ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ποιότητα υπηρεσίας στο διαδίκτυο των πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μουρατίδη Ιωάννη

Επιβλέπων: Μιχάλας Άγγελος, Καθηγητής

ΚΟΖΑΝΗ/ΦΕΒΡΟΥΑΡΙΟΣ/2024

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING

ece.uowm.gr

Quality of service in the internet of things

THESIS

MOURATIDIS IOANNIS

SUPERVISOR: Angelos Michalas, Professor

KOZANI/FEBRUARY/2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο “Ποιότητα υπηρεσίας στο διαδίκτυο των πραγμάτων*Quality of service in the internet of things” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Άγγελος Μιχάλας αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή:

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη ενός συστήματος διαχείρισης υπολογιστικών πόρων στα δίκτυα Fog-RAN. Τα τελευταία χρόνια παρατηρείται έκρηξη των συσκευών και των δεδομένων που παράγονται από αυτές στο πλαίσιο των εφαρμογών διαδικτύων των πραγμάτων (IoT). Παράλληλα, οι αυξημένες απαιτήσεις των εφαρμογών σε υπολογιστικούς πόρους δυσχεραίνουν τη διαχείριση τους. Για το λόγο αυτό, αναπτύχθηκε ένα μοντέλο διαχείρισης υπολογιστικών πόρων, το οποίο μέσω ενός ελεγκτή άκρης κατανέμει τις διεργασίες στα Fog Nodes των clusters, με στόχο τη βελτιστοποίηση της χρήσης τους.

Η συνεισφορά της διπλωματικής έχει 4 βασικά χαρακτηριστικά που την ξεχωρίζει από τις ίδιες έρευνες.

- Μοντελοποιεί ένα σύστημα διαχείρισης υπολογιστικών πόρων στην άκρη του δικτύου βασισμένο στην υπάρχουσα τεχνολογία.
- Σχεδιάζει έναν αλγόριθμο λήψης αποφάσεων με παραλλαγή του reward για καλυτέρα αποτελέσματα, μέσω τεχνικών ενισχυμένης μάθησης για τη διαχείριση των υπολογιστικών πόρων.
- Υλοποίει ένα προσαρμοσμένο περιβάλλον ειδικά για την αξιολόγηση του μοντέλου.
- Εκτελεί πειράματα και αναλύει τα αποτελέσματα για τη μέτρηση βασικών δεικτών ποιότητας υπηρεσίας.
- Συγκρίνει το προτεινόμενο μοντέλο με άλλες υλοποιήσεις.

Τα αποτελέσματα του πειράματος έδειξαν ότι το προτεινόμενο μοντέλο μπορεί να διαχειριστεί με επιτυχία τους περιορισμένους υπολογιστικούς πόρους σε σχέση με στατικές μεθόδους και άλλες υλοποιήσεις αλγορίθμων, βελτιστοποιώντας την αξιοποίηση τους και εξασφαλίζοντας υψηλή ποιότητα υπηρεσίας στα IoT. Η παρούσα μελέτη συνεισφέρει στην καλύτερη κατανόηση των σύγχρονων συστημάτων IoT και παρέχει τη βάση για περαιτέρω βελτιώσεις προς υλοποίηση αποδοτικότερων λύσεων διαχείρισης πόρων.

Λέξεις Κλειδιά

Διαδίκτυο των πραγμάτων, διαχείριση υπολογιστικών πόρων, δίκτυα άκρης, ελεγκτή άκρης, ποιότητα υπηρεσίας.

Abstract

This thesis examines the challenge of managing Quality of Service in Internet of Things networks. A resource management model is presented, operating within the framework of edge networks. Through the operation of an Edge Controller (EC) and the use of machine learning techniques, the model can optimize the distribution of computational resources across various edge clusters. The model has been evaluated through an experiment, the results of which demonstrate its ability to optimize resource utilization and ensure high quality of service.

The proposed model offers significant advantages compared to other approaches, such as the use of reinforced learning for dynamic adaptation. This allows it to "learn" much more rapidly and effectively from experience, compared to other techniques of more "superficial" reinforced learning, making it an ongoing optimization process. Furthermore, the segmentation into clusters and centralized management by the EC ensures greater homogeneity and efficiency in the allocation of computational resources.

However, some improvements could be made, such as the application of more advanced machine learning algorithms by the EC. Overall, this work contributes to understanding distributed IoT systems and provides a basis for further enhancements leading to smarter future solutions.

Keywords

Edge Controller, edge networks, internet of Things, machine learning, resource management, quality of service.

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το τελευταίο στάδιο της φοίτησης μου στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Δυτικής Μακεδονίας. Θα ήθελα να ευχαριστήσω τον κ. Μιχάλα Άγγελο για την καθοδήγηση, την άψογη συνεργασία που είχαμε και την εμπιστοσύνη που μου έδειξε στην ανάθεση της διπλωματικής εργασίας. Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου και ειδικά τους γονείς μου για την στήριξη που μου προσέφεραν σε όλη την διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1: Εισαγωγή

1.1 Εισαγωγή

Η συνεχής ανάπτυξη των έξυπνων δικτύων και η εκθετική αύξηση των συσκευών και των αισθητήρων που τα αποτελούν, έχουν οδηγήσει σε μια ραγδαία αύξηση του όγκου δεδομένων. Παράλληλα, τα δίκτυα επόμενης γενιάς έρχονται να προσφέρουν ακόμα υψηλότερες ταχύτητες και πυκνότητες συσκευών, δημιουργώντας ένα περιβάλλον με τεράστιους όγκους δεδομένων.

Για να ανταπεξέλθουμε σε αυτές τις προκλήσεις, οι τεχνολογίες τεχνητής νοημοσύνης και μηχανικής μάθησης αποτελούν απαραίτητα εργαλεία για την αποτελεσματική επεξεργασία και διαχείριση των περιορισμένων υπολογιστικών πόρων που διατίθενται. Η έννοια της "Ποιότητας της Υπηρεσίας" (quality of services - QoS) είναι αυτή που θα μας βοηθήσει στην επίτευξη της σωστής διαχείρισης υπολογιστικών πόρων και αναφέρεται σε ένα σύνολο μετρήσιμων απαιτήσεων του συστήματος, το οποίο εξαρτάται από την κάθε εφαρμογή. Ως εκ τούτου, δεν εγγυάται την επίτευξη οποιουδήποτε στόχου, αλλά προσφέρει έναν μηχανισμό για την κατάλληλη κατανομή των πόρων.

Η τελευταία δεκαετία έχει δείξει μια σημαντική εξέλιξη στον τομέα των δικτύων επικοινωνίας, με τα δίκτυα B5G να έρχονται στο προσκήνιο ως μια από τις πιο προηγμένες τεχνολογίες στο χώρο. Αυτή η τεχνολογία έχει αποδειχθεί κρίσιμη για την ανάπτυξη του διαδικτύου των πραγμάτων (IoT - Internet of Things). Στο πλαίσιο αυτό, η ικανότητα των δικτύων B5G να υποστηρίξουν μεγαλύτερους όγκους δεδομένων και να προσφέρουν ταχύτερες ταχύτητες είναι ζωτικής σημασίας. Αυτό δίνει τη δυνατότητα στις εφαρμογές IoT να λειτουργούν αποτελεσματικότερα, με μεγαλύτερη ενεργειακή αποδοτικότητα. Αυτό έχει αποδειχθεί κρίσιμο για την ανάπτυξη σύγχρονων εφαρμογών IoT και μας ετοιμάζει για την γενιά 6G συστημάτων επικοινωνίας.

Καθώς ετοιμαζόμαστε όμως για τη γενιά των 6G συστημάτων επικοινωνίας θα πρέπει να δημιουργήσουμε τις κατάλληλες υποδομές, αφού προβλέπεται μια επανάσταση με έως και 100 φορές μεγαλύτερες ταχύτητες δικτύου σε σχέση με την προηγούμενη γενιά 5G [1], ιδιαίτερα στο δίκτυο πρόσβασης. Αυτό θα εξασφαλίσει μια ακόμα πιο απρόσκοπτη και σχεδόν άμεση συνδεσιμότητα μεταξύ των συσκευών, καθιστώντας τη μεταφορά και την ανταλλαγή πληροφοριών πιο αξιόπιστη. Αυτό θα καταστήσει τις εφαρμογές IoT πιο αποδοτικές, επιτρέποντας την άμεση ανταλλαγή πληροφοριών χωρίς καθυστερήσεις δικτύου.

Στην αμέσως επόμενη γενιά δικτύων, η ικανότητα για γρήγορη μεταφορά δεδομένων χωρίς καθυστερήσεις θα είναι αποφασιστική για τη λειτουργία εφαρμογών που απαιτούν άμεση ανταπόκριση, όπως οι κρίσιμες εφαρμογές στο πλαίσιο του τριγώνου των εφαρμογών του 5G, το οποίο εξελίσσεται σε εξάγωνο στο 6G και είναι σημαντικό και για τις επόμενες γενιές δικτύων που θα έρθουν. Από την άλλη πλευρά, η αποτελεσματική διαχείριση των δεδομένων και των πόρων θα εξασφαλίζει την ομαλή εκτέλεση των βασικών διαδικασιών, βελτιώνοντας τη γενική απόδοση του συστήματος.

1.1.1 Αξιοποίηση των IoT τεχνολογιών

Για να διασφαλιστεί η αξιοπιστία του δικτύου, είναι ζωτικής σημασίας η αποτελεσματική ανάθεση των υπολογιστικών πόρων. Οι διεργασίες που απαιτούν υψηλά επίπεδα υπολογιστικής ισχύος ανατίθενται στο νέφος (cloud), ενώ οι διεργασίες που απαιτούν χαμηλότερα επίπεδα υπολογιστικής ισχύος εκτελούνται τοπικά σε κάποιο edge cluster για μείωση του χρόνο απόκρισης του συστήματος (καθυστέρηση). Επιπλέον, σε συστήματα που επηρεάζουν την ανθρώπινη ασφάλεια, όπως η ειδοποίηση σε περίπτωση ατυχήματος για οχήματα χωρίς οδηγό, είναι κρίσιμη η γρήγορη ανταπόκριση και η ακρίβεια. Τέτοιες διεργασίες πρέπει να εκτελούνται στο edge cluster για να διασφαλιστεί η άμεση ενημέρωση και να αποφευχθούν περαιτέρω προβλήματα.

Η εξέλιξη των δικτύων επικοινωνίας B5G και η προοπτική των επερχόμενων δικτύων 6G είναι σημαντική για την αξιοπιστία του δικτύου, σε συνδυασμό με την πληθώρα αισθητήρων και των δυνατοτήτων που παρέχουν, όπως ραντάρ, lidars, βιντεοκάμερες υψηλής ευκρίνειας, GPS κ.α. [2], αφού θα προκαλέσουν την παραγωγή τεράστιων ποσοτήτων δεδομένων καθημερινά. Η ικανότητα να επεξεργαζόμαστε και να διαχειριζόμαστε αυτά τα δεδομένα με αποτελεσματικότητα θα βελτιώσει την ποιότητας υπηρεσιών του τελικού χρήστη και θα καθορίσει την αποδοτικότητα και την βιωσιμότητα των μελλοντικών IoT εφαρμογών.

Με την εξέλιξη αυτή, αναμένεται να διαμορφωθεί μια ευφυής κοινότητα που τα δίκτυα επικοινωνίας θα παίζουν καθοριστικό ρόλο στις πόλεις και οι τεχνολογίες IoT, μαζί με την αποτελεσματική διαχείριση των πόρων και την ποιότητα της υπηρεσίας που θα παρέχουν, θα προωθήσουν την αυτοματοποίηση και την ανάπτυξη καινοτόμων εφαρμογών σε όλους τους τομείς της κοινωνίας, όπως η υγεία, η μεταφορά, η ενέργεια, η παραγωγή και η ψυχαγωγία. Η συνδυασμένη αξιοποίηση των νέων τεχνολογιών και των μοντέλων διαχείρισης θα προωθήσει την ανάπτυξη εξυπνότερων συστημάτων που θα λειτουργούν αυτόνομα και αποτελεσματικά. Τα συνδεδεμένα αντικείμενα θα ανταλλάσσουν δεδομένα και πληροφορίες μεταξύ τους, λαμβάνοντας αποφάσεις που θα βελτιστοποιήσουν τις λειτουργίες. Αυτό θα προσφέρει πρακτικά οφέλη στην καθημερινότητα του ανθρώπου, όπως εξοικονόμηση χρόνου και προσαρμοστικότητα σε διάφορες καθημερινές δραστηριότητες.

Έτσι, θα οδηγήσει σε ένα πιο έξυπνο και αειφόρο τρόπο ζωής. Οι εφαρμογές IoT δεν θα αποτελεί απλώς μια πηγή άνεσης και ψυχαγωγίας, αλλά θα βοηθάει στην αντιμετώπιση σημαντικών κοινωνικών προκλήσεων, όπως η προστασία του περιβάλλοντος και η πρόληψη ατυχημάτων. Με τις εφαρμογές IoT θα είμαστε σε θέση να παρακολουθούμε και να προλαμβάνουμε τέτοιου είδους προβλήματα.

1.2 Επιλογή θέματος

Η επιλογή του θέματος για την παρούσα έρευνα ήταν εν μέρει εμπνευσμένη από την αναγνώριση ενός σημαντικού κενού στην υπάρχουσα βιβλιογραφία, σχετικά με την βελτιστοποίηση της ποιότητας υπηρεσίας εφαρμογών IoT και τον αποτελεσματικό καταμερισμό υπολογιστικών πόρων σε αυτές.

Παρά το πλήθος των μελετών που έχουν δημοσιευτεί σε αυτόν τον τομέα, υπάρχει ακόμη περιθώριο βελτίωσης μέσω της προσφοράς καινοτόμων λύσεων, ικανών να επηρεάσουν σημαντικά την πορεία ανάπτυξης της τεχνολογίας IoT. Γι' αυτό και θεωρήθηκε ιδιαίτερα σημαντικό για την παρούσα έρευνα να εξεταστεί αυτό το ερευνητικό θέμα.

Ένας σημαντικός παράγοντας για την ποιότητας υπηρεσίας είναι η αξιοπιστία του δικτύου. Κάθε συσκευή στο IoT πρέπει να είναι σε θέση να συνδεθεί στο δίκτυο και να μεταδώσει τα δεδομένα της με αξιοπιστία. Αν το δίκτυο είναι ασταθές ή υπερφορτωμένο, μπορεί να οδηγήσει σε καθυστερήσεις ή αποτυχία στη μετάδοση δεδομένων, με αρνητικές συνέπειες για τη λειτουργία του συστήματος. Αυτός είναι ένας βασικός παράγοντας που θα εξεταστεί κατά ποσό δηλαδή η αλλαγές δικτύου επηρεάζουν την ποιότητα της υπηρεσίας. Για το λόγο αυτό, αναπτύσσουμε ένα μοντέλο καταμερισμού υπολογιστικών πόρων δικτύου που βασίζεται σε ένα σύμπλεγμα κόμβων ομίχλης (fog nodes-FNs) που συντονίζονται με έναν ελεγκτή άκρου(edge controller - EC) για την αποτελεσματική ανάθεση διεργασιών και χρήση των περιορισμένων υπολογιστικών πόρων στην άκρη του δικτύου.

Το EC δουλεύει ώστε να εξασφαλίσει την αξιοπιστία του δικτύου, να μειώσει τις επιπτώσεις τυχόν αστάθειας ή υπερφόρτωσης και να διαμοιράζει τις διεργασίες για εκτέλεση στα FNs ή να τις απορρίπτει με αποτέλεσμα αυτές να εκτελεσθούν στο cloud, αυτός ήταν ο κύριος σκοπός για την ανάπτυξη της παρούσας διπλωματικής. Συνοψίζοντας, η επιλογή αυτού του θέματος δίνει την δυνατότητα για να εξεταστούν τα πλεονεκτήματα και τις προκλήσεις που συνοδεύουν την ανάπτυξη του IoT, με έμφαση στην ποιότητα υπηρεσιών.

1.2.1 Συνεισφορά διπλωματικής

Το μοντέλο που ετοιμάζουμε θα καταφέρει να λειτουργήσει επιτυχώς χάρη σε μια συνδυασμένη προσέγγιση που εκμεταλλεύεται τις προηγμένες τεχνολογίες και τις αρχές της ποιότητας υπηρεσίας για να καταφέρει να κάνει ένα πιο αποτελεσματικό τεμαχισμό υπολογιστικών πόρων στις διάφορες διεργασίες.

Μέσω του προτεινόμενου μοντέλου, ο Edge Controller αναλαμβάνει τον δυναμικό συντονισμό των υπολογιστικών πόρων μεταξύ των διαφόρων edge clusters. Συγκεκριμένα, ο EC παρακολουθεί σε πραγματικό χρόνο τους διαθέσιμους υπολογιστικούς πόρους σε κάθε cluster και τις απαιτήσεις των εφαρμογών και των διεργασιών που ζητούν εκτέλεση. Ο EC χρησιμοποιώντας αλγόριθμους τεχνητής νοημοσύνης και τις απαιτήσεις των διεργασιών, αποφασίζει ποια cluster μπορούν να υποστηρίξουν κάθε νέα διεργασία και πόσους ακριβώς πόρους να αναθέσει σε αυτή, προκειμένου να αξιοποιηθούν αποδοτικά οι υπολογιστικοί πόροι του άκρου.

Όπου λέμε υπολογιστικοί πόροι περιλαμβάνονται κυρίως η επεξεργαστική ισχύ (CPU) και η μνήμη RAM . Η CPU αντιπροσωπεύει την ικανότητα επεξεργασίας και εκτέλεσης εργασιών, ενώ η μνήμη αποθηκεύει προσωρινά δεδομένα κατά την εκτέλεση.

