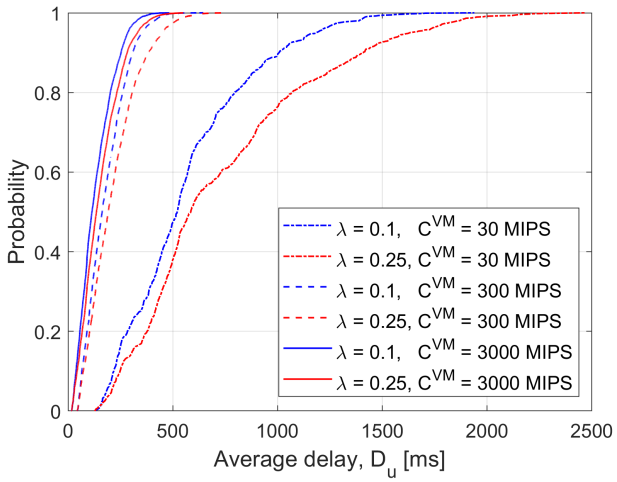



**Figure 6.12:** The CDF of the average delay based on different values of Poisson $\lambda$ coefficient and eNBs' computing power $C^{VM}$ in a system with 60 UEs

## 6.3 Comparison with other approaches

The performance of the proposed algorithm (MNO) is compared with several other approaches.

- Serving Only (SO) [34], where the UE only offloads to the eNB with the highest $RSS$ (its serving eNB).

- Heuristic Algorithm by Liu and Zhang (LZ) proposed in [7]. The task allocation for one UE is independent on the others. When the UEs task is allocated and eNBs within its computing cluster become unavailable to other UEs for offloading, collisions occur. The UEs do not adapt the size of their computing clusters, but instead wait for the eNBs to become available again.

- Local Computation (LC), where the UE computes the generated tasks locally. In order to properly compare both these algorithms, the exact same tasks offloaded by MNO algorithm are computed locally, leading to a waiting period when a new task is generated while another is still being computed at the UE.

Additionally, the MNO algorithm is compared to the proposed algorithm without the optimal subtask sequence, denoted as Multiple Nodes (MN). While the selection of computing clusters $\mathcal{M}_u$ and the task allocation stays the same as in MNO, the task partitioning algorithm is omitted. The sequential offloading is not optimized, the tasks are partitioned in equal parts between the computing eNBs.

Figures 6.13 and 6.14 show the average delay based on the computing power of eNBs, with fixed Poisson coefficient $\lambda = 0.25$. Both reflect the redundancy of increasing the computation power after a certain point. While the values of average delay show a steep decreasing tendency for higher values of computation power at first, there is not a significant difference after reaching 1000 MIPS for any of the compared algorithms. The proposed algorithm shows a decrease in average delay up to 68% and 48% compared to SO and LZ, respectively. The influence of the optimal partitioning and subtask sequencing on the proposed algorithm is indicated by its comparison with MN and reaches up to 22%. For the system with more UEs, and therefore more generated tasks to offload, the communication and computation resources become more scarce, resulting in higher average delays. The LC algorithm is not dependent on the eNBs' computing power, but it is shown for comparison of the offloading efficiency. In both figures, its value of average delay is lower for $C^{VM} = 30$ MIPS, indicating that for low computing power of eNBs, offloading to the MEC gets worse results in terms of delay than local computation.

