National Technical University of Athens

School of Electrical and Computer Engineering

Division of Communication, Electronic and Information Engineering

Network Management and Optimal Design Laboratory

Dynamic Resource Allocation and Computational Offloading at the Network Edge for Internet of Things Applications

A dissertation submitted for the degree of Doctor of Philosophy

of

Marios Avgeris

Athens

September 29,2021

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering
Network Management and Optimal Design Laboratory

Dynamic Resource Allocation and Computational Offloading at the Network Edge for Internet of Things Applications

A dissertation submitted for the degree of Doctor of Philosophy

of

Marios Avgeris

Advisory Committee: Symeon Papavassiliou
Efstathios Sykas
Ioanna Roussaki

Approved by the seven-member committee on September 29, 2021

.....	.....	.....
S. Papavassiliou	E. Sykas	I. Roussaki
Professor, NTUA	Professor, NTUA	Assistant Professor, NTUA
.....	.....	.....
N. Mitrou	E. Varvarigos	V. Karyotis
Professor, NTUA	Professor, NTUA	Associate Professor, Ionian University

.....
P. Papadimitriou
Assistant Professor, University of Macedonia

Athens, September 2021

.....

Αυγέρης Μάριος

Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

Acknowledgements

This thesis was supported by the Special Account for Research Funding (E.L.K.E), NTUA.

(Η διδακτορική διατριβή αυτή εκπονήθηκε με χρηματοδότηση από τον Ειδικό Λογαριασμό Έρευνας Ε.Μ.Π.)

Writing this chapter and recalling the years during which I pursued my Ph.D., I feel that there are a lot of people I should thank for their support.

First of all, I would like to thank my advisor, Professor Symeon Papavassiliou, for his guidance and support throughout my Ph.D. studies. I am grateful for his trust in me as well as his valuable input to the problems that arose during my studies. He supported me in my research endeavours, always willing to help and provided the necessary encouragement. Moreover, he gave me the chance to join the NETMODE team and collaborate with many esteemed colleagues.

A special thanks is owed to Dr. Dimitris Dechouniotis. Since the beginning of my Ph.D. studies, Dimitris inspired me to study the field of computational offloading and resource allocation in modern networks. I thank him for his valuable support during my Ph.D. studies, his guidance, the productive collaboration and the many interesting discussions we had during the past years.

I am also deeply grateful to my good friends and colleagues Giorgos, Dimitris, Giannis, Kostas and Vasilis. In addition to the fruitful collaboration I had with both of them, I feel honored to have met them and consider them as my good friends who were always willing to listen to my problems and provide me with the necessary psychological support. I would also like to thank all the other colleagues in NETMODE laboratory for the precious (and fun) moments we shared the past years.

From the bottom of my heart, I would like to thank my family, my father Alexandros and my mother Sofia , for their unconditional love, understanding and support. I owe a lot to my parents for making every sacrifice in order for me to be able to pursue my academic goals and for teaching me valuable life lessons.

for my parents, Alexandros and Sofia

Abstract

In the Internet of Things (IoT) era, mobile devices possess powerful hardware and networking capabilities, however they still fall short when it comes to executing compute-intensive applications. Computation Offloading, i.e., delegating resource consuming tasks to servers located at the Network Edge, contributes towards moving to a Mobile Cloud Computing paradigm, which will potentially assist towards alleviating the computational strain from the mobile devices. The motivation for this thesis is to deal with the inherent challenges of computational offloading, the most important of which is resource allocation under constraints, while guaranteeing the Quality of Service (QoS) and Quality of Experience (QoE) delivered to the users. Throughout this thesis, Control Theory concepts are leveraged as this domain offers a plethora of tools that can be utilised to tackle the emerging challenges. Additionally, concepts from Probability Theory are exploited as well.

Specifically, in this work, an effort is made to address the following crucial research challenges in resource allocation and computational offloading: i) modeling the heterogeneous entities of the system (i.e., the infrastructure, users, applications resources and the interactions among them), ii) estimating the workload that will be offloaded during a future time window (by predicting the mobile devices' positions), iii) optimizing resource allocation by dynamically allocating (scaling) the available resources at the network edge while respecting the given constraints, e.g., guaranteed execution of the estimated workload, delivering the expected QoS/QoE and minimizing energy consumption of the edge infrastructure and, finally, iv) optimizing computational offloading strategies. The aim of this thesis is to offer solutions to the above-mentioned challenges, which can be combined into frameworks applicable to real-world edge infrastructures that will allow IoT devices and applications to reach their full potential.

To this end, a two-level dynamic resource allocation and admission control mechanism for a cluster of edge servers is developed, to offer an alternative choice to mobile users for executing their tasks. At the lower level, the dynamic behavior of edge servers is modeled by a set of linear systems, and linear controllers are designed to meet the system's constraints

and QoS metrics, while at the upper level, an optimizer tackles the problems of load balancing and application placement (bundled in Virtual Machines, VMs) towards maximizing of the number the offloaded requests. The evaluation illustrates the effectiveness of the proposed offloading mechanism regarding the performance indicators (e.g., the application's average response time) and the optimal utilization of the computational resources of the edge servers.

Then, the main mechanism of this framework is put to a test; a three-level Cyber-Physical Social System (CPSS) for early fire detection is presented, with an aim to assist public authorities to promptly identify and act on emergency situations. In general, a CPSS tightly integrates computer systems with the physical world and human activities. Specifically, at the bottom level, the system's architecture involves IoT nodes enabled with sensing and forest monitoring capabilities. Additionally, in this level, the crowd sensing paradigm is exploited to aggregate environmental information collected by end user devices present in the area of interest. Since the IoT nodes suffer from limited computational and energy resources, the resource allocation and admission control mechanism, at the middle level, facilitates the offloaded data processing, regarding possible fire incidents. At the top level, a decision-making service deployed on Cloud nodes, integrates data from various sources, including users' information on social media, and evaluates the situation's criticality. In this part, the dynamic resource scaling mechanism is designed to address the demanding QoS requirements of this IoT-enabled time and mission critical application. The experimental results indicate that the vertical and horizontal scaling on the Edge Computing layer is beneficial for both the performance and the energy consumption of the IoT nodes.

Additionally, a switching computational offloading mechanism for Industry 4.0 applications is discussed. These applications rely on mobile robotic agents that execute many complex tasks that have strict safety and time requirements. Under this setting, the Edge Computing service delivery model allows the robotic agents to offload their computationally intensive tasks to a powerful computing infrastructure in their vicinity. In this part, a novel switching offloading mechanism for such robotic applications is proposed. In particular, opportunistic offloading strategies for the path planning and localization services of mobile robots are designed. The offloading decision is based on the uncertainty of the robot's pose, the resource availability at the Edge of the network and the difficulty of the path planning.

The proposed switching offloading framework is implemented and evaluated using a robot in a real Edge Computing testbed, where the trade-off between execution time and the successful completion of the robot trajectory is highlighted.

Finally, a Markovian Random Field (MRF)-based computational offloading and resource allocation mechanism is developed. The proposed mechanism leverages switching systems for modeling the computational resources at the network edge and allocating them dynamically, while minimizing energy consumption. This mechanism consists of two repeated stages; during the first, a Markov Chain-based technique is used to predict the mobile users' movements and subsequently to estimate the offloaded workload demand. During the second, a novel MRF-based technique undertakes the balancing of the offloaded tasks to the available computational resources. These tasks cannot be executed locally, i.e., on the user devices, under the given energy constraints and for a specific QoS. The proposed framework manages to improve energy consumption in the edge infrastructure, while taking into consideration the additional network delays induced by the MRF-based load balancing. Moreover, the efficiency of the proposed scheme is evaluated via modeling and simulation and is shown to outperform a well-known task offloading solution.

Summarizing, in this thesis, dynamic resource allocation mechanisms for computational offloading are proposed, based on workload prediction, horizontal scaling, vertical scaling and workload balancing. Research is conducted, also, on formulating offloading strategies that work in harmony with these mechanisms, in order to guarantee a level of QoS and QoE, and minimize energy consumption, while the frameworks emerging from combining these techniques, are evaluated in the realistic, demanding environments of Industry 4.0, Natural Disaster Management and Smart Environments, with successful results.

Keywords: Control Theory, Resource Allocation, Computational Offloading, Admission Control, Internet of Things (IoT), Edge Computing, Cyber-Physical Systems, Industry 4.0, Switching Systems, Markovian Random Fields, Energy Consumption Minimization

Abstract in Greek

Περίληψη στα Ελληνικά

Στην εποχή του Διαδικτύου των Αντικειμένων (ΔτΑ, Internet of Things), οι κινητές συσκευές είναι εφοδιασμένες με ισχυρές υλικές και διαδικτυακές δυνατότητες· παρ' όλα αυτά, συνεχίζουν να μην μπορούν να ανταπεξέλθουν στην εκτέλεση υπολογιστικά επίπονων διεργασιών. Η Μεταφόρτωση Υπολογιστικών Διεργασιών (Computational Offloading), δηλαδή η ανάθεση των διεργασιών που καταναλώνουν πολλούς πόρους σε εξυπηρετητές τοποθετημένους στα Άκρα του Δικτύου (Network Edge), συνεισφέρει στην κατεύθυνση της υιοθέτησης του προτύπου του Κινητού Υπολογιστικού Νέφους (Mobile Cloud Computing) το οποίο δυνητικά θα βοηθήσει στην ελάφρυνση του υπολογιστικού φόρτου των κινητών συσκευών. Κίνητρο για τη συγγραφή αυτής της διατριβής αποτελεί η αντιμετώπιση των εγγενών προκλήσεων της μεταφόρτωσης υπολογιστικών διεργασιών, η σημαντικότερη από τις οποίες είναι η κατανομή πόρων υπό περιορισμούς, προσφέροντας ταυτόχρονα εγγυήσεις σχετικά με το επίπεδο ποιότητας υπηρεσιών (Quality of Service – QoS) και το επίπεδο ποιότητας εμπειρίας (Quality of Experience, QoE) που παρέχεται στους χρήστες. Στο πέρας αυτής της διατριβής γίνεται χρήση εννοιών από την Θεωρία Ελέγχου καθώς πρόκειται για ένα πεδίο το οποίο προσφέρει πληθώρα εργαλείων προς αντιμετώπιση των παραπάνω προκλήσεων. Επιπλέον, γίνεται χρήση εννοιών από την Θεωρία των Πιθανοτήτων.

Πιο συγκεκριμένα, η παρούσα διατριβή εστιάζει σε τέσσερις σημαντικές ερευνητικές περιοχές που αφορούν στην ανάπτυξη των παραπάνω τεχνικών, οι οποίες είναι: α) η μοντελοποίηση των ετερογενών οντοτήτων του εξεταζόμενου συστήματος (τύπος υποδομής-χρηστών-εφαρμογών-δικτυακών/υπολογιστικών πόρων), β) ο υπολογισμός του όγκου των διεργασιών που αναμένεται να εκτελεστούν στα άκρα του δικτύου (μέσω της πρόβλεψης των θέσεων των κινητών συσκευών), γ) η δυναμική κατανομή των διαθέσιμων πόρων της υποδομής, καθοδηγούμενη από κριτήρια όπως είναι η διατήρηση ενός συμφωνημένου επιπέδου ποιότητας υπηρεσιών και εμπειρίας, η παράλληλη φιλοξενία πολλαπλών εφαρμογών στους ίδιους διαθέσιμους πόρους, καθώς και η ελαχιστοποίηση της ενεργειακής κατανάλωσης του συστήματος και δ) η βελτιστοποίηση των στρατηγικών μεταφόρτωσης των διεργασιών στους κατανεμημένους πόρους. Στόχος αυτής της διατριβής είναι ο εντοπισμός των σημαντικότερων ζητημάτων σε αυτές τις ερευνητικές περιοχές και η ανάπτυξη κατάλληλων λύσεων για την βελτιστοποίηση των

επιμέρους διαδικασιών, δεδομένων των υφιστάμενων περιορισμών. Οι λύσεις αυτές στη συνέχεια συνδυάζονται δημιουργώντας πλαίσια τα οποία μπορούν να εγκατασταθούν σε υποδομές στα άκρα του δικτύου και να ελαφρύνουν την υπολογιστική καταπόνηση των -υπολογιστικά περιορισμένων- φορητών συσκευών, βελτιώνοντας ταυτόχρονα την ποιότητα εμπειρίας που αποκομίζουν οι χρήστες.

Για το σκοπό αυτό, αναπτύσσεται ένας διεπίπεδος μηχανισμός δυναμικής κατανομής πόρων και ελέγχου μεταφόρτωσης διεργασιών για συστάδες εξυπηρετητών στα άκρα του δικτύου. Στο χαμηλότερο επίπεδο, η δυναμική συμπεριφορά των εξυπηρετητών μοντελοποιείται με χρήση γραμμικών συστημάτων, ενώ γραμμικοί ελεγκτές σχεδιάζονται για να διατηρούν το σύστημα εντός των δοθέντων περιορισμών (π.χ. συμφωνημένο επίπεδο ποιότητας υπηρεσιών). Στο υψηλότερο επίπεδο, ένας μηχανισμός βελτιστοποίησης αναλαμβάνει την τοποθέτηση των εφαρμογών/υπηρεσιών στους διαθέσιμους εξυπηρετητές (οι οποίες βρίσκονται σε μορφή εικονικών μηχανών, Virtual Machines – VMs) και τον καταμερισμό του φόρτου εργασίας μεταξύ τους, με σκοπό την μεγιστοποίηση του αριθμού των διεργασιών που θα εκτελεστούν επιτυχώς σε αυτούς. Η αποτίμηση του εν λόγω μηχανισμού αποδεικνύει την αποτελεσματικότητά του, τόσο όσον αφορά στην προσφερόμενη ποιότητα υπηρεσιών, όσο και στην βέλτιστη διαχείριση των υπολογιστικών πόρων στα άκρα του δικτύου.

Στη συνέχεια, ο παραπάνω μηχανισμός εντάσσεται στο πλαίσιο ενός Κυβερνο-Φυσικού Κοινωνικού Συστήματος – ΚΦΚΣ (Cyber-Physical Social System – CPSS) τριών επιπέδων, το οποίο προορίζεται για τον έγκαιρο εντοπισμό πυρκαγιών. Γενικά, ένα ΚΦΚΣ αφομοιώνει τα υπολογιστικά συστήματα με τον φυσικό κόσμο και τις ανθρώπινες δραστηριότητες. Εν προκειμένω, στο χαμηλότερο επίπεδο, η αρχιτεκτονική του ΚΦΚΣ περιλαμβάνει συσκευές ΔτΑ με αισθητήρες ανίχνευσης και παρακολούθησης δασών. Επιπλέον, σε αυτό το επίπεδο, γίνεται χρήση του προτύπου της αίσθησης πλήθους (crowd sensing), κατά το οποίο συλλέγονται πληροφορίες σχετικά με το περιβάλλον από συσκευές χρηστών οι οποίοι βρίσκονται στην περιοχή ενδιαφέροντος. Δεδομένου ότι οι συσκευές ΔτΑ χαρακτηρίζονται από περιορισμένους υπολογιστικούς και ενεργειακούς πόρους, ο μηχανισμός που αναπτύχθηκε στο Κεφάλαιο 3 εγκαθίσταται στο μεσαίο επίπεδο του ΚΦΚΣ και αναλαμβάνει την μεταφόρτωση των υπολογιστικά ακριβών διεργασιών των συσκευών ΔτΑ σε μια υποδομή στα άκρα του δικτύου. Σε αυτό το επίπεδο, ο μηχανισμός δυναμικής κατανομής πόρων επιτυγχάνει την τήρηση των χρονικών απαιτήσεων απόκρισης των εφαρμογών ανίχνευσης και παρακολούθησης. Στο υψηλότερο επίπεδο, ένας μηχανισμός

λήψης αποφάσεων εγκατεστημένος σε εξυπηρετητές Υπολογιστικού Νέφους (Cloud), συλλέγει δεδομένα από τις διάφορες πηγές (συσκευές ΔτΑ, κοινωνικά δίκτυα χρηστών) και αποτιμά την κρισιμότητα της κατάστασης. Τα πειραματικά αποτελέσματα υποδεικνύουν τη σημασία του μηχανισμού δυναμικής κατανομής πόρων, τόσο στην εγγύηση της έγκαιρης εκτέλεσης των σημαντικών διεργασιών, όσο και στη μείωση της ενεργειακής κατανάλωσης των συσκευών ΔτΑ.

Επιπροσθέτως, αναπτύσσεται ένας εναλλακτικός διακοπτικός (Switching Systems -based) μηχανισμός μεταφόρτωσης διεργασιών για εφαρμογές της Βιομηχανίας 4.0. Οι εφαρμογές αυτές απευθύνονται σε ρομπότ τα οποία εκτελούν περίπλοκες διεργασίες, οι οποίες παρουσιάζουν αυστηρές απαιτήσεις τόσο σε χρονική απόκριση όσο και σε ασφάλεια. Σε αυτό το πλαίσιο, η μεταφόρτωση των διεργασιών στα άκρα του δικτύου επιτρέπει στα ρομπότ να ελαφρύνουν τον υπολογιστικό τους φόρτο, αναθέτοντας την εκτέλεση των παραπάνω διεργασιών σε μία ισχυρή υπολογιστική υποδομή σε κοντινή απόσταση. Σε αυτό το κεφάλαιο, λοιπόν, προτείνεται ένας διακοπτικός μηχανισμός μεταφόρτωσης διεργασιών, ενώ σχεδιάζονται ευκαιριακές στρατηγικές μεταφόρτωσης για εφαρμογές που αφορούν στον προγραμματισμό της πορείας και τον εντοπισμό της θέσης των ρομπότ. Η απόφαση για την μεταφόρτωση λαμβάνεται βάσει της αβεβαιότητας ως προς την τρέχουσα θέση του ρομπότ και την διαθεσιμότητα υπολογιστικών και δικτυακών πόρων στα άκρα του δικτύου την δεδομένη στιγμή. Το διακοπτικό αυτό σύστημα υλοποιείται και αξιολογείται χρησιμοποιώντας ένα πραγματικό ρομπότ σε μια πραγματική υποδομή στα άκρα του δικτύου· κατά την αξιολόγηση τονίζεται το αντιστάθμισμα ανάμεσα στο χρόνο ολοκλήρωσης των διεργασιών και την επιτυχή έκβαση της αποστολής τους.

Τέλος, μελετάται η μεταφόρτωση και ο διαμοιρασμός των διεργασιών με χρήση Μαρκοβιανών τυχαίων πεδίων. Συγκεκριμένα, στον προτεινόμενο μηχανισμό γίνεται χρήση διακοπτικών συστημάτων για την μοντελοποίηση των υπολογιστικών πόρων στα άκρα του δικτύου και την δυναμική κατανομή τους, βάσει κριτηρίων ενεργειακής κατανάλωσης. Ο μηχανισμός αποτελείται από δύο επαναλαμβανόμενα στάδια. Κατά το πρώτο, γίνεται χρήση μιας τεχνικής βασισμένης σε Μαρκοβιανές αλυσίδες, η οποία προβλέπει τις κινήσεις των χρηστών στο χώρο για την υπολογισμό του όγκου των διεργασιών που αναμένεται να μεταφορτωθούν στα άκρα του δικτύου. Κατά το δεύτερο, μια καινοτόμα τεχνική βασισμένη σε Μαρκοβιανά τυχαία πεδία αναλαμβάνει τον διαμοιρασμό των υπολογιστικών διεργασιών στους διαθέσιμους πόρους. Οι διεργασίες αυτές δεν δύνανται να εκτελεστούν τοπικά στις συσκευές τον χρηστών υπό συγκεκριμένους ενεργειακούς περιορισμούς και για συγκεκριμένο επίπεδο ποιότητας υπηρεσιών.

Ο προτεινόμενος μηχανισμός επιτυγχάνει βελτιωμένη ενεργειακή κατανάλωση, λαμβάνοντας υπόψιν τις επιπρόσθετες δικτυακές καθυστερήσεις που επιφέρει ο διαμοιρασμός των εργασιών στην υποδομή. Ακόμη, συγκρίνοντας τον προτεινόμενο μηχανισμό με μια γνωστή αντίστοιχη δουλειά στη βιβλιογραφία, επιδεικνύεται η αποτελεσματικότητά του τόσο στη βελτιστοποίηση της κατανάλωσης ενέργειας, όσο και στην ποιότητα των παρεχόμενων υπηρεσιών.

Συνοψίζοντας, στη διατριβή αυτή προτείνονται μηχανισμοί δυναμικής κατανομής πόρων για μεταφόρτωση υπολογιστικών διεργασιών, βασισμένοι στην πρόβλεψη του φόρτου εργασίας, στην οριζόντια και κατακόρυφη κλιμάκωση, καθώς και στην εξισορρόπηση φόρτου. Μελετάται, επίσης, ο σχεδιασμός στρατηγικών μεταφόρτωσης οι οποίες δουλεύουν σε αρμονία με τους παραπάνω μηχανισμούς, με σκοπό να εγγυηθούν ένα επίπεδο ποιότητας υπηρεσιών, ποιότητας εμπειρίας και να ελαχιστοποιηθεί η κατανάλωση ενέργειας, ενώ, οι ολοκληρωμένες λύσεις που προκύπτουν από τον συνδυασμό των παραπάνω τεχνικών, αξιολογούνται στα ρεαλιστικά και απαιτητικά περιβάλλοντα της Βιομηχανίας 4.0, της Διαχείρισης Φυσικών Καταστροφών και των Έξυπνων Περιβαλλόντων, με επιτυχία.

Λέξεις Κλειδιά: Θεωρία Ελέγχου, Κατανομή Πόρων, Μεταφόρτωση Υπολογιστικών Διεργασιών, Έλεγχος Μεταφόρτωσης Διεργασιών, Διαδίκτυο των Αντικειμένων (ΔτΑ), Υπολογιστική στα Άκρα του Δικτύου, Κυβερνο-Φυσικό Κοινωνικό Σύστημα (ΚΦΚΣ), Βιομηχανία 4.0, Διακοπτικά Συστήματα, Μαρκοβιανά Τυχαία Πεδία, Ελαχιστοποίηση Ενεργειακής Κατανάλωσης

Contents

List of Figures

List of Tables

Extensive Summary in Greek

Εκτεταμένη Περίληψη στα Ελληνικά

Η παρούσα διατριβή εστιάζει στην ανάπτυξη τεχνικών για τη δυναμική κατανομή πόρων και τη μεταφόρτωση υπολογιστικά απαιτητικών διεργασιών εφαρμογών του Διαδικτύου των Αντικειμένων – ΔτΑ (Internet of Things – IoT) στα άκρα του δικτύου (Network Edge). Παράδειγμα εφαρμογών που μπορούν να επωφεληθούν από τις παραπάνω τεχνικές αποτελούν οι διάφορες ρομποτικές εφαρμογές και οι εφαρμογές επαυξημένης πραγματικότητας (Augmented Reality – AR) που εκτελούνται σε κινητές συσκευές. Για την ανάπτυξη των τεχνικών αυτών, χρησιμοποιούνται πρωτίστως τεχνικές και εργαλεία από τους τομείς της Θεωρίας Συστημάτων Αυτόματου Ελέγχου, καθώς και μερικές από τη Θεωρία Πιθανοτήτων.

Πιο συγκεκριμένα, η παρούσα διατριβή εστιάζει σε τέσσερις σημαντικές ερευνητικές περιοχές που αφορούν στην ανάπτυξη των παραπάνω τεχνικών, οι οποίες είναι: α) η μοντελοποίηση των ετερογενών οντοτήτων του εξεταζόμενου συστήματος (τύπος υποδομής-χρηστών-εφαρμογών-δικτυακών/υπολογιστικών πόρων), β) ο υπολογισμός του όγκου των διεργασιών που αναμένεται να εκτελεστούν στα άκρα του δικτύου (μέσω της πρόβλεψης των θέσεων των κινητών συσκευών), γ) η δυναμική κατανομή των διαθέσιμων πόρων της υποδομής, καθοδηγούμενη από κριτήρια όπως είναι η διατήρηση ενός συμφωνημένου επιπέδου ποιότητας υπηρεσιών και εμπειρίας, η παράλληλη φιλοξενία πολλαπλών εφαρμογών στους ίδιους διαθέσιμους πόρους, καθώς και η ελαχιστοποίηση της ενεργειακής κατανάλωσης του συστήματος και δ) η βελτιστοποίηση των στρατηγικών μεταφόρτωσης των διεργασιών στους κατανεμημένους πόρους. Στόχος αυτής της διατριβής είναι ο εντοπισμός των σημαντικότερων ζητημάτων σε αυτές τις ερευνητικές περιοχές και η ανάπτυξη κατάλληλων λύσεων για την βελτιστοποίηση των επιμέρους διαδικασιών, δεδομένων των υφιστάμενων περιορισμών. Οι λύσεις αυτές στη συνέ-

χεια συνδυάζονται δημιουργώντας πλαίσια τα οποία μπορούν να εγκατασταθούν σε υποδομές στα άκρα του δικτύου και να ελαφρύνουν την υπολογιστική καταπόνηση των -υπολογιστικά περιορισμένων- φορητών συσκευών, βελτιώνοντας ταυτόχρονα την ποιότητα εμπειρίας που αποκομίζουν οι χρήστες.

Κεφάλαιο 2

Στο Κεφάλαιο 2 παρουσιάζονται οι ορισμοί των βασικών στοιχείων από τους τομείς της Θεωρίας Συστημάτων Αυτόματου Ελέγχου και της Θεωρίας Πιθανοτήτων. Η γνώση του βασικού μαθηματικού υποστρώματος που δίνεται περιεκτικά στο κεφάλαιο αυτό, είναι κρίσιμη για την κατανόηση των προβλημάτων αλλά και των αλγορίθμων που παρουσιάζονται στην συνέχεια στην προτεινόμενη διατριβή. Συγκεκριμένα, εδώ δίνονται συνοπτικά οι βασικές έννοιες γύρω από τα γραμμικά χρονικά αμετάβλητα (Linear Time-Invariant, LTI) μοντέλα, που χρησιμοποιούνται για τη μοντελοποίηση και τον έλεγχο των συστημάτων ενδιαφέροντος, καθώς και οι ορισμοί και τα θεωρήματα ευστάθειας που εφαρμόζονται για την αντιμετώπιση της δυναμικότητας τους. Ακόμα, δίνονται οι βασικοί ορισμοί γύρω από τις μαρκοβιανές αλυσίδες (Markov Chains) και τα μαρκοβιανά τυχαία πεδία (Markov Random Fields), εργαλεία που χρησιμοποιούνται για την πρόβλεψη του όγκου των διεργασιών και τη μεταφόρτωσή τους στους διαθέσιμους πόρους αντίστοιχα.

Κεφάλαιο 3

Στο Κεφάλαιο 3 περιγράφεται η ανάπτυξη του βασικού διεπίπεδου μηχανισμού δυναμικής κατανομής πόρων και ελέγχου μεταφόρτωσης διεργασιών για συστάδες εξυπηρετητών στα άκρα του δικτύου. Στο χαμηλότερο επίπεδο, η δυναμική συμπεριφορά των εξυπηρετητών μοντελοποιείται με χρήση γραμμικών συστημάτων, ενώ γραμμικοί ελεγκτές σχεδιάζονται για να διατηρούν το σύστημα εντός των δοθέντων περιορισμών (π.χ., συμφωνημένο επίπεδο ποιότητας υπηρεσιών) (*Μηχανισμός Κατακόρυφης Κλιμάκωσης*). Στο υψηλότερο επίπεδο, ένας μηχανισμός βελτιστοποίησης αναλαμβάνει την τοποθέτηση των εφαρμογών/υπηρεσιών στους διαθέσιμους εξυπηρετητές (οι οποίες βρίσκονται σε μορφή εικονικών μηχανών - EM, Virtual Machines – VMs) και τον καταμερισμό του φόρτου εργασίας μεταξύ τους, με σκοπό την μεγιστοποίηση του αριθμού των διεργασιών που θα εκτελεστούν επιτυχώς σε αυτούς (*Μηχανισμός Οριζόντιας*

Κλιμάκωσης). Η αποτίμηση του εν λόγω μηχανισμού αποδεικνύει την αποτελεσματικότητά του, τόσο όσον αφορά στην προσφερόμενη ποιότητα υπηρεσιών, όσο και στη βέλτιστη διαχείριση των υπολογιστικών πόρων στα άκρα του δικτύου.

Αναλυτικότερα, η μεταφόρτωση υπολογιστικών διεργασιών μετριάζει την κατανάλωση ενέργειας των υλικά-περιορισμένων κινητών συσκευών αναθέτοντας την εκτέλεση των υπολογιστικά ακριβών διεργασιών σε μία συστάδα εξυπηρετητών στα άκρα του δικτύου, οι οποίοι είναι τοποθετημένοι στην χωρική εγγύτητα των χρηστών. Η τοποθέτηση αυτή επιτρέπει την - χαμηλή σε καθυστέρηση - πρόσβαση στους εξυπηρετητές, σε αντίθεση με την πρόσβαση σε απομακρυσμένους εξυπηρετητές (π.χ., Υπολογιστικού Νέφους) μέσω του Διαδικτύου, η οποία εμφανίζεται απρόβλεπτη όσον αφορά τους χρόνους απόκρισης. Η εικόνα 3.1 παρουσιάζει την αρχιτεκτονική μεταφόρτωσης υπολογιστικών διεργασιών η οποία μελετάται σε αυτό το κεφάλαιο. Συγκεκριμένα, η κίνηση η οποία μεταφορτώνεται και η οποία δημιουργείται από τις κινητές συσκευές, κατευθύνεται στον Μηχανισμό Οριζόντιας Κλιμάκωσης μέσω ενός τοπικού ασύρματου σημείου πρόσβασης (WiFi, 4G, LTE). Εκεί διενεργείται η υψηλότερη διαδικασία ελέγχου του προτεινόμενου μηχανισμού, κατά την οποία επιλέγεται η κατάλληλη τοποθέτηση ΕΜ σε κάθε εξυπηρετητή που είναι απευθείας συνδεδεμένος σε αυτό και εν συνεχεία διαμοιράζεται σε αυτούς η μεταφορτωμένη κίνηση κατάλληλα. Η διαδικασία αυτή καθορίζει όχι μόνο το πλήθος των εξυπηρετητών που θα ενεργοποιηθούν, αλλά και το πλήθος των ΕΜ που θα τοποθετηθούν σε αυτούς, καθώς και την κατάσταση λειτουργίας τους. Ακόμη, η διαδικασία αυτή έχει προληπτικό χαρακτήρα, καθώς γίνεται χρήση ενός εσωτερικού μηχανισμού πρόβλεψης του φόρτου που θα μεταφορτωθεί (*Προβλεπτής Φόρτου*), ο οποίος δύναται να υπολογίσει την κίνηση που θα κληθεί το σύστημα να διαχειριστεί, για ένα παράθυρο χρόνου στο μέλλον. Η διαδικασία πρόβλεψης αυτή τροφοδοτείται από το *Σύστημα Παρακολούθησης*, το οποίο είναι υπεύθυνο για την συλλογή δεδομένων που αφορούν τόσο στην δικτυακή κίνηση (π.χ., πλήθος αιτήσεων προς μεταφόρτωση, χρόνοι απόκρισης του συστήματος), όσο και στη χρησιμοποίηση των διαθέσιμων πόρων (π.χ., ποσοστό χρησιμοποίησης της κεντρικής μονάδας επεξεργασίας ενός εξυπηρετητή) στο άμεσο παρελθόν.

Σε χαμηλότερο επίπεδο, κάθε εξυπηρετητής είναι εξοπλισμένος με έναν *Τοπικό Ελεγκτή*, ο οποίος δύναται να δημιουργήσει, να εκτελέσει, να κλιμακώσει και να σταματήσει την εκτέλεση των ΕΜ που έχουν ανατεθεί σε κάθε εφαρμογή, βοηθώντας έτσι στην υλοποίηση των επιλεγμένων τοποθετήσεων ΕΜ του παραπάνω μηχανισμού για το τρέχον χρονικό παράθυρο. Επιπροσθέτως, σε αυτό το επίπεδο υλοποιείται μια διαδικασία ελέγχου κατά την οποία κλιμακώνονται

κάθετα οι ΕΜ, βάσει των δεδομένων που συλλέγονται στο Σύστημα Παρακολούθησης. Κατά αυτόν τον τρόπο, βεβαιώνεται ότι οι ΕΜ παραμένουν εντός των επιλεγμένων καταστάσεων λειτουργίας, παρέχοντας έτσι μια εγγύηση για ελάχιστους και σταθερούς χρόνους απόκρισης της εφαρμογής την οποία εκτελούνε.

Η εικόνα 3.2 παρουσιάζει τη ροή εργασιών του προτεινόμενου μηχανισμού μεταφόρτωσης υπολογιστικών διεργασιών. Σύμφωνα με αυτή την προσέγγιση, η λειτουργία των ΕΜ μοντελοποιείται από ένα σύνολο Γραμμικών Χρονικά Αμετάβλητων (ΓΧΑ) συστημάτων τα οποία υπόκεινται σε επιπρόσθετες εξωγενείς διαταραχές και τα οποία έχουν τη μορφή της εξίσωσης (3.1). Οι παράμετροι των συστημάτων αυτών αναγνωρίζονται μέσω πειραματικών διεργασιών. Αρχικά, υπολογίζεται για κάθε ΓΧΑ σύστημα ένα εφικτό σημείο ισορροπίας για το ονομαστικό, χωρίς διαταραχές μοντέλο του ΕΜ. Κάθε σημείο ισορροπίας αντιστοιχεί σε μια κατάσταση λειτουργίας («προφίλ») ενός ΕΜ χωρίς να λαμβάνονται υπόψιν διαταραχές. Για παράδειγμα, μια κατάσταση ισορροπίας μπορεί να αντιστοιχεί σε δυνατότητα εκτέλεσης 3 αιτήσεων μεταφόρτωσης ανά δευτερόλεπτο, με ταυτόχρονη εγγύηση χρησιμοποίησης (το πολύ) του 20% της παρεχόμενης υπολογιστής ισχύος και ενός μέσου χρόνου απόκρισης της εφαρμογής που δεν ξεπερνά τα 3 δευτερόλεπτα. Για κάθε ένα σημείο ισορροπίας σχεδιάζεται ένας γραμμικός ελεγκτής ανατροφοδότησης κατάστασης, εντός του Τοπικού Ελεγκτή, λαμβάνοντας υπόψιν τις διαταραχές του συστήματος. Συγκεκριμένα, ρυθμίζοντας την παρεχόμενη υπολογιστική ισχύ και το πλήθος των αιτήσεων προς μεταφόρτωση (που θα ανατεθούν σε κάθε ΕΜ) σχεδιάζεται ένας ελεγκτής έτσι ώστε το σύστημα κλειστού βρόγχου: α) να είναι ευσταθές, β) να ικανοποιεί τους περιορισμούς και τις προδιαγραφές ποιότητας υπηρεσιών, για κάθε χρονική στιγμή και για κάθε αρχική συνθήκη, ξεκινώντας από το σύνολο των περιορισμών και γ) να συμπεριφέρεται βέλτιστα στη σταθερή του κατάσταση. Δεδομένου ότι ο προτεινόμενος μηχανισμός διαχείρισης πόρων προσφέρει εγγυημένο χρόνο απόκρισης της εφαρμογής στους χρήστες, η απόφαση μεταφόρτωσης υπολογιστικών διεργασιών μετατρέπεται σε μια απλή σύγκριση ανάμεσα στον εκτιμώμενο χρόνο εκτέλεσης της εφαρμογής στη συσκευή του χρήστη και στον εγγυημένο χρόνο απόκρισης των εξυπηρετητών στα άκρα του δικτύου. Τα παραπάνω συνοψίζονται φορμαλιστικά στα προβλήματα **P1, P2**. Για την επίλυση αυτών των προβλημάτων, σχεδιάζεται και λύνεται ένα πρόβλημα βελτιστοποίησης ((3.15a)-(3.15i)), από τη λύση του οποίου προκύπτει τόσο η κατάσταση λειτουργίας των ΕΜ όσο και ο έλεγχος που θα εφαρμοστεί. Το σύνολο των καταστάσεων λειτουργίας στο οποίο εγγυάται ο μηχανισμός ότι θα παραμείνουν οι ΕΜ, ξεκινώντας από το

σημείο ισορροπίας και παρά τις όποιες (φραγμένες) διαταραχές, αποτελεί το «ελάχιστο θετικά αμετάβλητο σύνολο», δίνεται από την εξίσωση (3.12) και αποδεικνύεται ότι προκύπτει από το όριο της ακολουθίας των προσβάσιμων συνόλων.

Στο υψηλότερο επίπεδο, ο Μηχανισμός Οριζόντιας Κλιμάκωσης στοχεύει στο να συμβιβάσει τους αμοιβαία αποκλειόμενους στόχους της επίτευξης επιδόσεων και της ελαχιστοποίησης χρήσης πόρων. Συγκεκριμένα, δεδομένου ότι οι πόροι των εξυπηρετητών στα άκρα του δικτύου δεν βρίσκονται σε αφθονία, οι ανεξέλεγκτες απαιτήσεις για επιδόσεις από κάθε εφαρμογή θα οδηγούσαν σε κατανομή μεγάλου αριθμού υπολογιστικών πόρων σε όλες τις αντίστοιχες ΕΜ, πράγμα ανέφικτο. Γι' αυτόν το λόγο, ο μηχανισμός αυτός είναι υπεύθυνος για την βελτιστοποίηση της τοποθέτησης των ΕΜ (ελαχιστοποίηση κατανάλωσης ενέργειας/χρησιμοποίησης πόρων υποδομής) και την κατάλληλη κατανομή των αιτήσεων προς μεταφόρτωση μεταξύ τους. Για κάθε εφαρμογή, ο Μηχανισμός Οριζόντιας Κλιμάκωσης δέχεται ως είσοδο μία πρόβλεψη των αιτήσεων προς μεταφόρτωση, από τον Προβλεπτή Φόρτου, και το σύνολο των εφικτών καταστάσεων λειτουργίας που έχουν υπολογιστεί στον ελεγκτή ανατροφοδότησης κατάστασης. Αξίζει εδώ να σημειωθεί ότι η λειτουργία του Προβλεπτή Φόρτου βασίζεται σε ένα γραμμικό εκθετικό φίλτρο εξομάλυνσης Holt, το οποίο συλλαμβάνει τη γραμμική τάση χρονοσειρών και το οποίο περιγράφεται από την εξίσωση (3.17). Στη συνέχεια, βασισμένος σε αυτήν την πληροφορία, ο Μηχανισμός Οριζόντιας Κλιμάκωσης αποφασίζει τον ελάχιστο αριθμό των εξυπηρετητών που πρέπει να ενεργοποιηθούν και την τοπολογία των ΕΜ που θα τοποθετηθούν σε αυτούς, για να ικανοποιηθεί η συνολική ζήτηση για μεταφόρτωση διεργασιών, από κάθε εφαρμογή, και να επιτευχθεί το ζητούμενο επίπεδο ποιότητας υπηρεσιών. Δεδομένου ότι το σύνολο των εξυπηρετητών σε μια υποδομή στα άκρα του δικτύου είναι μικρό σε πλήθος, προτείνεται μια ευρυστική λύση, η οποία μπορεί να προσφέρει τα επιθυμητά αποτελέσματα με μικρό υπολογιστικό κόστος στο παραπάνω πρόβλημα. Έτσι, προτείνεται ένας αλγόριθμος αποτελούμενος από δύο βήματα: στο πρώτο βήμα, υπολογίζονται όλες οι εφικτές τοποθετήσεις ΕΜ σε έναν εξυπηρετητή, οι οποίες βασίζονται στις καταστάσεις λειτουργίας που έχουν υπολογιστεί προηγουμένως. Στο δεύτερο βήμα υπολογίζεται ο αριθμός των εξυπηρετητών που θα ενεργοποιηθεί, λύνοντας το πρόβλημα μικτού ακέραιου γραμμικού προγραμματισμού που περιγράφεται στις εξισώσεις (3.16a)-(3.16e).

Αυτή η συνεργασία των δύο επιπέδων ελέγχου, εξασφαλίζει ότι η επιλεγμένη κατάσταση λειτουργίας της κάθε ΕΜ από τον Μηχανισμό Οριζόντιας Κλιμάκωσης θα υλοποιηθεί από τον

αρμόδιο ελεγκτή ανατροφοδότησης. Εν συντομία, λοιπόν, οι βασικές συνεισφορές και οι διαφοροποίηση του προτεινόμενου μηχανισμού σε αυτό το κεφάλαιο συνοψίζονται ως εξής:

1. η προτεινόμενη μοντελοποίηση μπορεί να συλλάβει με ακρίβεια τη δυναμική συμπεριφόρα των ΕΜ της κάθε εφαρμογής, κάτω από ποικίλες καταστάσεις λειτουργίας.

2. ένα πλήθος εφικτών καταστάσεων λειτουργίας μπορεί να υπολογιστεί, λαμβάνοντας υπόψιν τα διάφορα κόστη στην επίδοση και στη χρησιμοποίηση των πόρων, πράγμα το οποίο επιτρέπει τον σχεδιασμό διαφορετικών στρατηγικών ελέγχου για διαφορετικά ζεύγη φόρτου εργασίας και εφαρμογών.

3. παρέχονται φορμαλιστικές εγγυήσεις σχετικά με την κατανομή πόρων και τις προδιαγραφές στην ποιότητα υπηρεσιών της εφαρμογής.

4. ενεργοποιείται ο ελάχιστος αριθμός των εξυπηρετητών στα άκρα του δικτύου για να ικανοποιηθεί ο συνολικός φόρτος εργασίας όλων των εφαρμογών, με βάση το σύνολο των εφικτών καταστάσεων λειτουργίας στο χαμηλότερο επίπεδο.

Η αξιολόγηση του παραπάνω διεπίπεδου μηχανισμού μεταφόρτωσης υπολογιστικών διεργασιών και κατανομής πόρων, επιβεβαιώνει την επιτυχία της προτεινόμενης προσέγγισης στην εγγύηση της ευστάθειας των χρόνων απόκρισης των εφαρμογών εντός ενός αποδεκτού περιθωρίου. Ακόμη, τονίζεται η επίτευξη της βελτιστοποίησης της κατανομής πόρων, όσον αφορά στους εξυπηρετητές στα άκρα του δικτύου που ενεργοποιούνται για να εξυπηρετήσουν τον φόρτο εργασίας από αιτήσεις μεταφόρτωσης. Συγκεκριμένα, η αξιολόγηση ξεκινά με τον υπολογισμό της υπολογιστικής πολυπλοκότητας του Μηχανισμού Οριζόντιας Κλιμάκωσης, σε σχέση με τις κυρίαρχες παραμέτρους του βασικού προβλήματος βελτιστοποίησης. Η εικόνα 3.3 αποτυπώνει την αύξηση του χρόνου υπολογισμού των λύσεων συναρτήσει της αύξησης του αριθμού των εφαρμογών, των εφικτών καταστάσεων λειτουργίας των ΕΜ και των διαθέσιμων εξυπηρετητών στα άκρα του δικτύου, ο αριθμός των οποίων φαίνεται να έχει τη σημαντικότερη επίδραση στον συνολικό χρόνο υπολογισμού της λύσης. Υπενθυμίζουμε όμως ότι αυτό δεν δημιουργεί πρόβλημα στη συγκεκριμένη εφαρμογή, καθώς ο αριθμός των διαθέσιμων εξυπηρετητών στα άκρα του δικτύου είναι χαμηλός σε αντίστοιχες υποδομές. Προχωρώντας στο βασικό μέρος της πειραματικής αξιολόγησης, στα πρώτα γραφήματα των εικόνων 3.5a, 3.6a και 3.7a και στα πρώτα των 3.5b, 3.6b και 3.7b, γίνεται φανερή η ικανότητα του μηχανισμού να διατηρεί το μέσο

χρόνο απόκρισης των εφαρμογών εντός του ελάχιστου θετικά αμετάβλητου συνόλου, εντός των δοθέντων περιορισμών και παρά τις διακυμάνσεις στις αιτήσεις μεταφόρτωσης. Στα μεσαία γραφήματα των εικόνων 3.5-3.7, αποτυπώνεται η ικανότητα του Μηχανισμού Οριζόντιας Κλιμάκωσης να ανταποκρίνεται σε αυτές ακριβώς τις διακυμάνσεις, διαλέγοντας δυναμικά τις κατάλληλες τοπολογίες ΕΜ ως απόκριση. Στα τελευταία γραφήματα των παραπάνω εικόνων φαίνεται, επίσης, η χρησιμοποίηση της παρεχόμενης υπολογιστικής ισχύος για κάθε ΕΜ.

Επιπλέον, στο πλαίσιο της αξιολόγησης του παραπάνω διεπίπεδου μηχανισμού γίνεται η σύγκριση του με το [1], μία δουλειά που στοχεύει στην εξοικονόμηση ενέργειας κατά τη μεταφόρτωση υπολογιστικά ακριβών διεργασιών στα άκρα του δικτύου, χρησιμοποιώντας όμως εξυπηρετητές με σταθερή κατανομή υπολογιστικής ισχύος. Η απόφαση για τη μεταφόρτωση αυτών των διεργασιών λαμβάνεται, βάσει μιας συμφωνίας επιπέδου εξυπηρέτησης σχετικά με το μέσο χρόνο απόκρισης της εφαρμογής. Στην πρώτη σειρά της εικόνας 3.9, γίνεται εμφανής η δυνατότητα του προτεινόμενου μηχανισμού στο να διατηρεί τον μέσο χρόνο απόκρισης εντός των αποδεκτών ορίων, συγκριτικά με το [1], το οποίο παρουσιάζει παραβάσεις στο επίπεδο εξυπηρέτησης, για την ίδια διακύμανση φόρτου, και 20% λιγότερη συνολική εξυπηρέτηση αιτημάτων μεταφόρτωσης.

Κεφάλαιο 4

Στη συνέχεια, στο Κεφάλαιο 4, ο μηχανισμός δυναμικής κατανομής πόρων και ελέγχου μεταφόρτωσης διεργασιών που παρουσιάστηκε στο Κεφάλαιο 3, εντάσσεται στο πλαίσιο ενός Κυβερνο-Φυσικού Κοινωνικού Συστήματος – ΚΦΚΣ (Cyber-Physical Social System – CPSS) τριών επιπέδων, το οποίο προορίζεται για τον έγκαιρο εντοπισμό πυρκαγιών. Γενικά, ένα ΚΦΚΣ αφομοιώνει τα υπολογιστικά συστήματα στον φυσικό κόσμο και τις ανθρώπινες δραστηριότητες. Εν προκειμένω, στο χαμηλότερο επίπεδο, η αρχιτεκτονική του ΚΦΚΣ περιλαμβάνει συσκευές ΔτΑ με αισθητήρες ανίχνευσης και παρακολούθησης δασών. Επιπλέον, σε αυτό το επίπεδο, γίνεται χρήση του προτύπου της αίσθησης πλήθους (crowd sensing), κατά το οποίο συλλέγονται πληροφορίες σχετικά με το περιβάλλον από συσκευές χρηστών, οι οποίοι βρίσκονται στην περιοχή ενδιαφέροντος. Δεδομένου ότι οι συσκευές ΔτΑ χαρακτηρίζονται από περιορισμένους υπολογιστικούς και ενεργειακούς πόρους, ο μηχανισμός που αναπτύχθηκε στο Κεφάλαιο 3 εγκαθίσταται στο μεσαίο επίπεδο του ΚΦΚΣ και αναλαμβάνει τη μεταφόρτωση των υπολογιστικά ακριβών διεργασιών των συσκευών ΔτΑ σε μια υποδομή στα άκρα του δικτύου. Σε αυτό το επί-

πεδο, ο μηχανισμός δυναμικής κατανομής πόρων επιτυγχάνει την τήρηση των χρονικών απαιτήσεων απόκρισης των εφαρμογών ανίχνευσης και παρακολούθησης. Στο υψηλότερο επίπεδο, ένας μηχανισμός λήψης αποφάσεων, εγκατεστημένος σε εξυπηρετητές Υπολογιστικού Νέφους (Cloud), συλλέγει δεδομένα από τις διάφορες πηγές (συσκευές ΔτΑ, κοινωνικά δίκτυα χρηστών) και αποτιμά την κρισιμότητα της κατάστασης. Τα πειραματικά αποτελέσματα υποδεικνύουν την σημαντική συμβολή του μηχανισμού δυναμικής κατανομής πόρων, τόσο στην εγγύηση της έγκαιρης εκτέλεσης των σημαντικών διεργασιών, όσο και στη μείωση της ενεργειακής κατανάλωσης των συσκευών ΔτΑ. Οι βασικές συνεισφορές αυτού του κεφαλαίου συνοψίζονται ως εξής:

1. σχεδιασμός και υλοποίηση ενός μηχανισμού κατακόρυφης κλιμάκωσης: όπως έχει αναφερθεί και προηγουμένως, η ταυτόχρονη συνύπαρξη περισσότερων της μίας εφαρμογής σε εξυπηρετητές στα άκρα του δικτύου, δύναται να θέσει σε ρίσκο το επίπεδο της ποιότητας των προσφερόμενων υπηρεσιών για τον έγκαιρο εντοπισμό καταστάσεων έκτακτης ανάγκης, λόγω των εκ φύσεως περιορισμένων πόρων που είναι διαθέσιμοι σε αντίστοιχες υποδομές.

2. σχεδιασμός και υλοποίηση ενός μηχανισμού οριζόντιας κλιμάκωσης: στην ίδια κατεύθυνση, αυτός ο μηχανισμός βελτιστοποίησης είναι υπεύθυνος για την ενεργοποίηση / απενεργοποίηση κάθε εξυπηρετητή, την τοποθέτηση των ΕΜ των εφαρμογών μέσα σε αυτούς και τον διαμοιρασμό του εισερχόμενου φόρτου από αιτήσεις μεταφόρτωσης των συσκευών ΔτΑ.

3. σχεδιασμός και υλοποίηση ενός μηχανισμού λήψης αποφάσεων στο υπολογιστικό νέφος: ανάμεσα στις βασικότερες προκλήσεις για τον έγκαιρο εντοπισμό καταστάσεων έκτακτης ανάγκης, βρίσκεται ο πλούτος των δεδομένων που συλλέγονται από τις διάφορες πηγές (μεταφορτωμένα δεδομένα από αισθητήρες ή ανθρώπους), η αποδοτική και γρήγορη επεξεργασία τους και τέλος η ορθή εκτίμηση του επιπέδου κρισιμότητας της κατάστασης. Σε αυτό το κεφάλαιο, ο μηχανισμός λήψης αποφάσεων που σχεδιάζεται, στοχεύει στο να συνδυάσει δεδομένα από ετερογενείς πηγές, όπως φωτογραφίες τραβηγμένες από συσκευές ΔτΑ, πληροφορίες από δορυφόρους, ιστορικά δεδομένα καιρικών συνθηκών και κοινωνικών μέσων δικτύωσης, ενώ ταυτόχρονα να παρέχει εγκαίρως πορίσματα για την κρισιμότητα των καταστάσεων.

Κατά τη σχεδίαση του προτεινόμενου ΚΦΚΣ αναγνωρίστηκαν οι παρακάτω απαιτήσεις, οι οποίες αποδεικνύουν τη χρησιμότητα ενός κλιμακώσιμου μηχανισμού για μεταφόρτωση υπολογιστικά ακριβών διεργασιών σε εξυπηρετητές στα άκρα του δικτύου:

- Έγκαιρος εντοπισμός και αναγνώριση καταστάσεων έκτακτης ανάγκης.
- Βέλτιστη χρησιμοποίηση των πόρων των συσκευών ΔτΑ.
- Δυνατότητα αντιμετώπισης των αναγκών των εφαρμογών για γρήγορη κλιμάκωση.
- Διαλειτουργικότητα των διάφορων αισθητήρων σε επίπεδο δεδομένων.
- Προστασία προσωπικών δεδομένων.

Στην εικόνα 4.1 παρουσιάζεται η αρχιτεκτονική του προτεινόμενου ΚΦΚΣ, η οποία αποτελείται πρωτίστως από δύο βασικές συνιστώσες: έναν μηχανισμό δυναμικής κατανομής πόρων και ελέγχου μεταφόρτωσης διεργασιών και έναν ευφυή μηχανισμό λήψης αποφάσεων, και δευτερευόντως από τέσσερις υφιστάμενες συνιστώσες, οι οποίες αλληλεπιδρούν μεταξύ τους και συνεισφέρουν στην αντιμετώπιση των καταστάσεων έκτακτης ανάγκης: τις συσκευές ΔτΑ, τους αισθητήρες, τα μέσα κοινωνικής δικτύωσης και τις δημόσιες αρχές. Αν και στο συγκεκριμένο κεφάλαιο γίνεται εξειδίκευση του ΚΦΚΣ στην έγκαιρη αντιμετώπιση πυρκαγιών σε δάση, η προσαρμογή του σε άλλες καταστάσεις έκτακτης ανάγκης είναι εφικτή.

Ο μηχανισμός δυναμικής κατανομής πόρων και ελέγχου μεταφόρτωσης διεργασιών που σχεδιάζεται και υλοποιείται σε αυτό το κεφάλαιο (εικόνα 4.2), απευθύνεται σε εφαρμογές ανίχνευσης πυρκαγιών οι οποίες είναι πακεταρισμένες σε εικονικά «δοχεία» (Containerized Applications, ΕΔ), αντί για ΕΜ που χρησιμοποιήθηκαν στο προηγούμενο κεφάλαιο, η ανάπτυξη των οποίων, εν προκειμένω, βασίζεται σε τεχνικές μηχανικής μάθησης στην αναγνώριση εικόνων. Τα αιτήματα μεταφόρτωσης αυτών των εφαρμογών απο τις συσκευές ΔτΑ στους εξυπηρετητές στα άκρα του δικτύου κατευθύνονται σε έναν *Κεντρικό Ελεγκτή*, ο οποίος αποτελεί μία επέκταση του Μηχανισμού Οριζόντιας Κλιμάκωσης που παρουσιάστηκε στο Κεφάλαιο 3. Ο Κεντρικός Ελεγκτής επιλέγει με παρόμοιο τρόπο την τοπολογία των ΕΔ που θα υλοποιηθεί στους εξυπηρετητές και αναλόγως κατανέμει τον εισερχόμενο φόρτο εργασίας, για το ερχόμενο χρονικό παράθυρο (οριζόντια κλιμάκωση). Αντίστοιχα δομικά στοιχεία της αρχιτεκτονικής του Κεφαλαίου 3, όπως ο *Προβλεπτής Φόρτου* και το *Σύστημα Παρακολούθησης*, τα συναντάμε και

σε αυτήν την αρχιτεκτονική και οι έξοδοί τους αποτελούν τις εισόδους ενός νέου δομικού στοιχείου, του *Βελτιστοποιητή* (βλ. εικόνα 4.4), ο οποίος αναλαμβάνει την βελτιστοποίηση της κατανομής πόρων στην υποδομή στα άκρα του δικτύου, λύνοντας διαδοχικά δύο μικτά ακέραια προβλήματα γραμμικής βελτιστοποίησης. Το πρώτο αφορά στην ενεργοποίηση των ελάχιστων εξυπηρετητών που θα χρειαστούν για την εξυπηρέτηση του εκτιμώμενου φόρτου εργασίας και η μορφή του δίνεται από την εξίσωση (4.5). Το δεύτερο αφορά στην ελαχιστοποίηση των πόρων που παρέχονται σε κάθε ΕΔ για την ικανοποίηση του φόρτου που θα τους ανατεθεί και έχει τη μορφή που δίνεται από την εξίσωση (4.6). Επιπροσθέτως, στο τέλος κάθε χρονικού παραθύρου, τα αποτελέσματα της αναγνώρισης εικόνων από τα αιτήματα που μεταφορτώθηκαν και εκτελέστηκαν επιτυχώς, χρησιμοποιούνται για να παρθεί η απόφαση σχετικά με την κρισιμότητα της κατάστασης. Επίσης, ο χρόνος μεταξύ της λήψης των φωτογραφιών που θα μεταφορτωθούν και της λήψης της απόφασης, ορίζεται ως ο χρόνος απόκρισης της εφαρμογής.