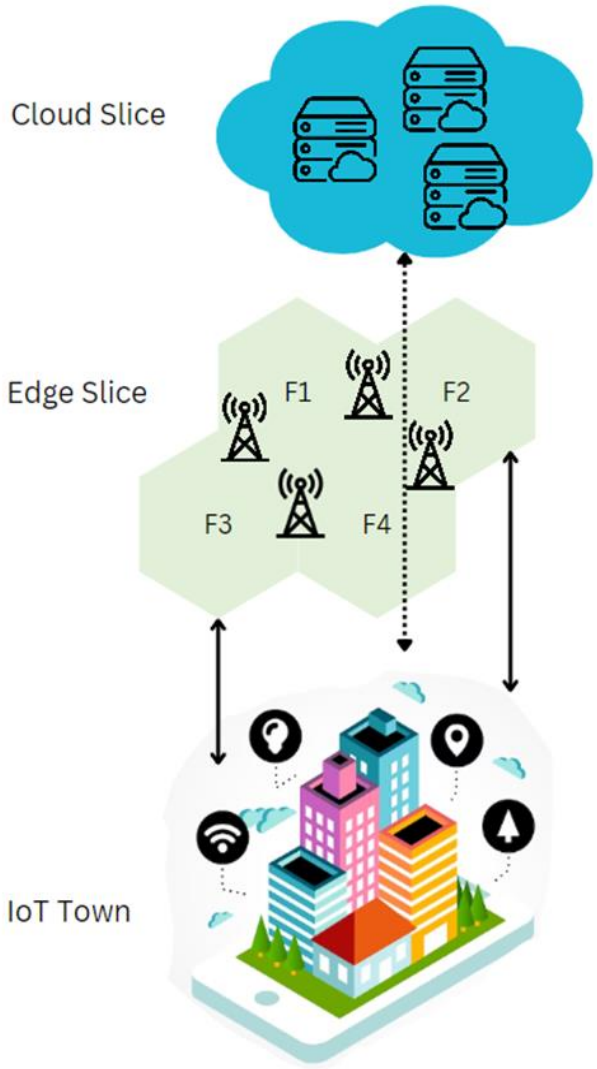

Figure 1. Μια σύγχρονη IoT πόλη.

Στο Figure 1, παρουσιάζεται η απεικόνιση μιας έξυπνης πόλης.

Αρχικά, το επίπεδο άκρης συνδέεται με τη ζώνη σύννεφου μέσω υψηλής χωρητικότητας συνδέσεων μεταφοράς, οι οποίες αναπαρίστανται ως στερεές γραμμές. Τα βέλη αντιπροσωπεύουν τις υπηρεσίες στη ζώνη άκρης που προσφέρονται για να ικανοποιήσουν τις απαιτήσεις ποιότητας υπηρεσίας. Επιπλέον, το διακεκομμένο βέλος αναπαριστά την αναφορά εργασίας προς το σύννεφο, καθώς αυτό βοηθά στην εξοικονόμηση των περιορισμένων πόρων της ζώνης άκρης.

1.3 Στόχος διπλωματικής

Ο στόχος της διπλωματικής εργασία εντοπίζεται στην βελτίωση της ποιότητας υπηρεσίας εφαρμογών IoT. Συγκεκριμένα, η διπλωματική εργασία επικεντρώνεται στην αξιολόγηση, βελτίωση ή βελτιστοποίηση της ποιότητας υπηρεσίας για διάφορες εφαρμογές IoT, μαζί με την προσομοίωση που θα προσπαθήσει αποδοτικά να κάνει καταμερισμό υπολογιστικών πόρων που είναι διαθέσιμοι στο άκρο του δικτύου. Στη συνέχεια, γίνεται μια ανάλυση στις βασικές πτυχές της αξιολόγησης ποιότητας υπηρεσίας και καταλήγει στο συμπέρασμα όπου θα γίνει μια ανάλυση στο τι είδαμε και τι καταφέραμε.

Η αξιολόγηση της ποιότητας υπηρεσίας σε δίκτυα που υποστηρίζουν IoT εφαρμογές είναι κρίσιμη για να διασφαλίσουν την ομαλή λειτουργία και την υψηλή απόδοση τους. Η αξιολόγηση QoS περιλαμβάνει τη μέτρηση, την ανάλυση, και τη βελτίωση μετρικών όπως είναι ο δείκτης cluster GoS, ο δείκτης resource utilization και ο δείκτης cloud contention που επηρεάζουν την ποιότητα της εμπειρίας των χρηστών ή την απόδοση του συστήματος.

Κεφάλαιο 2: Θεωρητικό υπόβαθρο

Το δεύτερο κεφάλαιο της εργασίας έχει ως στόχο να παρουσιάσει τις προηγούμενες μελέτες και έρευνες που έχουν διενεργηθεί σχετικά με το αντικείμενο της διπλωματικής.

Σε αυτό το κεφάλαιο γίνεται ανασκόπηση των σημαντικότερων ερευνητικών εργασιών που έχουν δημοσιευτεί στον τομέα του IoT και τις τεχνολογίες B5G που το περικλείουν. Εστιάζεται ιδιαίτερα σε εργασίες που αφορούν τη βελτίωση της ποιότητας υπηρεσίας και την αποτελεσματική διαχείριση των υπολογιστικών πόρων.

Αυτή η συνεχής εξέλιξη των εργασιών πηγάζει από τον ανθρώπινο εφευρετικό πνευματισμό και την ανάγκη για βελτίωση της ζωής μας. Καθώς συνεχίζουμε να εξερευνούμε το IoT και τις ατελείωτες δυνατότητές του, η έρευνα και η ανάπτυξη παραμένουν το κινητήριο της δύναμη μας για έναν πιο συνδεδεμένο, έξυπνο και βιώσιμο πλανήτη [3].

Τέλος, γίνεται μια σύντομη ανασκόπηση των κυριότερων αρχιτεκτονικών μοντέλων και του αλγορίθμου DQN που έχει χρησιμοποιηθεί στο μοντέλο, με έμφαση στην τεχνολογία του δικτυακού τεμαχισμού και στον τρόπο διαχείρισης των πόρων.

2.1 Ανασκόπηση της βιβλιογραφίας

Στον επερχόμενο κόσμο, η συνεχής έρευνα και κατανόηση της τεχνολογίας του IoT αποτελεί κεντρικό σημείο ενδιαφέροντος στην ανοιχτή βιβλιογραφία. Ο πρωταρχικός στόχος αυτών των μελετών είναι να κατανοήσουν τον πυρήνα της έννοιας και της ιδέας που κρύβονται πίσω από το IoT, μαζί με τη διερεύνηση του τρόπου λειτουργίας αυτής της τεχνολογίας. Το άρθρο [4] μας κάνει μια ιστορική ανασκόπηση και εξέλιξη του IoT, στην εισαγωγή του όρου και στη σημασία του στο μέλλον του διαδικτύου. Το άρθρο επισημαίνει ότι τα IoT χαρακτηρίζονται από την ικανότητα να συνδέουν διαφορετικές συσκευές μεταξύ τους, επεκτείνοντας τα όρια μεταξύ του φυσικού και του ψηφιακού κόσμου.

Πέρα από την απλή εξήγηση των βασικών αρχών, της ιστορικής αναδρομής και της δομής των συστημάτων, οι μελέτες [5] [6] εστιάζουν στην αποκάλυψη των πολλαπλών δυνατοτήτων που έχουν τα IoT στους διάφορους τομείς. Τομείς όπως οι έξυπνες πόλεις, όπου τα συστήματα IoT βοηθούν στη διαχείριση της κυκλοφορίας και την παρακολούθηση της ποιότητας του αέρα, αλλά και στη βελτιστοποίηση των δημοτικών υπηρεσιών, στην υγεία για την παρακολούθηση των ασθενών, στη γεωργία για την αυτοματοποίηση των διαδικασιών και στη βιομηχανία για την

αυτοματοποίηση και την εξατομίκευση της παραγωγής, είναι μόνο λίγα παραδείγματα της εφαρμογής του IoT, πέρα από τις πολλές εφαρμογές που επωφελούνται από τα IoT, υπάρχουν σημαντικές προκλήσεις που πρέπει να αντιμετωπιστούν, όπως η ασφάλεια, η εμπιστοσύνη, η τυποποίηση και η διακυβέρνηση.

Εξίσου σημαντική επιδίωξη των μελετών είναι η βελτίωση των IoT εφαρμογών. Οι μελέτες παρακάτω εστιάζουν στην ανάπτυξη προσεγγίσεων και τεχνικών που επιτρέπουν τη βελτίωση της αξιοπιστίας στα συστήματα αυτά.

Συγκεκριμένα το [7], εστιάζει στον ολοένα αυξανόμενο ρόλο των τεχνολογιών IoT στη βελτίωση διαφόρων πτυχών της ζωής, ιδιαίτερα στον στόχο τους να απλοποιούν τις διαδικασίες και να βελτιώνουν την ποιότητα ζωής. Ωστόσο, με την ταχεία εξέλιξη των τεχνολογιών IoT, είναι ζωτικής σημασίας η αξιολόγησή τους από περιβαλλοντικής άποψης για να διασφαλίσουμε τη βιώσιμη χρησιμοποίηση των πόρων. Το άρθρο αναδεικνύει τις συνεισφορές και τις συζητήσεις από το συνέδριο SpliTech 2019. Καλύπτονται τέσσερις κύριες περιοχές: Βιώσιμη Ενέργεια και Περιβάλλον, Έξυπνη Πόλη, Ηλεκτρονική Υγεία και IoT στις Μεταφορές.

1. **Βιώσιμη Ενέργεια και Περιβάλλον**: Σε αυτήν την ενότητα του άρθρου, εξετάζονται τεχνολογίες που στοχεύουν στη βελτίωση της αποδοτικότητας της ενέργειας, την ανάπτυξη βιώσιμων πηγών ενέργειας και την προστασία του περιβάλλοντος.

2. **Έξυπνη Πόλη**: Οι συζητήσεις εδώ επικεντρώνονται στην εφαρμογή των τεχνολογιών IoT για τη δημιουργία πόλεων που είναι πιο αποδοτικές, βιώσιμες και φιλικές προς τους κατοίκους τους.

3. **Ηλεκτρονική Υγεία**: Η τεχνολογία διαδραματίζει ολοένα και περισσότερο έναν κρίσιμο ρόλο στην υγεία. Στην ενότητα αυτή, εξετάζονται νέες λύσεις για τηλεϊατρική, παρακολούθηση ασθενών και άλλες εφαρμογές που βελτιώνουν την ποιότητα της περίθαλψης.

4. **IoT στις Μεταφορές**: Η εφαρμογή των τεχνολογιών IoT στον τομέα των μεταφορών μπορεί να προσφέρει λύσεις για τη βελτίωση της κυκλοφορίας, της ασφάλειας και της αποδοτικότητας των μεταφορικών μέσων.

Εκτός από τις πολλές δυνατότητες που προσφέρουν, οι τεχνολογίες IoT φέρουν και προκλήσεις που σχετίζονται με την παρακολούθηση του δικτύου και τη συντήρηση του. Ένα μεγάλο πρόβλημα που δημιουργείτε συμφωνά με το άρθρο [7], είναι η αυξημένη παραγωγή ηλεκτρονικού εξοπλισμού που μπορεί να οδηγήσει σε υπερβολική κατανάλωση πρώτων υλών, αυξάνοντας τις ανησυχίες για τη βιωσιμότητα.

2.1.1 Βιβλιογραφική ανασκόπηση αρχιτεκτονικών διαχείρισης δικτυακών πόρων

Τα τελευταία χρόνια έχουν αναπτυχθεί πολλές αρχιτεκτονικές στο πλαίσιο των δικτύων προκειμένου να εξυπηρετηθούν οι αυξημένες απαιτήσεις και οι περιορισμοί των νέων εφαρμογών και υπηρεσιών.

Στην υποενότητα αυτή γίνεται μια σύντομη ανασκόπηση των βασικών αρχιτεκτονικών που έχουν προταθεί, εστιάζοντας στα κυριότερα χαρακτηριστικά τους και στον τρόπο με τον οποίο διαχειρίζονται τους πόρους σε σχέση με τις απαιτήσεις των εφαρμογών. Αυτό θα δώσει μια εποπτική εικόνα του τομέα προτού εξεταστούν αναλυτικότερα ορισμένες από αυτές.

Στην εξέλιξη των δικτύων 5G, ο τεμαχισμός δικτύου (network slicing) σε συνδυασμό με τα FNs αποτελεί την κινητήρια δύναμη για την υποστήριξη ευφυών οχηματικών συστημάτων (IoV) και εφαρμογών έξυπνων πόλεων. Η προτεινόμενη αρχιτεκτονική που έχει πολλά κοινά με την δικιά μας [2] επικεντρώνεται στην ανάπτυξη ενός πλαισίου "network slicing" με βάσει την "ομαδοποίηση στην άκρη" (edge clustering), χρησιμοποιώντας deep reinforcement learning (DRL) για την αποτελεσματική κατανομή των πόρων των FNs. Αυτό επιτρέπει την προσαρμοστική διαχείριση των πόρων στην άκρη του δικτύου, ανταποκρινόμενη στις ετερογενείς απαιτήσεις των εφαρμογών και βελτιστοποιώντας την ποιότητας υπηρεσίας.

Η εργασία [8] έχει παρόμοια αρχιτεκτονική με την δικιά μας, όμως είναι βασισμένη στην End-to-End τεχνολογία και παρουσιάζει μια προηγμένη τεχνική για την ανάλυση και την αλληλεπίδραση δικτύων 5G, με την εφαρμογή του network slicing στην End-to-End επικοινωνία καταφέρνει να κάνει διαχείρισή πόρων μέσο της λειτουργία αλυσίδας SFC. Ο αλγόριθμος που προτείνεται βασίζεται στο DQN και στοχεύει στη δυναμική κατανομή πόρων για τη μεγιστοποίηση του αριθμού των χρηστών που έχουν πρόσβαση. Ο DQN χρησιμοποιείται για να αντιδράσει δυναμικά στις αλλαγές του περιβάλλοντος, ενσωματώνοντας την στην πλευρά πρόσβασης του δικτύου.

Το άρθρο [9] εξετάζει τις προκλήσεις της διαχείρισης πόρων και της ποιότητας υπηρεσίας στα δίκτυα LoRa 5G για εφαρμογές IIoT, το κοινό με την δικιά μας αρχιτεκτονική είναι η χρησιμοποίηση του network slicing. Με την αύξηση της πυκνότητας των συσκευών, το network slicing αποτελεί ένα ζωτικό εργαλείο για την αποτελεσματική διαχείριση των πόρων. Ως απάντηση σε αυτές τις προκλήσεις, το άρθρο προτείνει μια καινοτόμα τεχνική, το deep federated Q-Learning (DFQL), που συνδυάζει την εκμάθηση ενίσχυσης με τεχνικές edge computing, επιτρέποντας έτσι μια πιο προσαρμοστική και ολιστική προσέγγιση στη διαχείριση των πόρων στα δίκτυα 5G IIoT.

Το άρθρο [10], ασχολείται με την προσαρμογή της τεχνολογίας network slicing σε μεγάλης κλίμακας δίκτυα IoT, βασισμένη στη θεωρία του παιχνιδιού, εδώ έχουμε μια σημαντική έρευνα για

της μεγάλας κλίμακας δίκτυα IoT που βοηθάει να καταλάβουμε καλύτερα την αρχιτεκτονική μας. Μέσα από μια SDN-βασισμένη αρχιτεκτονική, το άρθρο προτείνει έναν κατανεμημένο τρόπο διαχείρισης των πόρων του δικτύου, επιτρέποντας στα gateways να συνεργάζονται και να αποφασίζουν για την εκχώρηση των πόρων με βάσει τις απαιτήσεις των διαφορετικών εφαρμογών IoT. Το άρθρο υπογραμμίζει τη σημασία της παροχής ποιότητας υπηρεσίας και παρουσιάζει τρεις διαφορετικές προσεγγίσεις τομοποίησης βασισμένες στη θεωρία του παιχνιδιού. Ειδικότερα, τονίζει την ανάγκη για ολοκληρωμένες λύσεις που θα μπορούν να ανταπεξέλθουν στις αυξανόμενες απαιτήσεις των συσκευών IoT, ενώ ταυτόχρονα θα είναι κλιμακούμενες και αποδοτικές στη χρήση των πόρων.

Η μελέτη [11], εξετάζει συστήματα ΙΟΤ που έχουν ως στόχο να αυτοματοποιήσουν την ανθρώπινη ζωή προσφέροντας υπηρεσίες, το άρθρο μας δίνει μια απλή αρχιτεκτονική για να καταλάβουμε την λογική των εφαρμογών IoT. Η IoT τεχνολογία έχει τη δυνατότητα να συνδέσει διάφορες έξυπνες συσκευές με το διαδίκτυο, καθιστώντας τη ζωή πιο ασφαλή και άνετη. Ωστόσο, για να αξιοποιηθεί πλήρως το δυναμικό του IoT δικτύου, είναι απαραίτητο να οριστούν σαφείς μετρικές ποιότητας υπηρεσίας. Η έρευνα προτείνει μια αρχιτεκτονική IoT, η οποία βασίζεται σε τρία βασικά στοιχεία: τις συσκευές ΙΟΤ, την επικοινωνία και τον υπολογισμό. Οι ΙΟΤ συσκευές ενσωματώνουν αισθητήρες και έχουν τη δυνατότητα σύνδεσης με το διαδίκτυο. Ο τομέας της "επικοινωνίας" καλύπτει τα δίκτυα και τα πρωτοκόλλα που επιτρέπουν την αλληλεπίδραση μεταξύ των συσκευών, ενώ ο "υπολογισμός" αναφέρεται στην επεξεργασία και ανάλυση των δεδομένων που συλλέγονται από τις συσκευές. Το κυριότερο συμπέρασμα της έρευνας είναι η αναγνώριση και περιγραφή διαφόρων μετρικών QoS στο πλαίσιο των IoT αρχιτεκτονικών. Η σαφής οριοθέτηση αυτών των μετρικών είναι ζωτικής σημασίας για τις υπηρεσίες IoT.