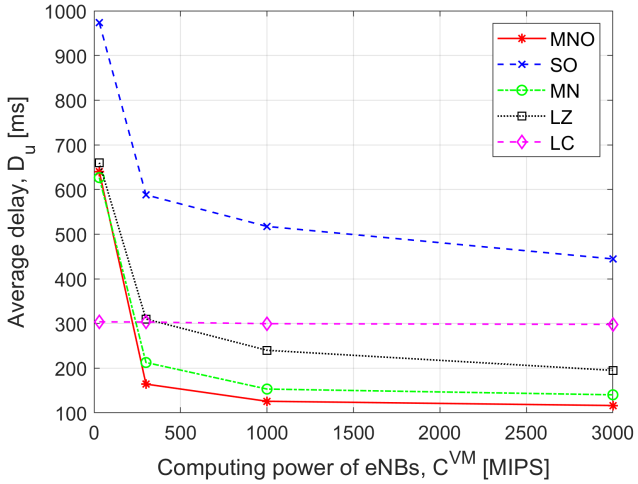



**Figure 6.13:** Average delay based on the computing power of eNBs for different algorithms in a system with 30 UEs, $\lambda = 0.25$

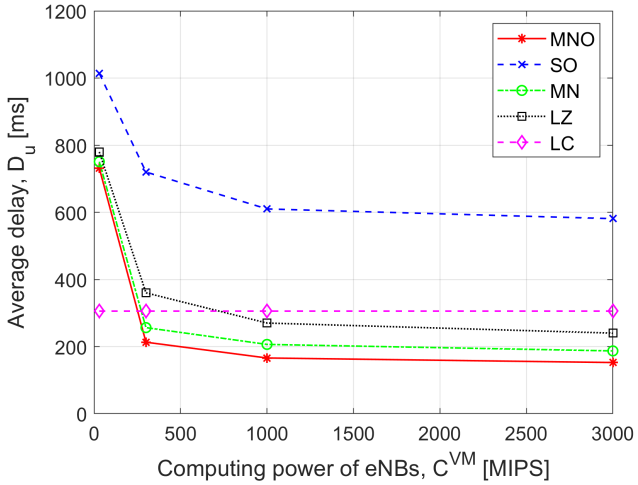



**Figure 6.14:** Average delay based on the computing power of eNBs for different algorithms in a system with 60 UEs, $\lambda = 0.25$

To compare the efficiency of the algorithms for different task generation values, figures 6.15 and 6.16 display the dependency of the average delay on the Poisson coefficient. Given the comparison of algorithms with fixed value of $C^{VM} = 3000$ MIPS, all algorithms based on partitioning (i.e. MNO, MN and LZ) show better performance than SO and LC algorithms. However, the MNO and MN demonstrate slower rise of average delay towards larger values of $\lambda$ than LZ algorithm. Comparing the results numerically, the proposed algorithm shows improved performance over SO, LZ and LC by up to 78%, 40% and 61%, respectively. The task partitioning and sequencing improves the efficiency by 19%, as shows the algorithm's comparison with MN. The system with more UEs once again shows larger average delay.

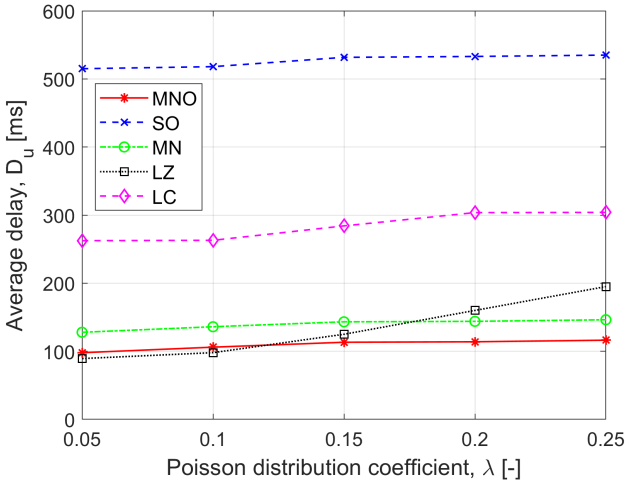



**Figure 6.15:** Average delay based on the Poisson $\lambda$ coefficient for different algorithms in a system with 30 UEs, $C^{VM} = 3000$ MIPS