Σε ένα χαμηλότερο επίπεδο ελέγχου, κάθε εξυπηρετητής στα άκρα του δικτύου είναι εξοπλισμένος με έναν *Τοπικό Ελεγκτή*, ο οποίος συγκεντρώνει τα ιστορικά στοιχεία λειτουργίας των ΕΔ που βρίσκονται εγκατεστημένα σε αυτόν, καθώς και τις προβλέψεις για τον μελλοντικό φόρτο εργασίας που θα κληθεί να εκτελέσει και αντίστοιχα ρυθμίζει τους πόρους που παρέχονται στα ΕΔ για το επόμενο χρονικό παράθυρο (κατακόρυφη κλιμάκωση), υλοποιώντας τις αποφάσεις του Κεντρικού Ελεγκτή. Η λειτουργία των ΕΔ μοντελοποιείται με χρήση διακοπτικών συστημάτων (Switching Systems), με το κριτήριο αλλαγής να είναι ο αριθμός των κεντρικών μονάδων επεξεργασίας που παρέχονται σε κάθε ΕΔ. Η μοντελοποίηση αυτή, δύναται να συλλάβει τη δυναμικότητα της λειτουργίας των ΕΔ, ενώ επιτρέπει την εύκολη κατανομή πόρων. Συγκεκριμένα, η λειτουργία ενός ΕΔ περιγράφεται από ένα διακριτό γραμμικό σύστημα της μορφής (4.1). Για την ευσταθή λειτουργία των ΕΔ γύρω από μια συγκεκριμένη περιοχή, αναγνωρίζονται οι εφικτές καταστάσεις λειτουργίας τους (σημεία ισορροπίας του γραμμικού συστήματος που τα χαρακτηρίζει), λύνοντας το πρόβλημα γραμμικού προγραμματισμού (4.2), για διάφορες τιμές των περιορισμών. Στη συνέχεια, για κάθε εφικτή κατάσταση λειτουργίας, σχεδιάζεται ένας ελεγκτής ανατροφοδότησης κατάστασης της μορφής (4.3), με σκοπό την επίτευξη των απαιτήσεων σε χρόνο απόκρισης. Στην εικόνα 4.4 συνοψίζεται η λειτουργία ενός Τοπικού Ελεγκτή. Η συνύπαρξη αυτών των δύο ιεραρχικών επιπέδων ελέγχου και κλιμάκωσης εξασφαλίζει την ομαλή λειτουργία των ΕΔ εντός των συμφωνηθέντων επιπέδων ποιότητας υπηρεσιών, όσον αφορά το χρόνο απόκρισης των εφαρμογών.

Από την άλλη, ο ευφυής μηχανισμός λήψης αποφάσεων τοποθετείται στο υπολογιστικό νέφος και δέχεται τα αποτελέσματα της αναγνώρισης των εικόνων από τον Κεντρικό Ελεγκτή. Επιπλέον, ένας *Μηχανισμός Συλλογής Δεδομένων* χρησιμοποιείται για να συλλέξει δεδομένα από διάφορες πηγές, τα οποία θα αποσταλούν στον *Αλγόριθμο Απόφασης*, ο οποίος θα εφαρμόσει λογικούς κανόνες σε αυτά, με σκοπό να συμπεράνει το επίπεδο κρισιμότητας της κατάστασης έγκαιρα. Όπως απεικονίζεται στην εικόνα 4.3, οι κύριες πηγές δεδομένων που έχουν συμπεριληφθεί σε αυτόν τον μηχανισμό είναι: α) τα αποτελέσματα της επεξεργασίας των φωτογραφιών από τις ΔτΑ, μέσω του Κεντρικού Ελεγκτή (ώρα, συντεταγμένες, αποτέλεσμα κρισιμότητας), β) καιρικά δεδομένα από το Ευρωπαϊκό Σύστημα Πληροφοριών για Πυρκαγιές Δασών (European Forest Fire Information System), τα οποία συγκροτούν πέντε κατηγορίες πιθανότητας πυρκαγιάς ανά δάσος (χαμηλή, μέτρια, υψηλή, πολύ υψηλή, ακραία) και γ) δεδομένα από μέσα κοινωνικής δικτύωσης και συγκεκριμένα από το Twitter. Ο Αλγόριθμος Απόφασης ομογενοποιεί τα παραπάνω δεδομένα πριν να εξαγάγει το συνολικό βαθμό κρισιμότητας της κατάστασης, ο οποίος δίνεται από την εξίσωση (4.7).

Η αξιολόγηση του παραπάνω μηχανισμού έγινε σε μια πραγματική εγκατάσταση στο Εργαστήριο Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής (NETMODE) της Σχολής Ηλεκτρολόγων Μηχ. & Μηχ. Υπολογιστών του Ε.Μ.Π., ενώ για τις εφαρμογές οι οποίες επιλέχθηκαν, αναπτύχθηκαν δύο αλγόριθμοι αναγνώρισης πυρκαγιών σε εικόνες από δάση. Ως συσκευές ΔτΑ, χρησιμοποιήθηκαν συσκευές Raspberry Pi, οι οποίες ανέλαβαν το ρόλο μη επανδρωμένων αεροσκαφών που τραβούσαν τις φωτογραφίες και μεταφόρτωναν την επεξεργασία τους σε εξυπηρετητές στα άκρα του δικτύου. Στις εικόνες 4.5 και 4.6, παρατηρούμε ότι ο Κεντρικός Ελεγκτής κλιμακώνει οριζόντια τους διαθέσιμους πόρους, ενεργοποιώντας επιπλέον εξυπηρετητές όταν αυτό είναι αναγκαίο (π.χ., όταν υπάρχει υψηλή πιθανότητα πυρκαγιάς), για να ικανοποιήσει τις αυξημένες ανάγκες σε υπολογιστική ισχύ λόγω της αύξησης των αιτημάτων μεταφόρτωσης. Στον Πίνακα 4.1, φαίνονται οι περιοχές λειτουργίας που έχουν αναγνωριστεί για τα ΕΔ που τοποθετούνται μέσα σε αυτούς τους εξυπηρετητές. Όσον αφορά στην κατακόρυφη κλιμακωσιμότητα του μηχανισμού, οι Τοπικοί Ελεγκτές κάθε εξυπηρετητή απορρίπτουν περίπου το 20% των εισερχόμενων αιτημάτων μεταφόρτωσης διεργασιών, πράγμα που οφείλεται στα πιθανά σφάλματα του Προβλεπτή Φόρτου. Οι δύο αυτές διαδικασίες ελέγχου και κλιμάκωσης, επιτυγχάνουν να διατηρήσουν το μέσο χρόνο απόκρισης των εφαρμογών εντός των επιτρεπτών ορίων, τα οποία καθορίζονται από το συμφωνηθέν επίπεδο ποιότητας παροχής

υπηρεσιών των εν λόγω εφαρμογών, παρά τις διακυμάνσεις του συνολικού φόρτου εργασίας. Επιπλέον, γίνεται μια υποτυπώδης σύγκριση του δυναμικού αυτού μηχανισμού με μία στατική κατανομή πόρων, η οποία φαίνεται επίσης στις εικόνες 4.5 και 4.6 και η οποία πάσχει από τα προβλήματα της υπερ- και υπο- προμήθειας πόρων στους εξυπηρετητές στα άκρα του δικτύου, ανάλογα με τις αυξομειώσεις του φόρτου εργασίας. Αυτά τα προβλήματα είναι εξαιρετικά σημαντικά σε τέτοιου είδους εφαρμογές, καθώς δύνανται να βάλουν σε ρίσκο το αποτέλεσμα της αποστολής.

Το Κεφάλαιο 4 τελειώνει με την αξιολόγηση του ευφυούς μηχανισμός λήψης αποφάσεων, η οποία επικεντρώνεται στον επιπλέον χρόνο που προστίθεται στη διαδικασία αναγνώρισης της κρισιμότητας της κατάστασης. Από την εικόνα 4.7, μπορεί να συμπεράνει κανείς ότι ο χρόνος αυτός επηρεάζεται κατά κύριο λόγο από την ανάλυση των δεδομένων από τα μέσα κοινωνικής δικτύωσης. Παρ' όλα αυτά, συνολικά το προτεινόμενο ΚΦΚΣ επιτυγχάνει το στόχο της μείωσης του χρόνου απόκρισης των εφαρμογών έκτακτης ανάγκης και την μείωση του υπολογιστικού φόρτου των συσκευών ΔτΑ.

Κεφάλαιο 5

Το Κεφάλαιο 5 εστιάζει στην ανάπτυξη ενός εναλλακτικού διακοπτικού μηχανισμού μεταφόρτωσης υπολογιστικών διεργασιών στα άκρα του δικτύου, για εφαρμογές της Βιομηχανίας 4.0. Οι εφαρμογές αυτές, απευθύνονται σε ρομπότ τα οποία εκτελούν περίπλοκες διεργασίες, οι οποίες παρουσιάζουν αυστηρές απαιτήσεις τόσο σε χρονική απόκριση όσο και σε ασφάλεια. Σε αυτό το πλαίσιο, η μεταφόρτωση των διεργασιών στα άκρα του δικτύου επιτρέπει στα ρομπότ να ελαφρύνουν τον υπολογιστικό τους φόρτο, αναθέτοντας την εκτέλεση των παραπάνω διεργασιών σε μία ισχυρή υπολογιστική υποδομή σε κοντινή απόσταση. Σε αυτό το κεφάλαιο, λοιπόν, προτείνεται ένας διακοπτικός μηχανισμός μεταφόρτωσης διεργασιών, ενώ σχεδιάζονται ευκαιριακές στρατηγικές μεταφόρτωσης για εφαρμογές που αφορούν στον προγραμματισμό της πορείας και τον εντοπισμό της θέσης των ρομπότ. Η απόφαση για τη μεταφόρτωση λαμβάνεται βάσει της αβεβαιότητας ως προς την τρέχουσα θέση του ρομπότ και την διαθεσιμότητα υπολογιστικών και δικτυακών πόρων στα άκρα του δικτύου, την δεδομένη στιγμή. Το διακοπτικό αυτό σύστημα υλοποιείται και αξιολογείται χρησιμοποιώντας ένα πραγματικό ρομπότ σε μια πραγματική υποδομή στα άκρα του δικτύου· κατά την αξιολόγηση τονίζεται το αντιστάθμισμα ανάμεσα στο χρόνο ολοκλήρωσης των διεργασιών και την επιτυχή έκβαση της αποστολής τους.

Αναλυτικά, το σενάριο που παρουσιάζεται σε αυτό το κεφάλαιο περιγράφει ένα ρομπότ εξοπλισμένο με αισθητήρες και υπολογιστικούς και δικτυακούς πόρους, το οποίο επιχειρεί να φτάσει από ένα αρχικό σε ένα τελικό σημείο, εντός ενός εργοστασιακού χώρου, ανάμεσα σε εμπόδια. Αυτή η λειτουργικότητα είναι βασική για την υλοποίηση εφαρμογών που αφορούν στον εφοδιασμό και την αποθήκευση εμπορευμάτων. Ένα σύνηθες πρόβλημα που αντιμετωπίζεται σε τέτοιου είδους σενάρια είναι η αβεβαιότητα γύρω από την ακριβή «στάση» (θέση και προσανατολισμό) ενός ρομπότ, η οποία αυξάνεται με τον χρόνο κατά την κίνηση, λόγω συσσωρευόμενων ανακριβειών των αισθητήρων, ολισθημάτων των τροχών και αστοχιών στο υλικό. Συνεπώς, η ανάγκη για μια ακριβή, δυναμικά ρυθμιζόμενη τεχνική εντοπισμού θέσης είναι εμφανής. Οι βασικές συνεισφορές αυτού του κεφαλαίου συνοψίζονται, λοιπόν, ως εξής:

1. σχεδίαση και υλοποίηση ενός πρωτότυπου μηχανισμού μεταφόρτωσης υπολογιστικών διεργασιών για ρομποτικές εφαρμογές, ο οποίος χρησιμοποιεί μια υπολογιστική υποδομή στα άκρα του δικτύου ενός βιομηχανικού χώρου, για να βελτιώσει την ακρίβεια του εντοπισμού θέσης και της πορείας του ρομπότ.

2. σχεδίαση και υλοποίηση ενός αλγορίθμου απόφασης μεταφόρτωσης υπολογιστικών διεργασιών, ο οποίος λαμβάνει υπόψιν την δυναμική φύση των κινήσεων του ρομπότ και αντιμετωπίζει την αβεβαιότητά που προκαλούνε στον ακριβή εντοπισμό της θέσης τους στο χρόνο.

3. σχεδίαση και υλοποίηση καινοτόμων αλγορίθμων εντοπισμού θέσης και προσανατολισμού, οι οποίοι επιτυγχάνουν υψηλή ακρίβεια χρησιμοποιώντας το απλούστερο σύστημα καμερών και τον ελάχιστο αριθμό εντοπισμένων διακριτών σημείων στο περιβάλλον.

Η αρχιτεκτονική του προτεινόμενου συστήματος απεικονίζεται στην εικόνα 5.1. Στο εξεταζόμενο σενάριο, οι προϋποθέσεις για μεταφόρτωση υπολογιστικών διεργασιών δημιουργούνται από τις δύο βασικές εφαρμογές που εκτελούν τα ρομπότ: την εκτίμηση της στάσης και τον σχεδιασμό της διαδρομής τους. Για να υποστηριχθεί η μεταφόρτωση αυτών των διεργασιών, εισάγονται τα παρακάτω δομικά στοιχεία της αρχιτεκτονικής, τα οποία εγκαθίστανται στο ρομπότ: α) ο *Ελεγκτής Παρακολούθησης* (ΕΠ), β) ο *Τοπικός Εκτιμητής με βάση την Οδομετρία* (ΤΕΟ), γ) ο *Τοπικός Εκτιμητής με βάση Διακριτά Σημεία στο περιβάλλον* (ΤΕΔΣ), δ) ο *Τοπικός Σχεδιαστής Διαδρομής* (ΤΣΔ) και ε) ο *Μηχανισμός Απόφασης Μεταφόρτωσης Διεργασιών* (ΜΑΜΔ). Στην πλευρά των εξυπηρετητών στα άκρα του δικτύου, εγκαθίστανται τα παρακάτω δομικά

στοιχεία: στ) ο *Απομακρυσμένος Εκτιμητής με βάση Διακριτά Σημεία στο περιβάλλον* (ΑΕΔΣ) και ζ) *ο Απομακρυσμένος Σχεδιαστής Διαδρομής* (ΑΣΔ).

Μία τυπική εκτέλεση ενός σεναρίου της εξεταζόμενης εφαρμογής εκκινεί με τον ΤΣΔ να υπολογίζει μια τροχιά από την αρχική στην τελική θέση. Αυτό δίνει το πρώτο έναυσμα στον ΜΑΜΔ, ο οποίος αναλύει την τροχιά και αποφασίζει αν η μεταφόρτωση του υπολογισμού στα άκρα του δικτύου και τον ΑΣΔ μπορεί να έχει ως αποτέλεσμα μια βελτιωμένη έκδοσή της. Από την μία, η υλοποίηση του ΤΣΔ βασίζεται σε μια απλή, ελαφριά έκδοση του γνωστού αλγορίθμου A^*, με χαμηλή όμως ποιότητα παραγόμενης τροχιάς. Από την άλλη, η υλοποίηση του ΑΣΔ βασίζεται σε μία περίπλοκη υπολογιστικά έκδοση του αλγορίθμου Dijkstra, η οποία επιλέγει το καταλληλότερο ανάμεσα σε υψηλής ποιότητας μονοπάτια, τα οποία έχουν υπολογιστεί πριν την εκτέλεση του σεναρίου. Στη συνέχεια, ο ΕΠ αναλαμβάνει να μετακινήσει το ρομπότ στις ενδιάμεσες θέσεις της τροχιάς. Η κίνηση του ρομπότ μοντελοποιείται με ένα σύστημα συνεχούς χρόνου, το οποίο έχει την αναπαράσταση κατάστασης-χώρου που δίνεται στις εξισώσεις (5.4) - (5.6). Κάθε ενδιάμεση κίνηση αναλύεται σε δύο επιμέρους κινήσεις: α) μια περιστροφική κίνηση, ακολουθούμενη από β) μία μεταφορική. Συγκεκριμένα, η δυναμική κλειστού βρόχου που χρησιμοποιείται, περιγράφεται από τις εξισώσεις (5.7) και (5.8), ενώ στην εικόνα 5.3 δίνεται το αυτόματο που περιγράφει τη λειτουργία του ΕΠ. Κατά τη διάρκεια αυτών των ενδιάμεσων κινήσεων, δίνεται το δεύτερο έναυσμα στον ΜΑΜΔ ώστε να αποφασίσει, ανάλογα με την συνολική συσσωρευμένη αβεβαιότητα στην στάση του ρομπότ και ανάλογα με τη διαθεσιμότητα πόρων στους εξυπηρετητές στα άκρα του δικτύου, αν ο ΕΠ θα κινείται με βάση ανατροφοδότηση από τον ΤΕΟ και τον ΤΕΔΣ ή αν θα μεταφορτωθούν οι υπολογισμοί στα άκρα του δικτύου και τον ΑΕΔΣ. Η λειτουργίας του ΤΕΟ είναι αρκετά ελαφριά όσον αφορά τους υπολογισμούς, καθώς βασίζεται σε μετρήσεις που παρέχουν οι κωδικοποιητές που βρίσκονται στους τροχούς, αλλά συσσωρεύει αβεβαιότητα στον εντοπισμό της θέσης με την πάροδο του χρόνου. Από την άλλη, οι ΤΕΔΣ και ΑΕΔΣ βασίζονται στην αναγνώριση διακριτών σημείων και μάλιστα κυλινδρικών ορόσημων στον περιβάλλοντα χώρο. Η τεχνική αυτή είναι σημαντικά πιο ακριβή υπολογιστικά, καθώς χρησιμοποιούνται τεχνικές αναγνώρισης εικόνων για να εντοπιστούν τα ορόσημα σε πραγματικό χρόνο, τα οποία στη συνέχεια αναλύονται και εξάγονται τα απαραίτητα χαρακτηριστικά τους για να υπολογιστεί η θέση του ρομπότ, μέσω μιας τεχνικής προβολικής γεωμετρίας. Ο βρόχος κίνησης κλείνει με τον έλεγχο του ρομπότ σχετικά με την προσέγγιση της τελικής θέσης. Στην εικόνα 5.2 γίνεται φανερή η απεριοδικότητα της κλήσης των παραπάνω δομικών

στοιχείων.

Όπως αναφέρθηκε και προηγουμένως, οι αποφάσεις του ΜΑΜΔ για μεταφόρτωση διεργασιών παίρνονται με βάση ένα διακοπτικό σύστημα το οποίο αποτελείται από τρεις διακόπτες (εικόνα 5.11). Ο πρώτος διακόπτης αφορά στην επιλογή του τρόπου εκτίμησης της θέσης και του προσανατολισμού του ρομπότ μεταξύ των παρακάτω: α) τους αισθητήρες οδομετρίας και β) το σύστημα κάμερας. Η μετάβαση από την επιλογή α) στην β) γίνεται βάσει ενός ορίου επιτρεπόμενης αβεβαιότητας σχετικά με τη στάση του ρομπότ. Ο δεύτερος διακόπτης ενεργοποιείται μόνο σε περίπτωση που στον πρώτο έχει επιλεχθεί ο β) τρόπος εκτίμησης στάσης και αφορά στην μεταφόρτωση ή όχι της εκτέλεσης του αλγορίθμου, με βάση την διαθεσιμότητα υπολογιστικών και δικτυακών πόρων στα άκρα του δικτύου. Η διαθεσιμότητα των υπολογιστικών πόρων για το ερχόμενο χρονικό διάστημα, υπολογίζεται με τη χρήση ενός φίλτρου Kalman πάνω σε ιστορικά δεδομένα της διαθεσιμότητας και ο εκτιμώμενος χρόνος εκτέλεσης δίνεται από μια γραμμική σχέση. Ο εκτιμώμενος χρόνος μεταφοράς των δεδομένων στα άκρα του δικτύου δίνεται από τις σχέσεις (5.9) - (5.12) και βασίζεται σε ένα λογαριθμικό μοντέλο των απωλειών σε σχέση με την απόσταση, καθώς και στην αναλογία του σήματος προς το θόρυβο. Τέλος, ο τρίτος διακόπτης αφορά στην επιλογή ανάμεσα σε μία χαμηλής (τοπικά υπολογιζόμενης) και σε μία υψηλής (από μεταφόρτωση στα άκρα του δικτύου) ποιότητας τροχιά προς το στόχο και λαμβάνει υπόψιν δυο παραμέτρους: α) ένα βαθμό εγγύτητας της χαμηλής ποιότητας τροχιάς στα διάφορα εμπόδια του χώρου και β) την καμπυλότητά της. Με βάση αυτές, αποφασίζει αν υπάρχει περιθώριο σημαντικής βελτίωσης της μορφής της τροχιάς με τη μεταφόρτωση των υπολογισμών στα άκρα του δικτύου.

Για την πειραματική αξιολόγηση των παραπάνω μηχανισμών χρησιμοποιήθηκε η ρομποτική πλατφόρμα AlphaBot, ενώ στήθηκε ένας χώρος στο Εργαστήριο Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής (NETMODE) της Σχολής Ηλεκτρολόγων Μηχ. & Μηχ. Υπολογιστών του Ε.Μ.Π., ειδικά για να προσομοιάζει το εργοστασιακό περιβάλλον της εφαρμογής (εικόνα 5.14). Για να μπορέσει να αξιολογηθεί σωστά ο μηχανισμός υπολογιστικής μεταφόρτωσης ως προς τον χρόνο εκτέλεσης της εφαρμογής, πρώτα έπρεπε να αξιολογηθεί η αποτελεσματικότητα των εφαρμογών τις οποίες διαχειρίζεται. Σχετικά με την τεχνική εντοπισμού θέσης, στην εικόνα 5.12 φαίνεται ότι παρόλο που η απόκλιση από την πραγματική θέση αυξάνεται όσο αυξάνεται η απόσταση του ρομπότ από τα ορόσημα, η ακρίβεια της δεν πέφτει ποτέ κάτω από 93%. Σχετικά με τον σχεδιασμό της πορείας του ρομπότ, στην εικόνα 5.13 γίνεται φα-

νερή η εγγύτητα μιας υπολογισμένης τροχιάς με την πραγματικά βέλτιστη, η οποία βρίσκεται εντός αποδεκτών πλαισίων για την εφαρμογή. Προχωρώντας στο βασικό κομμάτι της αξιολόγησης, αυτό του διακοπτικού μηχανισμού μεταφόρτωσης υπολογιστικών διεργασιών, γίνεται η σύγκρισή του με τις δύο ακραίες αλλά απλοϊκές προσεγγίσεις εκτέλεσης των υπολογιστικών διεργασιών: α) μόνο τοπική εκτέλεση και β) μόνο υπολογιστική μεταφόρτωση. Στον πίνακα 5.2 δίνονται οι μέσοι χρόνοι εκτέλεσης του ίδιου πειράματος, για τις τρεις διαφορετικές τεχνικές εκτέλεσης. Από τα αποτελέσματα, γίνεται εμφανής η υπεροχή του προτεινόμενου μηχανισμού, τόσο στον μέσο χρόνο εκτέλεσης όσο και στο ποσοστό επιτυχούς έκβασης των αποστολών που κλήθηκε να υλοποιήσει το ρομπότ. Αναλυτικότερα, στη μόνο τοπική εκτέλεση της σχεδίασης διαδρομής, ο χρόνος εκτέλεσης του πειράματος είναι γραμμικώς ανάλογος των βημάτων που αποφασίζει ο $A*$ αλγόριθμος που εκτελείται (εικόνα 5.15), ενώ η αβεβαιότητα στον εντοπισμό της θέσης του ρομπότ δεν σταματά ποτέ να αυξάνεται (αφού το σύστημα καμερών δεν αξιοποιείται ποτέ (εικόνα 5.16). Συνεπώς δεν υπάρχει κάποια εγγύηση σχετικά με την επιτυχή περάτωση του πειράματος. Στην περίπτωση β), της μεταφορτωμένης εκτέλεσης μόνο, από τη μία έχουμε την εγγύηση για την επιτυχή περάτωση του πειράματος, αφού το σύστημα καμερών χρησιμοποιείται για τον εντοπισμό της θέσης του ρομπότ κάθε φορά, έχουμε όμως αυξημένο μέσο χρόνο εκτέλεσης της αποστολής, λόγω ακριβώς αυτής της επιλογής (εικόνα 5.17): η υπολογιστικά πιο ακριβή τεχνική εντοπισμού θέσης χρησιμοποιείται άκριτα, κάθε φορά που χρειάζεται εντοπισμός, ακόμα και όταν η αβεβαιότητα σχετικά με την στάση είναι ελάχιστη και δυνητικά το ρομπότ θα μπορούσε να κινηθεί για κάποιο χρονικό διάστημα και με τα αποτελέσματα του ΤΕΟ. Η σύγκριση αυτή καταλήγει με τα αποτελέσματα του διακοπτικού μηχανισμού μεταφόρτωσης διεργασιών στα άκρα του δικτύου, ο οποίος τα καταφέρνει σημαντικά καλύτερα σε σχέση με τις άλλες δύο προσεγγίσεις, παρέχοντας τόσο εγγυήσεις για την επιτυχή περάτωση του πειράματος, όσο και χαμηλότερους μέσους χρόνους εκτέλεσης (εικόνα 5.18).

Κεφάλαιο 6

Το Κεφάλαιο 6 πραγματεύεται την μεταφόρτωση και τον διαμοιρασμό των υπολογιστικών διεργασιών με χρήση μαρκοβιανών τυχαίων πεδίων. Συγκεκριμένα, στον προτεινόμενο μηχανισμό γίνεται χρήση διακοπτικών συστημάτων για την μοντελοποίηση των υπολογιστικών πόρων στα άκρα του δικτύου και την δυναμική κατανομή τους, βάσει κριτηρίων ενεργειακής κατανάλωσης. Ο προτεινόμενος μηχανισμός αποτελείται από δύο επαναλαμβανόμενα στάδια. Κατά το

πρώτο, γίνεται χρήση μιας τεχνικής βασισμένης σε μαρκοβιανές αλυσίδες, η οποία προβλέπει τις κινήσεις των χρηστών στο χώρο, για τον υπολογισμό του όγκου των διεργασιών που αναμένεται να μεταφορτωθούν στα άκρα του δικτύου. Κατά το δεύτερο, μια καινοτόμα τεχνική βασισμένη σε μαρκοβιανά τυχαία πεδία, αναλαμβάνει τον διαμοιρασμό των υπολογιστικών διεργασιών στους διαθέσιμους πόρους. Οι διεργασίες αυτές δεν δύνανται να εκτελεστούν τοπικά στις συσκευές τον χρηστών, υπό συγκεκριμένους ενεργειακούς περιορισμούς και για συγκεκριμένο επίπεδο ποιότητας υπηρεσιών. Ο προτεινόμενος μηχανισμός επιτυγχάνει βελτιωμένη ενεργειακή κατανάλωση, λαμβάνοντας υπόψιν τις επιπρόσθετες δικτυακές καθυστερήσεις που επιφέρει ο διαμοιρασμός των εργασιών στην υποδομή. Ακόμη, συγκρίνοντας τον προτεινόμενο μηχανισμό με μια γνωστή αντίστοιχη δουλειά στη βιβλιογραφία, επιδεικνύεται η αποτελεσματικότητά του τόσο στη βελτιστοποίηση της κατανάλωσης ενέργειας, όσο και στην ποιότητα των παρεχόμενων υπηρεσιών. Αναλυτικότερα, οι καινοτόμες συνεισφορές αυτής της δουλειάς συνοψίζονται ως εξής:

1. προτείνεται μια προσέγγιση μοντελοποίησης βασισμένη στη θεωρία διακοπτικών συστημάτων, σύμφωνα με την οποία προσδιορίζονται εικονικά προφίλ («προφίλ ΕΜ») του υλικού της υπολογιστικής υποδομής στα άκρα του δικτύου, και η οποία παρέχει εγγυήσεις επιπέδου ποιότητας παροχής υπηρεσιών στις εφαρμογές που φιλοξενούνται στην υποδομή, για διάφορες συνθήκες λειτουργίας. Η μοντελοποίηση αυτή επιτρέπει τη δυναμική επιλογή και κατανομή των κατάλληλων πόρων σε κάθε εφαρμογή (δηλ. την εναλλαγή ανάμεσα στα διάφορα εικονικά προφίλ), βάσει του εκτιμώμενου φόρτου εργασίας. Οι δυνατότητες που προσφέρουν τα διακοπτικά συστήματα, επιτρέπουν το σχεδιασμό ενός προληπτικού μηχανισμού κατανομής πόρων δύο σταδίων, με έμφαση στη μείωση της ενεργειακής κατανάλωσης.

2. κατά το πρώτο στάδιο, επεκτείνονται οι δουλειές που παρουσιάστηκαν στα Κεφάλαια 3 και 4, οι οποίες αντιμετωπίζουν ταυτόχρονα τα προβλήματα της δυναμικής κατανομής πόρων και της μεταφόρτωσης υπολογιστικών διεργασιών στα άκρα του δικτύου, με σκοπό αυτή τη φορά τη μείωση της συνολικής κατανάλωσης ενέργειας της υπολογιστικής υποδομής και την παροχή εγγυήσεων ως προς την ποιότητα των παρεχόμενων υπηρεσιών. Για την απαραίτητη εκτίμηση του πλήθους των αιτήσεων προς μεταφόρτωση που χρειάζεται σε αυτό το επίπεδο, χρησιμοποιείται ένας μηχανισμός πρόβλεψης της κίνησης των χρη-

στών ανάμεσα στις διαφορετικές τοποθεσίες της υποδομής, βασισμένος σε μαρκοβιανές αλυσίδες.

3. κατά το δεύτερο στάδιο, οι παραπάνω μηχανισμοί συνδυάζονται με έναν καινοτόμο μηχανισμό βασισμένο σε μαρκοβιανά τυχαία πεδία, ο οποίος στοχεύει στην ανακατεύθυνση των «πλεοναζόντων» αιτήσεων μεταφόρτωσης μεταξύ των τοποθεσιών της υποδομής. Οι αιτήσεις αυτές δεν δύνανται να εξυπηρετηθούν τοπικά στις τοποθεσίες που δημιουργήθηκαν, υπό συγκεκριμένους περιορισμούς σχετικά με την ενεργειακή κατανάλωση και την ποιότητα των υπηρεσιών. Με αυτόν τον τρόπο, επιτυγχάνεται εξισορρόπηση του φόρτου σε όλη την υποδομή στα άκρα του δικτύου, ενώ ταυτόχρονα βελτιστοποιείται η κατανάλωση ενέργειας. Πρόκειται για την πρώτη φορά που εφαρμόζεται μια τέτοιου είδους προσέγγιση στην βιβλιογραφία. Η ενσωμάτωση του προτεινόμενου αυτού μηχανισμού εξισορρόπησης στους παραπάνω μηχανισμούς, συνθέτει μια ολιστική λύση για την ενεργειακά προσανατολισμένη μεταφόρτωση διεργασιών και κατανομή πόρων που ανταποκρίνεται στις δυναμικές (σε χώρο και χρόνο) απαιτήσεις των αιτήσεων μεταφόρτωσης.

Όπως φαίνεται και στην εικόνα 6.1, η φυσική υποδομή μοντελοποιείται στο εξεταζόμενο σενάριο ως γράφος. Κάθε τελική συσκευή αιτείται τη μεταφόρτωση των διεργασιών της μέσω μιας IEEE 802.11ac σύνδεσης δικτύου, στους εξυπηρετητές της υποδομής που βρίσκονται στην τοποθεσία της, στους οποίους εγκαθίστανται ΕΔ ή ΕΜ για την εξυπηρέτηση των διεργασιών. Με την εφαρμογή εκ των προτέρων του αλγόριθμου Minstrel, παράγεται ένα σύνολο δεδομένων, το οποίο χρησιμοποιείται για τη μετατροπή του εκτιμώμενου αριθμού χρηστών κάθε τοποθεσίας σε εκτιμώμενο φόρτο εργασίας για την τοποθεσία αυτή. Τα προφίλ των ΕΔ/ΕΜ που εγκαθίστανται στους εξυπηρετητές, για κάθε εφαρμογή, περιγράφουν πρακτικά τη σχέση ανάμεσα στην παρεχόμενη υπολογιστική ισχύ και την δυνατότητα εκτέλεσης των αιτήσεων μεταφόρτωσης, υπό συγκεκριμένους χρονικούς και ενεργειακούς περιορισμούς. Το πλεονέκτημα της χρήσης τους είναι διπλό: από τη μία, επιτρέπουν την περιγραφή της δυναμικής συμπεριφοράς των ΕΔ/ΕΜ, υπό ποικίλες συνθήκες λειτουργίας. Από την άλλη, η χρήση τους επιτρέπει στο διακοπτικό σύστημα να ρυθμίζει άμεσα την εγκατεστημένη τοπολογία ΕΔ/ΕΜ στην υποδομή, ανταποκρινόμενο στον δυναμικό φόρτο εργασίας και εγγυώμενο ένα επίπεδο παροχής υπηρεσιών. Για την αναγνώριση των προφίλ των ΕΔ/ΕΜ κάθε εφαρμογής, χρησιμοποιείται μια παρόμοια τεχνική με αυτή που παρουσιάστηκε στο Κεφάλαιο 4 για την αναγνώριση των σημείων ισορροπίας: για κάθε μία από

αυτές, σχεδιάζεται ένα γραμμικό χρονικά αμετάβλητο σύστημα της μορφής (6.1), το οποίο περιγράφει τη σχέση ανάμεσα στο χρόνο απόκρισης της εφαρμογής και την παρεχόμενης υπολογιστικής ισχύ στο ΕΔ/ΕΜ. Η παρεχόμενη υπολογιστική ισχύς αποτελεί και το διακοπτικό κριτήριο για την επιλογή των προφίλ. Στη συνέχεια, οι παράμετροι κάθε συστήματος προσδιορίζονται λύνοντας το πρόγραμμα γραμμικού προγραμματισμού των εξισώσεων (6.2b) - (6.2d). Σημαντικό κομμάτι της μελέτης που παρουσιάζεται σε αυτό το κεφάλαιο, αποτελεί επίσης η μοντελοποίηση της κατανάλωσης ισχύος του συστήματος, η οποία δίνεται από τη σχέση (6.4). Επίσης, για την πρόβλεψη των αιτημάτων μεταφόρτωσης για το επόμενο χρονικό παράθυρο, χρησιμοποιείται, όπως ειπώθηκε, μία παραλλαγή των μαρκοβιανών αλυσίδων ν-κινητικότητας. Σύμφωνα με αυτήν, χρησιμοποιούνται οι δύο προηγούμενες τοποθεσίες που επισκέφθηκε ο κάθε χρήστης και πιθανοτικά εκτιμάται η επόμενή του θέση. Ως προαπαιτούμενο για να λειτουργήσει σωστά αυτή η τεχνική, δημιουργείται ένας πίνακας μεταβάσεων ο οποίος προκύπτει από την καθημερινή παρακολούθηση κινήσεων χρηστών σε αντίστοιχα περιβάλλοντα, η οποία δημιουργείται από σχετικά σύνολα δεδομένων.

Προχωρώντας στην περιγραφή του πρώτου σταδίου της λειτουργίας του μηχανισμού κατανομής πόρων, συναντάμε την γνώριμη τακτική της επιλογής της κατάλληλης τοπολογίας ΕΔ/ΕΜ, η οποία θα υλοποιηθεί στο ερχόμενο χρονικό παράθυρο για την ικανοποίηση του εκτιμώμενου φόρτου εργασίας (εικόνα 6.2). Για την βελτιστοποίηση της επιλογής της τοπολογίας, αρχικά υπολογίζονται όλες οι πιθανές εφικτές τοποθετήσεις προφίλ ΕΔ/ΕΜ για έναν μοναδικό εξυπηρετητή. Ως εφικτή τοποθέτηση, νοείται αυτή στην οποία το σύνολο της υπολογιστικής ισχύος που ζητείται από τα εμπλεκόμενα προφίλ ΕΔ/ΕΜ δεν υπερβαίνει τη συνολική διαθέσιμη υπολογιστική ισχύ του εξυπηρετητή. Η περιγραφή της δίνεται στην εξίσωση (6.5). Στη συνέχεια, λαμβάνοντας υπόψιν τις παραπάνω τοποθετήσεις, λύνεται ένα πρόβλημα βελτιστοποίησης (εξισώσεις (6.6a) - (6.6e)) το οποίο εγκαθιστά τις ΕΔ/ΕΜ, με ενεργειακά προσανατολισμένο τρόπο, σε μία περιοχή της υποδομής, ικανοποιώντας παρόλα αυτά τον συνολικό προβλεπόμενο φόρτο εργασίας για το επόμενο διάστημα.

Στη λειτουργία του παραπάνω σταδίου, αναγνωρίζουμε δύο φαινόμενα τα οποία δημιουργούν τον λεγόμενο πλεονάζων φόρτο εργασίας: το πρώτο αφορά στις περιπτώσεις κατά τις οποίες μία περιοχή της υποδομής δεν δύναται να εξυπηρετήσει όλα τα εκτιμώμενα αιτήματα μεταφόρτωσης διεργασιών, λόγω έλειψης των απαιτούμενων υπολογιστικών πόρων. Το δεύτερο αφορά στις περιπτώσεις στις οποίες κρίνεται ενεργειακά ακριβή η ενεργοποίηση ενός ολόκλη-

ρου εξυπηρετητή για την εξυπηρέτηση ενός μικρού φόρτου εργασίας. Κατά το δεύτερο στάδιο της λειτουργίας του προτεινόμενου μηχανισμού, λοιπόν, λαμβάνει χώρα η εξισορρόπηση του πλεονάζοντος φόρτου εργασίας σε όλη την υποδομή. Για να επιτευχθεί αυτό, χρησιμοποιείται η θεωρία των μαρκοβιανών τυχαίων πεδίων, λόγω του ευέλικτου σχεδιασμού τους, ο οποίος επιτρέπει την κατανεμημένη λήψη αποφάσεων, με αποτελέσματα που προσεγγίζουν τα βέλτιστα σε πολύ χαμηλούς χρόνους σύγκλισης. Οι εξισώσεις (6.9) - (6.12) περιγράφουν φορμαλιστικά τη λειτουργία του παραπάνω μηχανισμού: εν συντομία, οι μονήρεις όροι της εξίσωσης (6.11) εκφράζουν τη στόχευση της κάθε τοποθεσίας της υποδομής να ελαχιστοποιήσει τοπικά την κατανάλωση ενέργειάς της. Επίσης, σε αυτούς τους όρους, συμπεριλαμβάνεται ένα κομμάτι που τείνει να οδηγήσει το σύστημα σε λύσεις που ελαχιστοποιούν παράλληλα και τη συνολική επιπρόσθετη καθυστέρηση που συμβαίνει λόγω των ανακατευθύνσεων των πλεοναζόντων αιτημάτων μεταφόρτωσης. Από την άλλη, οι διπλοί όροι αφορούν στις αλληλεπιδράσεις των γειτονικών τοποθεσιών μεταξύ τους και ιδανικά οδηγούν το σύστημα σε καταστάσεις στις οποίες ανταλλάσσονται πλεονάζοντα αιτήματα μεταφόρτωσης, με σκοπό αυτά να συγκεντρωθούν σε συγκεκριμένες τοποθεσίες, αποφεύγοντας την διασπορά τους και άρα την ενεργοποίηση περίσσειας εξυπηρετητών. Μία επισκόπηση της διαδικασίας αυτής δίνεται στην εικόνα 6.3, ενώ η εικόνα 6.4 δίνει την σχέση ανάμεσα στις χρονικές οντότητες στις οποίες συμβαίνουν οι παραπάνω διαδικασίες. Επιπλέον, στην εικόνα 6.5 απεικονίζεται η αρχική και η τελική κατάσταση μιας υποδομής στην οποία έχει εφαρμοστεί η παραπάνω τεχνική εξισορρόπησης του πλεονάζοντος φόρτου εργασίας, για καλύτερη κατανόηση της λειτουργίας του, ενώ στον Αλγόριθμο 1 δίνεται με ψευδογλώσσα η σειρά των βημάτων που ακολουθούνται.

Για την αξιολόγηση του παραπάνω μηχανισμού, γίνεται προσομοίωση ενός έξυπνου μουσείου, στις διάφορες τοποθεσίες του οποίου βρίσκονται εγκατεστημένα διαδραστικά εκθέματα εξοπλισμένα με εξυπηρετητές με υπολογιστική ισχύ. Οι εφαρμογές που φιλοξενούνται σε αυτήν την υποδομή, είναι βασισμένες σε τεχνικές εικονικής και επαυξημένης πραγματικότητα. Συγκεκριμένα, η αξιολόγηση ξεκινά με την εκτίμηση της επίπτωσης των σφαλμάτων στην εκτίμηση του φόρτου εργασίας. Στην εικόνα 6.6 φαίνεται η επίπτωση αυτή στο συνολικό βαθμό εξυπηρέτησης των αιτημάτων, καθώς και στο μέσο χρόνο εκτέλεσής τους. Και στις δύο αυτές μετρικές, η εφαρμογή της εξισορρόπησης πλεονάζοντος φορτίου, ελαφρύνει τις διαταραχές που προκαλούν τα λάθη της πρόβλεψης. Εν συνεχεία, στην εικόνα 6.7 αποτυπώνεται η αντίδραση του μηχανισμού στις δυναμικές συνθήκες του δικτύου μιας περιοχής και συγκεκριμένα στην κινητικότητα

των χρηστών που αιτούνται μεταφόρτωση διεργασιών. Πιο λεπτομερώς, αποτυπώνεται η αποτελεσματικότητα του βελτιστοποιητή στο να διαλέγει την ενεργειακά βέλτιστη λύση για την εξυπηρέτηση του προβλεπόμενου φόρτου εργασίας. Προχωρώντας στην αξιολόγηση συγκεκριμένα του δεύτερου σταδίου λειτουργίας του μηχανισμού, στην εικόνα 6.8 επιδεικνύεται η ταχύτητα σύγκλισης της εξισορρόπησης πλεονάζοντος φορτίου, για δύο διαφορετικά μεγέθη υποδομών, ενώ η εικόνα 6.9 απεικονίζει τη βελτίωση στο συνολικό επίπεδο της ποιότητας των παρεχόμενων υπηρεσιών που προκαλεί το συγκεκριμένο στάδιο, όταν το εξισορροπούμενο πλεονάζων φορτίο προέρχεται από τοποθεσίες που δεν δύνανται να το ικανοποιήσουν. Στην περίπτωση που αυτό έχει προκύψει από ενεργειακά κριτήρια, στην εικόνα 6.10 φαίνεται η οικονομία που γίνεται στην κατανάλωση ενέργειας. Επιπλέον, στην εικόνα 6.11 φαίνεται η ελαχιστοποίηση της επιπλέον καθυστέρησης που προκαλείται στο συνολικό χρόνο εκτέλεσης λόγω των ανακατευθύνσεων των αιτημάτων. Τέλος, ο προτεινόμενος μηχανισμός συγκρίνεται με μια παρόμοια, γνωστή λύση του ίδιου προβλήματος στη βιβλιογραφία [99]. Για το σκοπό αυτό, διεξάγονται δύο σετ πειραμάτων: στο πρώτο στοχεύεται η ελαχιστοποίηση της κατανάλωσης ενέργειας και για τους δύο μηχανισμούς, το οποίο έχει ως αποτέλεσμα τις διπλάσιες περίπου παραβάσεις στο συμφωνηθέν επίπεδο παρεχόμενων υπηρεσιών (χρόνος απόκρισης της εφαρμογής, εικόνα 6.12), για τον συγκρινόμενο μηχανισμό. Στο δεύτερο σετ πειραμάτων, στοχεύεται η ικανοποίηση του συμφωνηθέντος επιπέδου παρεχόμενων υπηρεσιών, πράγμα το οποίο έχει ως αποτέλεσμα την κατά 65% μεγαλύτερη ενεργειακή κατανάλωση του συγκρινόμενου μηχανισμού (εικόνα 6.13).

Κεφάλαιο 7

Κλείνοντας, το Κεφάλαιο 7, συνοψίζει το σύνολο της διατριβής, επιχειρηματολογώντας για τη σπουδαιότητα των εξεταζόμενων ερευνητικών προβλημάτων και των μεθόδων που επιλέχτηκαν για την επίλυση τους, ενώ παράλληλα παραθέτει συγκεντρωμένα τα κύρια συμπεράσματα που ανέκυψαν. Ακόμη, προτείνονται ανοιχτά ερευνητικά θέματα για μελλοντική εργασία που είτε θα μπορούσαν να αποτελούν τη συνέχεια αυτής της ερευνητικής προσπάθειας, είτε μπορούν να εκμεταλλευτούν την αποκτημένη γνώση προκειμένου να την εφαρμόσουν σε νέους τομείς και δραστηριότητες.

Preface

Structure

The thesis will be structured as follows.

In **Chapter 1**, a general introduction on the topics that will concern this thesis will be made, the environment which motivated this research and which will be considered for the development of the proposed methods will be set and the contributions that were made in this PhD thesis will be exhibited.

In **Chapter 2**, some basic mathematical background that is deemed necessary to understand the methods used in the approaches to tackle the proposed problems will be set. Additional information will be provided in the main part of the thesis whenever it is needed.

The following chapters will be the main chapters of the thesis. Each chapter will present a more specific problem that was considered important and was solved. First a general setting specific to the problem and the related work on the topic will be provided. Then the proposed solution will be discussed by presenting the system model, the mathematical formulation of the problem and the architecture of our solution. Finally the proposed framework will be evaluated.

In **Chapter 3**, an adaptive resource allocation and admission control mechanism for computational offloading in Network Edge settings will be discussed. This framework utilises concepts from Control Theory to offer mobile users a way of executing their tasks at the Edge of the network, which guarantees a level of QoS while optimizing the resource utilisation at the infrastructure.

In **Chapter 4**, the extension, integration and evaluation of the previous mechanism in a Cyber-Physical System for early fire detection will be presented. This novel computational

offloading mechanism is enhanced with a Cloud-based decision making service, which accurately predicts incident severity and notifies the responsible authorities. Its role is to alleviate the computational stress of the IoT nodes involved in natural disaster confrontation.

Chapter 5 focuses on a switching system for computational offloading of robotic applications in Edge computing ecosystems. During this work both local (robot-based) and remote (edge-based) application controllers were designed and implemented, followed by a scheduling mechanism. These controllers are treated as switches and compose a system, which is adaptive and can operate under different scenarios and usages.

Chapter 6 deals with an energy-aware framework that addresses jointly the full task offloading and resource allocation problems in a multi-site setting. In this setting, a holistic energy-aware resource optimization approach is proposed, based on the design of the VM flavors complemented with an innovative distributed load balancing technique based on Markov Random Fields, with the penultimate goal to minimize the total energy consumption without sacrificing the QoS in terms of latency.

Finally in **Chapter 7**, the problems addressed in this thesis will be summarized, giving the reader a comprehensive overview of the most important conclusions drawn from this study. Then, recommendations for future work will be provided, which can be carried out as an extension of the work presented in the thesis.

Chapter 1

Introduction

In today's information technology age, data is the main commodity; possessing more data typically generates more value in data-driven businesses [2]. The amount of digital data generated surpassed 1 zettabyte in 2010, according to the International Data Corporation (IDC) [3]. Additionally, 2.5 exabytes of new data is generated daily since 2012 [4]. This proliferation of data, alongside the significant growth in the processing and networking capabilities of mobile devices over the past decades, has allowed for the development of mobile applications for a wide range of human daily activities, including healthcare and wellness, education, commerce and social media. Cisco estimates that there will be around 50 billion connected devices by the end of this year [5]. These connected devices constitute the *Internet of Things (IoT)* and potentially generate a massive amount of data. In current implementations of IoT applications, most data that needs storage, analysis, and decision making is sent to the data centers in the Cloud [6]. However, with this astronomical amount of data, the current mobile network architectures will have trouble managing the resulting momentum and magnitude.

Cloud Computing (CC) has taken the world by storm as it facilitates users and devices with several opportunities by providing a wide range of services and virtually unlimited available resources (e.g., network, servers, storage) in a multi-tenant model [7, 8]. These resources are available over a network and are accessed through standard mechanisms. The cloud computing paradigm provides a variety of deployment and service models, from public clouds (organizations that provide cloud computing services to any customer) to private

clouds (organizations that deploy their own private cloud computing platform) and from Infrastructure as a Service models (IaaS, where fundamental computing resources are offered as a capability) to Software as a Service models (SaaS, where applications are offered as a capability), among other things. This large pool of resources and services has enabled the emergence of several new applications, such as virtual reality [9], smart grids [10, 11, 12], and smart environments [13]. The benefits of cloud computing – minimal management effort, convenience, rapid elasticity, pay per use, ubiquity – have given birth to a multi-billion industry that is growing worldwide.

Generally, public cloud vendors have built large data centers in various parts of the world. These large-scale, commodity-computer data centers have enough computing resources to serve a very large number of users. However, the CC-induced euphoria transforms into a problem as the speed and volume of the transferred data increases; this centralization of resources implies a large average separation between end user devices and their clouds, which in turn increases the average network latency and jitter [14]. As a result, moving the big data from devices to the cloud might not be efficient, or might be even infeasible in some cases, due to bandwidth constraints. On the other hand, as time-sensitive and location-aware applications emerge (such as patient monitoring, real-time manufacturing, self-driving cars, flocks of drones, augmented reality or cognitive assistance), the distant cloud will not be able to satisfy the ultra-low latency requirements of these applications, provide location-aware services, or scale to the magnitude of the data that these applications produce, as cloud services are not able to directly access local contextual information, such as precise user location, local network conditions, or even information about users' mobility behavior [15]. The problem becomes clearer and more intense as several smart devices and objects are getting involved in human's life, as in the case of smart cities [16] or Internet of Things [17]. The current cloud computing paradigm [18] is unable to meet the requirements of low latency, location awareness, and mobility support [19]. Moreover, in some applications, sending the data to the cloud may not be a feasible solution due to privacy concerns.

Several approaches have been proposed over the last few years by the research community to satisfy the quintessential need for a computing paradigm that takes place closer to the connected devices, in order to address the issues of high-bandwidth, geographically-dispersed, ultra-low latency, and privacy-sensitive applications. The emerged novel paradigms include

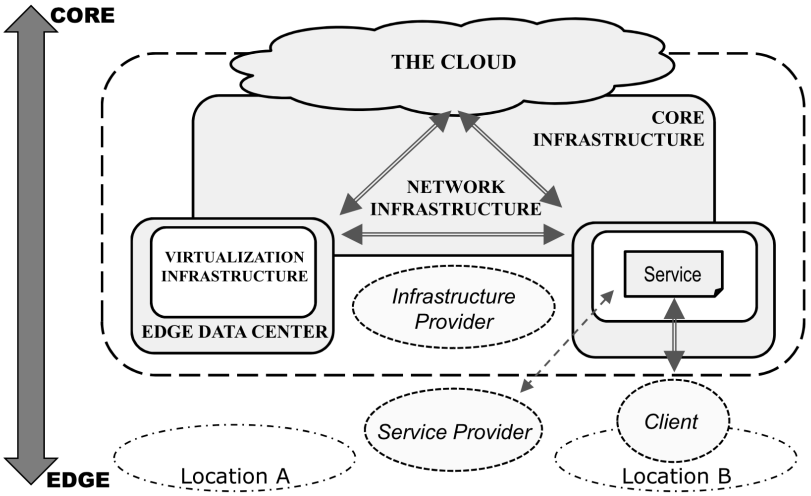


Figure 1.1: Functional structure of edge computing paradigms.

Mobile Cloud Computing [20], *Fog Computing* [21], and *Mobile Edge/Multi-access Computing* [22] among others [23, 24]. The common denominator in these edge paradigms is the deployment of cloud computing-like capabilities at the edge of the network. Most edge paradigms follow the structure depicted in Figure 1.1, [25]. Edge data centers, which are owned and deployed by infrastructure providers, implement a multi-tenant virtualization infrastructure [26]. Any customer – from third-party service providers to end users and the infrastructure providers themselves – can make use of these data centers' services. In addition, while edge data centers can act autonomously and cooperate with one another, they are not disconnected from the traditional cloud. It is therefore possible to create a hierarchical multi-tiered architecture, interconnected by a network infrastructure. Besides, the potential existence of an underlying infrastructure, or core infrastructure (e.g., mobile core networks, centralized cloud services) that provide various support mechanisms, such as management platforms and user registration services has to be considered. Finally, one trust domain (i.e., edge infrastructure that is owned by an infrastructure provider) can cooperate with other trust domains, creating an open ecosystem where multitude of customers can be served.

There are various differences among edge paradigms, such as the focus on mobile network operators as infrastructure providers in mobile edge computing, the existence of user-owned edge data centers (i.e., personal cloudlets) in mobile cloud computing, and the use of differ-

ent underlying protocols and interfaces, among others. Nonetheless, there remain numerous similarities. Still, little of the research in these fields takes into consideration these similarities. Most architectures, protocols, services, and mechanisms are designed with only one edge paradigm in mind, and they do not consider the state of the art of other edge paradigms.

1.1 Overview of Edge Computing Paradigms

1.1.1 Mobile Cloud Computing

Mobile Cloud Computing (MCC) is the emerging service delivery paradigm that integrates cloud computing into the mobile environment. MCC mainly focuses on the notion of "mobile delegation": due to the limited resources available to mobile devices, the storage of bulk data and the execution of computationally intensive tasks should be delegated to remote entities. In this context, MCC provides on-demand, low-latency and secure access to a resourceful group of servers in the spatial vicinity of mobile users. This comes again complementary to the CC paradigm which suffers from latency issues due to the connection to remote servers in the cloud through public Internet. In the original MCC concept, introduced in 2009, only centralized cloud computing platforms were considered as the most viable solution to implement the remote execution of tasks [27]. Later, other researchers expanded the scope of MCC. In this new vision, tasks could also be delegated to devices located at the edge of the network [28]. At present, both visions of MCC coexist [29]. This thesis will mostly focus on the latter.

Initially, MCC sought to provide novel solutions to services such as mobile learning, mobile healthcare, searching services, and others [30]. Nowadays, many of these services can be implemented in a centralized cloud (e.g., voice-based search) or in the mobile devices themselves (e.g., text-to-speech engines). Nevertheless, the concept of MCC is still relevant, as its potential has not been fully exploited. There are certain applications, such as augmented reality and augmented interface applications, where the existence of an execution platform located at the vicinity of the mobile devices can provide several benefits such as lower latency and access to context information. Moreover, as mobile devices are equipped with functional units such as sensors and high resolution cameras, it is possible to develop novel

crowdsourcing and collective sensing applications that make use of the location information [20].

1.1.2 Fog Computing

Fog Computing has been proposed to address the CC-related issues and to quench the need for a computing paradigm closer to the connected devices [31]. Fog computing bridges the gap between the cloud and IoT devices by enabling computing, storage, networking, and data management on the network nodes within the close vicinity of IoT devices. Therefore, computation, storage, networking, decision-making, and data management occur along the path between IoT devices and the cloud, as data moves to the cloud from the IoT devices. Thus, fog computing does not compete with cloud computing, but rather complements it: the fog architecture facilitates the creation of a hierarchical infrastructure, where the analysis of local information is performed at the 'ground' and the coordination and global analytics are performed at the 'cloud'. Here, cloud services are deployed mostly at the edge of the network, but they can also be deployed in other locations, such as IP/multiprotocol label switching (MPLS) backbones. In fact, the fog network infrastructure is heterogeneous, where high-speed links and wireless access technologies will coexist [32].

The initial definition of fog computing was later expanded and revised by various researchers ([21, 33]). Although this extended definition is debatable, it reveals all the advances that the fog might introduce. Under this new definition, fog computing does not become a mere extension of cloud computing, but a paradigm of its own. The elements that implement the cloud services, the fog nodes, can now range from resource-poor devices (e.g., end devices, local servers) to more powerful cloud servers (e.g., Internet routers, 5G base stations). Also, all these elements can be able to interact and cooperate with each other in a distributed fashion. This generates a three-tier architecture (end devices - fog nodes - central servers) where centralized cloud servers coexist with fog nodes but are not essential for the execution of fog services [34].

Originally, fog computing was defined as a platform that enabled the creation of new applications and services in the context of IoT. Examples of such services include hierarchical Big Data analytics systems and smart infrastructure management systems (e.g., wind farms, traffic lights) [21, 31]. Yet, at present, there are several studies that examined how

this paradigm could be used to implement other types of services: low-latency augmented interfaces for constrained (mobile) devices (e.g., brain–computer interfaces using wireless electroencephalogram headsets [35], augmented reality and real-time video analytics [36]), cyber-physical systems [37], novel content delivery and caching approaches under the context of fog computing [38], and various vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) services such as shared parking systems [39].