Στην παρούσα υπό ενότητα παρουσιάστηκε μία ανασκόπηση των βασικών αρχιτεκτονικών που έχουν αναλυθεί σε πολύ κύρια paper που έχουν προταθεί για το τεχνολογία B5G, με έμφαση στην τεχνολογία του network slicing. Διαπιστώθηκε η σημασία του network slicing για την ικανοποίηση των υψηλών απαιτήσεων των νέων δικτύων και εφαρμογών IoT. Επίσης, παρουσιάστηκαν ορισμένες καινοτόμες τεχνικές που βασίζονται στην τεχνητή νοημοσύνη και ενισχυμένη μάθηση για την αποδοτική διαχείριση των πόρων σε δίκτυα μεγάλης κλίμακας.

2.1.2 Βιβλιογραφική ανασκόπηση τεχνολογιών παροχής ποιότητας Υπηρεσίας (QoS) σε IoT

Σε αυτήν την υποενότητα, θα εξετάσουμε το θεωρητικό υπόβαθρο της έρευνάς μας σχετικά με την ποιότητας υπηρεσίας στο IoT και πώς αυτή εξελίσσεται με την πάροδο του χρόνου κοιτώντας την από την σκοπιά των μελετών.

Στη μελέτη [12], αναλύει τις προκλήσεις που σχετίζονται με την ποιότητας υπηρεσίας στη μετάδοση πληροφοριών ευαίσθητων στην καθυστέρηση εντός του IoT δικτύου, η ανάλυση αυτή θα μας βοηθήσει να μειώσουμε την ταχύτητα εκτέλεσης του μοντέλου μας. Παρότι το δημοφιλές μοντέλο υπηρεσίας "Best Effort" χρησιμοποιείται ευρέως, το άρθρο τονίζει ότι δεν είναι κατάλληλο για την αντιμετώπιση της καθυστερημένης κυκλοφορίας. Λαμβάνοντας υπόψη τα χαρακτηριστικά των συσκευών IoT και τη σημασία των πληροφοριών ευαίσθητων στην καθυστέρηση, η μελέτη προτείνει ένα αναλυτικό μοντέλο για ένα σύστημα ουράς περιορισμένης χωρητικότητας με προτεραιότητα υπηρεσίας προληπτικής επαναφοράς και σχέδιο διαχείρισης buffer push-out. Το μοντέλο εφαρμόζεται για να αξιολογήσει την απόδοση των έξυπνων συσκευών υπό διάφορες συνθήκες κυκλοφορίας με στόχο την τήρηση των περιορισμών ποιότητας υπηρεσίας. Τα αποτελέσματα δείχνουν ότι υπάρχει σαφής βελτίωση στην απόδοση της κυκλοφορίας υψηλής προτεραιότητας σε σύγκριση με την κυκλοφορία χαμηλής προτεραιότητας.

Μια πρόσφατη έρευνα με τίτλο [13] υπογραμμίζει τη σημασία της τεχνολογίας Fog computing για τα συστήματα IoT, αυτή η μελέτη θα μας βοηθήσει να δούμε την αξιοποίηση της ποιότητας υπηρεσίας σε αλλά μοντέλα. Εξετάζεται συγκεκριμένα η ανάγκη για γρήγορη απόκριση και ελαχιστοποίηση καθυστερήσεων, ειδικά στα Vehicular Ad hoc Networks (VANET). Η έρευνα προτείνει την media fog resource estimation (MeFoRE), μια μεθοδολογία που επικεντρώνεται στη βελτίωση της ποιότητας υπηρεσίας με βάση την ποιότητα της εμπειρίας (QoE).

Το άρθρο [14], εξετάζει IoT αρχιτεκτονικές και την επίδραση της τοποθεσίας στην πρόσβαση και επεξεργασία των δεδομένων. Η μελέτη αυτή είναι για καλή ευκαιρία για ανάπτυξη του υπάρχοντας μοντέλου μας. Αρχιτεκτονικές IoT, που ενσωματώνουν την υπολογιστική νοημοσύνη, προσφέρουν πολλαπλές εφαρμογές σε διάφορους τομείς και ανοίγουν νέες ευκαιρίες για τις επιχειρήσεις. Αντιμετωπίζοντας την πρόκληση της καθυστέρησης και της αυξημένης χρήσης δικτύου στο περιβάλλον του cloud computing, το άρθρο παρουσιάζει μια νέα αξιόλογη τεχνική το LAFF, ένα ελαφρύ πλαίσιο που βασίζεται σε έναν αλγόριθμο ενημέρωσης τοποθεσίας. Ο LAFF βελτιώνει την ποιότητας υπηρεσίας μειώνοντας τη χρήση δικτύου, τον χρόνο εξυπηρέτησης και την καθυστέρηση. Συγκεκριμένα, επιτυγχάνει μειώσεις στη μέση καθυστέρηση, τη χρήση δικτύου και τον χρόνο εξυπηρέτησης σε σχέση με άλλα πλαίσια. Το LAFF είναι επίσης σχεδιασμένο να είναι ένα ελαφρύ πλαίσιο, μειώνοντας τη χρήση πόρων της RAM και της CPU.

Στο άρθρο [15], γίνεται αναφορά στο IoT που αποτελεί τη συνένωση διαφόρων τεχνολογιών που αναπτύσσονται γρήγορα παγκοσμίως και μας δίνει μια γενική εικόνα της ποιότητας υπηρεσίας στις εφαρμογές IoT. Για να διασφαλίσουμε την ποιότητα της πρακτικής χρήσης του IoT, είναι απαραίτητο να εξετάσουμε τις απαιτήσεις μέσω της ανάλυσης των χαρακτηριστικών της ποιότητας υπηρεσίας εφαρμογών και του δικτύου μετάδοσης. Το άρθρο προτείνει μια αρχιτεκτονική ποιότητας υπηρεσίας βασισμένη στην επίπεδη δομή του IoT, η οποία θέτει τον παράγοντα ποιότητας υπηρεσίας στα κατώτερα επίπεδα και μεταδίδει τις απαιτήσεις της ποιότητας

υπηρεσίας, προσπαθώντας να εξασφαλίσει τη συνοχή καθώς και να χρησιμοποιήσει αποτελεσματικά την παρεχόμενη ποιότητας υπηρεσίας.

Συμπερασματικά, μπορούμε να πούμε ότι η βιβλιογραφία των τελευταίων ετών έχει δώσει ιδιαίτερη έμφαση στην ποιότητας υπηρεσίας εφαρμογών στο πλαίσιο του IoT, που αυτό μας βοηθάει να αναπτύξουμε το μοντέλο μας με μεγαλύτερη επιτυχία.

2.1.3 Βιβλιογραφικής ανασκόπηση Deep Q-Networks (DQN)

Σε αυτή την υποενότητα θα αναλυθεί ο αλγόριθμος DQN που χρησιμοποιείται στο πλαίσιο της διπλωματικής εργασίας και θα γίνει μια ανάλυση του από τα πρώτα χρονιά που συστήθηκε μαζί με τις νέες εξελίξεις και τα επόμενα βήματα της έρευνας.

Ο DQN που προτάθηκε στο άρθρο [16] το 2015 αναγνωρίζεται ως ένα βασικό σημείο αναφοράς για την έρευνα στη βαθιά ενισχυτική μάθηση, εκεί βασίστηκε και η δικιά μας ιδέα αξιοποίησης του αλγορίθμου στην μελέτη των υπολογιστικών πόρων.

Ο DQN είναι μια αλλαγή του κλασικού αλγορίθμου Q-Learning που παρουσιάζει τρεις κύριες συνεισφορές: μια βαθιά αρχιτεκτονική νευρωνικού δικτύου για την εκτίμηση της Q-συνάρτησης, τη χρήση μίνι-δέσμες τυχαίων δεδομένων εκπαίδευσης αντί για μοναδικές ενημερώσεις βασισμένες στην τελευταία εμπειρία, και τη χρήση παλαιότερων παραμέτρων δικτύου για την εκτίμηση των Q-τιμών της επόμενης κατάστασης. Παρόλα αυτά, η εφαρμογή του αλγορίθμου DQN συνοδεύεται από διάφορες προκλήσεις, όπως η υπερεκπαίδευση όπου η απόδοση του πράκτορα μπορεί να πέσει δραματικά μετά από μια περίοδο εκπαίδευσης.

Το άρθρο [17], παρέχει μια εξαιρετική ανάλυση του αλγορίθμου DQN, εξετάζοντας διάφορες βελτιώσεις και προσαρμογές που μπορούν να γίνουν για την αποτελεσματική εφαρμογή του σε συγκεκριμένα περιβάλλοντα ενισχυτικής μάθησης, προσπαθώντας επίσης να λύσει και το κενό στην βιβλιογραφία που υπήρχε από τις αρχικές έρευνες. Το άρθρο εξετάζει εκτενώς τα πλεονεκτήματα, τις προκλήσεις και τις δυνατότητες του DQN, παρέχοντας σημαντικές κατευθυντήριες γραμμές για την πρακτική εφαρμογή αυτής της τεχνικής.

Βασικό πλεονεκτήματα του DQN περιλαμβάνει τη δυνατότητα αποθήκευσης και επανεξέτασης παραδειγμάτων χρησιμοποιώντας μια μνήμη επανάληψης, βοηθώντας έτσι στην εκπαίδευση του δικτύου σε περιβάλλοντα με μεγάλο όγκο δεδομένων, κάτι που οι προηγούμενες έρευνες παραλείπαν. Επιπλέον, ο DQN αντιμετωπίζει το πρόβλημα των συσχετίσεων μεταξύ διαδοχικών παραδειγμάτων, προσφέροντας σταθερότητα στην εκπαίδευση. Τέλος, μέσω της τεχνικής δικτύων στόχων, ο DQN καταφέρνει να παρέχει συνεχείς και αξιόπιστες εκτιμήσεις της συνάρτησης αξίας, βελτιώνοντας έτσι την απόδοση και την αξιοπιστία της μεθόδου.

Στη σύγχρονη έρευνα πάνω στην ενισχυτική μάθηση, ο αλγόριθμος DQN έχει αποκτήσει ιδιαίτερη δημοτικότητα λόγω της ικανότητάς του να εντοπίζει βέλτιστες πολιτικές για σειριακά προβλήματα απόφασης. Ωστόσο, έχει παρατηρηθεί ότι υπάρχει μια τάση υπερεκτίμησης των τιμών ενέργειας υπό ορισμένες συνθήκες, που αντιμετωπίζεται στην έρευνα που ακολουθεί.

Η μελέτη [18] επεκτείνει την ιδέα του Double Q-learning, προτείνοντας μια προσαρμογή του γνωστού DQN αλγορίθμου, η οποία δείχνει να μειώνει σημαντικά τις υπερεκτιμήσεις. Τα αποτελέσματα από τη χρήση του Double DQN είναι ενθαρρυντικά, καθώς φαίνεται να παράγει πιο ακριβείς εκτιμήσεις τιμών, καθώς και να βελτιώνει την απόδοση σε διάφορες εφαρμογές σύμφωνα με τα ευρήματα της έρευνας, τα αποτελέσματα όμως που έχουμε είναι σε πρωταρχικό στάδιο, οπότε θα κρατήσουμε τον DQN για το πείραμα μας.

2.1.4 Ανάλυση του τρόπου λειτουργίας του EC

Πριν προχωρήσουμε σε συγκρίσεις και αναλύσεις του μοντέλου, θα πρέπει να δούμε την βασική λειτουργεία του προτεινομένου μοντέλου και αξίζει να γίνει μια αναφορά συγκεκριμένα στον τρόπο λειτουργίας του EC. Ο EC είναι υπεύθυνος για τη διαχείριση των υπολογιστικών πόρων εφαρμογών IOT με στόχο τη βελτιστοποίηση της ποιότητας υπηρεσίας. Παρακολουθεί διαρκώς παραμέτρους και παίρνει αποφάσεις για την διαχείριση των διεργασιών.

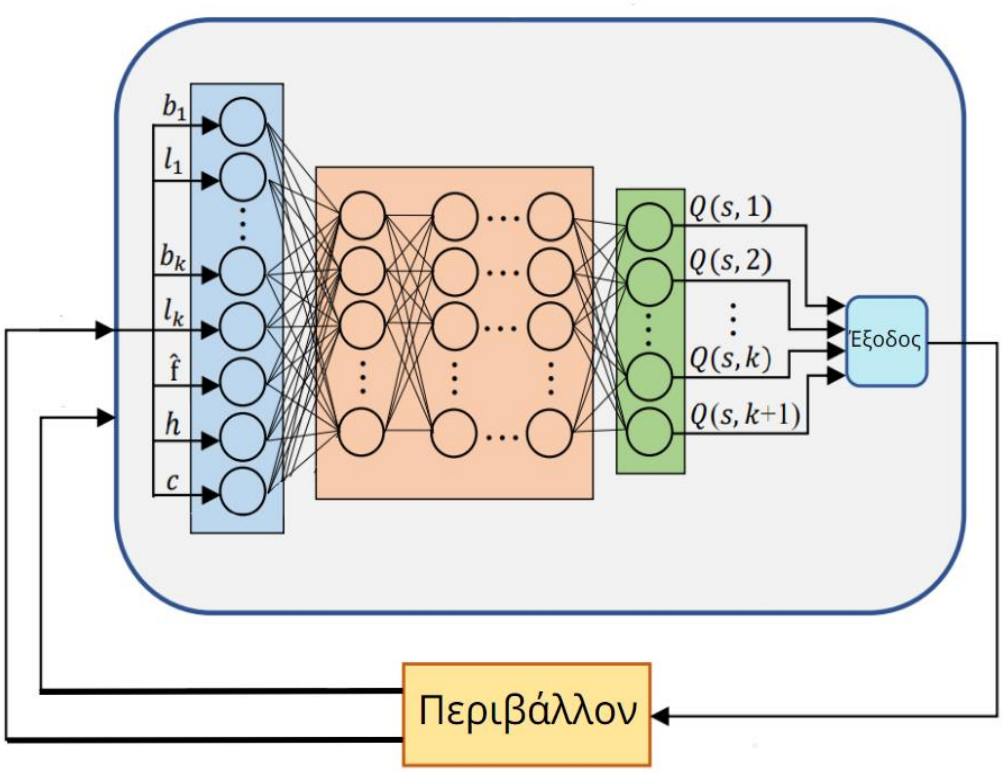

Figure 2. Απεικόνιση του DQN

Το Figure 2 είναι η απεικόνιση του DQN του αλγορίθμου που θα χρησιμοποιήσουμε και αποτελείται από τρία βασικά στάδια, τα οποία συνδυαστικά υλοποιούν ένα ισχυρό σύστημα μηχανικής μάθησης για τη βελτιστοποίηση του δικτυακού κόψιμου.

Στο πρώτο στάδιο της παρατήρησης, λαμβάνονται ως είσοδος οι παράμετροι του περιβάλλοντος από το 5G δίκτυο και τις εφαρμογές. Αυτές περιλαμβάνουν το φορτίο εργασιών, τα χαρακτηριστικά των επερχόμενων αιτημάτων (όνομα κάθε διεργασίας, απαιτούμενοι πόροι και χρόνο εκτέλεσης). Επιπλέον, λαμβάνονται υπόψη οι παράμετροι κάθε cluster, όπως ο αριθμός των ενεργών διεργασιών για αυτήν την χρονική στιγμή και το συνολικό υπόλοιπο φορτίου προς εκτέλεση που αυτά θα του προσαρμόσουν τις ενέργειες σου .

Στη συνέχεια, στο στάδιο της Εκπαίδευσης, ένα βαθύ νευρωνικό δίκτυο παίρνει «μαθήματα» από τα συλλεχθέντα δεδομένα με βάση τον αλγόριθμο DQN, μαθαίνοντας να αναγνωρίζει πρότυπα και να εξάγει χρήσιμα συμπεράσματα.

Τέλος, κατά τη φάση της εφαρμογής, το δημιουργηθέν μοντέλο χρησιμοποιείται για τη λήψη αποφάσεων και διαρκώς βελτιώνει τις ικανότητές του μέσω αυτόματης εκπαίδευσης και ανατροφοδότησης όπως απεικονίζεται ξεκάθαρα και στο διάγραμμα. Οι επαναλαμβανόμενες εκπαιδεύσεις της εξίσωσης του Bellman δημιουργούνε ένα σύστημα που εκτός από τις συνεχές βελτιώσεις, καταφέρνει επίσης να εκ μηδενίσει το λάθος στις αποφάσεις του.

2.2 Βιβλιογραφική ανασκόπηση του προτεινόμενου μοντέλου EC έναντι άλλων μοντέλων

Στην παρούσα ενότητα θα πραγματοποιήσουμε μια συγκριτική ανάλυση του προτεινόμενου μοντέλου διαχείρισης πόρων μέσω EC όπως αναφέρθηκε στην ενότητα 2.1.4 Ανάλυση του τρόπου λειτουργίας του EC με άλλα σχετικά μοντέλα και προσεγγίσεις που αναφέρονται στη βιβλιογραφία. Στόχος αυτής της ανάλυσης είναι να αξιολογήσουμε το δικό μας μοντέλο σε σύγκριση με άλλες επιλογές και να εντοπίσουμε τα ισχυρά και τα αδύναμα σημεία του.