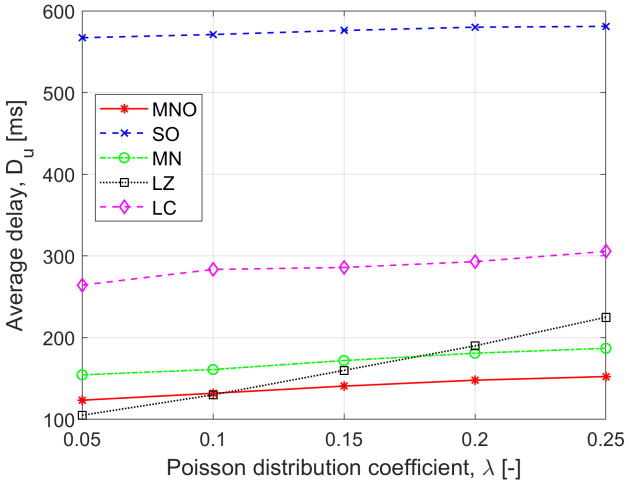
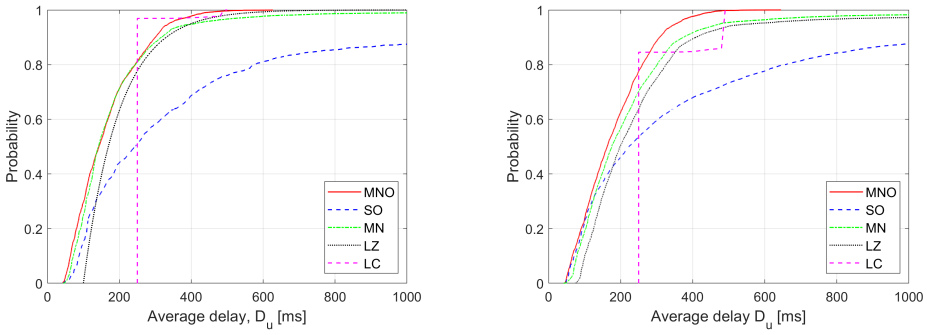



**Figure 6.16:** Average delay based on the Poisson $\lambda$ coefficient for different algorithms in a system with 60 UEs, $C^{VM} = 3000$ MIPS

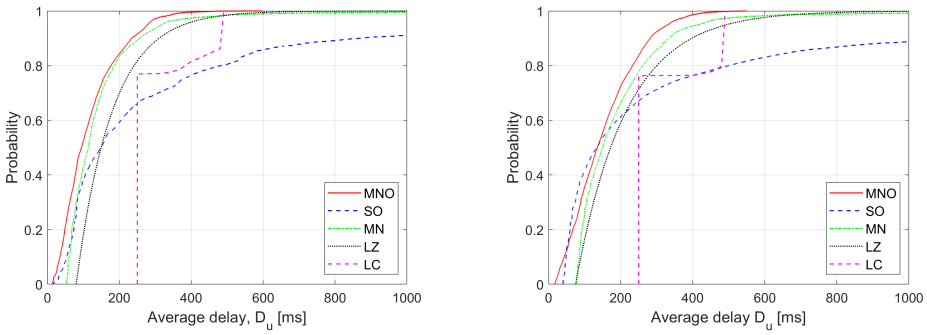
Finally, the CDF of the average delay is used to compare the algorithms. Figure 6.18 shows the CDF for $\lambda = 0.1$ and $C^{VM} = 300$ MIPS. The proposed algorithm shows the best performance for both the simulations with 30 and 60 UEs, closely followed by MN and LZ algorithms. The LC algorithm shows constant values of delay for most part, signifying that in the system with 30 and 60 UEs, for around 96% and 84% of tasks, respectively, computation starts without further delay, while the rest of generated tasks need to wait for them to finish.