1.1.3 Mobile/Multi-access Edge Computing

Mobile Edge Computing (MEC) was introduced to bring the cloud services and resources closer to the user proximity by leveraging the available resources in the edge networks [40]. To meet the above mentioned requirements of IoT applications, the mobile operators are planning to integrate the computing, networking, and storage resources with the base station in the form of a MEC platform. Similarly to Fog and Cloudlet [41, 42], MEC is not replacing but complimenting the cloud computing model. The delay sensitive part of an application can be executed on a MEC server, whereas the delay-tolerant, compute-intensive part of the application can be executed on a cloud server. MEC aims to enable billions of connected mobile devices to execute the real-time compute-intensive applications directly at the network edge. The distinguishing features of MEC are its closeness to end-users, mobility support, and dense geographical deployment of the MEC servers. In the MEC World Congress 2016, MEC ISG has renamed Mobile Edge Computing as Multi-access Edge Computing, in order to reflect the growing interests from non-cellular operators [43].

The benefits of deploying cloud services at the edge of mobile networks, like 5G, include low latency, high bandwidth, and access to radio network information and location awareness. Thanks to this, it will be possible to optimize existing mobile infrastructure services, or even implement novel ones. An example is the Mobile Edge Scheduler [44], which minimizes the mean delay of general traffic flows in the LTE downlink. Moreover, the deployment of services will not be limited to mobile network operators, but it will also be opened to third party service providers as well. Some of the expected applications include augmented reality, intelligent video acceleration, connected cars, and Internet of Things gateways, amongst others [45].

In order to implement the MEC environment, it is necessary to deploy virtualization

servers (i.e., MEC servers) at multiple locations at the edge of the mobile network. Some deployment locations considered by the MEC ISG are LTE/5G base stations (eNodeB), 3G Radio Network Controllers (RNC), or multi-Radio Access Technology (3G/LTE/WLAN) cell aggregation sites—which can be located indoors or outdoors. Besides, the MEC ISG has suggested that this virtualization infrastructure should host not only MEC services, but also other related services such as Network Function Virtualization (NFV) and Software Defined Networking (SDN) [45]. Such a deployment would reduce the deployment costs and provide a common management and orchestration infrastructure for virtualized services.

Concluding, even if there are several structural differences between the above mentioned paradigms, it does not mean that they should exist in a vacuum, ignoring the advances in other related fields. Due to the similarities between the paradigms, it is safe to assume that there will be mechanisms and platforms that can provide a generic solution to a shared problem. Such solutions can then be adapted to other edge paradigms. Hence, following this notion, for the rest of the thesis, the servers used in the framework are called *Edge Servers* independently of the architecture; edge servers receive and execute compute-intensive tasks of mobile applications.

1.2 Computational Offloading

One of the most active areas of research in the field of edge computing is the delegation of tasks to external services [29]. In this context, the computation migration is endeavoured as a significant software-level solution that mitigates resource constraints of mobile devices by migrating applications to available stationary (or not) computers [46, 47, 48]. There are various solutions that allow applications to migrate part of their code from the mobile devices to cloud-based computing resources located at the edge. Applications are usually implemented using frameworks like .NET and JVM, which makes the code migration process easier. Some research results allow mobile devices to migrate only part of their code, thus it is necessary to statically or dynamically identify the code that needs to be offloaded. Other researchers take a more extreme stance: an entire execution environment (i.e., clone), representative of the mobile device, is created. Then, part of the mobile application (including memory image, CPU state, and others) is loaded into the clone. Finally, some approaches

make use of mobile agents' infrastructures, where the mobile device creates a mobile agent that will acquire/process information on its behalf, which is roughly the solution that is mainly adopted in this thesis. There are even approaches, such as the concept of Aqua Computing, that mix the notion of mobile agents and clones [49].

In any case, similarly to other research areas such as communication networks and distributed systems, optimization techniques are also widely used in application execution frameworks of edge computing. The execution frameworks consider diverse optimization objective functions as follows: saving processing power, efficient resource and bandwidth utilization and minimizing energy consumption [50]. In short, the frameworks are designed to optimize the execution cost. The overall aim of all such approaches is to enable the compute-intensive mobile applications on resource-constrained mobile devices. The problem of determining what task, where and whether it should be offloaded in order to save energy and/or meet time constraints is known as *Computational Offloading* [51].

1.3 Applications

The synergistic combination of recent technological developments improves our ways of living in various societal domains. In the smart computing context, sensor networks, edge computing, IoT and big data analytics are properly orchestrated to provide assisting applications for human daily activities such as education, health and transportation. Apart from these classic IoT applications, this thesis will explore how natural disaster management and Industry 4.0 applications can benefit from edge computing concepts such as computational offloading. These complex applications are either time- and mission- critical applications with stringent requirements.

1.3.1 Natural Disasters

Dealing with natural disasters, like wildfires, is an interesting field for the development of edge computing -enabled IoT applications, since the early and precise detection of a forest fire is the most important step for in-time firefighting. Various IoT node arrangements, for example Wireless Sensor Networks (WSN) and, more recently, Unmanned Aerial Vehicle (UAV) clusters equipped with remote sensing capabilities, have enabled the detection of

wildfires [52, 53] and the automatic notification of the responsible public authorities. The ability to perceive their environment and react to its changes, perform basic data processing and exchange information, alongside the excellent scalability and the low capital and operational expenditures, make IoT networks a reliable solution for autonomous monitoring of large forest areas. However, as these networks typically comprise of small battery powered devices performing multiple tasks, limited energy resources and the scarcity of computation capabilities for real-time processing are the most important disadvantages towards their wide adoption in the fire-fighting domain. Dealing with these shortcomings, edge computing can provide the required rich computation resources near the IoT portable devices [54]. In the specific case of wildfires, a cluster of powerful servers is placed at the edge of the network and enables the offloading and processing of the IoT nodes' computation-intensive tasks, e.g., image recognition, in order to reduce their energy consumption and achieve the application's strict time constraints.

Over the last years, forest fire fighting technology, enabling smart computing features on Unmanned Aerial Vehicles (UAVs), has shown significant progress, making the deployment of small-sized UAVs for forest fire detection a natural and increasingly realistic option [55]. UAVs are relatively inexpensive, easily manoeuvrable, can cover various terrain types under different weather settings, both at day and night and, most importantly, their missions can be achieved autonomously with minimal or even with zero human involvement. UAVs equipped with remote sensing and data communication facilities demonstrate excellent potential for monitoring, detecting and fighting forest fires. On the other hand, the potential advantages of UAVs, depend on many factors, such as aircraft type, sensor types, mission objectives, and the current regulatory requirements in the application domain the UAV operates in [42]. Specifically for the fire-fighting domain, UAV technology still faces various obstacles that need to be confronted in order to be applied in fully operational environments. One of these barriers is, as mentioned, the scarcity of available energy and processing resources on UAV platforms. Aerial monitoring of large fields and forests during dry spells to reduce the risk of wildfires requires increased energy resources for UAVs to prolong their mission's endurance, which is very difficult to secure.

The proliferation of social media usage by large population proportions, alongside recent advances in data sensing, collection, storage and analysis, supports the realization of

the participatory data gathering paradigm, also known as social sensing. Data that are produced on social media services can act as an additional source of information valuable in various application domains, including the scope of disaster prevention, detection, control and assessment. For example, recent research efforts have evaluated the use of social media in relation to extreme weather incidents [56], earthquake events detection [57] and to estimate diseases spread such as influenza [58] and malaria [59]. These initial findings suggest that social media may provide a promising approach for detecting and mapping environmental hazards and climate-related impacts, however a robust methodology has yet to be defined and validated. In particular, within the scope of wild-fire detection, social media users, who happen to be in the proximity of a fire incident, can provide valuable information and testimony about the current situation and help to timely and accurately detect a wildfire [60]. The potential of this approach has been also confirmed by European Commission's JRC initiative named Digital Earth Nervous System [61].

1.3.2 Industry 4.0

As mentioned earlier, computation offloading in current and next-generation networks is becoming increasingly important due to the proliferation of IoT real world applications [62]. As discussed, these applications introduce a vast number of low-capability, low-energy devices to the networking ecosystem, which regularly need to perform computationally intensive and/or energy-hungry tasks. However, when latency and energy consumption minimization are required, the limited resources of the IoT devices prove inadequate [63]. In Industry 4.0 and especially in collaborative robotics, where humans and robots work together in dynamic environments, computationally heavy algorithms enable IoT devices in sensing and actuating [64]. Consequently, a large amount of information has to be processed and complex algorithms need to be executed in real-time.

The increasing availability of networking in the Edge and Cloud supports new approaches, where processing is performed remotely, with access to extensive computing and memory resources. In this direction, Edge Computing alongside Fog Computing [65] constitutes a particularly prominent way of dealing with the aforementioned shortcomings of IoT devices. It offers an attractive alternative providing low-latency and high energy efficient operation, while maximizing system performance. This paradigm is currently more relevant

than ever, especially in the context of the much-anticipated Industry 4.0 revolution [66] and Industrial IoT (IIoT), where the concept of Fog Robotics (FR) is introduced. FR can be defined as the architecture that distributes computing, storage and networking functions at the Edge/Cloud continuum in a federated manner [67], i.e., where robots and automation systems rely on data or code from a network to support their operation.

1.3.3 Smart Environments

Smart environments (SEs) (i.e., the open and dynamic systems typically extending over an area and including a large number of interacting devices with a heterogeneous nature) and the IoT paradigm, share the common vision of enabling a pervasive presence in the environment of a variety of smart things that are able to interact with each other, with the aim of creating new applications and reaching common goals. In this context, the research and development challenges in creating a smart system are numerous [68]. Augmented Reality (AR), i.e., the combination of a representation of the real-world environment and computer-generated input from sensors, is a typical enabler of smart environments. An example of an AR application in a smart environment is AR enhancing the experience of a visitor to a smart museum; consider a visitor to a museum, art gallery, city monument, music or sports event, pointing their mobile device towards a particular point of interest with the application related to their visit activated (i.e., the museum application). The camera captures the point of interest and the application displays additional information related to what the visitor is viewing [69].

Offloading an Augmented Reality service on an platform located at the Network Edge instead of the Cloud is advantageous since supplementary information pertaining to a point of interest is highly localised and is often irrelevant beyond the particular point of interest. In this setting, the processing of user location or camera view can be performed on the Edge infrastructure rather than on a more centralized server. In AR applications there may be a need to update information quickly, depending on how the users move, and the context in which the augmented reality service is used (e.g., in an art gallery, exhibits are positioned only a few metres apart and each piece is supplemented with additional text on the artist, the interpretation of the artwork, etc.) In other words, AR data requires low latency and a high rate of data processing in order to provide the correct information to the user's device,

depending on their location and orientation, making the Edge the perfect candidate for AR tasks offloading.

1.4 Challenges and Motivation

As thoroughly discussed, the decentralization and proximity of the service infrastructure to the edge brings various benefits (e.g., low latency, scalability), but it also brings new issues that must be carefully considered. Despite the several advantages, realizing the vision of an edge computing framework is a challenging task because of the procedural and security concerns involved (e.g., discovering edge nodes, partitioning and offloading tasks, using edge nodes publicly and securely). Apparently, there is a need to investigate the key requirements and potential opportunities for enabling the vision of edge computing. Thus, it can be concluded that in the emerging edge computing paradigm, several problems arise and innovative research is needed to address them. Technology has evolved in such a way that we now have multiple tools, which enable us to come up with nifty solutions. For that reason, this opportunity allows for solving modern and interesting problems concerning the following topics:

- **Optimizing Resource Utilization:** This constitutes the main motivation behind this thesis, as resource allocation is the main challenge faced in edge settings where, naturally, resources are not considered abundant. Despite the earlier and ongoing work on various aspects of edge computing, the problem of how to efficiently deploy these new edge applications within an edge cloud has not been systematically studied. Simply duplicating the successful cloud computing design will not work for the edge applications. As the offloaded IoT workloads are required to be processed by different types of applications, usually running on Virtual Machines (VMs) hosted on the edge servers, the decision on the number of instances and the computing resources to assign to each of them becomes challenging and has a direct impact on the response time achieved. Furthermore, as many devices may be requiring the edge servers capabilities at the same time, efficient and dynamic assignment of their workloads to the hosted applications is required. Thus, resource management arises as an important concern in the emerging computing paradigms [70]. Also, available resources on the edge

servers are limited compared to the clouds. Therefore, the optimization of resource utilization is necessary for gaining the better performance with limited resources [50, 71]. Generally, the optimization of resource utilization is a multi-objective function that becomes a challenging task to solve because of the diverse nature of applications, varying user demands, and varying users' requirements. Most of the proposed studies in the literature utilize queuing theory to model mobile devices and edge servers, along with an optimization method for finding the optimal offloading policy. The most commonly used criteria are the energy consumption of the mobile device and the request throughput. However, there is a major shortcoming in these approaches that can lead to the deterioration of the system's performance: the static modeling of the servers for fluctuating workload, which can lead to over provisioning or under provisioning [72].

- **Dealing with Resource Heterogeneity:** Another challenge is imposed due to the highly heterogeneous nature of edge clouds. Unlike central clouds, edge clouds are often comprised of heterogeneous computation nodes with widely diverse communication, computation, and storage capabilities. The edge nodes can include micro servers, IoT gateways, routers, mobile devices, etc. A major challenge in edge computing settings is to decide where to place the services and how many resources to allocate to it, while taking into account the heterogeneity of edge nodes, services, and users. For example, the response time of edge services can vary significantly depending on the network interface and hardware configuration of edge nodes. Thus, the need to model the available resources in a way that smooths the inherent heterogeneity out, as well as to design appropriate resource allocation controllers that make use of these models, is evident.

- **Uncompromising Quality-of-Service (QoS) and Experience (QoE):** Quality delivered by the edge nodes can be captured by QoS and quality delivered to the user by QoE [73]. One principle that will need to be adopted in edge computing is to avoid overloading nodes with computationally intensive workloads [22, 74]. The challenge here is to ensure that the nodes achieve high throughput and are reliable when delivering for their intended workloads if they accommodate additional workloads

from a data center or from mobile devices. Regardless of whether an edge node is exploited, the user of an edge device or a data center expects a minimum level of service. For example, when a base station is overloaded, it may affect the service provided to the mobile devices that are connected to the base station. A thorough knowledge of the peak hours of usage of edge nodes is required, so that tasks can be partitioned and scheduled in a flexible manner. The role of a management framework will be desirable but raises issues related to monitoring, scheduling and re-scheduling at the infrastructure, platform and application levels. In most of the above studies there is no formal guarantee of satisfying the physical constraints, i.e., CPU and memory sharing, or meeting the QoS specifications, such as the average response time. Since all edge computing architectures use a small cluster of servers, a fallacious resource allocation mechanism can hamper the offloading performance.

- **Optimizing Computational Offloading:** The requirements for deploying application workloads on edge computing frameworks, have to be well understood. Deployment strategies - where to place a workload, connection policies - when to use the edge nodes and heterogeneity - how to deal with different types of nodes, need to be taken into account for deploying applications on the edge. The execution of compute-intensive components of IoT applications in edge computing infrastructures, involves the complex application partitioning at different granularity levels and component migration to the edge server node. Furthermore, suitable as it may seem, solely utilising remote computational resources might not be enough; a number of unwanted phenomena potentially take place in the transmission and processing of the information, such as network latency, variable QoS and downtime. For these reasons, for example, autonomous mobile agents (e.g., robots, unmanned vehicles) often have some capacity for local processing when targeting low-latency responses and during periods where network access is unavailable or unreliable. Consequently, a major challenge, from a control design, estimation, and network optimization point of view is to combine local and remote resources in an efficient way.

- **Dealing with User Mobility:** User mobility is one of the most critical components when it comes to making the computational offloading decision. End devices can either

be considered as static or mobile for the time window which spans between initiating and finishing the offloading of their tasks. In the latter case, mobility adds another level of splitting decision, as it needs to be decided at which edge site should the tasks be offloaded while the user is on the move. Even though mobility is considered a challenge, it can generate a number of opportunities for the task offloading. First of all, it can initiate a load balancing technique to allow the system to provide the necessary services in distributed Edge site scenarios. Secondly, complementing mobility with appropriate prediction solutions can enhance the system's capacity, by finding the potential next associated base station of the user. This can be even more beneficial in a dense scenario, where the system can analyse the active users and their mobility patterns and allocate the resources in an online manner to existing and newly requested services. Moreover, mobility can benefit from handover mechanisms that can enable service migrations between base stations and their edge servers. However, as the requirements of zero millisecond handover are studied by the 5G community, mobility with prediction mechanisms is starting to gain attention, in order to predict beforehand where the tasks should be offloaded. This behavior can be decisive for the overall performance in dense Edge deployments with multiple mobile users, and a major challenge in designing computational offloading and resource allocation strategies [75].

1.5 Contributions

This thesis tries to tackle some of the aforementioned problems that arise on the edge computing architecture. Focus is mainly placed on the decision making process, where the devices need to make some choices regarding the resource allocation and exploitation inside their environment. The contributions on the above topics can be summarized in the following:

1. **Modeling Abstractions to Capture System Dynamics:** Performance modeling of cloud applications is an open research problem and it is coupled with resource allocation. For monolithic applications, static models, queue models and state-space models are used to describe the dynamic relation between control variables (computing and network resources) and application's outputs (latency or throughput). Additionally,

various prediction approaches have been used to estimate the magnitude of the incoming tasks or the user's behaviour, which facilitate the design of successful resource allocation strategies. Available resource models are usually single-input single-output. Energy or response time are typically the model's output, while computing resources (e.g., CPU, memory), incoming requests, and network bandwidth are the control variables. In most of the current studies, the relation between input and output is fixed and empirically derived. For example, the processing time of a request is proportional to its file size and inversely proportional to the service rate measured in CPU cycles or millions of instructions per second. Although this assumption is reasonable, the actual processing time depends on many time-varying parameters, which are not easily measured. Furthermore, in combination with a static resource allocation, the offloading decision performs adequately only for specific operating conditions, being unable to guarantee stability under fluctuating workload and a heterogeneous IoT communication infrastructure.

Contrary to current approaches that provide empirical static models, the aim is to develop formal, realistic and dynamic traffic and resource models applicable to emulate the generated traffic from various applications. To this purpose, a performance modeling approach is proposed, based on System Theory, that has the capacity to include several performance metrics (i.e., state variables) and resources as control parameters (input variables) and describe their relationship under various operating conditions and QoS guarantees. The resulting models are called *flavors*. The computation of these flavors is based on switched systems from the System Theory. The advantage of the flavor design is two-fold. First, a flavor is actually a feasible operating point and facilitates the dynamic resource allocation process. Secondly, the combination of these flavors with feedback controllers can enable fine-grained vertical and horizontal scaling approaches for time- or mission- critical applications. Furthermore, a flavor-oriented modeling and infrastructure design, paves the way for tackling resource heterogeneity, as will be shown in the following sections.

2. **A Dynamic Resource Allocation and Admission Control Mechanism:** Currently, and regardless of the adopted edge computing architecture, the computation

offloading decision is coupled with the resource allocation in the edge servers. Consequently, together with the offloading decision, a dynamic resource allocation and admission control mechanism is proposed that leverages the preceding modeling abstractions, which is called *Vertical and Horizontal Scaling Mechanism*. This mechanism is responsible for the (de)activation of the edge servers, the placement of the application instances and the distribution of offloaded requests among them (*Horizontal Scaling*), alongside the admission control and resource allocation for each application instance within the active servers (*Vertical Scaling*). A workload profile estimator and the dynamic modeling of the resources and overall status of the network/servers provide the foundation upon which the resource allocation algorithm works. Specifically, the objective is to develop a joint communication and computing virtualization paradigm that is updated and adapted dynamically. To this purpose, the problem of simultaneously (i) allocating computing and communication resources, (ii) modifying network topology/ protocol and (iii) structuring the edge computing data centres (such as VMs distribution) is considered. This approach gives emphasis on the dynamic behaviour of the resource allocation. Also, the obtained dynamic models are utilized and system-theoretic analysis methods and stabilizing controllers are developed. These algorithms are designed to be practicable, and guarantee feasibility and performance specifications, such as robustness to rapid changes in the workload, resource availability, and unwanted network phenomena. This research addresses the so far untouched challenge, of designing controllers that address a mixture of these unwanted phenomena by changing the provisioning of the resources to the control algorithm, if this is deemed necessary. This type of controllers are made possible by the merging of two sets of hybrid models, namely a) the performance model, having as internal variables performance metrics of the infrastructure and as inputs the resource distribution and utilization, and b) the process model, having, for example, variables related to position, orientation, and velocity of mobile agents.

3. **A Cyber-Physical Social System for Assistance in Emergency Situations:** In the past, unexpected wildfires would mainly have an impact on the wild fauna and flora, however, the interaction of humans with wildfires has significantly changed during last decades; the expansion of urban areas near forests, called Urban-Wildland

Interface, put human population and their assets at higher risk of wildfires than ever before [76]. Thus, the strategy of the fire management and the preparedness towards the continuously extended severe fire danger season must be updated and enhanced. That is why all the aforementioned technologies are combined in a Cyber-Physical Social System (CPSS), which integrates computing, networking and human resources. In detail, a network consisting of static or mobile IoT nodes (i.e., UAVs) which monitors forest areas for detecting fires at their initial stage is integrated. These nodes are equipped with embedded camera modules to enable computer vision-based fire detection. This operation is assisted by a Scalable edge coMputing framewOrK for early firE detection (SMOKE) which hosts classification services and overtakes the computational workload of processing field snapshots captured from the UAVs. Utilizing and extending the previously mentioned contributions, SMOKE comprises a dynamic resource scaling mechanism for IoT-enabled, time- and mission- critical applications, meant to be deployed at a cluster of servers at the network edge, in the nodes' proximity, assisting the offloading of computationally intensive, energy hungry tasks. Finally, a cloud-based decision-making service is leveraged that combines the classification results of the previous level, users' actions on the social media and other services, such as weather information services, in order to accurately infer the fire incident severity and notify the responsible authorities.

4. **A Switching Computational Offloading Mechanism for Robotic Applications:** In modern manufacturing, the current trend is to remove robots from confined spaces and allow free movement in the factory floor, enhancing cooperation and collaboration with humans, thus, increasing productivity and efficiency. Open challenges in this area are concerned with developing adaptive multi-robot/machine control, capturing, modelling, predicting and anticipating human-robot interactions and designing distributed control and path planning algorithms that deliver flexible and safe working environments. Thus, it seems only natural to utilize distributed computing and storage resources to solve position estimation and path planning problems. These are the two computational offloading opportunities that arise. In the studied case, to accommodate the offloading of these computationally demanding tasks, a small-scale network infrastructure is set up, connecting the Robotic Agent wirelessly to an Access Point

(WLAN), located within the Agent's network range, which in turn connects via a wired connection to an edge server (LAN) and a cloud server (Internet). Both these servers offer the same services, specifically designed and developed for this thesis, virtualized as Docker [77] containers, listening for potential offloading requests coming from the Agent, at all times, in a client-server way. These two remotely located servers differ from each other in the usual "edge computing vs. cloud computing" way, meaning that the former exists in the same LAN network with the Robotic Agent (low network latency) and is more adaptive when it comes to resource allocation management, but has limited computational resources overall, while the former exists on a Cloud Service Provider in the Internet (higher network latency) but its computational resources are considered abundant. This design results in an emerging decision-making problem on *where it is more profitable for the Agent to offload its workload*. Thus, in order to orchestrate this decision-making problem, three components are developed: a local one implemented on the Robot and two remote ones, one running on the edge server and one on the cloud server. Eventually, the required computations are either executed locally on the Robot or offloaded to the Edge and/or Cloud, based the following identified use-case requirements: Time and Energy Efficiency, Optimality in Trajectory Planning and Robust Navigation. Consequently, a major challenge, from a control synthesis, estimation as well as network optimization point of view, is to combine local and remote methods in an efficient, safe and near-optimal way.

5. **A Framework for Distributed Energy-aware Resource Allocation at the Edge for Smart Environments:** Apart from satisfying the users, an efficient resource allocation technique is required for infrastructure providers as well. On the provider side, the primary goal is the minimization of the energy consumption of the data center, which is mainly affected by the number of active servers and the amount of their allocated resources. Thus, task offloading and resource allocation are coupled and must be jointly addressed. To this end, a synergistic and distributed approach between the end-devices and the edge infrastructure is necessary to accommodate the dynamic demand of the applications. The main challenge of such an approach is to estimate the amount of the offloaded tasks and make appropriate decisions on where the offloaded tasks should be executed. Taking into consideration the network conditions,

the complexity of this resource allocation problem increases exponentially. Dynamic physical channel conditions and user mobility require a proactive and dynamic resource allocation technique to select the necessary computational and networking resources at the Edge in an adaptive manner. In this direction, appropriate resource allocation strategies enhanced with mobility prediction techniques are investigated, to further ameliorate the delay and energy savings of both end-devices and edge infrastructure. In more detail, a task offloading setting with applications of different characteristics and requirements is considered and an optimal resource allocation framework leveraging the amalgamation of the edge resources is proposed. To balance the tradeoff between retaining low total energy consumption, low end-to-end delay and load balancing at the Edge, a Markov Random Field -based mechanism is introduced for the distribution of the excess offloaded workload. The proposed approach investigates a realistic scenario, including different categories of mobile applications, edge devices with different computational capabilities and dynamic wireless conditions modeled by the dynamic behavior and mobility of the users. The framework is complemented with a prediction mechanism that facilitates the orchestration of the physical resources. The efficiency of the proposed scheme is evaluated via modeling and simulation and is shown to outperform a well-known task offloading solution.

6. **Evaluation of proposed frameworks with numerical results through real experimentation and simulations:** For all the above contributions mentioned, either real experimentation or simulations are performed in order to capture the effectiveness and the efficiency of the proposed frameworks.

In the following, the proposed methods are presented, alongside a discussion on the main contributions of this thesis. Then, each Chapter focuses on one of the aforementioned settings, presenting related work on the field and introducing the developed solutions together with some indicative evaluation that justifies the benefits of their adoption.

Chapter 2

Background

In this chapter, the essential mathematical background necessary to understand the methods that are used in the following work will be presented. Any additional information required will be provided in the main part of the thesis, whenever it is needed.

2.1 Basic Definitions of Modeling and Control Theory

2.1.1 Linear Time Invariant State Space Models

A state space model of a system is the mathematical description of the relationship between *the cause and the effect* or *the inputs and the outputs* of the system [78]. In this thesis, discrete time state space models are solely utilised. The general form of a discrete time state space model is

$$x(k+1) = f(x(k), u(k)), f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \tag{2.1}$$

where $x(k) \in \mathbb{R}^n, u(k) \in \mathbb{R}^m$ are the state and the input vector respectively and $k \in \mathbb{N}$. The most widely used state space models are the linear time invariant (LTI) state space models, where the function $f(x(k), u(k))$ of (2.1) is linearly dependent on $x(k)$ and $u(k)$,

$$x(k+1) = Ax(k) + Bu(k), \tag{2.2}$$

$$y(k) = Cx(k) + Du(k). \tag{2.3}$$

Here, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are constant, time invariant matrices that describe the system's dynamics and $y(k) \in \mathbb{R}$ is the system's output vector.

2.1.2 Stability

In modern control theory, the notion of stability is strongly connected with the dynamic behavior of a system. Many different kinds of stability are defined, i.e. input-output stability or stability of an equilibrium point. In general, a stable system means that the state variables of the system are driven to a specific equilibrium point or inside a desired area and they remain there despite any insisting or momentary disturbances. In this chapter, focus is put on the stability of an equilibrium point (x_{eq}, ueq). An equilibrium point[1] satisfies the equality

$$x_{eq} = Ax_{eq} + Bu_{eq}. \tag{2.4}$$

The most general type of equilibrium point stability is Lyapunov stability, which guarantees that the system trajectories will remain close to x_{eq}, if they start from a neighborhood in the equilibrium point's vicinity [79].

Furthermore, in the following sections, asymptotic stability is adopted, together with the set-theoretic notions of stability analysis and control design problem, which identify and characterize subsets of the state space, containing the desired equilibrium state with special properties: positively invariant sets are introduced here. In the forthcoming paragraphs, without loss of generality, all the necessary definitions are given assuming that $x_{eq} = 0$.

Definition 1. *A sphere B_s with radius $s > 0$ and the origin as its center, is denoted as*

$$B_s = \{x \in \mathbb{R}^n : \|x\| \leq s\}, \tag{2.5}$$

[1]It should be noted here that, for the rest of this thesis, the terms "equilibrium point", "operating point" and "flavor" will be used interchangeably.

where $\|\cdot\|$ *is any possible norm of vector* x.

Definition 2. *Assuming a discrete time system of the following form*

$$x(k+1) = f(x(k)), \tag{2.6}$$

then the equilibrium point $x_{eq} = 0$ *is locally Lyapunov stable, if and only if* $\forall \epsilon > 0, \exists \delta = \delta(\epsilon) > 0$. *Then,*

$$x_0 \in B_{\delta(\epsilon)} \Rightarrow x(t; x_0) \in B_\epsilon, \forall t \geq 0. \tag{2.7}$$

Definition 3. *The zero equilibrium point of* (2.6) *is contractive in a region* $D \subset \mathbb{R}^n$ *if* $\forall x_0 \in D$ *and*

$$\lim_{t \to \infty} x(t; x_0) = 0. \tag{2.8}$$

The region D is called Domain of Attraction (DoA) of the equilibrium point.

Definition 4. *The zero equilibrium point of* (2.6) *is asymptotically stable if and only if it is Lyapunov stable and contractive,*

$$\lim_{t \to \infty} x(t; x_0) = 0, x_0 \in B_{\delta(\epsilon)} \subseteq D, \tag{2.9}$$

where D is DoA.

In the following definitions, the essential results of the Lyapunov theory (second or direct Lyapunov method), which connect the stability property with a specific type of functions, are presented.

Definition 5. *Assuming a continuous function* $V(x), V : D \to \mathbb{R}$, *where D contains the origin. Then V(x) is positive (semi)definite in D if*

$$V(x) > (\geq) 0, \forall x \in D \backslash \{0\}, \tag{2.10}$$

$$V(0) = 0. \tag{2.11}$$

Definition 6. *Function $V(x)$ is negative (semi)definite if $-V(x)$ is positive (semi)definite.*

Definition 7. *For discrete time systems* (2.6)*, the total difference of function $V(x), V : \mathbb{R}^n \to \mathbb{R}$, respectively to system* (2.6) *is*

$$\Delta V(x)_{(2.6)} = V(f(x)) - V(x). \tag{2.12}$$

Then, the Lyapunov theorem for discrete time system is formulated as,

Theorem 1. *([78, 79]) Assuming a positive definite function $V(x), V : D \to \mathbb{R}$, then*

- *If the total difference $\Delta V(x)_{(2.6)}$ of* (2.12) *is negative semidefinite $\forall x \in D$, the system is locally Lyapunov stable.*
- *If the total difference $\Delta V(x)_{(2.6)}$ of* (2.12) *is negative definite $\forall x \in D$, the system is locally asymptotically stable.*

The function $V(x)$, which satisfies the above theorem, is called Lyapunov Function (LF). From the previous analysis, the stability problem is equal to finding a positive definite function which is non-increasing or decreasing along the trajectories of the system (2.6). Finding an LF, allows for defining sets with special properties respectively to the equilibrium point. For example, if there exists an LF $V(x)$ that guarantees the stability or asymptotic stability in a region D and the sets $R(V;\gamma) = \{x \in \mathcal{R}^n : V(x) \leq \gamma\} \subseteq D$ are close and contain the zero equilibrium point, then these sets consist an estimation of the DoA. The essential definitions follow.

Definition 8. *[80] The set $S \subseteq \mathbb{R}^n$ is positively invariant to* (2.6)*, if and only if $\forall x_0 \in S$ the system trajectory remains inside S for all future moments $x(t;x_0) \in S, \forall t \geq 0$.*

Definition 9. *[80] For a convex compact set $S \subseteq \mathbb{R}^n$ which contains the origin, the following function is called the Minkowski function.*

$$\Psi_s(x) = inf\{\lambda \in \mathbb{R} : \lambda \geq 0, x \in \lambda S\}. \tag{2.13}$$

Then, the set S is ϵ-contractive to (2.6)*, $0 \leq \epsilon < 1$, if $\forall x_0 \in S$*

$$\Psi_s(x(t;x_0)) \leq \epsilon^t \Psi_s(x_0), t \geq t_0. \tag{2.14}$$

Now the connection between the Lyapunov function, the positive invariant set, the DoA and the ϵ-contractive set can be shown.

Remark 1. *Assume the system* (2.6) *and a candidate LF V(x); if the set* $R(V;\gamma) = \{x \in \mathbb{R}^n : V(x) \leq \gamma\}$ *is convex, compact and contains the origin, then the following conclusions apply:*

- *If* $V(x)$ *is an LF that guarantees Lyapunov stability, then the set* $R(V;\gamma)$ *is positive invariant to system* (2.6).
- *If* $V(x)$ *is an LF that guarantees asymptotic stability, then the set* $R(V;\gamma) \subseteq D$ *is positive invariant and DoA to system* (2.6).
- *If* $V(x)$ *is an LF that guarantees Lyapunov stability and it applies that*

$$\Delta V(x)_{(2.6)} \leq (\epsilon - 1)V(x(t)), 0 \leq \epsilon < 1, \tag{2.15}$$

then the set $R(V;\gamma) \subseteq D$ *is* ϵ*-contractive to system* (2.6).

The most important benefits from the Lyapunov theory is the characterization of the stability of the equilibrium point and the possibility of defining sets with interesting properties. For example, if an LF can be found, which ensures the asymptotic stability of the equilibrium point of a constrained system $x(t) \in S_x \subset \mathbb{R}^n$, then any trajectory that begins from a point inside $R(V;\gamma) \subseteq S_x$ will be driven to the equilibrium point, without violating the system constraints.

2.2 Basic Definitions of Markovian Random Fields

Let us continue with some necessary definitions regarding the Markovian Random Field (MRF) structure, used in this thesis to model computational offloading problems and acquire solutions in the area of resource allocation and workload balancing at the network edge. As in [81], a finite set S, $|S| = n$, is assumed, with elements $s \in S$ referred to as sites or nodes. Every site s is associated with a random variable X_s that expresses its state. Let the phase space Λ be the set of possible states of each $s \in S$, i.e., X_s takes a value $x_s \in \Lambda$. The

collection $X = \{X_s,\ \forall s \in S\}$ of random variables with values in Λ consists of a Random Field (RF) on S with phases in Λ. A configuration $\omega = \{x_s : x_s \in \Lambda,\ \forall s \in S\}$ corresponds to one of all possible states of the system and the product space Λ^n, $\omega \in \Lambda^n$ denotes the configuration space. A neighborhood system on S is defined as a family $\mathcal{N} = \{\mathcal{N}_s\}_{s \in S}$ of subsets $\mathcal{N}_s \subset S$, such that for every $s \in S$, $s \neq \mathcal{N}_s$ and $r \in \mathcal{N}_s$ if and only if $s \in \mathcal{N}_r$. $\mathcal{N}_s$ is called the neighborhood of site (node) s. The RF X is called a Markov Random Field (MRF) with respect to $\mathcal{N}$, if for every site $s \in S$,

$$\mathbb{P}(X_s = x_s \mid X_r = x_r, r \neq s) = \mathbb{P}(X_s = x_s \mid X_r = x_r, r \in \mathcal{N}_s). \quad (2.16)$$

A RF X is called a Gibbs Random Field (GRF) if it satisfies:

$$\mathbb{P}(X = \omega) = \frac{1}{Z} e^{-\frac{U(\omega)}{T}}, \quad (2.17)$$

where $Z := \sum_{\omega \in \Lambda^n} e^{-\frac{U(\omega)}{T}}$ is the partition function and T is the temperature of the system. $U(\omega)$ is called the potential function and represents a quantitative metric of the current state of the configuration ω. The potential function is not unique. A very useful class of potential functions, which will be employed in the studied approach, is one in which $U(\omega)$ is decomposed into a sum of clique (maximally connected subgraph) potential functions, as $U(\omega) = \sum_{c \in \mathcal{C}} V_c(\omega)$, where $\mathcal{C}$ is the set of the cliques formed by the sites and each clique potential V_c depends only on the states of the cliques formed in the underlying system graph. The Hammersley-Clifford theorem [82] asserts that a GRF with distribution $\mathbb{P}(X = \omega) = \frac{1}{Z} e^{-\frac{U(\omega)}{T}}$ and potential function expressed in terms of clique potentials leads to an MRF with conditional probabilities $\mathbb{P}(X_s = x_s \mid X_r = x_r, r \neq s) = \mathbb{P}(X_s = x_s \mid X_r = x_r, r \in \mathcal{N}_s)$ and vice-versa. This property will be also employed for the design of the potential function and the implementation of distributed decision-making via Gibbs sampling.

More explanations on modeling and control theory, as well as MRF-related concepts will also be provided in this thesis whenever needed.

Chapter 3

Adaptive Resource Allocation for Computation Offloading: A Control-Theoretic Approach

3.1 General Setting

Contrary to the cloud computing environments where dynamic modeling and control mechanisms have been extensively adopted [83], [84], little attention has been given to the optimal use of the edge servers. In this chapter, a two-level cooperative resource allocation mechanism for a single cluster of edge servers hosting a group of applications is presented, that allows IoT devices/mobile users, within the coverage area, to offload application-specific tasks. It should be noted, however, that user mobility within the cluster's proximity has not been considered in this setting; this problem is covered in the following chapters. The proposed mechanism can on-demand allocate the edge servers' resources to different applications using virtual machines (VMs). At the lower level, the dynamic operation of VMs is captured by linear dynamics. Local controller components are responsible for regulating allocated CPU shares and accepted offloading requests, according to a varying, however bounded in a given interval, incoming workload. This comprises the *Vertical Scaling* part of the mechanism. At the upper level, a horizontal scaling process is responsible for activating

the essential number of edge servers and placing the appropriate VMs into them. This comprises the *Horizontal Scaling* part. In particular, the incoming requests are distributed to the activated servers in order to serve the total demand. This process is orchestrated while taking the local controllers into consideration, making this mechanism cooperative.

3.2 Related Work

One of the initial and influential works on MCC [85] proposed a dynamic VM synthesis of a cloudlet infrastructure. The position paper [86] presented the potentials of MCC ecosystems; wearable devices, Internet of Things (IoT) applications, automotive and industrial environments alongside tactile Internet can leverage from the mobile-cloud convergence. The extended survey [87] presented a definition of MCC, the vision and the challenges, a taxonomy of heterogeneity in MCC and open issues. The survey paper [88] analyzed the challenges of Fog Computing in terms of architecture, service and security and classified the existing studies according to these criteria.

Surveys of existing computation offloading approaches are provided in [51], [89]. The authors of [90] addressed computation offloading as an admission control problem in MCC hotspots with a cloudlet, using semi-Markov decision process modeling and linear programming. The resource constraints were considered when obtaining the optimal solution. A similar dynamic offloading algorithm was proposed in [91]. Therein, the admission control problem on cloudlets was modeled and solved as a Markov decision process, aiming to minimize the computation and offloading costs. Also the mobility of the users was taken into account. Khojasteh et al. [92] presented two flexible resource allocation algorithms for computation offloading. The resource allocation process and VM provisioning were modeled by a Markovian multiserver queuing system with priority levels and a multidimensional Markov system, based on a Birth-Death queuing system with finite population, respectively. In [93] three resource allocation schemes were proposed for computation offloading. Several stochastic sub-models captured the operation of a physical machine, under the policy of each scheme. The Markov Reward Model was applied to obtain the output of the sub-models and the decision criteria consist of the request rejection probability and mean response delay. The authors in [94] proposed a hierarchical MCC architecture where users could of-

fload their tasks, modeled by queuing models, either to local cloudlets or the remote public cloud. Computation offloading was modeled as a generalized Nash equilibrium problem and a distributed algorithm computed an equilibrium strategy for each user.

Many studies focus on energy-aware offloading. In the work of Xia et al. [1] a two-tier MCC environment was adopted; mobile devices, cloudlets and the remote cloud were described by static models and an algorithm that optimized the minimum residual energy ratio was developed. Jalali et al. [95] proposed static, flow-based and time-based energy consumption models. They presented a detailed energy consumption comparison between cloud computing and fog computing architectures while taking the network equipment into account. Their numerical results demonstrated how offloading leads to energy saving for IoT applications. In the work of Kiani and Ansari [96], a task scheduling scheme for code partitioning in a hierarchical cloudlet environment was proposed for two different use cases. The one finds the optimal task scheduling for already defined radio parameters, while the other optimizes both the task scheduling and the transmission power of the mobile devices.

Finally, there are some interesting studies that examine other problems in the area of computation offloading. In the work of Barbera et al. [97], the feasibility of computation offloading and data backups in real-life scenarios was examined. Since communication is not free, the authors focused on bandwidth and power consumption of WiFi, 2G and 3G technologies. A real testbed with smartphones and Amazon EC2 nodes was used for thorough analysis. Xu et al. [98] focused on cloudlet placement in order to minimize the average cloudlet access delay between mobile devices and cloudlets. A heuristic scalable algorithm was proposed for the special case of homogeneous cloudlets. Jia et al. [99] used the placement of Xu et al. [98] and proposed a load balancing algorithm to utilize fairly a group of cloudlets. Queuing models were adopted for cloudlets and a scalable algorithm computed the optimal request redirection such that the maximum of the average response times at cloudlets was minimized. Trying a different approach, Liu et al. [100] proposed a game-based distributed MEC scheme where the users competed for the cloudlet's finite computation resources via a pricing approach, modeled as a Stackelberg game. The algorithms examined there were implemented in a distributed manner. In [101], the authors proposed two algorithms for maintaining the low end-to-end delay between the mobile devices and the cloudlets when the users move around the network topology. The key idea

lies in optimally deploying the mobile device's corresponding VMs in the available cloudlets, while adapting to the user's movement. Dealing with the opposite data flow, i.e., offloading from the cloud to the edge, the authors of [102] presented a collaborative content caching system at the network edge. They developed a model to instruct the edge node to trigger on demand caching when popular content has been identified. SDN techniques were leveraged to manage and distribute the content among the access nodes in a coordinated manner.

A shortcoming of studies [1], [95], [97], [98] and [99] is that the modeling of the edge servers captures accurately a single operating point and not the whole operating range. On the other hand, for the various Markov process approaches [90], [91], [92] and [93], the execution time of each request derived from a fixed service rate. However, these assumptions on the static operating range and service rate apply only when the operating conditions are close to that point. Furthermore, in the preceding studies, a systematic analysis on satisfaction of the QoS specifications and the constraints is missing. This thesis aims to address the aforementioned shortcomings. Thus, state-space modeling is used to capture the dynamic behavior of the edge server under different operating conditions. The local controller computes the system's feasible operating (equilibrium) points while considering different competitive criteria and guarantees the stability and confinement in a specific area around them. The Horizontal Scaler takes these operating points into account and determines the appropriate placement that serves the incoming varying workload.

3.3 Contribution & Outline

Specifically, to overcome the aforementioned drawbacks, a two-level, adaptive and cooperative resource allocation mechanism for a single cluster of edge servers hosting a group of applications is proposed, which allows mobile users to offload application-specific tasks, while offering control-theoretic based QoS guarantees. As mentioned in Chapter 1, computation offloading mitigates the energy consumption of resource-constrained mobile devices by relocating the execution of the compute-intensive tasks to a group of edge servers that are placed in the *Mobile Users'* spatial vicinity. This placement enables low-latency access to the servers, contrary to accessing the remote cloud through the public Internet, which is unpredictable when it comes to response times. Figure 3.1 depicts the MCC computation

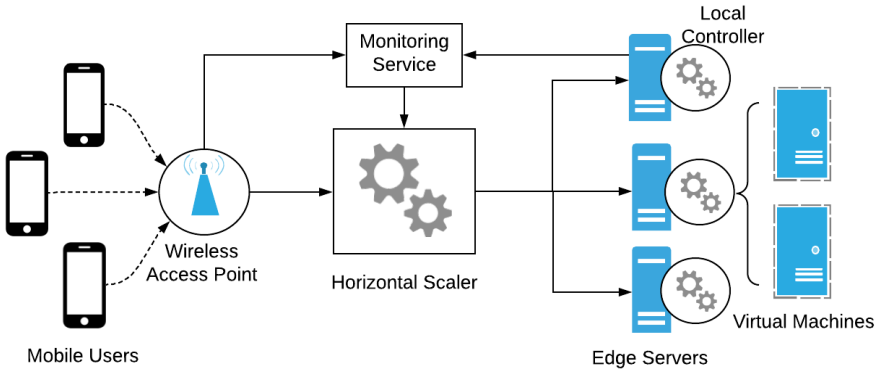


Figure 3.1: MCC Computation Offloading Architecture

offloading architecture studied in this chapter. Specifically, the offloaded traffic, generated from the mobile devices, is directed to the *Horizontal Scaler* through the local *Wireless Access Point* (with WiFi, 3G/4G or LTE support). There lies the upper level control process of the proposed mechanism; this component selects an appropriate VM placement to be implemented to each edge server directly connected to it and consequently distributes the incoming workload accordingly. This decision defines the number of active servers alongside the number and the operating state of the VMs to be placed in them. This upper level process is performed in an online and proactive manner, through the use of an internal prediction mechanism, the *Workload Predictor*, described in more details in Subsection 3.5.2, able to estimate the incoming offloading requests in the following time window. The essential input for this estimation process is provided by the *Monitoring Service* component, which is responsible for collecting data regarding both the network traffic (e.g., offloading requests issued, end-to-end response times) and the servers' resources utilization (e.g., CPU usage) at each given time. As mentioned earlier, this is the horizontal scaling part of the proposed mechanism and the theory behind it is described thoroughly in Subsection 3.5.2.

At the lower level, each edge server is equipped with a *Local Controller*, able to create, run, scale and stop application-specific VMs, thus assisting the realization of the selected VM placement for the given time window. Additionally, the lower level control process is implemented in this component, as it moderately scales the VMs vertically based on data coming from the Monitoring Service. In this way, it ensures that the VMs remain within the selected operating state, thus guaranteeing minimum and stable application response times. The theoretical design behind this control process is described in more detail in Subsection

3.5.1.

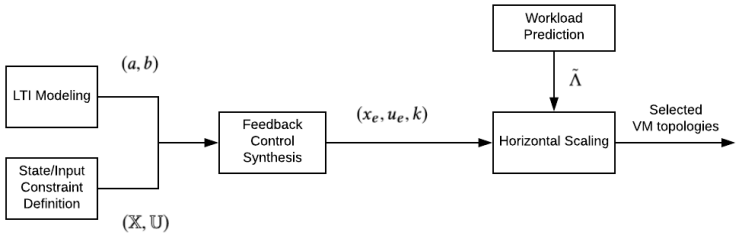


Figure 3.2: MCC Computation Offloading Mechanism Workflow

Figure 3.2 illustrates the workflow of the proposed MCC computation offloading mechanism. In the proposed approach, the operation of the VMs is modeled by a group of *Linear Time Invariant* (LTI) systems that are subject to additive exogenous disturbances. The parameters of the LTI systems are identified by experimental data. At first, for each LTI system, a feasible equilibrium of the nominal disturbance-free model of the VM, (x_e, u_e), is computed. Each equilibrium point corresponds to an operating state (flavor) of the VM without assuming disturbances. For example, an operating point might correspond to 3 requests per second, utilizing 20% of CPU allocation and resulting in an average response time of 3sec. For each equilibrium point a linear state feedback controller, meaning the control gain k, is designed by taking the disturbed system into account, within the Local Controller component. Specifically, by regulating the assigned CPU allocation and the number of admitted requests, a controller is designed such that the closed loop system **(i)** is stable, **(ii)** satisfies the constraints and the QoS specifications at all times and for any initial condition, starting from within the constraint set, and **(iii)** behaves optimally in steady state. Since the proposed resource management mechanism offers guaranteed response time to mobiles users, *the offloading decision breaks down to a simple comparison between the estimated execution time on the mobile device and the guaranteed response time provided by the edge servers.*

At the upper level, for each application the Horizontal Scaler receives an estimation of the forthcoming requests $\tilde{\Lambda}$, made by the Workload Predictor component, and the set of the feasible operating points (x_e, u_e), computed by the feedback controller, as input. Then, based on this information, it decides the minimum number of active edge servers and the VM placement to be implemented in them, towards the satisfaction of the total demand for each

application. This cooperation of the two control levels ensures that the selected operating point of each VM from the Horizontal Scaler will be realized by the feedback controller of the VM. Briefly, the basic contributions and differences of the proposed approach and framework are summarized as follows:

1. this modeling approach can accurately capture the dynamic behavior of the application-specific VM under different operating conditions.
2. a multitude of feasible operating points can be calculated, considering different performance and utilization costs, which allows to design different control strategies for different pairs of workload and applications.
3. formal guarantees regarding resource allocation and QoS specifications are provided.
4. the minimum number of edge servers is activated to satisfy the overall workload of all applications, based on the set of the feasible operating points of the lower level.

3.4 System Modeling

In this setting, a number of $N \geq 1$ different applications are hosted in isolated VMs in an edge server. For each application and for a range of incoming request rates, a scalar discrete LTI system is identified. To this purpose, let

$$\lambda(t) \in [\Lambda_m, \Lambda_M],$$

denote the incoming Request Rate (RR) per second at time instant t, which is varying in an interval $[\Lambda_m, \Lambda_M]$. The range of incoming RR is divided in L subintervals of the form $[\Lambda_{i,m}, \Lambda_{i,M}] \subset [\Lambda_m, \Lambda_M],\ i = 1, \ldots, L$. Consequently, *for each application and each request rate subinterval a linear system with additive disturbances is identified* of the form

$$x(t+1) = \max\{ax(t) + bu(t) + w(t), 0\}. \tag{3.1}$$

In the above equation, $x(t)$ is the average response time, $u(t) = [u_1(t) \quad u_2(t)]^\top \in \mathbb{R}^2$ is the input vector and $w(t) \in [w_m, w_M]$, $w_m < 0 < w_M$, is an unknown, however bounded, signal, which accounts for the disturbances induced by the communication between the edge

server and the mobile users and the modeling error of the identified model. In this case, focus is placed on CPU intensive applications, thus, VM memory is statically assigned and not included in the linear systems. However, the Horizontal Scaler takes memory constraints in the VM placement problem into account, as it described in Subsection 3.5.2. To simplify notation, the exposition is not indicated to be done for system (i), since this is arbitrary chosen.

The input $u_1 \in [u_{1m}, u_{1M}]$ corresponds to the allocated CPU share of the VM whereas $u_2 \in [\Lambda_{i,m}, \Lambda_{i,M}]$ is RR the controller admits. The parameter $a \geq 0$ is a known scalar while $b \in \mathbb{R}^{1\times 2}$ is a row vector. Both a, b can be estimated using the Recursive Least Square (RLS) algorithm [103]. The maximum operator in eq. (3.1) ensures that there are no negative average response times in the model.

The state and input variables x and u are inherently constrained due to the finite resources and the control specifications. In particular,

$$\mathbb{X} := \{x : \ 0 \leq x(t) \leq X_M\}, \tag{3.2}$$

$$\mathbb{U} := \{u : \ u_m \leq u(t) \leq u_M\}, \tag{3.3}$$

for all $t \geq 0$, where $u_m = [u_{1m} \quad \Lambda_{i,m}]^\top$, $u_M = [u_{1M} \quad \Lambda_{i,M}]^\top$.

By having a set of models (3.1) corresponding to different RR intervals, a better level of accuracy is provided than a single LTI model for the whole range of RR. Additionally, the number of models scales linearly with respect to L, since the co-hosted applications are considered decoupled and depend only on the number of subintervals of the RR.

At the lower level, the local controllers focus on the joint resource allocation and admission control of edge servers in order to perform guaranteed response time under the varying workload of the consolidated applications. For the lower control level (vertical scaling), the main goals are summarized as follows.

P1. Consider system (3.1) subject to the constraints (3.2), (3.3) that corresponds to a certain incoming request rate. Given a desired response time, find a *feasible* operating region for the system (3.1), which is optimal with respect to a well defined cost.

P2. For each operating condition calculated in **P1**, compute an admissible control strategy, which steers the closed loop system to it and respects the constraints at all times.

The following table sums up the main symbols used in the next subsections, alongside their description:

Table 3.1: Notation Table

t	Time instant
λ	Incoming Request Rate per second (RR)
x	Average Response Time (sec.)
u	Input $u(t) = [u_1(t) \quad u_2(t)]^\top$; u_1 is the allocated VM CPU; u_2 is the RR admitted at the VM
(x_e, u_e)	Feasible Operating Point
r_e	Statically allocated VM memory
w	Communication disturbances
a, b	LTI system parameters
k	Control gain
$\mathbb{X}, \mathbb{U}$	State and Input Constraints
$\mathcal{S}$	Invariant set
$\mathcal{P}$	The Set of Feasible VM placements
E	Number of edge servers
$\tilde{\Lambda}$	Predicted Incoming RR

3.5 System Architecture

3.5.1 Vertical Scaling

In this section, a discussion on how this approach tackles the problems **P1, P2** *simultaneously* is made. In specific, an optimization problem is formulated whose solution retrieves both the operating condition and the control strategy. This approach is less conservative than the multi-step approaches in the literature [104], [105].

Let us consider an admissible equilibrium pair (x_e, u_e) for the disturbance-free system (3.1), i.e., when $w(t) = 0$, for all $t \geq 0$. Clearly, x_e and u_e satisfy the equation

$$x_e = ax_e + bu_e$$

and satisfy the constraints (3.2), (3.3). An affine state feedback control laws of the form is considered

$$u(t) = k(x(t) - x_e) + u_e, \quad t \geq 0, \tag{3.4}$$

where $k \in \mathbb{R}^2$ is the control gain and $u_e \in \mathbb{R}^2$ is a constant vector. A state and input

coordinate transformation is applied by introducing $z(t), v(t)$, defined by

$$z(t) = x(t) - x_e,$$
$$v(t) = u(t) - u_e.$$

Consequently, the closed-loop form of the system (3.1) with the control strategy (3.4) becomes

$$z(t+1) = \max\{(a+bk)z(t) + w(t), -x_e\}. \tag{3.5}$$

Contrary to the nominal, disturbance-free case, for the actual system (3.5) each operating condition refers inevitably to a *set* of average response times x rather than a singleton due to the presence of additive disturbances. This set is known in the control literature as the *minimal robust positively invariant set* or the 0-*reachable set* [106], [107]. It represents the set of states that can be reached from the equilibrium point under a bounded disturbance.

Definition 10. *Consider system* (3.5)*. An interval* $\mathcal{S} = [s_m, s_M]$ *is called an invariant set*[1] *for system* (3.5)*, if* $z(0) \in \mathcal{S}$ *implies* $z(t) \in \mathcal{S}$*, for all* $t \geq 0$ *and any* $w(t) \in [w_m, w_M]$*. If, additionally,* $|a+bk| < 1$*, an interval* $\mathcal{S}_{\min}$ *is called the* minimal invariant set *with respect to* (3.5) *if it is invariant and it is included in any other invariant interval. Last, consider the constraints* $z(t) \in \mathcal{Z} = [z_m, z_M]$*. The interval* $\mathcal{S}_{\max} \subseteq \mathcal{Z}$ *is called the* maximal admissible invariant set *with respect to* (3.5) *if it is invariant and includes any other invariant interval.*

Computing the minimal invariant set exactly is difficult, since in the general case it is the limit of a set sequence which converges only asymptotically. Nevertheless, in this case since it was preferred to utilize scalar systems, it has an analytical description. This fact allows the simultaneous characterization of a stabilizing control gain and the minimal invariant set.

[1] By invariance, robust positive invariance is meant, or D-invariance, see, e.g., [107],[108].

Theorem 2. *Let $x_e \in \mathbb{R}, u_e \in \mathbb{R}^2$ and $k \in \mathbb{R}^2$ satisfy* (3.6)–(3.11)

$$(1-a)x_e = bu_e, \tag{3.6}$$

$$0 \le x_e \le x_M, \tag{3.7}$$

$$u_m \le u_e \le u_M, \tag{3.8}$$

$$\frac{w_M}{1-a-bk} \le x_M - x_e, \tag{3.9}$$

$$0 \le a + bk < 1, \tag{3.10}$$

$$\max\{\frac{u_m - u_e}{x_M - x_e}, \frac{u_M - u_e}{-x_e}\} \le k \le \min\{\frac{u_M - u_e}{x_M - x_e}, \frac{u_m - u_e}{-x_e}\}. \tag{3.11}$$

The following hold.