Το προτεινόμενο μοντέλο διαχείρισης αναπτύσσει ένα πλαίσιο για τη διαχείριση υπολογιστικών πόρων στα συστήματα δικτύου, το οποίο παρουσιάζει ομοιότητες αλλά και διαφορές σε σχέση με άλλα μοντέλα που έχουν προταθεί στη βιβλιογραφία, όπως αυτό που περιγράφεται στο αναφερόμενο άρθρο [2]. Κοινό σημείο αποτελεί ο στόχος της προσαρμοστικής κατανομής πόρων στην άκρη. Ενώ το προτεινόμενο μοντέλο βασίζεται σε μια κατανεμημένη προσέγγιση και χρησιμοποιεί το μαρκοβιανό μοντέλο, άλλα μοντέλα όπως αυτό του άρθρου περιλαμβάνουν στοιχεία edge/federated computing και ενισχυμένης μάθησης.

Το μοντέλο διαχείρισης πόρων που παρουσιάζεται στο άρθρο [8] έχει ορισμένες ομοιότητες αλλά και σημαντικές διαφορές σε σχέση με το προτεινόμενο μοντέλο μας. Και τα δύο μοντέλα βασίζονται στη χρήση της τεχνικής ενισχυμένης μάθησης DQN για τη λήψη αποφάσεων διαχείρισης πόρων. Ωστόσο, ενώ το μοντέλο της μελέτης εφαρμόζει τεχνολογίες 5G και δικτύωσης στην end-to-end επικοινωνία, το δικό μας μοντέλο εστιάζει στη διαχείριση στο άκρο του δικτύου. Επιπλέον, ενώ το μοντέλο του άρθρου έχει στόχο τη μεγιστοποίηση των χρηστών, το δικό μας επιδιώκει τη βελτιστοποίηση ποιότητας υπηρεσίας εφαρμογών IOT.

Συγκριτική ανάλυση με το άρθρο [19], το συγκεκριμένο άρθρο παρουσιάζει μια εκτενή επισκόπηση των τεχνολογιών και δυνατοτήτων του δικτύου 6G σε σχέση με τα IoT. Το προτεινόμενο μοντέλο που έχουμε σχεδιάσει έχει γίνει με τρόπο ώστε να είναι αρκετά ευέλικτο και να μπορεί να προσαρμοστεί στα πρότυπα 6G. Συγκεκριμένα, το μοντέλο μας βασίζεται σε αρχές που θα επιτρέψουν την ομαλή μετάβασή του στις αυξημένες απαιτήσεις ταχύτητας, επεξεργασίας και νέων υπηρεσιών του 6G που θα προσφερθούν σύμφωνα με το εν λόγω άρθρο. Με αυτό τον τρόπο, ο EC μπορεί να διασφαλίσει τη βέλτιστη διαχείριση των περιορισμένων πόρων στο άκρο του δικτύου, καλύπτοντας τις απαιτήσεις των χρηστών σε ποιότητα υπηρεσιών.

2.3 Αρχιτεκτονικές CRAN και FRAN

Στο παρόν κεφάλαιο περιγράφεται ο σχεδιασμός της έρευνας που αποτελεί το αντικείμενο της διπλωματικής εργασίας. Το μοντέλο C-RAN βασίζεται στην ενοποίηση των λειτουργιών βάσεως ραδιοσυχνοτήτων (BBU) σε ισχυρούς cloud controllers. Οι BBU εικονικοποιούνται και εκτελούνται στους cloud controllers, ενώ οι μονάδες εκπομπής/λήψης (RRU) παραμένουν στις τοποθεσίες τους συνδεδεμένες μέσω οπτικών ινών.

Το μοντέλο F-RAN επεκτείνει το C-RAN μέσω της κατανομής των BBU σε πολλαπλά επίπεδα στο δίκτυο, συμπεριλαμβανομένου του edge cloud που βρίσκεται κοντά στις RRU. Γι' αυτό εισάγεται το F-RAN, το οποίο επεκτείνει τις λειτουργίες του cloud στα άκρα του δικτύου, μέσω των FNs . Οι FNs είναι εξοπλισμένοι με υπολογιστικούς πόρους, επιτρέποντας την παροχή υπηρεσιών με χαμηλή καθυστέρηση [20] . Έτσι επιτυγχάνεται μεγαλύτερη αξιοπιστία και εγγύτητα στις συσκευές, βελτιωμένη απόκριση και ευελιξία στην κάλυψη των απαιτήσεων των εφαρμογών IoT.

Το πρόβλημα που παρουσιάζει το μοντέλο C-RAN είναι οι μεγάλες καθυστερήσεις που επιφέρει, λόγω της εξάρτησης από το cloud και τους περιορισμένους πόρους μετάδοσης. Σκοπός της παρούσας εργασίας είναι να εξετάσει ένα βελτιωμένο μοντέλο διαχείρισης υπολογιστικών πόρων στα άκρα του δικτύου, το οποίο θα μπορεί να αξιοποιεί πλήρως τους υπολογιστικούς πόρους τόσο του cloud όσο και των FN στην άκρη.

Συγκεκριμένα, βασίζεται σε ένα F-RAN όπου ένας EC αναλαμβάνει να συντονίζει την ανάθεση και τη διανομή των περιορισμένων υπολογιστικών πόρων μεταξύ των FNs που αποτελούν τα edge clusters, με στόχο τη βελτιστοποίηση της αξιοποίησης τους και την ικανοποίηση των απαιτήσεων των διαφόρων εφαρμογών IoT. Μέσω αλγορίθμων μηχανικής μάθησης, ο EC αποσκοπεί στην καλύτερη αξιοποίηση των υπολογιστικών πόρων, εξασφαλίζοντας χαμηλές καθυστερήσεις και ικανοποιητική ικανοποίηση της ποιότητας υπηρεσίας τον IoT εφαρμογών.

Κεφάλαιο 3: Αρχιτεκτονική του συστήματος

3.1 Εισαγωγή στην αρχιτεκτονική συστήματος

Στο παρόν κεφάλαιο, αναλύονται οι βασικές αρχές της αρχιτεκτονικής του συστήματος μας. Η ανάλυση μας επιτρέπει να κατανοήσουμε και να αξιολογήσουμε τον τρόπο λειτουργίας του συστήματος. Ειδικότερα, εξετάζεται η πιθανότητα ότι η υποδομή που απαιτείται για την υποστήριξη της τεχνολογίας edge computing ενδέχεται να μην είναι πάντα διαθέσιμη. Επίσης, γίνεται αναφορά στο ενδεχόμενο ορισμένες διεργασίες να χρειάζεται να διατίθενται στο cloud, παρά να εκτελούνται στο edge δίκτυο, εφόσον οι διαθέσιμοι υπολογιστικοί πόροι του edge δεν είναι επαρκείς.

Οι διάφορες εφαρμογές και υπηρεσίες απαιτούν διαφορετικούς υπολογιστικούς πόρους και χαρακτηριστικά από το δίκτυο, όπως εύρος ζώνης, καθυστέρηση και αξιοπιστία.

3.2 Προτεινόμενο μοντέλο

Ξεκινώντας πιο αναλυτικά θα αναφερθούμε στο μοντέλο μας, το μοντέλο μας θα κάνει διαχείριση των περιορισμένων υπολογιστικών πόρων στην άκρη του δικτύου, θα χρησιμοποιήσουμε ένα Fog Radio Access Network (F-RAN) με δυνατότητα δημιουργίας ξεχωριστών τεμαχίων δικτύου network slices.

Σύμφωνα με αυτό το μοντέλο, οι FN ομαδοποιούνται σε συστάδες και ένας κεντρικός FN σε κάθε συστάδα ορίζεται ως EC. Ο ρόλος του EC είναι καθοριστικός, καθώς αναλαμβάνει το συντονισμό των υπολογιστικών πόρων μεταξύ των FNs της συστάδας και την κατανομή τους στις διεργασίες που τις απαιτούν, με στόχο τη βέλτιστη αξιοποίηση των περιορισμένων υπολογιστικών πόρων στην άκρη του δικτύου.

Θα εξετάσουμε διάφορες προσεγγίσεις που μπορεί να ακολουθήσει το EC προκειμένου να λαμβάνει αποφάσεις για την ανάθεση των υπολογιστικών πόρων των κοντινών FN της συστάδας του σε διεργασίες, για την σωστή διαχείριση των αιτημάτων υπηρεσιών που λαμβάνονται.

Για κάθε αίτημα υπηρεσίας σε ένα συγκεκριμένο cluster, ο EC λαμβάνει την απόφαση ποιος FN θα εκτελέσει την απαραίτητη διεργασία. Συγκεκριμένα, ο EC έχει δύο επιλογές: είτε να εξυπηρετήσει

το αίτημα τοπικά στην άκρη αναθέτοντας τη διεκπεραίωση του σε έναν κατάλληλο FN που διαθέτει επαρκείς υπολογιστικούς πόρους, διαφορετικά αν ο FN δεν διαθέτει του υπολογιστικούς πόρους για την διεργασία, το EC την κατευθύνει προς επεξεργασία στο cloud.

Ο EC λαμβάνει αυτή την τελική απόφαση βάση των διαθέσιμων υπολογιστικών πόρων κάθε FN, των απαιτήσεων της εκάστοτε διεργασίας, με στόχο τη βελτιστοποίηση της χρήσης υπολογιστικών πόρων και της ποιότητας υπηρεσίας. Οι πόροι που κατανέμονται στις διάφορες διεργασίες αφορούν τους υπολογιστικούς πόρους του συστήματος.

Αρχικά, θα αναλύσουμε το μοντέλο μας και θα το εκτελέσουμε για να δούμε τα αποτελέσματα και μετά θα το συγκρίνουμε με άλλες στατικές κατανομές υπολογιστικών πόρων όπως είναι το firstfit και το bestfit, όπου οι αποφάσεις λαμβάνονται εκ των προτέρων κάτω από ορισμένους κανόνες.

Το μοντέλο μας εξετάζει τις πιο προηγμένες τεχνικές, όπως η χρήση τεχνικών μηχανικής μάθησης και ενισχυμένης μάθησης, προκειμένου το EC να μαθαίνει διαρκώς από την εμπειρία του και να προσαρμόζει ανάλογα τις αποφάσεις του, λαμβάνοντας υπόψη παράγοντες, όπως η τρέχουσα κατάσταση φόρτου και οι μεταβαλλόμενες απαιτήσεις.

3.2.1 Τρόπος λειτουργίας προτεινόμενου μοντέλου

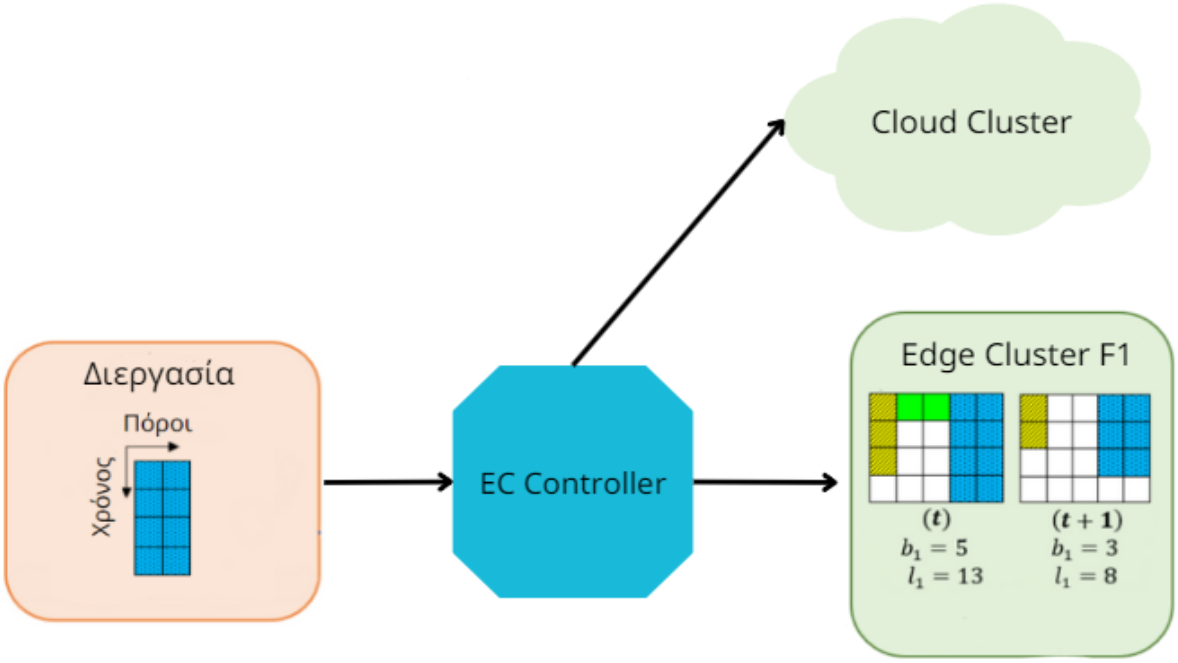

Figure 3. Εσωτερική απεικόνιση του μοντέλου

Το Figure 3, απεικονίζει ένα παράδειγμα τρόπου λειτουργίας του μοντέλου διαχείρισης υπολογιστικών πόρων που έχουμε υλοποιήσει.

Συγκεκριμένα, δείχνει την ανάθεση μιας διεργασίας στο edge cluster F1 από τον EC. Η συγκεκριμένη διεργασία έχει ανάγκη από 2 υπολογιστικούς πόρους για 4 χρόνους, αυτό γίνεται ξεκάθαρο από το πρώτο κουτί που βλέπουμε την διεργασία. Την χρονική στιγμή t, παρατηρούμε τον EC να αναθέτει την εργασία ανάλογα με τους παράγοντες που αναφέραμε, στον cluster F1, σύμφωνα με το μοντέλο που έχουμε αναπτύξει.

Σύμφωνα με το σχήμα ο F1 cluster, διαθέτει 20 υπολογιστικούς πόρους που μπορούν να διατεθούν σε διεργασίες.

- Το b1t δείχνει πόσοι υπολογιστικοί πόροι δεσμεύονται στο cluster κάθε χρονική περίοδο. Με διαφορετικό χρώμα εμφανίζονται οι υπολογιστικοί πόροι που έχουν ανατεθεί σε κάθε διεργασία.
- Το l1t δείχνει τους συνολικούς υπολογιστικούς πόρους που έχουν δεσμεύσει οι διεργασίες που έχουν ανατεθεί στον F1.

Σύμφωνα με το Figure 3 την χρονική στιγμή t εκτελούνται 3 διεργασίες στο cluster F1. Η κίτρινη διεργασία βλέπουμε ότι θέλει 1 υπολογιστικό πόρο για 3 χρόνους, η πράσινη έχει δεσμεύσει 2 υπολογιστικούς πόρους για έναν χρόνο, και η μπλε διεργασία δεσμεύει 2 υπολογιστικούς πόρους για 4 χρόνους. Τα άσπρα κουτάκια είναι οι διαθέσιμοι πόροι που μπορεί ο EC να αναθέσει άλλες διεργασίες που θα χρειαστούν υπολογιστικούς πόρους.

Στο χρονική στιγμή t+1, η πράσινη διεργασία έχει ολοκληρωθεί και οι αντίστοιχοι υπολογιστικοί πόροι της έχουν αποδεσμευτεί. Ταυτόχρονα, οι υπόλοιπες διεργασίες έχουν αποδώσει υπολογιστικούς πόρους πίσω στο σύστημα για 1 χρονική μονάδα, με την κίτρινη διεργασία να αποδίδει 1 υπολογιστικό πόρο και τη μπλε διεργασία να αποδίδει 2 υπολογιστικούς πόρους. Αυτοί οι πόροι έχουν τώρα απελευθερωθεί και είναι διαθέσιμοι για άλλες διεργασίες.

Με αυτό τον τρόπο παρουσιάζεται η δυναμική αλλαγή της κατάστασης των υπολογιστικών πόρων και των διεργασιών σε κάθε cluster σε διαδοχικές χρονικές στιγμές. Η προσέγγιση αυτή μας βοηθά να κατανοήσουμε καλύτερα τη λειτουργία του μοντέλου που αναπτύξαμε.

3.3 Παράγοντες που επηρεάζουν την αρχιτεκτονική του συστήματος

Σε αυτό το κεφάλαιο θα εξεταστούν οι βασικοί παράγοντες που επηρεάζουν τον σχεδιασμό του προτεινόμενου μοντέλου διαχείρισης υπολογιστικών πόρων. Το Internet of Vehicles (IoV) και οι εφαρμογές έξυπνης πόλης έχουν διαφορετικές απαιτήσεις σε ό,τι αφορά την υπολογιστική ισχύ,

το διαθέσιμο εύρος ζώνης, την καθυστέρηση, τη διαθεσιμότητα και την αξιοπιστία. Αυτό είναι απαραίτητο προκειμένου να επιτευχθεί το επιθυμητό επίπεδο ποιότητας υπηρεσίας στον τελικό χρήστη[21]. Ωστόσο, όπως εξηγήσαμε, οι εφαρμογές του Internet of Things έχουν διαφορετικές απαιτήσεις απόδοσης. Για να ικανοποιήσει τις ανάγκες όλων, το μοντέλο πρέπει να προσαρμόζεται δυναμικά μαζί με την αρχιτεκτονική του.