**(a) :** System with 30 UEs **(b) :** System with 60 UEs

**Figure 6.17:** The CDF of the average delay for different algorithms, $\lambda = 0.1$, $C^{VM} = 300$ MIPS

Figure 6.18 shows CDF with similar shapes, this time for the values of $\lambda = 0.25$ and $C^{VM} = 3000$ MIPS. The algorithms show slightly better performance than in 6.17 because of larger $\lambda$ and the higher computing power of eNBs. Since the higher computing power has no effect on LC algorithm, the delay of locally computed tasks increases as the task arrival grows. With more UEs in the system, more users need to wait for the resources. For both the simulations with 30 and 60 UEs in the system, the ratio of tasks that can start computing right away drops down to around 76%. This similarity is caused by more tasks being held while the one is being computed.




**(a) :** System with 30 UEs **(b) :** System with 60 UEs

**Figure 6.18:** The CDF of the average delay for different algorithms, $\lambda = 0.25$, $C^{VM} = 3000$ MIPS

# Chapter 7

# Conclusion and future work

In this thesis, the concept of computation offloading, together with allocation of computing and communication resources in the MEC has been introduced. The scenario with multiple users sequentially offloading their partitioned tasks to multiple eNBs. In order to achieve minimum latency of the offloaded tasks, the algorithm offloading to Multiple Nodes with Optimal task partitioning (MNO) is proposed. The algorithm determines the possible candidates for computing each of the generated tasks and performs the initial partitioning correspodingly. The algorithm then iteratively allocates available resources while continuously adjusting the partitioning.

Comparing to state of the art approaches exploiting parallel processing, the proposed algorithm reduces the average task completion delay by up to 48%. When comparing to non-partitioning method, the proposed algorithm reduces the delay up to 78%. The algorithm achieves the best performance for computationally demanding applications with a small amount of data to transfer, or generally, when the computing delay of the task is comparable to the task's transmission delay. The algorithm performs well when increasing the computing load, with the value of average delay increasing by up to 19% for simulated scenarios.

Future work should consider further optimization of the task allocation process by introducing an element of fairness to the competition for resources. Other possible enhancement is integrating device-to-device (D2D) communication, allowing the users to offload their task via nearby devices.

# Bibliography


[1] Shahryar Shafique Qureshi, Toufeeq Ahmad, Khalid Rafique, and Shuja ul islam. "Mobile cloud computing as future for mobile applications - Implementation methods and challenging issues". *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pp. 467-471, 2011.

[2] Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin S. Chan, and Kin K. Leung. "Dynamic service migration in mobile edge-clouds". *2015 IFIP Networking Conference (IFIP Networking)*, pp. 1-9, 2015.

[3] ETSI. "5G". https://www.etsi.org/technologies/5g, Accessed: 2019-02-12.

[4] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration". *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657-1681, 2017.

[5] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. "Mobile Edge Computing - A key technology towards 5G". *ETSI White Paper No. 11*, pp. 1-16, 2015.

[6] Jan Plachy, Zdenek Becvar, Emilio Calvanese Strinati, and Nicola di Pietro. "Dynamic Allocation of Computing and Communication Resources in Multi-Access Edge Computing for Mobile Users". *Submitted to IEEE Transactions on Mobile Computing*, 2019.

[7] Jianhui Liu and Qi Zhang. "Offloading Schemes in Mobile Edge Computing for Ultra-Reliable Low Latency Communications". *IEEE Access*, vol. 6, pp. 12825-12837, 2018.

[8] Jessica Oueis, Emilio Calvanese-Strinati, Antonio De Domenico, and Sergio Barbarossa. "On the Impact of Backhaul Network on Distributed Cloud Computing". *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 12-17, 2014.

[9] Jin Cao, Lei Yang, and Jiannong Cao. "Revisiting Computation Partitioning in Future 5G based Edge Computing Environments". *IEEE Internet of Things Journal*, 2018.

[10] International Telecommunication Union (ITU). "Report ITU-R M.2410-0 (11/2017); Minimum requirements related to technical performance for IMT-2020 radio interface(s)". *Technical report*, 2017.

[11] Pavel Mach and Zdenek Becvar. "Mobile Edge Computing: A Survey on Architecture and Computation Offloading". *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 1628-1656, 2017.