(i) *The set*

$$\mathcal{S}_{\min} = \left[\max\{x_e + \tfrac{w_m}{1-a-bk}, 0\}, x_e + \tfrac{w_M}{1-a-bk}\right] \tag{3.12}$$

is the minimal robust positively invariant set with respect to the system (3.1) *under state feedback* (3.4)*.*

(ii) *The set $\mathcal{S}_{\max} = \mathbb{X}$ is the maximal robustly invariant set with respect to the system* (3.1) *under state feedback* (3.4)*.*

(iii) *For any initial condition $x(0) \in \mathcal{S}_{\max}$ and any positive number ϵ, there is a time $T > 0$ such that*

$$\max_{y \in \mathcal{S}_{\min}} |x(T) - y| \le \epsilon. \tag{3.13}$$

Proof. **(i)** From (3.6)–(3.8), x_e is an admissible equilibrium point for the nominal system (3.1) with control input u_e. By (3.10) and [108, Theorem 4.1], the minimal invariant set with respect to (3.5) is given by the limit of the forward reachable sets sequence. In this case, this sequence is defined by the iteration[2]

$$\begin{aligned} \mathcal{R}_0 &= \{0\}, \\ \mathcal{R}_{i+1} &= \big((a+bk)\mathcal{R}_i \oplus [w_m, w_M]\big) \cap [-x_e, \infty). \end{aligned}$$

[2]For two sets $\mathcal{X}$, $\mathcal{Y}$, we have $\mathcal{X} \oplus \mathcal{Y} = \{x + y : x \in \mathcal{X}, y \in \mathcal{Y}\}$.

Since we are dealing with intervals, it is straightforward to see that for any $i \geq 0$

$$\mathcal{R}_i = \left[\max \left\{ \sum_{k=0}^{i-1} (a+bk)^i w_m, -x_e \right\}, \sum_{k=0}^{i-1} (a+bk)^i w_M \right],$$

and consequently, the minimal invariant set for the system (3.1) is directly given by (3.12).

(ii) By setting $z = x - x_e$, it is shown that $\mathcal{S}_{\max} = \mathcal{S}_1 \cap \mathcal{S}_2$, where $\mathcal{S}_1 := \{z : u_m - u_e \leq kz \leq u_M - u_e\}$ and $\mathcal{S}_2 := \{z : -x_e \leq z \leq x_M - x_e\}$. Specifically, it is shown that $\mathcal{S}_2$ is invariant and also $\mathcal{S}_2 \subseteq \mathcal{S}_1$. Since $\mathcal{S}_2$ is the translation of the state constraints $\mathbb{X}$ in z, the claim will be proved.

To this purpose, for $\mathcal{S}_2$, it is first assumed that $w = w_M$; then from (3.9) we get $w \leq (1 - a - bk)(x_M - x_e)$. Considering the maximum value of $\mathcal{S}_2$ $z_0 = x_M - x_e$, then $z_1 = x_M - x_e$, $z_1 \in \mathcal{S}_2$. Accordingly, considering the minimum value of $\mathcal{S}_2$ $z_0 = -x_e$, then $z_1 = (1 - a - bk)x_M - x_e$ and by applying (3.10) we still get that $z_1 \in \mathcal{S}_2$. Next, let us assume that $w = w_m$; as it stands, $w_m < w_M$ so the aforementioned paradigm let us conclude that $z_1 \in \mathcal{S}_2$. Thus, by induction, it is concluded that $-x_e \leq z_{t+1} \leq x_M - x_e$ while $z_t \in \mathcal{S}_{\max}$, for all $t \geq 0$ and any $w(t) \in [w_m, w_M]$.

To show $\mathcal{S}_1 \supseteq \mathcal{S}_2$, it suffices to show that $-x_e \in \mathcal{S}_1$ and $x_M - x_e \in \mathcal{S}_1$. Indeed, for $z_1 = x_M - x_e$ it holds that

$$\frac{u_m - u_e}{x_M - x_e} \leq k \leq \frac{u_M - u_e}{x_M - x_e},$$

while for $z_1 = -x_e$ we have that

$$\frac{u_M - u_e}{-x_e} \leq k \leq \frac{u_m - u_e}{-x_e}$$

Both sets of the inequalities are satisfied due to the hypothesis (3.11). Consequently, $\mathcal{S}_1 \supseteq \mathcal{S}_2$ and since $\mathcal{S}_2$ invariant, $\mathcal{S}_{\max}$ is invariant and admissible as well. Maximality of $\mathcal{S}_{\max}$ follows directly by observing that any $x_0 \notin \mathcal{S}_{\max}$ violates the state constraints (3.2).

(iii) It is shown that any trajectory beginning from $\mathcal{S}_{\max}$ is driven asymptotically (in fact exponentially) to $\mathcal{S}_{\min}$. To this purpose, for any $z_0 \in \mathcal{S}_{\max}$ then after i time intervals it holds that,

$$z_i = (a+bk)^i z_0 + \sum_{j=0}^{i-1} (a+bk)^j w_j, \tag{3.14}$$

where $w_j \in [w_m, w_M]$, $j = 0, ..., i-1$. By (3.10), the first term in (3.14) converges to zero exponentially, while the second term, as shown in (i), is bounded in $\mathcal{S}_{\min}$. Thus, given any $\epsilon > 0$ and setting $a + bk = l < 1$, from (3.14) we have that since $z_0 \in \mathcal{S}_{\max}$, then necessarily $z_i \in l^i \mathcal{S}_{\max} \oplus \mathcal{S}_{\min}$. Consequently $z_i \in \mathcal{S}_{\min}$.

Since $S_{\max}$ and $S_{\min}$ are intervals containing zero, a positive scalar can always be found d such that $S_{\max} = dS_{\min}$, thus $z_{i+1} = (l^i d + 1)\mathcal{S}_{\min}$. Thus, (3.13) can be satisfied for any T such that $(l^T d + 1)\mathcal{S}_{\min} \leq (1 + \epsilon)\,\mathcal{S}_{\min}$, or, $T \geq \log_l \frac{\epsilon}{d}$.

□

Remark 2. *Any choice of the control gain k that satisfies the relations* (3.2) *and* (3.3)*, will render the whole constraint set as invariant.*

Theorem 2 characterizes simultaneously the minimal invariant set and the gain of the associated control law. More importantly, it provides a tractable method of retrieving $S_{\min}$ and k. Specifically, for each model of (3.1) and given the equilibrium x_e, a feasible equilibrium pair (x_e, u_e), close to the pair of the desired values $(x_e, u_e^\star)$, and a state feedback control law of (3.4), which steers the closed loop system inside the minimal invariant set, can be calculated by solving the following linear programming problem,

$$\min_{u_e, k} \ \|u_e - u_e^\star\|_\infty \tag{3.15a}$$

subject to

$$(1 - a)x_e = bu_e \tag{3.15b}$$

$$u_m \leq u_e \leq u_M \tag{3.15c}$$

$$bk \leq 1 - a - \frac{w_M}{x_M - x_e} \tag{3.15d}$$

$$-u_e - (x_M - x_e)k \leq u_m \tag{3.15e}$$

$$u_e + x_e k \leq u_M \tag{3.15f}$$

$$u_m \leq u_e + (x_M - x_e)k \leq u_M \tag{3.15g}$$

$$u_m \leq u_e - x_e k \leq u_M \tag{3.15h}$$

$$0 \leq a + bk < 1 \tag{3.15i}$$

where constraint (3.15b) ensures that (x_e, u_e) is an equilibrium pair, (3.15c) means that the input constraints are satisfied. The constraint (3.15d), identically to (3.9), indicates that (x_e, u_e) belongs to $\mathcal{S}_{\min}$, whereas the constraints (3.15e)-(3.15h) are an analytical description of (3.11) ensuring that (x_e, u_e) belongs to $\mathcal{S}_{\max}$. Finally, the constraint (3.15i) is identical to (3.10).

Apart from the guarantee of the QoS metrics, the computed feasible operating points are used by the upper control level to determine the operating state of the activated VMs. The Horizontal Scaler, as it is described in Section 3.5.2, selects the operating area of each activated VM from the set of feasible operating point. Complementary to this, the local controller ensures that the chosen VM operating state will be realized by the described vertical scaling approach.

3.5.2 Horizontal Scaling

As discussed earlier, the upper control level consists of two essential components; the Horizontal Scaler and the Workload Predictor. The former aims to implement the appropriate VM placement on the minimum number of active edge servers, in order to satisfy the total workload of the co-hosted applications. The latter estimates the workload for the following time window, based on the previous actual value measured. This control level considers a cluster of edge servers located in a single place. Load balancing between geographically dispersed edge server clusters is not goal of this work, but is part of future research.

Horizontal Scaler

The Horizontal Scaler aims to compromise the mutually exclusive goals of performance and resource utilization. In particular, since the edge servers' resources are not abundant, unregulated performance demands for a single application would require the high allocation of computational resources on all servers, leaving the co-hosted applications in resource starvation. This is not desirable if the QoS requirements are met with less resources. The Horizontal Scaler component is responsible for optimizing the VMs' instantiation and for distributing the total requests of the implemented applications among them. The optimization objective of this approach is to minimize the number of the active edge servers, with the constraint of meeting the total workload demands. This indirectly results in reducing the

consumed energy and optimally allocating the resources in the server side. The proposed Horizontal Scaler component leverages the fact that the size of a cluster of edge servers is small compared to a cloud datacenter, thus a heuristic solution can be reached with small computation effort. In this approach, the assumption that each edge server hosts at most one VM per application is made. Taking this into consideration, the Horizontal Scaler's functionality breaks down in two steps; at the first off-line step, it computes all the feasible VM placements within a single server, based on the set of the VMs' feasible operating points. These feasible placements are the ones where the total CPUs and memory required from the co-hosted VMs' operating points do not exceed a predefined threshold. Since memory is not considered as a control variable, a static portion of memory r_e is assigned to every feasible operating point. For example, assume two applications App^x and App^y; a VM running App^x and instantiated at an operating point which requires 25% allocated CPU and 4GB of RAM, alongside a VM running App^y and instantiated at an operating point, which requires 55% allocated CPU and 8GB of RAM, is a feasible VM placement for a single edge server, as the total allocated CPU and memory do not exceed the threshold C_E, set at the 90% of the server's total CPU capacity and R_E set 32 GB of RAM respectively. More formally, the set of all feasible VM placements is defined as,

$$\mathcal{P} := \{p_i = \Big((u_{1e}^1, r_e^1), \ldots, (u_{1e}^N, r_e^N),\Big)\ ,\ i = 1, \ldots, N : \sum_{i=1}^{N} u_{1e}^i \leq C_E, \sum_{i=1}^{N} r_e^i \leq R_E\}$$

Then, assuming this set $\mathcal{P}$, this set's cardinality $|\mathcal{P}|$ and the total number of the edge servers E, it determines the number of servers to be activated E_A, by solving the following mixed integer linear program in an on-line fashion,

$$\min_{f_i, E_A} \{E_A\} \tag{3.16a}$$

subject to

$$f_i \geq 0,\ i = 1, \ldots, |\mathcal{P}| \tag{3.16b}$$

$$E_A = \sum_{i=1}^{|\mathcal{P}|} f_i \tag{3.16c}$$

$$0 \leq E_A \leq E \tag{3.16d}$$

$$\sum_{i=1}^{|\mathcal{P}|} f_i u_{2e}^j \geq \tilde{\Lambda}^j,\ j = 1, \ldots, N \tag{3.16e}$$

where the positive integer variables f_i denotes how many servers with the p_i VM placement of set $\mathcal{P}$ need to be activated. As the constraint (3.16c) denotes, the sum of these variables is equal to E_A. The constraint (3.16d) simply restricts these activated servers to the total number of the edge servers. Finally, the last N constraints of (3.16e) denote that the estimated total workload for each application $\tilde{\Lambda}^j$, as it is computed in the following subsection, is satisfied by the selected VM placements. It is important to point out that the Horizontal Scaler component is triggered only if a significant variation in any of the application's workload occurs. This intends to avoid the frequent server activation/deactivation, which leads to oscillation of resource allocation and degradation of the VM's performance.

Workload Predictor

For each application, the total incoming RR is estimated by the Holt linear exponential smoothing filter [109] that captures the linear trend of time series. For any time interval i, the one-step prediction $\tilde{\Lambda}(i)$ of the incoming request rate $\Lambda(i)$ is:

$$\begin{aligned}
&\tilde{\Lambda}(i) = \hat{\Lambda}(i) + c(i), \\
&\hat{\Lambda}(i) = \alpha\Lambda(i) + (1-\alpha)(\hat{\Lambda}(i-1) + c(i-1)), \\
&c(i) = \beta(\hat{\Lambda}(i) - \hat{\Lambda}(i-1)) + (1-\beta)c(i-1).
\end{aligned} \tag{3.17}$$

where α, β are smoothing constants, $\hat{\Lambda}(i)$ is the smoothed value and $c(i)$ denotes the linear trend in the measurement series. For the initialization, a random value of $\hat{\Lambda}(0)$ is used within

the range of the incoming RR and $c(0) = 0.5$.

3.6 Experimental Evaluation

In this section, an experimentation on the proposed computation offloading mechanism is presented together with the respective results. These results illustrate the success of this approach in guaranteeing the stability of application response times within an acceptable margin. The optimization of the resource allocation in terms of edge servers activated to serve the incoming workload is highlighted. Moreover, an experimental comparison between the vertical scaling part of this mechanism and [1] is demonstrated. The benchmarking is performed using CloudSim Plus [110], a simulation environment suitable for cloud computing and MCC experimentation, on a dual-core, macOS powered system with 8GB of available memory.

3.6.1 Horizontal Scaler's Complexity

Before proceeding with the detailed presentation of the experimental setup used throughout the detailed evaluation study and the presentation of the corresponding performance of the proposed computation offloading mechanism, some initial numerical results are presented regarding the complexity of the Horizontal Scaler. As expected, the problem solved is a combinatorial one expressed as a mixed integer linear program in (16). For treating the mixed integer problem of the Horizontal Scaler the GLPK solver [111] is used. The problem under consideration is generally NP-Hard and the lower bound of the computational complexity of the Branch-and-Cut algorithm used to find a solution is exponential [112]. Specifically, in the following, the performance of the Horizontal Scaler is analysed considering the dominant parameters of the optimization problem: the number of mobile applications, the total number of the feasible operating points of all applications and the number of available edge servers.

Figure 3.3 illustrates the effect of the preceding parameters. The left panel demonstrates the effect of the number of the feasible operating points. Three applications are co-hosted in a cluster of servers and the number of available operating points per application varies from 3 to 6, which produces a set $\mathcal{P}$ with a cardinality of 27 to 116 respectively. Subsequently,

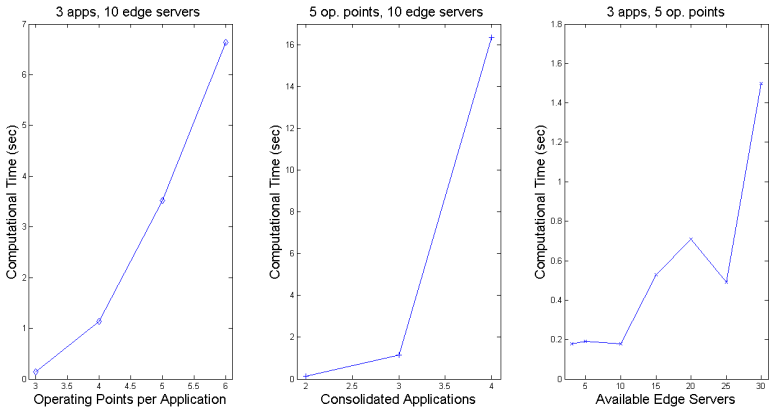


Figure 3.3: Analysis of Horizontal Scaler's Computational Complexity.

the computational time of (3.16a) increases accordingly. The middle panel of Figure 3.3 shows that the computational time also increases as the consolidated applications grow in numbers. More applications lead to more operating points, and consequently to the exponential increase of the computational time. Finally, at the right side of Figure 3.3, the effect of the number of the available edge servers is illustrated. As observed by the corresponding results, this parameter substantially affects the computational time only when the number of active edge servers is high. However, it should be noted that mobile edge computing, contrary to the traditional cloud environment, is usually based on small/medium data centers that typically are expected to host few applications.

3.6.2 Experiment Setup

In this simulation, which spans around $4h$ and $10min$, or $15000sec$, three physical machines with 32GB of RAM which are utilized as edge servers; as mentioned earlier, each of them is manually restricted to hosting at most two isolated VMs, each of which realizes one of

Table 3.2: VM Operating Points $(x_e, u_{1e}, u_{2e}, r_e)$.

VMs of App^1	VMs of App^2
$(0, 0, 0, 0)$	$(0, 0, 0, 0)$
$(3, 25, 2.95, 4)$	$(3.75, 25, 3.23, 4)$
$(3, 35, 4.63, 6)$	$(3.75, 35, 5.29, 6)$
$(3, 45, 6.18, 6)$	$(3.75, 45, 7.38, 6)$
$(3, 55, 8.02, 8)$	$(3.75, 55, 9.58, 8)$

the two supposed applications, ($N = 2$), named App^1 and App^2; the edge servers are also restricted to hosting no more than one VM per application. More specifically, this notation is followed: VM_{ij} corresponds to the VM running on the i_{th} server and implementing the j_{th} application. The mobile traffic is simulated with a Poisson distribution of requests arriving at the Horizontal Scaler component, while the length of each request follows an Exponential distribution. For both applications, the incoming offload RR varies between 1 and $25 req/sec$. However, for each of the application-specific VMs, the distributed RR range is divided in the following four subintervals: $[0, 3.5]$, $[3.5, 5.5]$, $[5.5, 7.5]$ and $[7.5, 10]$. A model of (3.1) is identified and an equilibrium point and a control law are computed by solving (3.15a) through (3.15i) for every subinterval. Thus, in total eight systems and their respective controllers are identified offline. The worst acceptable response time for the offloaded requests is set to $6sec$ and $7.5sec$ for App^1 and App^2 respectively. The desired average response time of the equilibrium points of the applications are set to the half of these values, $x_e^1 = 3$ and $x_e^2 = 3.75$. Indicatively, Table 3.2 depicts the operating points computed for both applications $(x_e^i, u_{1e}^i, u_{2e}^i, r_e^i)$. The first operating point, with zero input and average response time, corresponds to an inactive VM. Table 3.2, also, justifies the assumption of hosting only one VM per application per server. For example, co-locating two VMs of App^1, namely running on the second operating point of Table 3.2 would result in cumulatively serving less offloaded requests on average than deploying a single VM running on the fourth operating point, though the latter choice would result in allocating less CPU. This is a consequence of the related operating overhead of each separate VM deployment.

As described earlier, at the end of the time window, i.e., every $30sec$, the Workload Predictor estimates the incoming RR for the next window. When the previously predicted RR and the currently predicted RR, have an absolute difference greater than a predefined threshold, specific to the nature of this application, the Horizontal Scaler is triggered and selects the appropriate VM placement to be instantiated at the edge servers. For the particular applications, this threshold is set at $3req/sec$. The duration of this time window is selected after considering the maximum time it would take for an in-range user to take the decision to offload, connect, offload and receive the results. During this window the request rate remains relatively stable. Much larger time window would fail to adapt to the changes in the request rates, while much shorter time window would probably result in unneces-

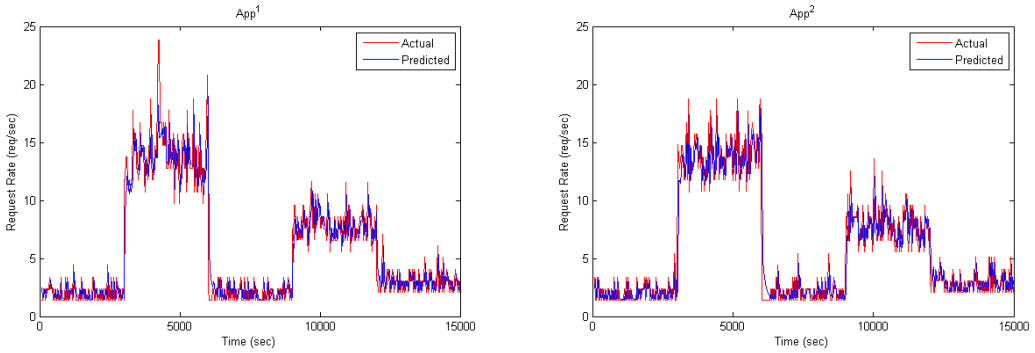


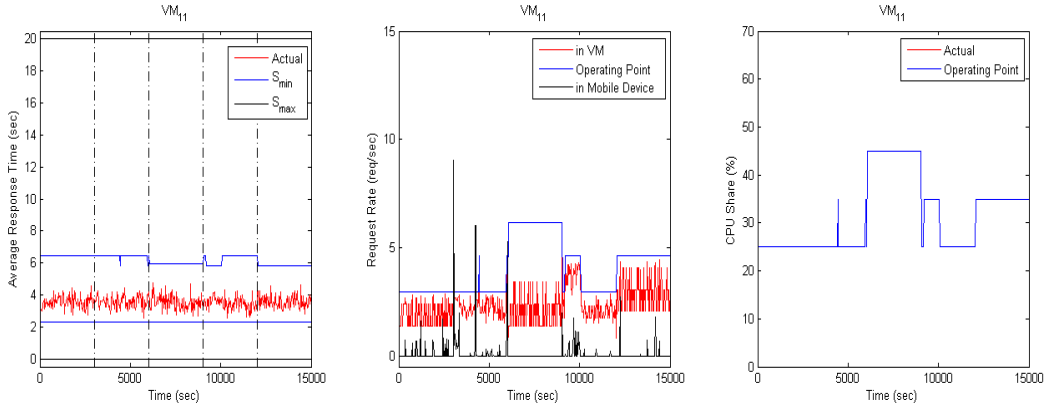
Figure 3.4: Incoming Offloading Request Rates for both App^1 and App^2.

sary invocations. Furthermore, the control interval of 30sec appears to be adequate for the computation of the VM placement by the Horizontal Scaler, as it is later shown in Section 3.6.3. However, for other types of applications, this control time interval could be selected differently: on the one hand to, it could be larger than the computational time of the mixed integer problem that needs to be solved and, on the other hand, it could be large enough to properly follow the variation of the incoming requests.

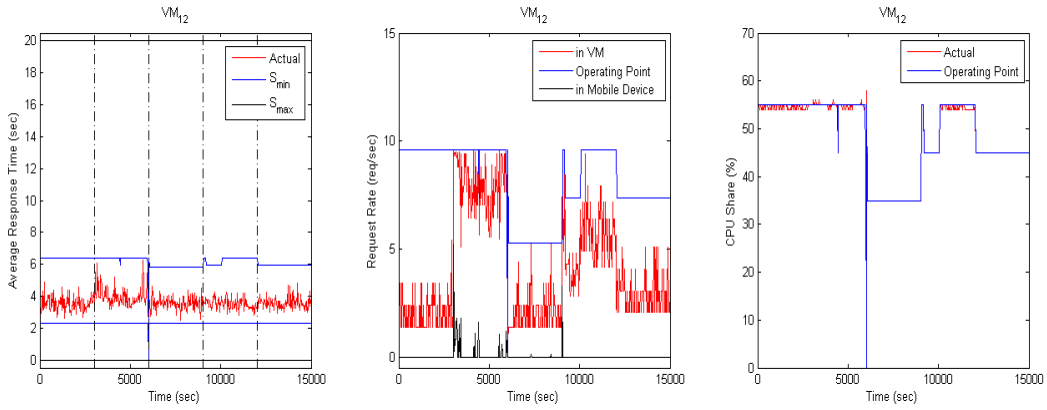
3.6.3 Numerical Results

The results depicted in Figures 3.4 through 3.8 are used to evaluate the efficiency of the proposed mechanism in the preceding scenario. Figure 3.4 depicts the fluctuations in the actual (red line) and the predicted (blue line) RR per application during the experiment. For both applications, the actual incoming RR is altered every $50min$, or $3000sec$. Figures 3.5 through 3.7 illustrate the measured average response time and the inputs per VM in each server respectively; the left graph of each figure depicts the actual application response time (red line) together with the boundaries of the positive invariant sets, $\mathcal{S}_{min}$ (blue line) and $\mathcal{S}_{max}$ (black line). The middle shows the actual RR served by the respective VM on the edge server (red line), together with the RR rejected by the particular VM and sent back to the mobile device for execution (black line). The nominal RR value of the selected VM's operating point is also shown (blue line). In the right graph, the actual CPU share allocated to the VM is shown (red line), alongside the operating point's nominal value (blue line), for each given moment.

It can be observed, in the left graph of subfigures 3.5a, 3.6a and 3.7a, that the aver-



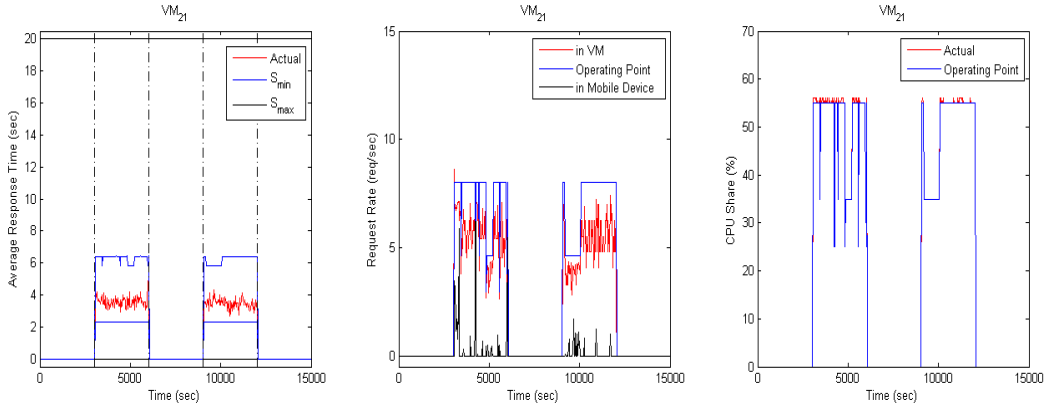
(a) VM_{11} Performance.



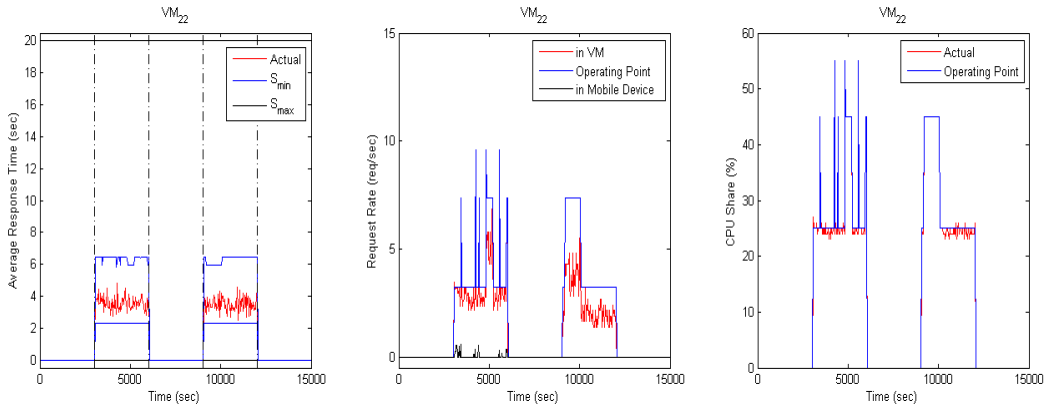
(b) VM_{12} Performance.

Figure 3.5: Average Response Time, Request Rate and CPU Share Allocation of VMs in first server.

age application response time for App^1 remains between the given constraints, despite the workload fluctuation. The similar results are observed in the left graph of subfigures 3.5b, 3.6b and 3.7b for App^2. This means that the theoretical guarantees of Theorem 2 (i) are translated in the response times not exceeding the boundaries of the minimal invariant set, $\mathcal{S}_{\min}$ and $\mathcal{S}_{\max}$. The middle graphs of Figures 3.5 through 3.7 depict how the Horizontal Scaler adapts to these fluctuations and selects the appropriate placement, in terms of number of active edge servers, VMs and their operating points, in order to meet the demanded RR. As shown in Figure 3.8, it activates one edge server between $0 - 3000sec$, $6000 - 9000sec$ and $12000 - 15000sec$; two edge servers between $9000 - 12000sec$ and three between $3000 - 6000sec$. Of these incoming workload fluctuations, the rapid ones, e.g., around the $3000sec$ area, allow to also demonstrate the Local Controllers' functionality; in



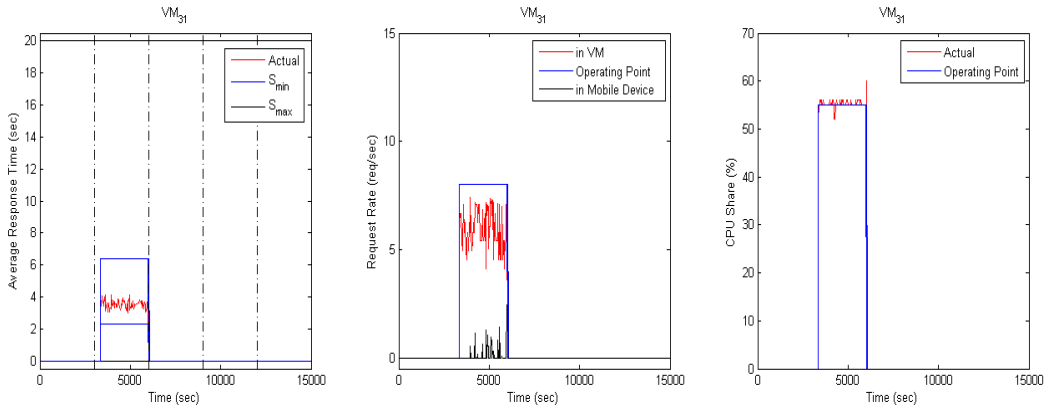
(a) VM_{21} Performance.



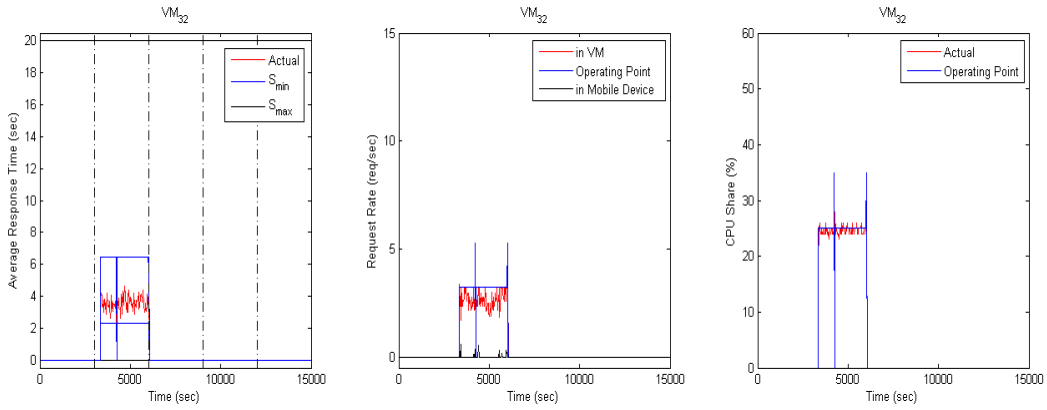
(b) VM_{22} Performance.

Figure 3.6: Average Response Time, Request Rate and CPU Share Allocation of VMs in second server.

such situations, the Workload Predictor component requires a time window to adapt, due to the fact that the estimated RR value is based on the previous actual incoming RR value. This results in the Horizontal Scaler failing to select the appropriate VM placement for the specific time window. However, each VM's Local Controller proves to be en garde by rejecting the excessive offloading requests and redirecting them back to the mobile device for execution, in order to guarantee the stability of the response time. This guarantee is also provided by the Local Controller in the form of vertically scaling the VM; the Workload Predictor's minor inaccuracies are handled by moderately regulating the CPU resources and the accepted RR within limits of the operating point's area. This procedure is illustrated in the middle graph of each subfigure of Figures 3.5 through 3.7; when the RR accepted in the VM has reached the value calculated from the Local Controller for the selected operating



(a) VM_{31} Performance.



(b) VM_{32} Performance.

Figure 3.7: Average Response Time, Request Rate and CPU Share Allocation of VMs in third server.

point, the excessive, rejected RR, which as a consequence is relocated to the mobile devices for execution, is increased. Also at the third graph of each subfigure, where some minor fluctuations are observed in the actual CPU share from the respective nominal values of the operating point. It is important to remark that for every VM and for the most part of the experiment, the actual and the nominal values of the CPU share overlap, making only the blue line observable. Furthermore, some short sudden changes in the selected operating points of the VMs, depicted in the second and third graphs of the subfigures, occur due to certain spikes in the incoming RR; these spikes are so acute that the Horizontal Scaler's trigger condition is satisfied. Consequently the appropriate VM placement is recalculated with the updated operating points. It can be seen that it is this combination of horizontal and vertical scaling that results in the overwhelming majority of offloading requests being

successfully served; 95.18% of the total requests for App^1 and 98.74% for App^2 respectively.

Another interesting remark is that the Horizontal Scaler selects a VM placement, which minimizes the number of active servers but not necessarily the total allocated CPU share. This happens due to the structure of the optimization problem's objective function in (3.16a). One approach to additionally include this optimization objective in this framework would be to revert to multi-objective optimization, either by using preemptive optimization or a multi-objective cost. However, this would significantly increase the time complexity of the decision-making part without envisioning substantial benefits.

3.6.4 Comparative Results

A second experiment better demonstrates the performance of the proposed vertical scaling mechanism alone and compares it to [1]. This is an energy-aware offloading approach, which uses edge server VMs with fixed CPU shares allocated. The offload decision depends on an SLA threshold for the response time of the offloaded requests, named T_d. At the end of each time window, i.e., every $30sec$, an estimation of the incoming RR for each application is computed by (3.17) and the input vector is updated according to (3.4), regarding the following window. The upper left graph of Figure 3.9 depicts the actual response time and the boundaries of $\mathcal{S}_{\min}$ and $\mathcal{S}_{\max}$ for $App1$. After the initial interval, the response time steers from $\mathcal{S}_{\max}$, which is equal to $\mathbb{X}$, to $\mathcal{S}_{\min}$ and remains within. This proves the validity of Theorem 2 (iii). In particular, by computing the control law solving the linear program (3.15a) through (3.15i), the convergence to the minimal invariant set is shown. The upper right graph shows the average response time for allocated $CPU = 25\%,\ 45\%$, and $T_d = 6$ of the approach [1]. In the first quarter of this graph, the SLA is violated for the under provisioned VM with $CPU = 25\%$. The second row of Figure 3.9 again illustrates the request rates served by the edge server and the mobile devices. On the left side, the discussed approach seems to adapt well against the various incoming RR. Once again, the observed

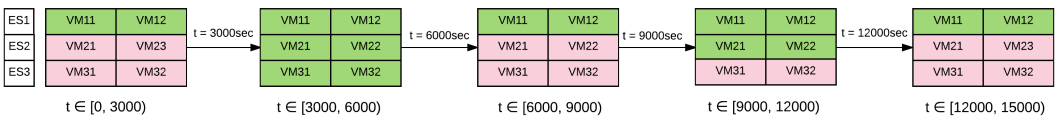


Figure 3.8: Active VMs in Edge Servers

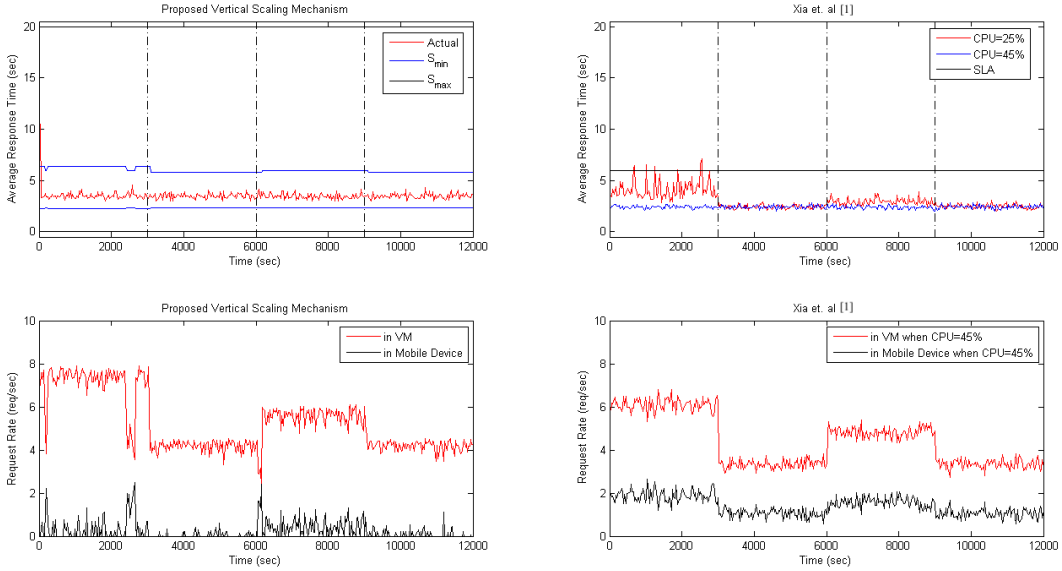


Figure 3.9: Evaluation and comparison of proposed approach with [1]

rapid fluctuation of the served requests exist due to false predictions of the incoming RR. As expected, this does not affect the response time. On the right side, it seems that the use of T_d restricts the amount of requests directed to the edge server. This explains the better response times of the upper right graph. For the proposed offloading mechanism, the requests served at the edge server approach 95.54% of the whole workload, whereas for [1] this percentage is limited to approximately 76%, for both CPU shares. It is clear that the proposed approach performs better against the varying workload because of the vertical scaling of the VMs.

Chapter 4

Control-based Resource Allocation for IoT in Natural Disaster Applications in Edge Computing

4.1 General Setting

As mentioned in Chapter 1, dealing with natural disasters, especially firefighting, can be an interesting field for the application of computation offloading mechanisms, since they can contribute to early and precise detection of forest fires, the most important step for in-time intervention. Thus, in this work, an extension of the work presented in Chapter 3 is integrated in a Cyber-Physical Social System (CPSS), together with computing, networking and human resources, for early fire detection.

Specifically, in this setting, at the bottom level, a network consisting of static or mobile IoT nodes (i.e., UAVs) monitors forest areas for detecting fires at their initial stage. Following the machine learning trend [113], these nodes are equipped with embedded camera modules to enable computer vision-based fire detection. Thus, at the middle level, a Scalable edge

coMputing framewOrK for early firE detection (SMOKE), hosts two image classification services, which process field snapshots captured from the IoT nodes. SMOKE is actually a dynamic resource scaling mechanism for IoT-enabled, time- and mission-critical applications, meant to be deployed at a cluster of servers at the network edge, in the nodes' proximity, assisting the offloading of computationally intensive, energy hungry tasks. At the top level, a Cloud-based decision-making service combines the classification results of the previous level, users' actions on the social media and other services, such as weather information services, in order to accurately infer the fire incident severity and notify the responsible authorities.

4.2 Related Work

In this section, the most interesting studies in bibliography from the perspectives of IoT, Edge and Cloud Computing, social media and image-based wildfire and/or emergency detection are presented.

Zhang et al. [114] composed a study on the application of IoT infrastructures on the fire fighting industries. In that work, the status quo regarding the usage and the main characteristics that make IoT devices appealing, was presented, while, subsequently, suggestions on the wider adoption of them in the fire fighting domain, were discussed, with China as the main example. In the same direction, the authors in [115] analyzed the trend of leveraging Cloud Computing and IoT techniques on agriculture and forestry. On the first part, several relevant applications of these paradigms were listed; there, forest monitoring for fire prevention held a significant place. On the second part, ideas on the combination of those two for maximizing vegetation benefits were proposed. Another IoT setting, this time in the form of a Wireless Sensor Network, was studied in [116], to support early fire detecting activities; this work briefly discussed some indoor as well as forest based installations. Finally, on their search for additional flexibility, the authors in [117] used images captured from UAVs to detect forest fires. The Forest Fire Detection Index was utilised, alongside other classification methods for vegetation classification and tonalities of flames and smoke in order to assess the spread rate.

One of the difficulties that software systems, which aim to integrate multiple information sources, have to face, is the homogenization of data. This issue, also known as

data interoperability, is one of the key requirements in building cross-domain IoT applications and it gets even more complicated when the goal is to combine information sources from completely diverse software services areas (e.g., IoT and social networking services). There are various ongoing standardization efforts toward this scope and organizations such as IEEE Standards Association, AIOTI, oneM2M and W3C are in collaboration trying to reach consensus on defining common APIs and data models within the IoT application domain [118]. In addition, and with regards to Machine-to-Machine interoperability the European Telecommunications Standards Institute (ETSI) contributes to the worldwide standardization efforts along with OneM2M through the *oneM2M Global Initiative* in order to standardize a common M2M service layer platform for globally applicable and access-independent M2M services [119]. In the study [120], authors defined a common approach and data model able to represent in a uniform way information both from IoT environment and social media services. The data homogenization issue in the presented approach is tackled based on the adaptation of semantic and syntactic interoperability mechanisms which are detailed presented in [121, 122, 123].

In the case of wildfires, social media can be a powerful crowd-sensing tool for situation awareness and fast data diffusion. A review on the use of social media on forest fire detection was presented in [124]. This study categorized the wildfire risk management systems and the social media methodologies followed, crowd-sourcing applications developed and social media frameworks deployed for disaster management. Furthermore, a sensing process based on social media data management and a general architecture of a wildfire social sensor management platform were proposed. The following social media-based studies are the most relevant to the discussed approach. Wang et al. [125] provided a Twitter-based spatial, temporal and content analysis for wildfires. The Kernel Density Estimation (KDE) method was used to analyze the possible spatial patterns of the tweets referring to the wildfires. This analysis was also combined with the temporal evolution of the tweets and a term frequency analysis to validate the ability of social media to characterize an emergency over time and space. Also, other parameters, such as the influence of the opinion leaders, were taken into account. Twitcident [126] was a web-based system connected to emergency broadcasting services that automatically searched, filtered and classified emergency situations. Additionally, analytical tools and users were allowed to make customized searches for

specific events, including wildfires.

In the 5G context, dynamic scaling of Edge and Cloud Computing resources, i.e., the on-runtime, on-demand provisioning of the amount and type of server resources, plays a key role for the performance guarantee of time- and mission-critical applications. On the contrary, a priori, static resource provisioning fails to deal with unanticipated changes in resource demands [127]. As explained earlier in Chapter 3, resource scaling can also be classified as either *Vertical* or *Horizontal*. With the term *Vertical Scaling*, a reference is made to the ability of increasing/decreasing the capacity of existing virtual machines or containers by dynamically adding/removing CPU cores, RAM or storage; on the other hand, *Horizontal Scaling* deals with the activation/deactivation of servers and the number of virtual machines or containers to be placed in them. It is reminded that the interested reader can refer to [128] for a complete survey on Cloud elasticity. Leontiou et al. [84] proposed a hierarchical vertical and horizontal scaling framework for Cloud services. At the bottom level, fuzzy Takagi-Sugeno systems were used to model the dynamic operation of the VMs and a robust controller was designed to guarantee the Quality of Service (QoS) requirements and a stability analysis was discussed. At the top level, an unbounded knapsack problem was solved in order to simultaneously tackle the application placement within the active servers and the load balancing of the incoming requests into the VMs. Saikrishna et al. [129] proposed an algorithm to develop a multi-objective switching controller that ensured asymptotic stability with pole placement and addressed the problem of performance management of a web-server hosted on a private Cloud. Moreover, in [130], Grimaldi et al. used a PID gain scheduler to horizontally scale the available resources dynamically and achieve a desired CPU use. Similar to this work, the authors tried to maintain the control error close to zero by splitting the operating spectrum of each VM to distinct regions and solving an optimization problem to calculate the controller gains within them. Finally, the authors of [131] used operating regions, as well, and designed specific models to represent the behavior of each one of them; multiple fixed PI feedback controllers which alternated on runtime based on the operating region, comprised a switching control system that dynamically allocated CPU capacity to the VMs in order to achieve a desired average response time.

Contrary to Cloud Computing, little attention has been given to the dynamic resource

scaling in Edge Computing settings. The resource provisioning problem on Edge Computing is usually dictated by the computation offloading strategy. Again, as explained in the previous chapter, *Computation Offloading* is the process of redirecting the heavy processing tasks of mobile or IoT devices to a nearby Edge Computing infrastructure for execution. Most of the proposed computation offloading studies used fixed modeling and static resource provisioning for the Edge Computing resources. However, these resources are, as mentioned earlier, limited and a dynamic resource scaling approach is necessary to guarantee QoS. Jia et al. [99] proposed a load balancing framework for geographically spread cloudlets, i.e., small-scale data centers or clusters of computers designed to quickly provide Cloud Computing services to mobile and IoT devices, within close geographical proximity. The operation of each cloudlet was modeled with the use of queuing models and static provisioning of cloudlet resources was adopted. A scalable load balancing algorithm was then used to minimize the maximum average application response time of the cloudlets. In [95], the authors presented a comprehensive analysis on energy consumption modeling in Edge Computing settings. These models were classified as either static, flow-based or time-based. Furthermore, considering various network parameters, the energy consumption in Cloud and Edge Computing related scenarios was discussed. The experimental results demonstrated that computation offloading can significantly reduce the energy consumption of IoT devices. MAGA [132] proposed a mobility-aware, genetic algorithm-based decision system that aimed to improve the offloading success rate and reduce the energy consumption on mobile devices while the response time requirements were met. Frequent user mobility patterns were inferred via a tail matching sub-sequence mobile access prediction method and a modified genetic algorithm decided which components of the work flow were to be offloaded or executed locally otherwise. The resource provisioning of the cloudlets was considered static.

4.3 Contribution & Outline

This work proposes a hierarchical CPSS that leverages the advantages of a control-based resource allocation mechanism, in terms of achieving high request throughput, while preserving the QoS above the acceptable levels and keeping the energy consumption minimum. In more detail, at the bottom level, the sensing capabilities of IoT devices are exploited for

continuous fire monitoring. The computation intensive image processing is offloaded to the middle level, where the SMOKE framework implements the horizontal and vertical scaling of the edge servers' resources, in order to guarantee specific response time requirements. Finally, at the top, Cloud Computing layer, the image classification results are forwarded and combined with various sensor data, as well as with a spatial and temporal analysis of social media actions and then a decision making service infers additional information on the incident severity. This information is subsequently forwarded to the responsible local authorities for further actions. As already mentioned, this mechanism is extending the work presented in Chapter 3 and scaling it to accommodate the needs of a CPSS, while now utilising containers instead of virtual machines for serving the workload of the hosted applications. The main contributions of it can be summarized to the following three topics:

1. a vertical scaling mechanism that fits the needs of an emergency detection, IoT-based setting; contrary to a Cloud Computing environment, the computational resources available on the servers located at the edge (specifically in a rural area) are limited [133]. Hence, the simultaneous tenancy of more than one emergency detecting applications at each edge server may risk the Quality of Service (QoS) satisfaction. Consequently, a dynamic resource allocation and admission control mechanism is developed with the use of a linear switching system and a state feedback controller.

2. a two-staged horizontal scaling mechanism; in the same direction, this optimization mechanism is responsible for the activation/deactivation of each edge server, the placement of the applications' instances within them and the distribution of the incoming offloaded requests among those instances, while taking into account various performance criteria.

3. a Cloud decision making service; Among the main challenges in early detection of fire related emergency situations is the richness of the data that are gathered from various sources (either sensors or humans), the efficient and fast processing of them and finally the estimation of the criticality level of the emergency situation. In this work, the decision support system aims to combine data from diverse sources such as IoT-generated images, satellite information, historic weather data and social media services but at the same time aims to produce decisions in a timely manner.

4.4 System Architecture

The hypothetical scenario addressed in this work describes an IoT network, consisting of IoT nodes equipped with camera modules (e.g., Raspberry Pis or IQ FireWatch [134]), capturing images in order to detect emergency incidents (i.e., fire). At the same time, it is assumed that wireless sensors are scattered in the same forest area; in this case, a wireless sensor is nothing more than a low cost sensor, monitoring gas-emissions, humidity or smoke in the trees or vegetation. The proposed CPSS is envisioned as a semi-rural area installation were forests are in the proximity of a populated area. This has a twofold effect on the system; first, its whole operation is based on a local private network (e.g., WLAN) and a cellular network is not necessary. Second, civilians with mobile devices are present. The sensors' data alongside the information provided from the IoT nodes and the social media traffic produced by the human factor, are fed to a decision-making service deployed on the Cloud, able to assert the risk level and give further guidance to public authorities. Finally, an *emergency mode* of the CPSS is specified, when, in the case of a possible fire outbreak and in order to better examine the incident, there is a rapid increase of the pictures needed to be analyzed.

Thus, regarding this IoT-based fire detection scenario, the following identified use-case requirements evince the importance of a scalable Edge Computing architecture to accommodate the offloading of the computationally demanding processes to the network edge.

- Timely incident detection and identification: The ability of wildfires to spread out extremely quickly [135], makes the detection and suppression at an early stage a necessity. Such time-critical applications, demand low-latency access to servers at the edge of the network, ability to perform rapid computations and take immediate decisions.

- Optimal use of IoT nodes resources: As mentioned above, although IoT nodes demonstrate excellent fire-detecting fitting capabilities like automation and control of their functionality in relation to their perception of the environment, wireless data transferring, small size and the ability to form scalable networks, they usually lack the computational and energy resources to perform complex tasks and operate autonomously for prolonged periods of time. As a result, frequent usage of sensors, communication and data processing has to be minimized in order to find a balance between increasing

battery life and accurate incident detection. The proposed Edge Computing approach enables IoT nodes to offload energy and/or computational hungry tasks (i.e., image recognition) to servers in proximity, wirelessly via a local network. This placement enables low-latency access to the servers, contrary to the access to the remote Cloud through the public Internet, which might be unpredictable when it comes to response times.

- Ability to handle the application's rapid scalability needs: The hypothesis that IoT nodes produce a fluctuating workload, depending on whether they operate normally or in the *emergency mode*, increases and decreases the offloading rate for a specific edge server rapidly. As a result, computational needs at the edge of the network may vary differently from time to time for the image recognition application. There is also the possibility that additional applications are co-located at the edge servers; this makes static resource provisioning a problematic situation that may lead to resource under-use and subsequently to hold the ability of applications to coexist at the same server back, or resource over-use which will possibly introduce delay to the execution off the offloaded requests and jeopardize the application's mission-critical aspect. Thus, the need for fine grained resource allocation and QoS guarantee is evident.

- Interoperability of sensors' Data: A critical obstacle when integrating information from heterogeneous sources is that the underlying information systems (e.g., IoT platforms) are mainly isolated and act as "vertical silos". The lack of interoperability among these systems impedes the creation of cross-domain, cross-platform and cross-organizational services. To overcome these obstacles syntactic and semantic interoperability solutions are necessary to be enforced. To this end, syntactic interoperability is associated with the ability of systems to exchange information in order to communicate on a technical abstraction level. Semantic Interoperability, denotes the ability of different applications and business entities to understand exchanged data in a similar way, implying a precise and unambiguous meaning of the exchanged information.

- Privacy protection of individuals: A common challenge that society has to address in the recent years is to keep a balance between preventing and mediating the damage that occurs from disastrous situations without on the same time violating human rights

such as the right of privacy protection of individuals. Advances on sensing technologies and data collection mechanisms make feasible the deployment of vast sensor networks that can potentially become intrusive and violate established regulations (e.g., GDPR). Edge computing paradigm assists in keeping the processing at the edge of the network thus avoiding the indiscriminate transmission and storage of sensitive information, such as image and video recordings.

In this section the design of the CPSS' architecture is described in more detail. As depicted in Figure 4.1, the designed system consists of two main agents, namely the SMOKE framework and the Intelligent Decision Making component, and four subordinate agents which interact with each other and contribute in a unique way to deal with the emergency incident; the IoT nodes, the Sensors, the Social Media and the Public Authorities. The Intelligent Decision Making agent operates as a Cloud component gathering data from the other components, operating as the top layer of this CPSS. Although this proposed architecture is intrinsically linked with the early-fire detection use case, which is studied in this work, it can be easily adapted to accommodate a variety of settings with similar requirements.

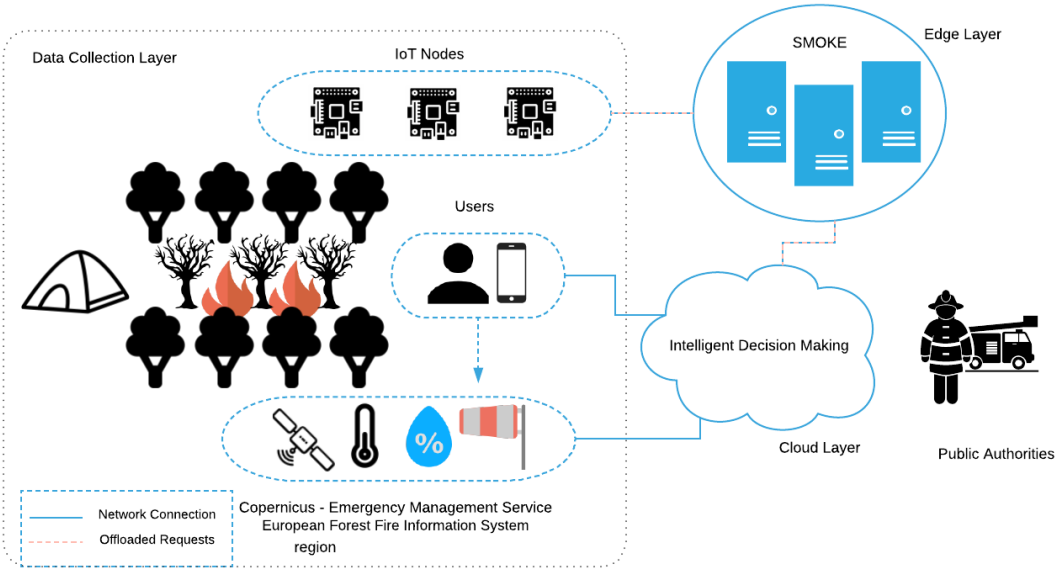


Figure 4.1: CPSS Architecture.

4.4.1 SMOKE

The SMOKE framework follows a top-down design in a manner that there exists a centralized controller that makes the decisions, which are in turn propagated to the lower levels of the architecture to be realized by local controllers. The proposed architecture is generally applicable in a single-site Edge Computing infrastructure but can be easily expanded for Edge-to-Cloud or Edge-to-Edge collaboration; its components are described in detail bellow.

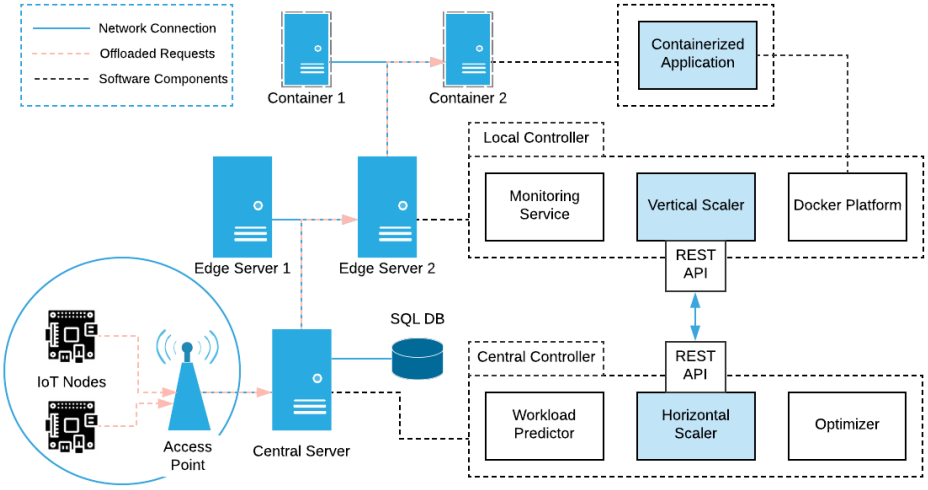


Figure 4.2: SMOKE Architecture.

Containerized Applications

SMOKE supports the simultaneous management of co-hosted applications that are able to receive offloaded requests on the edge servers. The only prerequisite is that those applications be containerized. In this work, for the sake of demonstrating the multitenancy efficiency of SMOKE, two TensorFlow-based object recognition applications, able to recognize images containing events of interest (i.e., fire), were developed and trained off-line in a supervised manner. These applications were then containerized and deployed on the Docker Platform installed of each edge server. Containers were selected instead of VMs, as a means of virtualization, because of their overall lower overhead, smaller footprint and lightweight vertical scalability.