Ορισμένες εφαρμογές στον τομέα του IoV, όπως η προηγμένη ανάλυση κίνησης σε πραγματικό χρόνο και η αυτόματη ανίχνευση και αντίδραση σε απρόβλεπτες καταστάσεις, μπορεί να απαιτούν περισσότερους υπολογιστικούς πόρους και συχνά να εξαρτώνται από τη σύνδεση με τον υπολογιστικό νέφος [21]. Λαμβάνοντας υπόψη τις απαιτήσεις αυτές των εφαρμογών σε πόρους και επεξεργαστική ισχύ, κρίνεται απαραίτητος ο σχεδιασμός της αρχιτεκτονικής του συστήματος με τρόπο που να επιτρέπει την πρόσβαση σε νέφος υπολογιστών όταν απαιτείται.

Αντίθετα, εφαρμογές όπως η αυτόνομη οδήγηση απαιτούν χαμηλές καθυστερήσεις (latency), γιατί εξαρτώνται ανθρώπινες ζωές και είναι σημαντικό να χρησιμοποιούν υπολογιστικούς πόρους άκρης. Επιπλέον, η εκτέλεση ανάλυσης βίντεο και η χρήση τεχνητής νοημοσύνης απαιτούν σημαντικούς υπολογιστικούς πόρους και επεξεργαστική ισχύ.

Από τα παραπάνω παραδείγματα γίνεται κατανοητό ότι απαιτείται μια σωστή αρχιτεκτονική συστήματος βασισμένη στο μοντέλο διαχείρισης πόρων που να μπορεί να διαχειρίζεται όλες αυτές τις διεργασίες με βάση την στιγμιαία διαθεσιμότητα σε υπολογιστικούς πόρους.

Μία τέτοια προσέγγιση αποτελεί ο τεμαχισμός στο cloud. Αυτός αποτελεί οικονομικά αποδοτική λύση, καθώς επιτρέπει τον διαχωρισμό των φυσικών πόρων του κάθε παρόχου cloud σε λογικά τμήματα με κατάλληλα χαρακτηριστικά, προκειμένου να εξυπηρετηθούν με αποδοτικό τρόπο οι ανάγκες των εφαρμογών IoV και έξυπνων πόλεων [2]. Αυτή η προσέγγιση ταιριάζει απόλυτα με την αρχιτεκτονική του προτεινόμενου συστήματος, καθώς επιτρέπει τον δυναμικό τεμαχισμό των πόρων του cloud σε τμήματα με ιδιότητες προσαρμοσμένες στις απαιτήσεις των εφαρμογών.

3.4 Λειτουργία κώδικα

Στο παρόν κεφάλαιο θα παρουσιαστεί ο τρόπος λειτουργίας του κώδικα που αναπτύχθηκε για την υλοποίηση του προτεινόμενου μοντέλο, θα αναλυθούν τα τμήματα του δικτυακού τεμαχισμού που μπορούν να αντιστοιχιστούν στις εφαρμογές IoT, έτσι ώστε να επιτευχθεί ο βέλτιστος καταμερισμός ανάμεσα στις απαιτήσεις κάθε εφαρμογής και τις ικανότητες τμημάτων του cloud, μαζί με τα βασικά στοιχεία κάθε κλάσης, ο κώδικας αποτελείται από τα τρία μέρη, την κλάση του αλγορίθμου DQN, το δικό μας περιβάλλον μοντελοποίησης του προβλήματος και την κύρια συνάρτηση.

1. **Κλάση DQNAgent:**
    - **Μοντέλο Νευρικού Δικτύου:** Το εσωτερικό μοντέλο του πράκτορα αποτελείται από τρία επίπεδα, δύο με 24 νευρώνες και ενεργοποίηση ReLU και ένα επίπεδο εξόδου με γραμμική ενεργοποίηση.
    - **Πολιτική εξερεύνησης:** Εφαρμόζει μια επιλογή epsilon-greedy, ώστε ο πράκτορας να μπορεί να διαλέξει είτε τυχαία είτε βάσει των προβλέψεών του την ενέργεια που θα πραγματοποιήσει.
    - **Εκπαίδευση:** Ο πράκτορας εκπαιδεύεται χρησιμοποιώντας την τεχνική επανάληψης εμπειρίας, όπου επιλέγεται ένα δείγμα από την μνήμη του και το μοντέλο του εκπαιδεύεται για να μειώσει το σφάλμα. Το σφάλμα ορίζεται ως η απόκλιση μεταξύ της εκτίμησης της Q τιμής που παράγει το νευρωνικό δίκτυο για μία κατάσταση και δράση, και της πραγματικής τιμής Q που θα έπρεπε να έχει μάθει βάσει της εμπειρίας του.
2. **Κλάση Environment:**
    - **Πόροι:** Το περιβάλλον διαθέτει έναν πίνακα πόρων που δείχνει τη χρησιμοποίηση των πόρων σε κάθε σύμπλεγμα.
    - **Διεργασίες:** Κάθε διεργασία που ανατίθεται στο περιβάλλον έχει ένα όνομα, την ανάγκη πόρων, τον χρόνο εκτέλεσης και το σύμπλεγμα στο οποίο ανατίθεται.
    - **Ενημέρωση Κατάστασης:** Όταν μια διεργασία ανατίθεται ή ολοκληρώνεται, η κατάσταση του περιβάλλοντος ενημερώνεται αναλόγως.
    - **Διαγραφή πόρων:** Όλες οι διεργασίες χρειάζονται υπολογιστικούς πόρους για συγκεκριμένο χρονικό διάστημα, το περιβάλλον αφαιρεί αυτούς τους πόρους ανά χρονική μονάδα. Για παράδειγμα, αν μια διεργασία απαιτεί 4 πόρους για 2 χρονικές μονάδες, τότε στην πρώτη χρονική μονάδα θα αφαιρούνται 4 πόροι από τη διεργασία, ενώ στη δεύτερη χρονική μονάδα θα αφαιρούνται οι εναπομείναντες 4 πόροι.
    - **Υπολογισμός Reward:** Για κάθε διεργασία που ανατίθεται σε κάποιο cluster, υπολογίζεται η αντίστοιχη ανταμοιβή βάσει του τύπου (3), όπου επιστρέφεται πίσω στον DQNAgent.
3. **Κύρια Συνάρτηση:**
    - Καθορίζει τον τρόπο με τον οποίο ο πράκτορας και το περιβάλλον αλληλεπιδρούν, δημιουργώντας μια διαδικασία εκπαίδευσης και ελέγχου.
    - Χρησιμοποιήθηκε ένα νευρωνικό δίκτυο με 1 εισαγωγικό, 2 κρυφά και 1 εξαγωγικό στρώμα.
    - Το εισαγωγικό στρώμα αντιστοιχεί στην κατάσταση και η έξοδος στις ενέργειες.

- Η εκπαίδευση έγινε με τη μέθοδο Reinforcement Learning DQN.
- Το reward ήταν ένας συνδυασμός της αξιοποίησης πόρων και του cluster GoS βάση του τύπου (3).
- Χρησιμοποιήθηκαν mini-batches 32 τυχαίων εμπειριών από τη μνήμη για κάθε βήμα εκπαίδευσης.
- Το νευρωνικό δίκτυο εκπαιδεύτηκε για 300 επεισόδια, με επανάληψη της διαδικασίας εκπαίδευσης σε κάθε επεισόδιο.
- Συλλέγει τα αποτελέσματα και τα στατιστικά στοιχεία από κάθε επεισόδιο και τα απεικονίζει σε γραφήματα για την ανάλυση.

Μέσω της εφαρμογής του DQN, ο πράκτορας εκπαιδεύεται να λαμβάνει αποφάσεις που βελτιστοποιούν αυτές τις μετρικές, προσφέροντας μια αυτοματοποιημένη λύση για τη διαχείριση υπολογιστικών πόρων.

Εδώ είναι ένας συνοπτικός απολογισμός της λειτουργίας του κώδικα:

1. **Αρχικοποίηση περιβάλλοντος και αυτόματου πράκτορα (DQNAgent):** Αρχικά, αρχικοποιείται το περιβάλλον με τον αριθμό των κόμβων και των διαθέσιμων πόρων ανά κόμβο, και ο DQN αυτόματος πράκτορας με τον αντίστοιχο αριθμό καταστάσεων παρατηρήσεων και ενεργειών.
2. **Εκτέλεση χρόνων:** Κατά τη διάρκεια κάθε χρόνου, ο αυτόματος πράκτορας αλληλεπιδρά με το περιβάλλον ως εξής:
    - Δημιουργείται μια διεργασία με τυχαίες τιμές σύμφωνα με τις τιμές των διαθέσιμων πόρων και του χρόνου.
    - Επιλέγει έναν κόμβο από μια συστάδα βάσει μιας πολιτικής εξερεύνησης-εκμετάλλευσης (εδώ χρησιμοποιείται η έψιλον-άπληστη πολιτική).
    - Αποφασίζει αν θα δοθούν πόροι στην επιλεγμένη ενέργεια για μια διεργασία που απαιτεί πόρους. Εάν υπάρχουν αρκετοί διαθέσιμοι πόροι στον επιλεγμένο κόμβο, οι πόροι δίνονται στη διεργασία αλλιώς ανατίθεται στο cloud.
    - Ανανεώνεται η κατάσταση του περιβάλλοντος.
    - Ο αυτόματος πράκτορας αποθηκεύει την εμπειρία του (κατάσταση, ενέργεια, ανταμοιβή, επόμενη κατάσταση) στη μνήμη του για μεταγενέστερη εκπαίδευση.
3. **Επανάληψη (Replay):** Ο αυτόματος πράκτορας εκπαιδεύεται με βάση τις αποθηκευμένες εμπειρίες του χρησιμοποιώντας 32 τυχαίες εμπειριών από τη μνήμη για κάθε βήμα εκπαίδευσης. Εδώ, εφαρμόζεται η εξίσωση Bellman για τον υπολογισμό της στόχο Q-τιμής και γίνεται εκπαίδευση του νευρωνικού δικτύου για την εκτίμηση των Q-τιμών βάση της εξίσωσης (4).

4. **Υπολογισμός της ανταμοιβής (Reward):** Η ανταμοιβή υπολογίζεται βάσει δύο παραγόντων, της resource utilization και του cluster GoS. Αυτοί οι παράγοντες είναι καίριοι για την επίτευξη των στόχων της προσομοίωσης και η ανταμοιβή αντιστοιχεί στον συνδυασμό τους.
5. **Εξομάλυνση δεδομένων (Smoothing):** Στο τέλος της εκτέλεσης των χρόνων, τα δεδομένα εξομαλύνονται χρησιμοποιώντας μια εκθετική κινούμενη μέση τιμή για βελτιωμένη οπτικοποίηση.
6. **Οπτικοποίηση (Visualization):** Οι μετρικές απεικονίζονται σε γραφήματα για να παρακολουθούμε την απόδοση της προσομοίωσης.

3.5 Περιγραφή παραμέτρων του μοντέλου

Στην παρούσα διπλωματική εργασία όπως έχει αναφερθεί αναπτύσσεται ένα μοντέλο διαχείρισης υπολογιστικών πόρων και τεμαχισμού δικτύου στην άκρη για δίκτυα Fog-RAN. Το πρόβλημα όμως του τεμαχισμού δικτύου στον edge cluster μοντελοποιείται ως ένα πρόβλημα λήψης αποφάσεων με Markov (MDP), το οποίο επιλύεται με τη χρήση ενισχυμένης μάθησης. Ο EC αποτελεί τον πράκτορα που μαθαίνει μέσω εμπειρίας να λαμβάνει τις βέλτιστες αποφάσεις για την ανάθεση διεργασιών των περιορισμένων υπολογιστικών πόρων στα FNs.

3.5.1 Λήψης αποφάσεων markov

Το σύνολο των καταστάσεων (States):

Ως κατάσταση ορίζεται διάνυσμα:

$$st = (b1t, l1t, b2t, l2t, \ldots, bkt, lkt, ft, ct, ht) \quad (1)$$

Όπου:

- b1t ...bkt είναι οι διαθέσιμοι υπολογιστικοί πόροι σε κάθε FN την τρέχουσα χρονική στιγμή t.
- l1t ...lkt είναι οι υπολογιστικοί πόροι που δεσμεύονται σε κάθε FN κατά τη χρονική στιγμή t.
- ft είναι το όνομα της διεργασίας που εξετάζεται.

- ct είναι οι υπολογιστικοί πόροι που απαιτεί η διεργασία.
- ht είναι ο χρόνος εκτέλεσης της διεργασίας.

Το σύνολο των δράσεων (Actions):

Οι δυνατές δράσεις του EC είναι οι εξής:

$$A = \{1, 2, \ldots, k\} \quad (2)$$

- όπου 1-k αντιστοιχούν στην εξυπηρέτηση της διεργασίας από έναν από τους k Fog Nodes του cluster. Ειδικότερα, η δράση 1 σημαίνει ανάθεση στον FN1, η δράση 2 στον FN2 κ.ο.κ.

Η περίπτωση ανάθεσης στο cloud γίνεται αν δεν υπάρχουν υπολογιστικοί πόροι στο διαθέσιμο FNs που έχει επιλέξει ο EC να αναθέσει διεργασίες.

Η ανταμοιβή (Reward):

Η ανταμοιβή (Reward) που λαμβάνει ο Edge Controller μετά την εκτέλεση κάθε δράσης υπολογίζεται με βάση δύο παράγοντες:

$$\text{Reward} = \alpha * \text{Resource Utilization} + \beta * \text{Cluster GoS} \quad (3)$$

- Resource Utilization: Αναφέρεται στο βαθμό αξιοποίησης των διαθέσιμων υπολογιστικών πόρων στα edge clusters. Υπολογίζεται ως ο λόγος των πραγματικά αξιοποιημένων πόρων προς τους συνολικά διαθέσιμους πόρους.
- Cluster GoS: Αναφέρεται στο ποσοστό των αιτημάτων που εξυπηρετήθηκαν στην άκρη χωρίς να απαιτηθεί η χρήση του cloud. Υπολογίζεται ως ο λόγος των διεργασιών που εξυπηρετήθηκαν από την άκρη προς τις συνολικές διεργασίες.
- Όπου τα σταθμιστικά βαρή α και β καθορίζουν τη σημασία κάθε παράγοντα στη συνολική ανταμοιβή. Μπορούν να οριστούν έτσι ώστε α+β=1 ώστε η ανταμοιβή να κυμαίνεται μεταξύ 0 και 1.

Η εξίσωση λήψης αποφάσεων (Bellman):

Ο EC χρησιμοποιεί τις εξισώσεις Bellman προκειμένου να εκπαιδεύσει ένα μοντέλο ενισχυμένης μάθησης (reinforcement learning model) για να εκτιμήσει τις αξίες των καταστάσεων και δράσεων. Οι εξισώσεις Bellman χρησιμοποιούνται στα προβλήματα λήψης αποφάσεων και περιγράφουν τη σχέση:

$$Q(s,a) = E[R_t + \gamma * \max Q(S_t + 1, a') \mid S_t = s, A_t = a] \quad (4)$$

- $Q(s,a)$: είναι η ανταμοιβή που αναμένεται αν ληφθεί η δράση a στην κατάσταση s.
- R_t: είναι η άμεση ανταμοιβή.
- γ: είναι ο συντελεστής διάχυσης.
- $\max Q(S_t + 1, a')$: είναι η μέγιστη μελλοντική ανταμοιβή.

3.5.2 Ανάλυση αλγορίθμου DQN

Σε αυτό το υποκεφάλαιο πραγματοποιείται μια ανάλυση του αλγορίθμου DQN που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου διαχείρισης υπολογιστικών πόρων. Στην βιβλιογραφική ανασκόπηση εξετάστηκαν αναλυτικά οι διάφοροι τύποι ερευνών που πραγματοποιήθηκαν σχετικά με τον αλγόριθμο DQN και πώς αυτός χρησιμοποιείται στο πλαίσιο της διπλωματικής, σε αυτήν την υπό ενότητα θα γίνει μια ανάλυση των χαρακτηριστικών της δικιάς μας υλοποίησης.

Ο αλγόριθμος DQN που υλοποιήσαμε ακολουθεί τη μέθοδο ενισχυμένης μάθησης Q-learning, με σκοπό να εκτιμήσει μια συνάρτηση αξίας Q που περιγράφει την ανταμοιβή που αναμένεται να λάβει το σύστημα αν χρησιμοποιήσει συγκεκριμένη δράση σε κάθε συνδυασμό κατάστασης-δράσης. Η κατάσταση αντιστοιχεί στα δεδομένα εισόδου του agent. Ένα βασικό στοιχείο του DQN που έχει και η δικιάς μας μέθοδο είναι ότι χρησιμοποιεί ένα βαθύ νευρωνικό δίκτυο (DNN) για να εκπαιδεύσει το μοντέλο Q. Αυτό του επιτρέπει να αντιμετωπίσει προβλήματα με μεγάλο αριθμό διαφορετικών καταστάσεων και δράσεων.

Ένα άλλο σημαντικό χαρακτηριστικό είναι ότι αποθηκεύει τις εμπειρίες σε μια μνήμη επανάληψης και τις χρησιμοποιεί για περαιτέρω εκπαίδευση. Αυτό του επιτρέπει να μαθαίνει αποτελεσματικά από μεγάλους όγκους δεδομένων. Τέλος, η εφαρμογή experience replay καταστρώνει τις εμπειρίες σε μίγμα, μειώνοντας έτσι τις επιπτώσεις της συσχέτισης μεταξύ τους. Έτσι είναι σε θέση να εκπαιδεύσει το δίκτυο για την εκτίμηση της Q συνάρτησης με αποδοτικό τρόπο, αυτά τα βασικά χαρακτηριστικά του DQN επιτρέπουν την αποτελεσματική εκπαίδευση του μοντέλου μας και μας επιφέρει τα καλά αποτέλεσμα που πέτυχαμε κατά την εκτέλεση του πειράματος.