[12] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. "Mobile Edge Computing: A Survey". *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, 2018.

[13] Stefania Sardellitti, Gesualdo Scutari, and Sergio Barbarossa. "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing". *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89-103, 2015.

[14] Chenmeng Wang, Chengchao Liang, F. Richard Yu, Qianbin Chen, and Lun Tang. "Computation Offloading and Resource Allocation in Wireless Cellular Networks With Mobile Edge Computing". *IEEE Transactions on Wireless Communications*, vol. 66, no. 8, pp. 4924-4938, 2017.

[15] Xu Chen, Lei Jiao, Wenzong Li, and Xiaoming Fu. "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing". *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795-2808, 2016.

[16] Maofei Deng, Hui Tian, and Bo Fan. "Fine-granularity Based Application Offloading Policy in Cloud-enhanced Small Cell Networks". *2016 IEEE International Conference on Communications Workshops (ICC)*, pp. 638-643, 2016.

[17] Junfeng Guo, Zhaozhe Song, and Ying Cui. "Energy-Efficient Resource Allocation for Multi-User Mobile Edge Computing". *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1-7, 2017.

[18] Lei Yang, Jiannong Cao, Hui Cheng, and Yusheng Ji. "Multi-User Computation Partitioning for Latency Sensitive Mobile Cloud Applications". *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253-2266, 2015.

[19] Karthik Kumar, Jibang Liu, and Yung-Hsiang Lu. "A Survey of Computation Offloading for Mobile Systems". *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129-140, 2013.

[20] Selmer Martin Johnson. "Optimal two-and three-stage production schedules with setup times included". *Naval research logistics quarterly*, vol. 1, no. 1, pp. 61-68, 1954.

[21] Lei Yang, Bo Liu, Jiannong Cao, Yuvraj Sahni, and Zhenyu Wang. "Joint Computation Partitioning and Resource Allocation for Latency Sensitive Applications in Mobile Edge Clouds". *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 246-253, 2017.

[22] Jessica Oueis, Emilio Calvanese-Strinati, Stefania Sardellitti, and Sergio Barbarossa. "Small Cell Clustering for Efficient Distributed Fog Computing: A Multi-User Case". *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, pp. 1-5, 2015.

[23] Zdenek Becvar, Jan Plachy, and Pavel Mach. "Path selection using handover in mobile networks with cloud-enabled small cells". *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pp. 1480-1485, 2014.

[24] Jan Plachy, Zdenek Becvar, and Pavel Mach. "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network". *Computer Networks*, vol. 108, pp. 357-370, 2016.

[25] Jan Plachy, Zdenek Becvar, and Emilio Calvanese Strinati. "Dynamic Resource Allocation Exploiting Mobility Prediction in Mobile Edge Computing". *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1-6, 2016.

[26] 3rd Generation Partnership Project (3GPP). "3GPP TS 22.261 V16.7.0 (2019-03); 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the 5G system; Stage 1 (Release 16)". *Technical report*, 2019.

[27] Claude Elwood Shannon. "Communication in the presence of noise". *Proceedings of the IEEE*, vol. 86, no. 2, pp. 447-457, 1998.

[28] 3rd Generation Partnership Project (3GPP). "3GPP TS 36.331 V15.5.1 (2019-04); 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (Release 15)". *Technical report*, 2019.

[29] 3rd Generation Partnership Project (3GPP). "3GPP TR 36.814 V9.2.0 (2017-03); 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects; (Release 9)". *Technical report*, 2017.

[30] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. "Towards wearable cognitive assistance". *ACM Mobile systems, applications, and services*, pp. 68-81, 2014.

[31] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. "A Survey on Mobile Edge Computing: The Communication Perspective". *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.

[32] Bijan Jabbari, Yong Zhou, and Frederick Hillier. "Random walk modeling of mobility in wireless networks". *VTC '98. 48th IEEE Vehicular Technology Conference. Pathway to Global Wireless Revolution (Cat. No.98CH36151)*, vol.1, pp. 639-643, 1998.