Central Controller

The offloaded traffic, generated by the IoT nodes, is directed to the Central Controller (located on the Central Server) through a local Wireless Access Point. Here lies the upper level control process of the proposed mechanism, as depicted in Figure 4.2. To accommodate this control process, time in the proposed framework is quantized in discrete time intervals; at the beginning of each time interval this component selects an appropriate container formation to be implemented to each edge server directly connected to it and consequently distributes the incoming workload accordingly. This formation defines the number of active servers alongside the number and the operating point of the containers to be placed in them. With the term *operating point* a reference is made to the number of the offloaded requests that each container will accept, the number of cores that it will be allowed to use, as well as the average response time it is requested to achieve during the next time interval. These operating points of the containers are calculated on the *Local Controllers*, as described in the next section. This control process, hereinafter referred to as *Horizontal Scaling*, is performed in an on-line and proactive manner, leveraging an internal prediction mechanism, the *Workload Predictor*, which provides an estimation on the number of requests to be expected on the time interval, for each application. The essential input for this estimation process is provided by the Monitoring Service component of the Local Controller deployed in each edge server, which is responsible for collecting data regarding both the network traffic (i.e., offloading requests admitted and end-to-end response times) and the containers' resources use (i.e., CPU usage) at each given time. Then, the *Optimizer* component uses the output of the Workload Predictor and the feasible operating points, calculated offline, and computes the optimal number of active servers and containers required to meet the different performance criteria. The theoretical background of this process is discussed in more detail in Section 4.5. Hence, depending on the aforementioned decision and considering the predicted workload for each time interval, the Central Controller dictates the creation, scaling and destruction of the application-specific containers to the Local Controllers accordingly. Also, at the end of each time interval, the average classification score of the offloaded images is calculated here. Additionally, when a classification score above a predefined threshold emerges, indicating a possible fire outbreak, information is transmitted to the respective IoT node regarding its new operating mode (normal or emergency). The time between the capturing of the

image of interest and the transmission of the updated IoT node information is defined as the application's response time. All the monitoring data involved in this process are stored in a relational database present in the Central Server, in order to be used for demonstration purposes. It is also worth noting here that, physically, the Central Server is nothing more than an edge server, located on the users' proximity, with a more advanced role; this of the decision maker.

Local Controller

At the lower level, each edge server is equipped with a Local Controller, responsible for both gathering the request-related statistics from the containers, as mentioned earlier, needed for the Monitoring Service and tackling the small fluctuations of the incoming workload, according to the predicted number of requests for each time interval. The lower level control process implemented in this component moderately scales the containers vertically, providing the required resources, thus realizing the decision made by the Central Controller. In more technical terms, the communication between the Central Controller and the Local Controllers is performed via a REST-API present in the latter; the Local Controller also uses the *Docker Platform* in order to scale the containers. This *Vertical Scaling* ensures that the containers remain within an area around the selected operating point, hence guaranteeing minimum and stable application response times, in order to meet certain QoS requirements. Section 4.5 provides further mathematical justification for this process.

4.4.2 Intelligent Decision Making

This Intelligent Decision Making (IDM) service is deployed on the Cloud Layer of the proposed architecture. The geo-tagged images are processed and classified at the SMOKE framework on the edge layer and, if the average classification score is above a confidence threshold of fire and smoke detection, then an ongoing emergency situation probably occurs and the IDM's operation is triggered. In order to further evaluate the criticality of the incident at the targeted area additional data are collected by the *Data Collection Engine* in order to be fed to the *Decision Algorithm*. The latter applies logical rules on the provided data collection in order to timely infer the level of the associated danger of the situation and render the respective estimation in an intuitive manner.

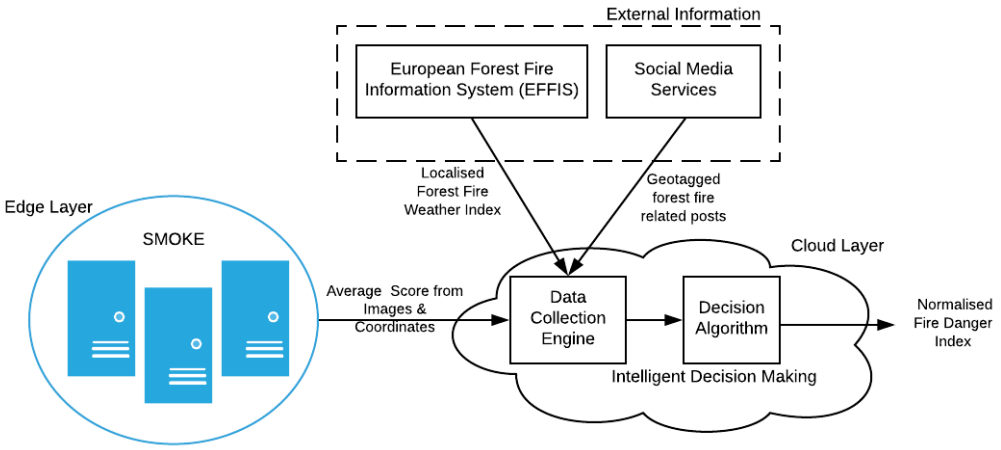


Figure 4.3: Intelligent Decision Making Architecture.

Data Collection Engine

As illustrated in Figure 4.3, the main data sources integrated with this system are:

- SMOKE service: The SMOKE service provides the classification score of the images along with the time and location, in terms of GPS coordinates corresponding to the IoT node's location, from the associated area that the images were captured. It is the image classification score that triggers the overall data gathering process, while additional data are retrieved based on location criteria indicated by the respective images coordinates. To this end, a reverse geo-coding process is applied in order to extract the associated readable addresses and place names which are particularly useful in the retrieval of data from social media services.

- European Forest Fire Information System: The European Forest Fire Information System (EFFIS) calculates on a daily bases an index, called Forest Fire Weather Index, for all the regions of EU based on environmental and weather related information, such as the humidity of the air at the beginning of the afternoon; the temperature in the middle of the afternoon; the precipitation during the last 24 h; the maximum speed of the average wind. The fire danger is mapped in five classes with a spatial resolution of about 16 km. The fire danger classes are the same for all EU countries and information is provided encoded as GeoTIFF format maps showing a harmonized picture of the spatial distribution of fire danger level throughout EU. The GeoTIFF

standard allows georeferencing information to be embedded within a TIFF file. The Fire Danger Forecast maps are updated daily and are freely available from the EFFIS online service [136]. The actual file has a small size (<1 MB) hence it can be efficiently processed. The five classes of the Forest Fire Weather Index embedded as different color codes (bands) in the GeoTIFF file are:

1. **LOW**—Fuels do not ignite readily from small firebrands although a more intense heat source, such as lightning, may start fires in duff or light fuels.
2. **MODERATE**—Fires can start from most accidental causes, but with the exception of lightning fires in some areas, the number of starts is generally low.
3. **HIGH**—All fine dead fuels ignite readily and fires start easily from most causes.
4. **VERY HIGH**—Fires start easily from all causes and, immediately after ignition, spread rapidly and increase quickly in intensity.
5. **EXTREME**—Fires start quickly, spread furiously, and burnt intensely. All fires are potentially serious.

- Social networking services. In the current implementation of the IDM the Twitter micro-blogging platform has been integrated. Twitter maintains a total number of 335 million monthly active users, who produce more than 500 million number of Tweets per day. The fact that 80 percent of Twitter users use the service through mobile devices, makes this social network an ideal platform for applying the social sensing paradigm. In addition, Twitter has been selected in the scope of the work presented in this thesis due to its openness and the almost unrestricted access to the publicly available user provided content and profile information through APIs. Data collection for the needs of the IDM is facilitated through hashtags and keywords associated with wildfires combined with tags denoting geo-reference. Although Twitter offers the option to geo-tag the provided Tweets, this feature is not frequently used, thus it can not be exploited effectively for the needs of the IDM service. On the other hand, it is a common practice for Twitter users to introduce their own tags in order to express the connection of their post with an area. The reverse geo-coding allows the IDM to extract a set of local area names, also expressed in local language, which will be

used as keyword criteria for retrieving Tweets that potentially are associated with an emerging wildfire incident.

Decision Algorithm

The Decision Algorithm aggregates information from the described sources, homogenizes their input and generates a normalized score that ranges from zero to one (highest value) denoting the emergency level. The overall operation is triggered periodically by SMOKE classification score and runs continuously, meaning that the IDM may receive multiple images' scores from various locations, where a decision should be extracted for each case. As it is already stated, the image classification score is the main criterion for the early identification of the fire incident while the additional data gathered are utilised in order to further evaluate the severity of the incident in terms of human presence in the area, potential fire spreading, etc.

4.5 System Modeling

4.5.1 SMOKE Scaling Framework

As described in the previous section, a single SMOKE deployment consists of many Local Controllers and one Central Controller; each Local Controller aims at controlling and regulating the operation of containers that run on the same edge server with it; the Central Controller makes the decisions on the activation of the edge servers and the respective containers, as well as the load balancing of the incoming requests. Trying to be compliant with the taxonomy defined on the survey on [128], the Local Controller was designed to use linear switching systems for modeling the containerized applications and a state feedback controller for each linear system, designed to apply admission control decisions. At the same time, the Central Controller solves a mixed integer programming problem to determine the number of active servers and containers, which are necessary for serving the total workload of the hosted applications.

Local Controller

As mentioned above, the dynamic operation of the containerized applications is modeled with the use of switching linear systems, with the switching criterion being the number of the allocated CPU cores to each container. This modeling approach captures the dynamic behavior of the containers under different operational conditions and enables performing indirect resource allocation. In this case, the various operational conditions, under which the modeling is performed, include different image sizes, resolutions and transmission delays (depending on the network congestion at each given moment) per request and a variety of request rate values. So, for each different CPU core allocation, the operation of the container is described by a discrete linear system of the following form,

$$x(t+1) = ax(t) + bu(t), \tag{4.1}$$

where $x(t)$ is the *state variable* that expresses the average application response time for time interval t and $u(t)$ is the *control variable* that represents the number of the admitted requests within time interval t. Here, with the term admitted the requests that are actually allowed to the container for processing are described. The parameters α and b of the above model are estimated by using the Recursive Least Square (RLS) algorithm [137].

Physically, a container with c allocated cores is constrained to serving up to u_e requests of the containerized application while maintaining an average response time of x_e. This pair (x_e, u_e) is called an operating point and generally, for each such switching system, a set of feasible operating points of this kind can be computed according to various performance criteria and while taking into account the constraints of the state and input variables. In the discussed case, these feasible operating points are computed by solving the following linear programming with the goal of maximizing the number of the admitted requests:

$$\begin{aligned} &\max_{x_m, x_M, x_e, u_m, u_M} \ u_e \\ &\text{subject to} \\ &x_e = ax_e + bu_e \\ &x_m \leq x_e \leq x_M \\ &u_m \leq u_e \leq u_M. \end{aligned} \tag{4.2}$$

The first constraint implies that each operating point must also be an equilibrium of the discrete linear system and this will guarantee its stability and confinement in a specific operating area around it. The second constraint dictates that the state variable must lay between a minimum (x_m) and a maximum value (x_M) set by the application's requirements, while the last constraint refers to the control variable varying between the minimum available (u_m) and the maximum available value (u_M).

For each operating point of every linear system, a state feedback controller is designed in order for the respective containerized application to meet the response time requirements. This control law is defined as,

$$u(t) = K(x(t) - x_e) + u_e, \tag{4.3}$$

where $K \in \mathbb{R}$ is the control gain. Applying (4.3) in (4.1), we get the closed loop form of the linear system,

$$x(t+1) = (a + bK)x(t) + c, \tag{4.4}$$

where $c = b(u_e - Kx_e)$ is a constant term. Regulating the eigenvalue $\lambda = a + bK$ of the system (4.4), the stability of the closed loop system and the convergence speed to the equilibrium point are affected. Thus, a stable eigenvalue is selected, which lays inside the unitary circle, in order to compute the control gain $K = \frac{\lambda - a}{b}$. To give a better understanding of the whole process Figure 4.4 illustrates a block diagram describing the closed loop system. The following list explains the role of each signal presented there,

- *Reference Input*: The average application response time for the respective operating point x_e.

- *Control error*: The difference between the actual average response time of the last interval and the reference value, $x - x_e$.

- *Controller*: An affine switched state feedback control process, as the main process in the Local Controller.

- *Control Input*: The maximum request rate to be admitted at the container for the next time interval, computed by (4.3).

- *Docker*: The Docker Platform as the the control system's actuator.

- *Container*: The containerized applications as the controlled process.

- *Measured Output*: The measured average application response time of the container for the previous time interval.

- *Feedback*: The sensor of the control system, monitoring and recording its current state at each time.

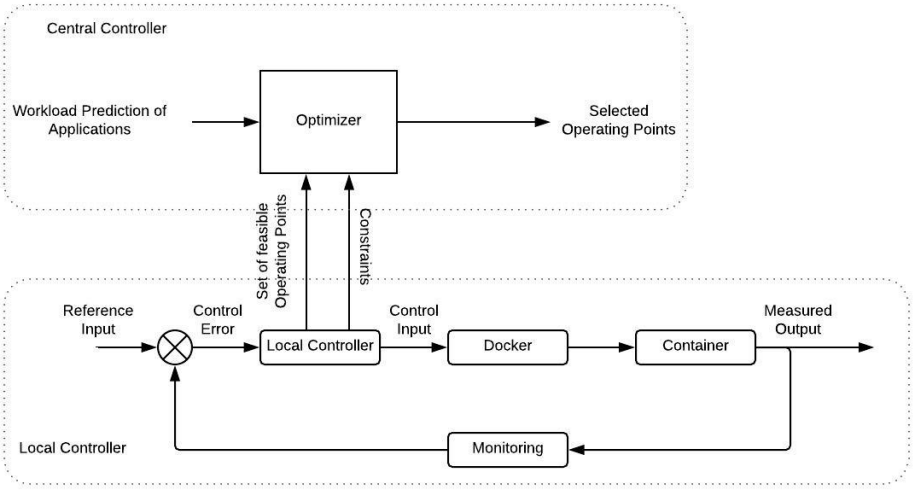


Figure 4.4: Feedback control system in Vertical Scaling.

Central Controller

The main functionality of the Central Controller on the top layer, is to decide the switching action for the Horizontal Scaling process, as described in subsection 4.4.1. As shown in the upper part of Figure 4.4, the Optimizer component consumes information regarding the available operating points and the predicted workload, for the next time interval, for each application. The prediction of incoming workload, estimated according to a linear trend forecasting procedure as described, in [83], alongside the operating points facilitate the formulation of an optimization problem solved by the Optimizer; as mentioned earlier, the assumption that each application is deployed to at most one container per edge server is made. Furthermore, at a preliminary stage, an offline exhaustive procedure where all the feasible combinations of the containerized applications' operating points within a server, is

calculated. Feasible combinations are the ones that do not exceed the total computational capacity of the respective server. Then, the Optimizer minimizes the number of active servers and the allocated CPU resources by solving, sequentially, two mixed integer linear optimization problems (MILPs). At the first MILP, the minimum number of edge servers to be activated on the next time interval, in order to serve the estimated workload, is determined by solving the following,

$$\begin{aligned} &\min_{S,P,A,\tilde{R}_i} \ S_a \\ &\text{subject to} \\ &S_a \in S \\ &p_{ij} \in P \\ &p_{ij} \in [0, S] \\ &\textstyle\sum_{\forall i \in A} \sum_{\forall j \in S} p_{ij} u_{ei} \leq \tilde{R}_i. \end{aligned} \tag{4.5}$$

Here, the first constraint dictates that the servers to be activated are selected from the pool of the available servers. The next two constraints imply that each of the selected containers to be deployed on the selected servers corresponds to an existing operating point and that their placement on the active servers must correspond to an acceptable combination. Finally, the last constraint means that the estimated total workload for each application, $\tilde{R}_i$, must be served by the containers. On the second MILP, the minimum amount of computational resources, in terms of CPU cores per container per server, are computed, while taking into account the result of the previous optimization,

$$\begin{aligned} &\min_{S_a,P,A,\tilde{R}_i} \ \textstyle\sum_{\forall i \in A} \sum_{\forall j \in S_a} p_{ij} C_{ij} \\ &\text{subject to} \\ &p_{ij} \in P \\ &p_{ij} \in [0, S] \\ &\textstyle\sum_{\forall i \in A} \sum_{\forall j \in S} p_{ij} u_{ei} \leq \tilde{R}_i, \end{aligned} \tag{4.6}$$

where C_{ij} is the number of allocated CPU cores to the container of the ith containerized application on the jth activated server. The reason behind distinguishing the optimization process into two distinct subproblems is that, at first, an attempt to solve a Multi-Objective

MILP was made by combining the two optimization targets to one weighted objective function, but that led to the one objective being successfully optimized while the other one concluded to a sub-optimal solution, regardless of the assigned weights. Also it should be noted that the number of edge servers is relatively small so the overall computation time of these optimization processes does not disrupt the Central Controller's smooth operation.

4.5.2 Decision Making Algorithm

As already stated, the decision making process is triggered by the SMOKE system when the fire-related average classification score of images, captured by the IoT node, is above a predefined threshold, thus indicating a considerable probability of fire occurrence. The respective score sent to the IDM is denoted by $S_{im}(t, p)$, where t is the time when the image was captured by the IoT node and $p = (x, y)$ expresses the GPS coordinates of the point where the IoT node was located at upon image capturing (latitude x and longitude y). As already stated, the value of $S_{im}(t, p)$ lies within the range of $[0, 1]$.

Once $S_{im}(t, p)$ is obtained, the IDM triggers a reverse geo-coding process in order to map the GPS coordinates to a specific country and language, and to extract multiple area names for the specified point. The output of this process is a set of location-related keywords or tags denoted by $L(p) = [l_1^p, l_2^p, ..., l_N^p]$. Various combinations $C_r^{lf}(p) = [l_i^p, l_j^p, ..., f_k, f_l, ...]$ of these tags coupled with terms f_i linked with fire emergency are generated and are subsequently fed to the Twitter API in order to fetch all Tweets that involve the specific term combinations and are posted in the latest window frame of duration Δ. The population N_r^{lf} of the retrieved Tweets $TW_r^{lf}(C_r^{lf}(p), [t - \Delta, t]) = [tw_{r1}^{lf}, tw_{r2}^{lf}, ..., tw_{rN}^{lf}]$ that use both keywords linked to the area of interest as well as terms related to fire emergencies is then averaged by the population N_r^l of all Tweets $TW_r^l(C_r^l(p), [t - \Delta, t]) = [tw_{r1}^l, tw_{r2}^l, ..., tw_{rN}^l]$ that only use keywords linked to the area of interest. A respective social media score $S_{sm}(t, p) = \frac{\sum_r \frac{N_r^{lf}(t,p)^2}{N_r^l(t,p)}}{\sum_r N_r^{lf}(t,p)}$ is eventually calculated that lies within the range of $[0, 1]$.

With regards to the Fire Weather Index (FWI) provided by EFFIS, the IDM retrieves the respective values from the GeoTIFF image that correspond to the specified GPS coordinates of the point of interest $p = (x, y)$. To deliver this, mapping of the GPS coordinates above to the GeoTIFF geo reference system is necessary. The value of the retrieved FWI for the specified coordinates is $FWI(t, p)$ and the respective normalized score is $S_{FWI}(t, p) = \frac{FWI(t,p)}{5}$

that lies within the range of $[0, 1]$.

Finally, the IDM estimates an overall score $S_{fire}(t, p)$ of fire incident at point p and time t that indicates the severity of fire incident occurrence and is expressed as

$$S_{fire}(t, p) = w_{im} \cdot S_{im}(t, p) + w_{sm} \cdot S_{sm}(t, p) + w_{FWI} \cdot S_{FWI}(t, p) \tag{4.7}$$

where the weights w_{im}, w_{sm}, w_{FWI} indicate the significance of the respective scores, lie within the range of $[0, 1]$ and for which it stands that $w_{im} + w_{sm} + w_{FWI} = 1$. The experiments conducted so far have assumed that $w_{im} \geq w_{sm} \geq w_{FWI}$, considering the reliability of the respective information sources. The nature of the assigned weights indicate that the image classification engine is considered as more important for the initial detection of the fire incident while the other two sources of information (social media data and EFFIS) have a complementary role in further evaluating the severity of the situation.

4.6 Experimental Evaluation

In this section the experimental setup is thoroughly presented alongside an evaluation of the obtained results. At first, the significant role of SMOKE is evinced, after intense experimentation for proof of concept of the Horizontal and Vertical resource scaling on an Edge Computing topology and a comparison of the proposed architecture with the naive solution of the static resource allocation is presented. Subsequently, a time-related evaluation on the socially-aware intelligent decision making component, with input from social media, additional sensors' data and image classification scores produced by the SMOKE framework, is discussed.

To demonstrate the operation of a SMOKE installation, the hypothetical setting of a forest and IoT nodes mounted on UAVs that fly over it, while capturing images in order to detect fire occurrence, is emulated. As discussed in Section 4.4, at the same time the presence of wireless sensors and civilians using social media is assumed in the field. Furthermore, two additional assumptions are made; first, that due to the large space UAVs need to cover and investigate, each one of them takes over a smaller area, while, second, in the case of a possible fire outbreak, they are dictated to gather to the area of interest to better examine

the incident, leading to a rapid increase of the pictures offloaded. UAVs were selected as the evaluation scenario because, although they pack amazing characteristics like operational versatility and durability in various weather conditions, they usually lack the resources to accommodate long-spanning missions, like vast forest areas surveillance. Additionally, the hypothesis that UAVs gather in a specific geographic area when certain events occur, and spread otherwise, allows for denoting the fine-grained, scaling-enabled resource allocation taking place in the system.

At this point, it should be made clear that UAV-based image analysis and fleet coordination algorithms are decoupled from the CPSS initial design and operation. Therefore, given the context and focus of this work, UAV actual, real world operational methods and algorithms—though very challenging and of high practical importance—are considered out-of-scope of this chapter. They just serve the purpose of a conventional use case to enable the demonstration of the capabilities of such a combination of an Edge Computing framework and a decision-making platform.

4.6.1 SMOKE Evaluation

The first experiment illustrates the performance of the SMOKE framework when deployed on the NETMODE testbed [138] at the National Technical University of Athens in Greece. In this case, two identical 16-core edge servers with 16 GB of RAM were used, that each hosted two TensorFlow [139]-based applications, deployed in separate Docker containers as explained in Section 4.4. Without loss of generality, the assumption that each edge server can host exactly one container of each of these applications is made, due to the fact that more than one instances of the same service would introduce additional overhead costs when deployed on different containers on the same host. The differentiation between those two applications, developed solely for the experimentation purposes, was that the first implemented an image classification for conventional, visible light pictures, while the second one for infrared pictures. The model trained for the conventional image recognition was fed with a specific dataset [140] containing either pictures of forest wildfires or plain forests. For the infrared recognition model, a synthetic dataset was used, generated from the aforementioned one, by applying an infrared Photoshop effect to each image. Following the architecture described earlier, these edge servers also hosted an instance of the Local

Controller. Regarding the emulation of the UAVs, two Raspberry Pi devices were acting as the mobile nodes, each of which was assigned to offloading requests to the edge servers, targeting one of the two applications. The mobile nodes were connected via Wi-Fi to an Alix3d2-based node hosting the Central Controller, which was in turn linked via Ethernet to the two identical edge servers. This whole setting represented a fully deployed SMOKE installation in a forest area. It should be once again noted that the process from the moment a UAV captured an image and offloaded it to the proximate edge server Infrastructure for processing until the result was calculated, as stated before, was defined as the response time. In addition to this, it was assumed that the edge servers were responsible for the UAVs' operating mode, depending on the situation's severity. Thus, the time until the detection of a possible fire incident, i.e., the response time, should kept below an acceptable value. Finally, the average classification score of the captured images was calculated and advertised to the IDM on the Cloud nodes, at each time interval.

In the presented emulation, this time interval, that additionally defines the overall operation of the SMOKE framework, as explained in Section 4.4, is set to 30 *sec*. To make the offloading patterns more plausible, the amount of requests produced within the interval follows a Poisson distribution, while the inter-arrival time between two successive requests follows an exponential distribution. As depicted in the third diagrams of Figures 4.5 and 4.6, the experiment scenario kicks off with the Raspberry Pis offloading an average of around 10 requests per time interval (blue-colored, solid line), for each application, simulating a normal period where no fire indications are present in the forest area (*tracking mode*). After approximately 5 *min*, the average number of requests per interval starts escalating, mimicking the UAVs' behaviour of gradually approaching the area of interest, i.e., the framework's proximity, when an emergency situation is detected *(emergency mode*). This average value peaks and stabilizes at 25 from the 20th to the 40th simulation *min*, where it is assumed that a fire has been recognized and more visual coverage of its area and spread rate is required. Finally, in the last part of the emulation, the offloading request generating rate returns to normal values, reflecting the wildfire being put under control and subsequently the UAVs reverting back to their normal operating mode.

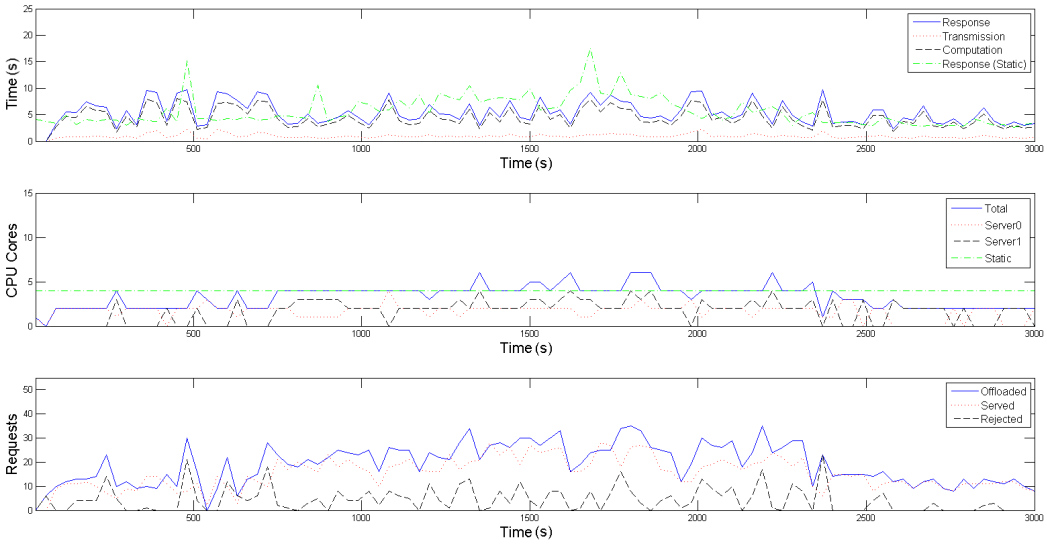


Figure 4.5: Conventional Tensorflow application.

Regarding the evaluation of the scaling mechanism of the proposed framework, one can observe the following; in the second diagrams of Figures 4.5 and 4.6, that the Central Controller's Horizontal Scaler adapts to the increased requirements in computational resources while on emergency mode (blue-colored solid line) and dictates the activation of a second server (black-colored, dashed line) between the 10th and 40th *min* of the emulation, in order to accommodate the mobile devices' higher average offloading request rate. On the contrary, for the most of the initial as well as the final part of the emulation, only one server is active (red-colored, dotted line), proving to be adequate for the tracking mode of the UAVs. As explained thoroughly in Sections 4.4 and 4.5, the Workload Predictor component estimates the incoming workload for the next time interval and then the Horizontal Scaler selects an appropriate formation, in terms of number and operating point of containers to be placed in the active servers, for each following time interval. Table 4.1 contains the offline calculated operating points for both application-specific containers, from which the Horizontal Scaler gets to choose; each operating point defines the nominal amount of offloaded requests, u_e, that the respective container is able to process, alongside the reference input x_e, when 1, 2, 3 or 4 Cores are allocated to it. A remark regarding the restriction of cores to be made available to each container to 4, is that this seemed to be a plateau where the containers became saturated and could not serve significantly more requests, despite allocating more cores to

them. Furthermore, one can observe that certain outliers in the Poisson distribution, of either rapid increase or decrease in the offloading requests, cause the Workload Predictor to poorly estimate the incoming workload for the following time interval; this results to an increased amount of rejected requests, as well as a slight oscillation on the number of activated servers.

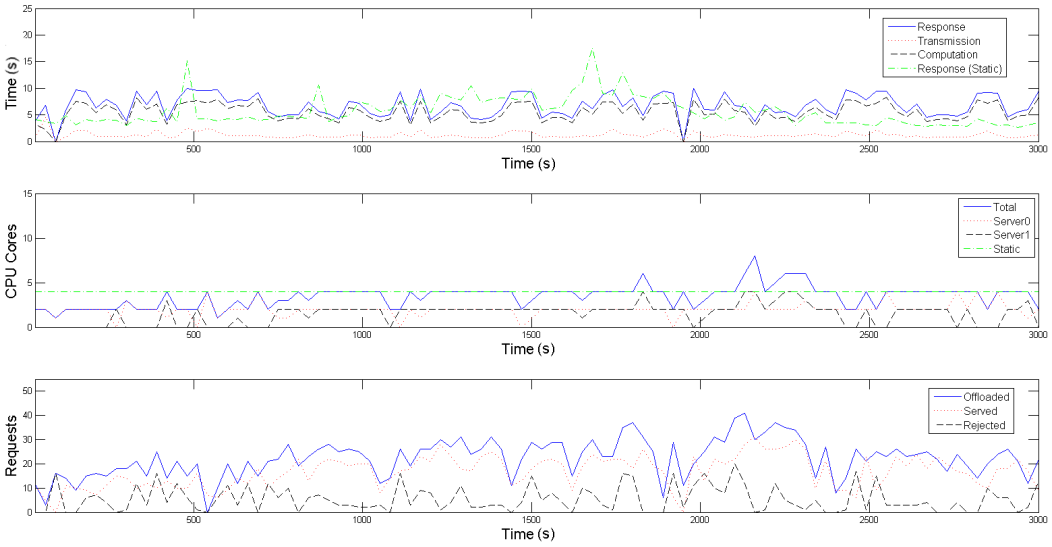


Figure 4.6: Infrared Tensorflow application.

Table 4.1: Server operating points.

	TensorFlow	**Conventional**	**TensorFlow**	**Infrared**
Cores	**X (s)**	**U (reqs)**	**X (s)**	**U (reqs)**
1	3.0	4.2417	3.5	4.9487
2	3.0	14.2357	3.5	16.6083
3	3.0	17.1731	3.5	20.0353
4	3.0	18.4604	3.5	21.5371

With respect to the Vertical Scaling part of SMOKE, the admission control process, executed on each Local Controller of each server, results in the rejection of approximately 19.58% of the offloaded requests for the conventional image recognition application and 23.01% for the infrared one (black-colored dashed line in each third diagram of Figures 4.5 and 4.6). This is a consequence of the real incoming workload of the interval exceeding the projected one. Although the rejected volume is not negligible, it is not detrimental to the event detection precision. As a reminder, both these scaling processes aim to maintain

the average response time below an acceptable value, $Tref$, which is an application-specific value. To achieve this, the reference input x_e of each operating point is empirically set near to $\frac{Tref}{3}$. In this scenario, $Tref$ is set equal to 10 *sec* for both applications, thus the reference inputs of the operating points in Table 4.1 are set to 3 *sec* and 3.5 *sec* respectively. This goal is achieved, as depicted with the blue-colored solid line in the first diagrams in both Figures 4.5 and 4.6; the response time remains within the limit of 10 *sec*, despite the workload fluctuations. In these diagrams, average transmission and computation times are also plotted with red-colored, dotted and black-colored, dashed lines respectively; one can easily note that the average response time follows the same patterns as the average computational time, meaning that it is mainly affected by it. The average transmission time is negligible due to the use of the IEEE 802.11ac standard, which provides high throughput for the images (size of about 5 MB) used in this experiment. Finally, the vertical scaling of SMOKE is compared to a static allocation of four cores for each application; the green-colored, dot dashed line on each first subfigure denotes the average response time of this static allocation, while on each second the total cores statically allocated. One can easily observe that when the experiment is on the tracking mode, it demonstrates better average response times for both applications, however, it still suffers from the phenomenon of overprovisioning; that is when a Container uses resources, that could be allocated other processes, without significant benefits. On the other hand, when the experiment enters the emergency mode, there are times when 4 statically allocated cores are inadequate for the processing requirements, resulting in violation of the 10 *sec* limit for the average application response time. This problem of providing less than the necessary resources is called underprovisioning and it potentially puts the mission's accuracy into risk.

4.6.2 IDM Evaluation

As stated in Section 4.5.2, the socially-aware Intelligent Decision Making process is triggered when the SMOKE component calculates an average classification score of the images offloaded in the last time interval that is associated with a high probability of fire incident detection. In this subsection, the evaluation results of the IDM component are presented focusing on the overall time overhead that is imposed until a final fire detection decision is reached. To this end, several experiments have been executed in order to identify the

average time delays imposed by the various individual steps of the IDM algorithm. These steps are the following:

- Step A: Reverse geocoding process of the provided GPS coordinates, which is based on the call of external APIs (e.g., Google Maps API) in order to obtain a set of area names and location identifying keywords.

- Step B: Extraction of Fire Weather Index extracted from the GeoTIFF image based on the provided GPS coordinates.

- Step C: Collection and analysis of Twitter posts that are related with the indicated area.

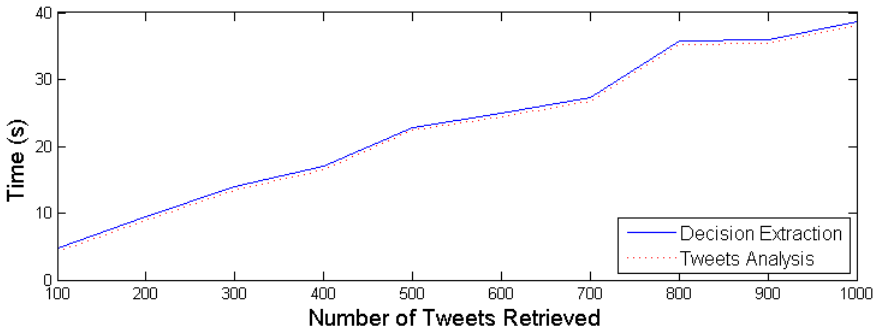


Figure 4.7: Decision extraction in relation to amount of Tweets retrieved.

After executing several experiments, the results obtained indicate that the average time required for Step A is 0.42 *sec*, while the average time needed for Step B is 0.08 *sec*. As one can easily observe, the introduced time overhead is minimal. On the other hand, Twitter data retrieval and Tweet processing may introduce a significant time overhead that depends heavily on the overall number of Tweets that comply with the retrieval criteria (e.g., Twitter posts containing tags and keywords related with the area and a forest-fire incident). Figure 4.7 illustrates the aggregated time needed for the IDM to perform steps A, B and C for increasing volumes of retrieved Twitter data. In order to estimate the expected volume of Twitter data that is generated during wild-fire incidents, a thorough review of existing approaches has been conducted and an analysis of the respective datasets indicating the evolution of the Tweet posts upon fire incident occurrence. The results, presented in Table 4.2,

show that the total number of Tweets are not more than 10.000 for the entire duration of the wild-fire incident. In the discussed case, the Twitter stream API is utilised and the retrieval and processing of Tweets is taking place in an online mode aiming to detect the incident as soon as possible. Hence, it is safe to assume a few hundred Tweets to be generated within the first minutes of the wild-fire ignition. Therefore, given the results captured in Figure 4.7, the overall time overhead imposed by the social media mining process lies between 5 and 40 *sec*, which enables the proposed IDM to reach a decision in a timely manner.

All in all, these two proposed CPSS mechanisms work seamlessly and efficiently towards the time- and mission-critical application of the early forest fire detection. The SMOKE framework manages to alleviate the IoT nodes' computational workload, timely dictate their operating mode and orchestrate their new formation, should it be deemed necessary, while the servers' resources are optimally used. Subsequently, the IDM is fed with the needed information that enables it to communicate the decision promptly to the public authorities.

Table 4.2: A review of wildfire incidents and fire-related Tweets volumes.

Year	**Country**	**Incident Location**	**Duration (Days)**	**Tweets**	**Ref.**
2012	USA	Colorado	32	4.2 K	[141]
2013	Australia	Australia	21	2.0 K	[141]
2014	Indonesia	Sumatra	92	9.7 K	[142]
2014	USA	San Marcos, Bernardo	9	1.3 K	[125]
2015	USA	California	52	1.9 K	[143]

Chapter 5

Switching Computational Offloading for Robotic Applications in Edge Computing

5.1 General Setting

In this chapter, the tradeoffs between computing and communication resources are investigated with a focus on control design, estimation, and implementation. Specific effort is placed on exploring the offloading opportunities of the decision-making and monitoring algorithms (control and estimation) in the path planning problem for autonomous agents. To this end, a control based, computation offloading mechanism for robotic applications in Edge Computing ecosystems is proposed [144]. In particular, an IoT-enabled localization and path planning framework is realized and the expected gains of computation offloading are verified by utilizing a real Edge Computing setting. To achieve this, local and remote localization and path planning controllers are designed and implemented, followed by a scheduling mechanism. The offloading mechanisms are treated as switches, leading to different dynamics of

the resulting closed-loop system.

In more detail, regarding the localization process, a decision algorithm is developed, that triggers an accurate image-based pose estimator in the cloud/edge, whenever the uncertainty of the robot's position and orientation becomes too large. Regarding the path planning part, a more accurate, however more computationally intensive decision algorithm, runs remotely in the edge/cloud, whenever a prediction cost indicates a possible amelioration of the current reference trajectory generated locally. Roughly, for both cases, the offloading decision takes into account the position of the robot, the network traffic and the available computation resources in the edge/cloud (estimated by a Kalman filter). These switches compose a switching system that is adaptive and can operate under different scenarios and applications. This architecture perspective, which constitutes the main contribution of this work, offers the proposed framework a degree of contextual awareness; that is the ability to sense and dynamically adapt to the robot's environment, implicitly enhancing to an extent the robustness of its operation, as well as improving the QoS of the supported applications.

5.2 Related Work

Many works have been exploring the benefits of that computational offloading provides to robotic applications [145]. Open challenges in this area throughout the literature are concerned with developing adaptive multi-robot/machine control, capturing, modelling, predicting and anticipating the agent's interactions and designing distributed control and path planning algorithms that deliver flexible and safe working environments. The offloading-based studies leverage the network and computing capabilities of edge servers to execute remotely navigation or localization algorithms. To begin with, the authors in [146] presented how two reference architecture concepts, namely Network Function Virtualization (NFV) and Multi-access Edge Computing (MEC), can be utilized on orchestrating network and computing resources for deploying robotic applications. Furthermore, they proposed an integration of a MEC architecture in an NFV environment. To demonstrate the benefits of this hybrid architecture, the coordination of a mobile robot swarm on two robotic applications was used.

Approaches similar to ours include [147], where gesture-based semaphore mirroring with

a humanoid robot is split to remotely and locally executed functionality; [148], in which the authors identify a three-layered environment (Robot, Edge and Cloud) to overcome the challenges of network limits in a Deep Robot Learning application and [149] where Dew Robotics is introduced; this concept posits that critical computations are executed locally so that the robot can always react properly, while less critical tasks are offloaded to the Fog and Cloud, so to exploit the larger availability in computing, storage, and power supply. In [150], finally, the authors utilize a Fog-Cloud infrastructure to alleviate the tasks of object detection, tracking and mapping in a confined area. In a different manner, the authors in [151] proposed a symbiotic robotic network for task offloading in the factory floor. Based on their vicinity, the robots formed clusters where members could offload tasks to each other. Additionally, a reward-based feedback task offloading mechanism was proposed to support delay-sensitive applications. Based on these rewards, each node had a social repute score which was used to select the appropriate node to offload the tasks and for the election of the cluster head. However, none of the aforementioned offloading decision schemes addresses the dynamic nature of the robot's environment.

5.3 Contribution & Outline

The scenario addressed in this work involves a mobile robot equipped with sensing, computing, and wireless communication capabilities, which makes its way from a starting position to a target position in an operating ground (e.g., a factory floor), navigating through obstacles. This functionality is a key component to realizing autonomous robotic navigation in Industry 4.0 use cases, e.g., warehousing and logistic robots which automate the process of storing and moving supply chain goods. Tracking the robot location is essential for a robust and safe trajectory planning. However, a common problem in such a scenario is that the uncertainty in estimating the exact pose (i.e.,position and orientation) grows over time in motion, due to inaccuracies in sensing, wheel slips, hardware failures, etc., [152]. Thus, the importance of an accurate, dynamically adjusted localization technique is evident. The key contributions of this work that differentiate it from the rest of the literature are summarized as follows:

1. A novel computation offloading mechanism for robotic applications that utilises an

Edge Computing setting to improve the accuracy of both the robot's localization and trajectory.

2. An offloading scheme, based on switched systems, that addresses the dynamic nature of the robot's movement and deals with the unpredictability in its exact pose over time.

3. An innovative position and orientation estimation component that achieves high precision while using the simplest camera system and the minimum amount of identified natural landmarks.

It should also be noted that the testbed of the proposed framework is a vehicular mobile robot development platform, called AlphaBot [153] (Figure 5.6), equipped with a single frontal pivoted camera and a Raspberry Pi 3 Model B+ as the control unit.

5.4 System Architecture

In the discussed setting, the image processing and decision algorithms become computationally intensive and energy-hungry tasks. The proposed framework realizes an IIoT-enabled assistive remote path planning mechanism, with the aim to find the expected gains of computation offloading in the edge and the cloud.

Specifically, self-localization through landmark assisted pose estimation is implemented; the robots are equipped with a camera module, while in their proximity unique cylindrical beacons are used as landmarks to assist in the pose estimation process. In the computationally demanding involved algorithms, two offloading opportunities are revealed in, namely, pose estimation and path planning. To this purpose, a small-scale network infrastructure is set up, connecting the robot to a wireless LAN (WLAN) through an Access Point located within the robots' network range, which in turn connects via a wired connection (LAN) to a server in the robot's proximity, the edge server.

Locally, the intangible assets include the (i) the Tracking Controller (TC), (ii) the Local Odometry-Based Estimator (LOE), (iii) the Local Beacon-Based Estimator (LBE), (iv) the Local Path Planner (LPP) and (v) the Offloading Decision Mechanism (ODM) components, all located within the robot; component (i) is responsible for carrying out movement-related

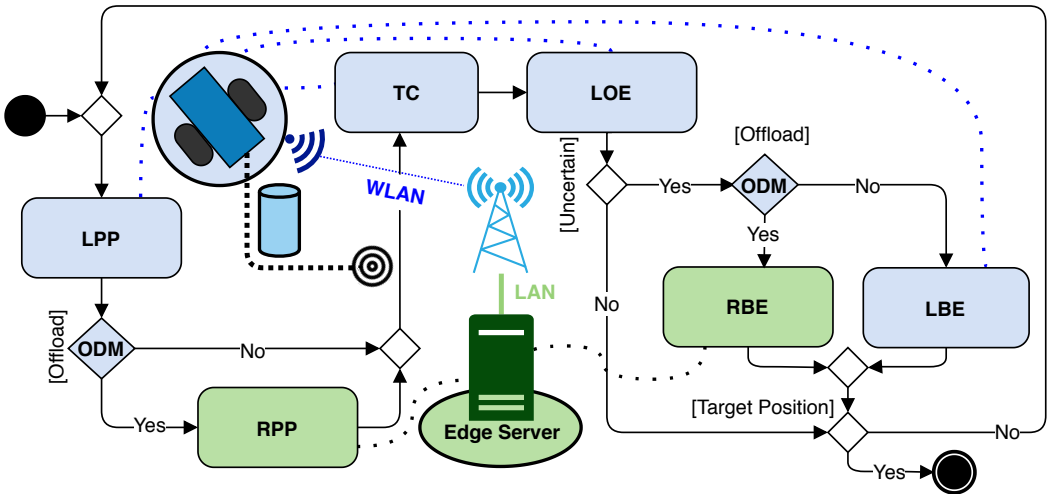


Figure 5.1: Architecture Overview. The locally executed components are highlighted with blue color, while the remotely executed ones with green.

decisions, (ii), (iii) and (iv) are the locally executed pose estimation and path planning applications respectively and (v) encompasses the intelligence of the proposed switching system by monitoring the offloading-related metrics and realizing the offloading decisions. On the remote side, containerized counterparts of the path planning and pose estimation applications are co-hosted on the edge server; these are namely (vi) the Remote Beacon-Based Estimator (RBE) and (vii) the Remote Path Planner (RPP) which are able to receive offloaded requests from the robot. A more detailed discussion on these components follows in Sections 5.5, 5.6 and 5.7.

In order to outline the sequence of interactions between the main components of the architecture, a representative scenario is showcased in which the proposed solution applies successfully. Figure 5.1 depicts an overview of this scenario. Without loss of generality, it is assumed that only one robot operates in the field. Also, its starting pose, the operating space dimensions and the obstacles' and beacons' positions and shapes are considered known a priori.

A typical activity flow of this scenario, initiates with Local Path Planner component calculating locally a trajectory from the starting position to the target position. This triggers the ODM for the first time; should a quick analysis on the projected trajectory indicate room for significant refinement of the selected path, the Remote Path Planner is invoked. This analysis is based on the trajectory curvature and the degree in which the more elegant

remote component is potentially able to smooth it around obstacles; Section 5.7.3 provides more insight on this process. Eventually, the resulted trajectory dictates the *intermediate positions* the robot needs to reach. In order to sequentially perform the transition to each of them, the Tracking Controller component is invoked.

After reaching the next position of its trajectory, an uncertainty indicator of the pose estimation is calculated; this indicator is a scalar that grows with time and actually accumulates the error between the estimated and the reference pose after each move, as explained thoroughly Section 5.7.1. Here, the second decision occurs; if this indicator measures bellow a predefined threshold, the robot continues to move based on the feedback coming from the Tracking Controller's monitoring process, i.e., the Local Odometry-Based Estimator, which leverages the robot's photoelectric sensors (encoders) attached to each wheel to measure the wheels' angular velocities during a period of time. Else, it invokes the more precise, but computationally heavy, Beacon-Based Pose Estimator, leveraging information coming from the beacons in the environment. That triggers the ODM once again; the edge server is queried to provide an estimation on the duration of the potentially offloaded pose estimation task. As described by the mathematical modelling in Section 5.7.2, this duration is proportional to the availability of the computational resources and is actually indicating the resources able to be dedicated for the execution of this task. Based on this estimated duration, a decision is made on whether to offload the pose estimation task to the Remote Beacon-Based Estimator, or execute it locally. The flow ends with the robot checking if the target position is reached. If not, it reverts to first step.

It is worth highlighting that the tracking controller, as well as the path planning and pose estimation are aperiodic. The position of the robot on the operating ground, is defined by the state vector $\boldsymbol{x^i} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$. The robot has to move towards the next reference position $\boldsymbol{x^i_{ref}} = [x_{1,\mathrm{ref}}(t_i) \quad x_{2,\mathrm{ref}}(t_i)]^\top$, generated by the path planning algorithms, to approach the target position. Figure 5.2 gives a brief insight on the timing sequence in which the rest of the sections will refer to. Let subscript i correspond to the step during which the robot reaches the next reference position in k_i actuation steps, while simultaneously tracking its pose. In particular, at time t_i^0 the robot is in the position x^i. When the next reference position x_{ref}^{i+1} is close, the uncertainty about the current estimation is calculated. Thus, the time duration T_i^1 corresponds to the time spent for localization. When the local odometry-based

estimator is used, this time is equal to zero, while the beacon-based estimation algorithm is time consuming. The time duration T_i^2 corresponds to the path planning algorithm running time either remote or local, which generates the next reference position. Similarly, the time to execute the local path planning algorithm is equal to zero.

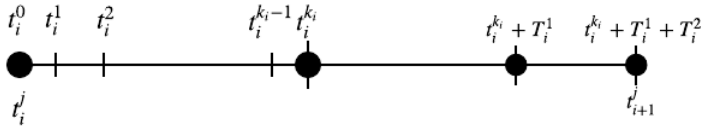


Figure 5.2: The timing sequence in the proposed scenario.

5.5 System Modeling

5.5.1 Robot dynamics

The differential-drive robot used in this study has two wheels that can turn at different rates, allowing motion by changing the orientation and the position (x_1, x_2) either separately or simultaneously. For the robot dynamics, the 2D coordinates, i.e.,position, and the orientation of the robot are denoted by the state variables z_1, z_2 and z_3. Hence, we consider $\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{x}^\top & \theta \end{bmatrix}^\top$. The robot is controlled by the angular velocities w_R and w_L, accounting for the right and left wheel respectively. The robot dynamics is defined by the following continuous time system, based on the work in [154], using the aforesaid state-space representation. Specifically, we have for any $t \geq 0$,

$$\dot{z}_1(t) = \frac{r}{2}(w_L(t) + w_R(t)) \cos z_3(t), \tag{5.1}$$

$$\dot{z}_2(t) = \frac{r}{2}(w_L(t) + w_R(t)) \sin z_3(t), \tag{5.2}$$

$$\dot{z}_3(t) = \frac{r}{l}(w_L(t) - w_R(t)), \tag{5.3}$$

where l, r are the distance between the two wheels and the radius of each wheel respectively. The odometry measurements $\tilde{w}_L(t_i^j)$,$\tilde{w}_R(t_i^j)$ are taken at each time instant t_i^j, $i = 0, 1, ...,$ $j = 0, \ldots, k_i$ of the timing sequence introduced in Section 5.4. The corresponding discretized

system using Euler forward method is:

$$\tilde{z}_1(t_i^{j+1}) = \frac{r}{2}(\tilde{w}_L(t_i^j) + \tilde{w}_R(t_i^j)) \cos \tilde{z}_3(t_i^j)(t_i^{j+1} - t_i^j) + \tilde{z}_1(t_i^j), \tag{5.4}$$

$$\tilde{z}_2(t_i^{j+1}) = \frac{r}{2}(\tilde{w}_L(t_i^j) + \tilde{w}_R(t_i^j)) \sin \tilde{z}_3(t_i^j)(t_i^{j+1} - t_i^j) + \tilde{z}_2(t_i^j), \tag{5.5}$$

$$\tilde{z}_3(t_i^{j+1}) = \frac{r}{l}(\tilde{w}_L(t_i^j) - \tilde{w}_R(t_i^j))(t_i^{j+1} - t_i^j) + \tilde{z}_3(t_i^j). \tag{5.6}$$

5.5.2 Tracking controller

As previously mentioned, the robot moves towards the next reference position $\boldsymbol{x_{ref}^i}$ to reach the target position. This transition is broken down into two parts: a rotational movement to the new orientation and a translational movement to the new position on the grid. For this actuation phase, given the specific robot dynamics, a tracking controller is proposed, which is executed locally on the robot, by fixing the control inputs w_L, w_R to be either equal or opposite. Therefore, the control input is w, while $|w| = |w_L| = |w_R|$. As a result, the motion of the robot is restricted to a straight line, i.e.,"translational motion", or a rotation around the center of the wheel axle, i.e.,"rotational motion", respectively. This control structure is chosen as it is efficient for tracking purposes, leading to a simple structure of the closed-loop system. Specifically, the closed-loop dynamics for the translational and rotational motion are

$$S_1^{Tran} : \begin{cases} \dot{z}_1(t) = \frac{r}{2}(w(t)) \cos z_3(t), \\ \dot{z}_2(t) = \frac{r}{2}(w(t)) \sin z_3(t), \\ \dot{z}_3(t) = 0 \end{cases} \tag{5.7}$$

$$S_2^{Rot} : \begin{cases} \dot{z}_1(t) = 0, \\ \dot{z}_2(t) = 0, \\ \dot{z}_3(t) = \frac{r}{l}(w(t)), \end{cases} \tag{5.8}$$

where S_1^{tran} is used for the translational motion and S_2^{rot} when the robot needs to rotate. Let $R(t_i^j) = \left\| \begin{bmatrix} \tilde{z}_1(t_i^j) \\ \tilde{z}_2(t_i^j) \end{bmatrix} - \begin{bmatrix} z_{1,\mathrm{ref}}(t_i) \\ z_{2,\mathrm{ref}}(t_i) \end{bmatrix} \right\|_2$ be the distance between the robot's current estimation and

the reference position and let $\phi(t_i^j) = \tilde{z}_3(t_i^j) - \tan^{-1}\left(\frac{\tilde{z}_2(t_i^j) - z_{2,\mathrm{ref}}(t_i)}{\tilde{z}_1(t_i^j) - z_{1,\mathrm{ref}}(t_i)}\right)$ be the angle between the robot's current estimation of orientation and the line connecting the robot and the reference position. Here, $\tilde{z}$ accounts for the estimation of its current pose calculated by Equations (5.4) – (5.6) at the time period of the actuation $t = t_i^j$, $j = 0, 1, \ldots, k_i$.

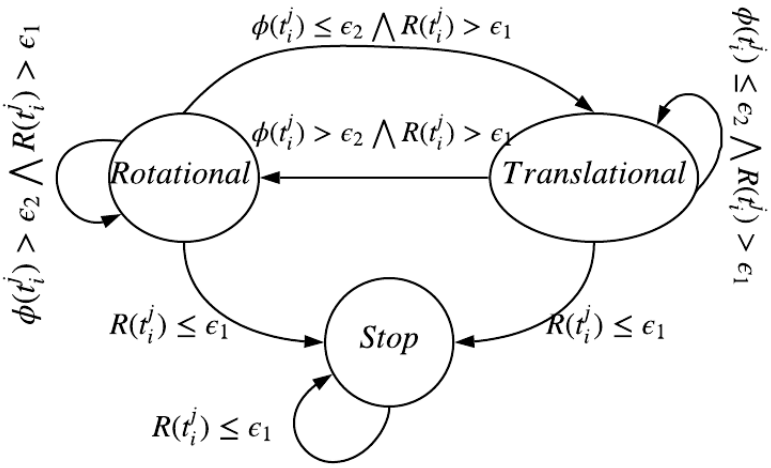


Figure 5.3: The hybrid automaton of the proposed system.

The closed-loop system with the tracking controller can be modeled by a discrete-event system, see, e.g., [155], as shown in Figure 5.3, where the control input can be calculated as follows:

$$w(t_i^j) = \begin{cases} L_1 R(t_i^j), & \phi(t_i^j) \leq \epsilon_2 \bigwedge R(t_i^j) > \epsilon_1, \text{Translational}, \\ L_2 \phi(t_i^j), & \phi(t_i^j) > \epsilon_2 \bigwedge R(t_i^j) > \epsilon_1, \text{Rotational}, \\ 0, & R(t_i^j) \leq \epsilon_1, \qquad\qquad \text{Stop}. \end{cases}$$

The quantities ϵ_1, ϵ_2 are positive constants, while the gains L_1, L_2 are constant control parameters.

The reference position is reached when the estimation of its position is close, and in particular is inside a ball of radius ϵ_1 close to the reference, i.e., centered at $\mathbf{B}(\boldsymbol{x}_{\boldsymbol{ref}}^{\boldsymbol{i}}, z(t_i^j)) = \{z \in \mathbb{R}^3 : \|z - \tilde{z}(t_i^j)\| \leq \epsilon_1\}$. The effect of the uncertainty is taken into account explicitly in the offloading decision, as it will be explained in section 5.7.

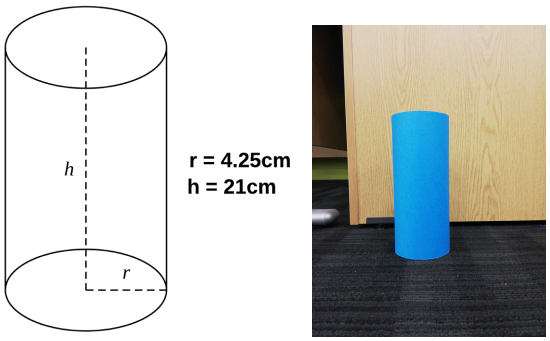
5.6 Localization and Path Planning

In what follows, the algorithms chosen for localization and path planning are presented, with a varying degree of complexity and accuracy, that are implemented locally and remotely accordingly.

5.6.1 Localization

The localization problem is equivalent to the pose estimation problem in the discussed setting. Two algorithms of different complexity are implemented, namely, (i) an odometry-based one, and (ii) a camera-based estimation. The first estimation algorithm is light enough to run efficiently on the robotic platform. Roughly, the robot's on-board wheel encoders readings are fed to the motion model of (5.4) – (5.6). While this is a lightweight and fairly accurate localization technique when it comes to short trajectories, odometry is known to be prone to accumulative errors [156].