Αρχικά, εξηγούμε τη συνάρτηση __init__() που δίνεται παρακάτω.

```
class DQNAgent:
    def __init__(self, state_size, action_size):
        # Initialize the agent's attributes
        self.state_size = state_size  # Size of the state space
        self.action_size = action_size  # Size of the action space
        self.memory = []  # List to store experiences for replay
        self.gamma = 0.95  # Discount factor for Q-learning
        self.epsilon = 1.0  # Initial exploration rate
        self.epsilon_min = 0.01  # Minimum exploration rate
        self.epsilon_decay = 0.995  # Decay rate for exploration
        self.learning_rate = 0.001  # Learning rate for neural network training
        self.model = self._build_model()  # Create the Q-network model
```

- “self.state_size”: Αυτό παίρνει το μέγεθος του χώρου καταστάσεων, που δηλώνει πόσα χαρακτηριστικά χρησιμοποιούνται για την αναπαράσταση κάθε κατάστασης.
- “self.action_size”: Αυτό παίρνει το μέγεθος του χώρου ενεργειών στην συγκεκριμένη περίπτωση είναι τα πόσα FNs έχουμε και καθορίζει πόσες διαφορετικές ενέργειες μπορεί να κάνει ο πράκτορας.

Επιπλέον, αυτές οι γραμμές αρχικοποιούν τα εξής χαρακτηριστικά του πράκτορα:

- “self.memory”: Μια λίστα για την αποθήκευση εμπειριών προκειμένου να χρησιμοποιηθούν για επαναληπτική εκπαίδευση.
- “self.gamma”: Ο παράγοντας ελαχιστοποίησης για το Q-learning, που επηρεάζει τον βαθμό μείωσης της αξίας των μελλοντικών ανταμοιβών.
- “self.epsilon”: Ο αρχικός ρυθμός εξερεύνησης για την επιλογή των ενεργειών.
- “self.epsilon_min”: Ο ελάχιστος ρυθμός εξερεύνησης, καθορισμένος ως κάτω όριο.
- “self.epsilon_decay”: Ο ρυθμός μείωσης του επιπέδου εξερεύνησης με την πάροδο του χρόνου.
- “self.learning_rate”: Ο ρυθμός μάθησης που χρησιμοποιείται κατά την εκπαίδευση του νευρικού δικτύου.
- “self.model”: Το νευρικό δίκτυο (Q-network) που χρησιμοποιείται από τον πράκτορα για την εκτίμηση των Q-τιμών των διαφόρων ενεργειών σε κάθε κατάσταση.

Ο constructor του DQNAgent παρέχει την αρχικοποίηση των βασικών παραμέτρων και τη δημιουργία του νευρικού δικτύου μέσω της μεθόδου _build_model.

Στην συνέχεια, εξηγούμε τη συνάρτηση _build_model(self) που δίνεται παρακάτω.

```
def _build_model(self):
    # Build the neural network model for Q-learning
    model = Sequential()
    # Input layer with 24 neurons and ReLU activation
    model.add(Dense(24, input_dim=self.state_size, activation='relu'))
    # Hidden layer with 24 neurons and ReLU activation
    model.add(Dense(24, activation='relu'))
    # Output layer with linear activation to predict Q-values
    model.add(Dense(self.action_size, activation='linear'))
    # Compile the model with mean squared error loss and Adam optimizer
    model.compile(loss='mse', optimizer=Adam(lr=self.learning_rate))
    return model
```

1. Δημιουργεί ένα νέο μοντέλο χρησιμοποιώντας την **Sequential**, η οποία είναι μια γραμμική στοίβα επιπέδων.
2. Προσθέτετε τρία επίπεδα στο μοντέλο:
    - Το πρώτο επίπεδο είναι το επίπεδο εισόδου (**input layer**) με 24 νευρώνες. Αυτό το επίπεδο δέχεται τις καταστάσεις (**state**) ως είσοδο και χρησιμοποιεί την συνάρτηση ενεργοποίησης ReLU.
    - Το δεύτερο επίπεδο είναι ένα κρυφό επίπεδο (**hidden layer**) με 24 νευρώνες και αυτό το επίπεδο χρησιμοποιεί την συνάρτηση ενεργοποίησης ReLU.
    - Το τρίτο επίπεδο είναι το επίπεδο εξόδου (**output layer**) που χρησιμοποιείται για την πρόβλεψη των Q-τιμών για κάθε δυνατή ενέργεια. Το επίπεδο αυτό έχει γραμμική ενεργοποίηση.
3. Στη συνέχεια, συνδυάζετε το μοντέλο προσδιορίζοντας τη συνάρτηση κόστους και του βελτιστοποιητή. Συγκεκριμένα:
    - Ορίζετε τη συνάρτηση κόστους (**loss**) ως "mse" (Mean Squared Error), που είναι η τυπική συνάρτηση κόστους που χρησιμοποιείται σε προβλήματα Q-learning.
    - Ορίζετε τον βελτιστοποιητή (**optimizer**) ως "Adam" με ρυθμό μάθησης (**learning rate**) που καθορίζεται από την τιμή της μεταβλητής **self.learning_rate**.
4. Τέλος, επιστρέφετε το νευρικό δίκτυο που έχετε δημιουργήσει.

Μετά έχουμε τις μέθοδος remember (self, state, action, reward, next_state, done) και def act (self, state) που σχετίζονται με τον τρόπο που ο πράκτορας αλληλεπιδρά με το περιβάλλον στο πλαίσιο του Q-learning:

```
def remember(self, state, action, reward, next_state, done):
    # Store an experience tuple into the replay memory
    self.memory.append((state, action, reward, next_state, done))

def act(self, state):
    # Choose an action based on epsilon-greedy policy
    if np.random.rand() <= self.epsilon:
        # Explore: choose a random action
        return random.randrange(self.action_size)
    # Exploit: choose the action with maximum predicted Q-value
    act_values = self.model.predict(state.reshape(1, -1))
    return np.argmax(act_values[0])
```

1. **Def remember (self, state, action, reward, next_state, done)**: Αυτή η μέθοδος χρησιμοποιείται για την αποθήκευση μιας εμπειρίας (experience) στη μνήμη αναπαραγωγής (replay memory) του πράκτορα. Οι παράμετροι της μεθόδου είναι οι εξής:

    - **state**: Η τρέχουσα κατάσταση (state) στην οποία βρίσκεται ο πράκτορας και θεωρείται η περιγραφή της κατάστασης των διαθέσιμων υπολογιστικών πόρων σε κάθε κόμβο (πχ διαθέσιμοι/δεσμευμένοι πόροι).

    - **action**: Η ενέργεια που επιλέγει ο πράκτορας στην τρέχουσα κατάσταση και θεωρείται η απόφαση του πράκτορα για την ανάθεση ή μη μιας διεργασίας σε έναν συγκεκριμένο κόμβο ή απόρριψη του και ανάθεση στο cloud, Οι πόροι του cluster δεν θα δεσμευτούν κατά την ανάθεση ενέργειας, αλλά κατά την διάρκεια του ελέγχου αν το συγκεκριμένο cluster διαθέτει του πόρους.

    - **reward**: Η ανταμοιβή (reward) που λαμβάνει ο πράκτορας για την ενέργεια που επιλέγει.

    - **next_state**: Η επόμενη κατάσταση στην οποία μετακινείται ο πράκτορας μετά την ενέργεια.

    - **done**: Ένα flag που υποδηλώνει αν η επιλογή ενέργειας οδηγεί στο τέλος της επικοινωνίας (done=True) ή όχι (done=False).

2. **Def act (self, state)**: Αυτή η μέθοδος χρησιμοποιείται για να επιλέξει την επόμενη ενέργεια που θα ακολουθήσει ο πράκτορας, λαμβάνοντας υπόψη την κατάστασή του. Η μέθοδος υλοποιεί μια στρατηγική εξερεύνησης (exploration) και εκμετάλλευσης (exploitation) που ορίζεται από την παράμετρο **epsilon**, η οποία αντιπροσωπεύει το επίπεδο εξερεύνησης του πράκτορα. Οι βήματα είναι τα εξής:

- Αν ένας τυχαίος αριθμός μεταξύ 0 και 1 είναι μικρότερος ή ίσος με το **epsilon**, τότε ο πράκτορας εξερευνά: δηλαδή, επιλέγει μια τυχαία ενέργεια από το σύνολο των διαθέσιμων ενεργειών.

- Διαφορετικά, ο πράκτορας εκμεταλλεύεται: δηλαδή, χρησιμοποιεί το νευρικό δίκτυο του για να προβλέψει τις Q-τιμές για όλες τις διαθέσιμες ενέργειες και επιλέγει αυτή που έχει τη μεγαλύτερη προβλεπόμενη Q-τιμή.

Η τελευταία def replay(self, batch_size) μέθοδος περιλαμβάνει τη λογική για την εκπαίδευση του Q-νευρικού δικτύου του πράκτορα DQN με χρήση ενός ελαφρύτερου δευτερεύοντος δικτύου μειωμένης πολυπλοκότητας.

```
def replay(self, batch_size):
    # Train the Q-network using a minibatch of experiences
    minibatch = random.sample(self.memory, batch_size)
    for state, action, reward, next_state, done in minibatch:
        target = reward
        if not done:
            # Compute the target Q-value using Bellman equation
            target = (reward + self.gamma * np.amax(self.model.predict(next_state.reshape(1, -1))[0]))
        # Get current Q-values
        target_f = self.model.predict(state.reshape(1, -1))
        # Update the Q-value for the chosen action
        target_f[0][action] = target
        # Fit the model to the new Q-values
        self.model.fit(state.reshape(1, -1), target_f, epochs=1, verbose=0)
    # Decay the exploration rate
    if self.epsilon > self.epsilon_min:
        self.epsilon *= self.epsilon_decay
```

1. **minibatch = random.sample (self. memory, batch_size)**: Επιλογή τυχαίων εμπειριών από τη μνήμη αναπαραγωγής (**self. memory**). Το **batch_size** καθορίζει τον αριθμό των εμπειριών που θα χρησιμοποιηθούν για εκπαίδευση.

2. Κατά τη διαδικασία εκπαίδευσης του DQN, χρησιμοποιείται η τεχνική του mini-batches για την αποτελεσματικότερη εκμάθηση του δικτύου. Συγκεκριμένα, κάθε φορά που γίνεται ενημέρωση των βαρών του δικτύου, δεν χρησιμοποιείται μία μόνο εμπειρία από τη μνήμη αλλά ένα μικρό υποσύνολο (mini-batches) τυχαίων εμπειριών. Αυτό βελτιώνει τη σταθερότητα της εκπαίδευσης και επιταχύνει τη διαδικασία μάθησης του δικτύου νευρώνων.

3. Για κάθε εμπειρία στο **mini-batch**, όπου κάθε εμπειρία αποτελείται από **(state, action, reward, next_state, done)**:

    - **target = reward**: Αρχικοποίηση του target Q-value με την αμοιβή που λήφθηκε από την εμπειρία.

4. Έλεγχος **if not done**:

- Αν ο καταμερισμός υπολογιστικών πόρων δεν ολοκληρώθηκε στην επόμενη κατάσταση (**next_state**), τότε εκτελείται ο ακόλουθος υπολογισμός για τον υπολογισμό του **target** βάσει της εξίσωσης Bellman:
    - **target = (reward + self. gamma * np. amax (self. model. predict (next_state. reshape (1, -1)) [0])**
    - Εδώ, το **self. model. predict (next_state. reshape (1, -1))** υπολογίζει τις Q-τιμές για την επόμενη κατάσταση **next_state**. Στη συνέχεια, το **np.amax** βρίσκει τη μέγιστη Q-τιμή στις προβλεπόμενες Q-τιμές.

5. **target_f = self. model. predict (state. reshape (1, -1))**: Υπολογισμός των τρεχουσών Q-τιμών για την τρέχουσα κατάσταση **state**.
6. **target_f [0] [action] = target**: Ενημέρωση της Q-τιμής που αντιστοιχεί στην ενέργεια **action** με τη νέα υπολογισμένη τιμή **target**.
7. **self.model.fit (state. reshape (1, -1), target_f, epochs=1, verbose=0)**: Εκπαίδευση του Q-νευρικού δικτύου (**self. model**) με τις ενημερωμένες Q-τιμές. Ο αλγόριθμος της επιλογής είναι το Adam, και η συνάρτηση απώλειας είναι η mean squared error (MSE).
8. **if self. epsilon > self. epsilon_min**: Έλεγχος για τον ρυθμό εξερεύνησης **epsilon**. Αν ο **epsilon** είναι μεγαλύτερος από το **epsilon_min**, τότε η τιμή του **epsilon** υποβαθμίζεται με την **epsilon_decay**. Αυτό σημαίνει ότι με την πάροδο του χρόνου, ο πράκτορας αρχίζει να εξερευνάει λιγότερο και να εκμεταλλεύεται περισσότερο τις εκτιμηθείσες Q-τιμές.

Στην Main, εξηγούμε της αρχικοποιήσεις που γίνονται.

```
# Number of nodes per cluster
num_nodes_per_cluster = 5
# Number of resources available per node
num_resources_per_node = 10
# Maximum time a resource can be used
max_time_uses = 5
# Total episodes for the simulation
num_times = 300
# Size of the batch to train the agent
batch_size = 32
# Maximum processes per time
max_process_per_time = 10

# Initialize the environment with the specified nodes and resources per node
env = Environment(num_nodes_per_cluster, num_resources_per_node)
# Initialize the DQN agent with the observation and action spaces from the environment
agent = DQNAgent(env.observation_space, env.action_space)
# Get the initial state (resources)
state = env.resources.flatten()
# Reset the environment to its initial state
env.reset()

# Counters to track total requests, served requests and false allocated on cloud
total_requests_received = 0
total_requests_served = 0
allocated_at_cloud = 0

# Lists to store metrics for visualization
resource_utilization_list = []
cloud_contention_list = []
cluster_GoS_list = []
allocated_at_cloud_list = []
```

1. **num_nodes_per_cluster** και **num_resources_per_node**: Αυτές οι μεταβλητές καθορίζουν τον αριθμό των κόμβων (nodes) ανά cluster και τον αριθμό των διαθέσιμων υπολογιστικών πόρων ανά κόμβο αντίστοιχα.

2. **max_time_uses**: Αυτή η μεταβλητή καθορίζει το μέγιστο χρόνο που μια διεργασία μπορεί να παραμείνει σε κάποιο κόμβων.

3. **num_times**: Το συνολικό αριθμό χρόνων που θα εκτελεστούν κατά τη διάρκεια της προσομοίωσης.

4. **batch_size**: Το μέγεθος της παρτίδας (batch) που θα χρησιμοποιηθεί για την εκπαίδευση του αυτόματου πράκτορα.

5. **env = Environment (num_nodes_per_cluster, num_resources_per_node)**: Δημιουργία του περιβάλλοντος με τους καθορισμένους αριθμούς κόμβων και πόρων ανά κόμβο.

6. **agent = DQNAgent (env. observation_space, env. action_space)**: Δημιουργία του αυτόματου πράκτορα DQN με τον αριθμό των καταστάσεων παρατηρήσεων (**observation_space**) και των ενεργειών (**action_space**) που προσφέρει το περιβάλλον.

7. **state = env.resources. flatten ()**: Αρχικοποίηση της αρχικής κατάστασης (**state**) ως την παραμόρφωση των πόρων στο περιβάλλον σε ένα διάνυσμα.

8. **env. reset ()**: Επαναφορά του περιβάλλοντος στην αρχική του κατάσταση.

9. **total_requests_received** και **total_requests_served**: Αυτοί οι μετρητές χρησιμοποιούνται για τον υπολογισμό των συνολικών αιτημάτων που λαμβάνονται και των αιτημάτων που εξυπηρετούνται κατά την προσομοίωση.

10. **resource_utilization_list**, **cloud_avoidance_list, cluster_GoS_list και allocated_at_cloud_list**: Αυτές οι λίστες χρησιμοποιούνται για την καταγραφή μετρικών.

Τώρα θα εξηγήσουμε το κύριο μέρος του κώδικα και της εκτέλεσης του DQN.

```
for episode in range(num_times):
    max_process = random.randint(1, max_process_per_time)
    for process in range(max_process):

        # Counter to track resource usage in the current episode
        total_resource_usage = 0
        # Randomly decide the resources needed for the current episode
        resources_needed = random.randint(1, num_resources_per_node)
        # Randomly decide the time duration for which the resources are needed
        time = random.randint(1, max_time_uses)
        # Get the action (node) recommended by the agent for the current state
        action = agent.act(state)
        # Initially, assume resources are not allocated
        allocated = False

        # Increment the total requests counter
        total_requests_received += 1
        # Calculate total resources used across all nodes
        for cluster_idx in range(num_nodes_per_cluster):
            total_resource_usage += np.sum(env.resources[cluster_idx])

        # Check if the resources can be allocated in the chosen node
        if np.sum(env.resources[action]) + (resources_needed * time) <= num_resources_per_node:
            total_requests_served += 1  # Increment the served requests counter
            allocated = True

        # If resources are allocated, update the environment with the allocation
        if allocated:
            process_name = f"Process {random.randint(0,999)}"
            next_state, done = env.step(action, resources_needed, process_name, time)
            # If resources are not allocated, continue with the current state
        else:
            next_state = state
            done = False

        if allocated == False and resources_needed * time < num_resources_per_node:
            allocated_at_cloud += 1

        # Train the agent if enough experiences are collected
        if len(agent.memory) > batch_size:
            agent.replay(batch_size)
```

```
# Calculate and store metrics for the current time
cluster_GoS = (total_requests_served / total_requests_received) * 100
cluster_GoS_list.append(cluster_GoS)

cloud_contention = (1 - (total_requests_served / total_requests_received)) * 100
cloud_contention_list.append(cloud_contention)

resource_utilization = total_resource_usage / (num_nodes_per_cluster * num_resources_per_node) * 100
resource_utilization_list.append(resource_utilization)

cluster_but_cloud_process = int ((allocated_at_cloud/total_requests_received) * 100)
allocated_at_cloud_list.append(cluster_but_cloud_process)


# Calculate the reward based on the current episode's metrics
reward = env.calculate_reward(cloud_contention, resource_utilization)
# Store the experience in the agent's memory
agent.remember(state, action, reward, next_state, done)
# Update the state to the next state
state = next_state

# Check every processes and free up resources based on time
# on every process remove resource/time
env.check_process()
# Store the current processes for future reference
env.save_previous_processes()
```

1. Έναρξη βρόχου επεισοδίων: Ο βρόχος εκτελείται για τον αριθμό των χρόνων που καθορίστηκε με τη μεταβλητή **num_times ,** σε αυτούς τους χρόνους έχουμε ορίσει να εκτελούνται παραπάνω από μια διεργασία με το δεύτερο for και την εντολή **max_process** .