[33] International Telecommunication Union (ITU). "Recommendation ITU-R P.525-3 (09/2016); Calculation of free-space attenuation; P Series; Radiowave propagation". *Technical report*, 2016.

[34] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. "Mobiscud: A Fast Moving Personal Cloud in the Mobile Network". *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pp. 19-24, 2015.

# Appendix

This section elaborates the equation 5.3 to obtain ratios of individual subtasks $s_m$.

$$s_1(D_1^{UL} + D_1^{comp} + D_1^{DL}) = s_1 D_1^{DL} + s_2(D_2^{UL} + D_2^{comp} + D_2^{DL}) =$$
$$= ... = \sum_{i=1}^{m-1} s_i D_i^{UL} + s_m(D_m^{UL} + D_m^{comp} + D_m^{DL}),$$

$$\sum_{j=1}^{i-1} s_j D_j^{UL} + s_i(D_i^{UL} + D_i^{comp} + D_i^{DL}) = \sum_{j=1}^{i} s_j D_j^{UL} + s_{i+1}(D_{i+1}^{UL} + D_{i+1}^{comp} + D_{i+1}^{DL}),$$

$$s_i(D_i^{comp} + D_i^{DL}) = s_{i+1}(D_{i+1}^{UL} + D_{i+1}^{comp} + D_{i+1}^{DL}),$$

$$s_{i+1} = s_i \frac{D_i^{comp} + D_i^{DL}}{D_{i+1}^{UL} + D_{i+1}^{comp} + D_{i+1}^{DL}},$$

$$s_{i+2} = s_{i+1} \frac{D_{i+1}^{comp} + D_{i+1}^{DL}}{D_{i+2}^{UL} + D_{i+2}^{comp} + D_{i+2}^{DL}},$$
$$...$$

$$s_m = s_{m-1} \frac{D_{m-1}^{comp} + D_{m-1}^{DL}}{D_m^{UL} + D_m^{comp} + D_m^{DL}},$$

$$s_m = s_{m-2} \frac{D_{m-2}^{comp} + D_{m-2}^{DL}}{D_{m-1}^{UL} + D_{m-1}^{comp} + D_{m-1}^{DL}} \cdot \frac{D_{m-1}^{comp} + D_{m-1}^{DL}}{D_m^{UL} + D_m^{comp} + D_m^{DL}},$$
$$...$$

$$s_m = s_1 \frac{D_1^{comp} + D_1^{DL}}{D_2^{UL} + D_2^{comp} + D_2^{DL}} \cdot \frac{D_2^{comp} + D_2^{DL}}{D_3^{UL} + D_3^{comp} + D_3^{DL}} \cdot \ldots \cdot \frac{D_{m-1}^{comp} + D_{m-1}^{DL}}{D_m^{UL} + D_m^{comp} + D_m^{DL}},$$

$$s_m = s_1 \frac{\prod_{i=1}^{m-1} (D_i^{comp} + D_i^{DL})}{\prod_{i=2}^{m} (D_i^{UL} + D_i^{comp} + D_i^{DL})}.$$

The first subtask $s_1$ can be expressed using 3.8:

$$\sum_{m=1}^{M} s_m = 1,$$

$$s_1 + s_1 \frac{D_1^{comp} + D_1^{DL}}{D_2^{UL} + D_2^{comp} + D_2^{DL}} + \ldots + s_1 \frac{\prod_{i=1}^{M-1} (D_i^{comp} + D_i^{DL})}{\prod_{i=2}^{M} (D_i^{UL} + D_i^{comp} + D_i^{DL})} = 1,$$

$$s_1 = \left( 1 + \sum_{i=2}^{M} \frac{\prod_{j=1}^{i-1} (D_j^{comp} + D_j^{DL})}{\prod_{j=2}^{i} (D_j^{UL} + D_j^{comp} + D_j^{DL})} \right)^{-1}.$$