The second localization technique is the computationally heavier beacon-based estimator that was also presented in [157]. Roughly, the technique is based on a bilateration method using principles of the projective geometry. Distance calculation is based on feature extraction from pictures depicting the landmarks, with the localization algorithm relying on minimum two strategically positioned landmarks. To address this requirement, the attached camera scans the area in front of the robot, capturing pictures and analysing them until two landmarks are detected. Hence, computationally intensive, real time image processing is required to achieve highly accurate results. Relevant works include [158] and [159].



(a) Beacon Schematic (b) Actual Beacon

Figure 5.4: Landmarks

In more detail, in order to estimate the AlphaBot's 3-degrees-of-freedom pose in the grid, in terms of (x, y) coordinates and θ heading orientation, a single vision -based self-localization algorithm was developed. This algorithm is classified as single vision -based, because the robot uses a single frontal camera. Moreover, it is landmark-assisted, since it requires a single landmark feature (width of beacon) to work. The approach taken here is feature-based and relies on the principles of projective geometry. In particular, the cylinder was selected as the shape of the Beacons (Figure 5.4), because of its interesting property; its 2D projection is a rectangle, independently of any viewing angle that has a rotation axis parallel to the cylinders axis. The Beacons are colored differently from the environment colors, e.g., blue, red, orange and green, in order to facilitate the detection from the AlphaBot's camera. The camera mounted on the AlphaBot has a 30-degree horizontal angle of view, so a rotation of six 30-degree steps is needed in order to cover the 180-degree-area in front of it, on aggregate, and detect two, at least, Beacons. This requirement is discussed further on Subsection 5.6.1. The following three steps are performed:

Beacon Recognition

In order to detect the presence of a Beacon within an image, the Python OpenCV Library[1] was utilized; first, the image is transferred to the HSV (Hue, Saturation and Value) color space, because this conversion is robust towards external lighting changes. In particular, in cases of minor changes in external lighting, such as pale shadows, Hue values vary relatively less than RGB values. After this, the algorithm applies an offline calculated HSV mask to the image, acting as a color filter for each of the Beacon colors. Then, it groups the adjacent filtered pixels and draws the minimum-area rectangles that surround each of these groups. This mask consists of a set of lower and upper values regarding the Hue, Saturation and Value of each color, acting as boundaries. In this study, the following ranges where used: $H \in [0^o, 180^o]$, $S \in [0, 255]$ and $V \in [0, 255]$. For example, the $[H, S, V]$ mask corresponding to blue colored pixels is: $lower[30, 75, 100]$ and $upper[110, 255, 255]$.

Next, from the rectangles drawn on the image, the ones that possess the following features are considered to be classifiable as a Beacon:

1. The identified rectangle is in upright position.

[1] https://github.com/opencv/opencv

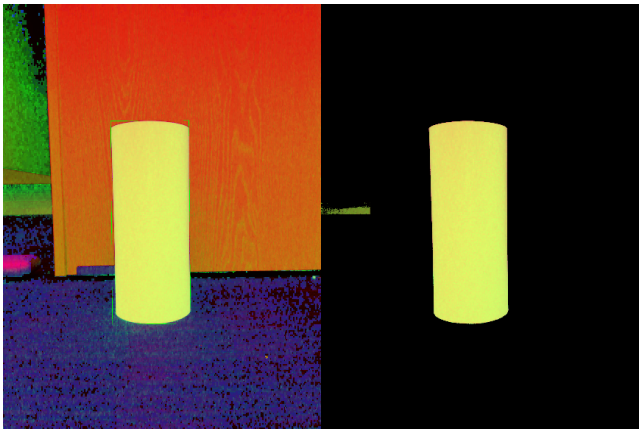


Figure 5.5: Beacon contour detection in HSV color space.

2. Its shorter side is parallel to the x axis and its longer side parallel to the y axis of the image plane.

3. Its aspect ratio stays between Beacon-specific, predefined, boundaries.

These simple criteria are defined to filter out objects on the field that are similarly shaped and colored as the Beacons. If no rectangle fits these criteria, then it is assumed that the image does not depict a Beacon in whole. On the other hand, if more than one Beacons are detected, a selection is made to consider only one of them. Figure 5.5 depicts the result of the above process. The resulting information retrieved is the perceived width p of the contour rectangle surrounding the Beacon, in pixels.

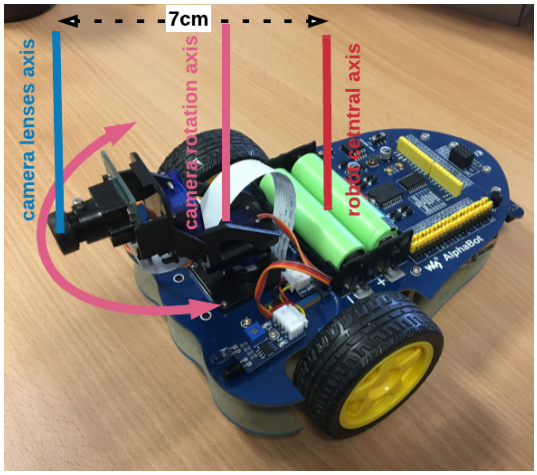


Figure 5.6: AlphaBot's main axes.

Distance and Angle Relative to Beacon Estimation

After detecting a Beacon within an image, the process of estimating the AlphaBot's distance and angle from this Beacon, takes place. Performing this step twice, for two distinct Beacons, allows the deduction of the exact position and pose of the AlphaBot on the grid. It is worth noting that in the scope of this work, the positions of the Beacons are assumed to be known. However, this method is extensible to the case where the positions of the Beacons is unknown, e.g., where a Simultaneous Localization and Mapping (SLAM) technique can be leveraged to initially identify these positions. The camera mounted on the AlphaBot follows the pinhole camera model [160]; that means that the relative size of the projected objects depends on their distance to the focal point. To find the distance, the *triangle similarity theorem* is utilised, i.e., the distance of the object to the camera, d_c, is given from the following equation:

$$d_c = \frac{wf}{p}$$

where:

w = Beacon width in cm.

f = camera's focal length in mm (known from camera's datasheet or computed through camera calibration).

p = perceived Beacon width in pixels (px).

It should be noted here that across the localization process the AlphaBot is considered to be a dimensionless point on the center of its wheel axis. However, the AlphaBot's camera lenses axis is placed $7cm$ from the robot's center, as shown in Figure 5.6. Thus, the actual estimated distance, d, between the Beacon and the AlphaBot is:

$$d = d_c + 7$$

The core novelty of the proposed localization method lies in the calculation of the angle between the AlphaBot and the Beacon. As shown in Figure 5.7, the 2D projection of the Beacon is assumed on the plane of the captured image. It is also assumed that the origin

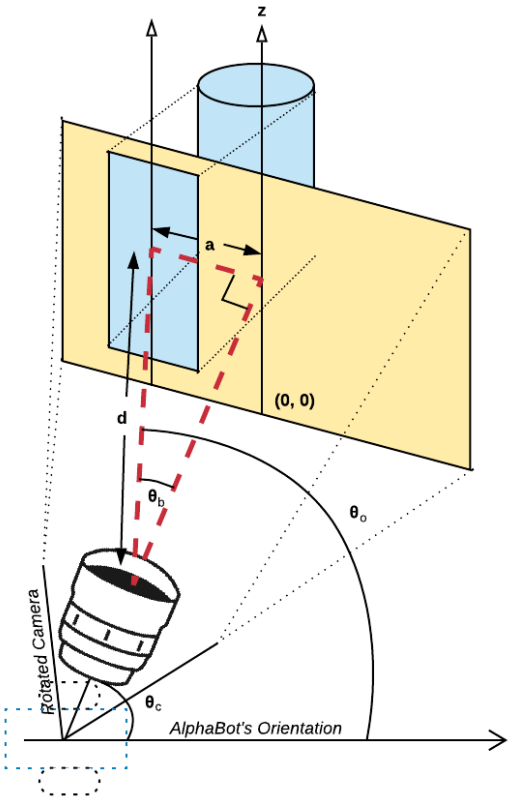


Figure 5.7: Method for calculating the angle between the AlphaBot and the Beacon.

$(0, 0)$ is at the middle of the bottom border of the image plane and the axis z, coming through it, as shown in Figure 5.7. To calculate the angle θ_b between the camera's line of sight and the line which starts from the camera lens and is perpendicular to the Beacon's axis, d and a are required. An insight of this angle's real-world nature would be this: "the angle that the camera has to rotate to horizontally centre the Beacon's 2D projection on the image plane". The distance a_p, in pixels, is the perpendicular distance between the axis z and the Beacon's axis, which are parallel to each other, and can be readily calculated as the contour's vertices coordinates are known from the last step. The distance between the Beacon's axis and the AlphaBot, d, in cm, was calculated in the previous step as well. Hence, it is only needed to translate the distance a_p to the distance a in cm. To enable this conversion, it is first ensured that the cm-per-pixel ratio, which applies to the Beacon's 2D projection on the image plane, is preserved throughout the rest of the plane. This holds true, as the real-world z axis and its projection on the image plane coincide, which subsequently means that the real-world distance a and its projection coincide as well. Moreover, the pixel

size has the same cm length, independently of the pixel's position through the camera's conformity with the pinhole model, which makes the projection free of any linear distortion. Consequently, we have

$$a = a_p(\frac{w}{p}),$$

where $\frac{w}{p}$ is equivalent to the cm-per-pixel ratio. With a and d known, we can calculate θ_b,

$$\theta_b = arcsin(\frac{a}{d}).$$

The approach still works when other objects, such as obstacles in the environment, are depicted in the projection. The only constraint is that the Beacon has to be captured in whole. The last thing to note is that, as mentioned earlier, the camera rotates on its pivoted system in order to scan the area in front of the robot for Beacons. However, the angle θ_c to which the camera is rotated is known. As a result, the overall θ_o to which the AlphaBot is rotated, with the given Beacon as reference, is

$$\theta_o = \theta_c + \theta_b.$$

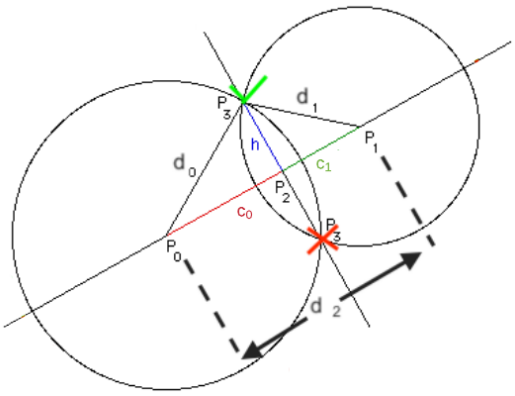


Figure 5.8: Bilateration method for calculating AlphaBot's position in the grid.

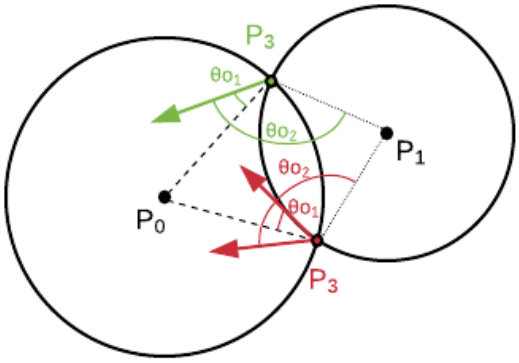


Figure 5.9: Method for invalidating one of the two possible solutions.

Grid Position and Orientation (Pose) Estimation

After having the distance, (d_0, d_1), and angle $(\theta_{o_0}, \theta_{o_1})$ from two Beacons available, the AlphaBot's pose in terms of position and orientation is estimated. The locus formed by the set of possible (x, y) locations whose distance from Beacon P_i equals the estimated distance $d_i, i \in [0, 1]$, is a circle. This observation allows for utilizing the bilateration method in order to estimate the position of the AlphaBot, as shown in Figure 5.8 with P_3; this method has been used extensively in previous works regarding localization in wireless sensor networks [161], as it requires much lower computational complexity, yet still retains the same localization accuracy, if the environmental setup allows it.

In the discussed setting, a unique solution of the location of the AlphaBot is feasible to be retrieved by combining the knowledge of the relative angle observations θ_{o_0} and θ_{o_1}. Indeed, for the two (at most) candidate locations that the observations were taken as shown in Figure 5.8, there is always exactly one feasible configuration that allows both angle values to be attained, or equivalently, that result in the same absolute angle θ estimation. This is demonstrated in Figure 5.9; for example, let $\theta_{o_1} = 20°$ and $\theta_{o_2} = 150°$. As shown there is only one feasible point where both angle measurements are verified.

The developed method requires only two beacons for localization, under the assumption of course that the measurements are accurate. Nevertheless, the problem of placing the minimal number of landmarks in the map still remains. This number depends on the viewing angle of the camera and the density of the obstacles, or equivalently, visibility of the beacons from all directions. For the discussed setting where the viewing angle is 180^o and for a

rectangular map with obstacles not obstructing the visibility of the beacons, the minimum number of beacons need to be placed is 4. The problem becomes significantly harder for non-convex and/or non-static maps and tall obstacles and it is deferred for future research.

The bilateration process is briefly depicted in Figure 5.8; the mathematical justification behind calculating the circles' intersection points is the following; considering the two triangles $P_0P_2P_3$ and $P_1P_2P_3$ we can write

$$c_0^2 + h^2 = {d_0}^2$$

and

$$c_1^2 + h^2 = {d_1}^2$$

where c_0 and c_1 are the distances of P_0 and P_1, respectively, from the bisector coming through the two intersection points of the circles and $c_0 + c_1$ equals the distance d_2 between the two Beacons. Using $d_2 = c_0 + c_1$ we can solve for c_0,

$$c_0 = \frac{{d_0}^2 - {d_1}^2 + {d_2}^2}{2d_2}$$

Then we solve for h by substituting c_0 into the first equation, $h^2 = {d_0}^2 - c_0^2$, so we get

$$P_2 = \frac{P_0 + c_0(P_1 - P_0)}{d_2}$$

And finally, $P_3 = (x_3, y_3)$ in terms of $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, is either

$$x_3 = \frac{x_2 + h(y_1 - y_0)}{d_2}, y_3 = \frac{y_2 + h(x_1 - x_0)}{d_2}$$

or

$$x_3 = \frac{x_2 - h(y_1 - y_0)}{d_2}, y_3 = \frac{y_2 - h(x_1 - x_0)}{d_2}$$

As mentioned above, one of the two solutions is always rejected as invalid.

The final part of the localization process is to calculate the AlphaBot's orientation in the grid with respect to a given reference point. As a first step, a Reference Point in a known

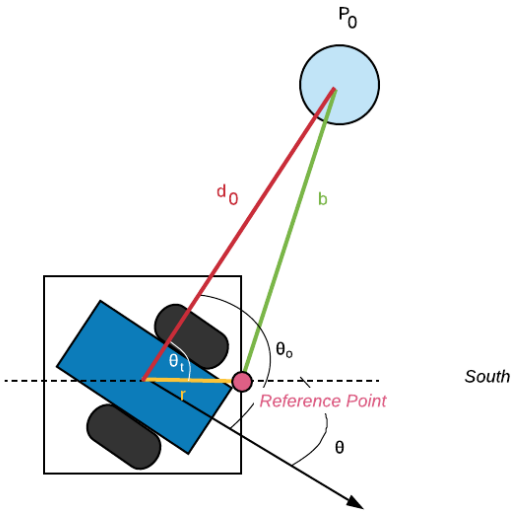


Figure 5.10: Method for calculating the angle between the AlphaBot and the Beacon.

location on the AlphaBot's South is asssumed, in Cartesian coordinates; as shown in Figure 5.10, the exact locations of both the Beacon (P_0) and the AlphaBot are known by now, thus calculating the distances b (Beacon - Reference Point) and r (AlphaBot - Reference Point) is straightforward. Also, distance d_0 (Beacon - AlphaBot) and angle θ_o (AlphaBot's angle with Beacon as reference) have been calculated in the previous steps; hence, by utilizing the cosine rule, the θ_t angle can be obtained:

$$\theta_t = arccos(\frac{b^2 + r^2 - {d_0}^2}{2br})$$

The actual orientation angle, θ is given from the following subtraction:

$$\theta = \theta_t - \theta_o$$

5.6.2 Path Planning

Many works exist in the literature addressing the path planning problem; a realistic robot navigation and smooth trajectory planning is a major challenge [162], [163]. Planning algorithms generate a trajectory consisting of intermediate reference positions to reach the final target position. In this work, graph-based methods of varying complexity are selected and adapted, see, e.g., [154, Chapter 8]. As a result, the algorithms described below, take as

input a graph that represents the real-space grid space along with the target positions, the obstacles and the starting position. This grid has a predefined cell size, that depends on the length of the robot. Each cell corresponds to a possible reference position. In this case, the obstacles are rectangular-shaped, for the sake of simplicity, however, arbitrarily-shaped obstacles could also be included.

On the one hand, a lightweight implementation of the $A^\star$ algorithm [164] acts as the Local Path Planner. Similar to [165], four directions of movement are allowed in the grid. The cells containing obstacles are not connected with the neighboring cells. The $A^\star$ algorithm returns a sequence of positions to reach the target position, according to a heuristic cost function; in this case this is the Manhattan Distance. The implementation is suitable for a robot with minimal computational resources providing a solid and quick solution, however the generated trajectory is not smooth.

The computationally intensive algorithm acting as the Remote Path Planner is deployed on the edge server. Similar to [166], the main process of the proposed algorithm is to locate a possible move towards a node that is closer to the target given the aforementioned graph. For this purpose, a multiple sources single destination problem is solved, utilising Dijkstra's shortest path algorithm, which calculates a path from each node towards the target position, offline. These precalculated paths, along with the total cost to reach the desired destination, are stored in a database on server's startup. When the Remote Path Planner is invoked, given the current location of the robot, a neighbour pruning is performed similar to [167]. A node of the graph is considered to be a neighbor of the current position if (i) the distance between them is less than twice the specified cell size and (ii) no obstacle is in the line of sight of the current position to that node. Consequently, to retrieve the set of possible neighbours, it is sufficient to search for avoidance of line clipping (intersection) between the line connecting the current position to each of the adjacent cells and the set of obstacles present in the real-space grid. The optimal path is chosen by comparing all possible neighbours. In particular, the cost to reach each one of them from the current position is added to the cost from each neighbour to reach the desired target. In this way, the algorithm allows "shortcuts" to the neighbouring nodes, while any-angle trajectories are feasible.

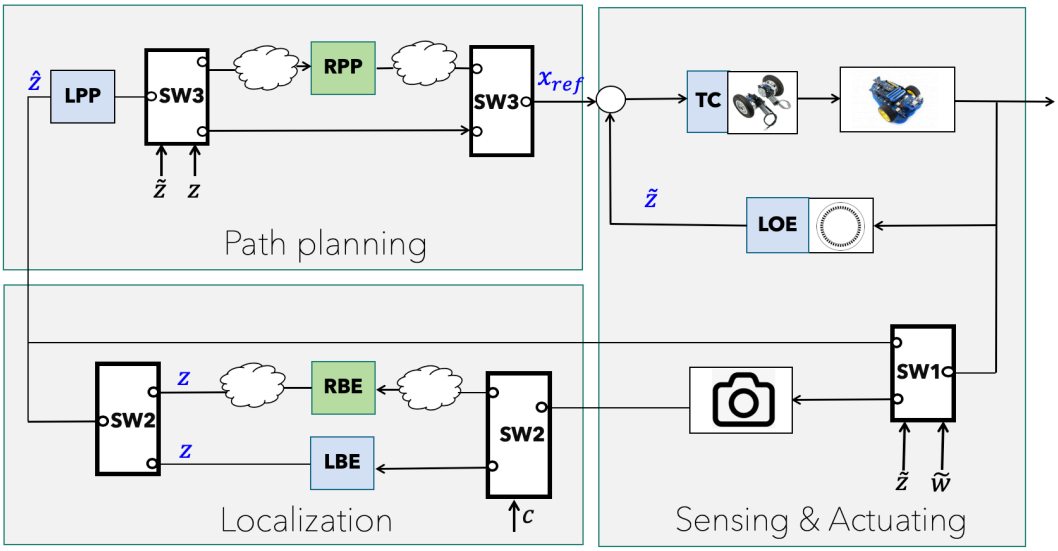


Figure 5.11: The block diagram of the switching system. Component abbreviations and colors follow the pattern introduced in Section 5.4.

5.7 Switching System

In this section, the switching mechanisms that are realizing the Offloading Decision Mechanism of the proposed framework are presented. It is assumed that starting from a position $\boldsymbol{x_0} = [x_1(0) \quad x_2(0)]^\top$, the closed-loop system converges asymptotically to a reference position $\boldsymbol{x_{ref}} = [x_{1,\text{ref}}(t_i) \quad x_{2,\text{ref}}(t_i)]^\top$ when exact measurements are available, i.e., when $\tilde{\boldsymbol{z}}(t) = \boldsymbol{z}(t)$. Two offloading opportunities are identified, related to the pose estimation and the path planning problem. In Figure 5.11 the proposed switching system is presented. In particular, switches $\mathcal{S}_1$ and $\mathcal{S}_2$ relate to the estimation procedure, and switch $\mathcal{S}_3$ concerns path planning part.

5.7.1 Sensor selection (Switch 1)

The measurement of the encoder is not perfect, while the model does not capture exactly the system behaviour. Consequently, there is an accumulating error between the state and its estimation. This error is modeled by a simple linear update mechanism. When the error becomes too large, the more precise, yet more computationally intensive remote localization algorithm is invoked. In order to decide when to offload, the variable $\delta(\cdot)$ is introduced that

describes the uncertainty in estimation. We set

$$\delta(t_i^{j+1}) = \delta(t_i^j) + b_0 + b_1\tilde{\delta}(t_i^j),$$

$j = 1, \ldots, k_i$, $i \in \mathcal{N}$, where $\tilde{\delta}$ is the deviation between the measurements of the states $\tilde{z}$, computed by the Equations (5.4) – (5.6) and the model-based estimations $\breve{\boldsymbol{z}}$, i.e.,

$$\tilde{\delta}(t_i^j) = \left\| \breve{\boldsymbol{z}}(t_i^j) - \tilde{\boldsymbol{z}}(t_i^j) \right\|_2,$$

where $\breve{\boldsymbol{z}}(t_i^j)$ consists of:

$$\begin{aligned}
\breve{z}_1(t_i^{j+1}) &= \frac{r}{2}(w_L(t_i^j) + w_R(t_i^j)) \cos \breve{z}_3(t_i^j)(t_i^{j+1} - t_i^j) + \breve{z}_1(t_i^j), \\
\breve{z}_2(t_i^{j+1}) &= \frac{r}{2}(w_L(t_i^j) + w_R(t_i^j)) \sin \breve{z}_3(t_i^j)(t_i^{j+1} - t_i^j) + \breve{z}_2(t_i^j), \\
\breve{z}_3(t_i^{j+1}) &= \frac{r}{l}(w_L(t_i^j) - w_R(t_i^j))(t_i^{j+1} - t_i^j) + \breve{z}_3(t_i^j),
\end{aligned}$$

which are the model-based estimation of the dynamics at time instants t_i^j, $j = 1, \ldots, k_i$ and w_L,w_R are the outputs of the tracking controller. At time t_0^0, the model-based estimation is equal to a known initial position, i.e., $\breve{\boldsymbol{z}}_1(t_0^0) = \breve{\boldsymbol{z}}_1^0$. As a result, δ linearly depends on the deviation, and is getting bigger as the robot actuates, especially when the actual motion of the robot differs from what the model dictates.

The offloading mechanism, aiming to reset the uncertainty, is triggered when δ becomes too large, namely larger than a prespecified threshold $\delta^\star$, i.e.,

$$S_1(t_i^{k_i}) = \begin{cases} \text{OFF}, & \text{if } \delta(t_i^{k_i}) \leq \delta^\star, \\ \text{ON}, & \text{else}, \end{cases}$$

where k_i refers to the time instant, when the robot's position, calculated by Equations (5.4) and (5.5), is close to the next reference position $\boldsymbol{x}_{\mathbf{ref},\boldsymbol{k}}$. Moreover, ON corresponds to using the beacon-based localization and OFF to proceeding based on the local odometry estimation. In the scope of this work, it is assumed that the uncertainty becomes equal to zero when the beacon-based localization is used. Hence, when $S_1(t_i^{k_i}) = \text{ON}$, then $\delta(t_{i+1}^0) =$

0, which means we get a valid measurement of the states $\boldsymbol{z}$. Otherwise, $\delta(t_{i+1}^0) = \delta(t_i^{k_j})$.

5.7.2 Estimation Offloading (Switch 2)

Switch S_2 decides whether the localization algorithm will be executed locally on the microcontroller mounted on the robot, or remotely on the edge server. The decision is based on the availability of the remote computing resources. Although the execution of such a computationally heavy algorithm on a battery-powered IoT device is energy-consuming, it may be preferable in some cases as offloading might result to larger response times due to lack of available resources on the remote server and network congestion.

Resource modelling and estimation

It is assumed that the resources of the localization service on the edge server are managed by the resource orchestrator of the infrastructure provider and the allocated resources can only be estimated through measurements. Thus, the resource allocation strategy on the edge server are modeled as a linear dynamical system subject to process and measurements uncertainty disturbances

$$c((k+1)T_s) = c(kT_s) + w(kT_s),$$
$$z(kT_s) = c(kT_s) + v(kT_s),$$

where c accounts for the virtual CPU cores of the container, z is the measurement of c and T_s is a constant sampling time. The terms w, v are the process and measurement noise respectively, both following a normal distribution. Based on previous measurements, a current estimation of the virtual CPU cores allocated to the container is computed, $\hat{c}$, by applying a Kalman Filter [168], which is a computationally light prediction method.

Processing time estimation

Having acquired the estimation of the available remote virtual CPU cores $\hat{c}$, the estimated processing time of the beacon-based localization algorithm can be calculated. To this purpose, the processing time, t_p is modeled as a linear relationship of the available resources, $t_p = a\hat{c} + b$. The coefficients a,b are calculated using the least squares fitting method, on a

set of pairs $(t_p, \hat{c})$ produced offline while experimenting with a dataset of pictures.

Regarding the wireless network transmission delay, the assumption is made that the wireless access technique between the robot and the access point is based on IEEE 802.11g. In this network deployment, a common effect that occurs when a signal travels through a communication channel is its power level decreases as the distance increases. To estimate this propagation loss, the well-accepted Log-Distance Path Loss (LDPL) model is utilized [169]. The LDPL model applies to indoor environments with the presence of obstacles, having a propagation exponent that indicates whether the environment has more or fewer obstacles, impacting on the computed loss. The respective path-loss is calculated as follows:

$$PL(d)_{dB} = PL(d_0)_{dB} + 10nlog_{10}(\frac{d}{d_0}),\ d \geq d_0, \tag{5.9}$$

where $PL(d_0)_{dB}$ is the path-loss at a reference distance $d_0 = 1m$, n is the path-loss exponent (PLE), which depends on the presence of obstacles in the environment. To set the upper bounds of the channel capacity the signal-to-noise-ratio (SNR) is leveraged metric,

$$SNR(d) = P_{dB} - PL(d)_{dB} - N_{dB}, \tag{5.10}$$

where P_{dB} is the incoming signal to the access point and N_{dB} is a Gaussian noise. Then, the channel capacity C can be calculated using the Shannon–Hartley theorem,

$$C(d) = B\ log_2(1 + SNR(d)), \tag{5.11}$$

where B is the available WLAN bandwidth (in Hz), giving in this way an estimation of the tightest upper bound on the information rate of data (in bits per second) that can be communicated at an arbitrarily low error rate using SNR. Having this bound available, an estimation of the task transmission duration (in seconds) can be calculated as follows:

$$t_{\text{net}}(d) = \frac{8\ m}{C(d)}, \tag{5.12}$$

where m is the size of the offloaded data in bytes.

Localization Offloading

The processing time is related directly to the CPU availability. The local beacon-based localization has an average time t_{loc} to be executed based on the robot's resources. Hence, Switch S_2 is formulated as:

$$S_2(t_i^{k_i}) = \begin{cases} \text{ON}, & \text{if } t_p + t_{net} \leq t_{loc}, \\ \text{OFF}, & \text{else}, \end{cases}$$

where k_i refers to the time instant that the robot must decide whether to offload or not the beacon-based localization algorithm. Moreover, ON corresponds to the remote execution of the self-localization algorithm and OFF to the local execution.

5.7.3 Path Planning Offloading (Switch 3)

Two path planning algorithms are implemented. By default, the computationally light $A^\star$ algorithm presented in Section 5.6.2, provides a reference trajectory on the robot. However, whenever a prediction cost indicates a possible amelioration by choosing a more refined path, the remote path planning algorithm is invoked. Both algorithms take as input the current estimation of the position and the reference position and generate a reference trajectory.

The offloading decision for the path planning depends on a cost consisting of two parts; the first part estimates the closeness of the generated reference trajectory to obstacles and the second part evaluates the curvature of the trajectory. Both terms follow theoretical aspects from standard works, e.g., [170]. The function $D(\boldsymbol{x})$ is defined, that quantifies the "density" of obstacles according to the estimation of the current position $\hat{\boldsymbol{x}}$, either computed by the beacon-based localization or the local odometry measurements.

$$D(\boldsymbol{x}) = \sum_{\hat{\boldsymbol{x}}_{\mathbf{obs}} \in \mathcal{X}_{\text{obs}}} \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}_{\mathbf{obs}}\|\right),$$

and $\boldsymbol{\mathcal{X}}_{\mathbf{obs}}$ is the set of positions that correspond to the centers of the cells that are unreachable, e.g., occupied by an obstacle.

Let $\{\check{\boldsymbol{x}}(i)\}_{i=1,\ldots,M}$ be the part of the path sequence consisting of the first M positions, generated by the local path planning algorithm.

The local path planning algorithm takes as input the current position estimation $\hat{\boldsymbol{x}}(t_i^{k_i})$ at $t = T_i^{k_i} + T_i^1$ and creates a reference trajectory sequence $\{\check{\boldsymbol{x}}(i)\}_{i=0,1,\ldots,M}$, with $\check{\boldsymbol{x}}(0) = \hat{\boldsymbol{x}}(t_i^{k_i} + T_i^1)$. We define:

$$J_{\text{local}}(\hat{\boldsymbol{x}}(t_i^{k_i} + T_i^1)) = \sum_{i=0}^{M-1} \left(\left\| \check{\boldsymbol{x}}(i+1) - \check{\boldsymbol{x}}(i) \right\| \right) - \left\| \check{\boldsymbol{x}}(M) - \check{\boldsymbol{x}}(0) \right\|,$$

as a cost describing the curvature of the reference local trajectory. The offloading strategy can be formulated as:

$$S_3(t_i^{k_i} + T_i^1) = \begin{cases} \text{OFF, if } D(\hat{\boldsymbol{x}}(t_i^{k_i} + T_i^1)) - J_{\text{local}}(\hat{\boldsymbol{x}}(t_i^{k_i} + T_i^1)) \leq J^\star, \\ \text{ON, else,} \end{cases}$$

where $t_i^{k_i} + T_i^1$ indicates the time instant after the actuation and pose estimation. The constant $J^\star$ accounts for the degree of difficulty of the next moves in terms of proximity to obstacles and curvature of the trajectory. When S_3 in ON, the remote path planning provides the next step to reach the target position. Otherwise, the robot relies on the local path planning trajectory. It should be mentioned that, contrary to Switch 2, here, the CPU availability does not take part in the offloading decision, as it is noticed that the remote path planner chosen is mainly memory intensive.

5.8 Experimental Evaluation

The experiments were conducted in an operating space of 2.5×2.5 meters, divided by 25×25 cells, with a cell size of 10×10cm. The length of the AlphaBot is $22cm$ and the radius of each wheel is $6.6cm$. The coloured beacons were placed at the periphery of the grid for the localization procedure described in Section 5.6. The rectangular-shaped obstacles were placed as depicted with grey colour in Figure 5.14. The map is considered known. The Access Point used was a MikroTik wireless SOHO AP, providing up to 100Mbs LAN connection, Single Band (2.4GHz). The edge server deployed on the NETMODE, testbed part of Fed4FIRE[2] initiative, was equipped an Intel Atom CPU, up to 1Gbit Ethernet port and 8GB of RAM. The services provided by the edge server were deployed as Docker

[2]https://www.fed4fire.eu/testbeds/netmode/

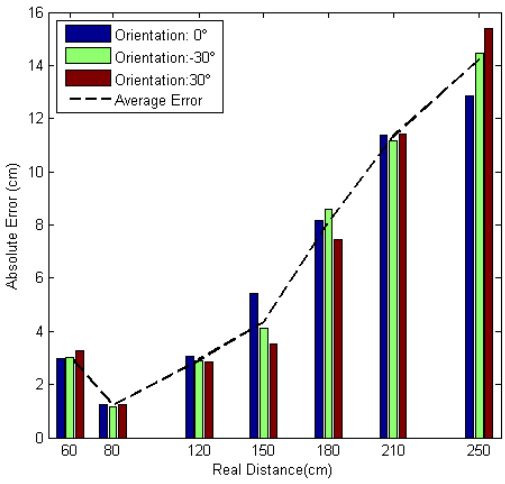


Figure 5.12: Absolute error of the estimated distance relative to the real distance from a Beacon, for different orientations.

containers. For each Docker container, one can set constraints, to limit a given container's access to the host machine's CPU cores, by provisioning a percentage of them as the virtual cores of the containers. Thus, containers can be assigned with partial virtual CPUs using decimal values. Using a collection of pictures from the actual experimentation room, from different positions and viewing angles, a dataset was created to estimate the time duration of the remote beacon-based localization.

In the first part of the evaluation, the effectiveness of the self localization technique described in Subsection 5.6, is briefly demostrated. The evaluation of the proposed technique is broken down into two parts; *i)* the association of the perceived distance's error with the real distance from the detected Beacon and *ii)* the overall accuracy of the final estimation of the AlphaBot's pose.

As depicted in Figure 5.12, the AlphaBot is located between $50cm$ and $250cm$ from the Beacon of interest. The distance of $50cm$ corresponds to the minimum distance from which a Beacon can be portrayed in whole with the current camera setup. One can notice that the absolute error of the distance-to-Beacon estimation increases gradually as the distance increases, but the accuracy never drops bellow 93%. Moreover, the different relative orientations of the AlphaBot seem to have a negligible effect in the accuracy of the distance estimation; $-30°$, $0°$ and $30°$ were randomly selected to illustrate this behaviour. It must

be highlighted that the most accurate estimations, though, were observed when the real distance between the AlphaBot and the identified Beacon was in the range $[80cm, 100cm]$, as the average of the estimation's absolute error was in the area of $1.3cm$, or approximately 1%.

When comparing the estimated poses with the real ones, it can be noticed that the combined coordinates error, after the bilateration of the two relative distances takes place, never exceeds 20% in either x or y axis. Regarding the estimation of the orientation, at each point, the absolute error lies in the $[2°, 12.5°]$ range. All in all, when a Beacon is correctly detected within the captured image, it is noticed that the proposed method is not only precise but also independent of environmental variables, e.g., light conditions, when it comes to pose estimation. To illustrate the overall accuracy of the proposed self-localization method, a random walk was composed for the AlphaBot to perform on the aforementioned operating space; the robot followed a predefined trajectory of random poses and estimated its position and orientation at each point. In Figure 5.13 the trajectory of the real positions is depicted with the blue dashed line, having at each point a specific orientation depicted with blue arrows, while the estimated positions and orientations are depicted with red dashed lines and green arrows respectively. The lines connecting the different points do not represent the actual movement of the AlphaBot but are drawn for clarity. The deviation between the real poses and the estimated ones produced by the proposed algorithm for this random walk, is considered acceptable for the selected application. It is noted that in a typical setting

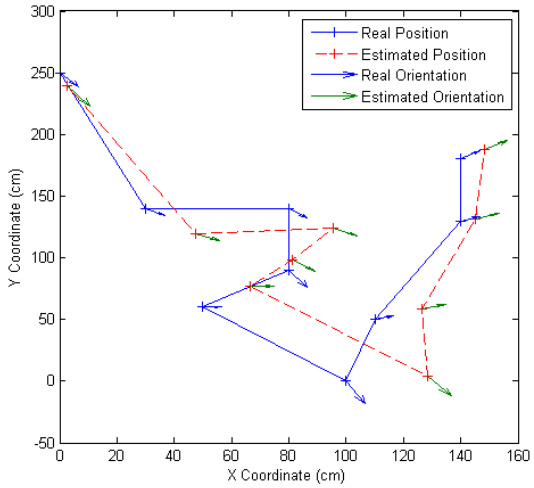


Figure 5.13: Real versus estimated AlphaBot trajectory.

where the robotic agent moves autonomously, the measurements generated by the discussed method can be fed to a state observer of the robot's position and orientation, improving significantly the accuracy.

Average Time per picture (sec), t_p	Virtual Allocated Cores, $\hat{c}$
2.41	0.25
1.06	0.5
0.56	0.75
0.39	1
0.30	1.25
0.26	1.5

Table 5.1: The average time for remote beacon-based estimation per virtual allocated core to the container.

In the second part of the evaluation, the benefits of the switching offloading mechanism are demonstrated. In Table 5.1, the values of the set of pairs $(t_p, \hat{c})$, introduced in Section 5.7.2, are presented. Using the least squares fitting method the coefficients $a = -1.34$ and $b = 1.675$ were calculated. Hence, the estimated processing time of the remote beacon-based localization is given by $t_p = -1.34\hat{c} + 1.675$. Provisioning over 1.5 cores resulted in similar computation time, thus, the maximum CPU allocation was set to that value. In the experiments conducted, the allocated cores of the containerized application were updated every 10sec, following a Normal Distribution with a mean value of 0.75 and 0.5 variance. The following values were used for the aforesaid constant values: $b_0 = 1$; $b_1 = 0.2$; $e_1 = 5$cm $e_2 = 5°$, $L_1 = 0.2$, $L_2 = 0.6$, $\delta^\star = 6$ and $J^\star = 3$.

Regarding the networking settings, a signal of power P_{dB} is assumed for the uplink, which is proportional to the distance between the robot and the access point it is connected to and which has a maximum value of $P_{dB}^{max} = 24dB$. Moreover, $PL(d_0)$ is fixed at $-20dBm$, based on the work of [169], which presents an access point with the same characteristics of ours and the same reference distance. The path-loss exponent n was set equal to 3.5, a value typical for a factory floor setting [171]. The size of offloaded data, in MB, followed a uniform distribution with a mean value of 0.075 and variance equal to 0.25. The Gaussian Noise N_{dB} was set equal to $-114dB$ while the bandwidth B allocated to the robot at any given time was set to $1MHz$.

Three experiments were conducted, namely, local only execution, remote only execution and the proposed switching offloading scheme. In Table 5.2 the average completion time

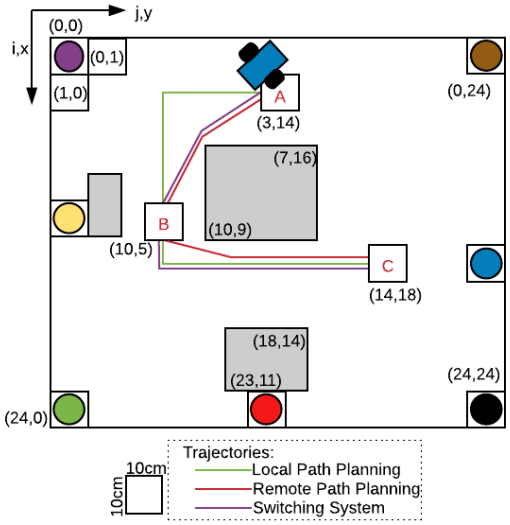


Figure 5.14: The experiment setup and the trajectories produced by the three experiments.

Experiment	Average completion time (*sec*)	Success Rate
Local Only Execution	61	40%
Remote Only Execution	105	100%
Switching System	90	100%

Table 5.2: The average completion time and success rate of 10 experiments for each setting.

and the average success rate for 10 experiments of each setting is presented. For the rest of the evaluation, the results of the best trials for each setting will be presented. Moreover, in Figure 5.14 the reference trajectories of these trials for the three experiments, are illustrated, with green colour for local only execution, red colour for remote only execution and purple colour for the switching system. As outlined in Section 5.6, the local $A^\star$ algorithm allows only four directions of movement, while the remote path planner allows any-angle movements. For better visualization, timelapse videos from the conducted trials for each setting have been uploaded[3]. In these experiments, the starting position for the AlphaBot was the already known position $A(3, 14)$, while the desired target reference positions were $B(10, 5)$ and $C(14, 18)$ in sequence. The scale of uncertainty is illustrated as a percentage of $\delta^\star$, i.e.,$\delta/\delta^\star$, which is the predefined quantity for Switch 1 to be ON.

[3]https://github.com/Dspatharakis/alphabot-ppl/tree/master/timelapsed-videos

Experiment A - Local Only Execution

In the first experiment, Switches 1 and 3 were ON, throughout the experiment and Switch 2 was never used. During this, the AlphaBot never used the beacon-based localization. As a result, the only estimation of its position is coming from the photoelectric sensors and the local estimation procedure. Moreover, the path planning algorithm chosen was the A* algorithm, executed locally at the Pi at each given time. This setting results to a fast, although not precise navigation with $\delta/\delta^{\star}$ growing monotonically. Without executing the sophisticated beacon-based localization or the modified Dijkstra's shortest path planning algorithm, the average duration was 61 *sec*. Also, in Figure 5.15, one can notice that the duration of the experiment is proportional to the number of steps produced by the A* solution. Thus, the actuation of the AlphaBot is the main time consuming process in the experiment.

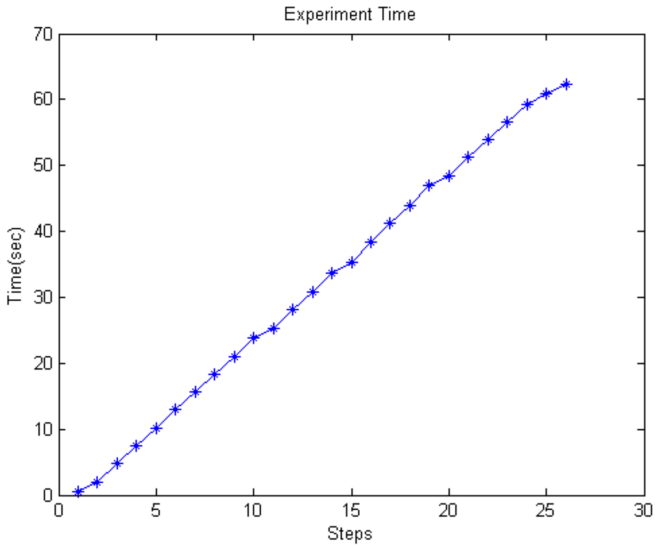


Figure 5.15: Experiment A - Duration of the steps of A* at the Raspberry

Finally, the amount of successful trials was low. Consequently, without a more sophisticated localization algorithm and a more precise path planning technique there is no guarantee the target reference position is reached; the pose uncertainty grows with time, as depicted in Figure 5.16.

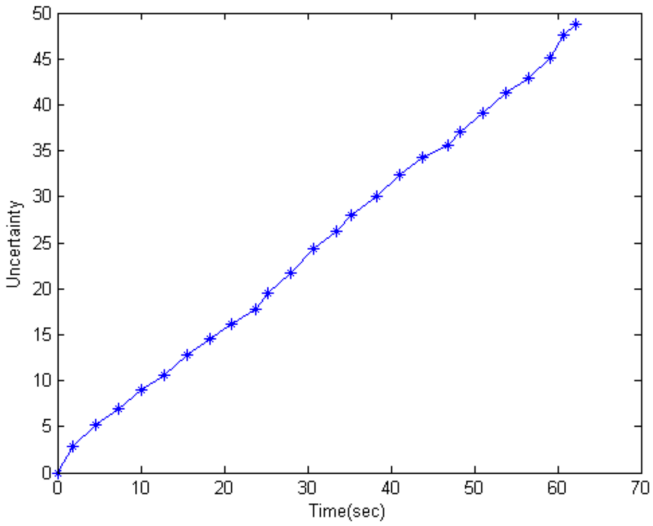


Figure 5.16: Experiment A - Uncertainty during Experiment A.

Experiment B - Remote Only Execution

In the second experiment, whenever the uncertainty about AlphaBot's pose grew over the predefined threshold $\delta^{\star}$, beacon-based localization was invoked (Switches 1 and 2 ON) on the edge server. Moreover, the path planning chosen was the modified Dijkstra's shortest path algorithm, which was also always executed remotely (Switch 3 ON). In this setting, the robot always reached the target positions, as shown in Table 5.2, although the completion time was heavily affected, as shown in Figure 5.17. Beacon-based localization was executed twice during this experiment and, as a result, $\delta/\delta^{\star}$ became equal to 0. The setup of the particular experiment underlines the importance of a slower but more precise navigation. The minimum transmission time for the photos used for beacon-based localization was 1sec, while the maximum was close to 1.5sec. Moreover, the time consumed for the Dijkstra's solution at the remote server had an average value of 0.9sec. This experiment involves a powerful server at the Edge layer with designated computing cores for the needs of each process. As a result, the computing time for each task is low. However, the overhead of transmission is significant in comparison to the previous experiment. Last but not least, although the computational time is significantly low, one must not forget that this is the result of overprovisioning the resources of a whole server. The resources of the server were underutilized for the most time during this experiment.

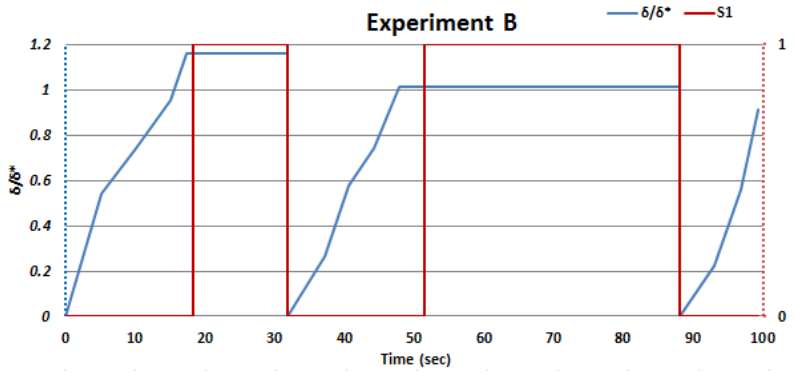


Figure 5.17: Experiment B - Remote Only Execution.

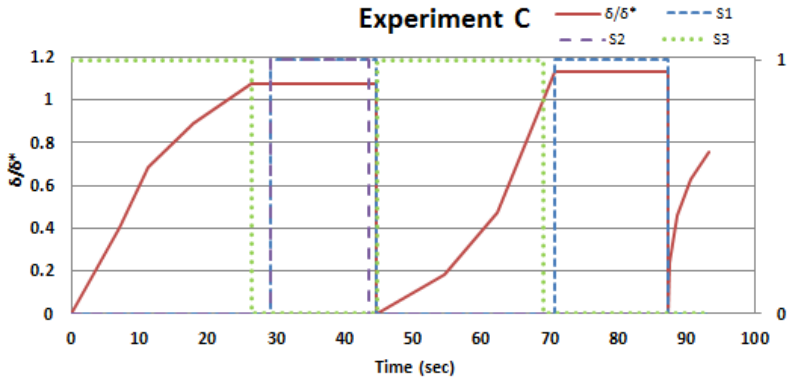


Figure 5.18: Experiment C - Switching System.

Experiment C - Switching System

In this experiment the full functionality of the switching system proposed in this work is highlighted. As described in Section 5.7.3, Switch 3 decides which path planning algorithm solution the AlphaBot will use to generate the next reference position. A server at the edge of the network was utilized as the remote server to draw attention to the advantages of the low transmission time due to WAN connection between the server and the Raspberry mounted on the AlphaBot. This communication between the server and the Raspberry was over http. In comparison to the "Local Only Execution" experiment, the time spent to reach the final goal was much longer, due to the time consuming beacon-based localization, and the increased computation time needed for the more advanced path planning solution of the modified Dijkstra implemented at the dedicated server. However, this setup provided a very precise and robust navigation for the robot, leading to a very high success rate of the experiments. At the dedicated server, a dynamic resource allocation mechanism was deployed, changing

the allocated CPU cores of the Docker Container which hosted the service waiting to execute offloaded tasks. As a result, this optimal resource allocation led to guaranteed low response times in the offloaded localization and navigation tasks, while overprovisioning of the CPU resources was avoided (opposed to the "Remote Only Execution" experiment).

In detail, when, the curvature function of the trajectory calculated by the $A^\star$ algorithm and the obstacle density function exceeded the threshold value $J^\star$, the remote path planning solution was selected; e.g., from the beginning of the experiment until the 25th *sec* of the simulation and from the 43rd sec till the 67th *sec*, as illustrated with green dashed line in Figure 5.18. In the same figure, with red solid line, $\delta/\delta^\star$ is depicted. Two times during the experiment the more precise beacon-based estimation was invoked to reset $\delta/\delta^\star$. The first estimation attempt, at the 25th *sec* of the experiment, was executed on the edge server, because S2 was ON. The second one, at the 71st *sec* of the experiment, was executed locally, as S2 dictated (OFF), because the estimation of the CPU availability of the edge server, provided by the Kalman Filter, along with the network delay for each picture, at that time, would have provided worse results than the local execution. This setup provided a very precise and robust navigation for the robot, leading to a very high success rate of the experiments, achieving a balance between execution time and trajectory accuracy.

Chapter 6

MRF-based Distributed Energy-aware Resource Allocation at the Network Edge

6.1 General Setting

As mentioned in the Introduction of this thesis, efficient resource allocation for task offloading at the Network Edge, is required for both the IoT devices/end users as well as the infrastructure providers. On the users' side, as discussed in previous chapters, efficient resource allocation is translated to an overall improvement in the experienced QoS and QoE. On the providers' side the benefits are pinpointed in the minimization of the energy consumption of the data centers, which is mainly affected by the number of the servers that are activated to serve the incoming workload and which, in turn, is directly proportional to the operating expenses of the infrastructure.

Thus, in this chapter a framework is introduced which simultaneously addresses energy consumption minimization and distributed load balancing, while respecting the applications' QoS requirements. Initially, a wireless protocol is simulated in order to extract the instantaneous throughput under dynamic wireless network conditions, and the mobility of the users is predicted with the use of an n-Mobility Markov Chain location prediction method. Based

on this prediction, pre-computed profiles of virtual machines (VMs) are leveraged to enable proactive and dynamic resource allocation at each edge site, ensuring the QoS constraints of any deployed application. Containers can also be considered as the virtualization units without any change in the modeling. Finally, a novel load balancing technique based on Markov Random Fields (MRF) is introduced to appropriately distribute the excess workload among the available edge sites, towards the minimization of the total energy consumption.

6.2 Related Work

The problem of task offloading falls into the knapsack resource allocation category which is NP-hard in general [172]. Most of the proposed approaches follow a partial or full offloading technique, according to whether the tasks are separated or not, with the goal to minimize the overall latency and/or energy. Furthermore, they propose static resource allocation schemes on the edge infrastructure. In this chapter, the design principles of [173] are adopted and the ENERDGE framework is proposed, which is a mobility-aware and full offloading approach in order to minimize the energy consumption of the edge infrastructure under specific QoS guarantees for the mobile applications hosted. In this context, there are three interesting and related directions in the literature: i) mobility prediction for task offloading, ii) single-site task offloading and resource allocation, and iii) multi-site task offloading and resource allocation.

6.2.1 Mobility Prediction for Task Offloading

The success of offloading decisions depends heavily on the dynamic nature of task behavior and user mobility. In particular, the users may move and resource prices for offloaded task execution may vary over time. This led the authors in [174] to propose an online algorithm with a logarithmic objective to minimize the resource usage of the edge infrastructure, while taking into account the impact of mobility in the latency. They also formulate a VM migration cost for the tasks that need to follow the users' movement. In a similar manner, Wang et al. [175] assume a mobility prediction with fixed accuracy and propose VMs migration based on predicting the future costs of their placement. A migration policy, however, for containers, is also formulated in [176], where the authors introduce an architecture in which

Fog Computing services constantly move in order to be always close enough to the served IoT mobile devices.

Since the mobility of the users can significantly impact the latency and increase the migration cost, the authors in [177] introduced a prediction mechanism to ameliorate the offloading performance. A similar approach is followed in [101], where the most popular services are proactively installed in the Edge servers located in the positions that the users will most probably visit, thus reducing the network delay during task offloading. Another approach, denoted as MAGA and introduced in [178], is based on frequent moving patterns of the users and a genetic algorithm to partially offload tasks to edge servers. However, in all of the aforementioned works the authors assume static resource allocation at the edge, in terms of amount of resources utilized.

6.2.2 Single-Site Offloading & Resource Allocation

In case of task offloading, a single edge site is usually available in close proximity to the users. The main focus in this type of resource allocation problem lies in the latency and energy minimization. For example, the authors in [179] investigate the task offloading of augmented reality applications emphasizing on the computation intensive tasks (i.e., object recognition and position tracking). A successive convex approximation approach is proposed to minimize energy consumption under latency constraints, while emphasizing on both the available computation and communication resources at the Edge. Another energy-efficient based approach is presented in [172], following a mixed discrete-continuous optimization approach along with a low-complexity heuristic based on Johnson's algorithm.

Regarding latency, authors in [72] study the admission control and resource allocation problem of computationally intensive IoT applications at the Edge. A Lyapunov dynamic stochastic optimization approach is used with the goal to reduce the end-to-end delay, while improving the overall throughput. Similarly, [180] investigates the mobile-edge computing offloading problem with the goal to minimize the latency in a multi-user scenario with joint communication and computational resources. The solution is based on the Lagrange multiplier method. However, such centralized task offloading approaches usually fail to apply to realistic scenarios of larger edge infrastructures with multiple, geographically distributed sites.

6.2.3 Multi-Site Offloading & Resource Allocation

In case there are multiple edge sites in close proximity to the devices, task offloading includes both the resource allocation of the tasks and the selection of the right administrative domain (i.e., edge infrastructure). In this context, an edge orchestrator can be used to assign the tasks to the appropriate domain, with the goal to maximize the number of successfully assigned task requests [181]. Sonmez et al. [182], proposed a fuzzy workload orchestrator for multiple Edge and Cloud infrastructures. For each offloaded request, a set of fuzzy rules determined the destination computational unit within a hierarchical multi-site architecture. However, the authors empirically defined the fuzzy rule sets, while assuming static resource provisioning on the edge servers, which might not be applicable to real conditions where services typically bear different workload characteristics. Plachy et al. [183] leveraged a probabilistic modeling of the mobile users' movements in order to pre-allocate computing resources on multiple edge base stations and alleviate the potentially unreliable mobility and channel predictions. Subsequently, a low complexity algorithm decides on the best communication path between the user and the selected base station.

Another goal can be the balancing of the load between edge servers while minimizing the application response time. In [99], over-utilized edge servers redirect part of their incoming workflow to resource-rich or under-utilized servers, using a minimum cost max flow algorithm towards achieving total balance in terms of average application response time in the whole edge infrastructure. An extension to this work is presented in [184], where a genetic algorithm is exploited for a distributed load balancing of traffic, yielding a solution that converges to the minimization of maximum task response time through gene mutations.

6.2.4 Markovian Random Field -based Solutions

The motivation behind the decision to utilize an MRF-based solution in the load balancing and resource allocation problem originates from the work in [185], where a distributed control approach is proposed for self-organization of autonomous swarms. The swarm is modeled as an MRF and the desired global behaviors can are encoded into the Gibbs potential function characterized by local interactions. The proposed scheme is scalable, the computational requirements remain the same as the number of nodes increases and it can can easily accom-

modate various constraints. Due to these inherent advantages, MRFs have been successfully utilised in various applications of different domains during the past years; for example, in computer vision, Geman et al. [186] used MRFs to model the correlation among neighboring pixels and capture certain statistics of natural images. In the field of recommender systems, Liu et al. [187] used the representational power of MRFs and Conditional Random Fields (CRF), to find out the predicted ratings for inducing unknown preference relations (PRs). On the other hand, Karyotis et al. [188] collects and presents works that incorporate MRFs in hybrid recommender system, which also combine the filtering of huge collections of items with analysing behavioral properties and the interplay between the entities of a social platform in order to achieve more precise recommendations. Finally, in the cognitive radio networks (CRN) domain, Anifantis et al. [189] adopted a Radio Channel Allocation (RCA) framework that combines a Markov Random Field formulation with Gibbs sampling, allowing distributed and efficient operation for each Secondary User (SU). According to this, every SU can calculate an "energy" function based on its current state and the states of its neighbors. The goal for each SU is to minimize interference through minimization of its local energy function. SUs asymptotically converge to global optimal solutions, by progressively updating their energy functions through local sampling. In [190], the authors also utilize an implementation of an MRF-based cross-layer framework for resource allocation among SUs in CRN environments.