2. **total_resource_usage** αρχικοποιείται για να καταγράψει το συνολικό αριθμό των πόρων που χρησιμοποιούνται σε όλους τους κόμβους όλων των συστάδων (cluster) κατά το τρέχον χρόνο, στο συγκεκριμένο παράδειγμα έχουμε μια συστάδα.

3. Επιλογή τυχαίων απαιτούμενων πόρων **resources_needed** και διάρκειας **time** για το τρέχον χρόνο.

4. Ο αυτόνομος πράκτορας DQN επιλέγει μια ενέργεια (κόμβο) με βάση την τρέχουσα κατάσταση (**state**) χρησιμοποιώντας τη μέθοδο **act(state)**.

5. Αρχικά, θεωρείται ότι οι πόροι από τον κόμβο που έχει επιλεγεί για να γίνει η ανάθεση της διεργασίας δεν έχουν δοθεί, γιατί ακόμα δεν ξέρουμε αν φτάνουν η διαθέσιμοι πόροι του κόμβου της συστάδας, επομένως η μεταβλητή **allocated** ορίζεται σε False.

6. Ο μετρητής **total_requests_received** αυξάνεται κατά ένα, δεδομένου ότι ένα αίτημα παραλήφθηκε.

7. Υπολογίζεται το συνολικό αριθμό των πόρων που χρησιμοποιούνται σε όλους τους κόμβους (clusters) και αποθηκεύεται στη μεταβλητή **total_resource_usage**.

8. Γίνεται έλεγχος για να διαπιστωθεί εάν οι απαιτούμενοι πόροι μπορούν να ανατεθούν στον επιλεγμένο κόμβο (**action**). Αν μπορούν, η μεταβλητή **allocated** ορίζεται σε True και αυξάνεται ο μετρητής **total_requests_served** για να καταγραφεί ότι το αίτημα εξυπηρετήθηκε.

9. Αν οι πόροι ανατέθηκαν, η μέθοδος **env.step** καλείται για να ενημερώσει το περιβάλλον με την ανάθεση πόρων και να επιστρέψει την επόμενη κατάσταση (**next_state**) και μια μεταβλητή **done** που υποδεικνύει εάν το επεισόδιο ολοκληρώθηκε.

10. Αν οι πόροι δεν ανατέθηκαν, το επεισόδιο συνεχίζεται με την τρέχουσα κατάσταση (**next_state** ισούται με την **state**) και η μεταβλητή **done** ορίζεται ως False και η διεργασία ανατίθεται στο cloud.

11. Εάν ο αριθμός των εμπειριών στη μνήμη του αυτόματου πράκτορα είναι μεγαλύτερος από το **batch_size**, τότε καλείται η μέθοδος **agent.replay(batch_size)** για την εκπαίδευση του πράκτορα.

12. Υπολογίζονται και αποθηκεύονται μετρικές για το τρέχον επεισόδιο, όπως το cluster GoS, η resource utilization και το cloud contention.

13. Υπολογίζεται η ανταμοιβή (**reward**) βάσει των μετρικών του επεισοδίου.

14. Η εμπειρία (**state**, **action**, **reward**, **next_state**, **done**) αποθηκεύεται στη μνήμη του αυτόματου πράκτορα.

15. Εκτελείται η μέθοδος **env.check_process()** για να ελευθερώσει πόρους από διαδικασίες για εκείνη την χρονική στιγμή.

16. Οι προηγούμενες διαδικασίες αποθηκεύονται στην **env.save_previous_processes()** για μελλοντική αναφορά.

```
# Smooth the resource utilization data for better visualization
smoothed_resource_utilization = smooth_data(resource_utilization_list)

# Plot the metrics
fig, ax = plt.subplots()
ax.plot(smoothed_resource_utilization, color='b', label='Resource Utilization')
ax.plot(cluster_GoS_list, color='g', label='Cluster GoS')
ax.plot(cloud_contention_list, color='r', label='Cloud Contention')
ax.set_xlabel('Time')
ax.legend()

fig, ax = plt.subplots()
ax.plot(allocated_at_cloud_list, color='g', label='Process False Served by Cloud')
ax.set_xlabel('Time')
ax.legend()
```

Τελευταίο θα εξηγήσουμε το μέρος του κώδικα με την οπτικοποίηση των δεδομένων.

1. **Εξομάλυνση Δεδομένων (Smoothing):**

    - Χρησιμοποιείται η συνάρτηση **smooth_data** για να εξομαλυνθούν τα δεδομένα χρήσης πόρων. Η συνάρτηση **smooth_data** λαμβάνει έναν πίνακα δεδομένων και εφαρμόζει τον αλγόριθμο εκθετικής κινούμενης μέσης τιμής για εξομάλυνση.

2. **Δημιουργία Διαγραμμάτων (Plotting):**

    - Δημιουργείται ένα διάγραμμα που παρουσιάζει την εξομαλυνθείσα χρήση πόρων, το Cluster GoS και την Cloud Contention.

    - Δημιουργείται ένα δεύτερο διάγραμμα που παρουσιάζει οι διεργασίες που εξυπηρετήθηκαν από το cloud λανθασμένα ενώ έπρεπε να εξυπηρετηθούν από το cluster (Process False Served by Cloud).

3.5.3 Εκφράσεις επίδοσης κατανομής υπολογιστικών πόρων

Σκοπός είναι η εκμάθηση από τον EC της βέλτιστης πολιτικής δράσης μέσω της αλληλεπίδρασής του με το περιβάλλον, ώστε να κατανέμει αποδοτικά τους υπολογιστικούς πόρους και να ικανοποιεί τις απαιτήσεις των εφαρμογών με διαφορετικές ανάγκες καθυστέρησης.

Για την ερμηνεία των αποτελεσμάτων, αξιολογήθηκε η αποδοτικότητα του προτεινόμενου μοντέλου βελτιστοποίησης του καταμερισμού υπολογιστικών πόρων μέσω των δεικτών resource utilization και cloud contention βασισμένοι στους τύπους που ορίστηκαν από το paper [2] και μια δική μας υλοποίηση του GoS, ο cluster GoS.

Ο δείκτης resource utilization αφορά το βαθμό αξιοποίησης των υπολογιστικών πόρων στο επίπεδο της άκρης και υπολογίζεται ως ο λόγος των πραγματικά αξιοποιημένων υπολογιστικών πόρων προς τους συνολικά διαθέσιμους υπολογιστικούς πόρους της συστάδας j.

$$\mathrm{RU}_j = \frac{\mathrm{Uused}}{\mathrm{Utotal}} = \frac{1}{\mathrm{T}} \sum_{\mathrm{t}=0}^{\mathrm{T}} \frac{\sum_{\mathrm{i}=1}^{\mathrm{k_j}} \mathrm{b_{it}}}{\sum_{\mathrm{i}=1}^{\mathrm{k_j}} \mathrm{N_i}} \quad (5)$$

- RU_j: είναι ο δείκτης Resource Utilization *της συστάδας j*.
- Uused: είναι οι πραγματικά αξιοποιημένοι υπολογιστικοί πόροι.
- Utotal: είναι οι συνολικά διαθέσιμοι υπολογιστικοί πόροι.
- T: είναι το πλήθος των χρονικών στιγμών ανάθεσης πόρων.
- $\mathrm{b_{it}}$: οι πόροι του $\mathrm{i}-$ οστού κόμβου στη χρονική στιγμή t που αξιοποιούνται απο διεργασίες.
- $\mathrm{N_i}$: οι συνολικοί πόροι του $\mathrm{i}-$ οστού κόμβου.
- $\mathrm{k_{ji}}$: είναι ο αριθμός των κόμβων (FNs) που απαρτίζουν την $\mathrm{j}-$ οστή συστάδα .

Ο δείκτης cluster grade of service που είναι μια δική μας υλοποίηση grade of service, υπολογίζει το ποσοστό των αιτημάτων που εξυπηρετήθηκαν στην άκρη χωρίς να απαιτηθεί η χρήση του σύννεφου. Αποτελεί μέτρο της ικανότητας του μοντέλου να αποφεύγει τον φόρτο στον πυρήνα του δικτύου και υπολογίζεται ως ο λόγος των διεργασιών που εξυπηρετήθηκαν από την άκρη από κάποια συστάδα προς τις συνολικές διεργασίες.

$$\text{Cluster Gos}_j = \frac{\mathrm{Pedge}_j}{\mathrm{Ptotal}_j} = \frac{\sum_{\mathrm{t}=0}^{\mathrm{T}} 1_{\{\mathrm{a}_t \in \{1,2,\ldots,\mathrm{k}_j\}\}}}{\mathrm{M}_j} \quad (6)$$

- Cluster $\mathrm{GoS_j}$: είναι ο δείκτης Cluster Grade of Service στην συστάδα j.
- $\mathrm{Pedge_j}$: είναι οι διεργασίες που εξυπηρετούνται από την j συστάδα.
- $\mathrm{Ptotal_j}$: είναι οι συνολικές διεργασίες που αιτούνται εκτέλεση στην j συστάδα.
- $1_{\{\}}$: είναι η συνάρτηση δείκτη (indicator function) παίρνει την τιμή 1 όταν η πρόταση στο ενδεικτικό της είναι αληθής, και 0 όταν είναι ψευδής.
- T: συνολικό χρονικό διάστημα που εξετάζεται
- a_t : αντιπροσωπεύει την επιλογή ενός κόμβου της συστάδας j για την εκτέλεση μιας διεργασίας.
- $\mathrm{M_j}$: Συνολικές διεργασίες που αιτούνται εκτέλεση στην j συστάδα κατά τη διάρκεια T.
- kj: Ανώτατος αριθμός των FN κόμβων της συστάδας j.

Ο δείκτης cloud contention υπολογίζει το ποσοστό των αιτημάτων που εξυπηρετήθηκαν στο νέφος. Αποτελεί μέτρο ελέγχου της της ικανότητας του μοντέλου να αποφεύγει τον φόρτο στον πυρήνα του δικτύου και υπολογίζεται ως ο 1 μείον λόγος των διεργασιών που εξυπηρετήθηκαν από την άκρη προς τις συνολικές διεργασίες.

$$\text{Cloud contention} = 1 - \sum_{j=1}^{l} Cluster\ Gos_j \quad (7)$$

- Cloud contention: ο δείκτης εξυπηρέτησης διεργασιών στο cloud.
- Cluster GoSj: είναι ο δείκτης Cluster Grade of Service στην συστάδα j.
- l: το πλήθος των συστάδων

Κεφάλαιο 4: Αποτελέσματα

4.1 Εισαγωγή στα αποτελέσματα

Στο τέταρτο κεφάλαιο της διπλωματικής εργασίας, παρουσιάζεται το πείραμα που διενεργήθηκε για την αξιολόγηση του προτεινόμενου μοντέλου διαχείρισης πόρων μέσω του EC. Το πείραμα σχεδιάστηκε και διενεργήθηκε με σκοπό την αξιολόγηση της απόδοσης και της αποτελεσματικότητας του προτεινόμενου συστήματος διαχείρισης πόρων στο περιβάλλον του edge cluster.

Το πείραμα αποτελεί κρίσιμο βήμα της έρευνάς μας καθώς μας επιτρέπει να εφαρμόσουμε τις θεωρητικές αρχές του μοντέλου σε πραγματικές συνθήκες και να δούμε αν όλα αυτά που διερευνήθηκαν στην αρχιτεκτονική του συστήματος, μπορούν να επαληθευτούν. Θα αναλυθούν επίσης οι ρυθμίσεις του πειράματος, όπως ο αριθμός των edge clusters και οι πόροι στα cluster, τα χαρακτηριστικά των διαφόρων εφαρμογών και των αιτημάτων.

Πιο συγκεκριμένα, θα περιγραφεί ο αριθμός και οι τεχνικές προδιαγραφές των συστατικών του δικτύου (Fog nodes, Edge clusters). Επίσης, θα παρουσιαστούν τα διαφορετικά προφίλ των εφαρμογών και αιτημάτων που εισήχθησαν, όπως οι παράμετροι φόρτου εργασίας και χρόνου εκτέλεσης. Τέλος, θα γίνει αναφορά στις διαδικασίες που ακολουθήθηκαν για την υλοποίηση των διαφόρων σεναρίων μαζί με σύγκριση υλοποιήσεων διαφορετικών αλγόριθμών.

Στη συνέχεια, θα παρουσιαστούν τα αποτελέσματα των δοκιμών που πραγματοποιήθηκαν με διάφορα σενάρια φόρτου και παραμέτρων, παρέχοντας μετρήσεις βασικών δεικτών QoS, ώστε να μην φορτώνεται το backbone δίκτυο και ο βαθμός αξιοποίησης των περιορισμένων πόρων στην άκρη [2].

Τα αποτελέσματα του πειράματος θα μας δώσουν ουσιαστικές ενδείξεις σχετικά με την αποδοτικότητα και τις δυνατότητες βελτίωσης του μοντέλου μαζί με μια ιδέα τι θα μπορούσε να ήταν αδύνατο να υλοποιηθεί σε ένα πραγματικό πείραμα και να ελεγχθεί η συμπεριφορά του μοντέλου σε διαφορετικά σενάρια φόρτου. Θα συζητηθούν τα αποτελέσματα από την μελέτη και ο βαθμός που επιτεύχθηκαν οι στόχοι της έρευνας και θα προχωρήσουμε στα συμπεράσματα όπου θα βγάλουμε το αποκορύφωμα της εργασίας και θα προτείνουμε τα επόμενα βήματα της έρευνας.

Με αυτό τον τρόπο, θα έχουμε μια πιο ολοκληρωμένη εικόνα για τη λειτουργία και την αποτελεσματικότητα του προτεινόμενου μοντέλου και θα έχουμε ολοκληρώσει τον σκοπό της διπλωματικής που είναι η ποιότητα υπηρεσίας στο διαδίκτυο των πραγμάτων μέσω αξιοποίησης των υπολογιστικών πόρων στην άκρη του δικτύου.

4.2 Ερμηνεία των αποτελεσμάτων

4.2.1 Αποτελέσματα αλγορίθμου DQN

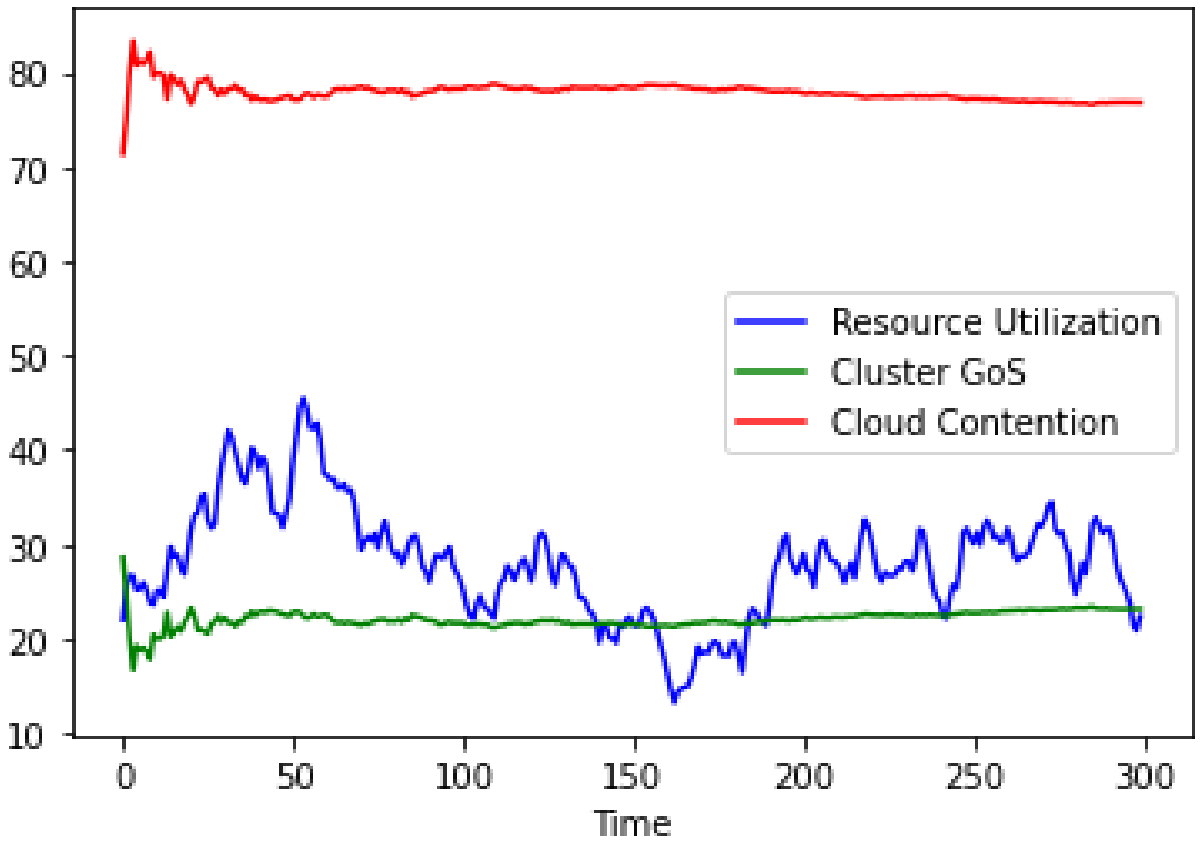

Figure 4. Αποτέλεσμα εκτέλεσης DQN

Στο Figure 4 έχουμε τα αποτελέσματα της ανάθεσης των υπολογιστικών πόρων σε εφαρμογές IoT στο cloud και στα cluster κάνοντας χρήση του προτεινόμενου μοντέλου που δημιουργήσαμε με την χρήση του DQN αλγορίθμου.