6.2.5 Contributions & Outline

In order to overcome the aforementioned challenges and achieve the discussed goals, a novel framework is proposed, referred to as ENERDGE, which jointly tackles task offloading and resource allocation of multiple edge data centers in a distributed and energy-efficient manner. The framework has a gradual operation, introducing the following key contributions:

- A performance modeling approach based on Switching Systems Theory is proposed, to define virtual hardware profiles, i.e., flavors, for the edge infrastructure, providing application QoS guarantees under various operating conditions. This modeling allows for dynamic selection and allocation of the appropriate amount of resources for each application (i.e., switching between the different hardware profiles), based on the

anticipated workload demands. Leveraging the capabilities provided by this switching, a two-stage distributed, energy-aware, proactive resource allocation mechanism is designed.

- During the first stage, the works of Chapter 3 and 4 are extended, that jointly address task offloading and resource allocation on a single edge site (i.e., [63, 191]), to simultaneously minimize the total energy consumption of each edge site and provide guaranteed satisfaction of the QoS requirements of each deployed application. In order to accommodate the workload prediction needs at this stage, an existing user mobility prediction mechanism is utilised, based on the concept of the n-Mobility Markov Chains location prediction [192], to estimate the movement of the mobile devices between different sites within the edge infrastructure.

- During the second stage, this approach is combined with a novel Markov Random Field (MRF) mechanism that incorporates in its objective function all optimization criteria; this mechanism aims at redirecting tasks that cannot be executed locally under the given energy and QoS requirements of the first step, balancing resource utilization throughout the whole infrastructure. Thus, it achieves a better total energy management optimization through an efficient state space search in a distributed fashion, while taking into consideration any additional network delays incurred. This is the first approach of such a combination, and it could potentially pave the way for other similar MRF designs as optimizers in relevant problems. The integration of the above modeling and resource allocation approaches composes a task offloading and energy-aware resource allocation mechanism for accommodating dynamic spatiotemporal workload demands.

- Finally, a detailed evaluation of the proposed approach is provided, in terms of energy consumption minimization and QoS satisfaction for both stages of the mechanism. Then, it is compared with a well-established study in [99]. Based on a realistic application simulation, the discussed solution outperforms the approach in terms of adaptation efficiency. In other words, the proposed approach yields less energy consumption for achieving the same QoS guarantees, or equivalently, it achieves higher QoS guarantees for the same energy consumption.

6.3 System Modeling

6.3.1 Edge Infrastructure & Applications

To facilitate the extensive modeling employed in this work, Table 6.1 summarizes the key notation used throughout this chapter. The physical infrastructure is modeled as a group of wireless access points, each directly connected with a cluster of homogeneous servers, as illustrated in Figure 6.1. These physical resources altogether form an edge data center, which hereafter is referred to as site s_k, with $S = \{s_k\}_{k=1}^{n}$ being the set of sites, for n sites in total. This set forms a graph, where each site corresponds to a node and the edges to the interconnections between them through routers, used only for forwarding purposes (i.e., backhaul network). Furthermore, the servers of the edge infrastructure are considered to be located in different sites are heterogeneous. This implies differentiation on processing capabilities and service completion time among sites.

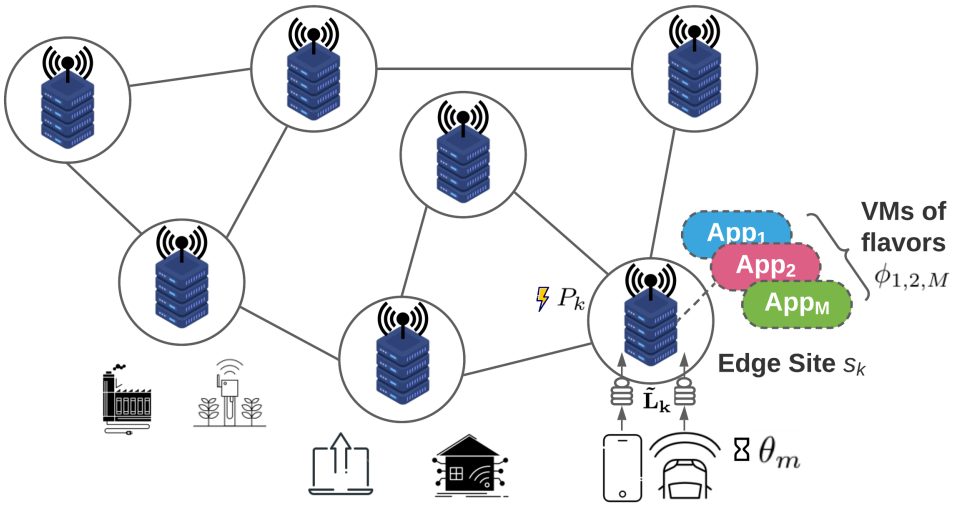


Figure 6.1: Example of considered edge infrastructure.

For the access layer, the existence of various and heterogeneous end-devices (e.g., IoT, mobile devices) is assumed, each associated with one of M specific mobile applications (i.e., augmented reality, wearables, etc.). Each application $m \in \{1, \ldots, M\}$ comes with specific requirements in terms of QoS (e.g., average response time) that will guide the allocation of the resources.

Table 6.1: Summary of the key notation.

Symbol	Interpretation
s_k	Site k
S	Set of sites, $n = \|S\|$ sites in total
M	Number of applications
θ_m	Acceptable response time for App. m
ϕ_m	VM flavor of application m
c_m	Cores requested by VM flavor ϕ_m
μ_m	Throughput guaranteed by VM flavor ϕ_m
Ser_{cpu}	Server's CPU capacity
P_{ser}	Server's power consumption
P_{max}	Server's max. power consumption
$P(\phi_m)$	Power consumption of VM flavor ϕ_m
$\mathbf{z}_i$	A feasible VM formation
$\mathcal{Z}_\mathbf{k}$	Set of feasible VM formations at site s_k
N	Size of $\mathbf{z}_i$ VM formation
C_k^{ser}	Servers' CPU cores threshold at site s_k
P_A	Edge infrastructure's power consumption
P_k	Power consumption of site s_k
f_i	Number of servers with $\mathbf{z}_i$ VM formation
E_k	Number of available servers in site s_k
p_i	Power consumption of VM formation $\mathbf{z}_i$
r_i^m	Max. workload served by VM formation $\mathbf{z}_i$
$\tilde{\mathbf{L}}_\mathbf{k} = [\tilde{L}_k^m]$	Predicted workload for site s_k
$\mathcal{N}_{s_k}$	Neighborhood of site s_k
$\mathbf{w}_k = [w_m^{(k)}]$	Excess workload for App. m at site s_k
$\mathbf{b}_k = [b_i^{(k)}]$	Number of servers of type i at site s_k
$P(\mathbf{b_k})$	Power consumption of $\mathbf{b_k}$
$\mathbf{X}_k = \{\mathbf{W}_k, \mathbf{B}_k\}_{k=1}^n$	Random field
$V(\boldsymbol{\omega})$	MRF potential function
$C_1, C_2, C_3, \Delta_1, \Delta_2$	Properly selected MRF constants
L, K, x_0	Parameters of reflected sigmoid function
t	Visiting epoch of MRF
w	MRF sweep index
$T(w)$	MRF temperature at sweep w

6.3.2 Task Offloading

As depicted in Figure 6.1, each end-device running an application m offloads its computational intensive processes to the Edge to reap the benefits of the more powerful computational resources. In this work, an IEEE 802.11ac access network to offload the tasks from the devices is assumed. Following the work of [193], the access network is modeled by using an indoor TGnAC Channel B, suitable for large open space and office environments [194]. Along the same lines, in order to capture the dynamic nature of the wireless network, the transmission rate of the devices is adjusted according to an enhanced version of the Minstrel algorithm [195]. In this manner, the devices are able to change the modulation and coding scheme (MCS) used, and thus the transmission rate, conforming to the varying channel conditions and interference from nearby devices (SINR). This procedure allows the creation of a realistic dataset containing tuples of the form <*number of users, offloading request rate of each user*>, which is publicly available[1], and utilize it to translate the predicted number of users to the anticipated request rate, for a specific edge site. Specifically, it is assumed that each user constantly offloads at his/her maximum achievable data rate, and, considering a fixed offloaded task size, producing the anticipated workload volume for the estimated number of users is feasible.

It is assumed that each end-device needs to fully offload its requests on edge servers following a VM/container-based provisioning method. Depending on the user's location, the offloaded tasks are assigned to the site where the wireless transmission occurs. Each VM/container of the site's servers serves the offloaded requests of the application m that it was assigned to. It is noted here that, for the sake of simplicity, focus is placed on scenarios and settings where the user's movement is typically limited close to the site of interest during the whole offloading procedure. Therefore, the offloading procedure for a single task is assumed to complete within the same site that it was initiated in and, consequently, no handover processes and costs are considered. The most important QoS requirement of the offloaded tasks of application m is the maximum acceptable response time θ_m value, which is application-specific. Under this setting, the end-device accelerates the execution of computationally intensive tasks and extends its battery lifetime.

[1] `https://github.com/maravger/netmode-cloudsim/blob/master/task_offloading_ds_verbose.xlsx`

6.3.3 VM Flavor Design

On each edge site, it is essential to facilitate the proactive dynamic resource allocation due to the varying number of the offloading requests received. The term VM (or container) flavor is introduced for every deployed application, which describes the relation between the application's response time, the allocated CPU cores and the number of the offloaded requests. The computation of these VM flavors is based on switching systems from the System Theory. The advantage of the VM flavor design is two-fold; first, this modeling approach allows for accurately capturing the dynamic behavior of the application-specific VMs, under various operating conditions. Second, calculating a multitude of VM flavors, allows for quickly adjusting the edge infrastructure to different pairs of workloads and applications, while providing a level of guarantee for the QoS specifications.

The VM (or container) flavor $\phi_m \in \Phi$ of application m is defined as a tuple that includes the QoS specifications of the hosted application, the requested resources for the VM that will provide the QoS guarantees and the maximum throughput of offloaded requests, for which the VM will be able to achieve these guarantees, $\phi_m : < \theta_m, c_m, \mu_m >$. Specifically, parameter θ_m denotes the average response time that the VM of flavor ϕ_m guarantees to achieve with c_m CPU cores allocated to it and for a maximum throughput of μ_m offloaded requests per time unit. The assumption is made that the response time consists of two terms: (a) transmission time and (b) service completion time. The transmission time includes the time to transmit/upload the application's request through a wireless link. In particular, since the wireless link has been modeled through the IEEE 802.11ac protocol, calculating this delay is feasible by leveraging the information of throughput achieved and the application's task size. Regarding, the time to download the response from the server, since the size of the output is generally much smaller than the input, this delay can be usually omitted [196]. Service completion time includes the VM/container startup time, as well as the queuing and processing time of the application tasks at the assigned servers. A flavor could also define the memory requested by the VM. However, it is omitted from the problem formulation due to the following reasons: First and foremost, memory power consumption is negligible compared to CPU power consumption [197]. Secondly, following the paradigm set by well-known edge computing frameworks like MAUI [198] and ThinkAir [199], focus is placed on the offloading of CPU-intensive tasks.

In principle, the performance of an application hosted on a VM is non-linear and cannot be described analytically. However, adopting linear modeling allows for an easier identification of the system, without significant loss of accuracy, and enables the implementation of various optimization and control methodologies. In order to extract the VM flavors for each application deployed on a site, the modeling approach of Chapter 4 [63] is modified; for each application and for each flavor ϕ_m of this applications' VMs, a scalar, discrete Linear Time-Invariant (LTI) system is identified. In particular, the VM flavors are mainly differentiated based on the number of CPU cores they require, which also constitutes the switching criterion of the proposed mechanism. Thus, during this identification phase, for each application and for each different CPU core allocation, the operation of the corresponding VM is described by a discrete linear system of the following form,

$$\theta(\tau + 1) = a\theta(\tau) + b\mu(\tau), \tag{6.1}$$

where $\theta(\tau)$ represents the average response time for the deployed application, within a time period τ and $\mu(\tau)$ the number of offloaded requests within the said time period. The coefficients $a \geq 0$ and $b \geq 0$ are known scalars which can be estimated by the Recursive Least Square algorithm [200].

Physically, a VM with c_m allocated cores can only serve up to μ_m offloaded requests of the deployed application while guaranteeing an average response time of θ_m for the specific time period. This constitutes the physical interpretation of a flavor ϕ_m and generally, for each such switching system, a set of feasible VM flavors of this kind can be computed according to certain performance criteria and input constraints. In this case, these feasible VM flavors are computed by solving the following linear programming problem with the goal to maximize the number of the offloaded requests:

$$\max_{\theta_m, c_m} \quad \mu_m \tag{6.2a}$$

$$\text{subject to} \quad \theta_m = a\theta_m + b\mu_m \tag{6.2b}$$

$$\theta_{\min} \leq \theta_m \leq \theta_{\max} \tag{6.2c}$$

$$\mu_{\min} \leq \mu_m \leq \mu_{\max} \tag{6.2d}$$

The first constraint dictates that each flavor must also be an *equilibrium point of the discrete linear system*, which will guarantee its stability and confinement in a specific operating area around it. The second constraint implies that the average response time must lay between a minimum ($\theta_{\min}$) and a maximum value ($\theta_{\max}$) set by the application's QoS requirements, while the last constraint refers to the offloaded requests varying within the applications anticipated throughput range.

By having a set of VM flavors corresponding to different core allocations and maximum throughputs, a better level of accuracy is provided than using a single LTI model for the whole operation. In such a way, the extracted VM flavors correspond to realistic operating conditions and constitute the fundamental elements for the ENERDGE resource allocation mechanism.

6.3.4 Power Modeling

When fully offloading tasks, the total computational and energy burden is shifted away from the devices. However, reviewing this shift from a complete network-wide view one can easily understand that the problem is simply pushed at the Edge. Thus, in this work, the minimization of power consumption at the edge infrastructure is also taken into consideration. This includes switching physical devices on and off and optimizing the computational resource usage during the offloading.

Usually, for the server power dissipation, an almost linear relationship between the power consumption of a server and its CPU utilization exists. The following model, can accurately predict the servers' power consumption P_{ser} with an error below 5% [197]:

$$P_{ser} = \gamma \cdot P_{\max} + (1 - \gamma) \cdot P_{\max} \cdot u, \tag{6.3}$$

where $P_{\max}$ is the maximum power consumed when the server is fully utilized, γ is the percentage of power consumed by an idle server (usually around 60% [201]) and u is the current CPU utilization.

In order to extract the power consumed by a VM of flavor ϕ_m provisioned in a server,

the above equation is transformed as follows:

$$P(\phi_m) = \begin{cases} \gamma \cdot P_{\max} + (1-\gamma) \cdot P_{\max} \cdot \frac{c_m}{Ser_{cpu}}, & \text{if } u = 0 \\ (1-\gamma) \cdot P_{\max} \cdot \frac{c_m}{Ser_{cpu}}, & \text{otherwise,} \end{cases} \tag{6.4}$$

where Ser_{cpu} is the total amount of the available computational resources in a server, i.e., CPU cores. Hence, for the first VM provisioned at a server the power consumption will include activating the server and the power consumption added by the particular VM. For the rest of the VMs only their power consumption is taken into consideration. It is worth mentioning, that an isolcpus technique [202] is assumed, where the requested CPU resources are isolated and pinned to the VM. This is a common technique for performance optimization when virtualizing x86 servers. Thus, each VM will have access only to its share of CPU resources consuming as well the corresponding power.

6.3.5 Mobility and Workload Prediction

As discussed in the previous subsections, each site hosts a group of IoT/mobile applications and serves the offloaded requests that are generated by the devices within the range of its wireless access point. However, in both mobile and IoT applications, mobility is a key feature and must be considered by the offloading decision and resource allocation mechanism, as it creates dynamic network conditions. Towards the optimal resource allocation policy, an accurate prediction of this is necessary.

In order to address this issue, a variation of the n-Mobility Markov Chains (n-MMC) location prediction method described in [192] is implemented. In a nutshell, this method incorporates the two previous visited sites of a mobile device and a Mobility Markov Chain in order to probabilistically predict the device's next location. As a prerequisite, this method requires a transition matrix available, containing all the feasible transitions of a device between the sites, associated with their probability of occurring.

In order to create this transition matrix, the Melbourne Museum dataset [203] is used, which comprises 158 complete real visitor pathways, in the form of time-annotated sequences of visited exhibit sites. After processing the data, each path was assigned a probability based on its frequency of occurrence. This resulted in a transition matrix whose rows represent

the three last visited sites and its columns represent the next site to be visited. In this way, predicting the next location of a visitor is simple. Their three most recently visited sites are traced, the row in the transition matrix that corresponds to this trace is searched and the column with the maximum probability of transition for this row is located. The site of this column is the predicted next location. Finally, having available the collective statistics regarding the predicted locations of the users for the upcoming time period, the predicted offloaded workload, $\tilde{\mathbf{L}}_{\mathbf{k}} = [\tilde{L}_k^m]$, is acquired for the respective site s_k and application m, as described in Subsection 6.3.2.

6.4 Resource Allocation & Workload Balancing

Leveraging the Switching System modeling approach introduced in the previous section, in this section a 2-stage distributed, energy-aware, proactive resource allocation mechanism is proposed. In the first stage, an initial resource allocation optimization takes place locally at the site of each Edge, which balances between energy consumption minimization and QoS satisfaction. In the second stage, a novel distributed technique is applied to redirect the excess workload to under-utilized sites, thus balancing the resource utilization and achieving a better energy management.

6.4.1 Resource Allocation Optimization

In order to accommodate a proactive and dynamic resource allocation, the work of Chapter 4 [63] is followed where time is considered slotted. In this stage, at the beginning of each system slot, a decision is made on the topology to be implemented on each site, which will enable it to handle the projected offloaded workload. This topology defines the number of edge servers to be activated in each site along with the VM formation to be placed in each edge server, i.e., the number and flavor of the VMs.

Feasible VM formations are the ones where the sum of the CPU cores requested from the co-hosted VMs' flavors does not exceed a predefined threshold. For instance, assume two applications *App*1 and *App*2. A VM running *App*1 and instantiated in a flavor that requests two CPU cores, along with a VM running *App*2 and instantiated in a flavor that requests one allocated CPU core, is a feasible VM formation for a single edge server, as

the cumulative number of allocated CPU cores does not exceed the threshold of three cores (75% of the server's total available CPU capacity, $Ser_{cpu} = 4$).

The set of all feasible VM formations for edge servers in site s_k is defined as,

$$\mathcal{Z}_{\mathbf{k}} := \{\mathbf{z_i} = \left(\phi_m^{(j)}, \ldots, \phi_m^{(N)}\right), m \in [1, M], \quad j \in [1, N] : \sum_{j=1}^{N} c_m^{(j)} \leq C_k^{ser}\} \quad (6.5)$$

where $i \in [1, |\mathcal{Z}_{\mathbf{k}}|]$ is the index of the VM formation, $\phi_m^{(j)}$ is the VM flavor, $c_m^{(j)}$ the number of cores requested by the flavor of VM j of application m, M is the number of applications available at site s_k, N is the total number of VMs contained in formation $\mathbf{z_i}$ and C_k^{ser} is the CPU cores threshold set for each edge server of s_k. Due to the fact that the edge servers within a single site are considered homogeneous in terms of their resources, C_k^{ser} has the same value for all of them that are tied to a site s_k.

The system cost is defined as the total power consumption of the edge infrastructure. Since in this stage of the resource allocation mechanism no exchange of workload takes place between the sites, minimizing locally the power consumption, P_k, of each individual site, s_k, results in minimizing the total power consumption, $P_A = \sum_{k=1}^{n} P_k$, where n stands for the total number of sites in the infrastructure. This can be achieved by optimizing the amount of edge resources that will be activated in each slot to serve the total predicted workload. Consequently, the corresponding optimization problem can be defined as:

$$\min_{f_i, P_k} \quad \{P_k\} \quad (6.6a)$$

$$\text{subject to} \quad f_i \geq 0, \ i = 1, \ldots, |\mathcal{Z}_{\mathbf{k}}| \quad (6.6b)$$

$$\sum_{i=1}^{|\mathcal{Z}_{\mathbf{k}}|} f_i \leq E_k \quad (6.6c)$$

$$P_k = \sum_{i=1}^{|\mathcal{Z}_{\mathbf{k}}|} f_i p_i \quad (6.6d)$$

$$\sum_{i=1}^{|\mathcal{Z}_{\mathbf{k}}|} f_i r_i^m \geq \tilde{L}_k^m, \ \forall m \in \{1, \ldots, M\}, \quad (6.6e)$$

where the positive integer variables f_i denote how many servers need to be activated with the $\mathbf{z_i}$ VM formation of set $\mathcal{Z}_{\mathbf{k}}$, assuming the total number of formations of edge servers in site s_k is $|\mathcal{Z}_{\mathbf{k}}|$ and the total number of the available edge servers is E_k. Then, the sum of the

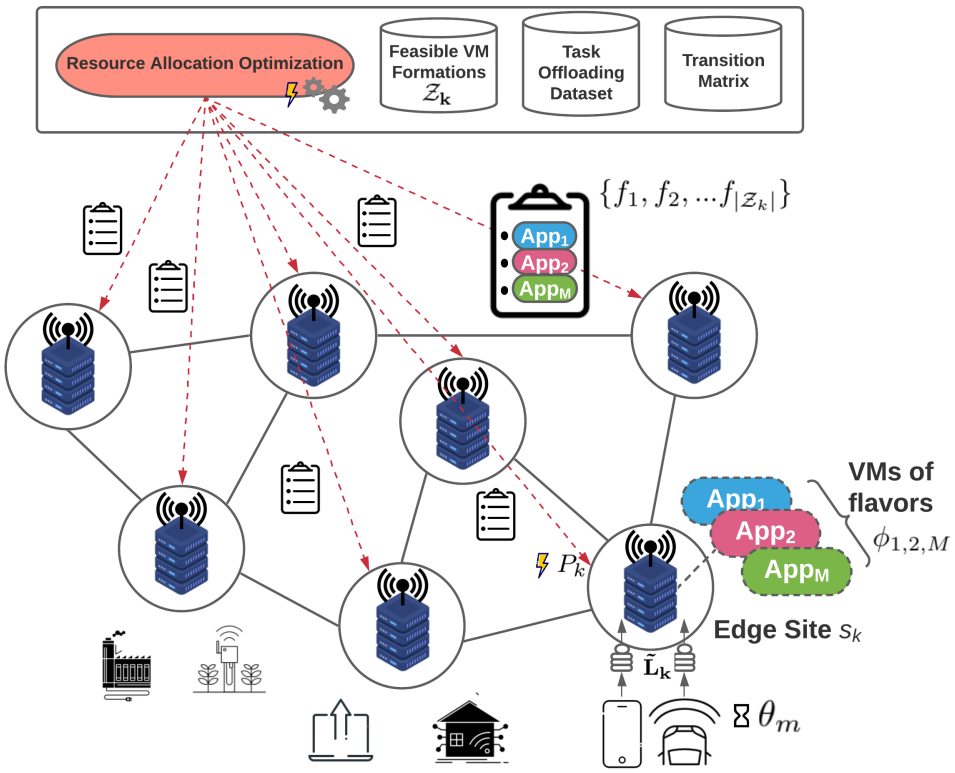


Figure 6.2: Resource Allocation Optimization Overview (Stage 1).

f_i variables cannot be greater than E_k (constraint (6.6c)). Constraint (6.6d) requires that a site's power consumption is equal to the sum of the power consumption of its activated edge servers. It should be noted that, following common considerations in the literature [63], it is assumed that the total number of available edge servers in a site is relatively small, thus the overall computation complexity of the optimization process is kept minimum, allowing the problem to be solved online.

As discussed in Subsection 6.3.4, the power consumption of each VM is proportional to its flavor. As a result, power consumption p_i of one edge server activated with the $\mathbf{z_i}$ VM formation is calculated as follows:

$$p_i := p(\mathbf{z_i}) = \sum_{j=1}^{N} P(\phi_m^{(j)}),\ m \in \{1, \ldots, M\}. \tag{6.7}$$

Finally, the last M constraints of (6.6e) denote that the total predicted workload for each application at s_k, $\tilde{L}_k^m$, for the next system slot, is satisfied by the activated edge servers in

each site. Again, as discussed in Subsection 6.3.3, the workload guaranteed to be served by one edge server with the z_i VM formation is:

$$r_i^m := r^m(\mathbf{z_i}) = \sum_{j=1}^{N} \mu_m^{(j)},\ m \in \{1, \ldots, M\}. \tag{6.8}$$

Problem (6.6a) is solved in a distributed fashion, locally in each site and proactively at the beginning of each system slot, after collecting all the required information (i.e., available resources and predicted workload). An overview of this process is depicted in Figure 6.2.

6.4.2 Inter-site Balancing of Excess Workload

In edge infrastructures, network traffic, therefore offloading requests, exhibit considerable variation. On the one hand, there are cases where the total predicted workload for a site exceeds its available resources' capabilities, in which case problem (6.6a) has no solution. In this situation, all the site's edge servers are activated with a fixed $\mathbf{z_{max}}$ formation, where $\mathbf{z_{max}}$ stands for the VM formation that accommodates the maximum possible number of offloaded requests for each application. Even so, a portion of the predicted workload will remain unserved (*overloaded site*). On the other hand, it is common that the total predicted workload for a site is lower than the predefined threshold that dictates whether the energy cost of activating the site's edge servers is worth serving it. Again, a portion of the predicted workload will remain unserved (*underloaded site*). The aggregation of the remaining predicted workload of each of these sites is denoted as the *excess workload* $\mathbf{w}_k$ of site s_k, and this is handled through the novel approach that follows.

This second stage aims towards better balancing the previous resource management decisions so that excess workload requests of a site are distributed in neighboring (or even farther apart) sites. The excess workload is handled in such a way that it does not allow sites to become operational for a number of requests lower than a threshold of their total capacity, which will ensure eventually better energy efficiency, as explained in previous subsections. To achieve this, the theory of Markov Random Fields (MRFs) [82] is employed, mainly due to its agile design and straightforward implementation, which allows simple distributed decision-making, while achieving results very close to the optimal ones (and frequently the optimal ones) with very low convergence times.

In this work, the sites $s_k \in S$ are considered, that correspond to access points of the considered infrastructure. A neighborhood system $\mathcal{N} = \{\mathcal{N}_{s_k}\}_{s_k \in S}$ is defined on S, while $\mathcal{N}_{s_k}$ denotes the neighborhood of site s_k and includes the nodes within single hop distance. Assume $\mathbf{w}_k = [w_m^{(k)}]$ is the vector indicating the amount of excess workload for application m at each site s_k and $\mathbf{b}_k = [b_i^{(k)}]$ the vector indicating the number of selected servers of type i, to be additionally activated at site s_k. Considering e_k, the number of available servers per site s_k, which is obtained from the solution of the initial resource optimization problem (6.6a), $\mathbf{b}_k$ is such that

$$\mathbf{b}_k = \left[b_i^{(k)}, \ldots, b_{|\mathcal{Z}_k|}^{(k)}\right], \sum_{i=1}^{|\mathcal{Z}_k|} b_i \leq e_k. \tag{6.9}$$

Vectors $\mathbf{w}_k, \mathbf{b}_k$ are stochastic, since their values depend on the instantaneous system state and user activity. The collection of random variables $\mathbf{X}_k = \{\mathbf{W}_k, \mathbf{B}_k\}_{k=1}^n$ is defined as a collection of random vectors $\mathbf{W}_k = \mathbf{w}_k, \mathbf{B}_k = \mathbf{b}_k, \forall\ k \in [1, n]$, defining the state of each site and cumulatively the state of the system with respect to excess workload and available servers at each site s_k. The random field $\mathbf{X} = \{\mathbf{X}_k\}_{k=1}^n$ takes values $\{\mathbf{X}_k = \mathbf{x}_k\}_{k=1}^n$ in $\mathbf{\Lambda} = \mathcal{W} \times \mathcal{B}$, which is the product space of phase spaces $\mathbf{w}_k \in \mathcal{W}, \mathbf{b}_k \in \mathcal{B}$, respectively. The configuration $\boldsymbol{\omega} = \{\mathbf{x}_k : \mathbf{x}_k \in \mathbf{\Lambda}, \forall s_k \in S\}$ corresponds to one of all possible states of the system state and $\mathbf{\Lambda}$ denotes the configuration space.

Due to the distributed topology of the sites, the above random field $\mathbf{X}$ can be considered an MRF, and based on the Hammersley-Clifford theorem, the potential function $V(\boldsymbol{\omega})$ is considered, which can be decomposed in clique potentials:

$$V(\boldsymbol{\omega}) = \sum_{C \in \mathcal{C}} V_C(\boldsymbol{\omega}) = \sum_{s_k \in S} V_{\{s_k\}}^{(1)}(\boldsymbol{\omega}) + \sum_{s_g \in \mathcal{N}_{s_k}} V_{\{s_k, s_g\}}^{(2)}(\boldsymbol{\omega}), \tag{6.10}$$

where $\mathcal{C}$ is the set of all cliques in the network of sites (where a clique denotes a subset of nodes, all of which are connected to each other). The potential function is the objective function to be minimized, and it will be used as a quantitative measure of the success of each system state to fulfil the optimization criteria, namely the reduction of the total power consumption of the Edge infrastructure. The lower the potential function, the more desired the corresponding system state will be. Due to the topology formed by the sites (i.e., the access points), only one-clique (cliques consisting of one node) and two-cliques (cliques

consisting of pairs only) exist, so that the potential function is decomposed in singleton $V^{(1)}_{\{s_k\}}(\boldsymbol{\omega})$ and doubleton (pairwise) $V^{(2)}_{\{s_k,s_g\}}(\boldsymbol{\omega})$ terms, respectively. Each singleton term is defined as follows:

$$V^{(1)}_{\{s_k\}}(\mathbf{x}_k) = \begin{cases} C_1 \cdot P(\mathbf{b}_k)\left[1 + \sum\limits_{m} \overline{\text{sig}}(w_m^{(k)})\right] + C_2 \cdot d \cdot a_k, & \text{if } \exists\ \mathbf{b}_k \\ & \sum_{i=1}^{|\mathcal{Z}_k|} b_i^{(k)} r_i^m > w_m^{(k)}, \forall m, \\ \Delta_1 > 0, & \text{otherwise,} \end{cases} \tag{6.11}$$

where C_1 and C_2 are properly selected constants and $\Delta_1 > 0$ is a constant with very high value. The power consumption of formation $\mathbf{b}_k$ is $P(\mathbf{b_k}) = \sum_{i=1}^{|\mathcal{Z}_k|} b_i^{(k)} p_i$. Function $\overline{\text{sig}}(\cdot) = L - \frac{L}{1+\exp^{-K(x-x_0)}}$ is the reflection of the sigmoid function with respect to the vertical axis through the inflection point $x = x_0$. The parameters of the reflected sigmoid function are L, the maximum value, K, the gain and x_0, the inflection point. By giving the inflection point a value equal to 0.5 r_i^m, the inclusion of this reflected sigmoid function tends to grow singleton terms that describe states where edge servers are under-utilised (i.e., when they serve less than 50% of their nominal workload capacity), close to the maximum value (undesired system state). The intuition behind this design is that the singleton terms express the goal of each site individually for lower energy consumption. Each site strives to reduce its consumption as much as possible, which in turn will drive its singleton term to lower values. At the same time, the term $d \cdot a_k$ tends to drive the system towards a solution which keeps the total additional delay, induced by the workload redirections, as low as possible; d stands for the single hop network delay in ms while a_k corresponds to the ingress workload (i.e., how much additional workload the edge site s_k will accommodate, compared to the original).

The doubleton terms are defined as follows:

$$V^{(2)}_{\{s_k,s_g\}}(\mathbf{x}_k,\mathbf{x}_g)=\begin{cases} C_3\mathbf{w}_k\cdot\mathbf{w}_g+C_4P(\mathbf{b}_g)\left[1+\sum_m \overline{\text{sig}}(w_m^{(g)})\right], & \text{if } \exists\ \mathbf{b}_g \\ & \sum_{i=1}^{|\mathcal{Z}_k|} b_i^{(g)} r_i^m > w_m^{(g)}, \\ & \forall m \\ \Delta_2>0, & \text{otherwise,} \end{cases} \tag{6.12}$$

where C_3 and C_4 are properly selected constants and $\Delta_2 > 0$ is again a constant with very high value. The intuition behind the design of the doubleton terms is that as far as the interactions of the neighboring sites are concerned, ideally the system should be driven to states where neighboring sites exchange the remaining workload so that it is concentrated in specific sites, thus avoiding having to maintain multiple active sites for a small value of excess workload. It is also important to point out that the MRF activates servers with the appropriate VMs of the flavors described in Subsection 6.3.3 in order to serve the balanced excess workload, thus the primary QoS requirement of the maximum acceptable response time is respected. An overview of the MRF balancing process is depicted in Figure 6.3.

Each site seeks to minimize its contribution to the cumulative potential function by minimizing its local neighborhood potential function comprised of the sum of its singleton and doubleton (pairwise) potentials with its one-hop neighbors. The state of each site depends only on the states and the information of its neighbors. Gibbs sampling [186] can be applied by each site individually, reaching global optima through local sampling. Cumulatively, this distributed sampling converges to global optimizers of the system. This approach has a very low computational overhead, $O(n)$, n being the number of sites, while reaching asymptotically the global optimal resource allocation solutions, frequently yielding the optimal ones. Furthermore, the signaling overhead is rather small, since each site s_k is only required to exchange system state information locally with its one-hop neighbors only.

The sequential Gibbs sampling method proceeds as follows. Consider a logarithmic annealing schedule of the form $T(w) = \frac{c_0}{\ln(1+w)}$, where c_0 is a constant (equal to 2 in the experiments) and $T(w)$ is called the "temperature" of the w-th annealing step. Also, consider

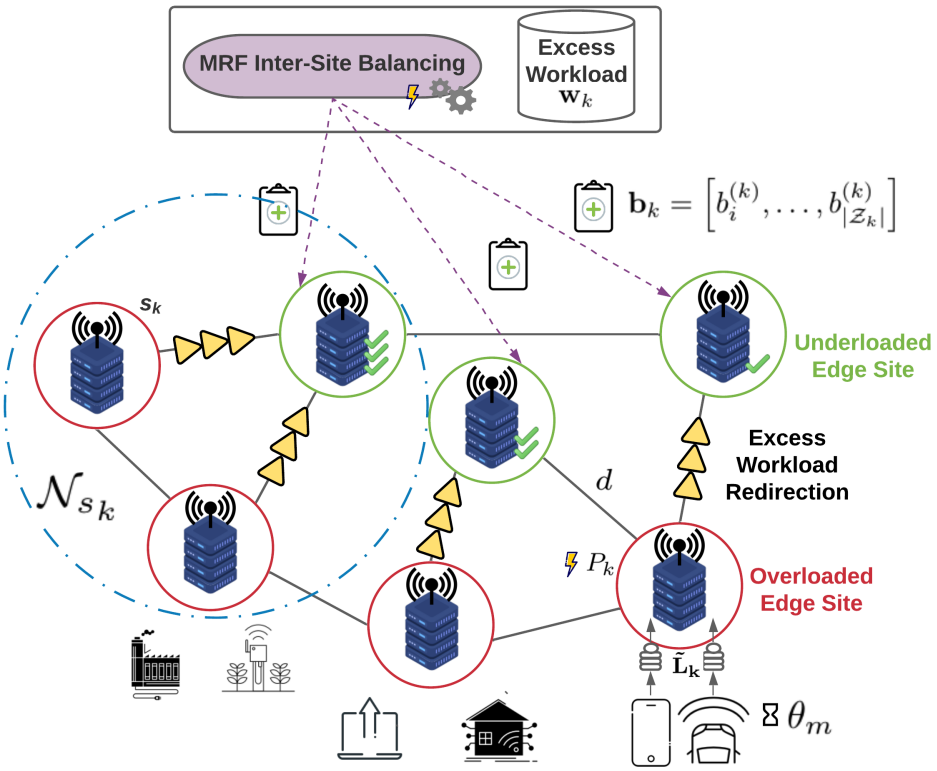


Figure 6.3: MRF Inter-Site Balancing Overview (Stage 2).

a sequential visiting scheme of all sites, where at each epoch t (mini-slot in a sweep) within a step w, only one site updates its value (Figure 6.4 depicts the relations of the system slots, sweeps and update epochs). Starting with an arbitrary initial configuration $\mathbf{X}(w = 0)$, at epoch t of w, let $\boldsymbol{\omega} = \mathbf{X}(t)$ and denote by $\boldsymbol{\omega}^{\mathbf{x}_k}$ the configuration that has value $\mathbf{x}_k$ at site s_k and agrees with $\boldsymbol{\omega}$ everywhere else. The update (decision to transition to a new state) at

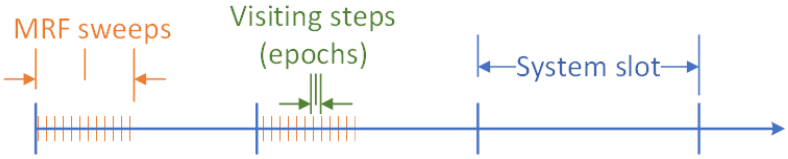


Figure 6.4: Relation of system slots, sweeps and update epochs.

site s_k takes place according to the distribution:

$$P(\mathbf{X}_k(t) = \mathbf{x}_k | \mathbf{X}_g(t) = \mathbf{x}_g, g \neq k) = \frac{\exp\left(-\frac{1}{T(w)} \sum_{C:s_k \in C} V_C(\boldsymbol{\omega}^{\mathbf{x}_i})\right)}{\sum_{\mathbf{x}_k \in \boldsymbol{\Lambda}} \exp\left(-\frac{1}{T(w)} \sum_{C:s_k \in C} V_C(\boldsymbol{\omega}^{\mathbf{x}_s})\right)}, \quad (6.13)$$

where C is the set of the cliques formed by the sites (here only one-clique and two-cliques are formed in the graph). Namely, with probability determined by (6.13), site s_k will choose $\mathbf{x}_k$ as its state in sweep $w+1$. The site states are updated sequentially within a sweep w. The annealing schedule represents a decreasing rate of system temperature $T(w)$, where w stands for the index of the w-th sweep (i.e., the system temperature is updated at the end of each sweep). The w-th annealing step is equivalent to the w-th sweep, and consists of n visiting epochs (denoted by t in the above), one for each site. Since sampling begins at high temperatures, where the local characteristics are practically uniform, it permits transitions to higher-potential function configurations, thus avoiding getting trapped in local minima. After each sweep, the resulting system states form an inhomogeneous Markov Chain on the configuration space that converges to the uniform distribution on the set of global potential function minimizers.

Figure 6.5 showcases an example of the effect of the MRF excess workload balancing, for two applications in an Edge infrastructure of nine sites, by comparing the starting and final state where the MRF has converged. As the starting formation for each site, the set of edge servers with the minimum number of allocated resources is selected in order to serve the excess workload locally. It can be observed that in the final state, the MRF yields a rather desired solution where it has grouped all the excess requests, w_k, in a single site, thus minimizing the associated energy consumption of the network, while serving properly the remaining requests, within the capacity bounds imposed in each site. Specifically, Table 6.2 shows the selected VM formation for the particular site, with three activated servers.

Table 6.2: VM formations selected by the MRF mechanism.

Server ($\mathbf{b}_k$)	**App1 VMs**	**App2 VMs**
1	1 × medium	1 × small
2	1 × medium	1 × small
3	1 × medium	-
Site Workload Capacity $(\sum_{i=1}^{\lvert\mathcal{Z}_k\rvert} b_i^{(k)} r_i^m)$	81	82

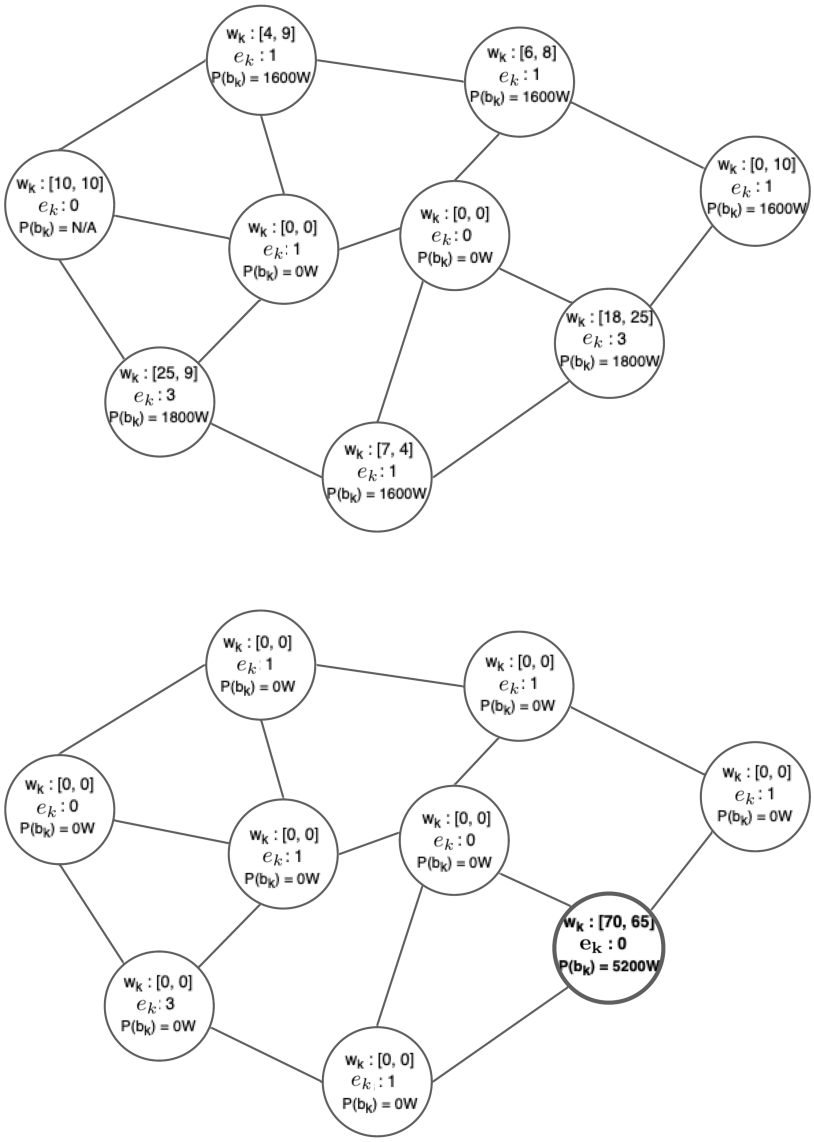


Figure 6.5: Workload Balancing example: starting and final states.

It is observed that this site formation fits to accommodate the workload. The total power consumption, $P(b_k)$, is $5200W$, which is around half of the $10000W$ power consumption of the initial site formations selected, had the excess workload been executed locally. The number of available servers per site e_k, is also depicted. Also, local execution would lead to some requests being rejected, as there is one site that has no available servers to accommodate its excess workload. Consequently, the MRF based mechanism emerges as rather effective in increasing the energy efficiency of the whole approach.

6.4.3 ENERDGE Core Algorithm

In this subsection, the core algorithm of a full ENERDGE deployment in an edge infrastructure is described:

Algorithm 1 ENERDGE Core Algorithm.

Data: Trajectory Dataset
Result: Optimal VM placement in Edge Infrastructure

begin
  `// Offline`
  1: create the Task Offloading Dataset, Sec. (6.3.2)
  2: **while** $\tau \leq identificationPhaseDuration$ **do**
    for $m \in M$ **do**
      for $c \in C^{ser}$ **do**
        `// Identify VM flavors`
        $\phi_m \longleftarrow$ solve Eq. (6.2)
      end
    end
  end
  3: create the Transition Matrix, Sec. (6.3.5)
  `// Online`
  4: track last position of users, Sec. (6.3.5)
  5: **for** $s_k \in S$ **do** `// Optimization`
    $\mathcal{Z}_\mathbf{k} \longleftarrow$ calculate VM formations, Eq. (6.5)
    for $m \in M$ **do**
      $\tilde{L}_k^m \longleftarrow$ predict workload, Sec. (6.3.5)
    end
    $\tilde{\mathbf{L}}_\mathbf{k} = [\tilde{L}_k^m]$ place VMs by solving, Eq. (6.6)
  end
  6: **for** $s_k \in S$ **do** `// MRF`
    $\mathbf{w}_k \longleftarrow$ calculate excess workload, Sec. (6.4.1)
    repeat
      $\mathbf{b}_k \longleftarrow$ calculate additional servers, Eq. (6.10)
    until *converges*
    activate extra servers, Sec. (6.4.2)
  end
  wait until next *system slot*
  go to 4
end

At first, the required datasets are produced and the VM flavor design procedure is performed offline. Then, as shown in Algorithm 1, the initial optimization and the distributed resource allocation for each site of the edge infrastructure take place, as explained in the previous sections. During this online phase, first, the mobility of users and devices is predicted using the n-MMC method. Then, the incoming workload at each site of the infrastructure

is estimated for the current system slot. The resource allocation optimization produces an initial solution subject to QoS and energy constraints for a given predicted workload at each site. For each site, the excess predicted workload or workload that cannot be served, along with the available resources, are computed. Finally, the excess workload is balanced between the extra servers activated in under-loaded sites, according to the MRF solution, achieving the minimization of the energy consumption for the edge infrastructure.

6.5 Performance Evaluation

In this section, the performance of the proposed resource allocation and load balancing mechanism is presented via modeling and simulation. The results illustrate the success of the proposed approach in minimizing the energy consumption while guaranteeing the stability of the application's QoS (i.e., response time) within an acceptable margin. The optimization of the resource allocation is highlighted in terms of the power consumption of the activated edge servers and the VM flavors used to serve the incoming workload. The benchmarking is conducted using CloudSim Plus [110], a Java-based simulator suitable for Edge and Cloud environment experimentation. Then, a comparison with a well-established study in the literature follows.

6.5.1 Smart Museum Experiment Setting

To demonstrate the operation of an ENERDGE real-world application, the environment of a smart museum is emulated, accommodating different categories of interactive exhibits, a large number of IoT sensors, edge devices with heterogeneous computational capabilities and dynamic network conditions modeled by the dynamic behavior and mobility of the users. In particular, the physical infrastructure consists of interactive exhibits-sites, each of which hosts an edge data center, resembling a smart museum floor. The applications deployed in the museum are classified in two categories with different characteristics and requirements: **Interactive Exhibit Apps:** On the one hand, the museum is considered to be leveraging Augmented Reality (AR) and Virtual Reality (VR) settings to provide rich and detailed access to artwork and artifacts, bring life to works of art and allow visitors to engage in adaptable visual guided tours by using their mobile devices. In order to achieve the high

QoS requirements of these types of applications, mobile devices can offload some workload by sharing video decoding tasks to the more powerful edge devices. Mobility is high in these applications as visitors move from one exhibit to the other.

Sensor Monitoring Apps: On the other hand, IoT is making it possible to deploy low-cost, automated monitoring of collections and museum facilities, e.g., static sensors for temperature, humidity, counting number of visitors. Such applications are low on delay requirements, i.e., the processing can be performed in a delay tolerable manner, sending data and information after a completion of an activity. However, they produce numerous requests to the edge servers.

Table 6.3: Identified VM flavors.

Flavor	**Small**		**Medium**		**Large**	
	App1	**App2**	**App1**	**App2**	**App1**	**App2**
Cores	1	1	2	2	4	4
QoS (sec)	3	3	3	3	3	3
Maximum Requests/Slot	11	38	27	82	59	173

One application of the Interactive Exhibit type is assumed, denoted as *App1*, and one of the Sensor Monitoring type, denoted as *App2*, co-hosted in each site. This means that VMs of both application types are able to run simultaneously in the edge servers, receiving offloading requests from their counterparts in the visitors' mobile devices and the IoT sensors, respectively. For demonstration purposes, both apps are assumed to be based on image recognition processes, thus their acceptable response time (QoS) is set at $3sec$, which lies within the margins of a typical image recognition service time [204] and provides a satisfying Edge Computing AR application experience to the user [205]. As the design of the proposed framework and modeling of the applications are independent of the level of the applications QoS requirements, applications that require lower (or higher) response times are naturally supported. Following the modeling approach explained in Subsection 6.3.3, the VM flavors shown in Table 6.3 are identified, tuned towards achieving the above QoS requirement. It should be noted here that *App1* requests require considerably heavier computations to achieve this response time than the ones of *App2*, a fact that limits the Maximum *App1* Requests served per Slot to a third of those served by the *App2* equally sized VMs. The system slot is arbitrarily set at $30sec$ and the experiments last for a period of 1 hour, or

120 system slots. The simulation code alongside any related dataset used in this section is publicly available[2].

6.5.2 Resource Allocation Evaluation

In this subsection, the evaluation of the resource allocation algorithm is presented. At first, the impact of the selected mobility method is assessed and then a summary of the core optimization results is provided. Finally, a comparison with a well-known work in the field is demonstrated.

Mobility Prediction Impact

As described in Subsection 6.4.3, predicting the visitors' positions in the next system slot is the first step of optimizing the allocation of the edge resources in each site. This provides an estimation on the projected workload. To quantify the impact of the mobility prediction accuracy, a sensitivity analysis is performed as illustrated in Figure 6.6; this assesses the impact of the prediction error on satisfying the required application QoS, both in terms of the average response time (ART) per request and the percentage of the violations occurred in respecting the QoS. Logarithmic scale is used to better visualize both impacts in a combined fashion.

Showcasing the impact analysis at the end of both Stages of the resource allocation mechanism separately, was preferred, so as to highlight the significant effect the MRF-based workload balancing has on alleviating the disruptions caused by the prediction error. The results are collected from running the simulation for $10,000$ system slots, for various topologies, and averaging the stats in batches of 10. Thus, the x axis of Figure 6.6 represents the range of the prediction error. The dataset used is again the Melbourne Museum one [203].

Underestimating the real incoming workload leads to under-provisioning of resources and subsequently to slight degradation of the response time. In detail, it is noticed that both the ART and the violations grow almost linearly with the prediction error. It is also clear that the application of the MRF-based balancing in each system slot has a great impact on respecting the QoS requirements, with the redirections of the excess projected workload from

[2] `https://github.com/maravger/netmode-cloudsim`

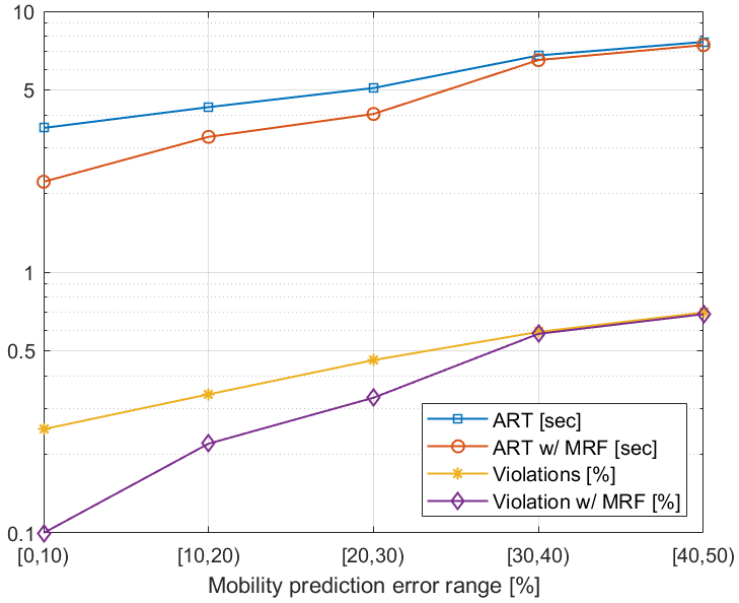


Figure 6.6: ART & QoS violations sensitivity to prediction error.

overutilised sites to underutilised ones; the ART lies around $2sec$ and the QoS violations do not exceed 10% of the offloaded requests, when the prediction error is less than 10%, while the ART grows to around $3sec$, which is still acceptable for both applications, and the violations to 20%, when the error is less than 20%. Beyond the point of a 30% prediction error, it is noticed that the extra, unpredicted workload puts excessive strain on the mechanism. However, this should not be a problem, as selecting an appropriate prediction mechanism, like the n-MMC used here and other comparable works, e.g., [206], leads to an average prediction accuracy of $70 - 95\%$.

Response to Dynamic Network Conditions

In this subsection, a close examination on how the resource allocation optimization reacts to the dynamic workload demands caused by the visitors' mobility is made, in terms of edge servers activated and the VMs placed in them. Figure 6.7 showcases the scalability of the proposed technique, as a response to the mobility of the visitors' devices and the fluctuations in the sensors' offloading rate. The behavior of a single site is presented, which is equipped with three servers of four cores each, and this acts as a baseline for the rest of the evaluation. With regard to power consumption, for demonstration purposes, it is assumed

that the average maximum power consumption of an edge server is $2000W$, in accordance to [207].

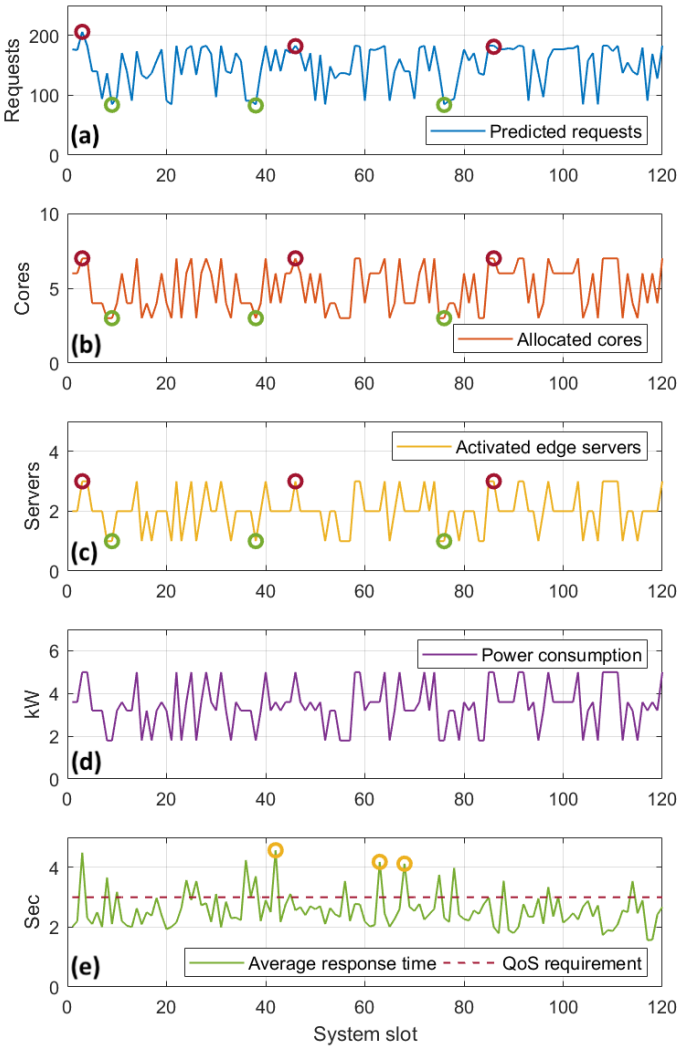


Figure 6.7: Dynamic resource allocation: allocated cores, activated edge servers, power consumption and ART as a response to the predicted requests, for a single site.

Figure 6.7a shows the predicted workload per system slot, as calculated in the previous step, while Figs. 6.7b-d demonstrate how the resource optimizer adapts to the fluctuations. In particular, they depict how the optimizer selects the appropriate topology in terms of number of active edge servers and their allocated cores, in order to meet the demands for the selected site. For instance, when the predicted requests are high, e.g., at system slots $\{3, 46, 86\}$, with $\{206, 182, 181\}$ predicted requests respectively for both applications (red-

colored marks), the optimization results in three activated edge servers and seven cores allocated among them. On the other hand, when the incoming request prediction is considerably lower, as in system slots $\{9, 38, 76\}$, with $\{84, 83, 84\}$ predicted requests respectively (green-colored marks), only one server with three allocated cores is activated. The results corroborate the total power consumption, as shown in Figure 6.7d.

Exploring further, an example is demonstrated regarding the specific VM formations selected for the above activated servers, at system slot 3. The total of 206 predicted requests consisted of 17 requests for *App1* and 189 requests for *App2*. Table 6.4 shows the selected VM formation for the three activated servers for this system slot. It can be seen that this VM formation fits to accommodate the predicted workload. The site's power consumption, in this slot, is $5000W$.

Table 6.4: VM formations in slot 3.

Server	**App1 VMs**	**App2 VMs**	**Allocated Cores**
1	1 × small	1 × medium	3
2	1 × small	1 × medium	3
3	-	1 × small	1
Site Workload Capacity	22	202	

While the proposed approach adapts very well against the various predicted incoming workloads in terms of allocated resources, satisfying the QoS for these applications is challenging. This is due to the fact that the VM topology to serve these requests is selected based on the predicted workload which is potentially fallacious, as explained in the previous subsection, and this leads to violations in the QoS. For instance, as shown in Figure 6.7e, in system slots $\{42, 63, 68\}$ (yellow-colored marks), the average response time for both applications was slightly above $4sec$, or approximately 35% larger than the reference value, set at $3sec$. This is an indication of under-provisioning due to incoming workload underestimation. Violations like this took place 17 times in this site, or 14% in a total of 120 system slots. This is considered to be an acceptable margin of error for the satisfaction of the perceived QoS. Finally, it should be pointed out that for this experimentation, the average service completion time mainly affected the measured response time. The average transmission time is negligible, due to the use of the IEEE 802.11ac standard, which provides high throughput for requests of application types used in this experiment.

MRF-based Excess Workload Balancing Evaluation

In this subsection, initially the convergence behavior of the MRF approach is demonstrated for a standard and a larger topology. Figure 6.8 demonstrates the variation of the cumulative potential function of the MRF (Eq. (6.10)) for a complete set of sweeps corresponding to an execution of the MRF in the beginning of a system slot. The results of this evaluation have been averaged over 100 different topologies, both for a 9-site (Medium) and a 36-site (Large) Edge infrastructure.

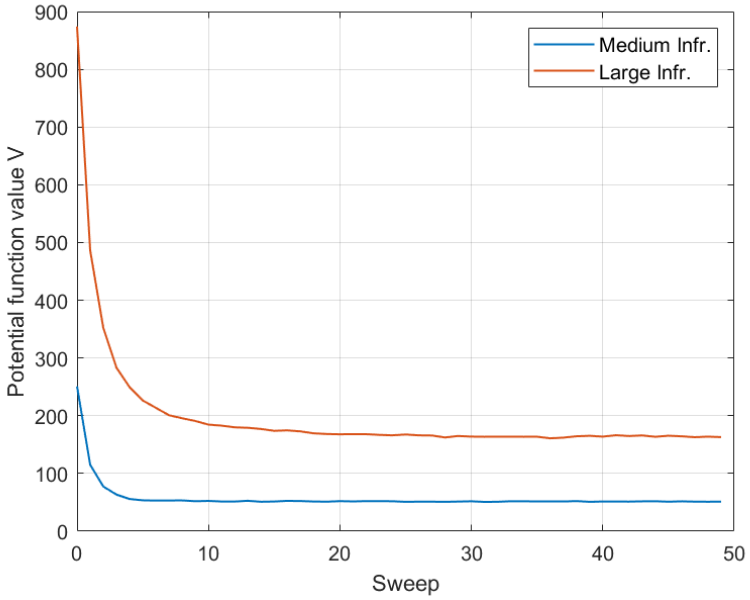


Figure 6.8: MRF-based workload balancing convergence.

It is observed that the Gibbs sampler converges rather quickly and it succeeds in reducing the variability of the potential value rapidly. The Gibbs sampler is able to identify the local neighborhood of desired solutions relatively fast, within the first five sweeps, and then fine-tune the search, eventually selecting one solution among the global minimizers of the potential function. As expected the larger topology exhibits greater variability of the cumulative potential function in the first sweeps, but eventually convergence is smooth and within the maximum number of designated sweep iterations (here employing a maximum of 50 sweeps).

To evaluate the efficiency of this second stage of the proposed mechanism, as discussed

in Section 6.4.2, two cases of excess workload at the end of the first stage are identified. Regarding the workload coming from overloaded sites, Figure 6.9 depicts the improvement in the QoS satisfaction that comes with the application of the MRF balancing. It is observed that, while both the ART and the violations metrics grow almost linearly with the excess workload, by applying the MRF balancing, the proposed mechanism achieves to provide the QoS guarantees (i.e., ART $\leq 3sec$ and violations $\approx 10\%$). This comes as a natural result as the overloaded sites are alleviated of the excess workload, which is balanced throughout the infrastructure.

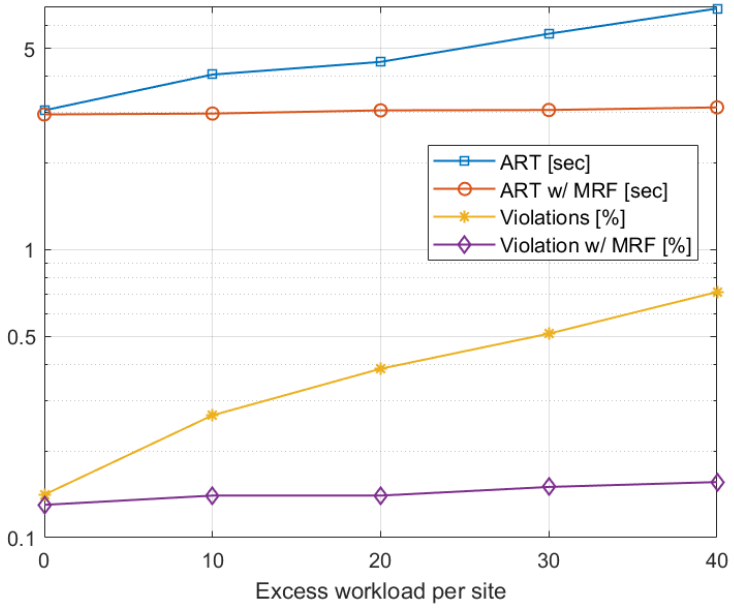


Figure 6.9: MRF QoS improvements for various excess workloads in overloaded sites.

On the other hand, regarding the underloaded sites, Figure 6.10 demonstrates the effect of the MRF-based excess workload balancing on the total energy consumption of the infrastructure, by comparing it to the case where no balancing of any kind takes place. During the latter, as the average excess workload increases, the power consumption increases radically, as extra, underloaded edge servers are activated in each site in order to accommodate the low volume of excess requests locally. From that point on, power consumption increases moderately, as larger VMs are installed to meet the increasing workload demands, until the point where all the resources are allocated in each site and the maximum power consumption of the infrastructure is reached. On the contrary, when MRF balancing is employed,

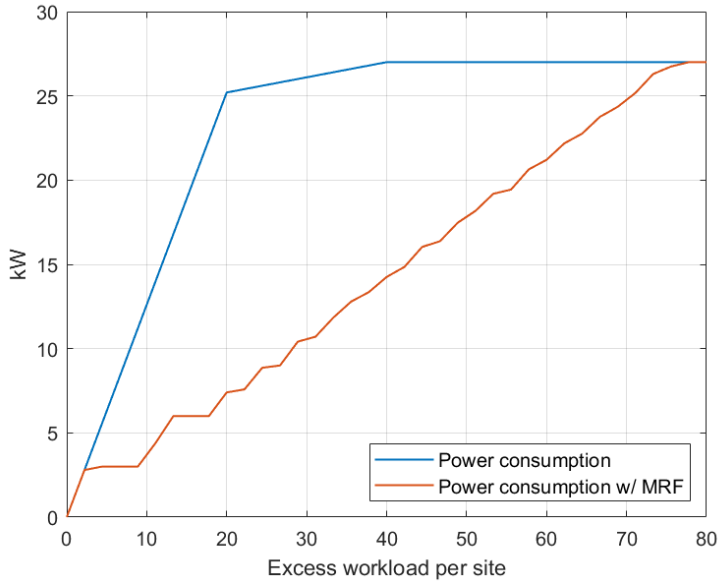


Figure 6.10: MRF Energy Savings for various excess workloads in underloaded sites.

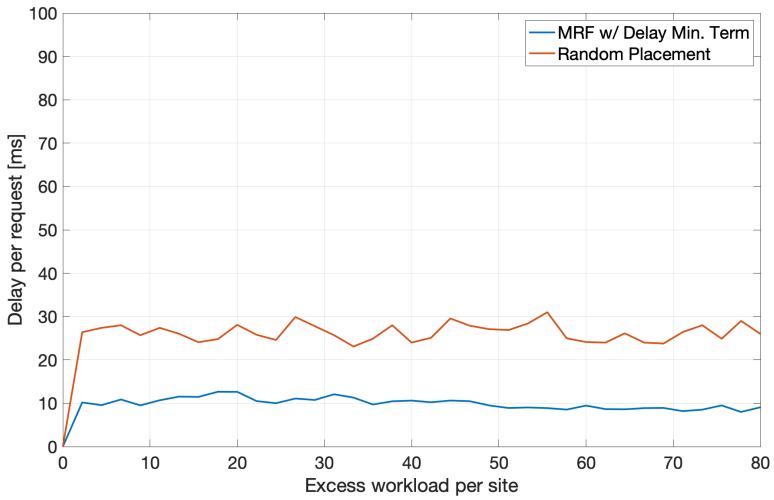


Figure 6.11: MRF Workload Redirection -Induced Delay Minimizing for various excess workloads.

power consumption adjustment is more fine-grained, as only the minimum combination of activated servers and installed VMs flavors are deployed in each case.

Finally, Figure 6.11 illustrates the impact of the delay minimizing term in the MRF-based workload balancing. It is clear that the MRF-based solution minimizes the redirection-induced overhead per request ($\approx 10ms$ average), when compared to a solution that randomly

places the excess workload to available edge sites ($\approx 26ms$ average) in a medium sized edge infrastructure. It should be also noted that the average additional delay is far more stable throughout the average excess workload increase, compared to the random placement case.

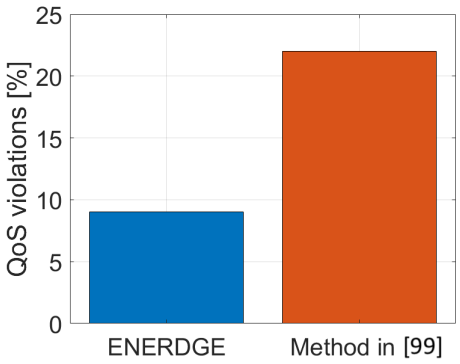
6.5.3 Comparison

Following, a comparative evaluation of the overall resource allocation of ENERDGE with the work presented in [99] is carried out. Similar to the proposed work, this study presents a setting of dispersed and interconnected clusters of computers, namely *cloudlets*, which form a wireless metropolitan area network. Contrary to ENERDGE, each cloudlet has a static VM provisioning method to serve offloaded requests. This study focuses on identifying over-utilized cloudlets and redirecting part of their incoming workload to under-utilized ones in order to achieve load balancing. In [99], instead of having an estimation of the incoming workload, it is considered known for each cloudlet and for each system slot. The offloaded workload served at each cloudlet is bounded by its service rate capabilities. The rest of it is rejected and redirected back to the end-device for local execution. The service rate of a cloudlet is defined as the number of requests that each VM can serve in a system slot.

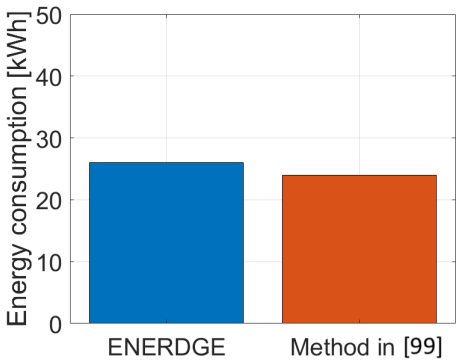
In order to highlight the importance of dynamic resource allocation towards simultaneously guaranteeing the QoS requirements and minimizing energy consumption, the proposed method is compared with two differently oriented resource provisioning settings of [99]. The first, attempts to minimize energy consumption (Experiment A), while the second aims at satisfying the QoS (Experiment B). To make the comparison fair, the exact same nine-site edge infrastructure, described in Subsection 6.5.1, is simulated for both methods. The generated workload traffic is the same for both methods as well.

Regarding Experiment A, a frugal static resource allocation for [99] is chosen, that would approximately match the total energy consumption of ENERDGE (Figure 6.12b). QoS violations were calculated for both methods based on the SLA threshold for the response time of the offloaded requests, set at $3sec$, as in Subsection 6.5.2. In one hour of experimentation, the ENERDGE sites reported 207 violations, or 9% of the offloaded requests, compared to the 470 violations or 22% of the requests in [99], as shown in Figure 6.12a.

On the contrary, in Experiment B, a resource-abundant static allocation was selected for [99], in order to match the QoS satisfaction of ENERDGE (Figure 6.13a). In this case, as

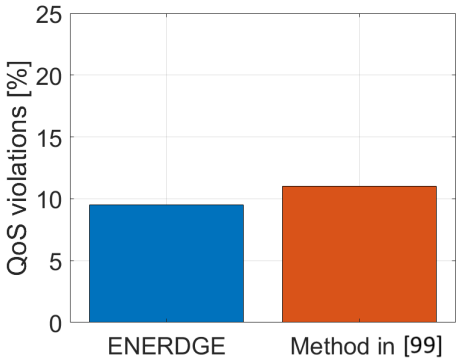


(a) QoS Violations.

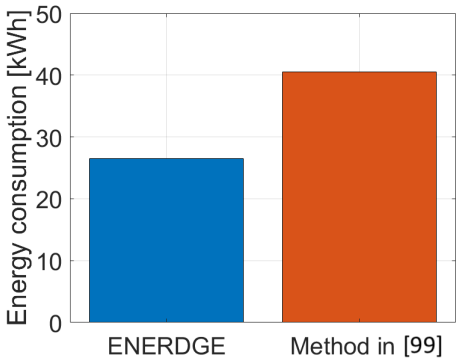


(b) Energy Consumption.

Figure 6.12: QoS violations and energy consumption during Experiment A.



(a) QoS Violations.



(b) Energy Consumption.

Figure 6.13: QoS violations and energy consumption during Experiment B.

shown in Figure 6.13b, energy consumption for one hour in [99] was roughly $41kWh$, or more than 65% bigger when compared to the $26.5kWh$ of the proposed method. In addition to the previous results, it is clear that even a static resource provisioning method enhanced with workload redirecting mechanisms is incapable of finding a balance between QoS satisfaction and infrastructure energy consumption minimization, the way ENERDGE does.

Chapter 7

Conclusion

In this study, multiple aspects for designing resource allocation mechanisms and computational offloading strategies have been studied, that will allow the network edge to become a reliable ally for IoT devices in their quest for transforming the world as we know it. In order to do so, the interdependency among various different types of actors has taken into account. Key problems have been identified, novel algorithms have been developed and their performance has been evaluated.

7.1 Outcomes

At first, the basic mechanism on which most of the work in this thesis is based, was introduced. Specifically, in **Chapter 3**, a cooperative, two-level computation offloading mechanism for IoT/mobile applications was presented. The VM operation was modeled by a group of LTI models and for each model an equilibrium operating point, a proper controller and the minimal and maximal positive invariant sets were computed. At the upper level a horizontal scaling procedure took place; an optimizer determined the number of active edge servers and the operating points of the VMs to be implemented in them, in order to serve the total workload for each application. This decision took into consideration the calculated equilibrium points for each underlying VM, thus guaranteeing the scalability of the presented mechanism towards major workload fluctuations. At the local level, a controller handled the minor workload fluctuations by scaling the VMs vertically, ensuring that the

average response time was stabilized and restricted in a specific range of values. The experimental evaluation showed that the proposed mechanism achieved high percentage of requests admitted in the edge servers, while the performance constraints were met, outperforming a well established energy aware offloading method.

Then, based on this work, in **Chapter 4**, a Cyber-Physical Social System for early fire detection was presented; a Scalable Edge Computing Framework, called SMOKE, was deployed on the network edge and was receiving captured images from several IoT nodes to evaluate the criticality of the situation; on a higher level, an intelligent decision-making service was deployed on the Cloud, receiving data from various sources (i.e., various low cost sensors, social media users and SMOKE) and communicated with the local authorities in case of emergency. This work focused on two main aspects, namely (a) the Horizontal and Vertical scaling of the available edge servers' resources, in order to achieve optimal allocation and use of resources and (b) the decision-making mechanism, for a time-critical application, that takes the social factor into consideration. The proposed computation offloading mechanism is generic and applicable on several types of Mobile Edge Computing (MEC) environments and applications.

The experiments conducted and presented in these two first chapters allowed for drawing some significant conclusions regarding the performance of the proposed frameworks. Horizontal and Vertical Scaling of edge servers is essential for guaranteeing the QoS metrics of time- and mission- critical applications, while dynamic resource allocation prevents over- or under- provisioning of the edge servers' resources. Moreover, admission control on the incoming offloaded requests is a key factor for time-critical applications with stringent requirements in terms of retention of the desired QoS levels, like average task execution and transmission latency. Specifically in Chapter 4, the evaluation with regards to the time needed for the IDM component to collect the necessary data and extract a decision, shows that the overall time overhead is not higher than 40 *sec.* This time duration is considered within the time limits of such a time-critical system especially considering the heterogeneity of the collected information types and sources.

Next, in **Chapter 5**, computation offloading strategies were investigated and a switching offloading mechanism for localization and path planning applications of mobile robots was introduced. The offloading decision for localization was based on pose uncertainty and the

availability of edge (network and computing) resources, while the offloading decision for path planning depended on the difficulty of the trajectory. The proposed framework is shown to achieve more precise navigation than the case of exclusive local execution of the applications, without paying the price of a slower execution time, like in the case of remote only execution of the algorithms. Also, it is modular and applicable to various scenarios, applications and objectives under the robot's dynamic environment.

Apart from that, a vision-based self localization approach for indoor autonomous mobile robots was proposed. Based on a bilateration method and some core principles of the projectile geometry, the proposed algorithm required the detection of distinct landmarks in the environment and the calculation of the robot's relative distance from them in order to obtain the robot's pose (i.e., position and orientation). Distance calculation was based on feature extraction from the landmarks. The localization algorithm had to rely on the minimum number of landmarks as they are scarce in the discussed application's setting, thus a bilateration approach that required the identification of two landmarks was used.

Finally, **Chapter 6** introduced the ENERDGE framework that addressed jointly the full task offloading and resource allocation problems in a multi-site setting. A holistic energy-aware resource optimization approach was proposed, based on the design of the VM flavors complemented with an innovative distributed load balancing technique based on MRFs, with the penultimate goal to minimize the total energy consumption without sacrificing the QoS in terms of latency. To minimize the inverse impact of user mobility during task offloading, ENERDGE considered the dynamic wireless conditions of the access network and supported a mobility prediction scheme to better guide the allocation solution. Numerical results showed that the prediction mechanism accurately predicts the mobile behavior of the users, while the ENERDGE resource optimizer outperforms a well-established load balancing technique in terms of both latency and energy consumption. The MRF scheme is shown to converge rapidly to minimum energy solutions, thus allowing further energy optimizations in an efficient manner.

7.2 Recommendations for Future Work

Concluding this thesis, this final section sheds light on some of the possible future research directions that can be followed based on the outcomes of the presented work and the challenges faced during the research process. While the thesis treats resource allocation and computational offloading problems that concern some of the most dynamic procedures in IoT environments, there still exists much room for further development.

Regarding the resource allocation mechanism, future work could focus on further investigating improvements on the modeling and control of the application-specific VMs and leveraging different combinatorial optimization criteria to improve the Horizontal Scaler's decision-making mechanisms, introduced in Chapters 3 and 4.

- Specifically, it should be noted that, as mentioned before, this work aimed at minimizing the number of active servers with the constraint of meeting the total workload demands. By offloading as many tasks as possible, while keeping the number of active servers low, energy efficiency on the mobile nodes and the edge servers is implicitly targeted as well. However, dealing explicitly with optimizing energy cost in the mobile nodes (e.g., maximizing the offloaded requests) is also an interesting and challenging problem.

- Additionally, minimizing functional costs like data transmission costs (e.g., how the requests are distributed among the servers), or maximizing revenue/income for the infrastructure providers (e.g., how many different VM-applications can be deployed per server) can be used as additional or alternative objectives for the Horizontal Scaler component of the proposed framework.

- Another issue that would be interesting to study is the use of the 5G or even 6G technologies as an alternative to the WiFi access points currently used for the communication between the IoT nodes and the Edge servers. This would allow the proposed frameworks to expand their operational range beyond indoor and semi-rural areas and enable the coverage of vast areas. Consequently, this would arise interesting challenges like workload balancing among geographically dispersed edge server clusters. Additionally, evaluating and conducting a comparative study on trying to minimize

the data transmission overhead, would be essential for a complete study in IoT-based applications.

Concerning the computational offloading strategies, future work could focus on extending the proposed algorithms to more sophisticated control ones, providing theoretical guarantees for stability and convergence of the devices' dynamics.

- In detail, developing more precise estimation and planning algorithms, specifically in multi-robot scenarios, and more sophisticated control algorithms in the co-design setting, that would manage and allocate the available networking resources on the infrastructure side as well, is of great interest as well.
- Effort should also be placed on the use of more dynamic environments (e.g., servers and robots moving at the same time) and designing dynamic allocation mechanisms that will manage the available resources appropriately, in order to provide theoretical guarantees for stability.
- Additionally, the establishment of more sophisticated estimation/control algorithms in a co-design setting should propose faster and more reliable planning strategies.
- Last, non-deterministic/stochastic approaches could be evaluated for estimation and control purposes and machine learning techniques could be integrated to the mobility prediction approaches to enable addressing errors in the predictions of dynamically estimated values of the position and number of the involved devices.

Appendix A

Author's Publications

International Peer Reviewed Journals

- **Avgeris, M.**, Spatharakis D., Dechouniotis D., Leivadeas A., Karyotis V. and Papavassiliou S., 2021. ENERDGE: Distributed Energy-aware Resource Allocation at the Edge. *IEEE Transactions on Network and Service Management.* (under review)

- Spatharakis D., **Avgeris, M.**, Athanasopoulos, N. and Papavassiliou S., 2021. Resource-aware Estimation and Control for Edge Robotics: a Set-based Approach. *IEEE Internet of Things Journal.* (under review)

- Saeik, F., **Avgeris, M.**, Spatharakis, D., Santi, N., Dechouniotis, D., Violos, J., Leivadeas, A., Athanasopoulos, N., Mitton, N. and Papavassiliou, S., 2021. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Computer Networks, 195*, p.108177.

- Papathanail, G., Pentelas, A., Fotoglou, I., Papadimitriou, P., Katsaros, K.V., Theodorou, V., Soursos, S., Spatharakis, D., Dimolitsas, I., **Avgeris, M.** and Dechouniotis, D., 2020. MESON: Optimized Cross-Slice Communication for Edge Computing. *IEEE Communications Magazine, 58(10)*, pp.23-28.

- **Avgeris, M.**, Dechouniotis, D., Athanasopoulos, N. and Papavassiliou, S., 2019. Adaptive resource allocation for computation offloading: A control-theoretic approach. *ACM Transactions on Internet Technology (TOIT), 19(2)*, pp.1-20.

- **Avgeris, M.**, Spatharakis, D., Dechouniotis, D., Kalatzis, N., Roussaki, I. and Papavassiliou, S., 2019. Where there is fire there is smoke: a scalable edge computing framework for early fire detection. *Sensors, 19(3)*, p.639.

- Kalatzis, N., Routis, G., Marinellis, Y., **Avgeris, M.**, Roussaki, I., Papavassiliou, S. and Anagnostou, M., 2019. Semantic interoperability for iot platforms in support of decision making: an experiment on early wildfire detection. *Sensors, 19(3)*, p.528.

International Conferences

- Saeik, F., Violos, J., Leivadeas, A., **Avgeris, M.**, Spatharakis, D., and Dechouniotis D., 2021, September. User association and behavioral characterization during task offloading at the edge. In *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (IEEE MeditCom2021)*. IEEE.

- Dimolitsas I., **Avgeris, M.**, Spatharakis, D., Dechouniotis D., and Papavassiliou S., 2021, September. Enabling industrial network slicing orchestration: A collaborative edge robotics use case. *In 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (IEEE MeditCom2021)*. IEEE.

- Spatharakis, D., **Avgeris, M.**, Kakkavas G., Papadakis-Vlachopapadopoulos k., Dimolitsas I., Dechouniotis D., Karyotis V., and Papavassiliou S., 2021, July. Industrial robotics experimentation over federated next generation internet testbeds. *In 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom): Demo Sessions (IEEE MeditCom 2021 - Demos)*. IEEE.

- Spatharakis, D., **Avgeris, M.**, Athanasopoulos, N., Dechouniotis, D. and Papavassiliou, S., 2020, November. A Switching Offloading Mechanism for Path Planning and Localization in Robotic Applications. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)* (pp. 77-84). IEEE.

- **Avgeris, M.**, Spatharakis, D., Athanasopoulos, N., Dechouniotis, D. and Papavassiliou, S., 2019, August. Single Vision-Based Self-Localization for Autonomous Robotic

Agents. In *2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)* (pp. 123-129). IEEE.

- Karyotis, V., **Avgeris, M.**, Michaloliakos, M., Tsagkaris, K. and Papavassiliou, S., 2018, October. Utility decisions for QoE-QoS driven applications in practical mobile broadband networks. In *2018 Global Information Infrastructure and Networking Symposium (GIIS)* (pp. 1-5). IEEE.

- Kalatzis, N., **Avgeris, M.**, Dechouniotis, D., Papadakis-Vlachopapadopoulos, K., Roussaki, I. and Papavassiliou, S., 2018, June. Edge computing in IoT ecosystems for UAV-enabled early fire detection. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)* (pp. 106-114). IEEE.

- **Avgeris, M.**, Kalatzis, N., Dechouniotis, D., Roussaki, I. and Papavassiliou, S., 2017, September. Semantic Resource Management of Federated IoT Testbeds. In *International Conference on Ad-Hoc Networks and Wireless* (pp. 25-38). Springer, Cham.

Bibliography

[1] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online algorithms for location-aware task offloading in two-tiered mobile cloud environments," in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pp. 109–116, IEEE, 2014.

[2] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[3] J. Gantz and D. Reinsel, "Extracting value from chaos," *IDC iview*, vol. 1142, no. 2011, pp. 1–12, 2011.

[4] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton, "Big data: the management revolution," *Harvard business review*, vol. 90, no. 10, pp. 60–68, 2012.

[5] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[6] B. Ravandi and I. Papapanagiotou, "A self-learning scheduling in cloud software defined block storage," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 415–422, IEEE, 2017.

[7] J. Shuja, A. Gani, M. H. ur Rehman, E. Ahmed, S. A. Madani, M. K. Khan, and K. Ko, "Towards native code offloading based MCC frameworks for multimedia applications: A survey," *Journal of Network and Computer Applications*, vol. 75, pp. 335–354, 2016.

[8] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran, and S. Guizani, "Mobile ad hoc cloud: A survey," *Wireless Communications and Mobile Computing*, vol. 16, no. 16, pp. 2572–2589, 2016.

[9] P. Abichandani, W. Fligor, and E. Fromm, "A cloud enabled virtual reality based pedagogical ecosystem for wind energy education," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pp. 1–7, IEEE, 2014.

[10] A. A. Khan, M. H. Rehmani, and M. Reisslein, "Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 860–898, 2015.

[11] M. H. Rehmani, A. Rachedi, M. Erol-Kantarci, M. Radenkovic, and M. Reisslein, "Cognitive radio based smart grid: The future of the traditional electrical grid," *Ad Hoc Networks*, vol. 100, no. 41, pp. 1–4, 2016.

[12] M. H. Rehmani, M. E. Kantarci, A. Rachedi, M. Radenkovic, and M. Reisslein, "IEEE access special section editorial smart grids: A hub of interdisciplinary research," *IEEE access*, vol. 3, pp. 3114–3118, 2015.

[13] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, "Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 10–16, 2016.

[14] M. Satyanarayanan, "A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets," *GetMobile: Mobile Computing and Communications*, vol. 18, no. 4, pp. 19–23, 2015.

[15] B. Zhang, N. Mor, J. Kolb, D. S. Chan, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, and J. Kubiatowicz, "The cloud is not enough: Saving iot from the cloud," in *7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, 2015.

[16] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, "The role of big data in smart city," *International Journal of Information Management*, vol. 36, no. 5, pp. 748–758, 2016.

[17] A. A. Khan, M. H. Rehmani, and A. Rachedi, "When cognitive radio meets the Internet of Things?," in *2016 international wireless communications and mobile computing conference (IWCMC)*, pp. 469–474, IEEE, 2016.

[18] Y. Saleem, F. Salim, and M. H. Rehmani, "Integration of cognitive radio sensor networks and cloud computing: A recent trend," in *Cognitive Radio Sensor Networks: Applications, Architectures, and Challenges*, pp. 288–312, IGI Global, 2014.

[19] P. Hédé, J. Joubert, C. Thornton, B. Naughton, J. R. Ramos, C. Chan, V. Young, S. J. Tan, D. Lynch, N. Sprecher, *et al.*, "Mobile-edge Computing Introductory Technical White Paper," in *Technical Report*, Mobile-edge Computing (MEC) industry initiative, 2014.

[20] Y. Wang, R. Chen, and D.-C. Wang, "A survey of mobile cloud computing applications: Perspectives and challenges," *Wireless Personal Communications*, vol. 80, no. 4, pp. 1607–1623, 2015.

[21] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[22] M. T. Beck and M. Maier, "Mobile edge computing: Challenges for future virtual network embedding algorithms," *Proc. The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2014)*, vol. 1, no. 2, p. 3, 2014.

[23] F. Manco, J. Martins, K. Yasukata, J. Mendes, S. Kuenzer, and F. Huici, "The case for the superfluid cloud," in *7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, 2015.

[24] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," 2015.

[25] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[26] G. Papathanail, A. Pentelas, I. Fotoglou, P. Papadimitriou, K. V. Katsaros, V. Theodorou, S. Soursos, D. Spatharakis, I. Dimolitsas, M. Avgeris, *et al.*, "Meson: Optimized cross-slice communication for edge computing," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 23–28, 2020.

[27] M. Ali, "Green cloud on the horizon," in *IEEE International Conference on Cloud Computing*, pp. 451–459, Springer, 2009.

[28] P. Bahl, R. Y. Han, L. E. Li, and M. Satyanarayanan, "Advancing the state of mobile cloud computing," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pp. 21–28, 2012.

[29] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 133–143, 2014.

[30] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[31] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.

[32] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*, pp. 169–186, Springer, 2014.

[33] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, pp. 37–42, 2015.

[34] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[35] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. Te Wang, T. Mullen, and T. P. Jung, "Augmented brain computer interaction based on fog computing and linked data," in *2014 International Conference on Intelligent Environments*, pp. 374–377, IEEE, 2014.

[36] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 68–81, 2014.

[37] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *2014 Australasian telecommunication networks and applications conference (ATNAC)*, pp. 117–122, IEEE, 2014.

[38] J. Su, F. Lin, X. Zhou, and X. Lu, "Steiner tree based optimal resource caching scheme in fog computing," *China Communications*, vol. 12, no. 8, pp. 161–168, 2015.

[39] O. T. T. Kim, N. D. Tri, N. H. Tran, C. S. Hong, *et al.*, "A shared parking model in vehicular network using fog and cloud environment," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 321–326, IEEE, 2015.

[40] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," 2017.

[41] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: integrating cloudlets and mobile edge computing," in *2016 23rd International conference on telecommunications (ICT)*, pp. 1–5, IEEE, 2016.

[42] S. G. Gupta, D. Ghonge, P. M. Jawandhiya, *et al.*, "Review of unmanned aircraft system (UAS)," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume*, vol. 2, 2013.

[43] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[44] J. O. Fajardo, I. Taboada, and F. Liberal, "Radio-aware service-level scheduling to minimize downlink traffic delay through mobile edge computing," in *International Conference on Mobile Networks and Management*, pp. 121–134, Springer, 2015.

[45] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[46] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond*, pp. 1–5, 2010.

[47] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using MapReduce," tech. rep., Carnegie-mellon univ Pittsburgh PA school of computer science, 2009.

[48] D. Kovachev, T. Yu, and R. Klamma, "Adaptive computation offloading from mobile devices into the cloud," in *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, pp. 784–791, IEEE, 2012.

[49] C. S. Magurawalage, K. Yang, and K. Wang, "Aqua computing: Coupling computing and communications," *arXiv preprint arXiv:1510.07250*, 2015.

[50] E. Ahmed, A. Gani, M. Sookhak, S. H. Ab Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *Journal of Network and Computer Applications*, vol. 52, pp. 52–68, 2015.

[51] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[52] K. Sha, W. Shi, and O. Watkins, "Using wireless sensor networks for fire rescue applications: Requirements and challenges," in *2006 IEEE International Conference on Electro/Information Technology*, pp. 239–244, IEEE, 2006.

[53] P. McKenna, P. D. Erskine, A. M. Lechner, and S. Phinn, "Measuring fire severity using UAV imagery in semi-arid central Queensland, Australia," *International Journal of Remote Sensing*, vol. 38, no. 14, pp. 4244–4264, 2017.

[54] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[55] C. Yuan, Y. Zhang, and Z. Liu, "A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques," *Canadian journal of forest research*, vol. 45, no. 7, pp. 783–792, 2015.

[56] A. P. Kirilenko, T. Molodtsova, and S. O. Stepchenkova, "People as sensors: Mass media and local temperature influence climate change discussion on Twitter," *Global Environmental Change*, vol. 30, pp. 92–100, 2015.

[57] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*, pp. 851–860, 2010.

[58] A. J. Ocampo, R. Chunara, and J. S. Brownstein, "Using search queries for malaria surveillance, Thailand," *Malaria journal*, vol. 12, no. 1, p. 390, 2013.

[59] S. Yang, M. Santillana, J. S. Brownstein, J. Gray, S. Richardson, and S. Kou, "Using electronic health records and Internet search information for accurate influenza forecasting," *BMC infectious diseases*, vol. 17, no. 1, p. 332, 2017.

[60] C. A. Boulton, H. Shotton, and H. T. Williams, "Using social media to detect and locate wildfires," in *tenth international AAAI conference on web and social media*, 2016.

[61] "Detecting forest fires from social media | Digital Earth." `https://digitalearthlab.jrc.ec.europa.eu/activities/detecting-forest-fires-social-media/57793`. (Accessed on 11/04/2020).

[62] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[63] M. Avgeris, D. Spatharakis, D. Dechouniotis, N. Kalatzis, I. Roussaki, and S. Papavassiliou, "Where there is fire there is smoke: a scalable edge computing framework for early fire detection," *Sensors*, vol. 19, no. 3, p. 639, 2019.

[64] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[65] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of things*, pp. 61–75, Elsevier, 2016.

[66] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (TSN)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[67] D. Song, A. K. Tanwani, K. Goldberg, and B. Siciliano, "Networked-, cloud- and fog-robotics," *Springer*, 2019.

[68] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri, "Edge computing and social internet of things for large-scale smart environments development," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2557–2571, 2017.

[69] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[70] Y. Saleem, F. Salim, and M. H. Rehmani, "Resource management in mobile sink based wireless sensor networks through cloud computing," in *Resource management in mobile computing environments*, pp. 439–459, Springer, 2014.

[71] J. Shuja, A. Gani, A. Naveed, E. Ahmed, and C.-H. Hsu, "Case of ARM emulation optimization for offloading mechanisms in mobile cloud computing," *Future Generation Computer Systems*, vol. 76, pp. 407–417, 2017.

[72] S. Li, N. Zhang, S. Lin, L. Kong, A. Katangur, M. K. Khan, M. Ni, and G. Zhu, "Joint admission control and resource allocation in edge computing for internet of things," *IEEE Network*, vol. 32, no. 1, pp. 72–79, 2018.

[73] V. Karyotis, M. Avgeris, M. Michaloliakos, K. Tsagkaris, and S. Papavassiliou, "Utility decisions for QoE-QoS driven applications in practical mobile broadband networks,"

in *2018 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 1–5, IEEE, 2018.

[74] P. Simoens, L. Van Herzeele, F. Vandeputte, and L. Vermoesen, "Challenges for orchestration and instance selection of composite services in distributed edge clouds," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1196–1201, IEEE, 2015.

[75] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, and S. Papavassiliou, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195, p. 108177, 2021.

[76] D. X. Viegas, A. Simeoni, G. Xanthopoulos, C. Rossa, L. Ribeiro, L. Pita, D. Stipanicev, A. Zinoviev, R. Weber, J. Dold, *et al.*, "Recent forest fire related accidents in Europe," *Office for Official Publications of the European Communities: Luxembourg*, 2009.

[77] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[78] C. Chen, "Linear system theory and design: Oxford University Press," *New York, USA*, 1999.

[79] N. Athanasopoulos, "Stability analysis and control of linear and nonlinear constrained systems via Polyhedral Lyapunov functions," *Unpublished Ph. D. dissertation). University of Patras, Greece*, 2010.

[80] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Springer, 2008.

[81] V. Karyotis, "A markov random field framework for modeling malware propagation in complex communications networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 551–564, 2017.

[82] R. Kindermann and J. L. Snell, "Markov random fields and their applications," *American Mathematical Society*, 1980.

[83] N. Leontiou, D. Dechouniotis, N. Athanasopoulos, and S. Denazis, "On load balancing and resource allocation in cloud services," in *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, pp. 773–778, IEEE, 2014.

[84] N. Leontiou, D. Dechouniotis, S. Denazis, and S. Papavassiliou, "A hierarchical control framework of load balancing and resource allocation of cloud computing services," *Computers & Electrical Engineering*, vol. 67, pp. 235–251, 2018.

[85] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, 2009.

[86] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani, "An open ecosystem for mobile-cloud convergence," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 63–70, 2015.

[87] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 369–392, 2014.

[88] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*, pp. 103–130, Springer, 2018.

[89] A. Bhattacharya and P. De, "A survey of adaptation techniques in computation offloading," *Journal of Network and Computer Applications*, vol. 78, pp. 97–115, 2017.

[90] D. T. Hoang, D. Niyato, and P. Wang, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pp. 3145–3149, IEEE, 2012.

[91] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529, 2015.

[92] H. Khojasteh, J. Misic, and V. Misic, "Prioritization of overflow tasks to improve performance of mobile cloud," *IEEE Transactions on Cloud Computing*, 2016.

[93] H. Raei and N. Yazdani, "Analytical performance models for resource allocation schemes of cloudlet in mobile cloud computing," *The Journal of Supercomputing*, vol. 73, no. 3, pp. 1274–1305, 2017.

[94] V. Cardellini, V. D. N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421–449, 2016.

[95] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.

[96] A. Kiani and N. Ansari, "Optimal Code Partitioning Over Time and Hierarchical Cloudlets," *IEEE Communications Letters*, 2017.

[97] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *INFOCOM, 2013 Proceedings IEEE*, pp. 1285–1293, IEEE, 2013.

[98] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, pp. 570–578, IEEE, 2015.

[99] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pp. 1–9, IEEE, 2016.

[100] M. Liu and Y. Liu, "Price-Based Distributed Offloading for Mobile-Edge Computing with Computation Capacity Constraints," *arXiv preprint arXiv:1712.00599*, 2017.

[101] X. Sun and N. Ansari, "Avaptive Avatar Handoff in the Cloudlet Network," *IEEE Transactions on Cloud Computing*, 2017.

[102] Z. Cao and P. Papadimitriou, "Collaborative content caching in wireless edge with SDN," in *Proceedings of the 1st Workshop on Content Caching and Delivery in Wireless Networks*, p. 6, ACM, 2016.

[103] P. Wellstead and M. B. Zarrop, *Self-tuning systems: control and signal processing*. John Wiley & Sons, Inc., 1991.

[104] D. Dechouniotis, N. Leontiou, N. Athanasopoulos, G. Bitsoris, and S. Denazis, "ACRA: a unified admission control and resource allocation framework for virtualized environments," in *Proceedings of the 8th International Conference on Network and Service Management*, pp. 145–149, IFIP/IEEE, 2012.

[105] D. Dechouniotis, N. Leontiou, N. Athanasopoulos, A. Christakidis, and S. Denazis, "A control-theoretic approach towards joint admission control and resource allocation of cloud computing services," *International Journal of Network Management*, vol. 25, no. 3, pp. 159–180, 2015.

[106] S. V. Rakovic, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne, "Invariant approximations of the minimal robust positively invariant set," *IEEE Transactions on Automatic Control*, vol. 50, no. 3, pp. 406–410, 2005.

[107] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Springer, 2008.

[108] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical problems in engineering*, vol. 4, no. 4, pp. 317–367, 1998.

[109] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting methods and applications*. John wiley & sons, 2008.

[110] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pp. 400–406, IEEE, 2017.

[111] GLPK-YALMIP, "Mixed-integer linear programming solver," September 2016.

[112] S. Dash, "Exponential lower bounds on the lengths of some classes of branch-and-cut proofs," *Mathematics of Operations Research*, vol. 30, no. 3, pp. 678–700, 2005.

[113] V. Vipin, "Image processing based forest fire detection," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 2, pp. 87–95, 2012.

[114] Z. Ying-cong and Y. Jing, "A study on the fire IOT development strategy," *Procedia Engineering*, vol. 52, pp. 314–319, 2013.

[115] Y. Bo and H. Wang, "The application of cloud computing and the internet of things in agriculture and forestry," in *2011 International Joint Conference on Service Sciences*, pp. 168–172, IEEE, 2011.

[116] M. Dener, Y. Özkök, and C. Bostancıoğlu, "Fire detection systems in wireless sensor networks," *Procedia-Social and Behavioral Sciences*, vol. 195, pp. 1846–1850, 2015.

[117] H. Cruz, M. Eckert, J. Meneses, and J.-F. Martínez, "Efficient forest fire detection index for application in unmanned aerial systems (UASs)," *Sensors*, vol. 16, no. 6, p. 893, 2016.

[118] "IEEE Standards Association, AIOTI, oneM2M and W3C Collaborate on Joint White Paper Covering Semantic Interoperability for the Internet of Things (IoT)." `https://standards.ieee.org/news/2016/semantic_interoperability.html`. (Accessed on 11/04/2020).

[119] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20–26, 2014.

[120] I. Roussaki, N. Kalatzis, N. Liampotis, P. Kosmides, M. Anagnostou, K. Doolin, E. Jennings, Y. Bouloudis, and S. Xynogalas, "Context-awareness in wireless and mobile computing revisited to embrace social networking," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 74–81, 2012.

[121] N. Kalatzis, M. Avgeris, D. Dechouniotis, K. Papadakis-Vlachopapadopoulos, I. Roussaki, and S. Papavassiliou, "Edge computing in IoT ecosystems for UAV-enabled early fire detection," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 106–114, IEEE, 2018.

[122] M. Avgeris, N. Kalatzis, D. Dechouniotis, I. Roussaki, and S. Papavassiliou, "Semantic Resource Management of Federated IoT Testbeds," in *International Conference on Ad-Hoc Networks and Wireless*, pp. 25–38, Springer, 2017.

[123] N. Kalatzis, G. Routis, Y. Marinellis, M. Avgeris, I. Roussaki, S. Papavassiliou, and M. Anagnostou, "Semantic interoperability for iot platforms in support of decision making: an experiment on early wildfire detection," *Sensors*, vol. 19, no. 3, p. 528, 2019.

[124] V. Slavkovikj, S. Verstockt, S. Van Hoecke, and R. Van de Walle, "Review of wildfire detection using social media," *Fire safety journal*, vol. 68, pp. 109–118, 2014.

[125] Z. Wang, X. Ye, and M.-H. Tsou, "Spatial, temporal, and content analysis of Twitter for wildfire hazards," *Natural Hazards*, vol. 83, no. 1, pp. 523–540, 2016.

[126] F. Abel, C. Hauff, G.-J. Houben, R. Stronkman, and K. Tao, "Twitcident: fighting fire with information from social web streams," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 305–308, 2012.

[127] M. Rahman and P. Graham, "Hybrid resource provisioning for clouds," *Journal of Physics: Conference Series*, vol. 385, no. 1, p. 012004, 2012.

[128] A. Ullah, J. Li, Y. Shen, and A. Hussain, "A control theoretical view of cloud elasticity: taxonomy, survey and challenges," *Cluster Computing*, vol. 21, no. 4, pp. 1735–1764, 2018.

[129] P. Saikrishna and R. Pasumarthy, "Multi-objective switching controller for cloud computing systems," *Control Engineering Practice*, vol. 57, pp. 72–83, 2016.

[130] D. Grimaldi, V. Persico, A. Pescapé, A. Salvi, and S. Santini, "A feedback-control approach for resource management in public clouds," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2015.

[131] T. Patikirikorala, A. Colman, and J. Han, "4M-Switch: Multi-mode-multi-model supervisory control framework for performance differentiation in virtual machine environments," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, pp. 145–153, IEEE, 2014.

[132] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 164–174, 2017.

[133] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE transactions on services computing*, 2017.

[134] "Home: IQ FireWatch." `https://www.iq-firewatch.com/`. (Accessed on 11/04/2020).

[135] E. Koo, P. Pagni, S. Stephens, J. Huff, J. Woycheese, and D. Weise, "A simple physical model for forest fire spread rate," *Fire Safety Science*, vol. 8, pp. 851–862, 2005.

[136] "COPERNICUS." `https://effis.jrc.ec.europa.eu/applications/data-and-services/`. (Accessed on 11/04/2020).

[137] L. Ljung, "System identification," *Wiley encyclopedia of electrical and electronics engineering*, pp. 1–19, 1999.

[138] "NETMODE: NETwork Management & Optimal DEsign Laboratory." `http://www.netmode.ntua.gr/main/index.php?option=com_content&view=article&id=103&Itemid=83`. (Accessed on 11/04/2020).

[139] "TensorFlow." `https://www.tensorflow.org/`. (Accessed on 11/04/2020).

[140] "Forestry Images." `https://www.forestryimages.org/browse/subimages.cfm?sub=740`. (Accessed on 11/04/2020).

[141] A. Olteanu, S. Vieweg, and C. Castillo, "What to expect when the unexpected happens: Social media communications across crises," in *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pp. 994–1009, 2015.

[142] M. Kibanov, G. Stumme, I. Amin, and J. G. Lee, "Mining social media to inform peatland fire and haze disaster management," *Social Network Analysis and Mining*, vol. 7, no. 1, p. 30, 2017.

[143] H. To, S. Agrawal, S. H. Kim, and C. Shahabi, "On identifying disaster-related tweets: Matching-based or learning-based?," in *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, pp. 330–337, IEEE, 2017.

[144] D. Spatharakis, M. Avgeris, N. Athanasopoulos, D. Dechouniotis, and S. Papavassiliou, "A switching offloading mechanism for path planning and localization in robotic applications," in *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 77–84, IEEE, 2020.

[145] S. Dey and A. Mukherjee, "Robotic SLAM: A Review from Fog Computing and Mobile Edge Computing Perspective," in *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*, MOBIQUITOUS 2016, (New York, NY, USA), p. 153–158, Association for Computing Machinery, 2016.

[146] K. Antevski, C. J. Bernardos, L. Cominardi, A. de la Oliva, and A. Mourad, "On the integration of NFV and MEC technologies: architecture analysis and benefits for edge robotics," *Computer Networks*, vol. 175, p. 107274, 2020.

[147] N. Tian, B. Kuo, X. Ren, M. Yu, R. Zhang, B. Huang, K. Goldberg, and S. Sojoudi, "A cloud-based robust semaphore mirroring system for social robots," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1351–1358, IEEE, 2018.

[148] A. K. Tanwani, N. Mor, J. Kubiatowicz, J. E. Gonzalez, and K. Goldberg, "A fog robotics approach to deep robot learning: Application to object recognition and grasp planning in surface decluttering," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4559–4566, IEEE, 2019.

[149] A. Botta, L. Gallo, and G. Ventre, "Cloud, Fog, and Dew Robotics: Architectures for next generation applications," in *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 16–23, IEEE, 2019.

[150] B. V. Bhausaheb and P. S. Saikrishna, "Control Algorithms for a Mobile Robot Application in a Fog Computing Environment," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, ICACR 2019, (New York, NY, USA), p. 30–36, Association for Computing Machinery, 2019.

[151] A. W. Malik, A. U. Rahman, M. Ali, and M. M. Santos, "Symbiotic robotics network for efficient task offloading in smart industry," *IEEE Transactions on Industrial Informatics*, 2020.

[152] L. A. Trinh, N. D. Thang, N. V. D. Hau, and T. C. Hung, "Position rectification with depth camera to improve odometry-based localization," in *2015 International Conference on Communications, Management and Telecommunications (ComManTel)*, pp. 147–152, IEEE, 2015.

[153] "AlphaBot." `https://www.waveshare.com/wiki/AlphaBot`. (Accessed on 07/01/2021).

[154] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[155] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[156] D. Pizarro, M. Mazo, E. Santiso, M. Marron, D. Jimenez, S. Cobreces, and C. Losada, "Localization of mobile robots using odometry and an external vision sensor," *Sensors*, vol. 10, no. 4, pp. 3655–3680, 2010.

[157] M. Avgeris, D. Spatharakis, N. Athanasopoulos, D. Dechouniotis, and S. Papavassiliou, "Single Vision-Based Self-Localization for Autonomous Robotic Agents," in *2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 123–129, IEEE, 2019.

[158] D. C. Yuen and B. A. MacDonald, "Vision-based localization algorithm based on landmark matching, triangulation, reconstruction, and comparison," *IEEE Transactions on robotics*, vol. 21, no. 2, pp. 217–226, 2005.

[159] A. Bais and R. Sablatnig, "Landmark based global self-localization of mobile soccer robots," in *Asian Conference on Computer Vision*, pp. 842–851, Springer, 2006.

[160] M. Young, "The pinhole camera: Imaging without lenses or mirrors," *The Physics Teacher*, vol. 27, no. 9, pp. 648–655, 1989.

[161] X. Li, B. Hua, Y. Shang, and Y. Xiong, "A robust localization algorithm in wireless sensor networks," *Frontiers of computer science in China*, vol. 2, no. 4, pp. 438–450, 2008.

[162] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta^*: Any-angle path planning on grids," in *AAAI*, vol. 7, pp. 1177–1183, 2007.

[163] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1148–1153, IEEE, 2015.

[164] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[165] J. Peng, Y. Huang, and G. Luo, "Robot path planning based on improved A* algorithm," *Cybernetics and Information Technologies*, vol. 15, no. 2, pp. 171–180, 2015.

[166] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[167] D. D. Harabor, A. Grastien, *et al.*, "Online graph pruning for pathfinding on grid maps.," in *AAAI*, pp. 1114–1119, 2011.

[168] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 03 1960.

[169] D. B. Faria *et al.*, "Modeling signal attenuation in ieee 802.11 wireless lans-vol. 1," *Computer Science Department, Stanford University*, vol. 1, 2005.

[170] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

[171] D. C. G. Valadares, J. M. F. R. de Araújo, M. A. Spohn, A. Perkusich, K. C. Gorgônio, and E. U. K. Melcher, "802.11 g signal strength evaluation in an industrial environment," *Internet of Things*, vol. 9, p. 100163, 2020.

[172] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji, "Energy-efficient resource allocation for multi-user mobile edge computing," in *Proc. GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

[173] D. Dechouniotis, N. Athanasopoulos, A. Leivadeas, N. Mitton, R. M. Jungers, and S. Papavassiliou, "Edge Computing Resource Allocation for Dynamic Networks: The DRUID-NET Vision and Perspective," *MDPI Sensors*, vol. 20, no. 8, p. 2191, 2020.

[174] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. ICDCS 2017-The 37th IEEE International Conference on Distributed Computing Systems*, pp. 1281–1290, IEEE, 2017.

[175] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2016.

[176] C. Puliafito, E. Mingozzi, C. Vallati, F. Longo, and G. Merlino, "Companion fog computing: Supporting things mobility through container migration at the edge," in *Proc. IEEE SMARTCOMP 2018-The 4th IEEE International Conference on Smart Computing*, pp. 97–105, IEEE, 2018.

[177] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE PIMRC 2016-27th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–6, IEEE, 2016.

[178] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 164–174, 2017.

[179] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[180] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, 2018.

[181] I. Farris, L. Militano, M. Nitti, L. Atzori, and A. Iera, "MIFaaS: A mobile-IoT-federation-as-a-service model for dynamic cooperation of IoT cloud providers," *Elsevier Future Generation Computer Systems*, vol. 70, pp. 126–137, 2017.

[182] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, 2019.

[183] J. Plachy, Z. Becvar, E. C. Strinati, and N. di Pietro, "Dynamic Allocation of Computing and Communication Resources in Multi-Access Edge Computing for Mobile Users," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2089–2106, 2021.

[184] M. Jia, W. Liang, Z. Xu, M. Huang, and Y. Ma, "Qos-aware cloudlet load balancing in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, 2018.

[185] J. S. Baras and X. Tan, "Control of autonomous swarms using Gibbs sampling," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 5, pp. 4752–4757, IEEE, 2004.

[186] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[187] S. Liu, G. Li, T. Tran, and Y. Jiang, "Preference relation-based Markov random fields for recommender systems," in *Asian Conference on Machine Learning*, pp. 157–172, PMLR, 2016.

[188] V. Karyotis, M. Vitoropoulou, N. Kalatzis, I. Roussaki, S. Papavassiliou, O. Khalid, S. Khan, and A. Zomaya, "Efficient and socio-aware recommendation approaches for big data networked systems," *Big Data Recommender Systems*, vol. 1, pp. 58–87, 2019.

[189] E. Anifantis, V. Karyotis, and S. Papavassiliou, "A spatio-stochastic framework for cross-layer design in cognitive radio networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2762–2771, 2014.

[190] G. Kakkavas, K. Tsitseklis, V. Karyotis, and S. Papavassiliou, "A software defined radio cross-layer resource allocation approach for cognitive radio networks: From theory to practice," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 740–755, 2020.

[191] M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou, "Adaptive resource allocation for computation offloading: A control-theoretic approach," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–20, 2019.

[192] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proc. MPM '12-First Workshop on Measurement, Privacy, and Mobility*, pp. 1–6, 2012.

[193] A. Leivadeas, T. Nilsson Y., A. Elahi, A. Keyhanian, and I. Lambadaris, "Link Adaptation for Fair Coexistence of Wi-Fi and LAA-LTE," in *Proc. ACM MobiWac 2018-The 16th ACM International Symposium on Mobility Management and Wireless Access*, pp. 43–50, ACM, 2018.

[194] V. Erceg, "IEEE P802. 11 wireless LANs TGn channel models," *IEEE 802.11-03/940r4*, 2004.

[195] "Madwifi project - Minstrel Algorithm [Online]." `https://sourceforge.net/p/madwifi/svn/HEAD/tree/madwifi/trunk/ath_rate/minstrel/minstrel.txt`. Accessed: 2021-05-08.

[196] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.

[197] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Going Green with the Networked Cloud: Methodologies and Assessment," *Wiley Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, pp. 351–374, 2015.

[198] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *Proc. ACM MobiSys 2010-The 8th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 49–62, 2010.

[199] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM 2012-The 31st Annual IEEE International Conference on Computer Communications*, pp. 945–953, IEEE, 2012.

[200] L. Ljung, "System Identification: Theory for the User," *Englewood Cliffs (N.J.): Prentice-Hall*, 1987.

[201] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," in *Elsevier Advances in Computers*, vol. 82, pp. 47–111, Elsevier, 2011.

[202] M. Falkner, A. Leivadeas, I. Lambadaris, and G. Kesidis, "Performance analysis of virtualized network functions on virtualized systems architectures," in *Proc. IEEE CAMAD 2016-21st IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks*, pp. 71–76, IEEE, 2016.

[203] F. Bohnert and I. Zukerman, "Personalised viewing-time prediction in museums," *Springer User Modeling and User-Adapted Interaction*, vol. 24, no. 4, pp. 263–314, 2014.

[204] J. Cao, Y. Zhao, X. Lai, M. E. H. Ong, C. Yin, Z. X. Koh, and N. Liu, "Landmark recognition with sparse representation classification and extreme learning machine," *Elsevier Journal of the Franklin Institute*, vol. 352, no. 10, pp. 4528–4545, 2015.

[205] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, *et al.*, "An empirical study of latency in an emerging class of edge com-

puting applications for wearable cognitive assistance," in *Proc. SEC '17-The Second ACM/IEEE Symposium on Edge Computing*, pp. 1–14, ACM, 2017.

[206] C. N. Le Tan, C. Klein, and E. Elmroth, "Location-aware load prediction in edge data centers," in *Proc. IEEE FMEC 2017-The 2nd International Conference on Fog and Mobile Edge Computing*, pp. 25–31, IEEE, 2017.

[207] C. Jin, X. Bai, C. Yang, W. Mao, and X. Xu, "A review of power consumption models of servers in data centers," *Elsevier Applied Energy*, vol. 265, p. 114806, 2020.

Index