Κάθε νέος χρόνος στη διάρκεια του πειράματος αντιστοιχεί σε αιτήματα νέων διεργασιών που μπορούν να έρθουν με μέγιστο αριθμό 10 διεργασίες ανά χρόνο. Κατά τη διάρκεια του χρόνου, λαμβάνουμε αποτελέσματα, στα οποία παρατηρούμε το ποσοστό των πόρων που χρησιμοποιούνται καθώς, πόσες διεργασίες καταφέρνουμε να αναθέτουμε στους διάφορους κόμβους των cluster σε σχέση με αυτές που παρουσιάζονται και κατά πόσο καταφέραμε να αποφύγουμε το cloud.

Ο χρόνος που η κάθε διεργασία μπορεί να εκτελείται στο cluster λαμβάνει τιμές από 1 έως 5, ενώ οι διεργασίες μπορούν να λαμβάνου υπολογιστικούς πόρους από 1 έως 10 για κάθε χρόνο. Ο EC προσπαθεί να κατανείμει αυτές τις διεργασίες σε cluster, εάν αυτό είναι δυνατό, αλλιώς ανατίθενται στο cloud.

Διαθέτουμε 5 nodes με 10 πόρους το κάθε ένα. Τα nodes είναι οργανωμένα σε συστάδες και για το συγκεκριμένο πείραμα διαθέτουμε μια συστάδα. Ο άξονας x αντιστοιχεί στο χρόνο της εκτέλεσης, ενώ ο άξονας y σε ποσοστά για κάθε μετρική. Ο DQN δεν δίνει σωστά αποτελέσματα κατά τους 32 πρώτους χρόνους επειδή το μοντέλο εξακολουθεί να μαθαίνει από τις παρατηρήσεις, είναι ασταθές και οι αποφάσεις που παίρνει είναι τυχαίες. Γι' αυτό, παρατηρούμε μια σημαντική αλλαγή στην τιμή της μεταβλητής cluster GoS και cloud contention, όμως μετά έχει σταθερή απόδοση.

Φαίνεται ότι αξιοποίηση των πόρων της συστάδας από διεργασίες είναι περίπου 20-25% και το cloud contention κοντά στο 80%, αλλά αυτές οι τιμές ισχύουν καθώς τα nodes μπορούν να εξυπηρετήσουν διεργασίες με χαμηλούς πόρους. Ως αποτέλεσμα, η συστάδα φαίνεται να αξιοποιεί περίπου το 30-50% των πόρων των κόμβων της κάθε χρονική στιγμή και οι διεργασίες φαίνονται να εκτελούνται κατά κύριο λόγο στο cloud

Στο συγκεκριμένο πείραμα οι διεργασίες μπορεί να απαιτούν από 1 έως 50 υπολογιστικούς πόρους, ενώ τα cluster μπορεί να παρέχουν έως 10 πόρους το καθένα σε διεργασίες, με 50 πόρους συνολικά καθώς έχουμε 5 nodes στην συγκεκριμένη συστάδα. Για τον λόγο αυτό οι διεργασίες εκτελούνται κατά κύριο λόγο στο cloud.

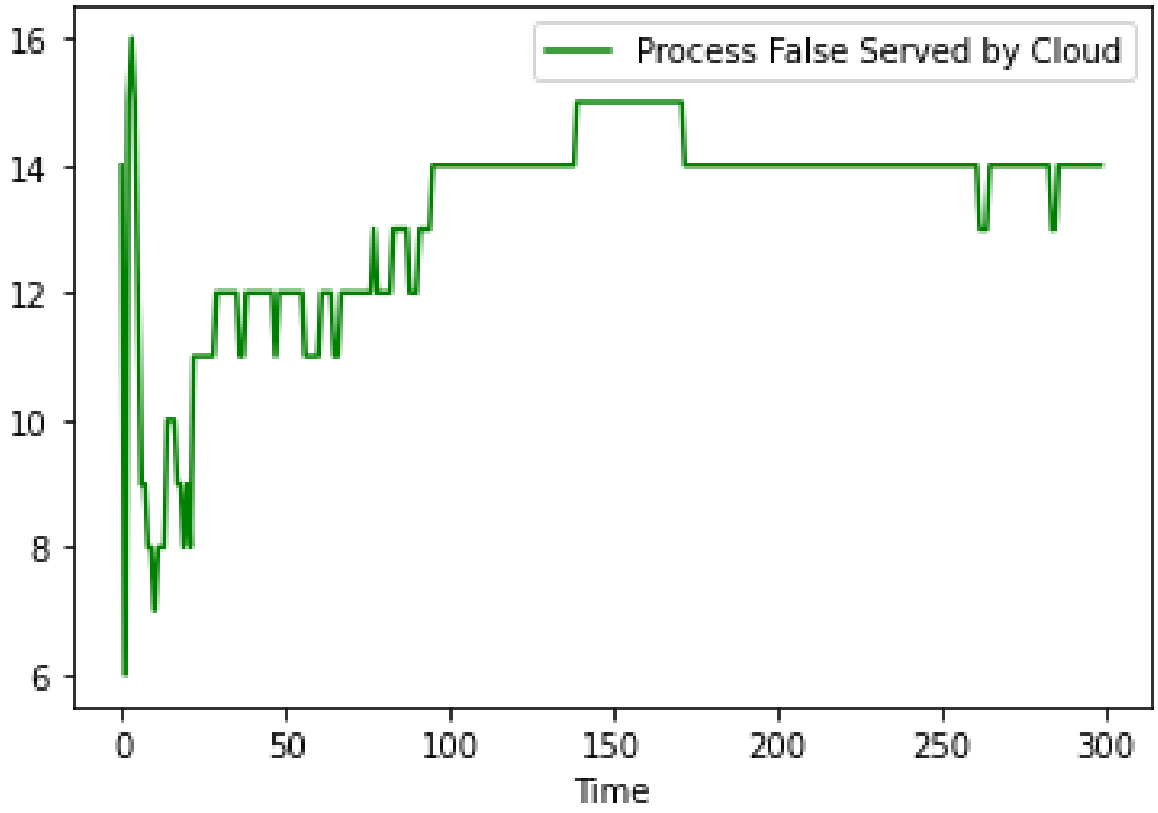

Figure 5. Αποτέλεσμα λανθασμένης εκτέλεσης στο Cloud.

Το Figure 5 που απεικονίζει το ποσοστό των διεργασιών που εξυπηρετήθηκαν από το cloud από λανθασμένη απόφαση του DQN, και μας παρέχει σημαντικές πληροφορίες σχετικά με την απόδοση του μοντέλου διαχείρισης πόρων που αναπτύξαμε.

Παρατηρούμε ότι το ποσοστό των διεργασιών που εκτελέστηκαν στο cloud ενώ θα έπρεπε να εκτελεστούν στο cluster κυμαίνεται στο 13-14% του συνόλου των διεργασιών. Αυτό σημαίνει ότι το μοντέλο μας επέτυχε ποσοστό επιτυχίας περίπου 86-87% στο να δρομολογήσει στα edge clusters εκείνες τις διεργασίες που απαιτούσαν άμεση επεξεργασία, εξασφαλίζοντας την γρήγορη και αποδοτική εκτέλεσή τους, αυτό έδειξε ότι το μοντέλο μας είναι σε θέση να εξασφαλίσει στις σημαντικές διεργασίες τους πόρους που χρειάζονται για εκτελεστούν γρηγορά χωρίς να τα δρομολογήσει στο cloud.

Ακολουθεί η συγκριτική ανάλυση του DQN με απλούς αλγορίθμους κατανομής υπολογιστικού φορτίου (ενότητα 4.2.2) και αποτελέσματα του DQN με διαφορετική υλοποίηση του reward.

4.2.2 Σύγκριση αποτελεσμάτων με στατικούς αλγορίθμους

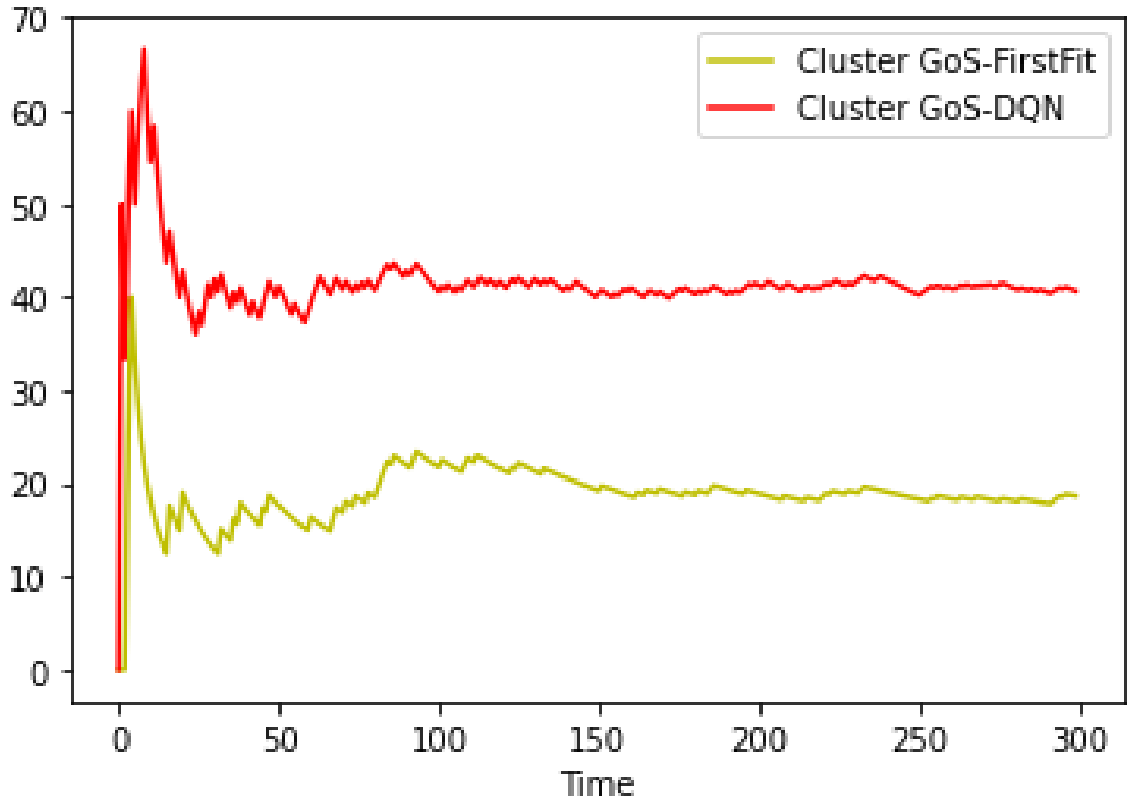

Figure 6. Σύγκριση DQN με την υλοποίηση First-Fit

Η υλοποίησή μας φαίνεται να επιτυγχάνει διπλάσιο cluster GoS σε σχέση με την μέθοδο first-fit σύμφωνα με το Figure 6. Αυτό καταδεικνύει την ανωτερότητα της προσέγγισής μας, όπου χρησιμοποιούμε μια πιο δυναμική και προσαρμοστική τεχνική για τη διαχείριση του δικτύου.

Για τον λόγο του πειράματος έχουμε 2 cluster με τους ίδιους κόμβους που λαμβάνουν τις ίδιες διεργασίες και η κάθε συστάδα επεξεργάζεται τα αποτελέσματα με βάσει την μέθοδο της. Βασισμένοι στα αποτελέσματα, είναι εμφανές ότι η μέθοδος DQN μας προσφέρει ένα σημαντικά βελτιωμένο πλαίσιο για την αποτελεσματική αξιοποίηση των διεργασιών, οδηγώντας σε μεγαλύτερη απόδοση και αποτελεσματικότητα. Αυτό μπορεί να αποδοθεί στην ικανότητα του DQN να μαθαίνει και να προσαρμόζεται στις διαφορετικές συνθήκες του δικτύου, ενώ η μέθοδος first-fit μπορεί να μην ανταποκρίνεται εξίσου καλά σε δυναμικά περιβάλλοντα.

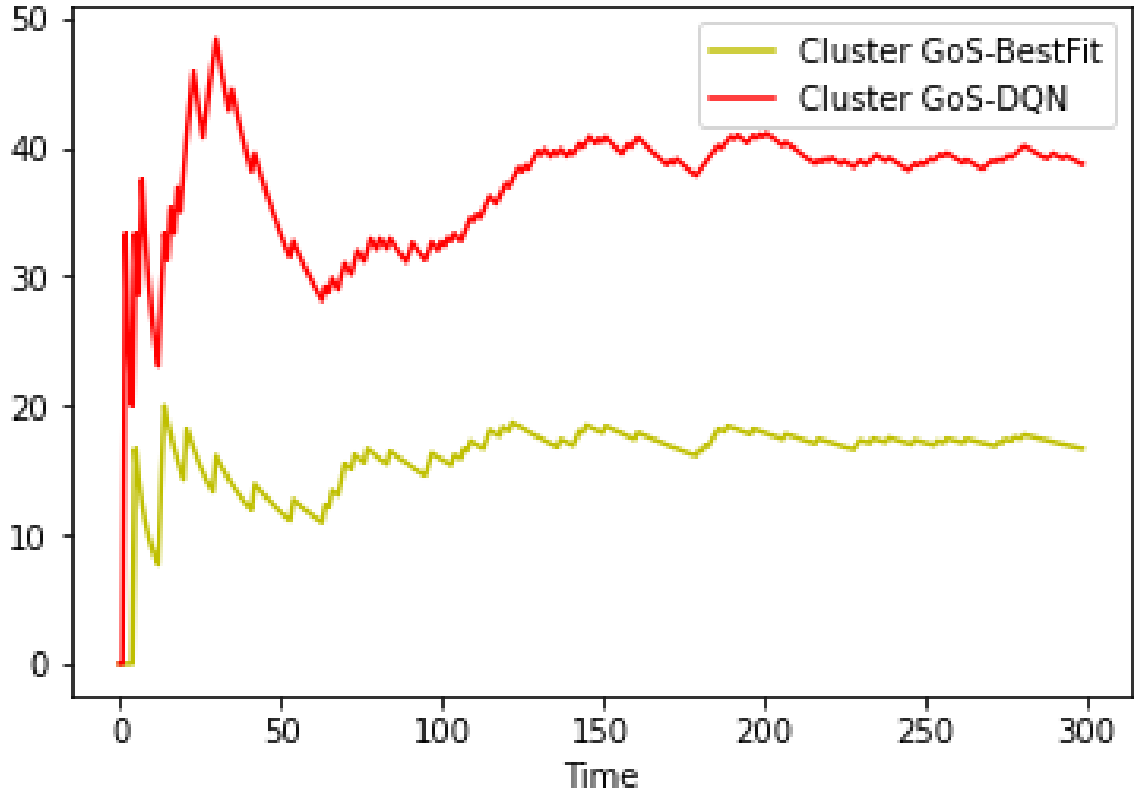

Figure 7. Σύγκριση DQN με την υλοποίηση Best-Fit

Στο Figure 7, παρατηρούμε μια σαφή σύγκριση μεταξύ της προσέγγισης DQN και την τεχνική best-fit. Ο best-fit όσο και ο first-fit παρουσιάζουν παρόμοια αποτελέσματα και υπολείπονται σημαντικά της απόδοσης που επιτυγχάνει ο αλγόριθμος DQN. Αυτή η παρατήρηση ενισχύει την πεποίθησή μας στην καινοτόμο δυναμική του DQN και επιβεβαιώνει την ανωτερότητά του έναντι παραδοσιακότερων και στατικότερων μεθόδων.

Για τον λόγο του πειράματος έχουμε 2 cluster που λαμβάνουν τις ίδιες διεργασίες και έχουν τα ίδια FN και η κάθε συστάδα επεξεργάζεται τα αποτελέσματα με βάση την μέθοδο της. Η διαφορά στα αποτελέσματα υπογραμμίζει την ανάγκη για πιο προηγμένες, μαθησιακές τεχνικές στον τομέα της διαχείρισης των υπολογιστικών πόρων μεταξύ του edge και του cloud, ιδιαίτερα σε περιβάλλοντα που απαιτούν μεγαλύτερη ευελιξία και προσαρμοστικότητα. Επίσης, η βελτιωμένη απόδοση της υλοποίησης μας μπορεί να οδηγήσει σε μεγαλύτερη οικονομία πόρων, μειώνοντας την ανάγκη για πλεονάζοντες υπολογιστικούς πόρους και ενεργειακό κόστος.

4.2.3 Σύγκριση DQN με διαφορετική υλοποίηση Reward

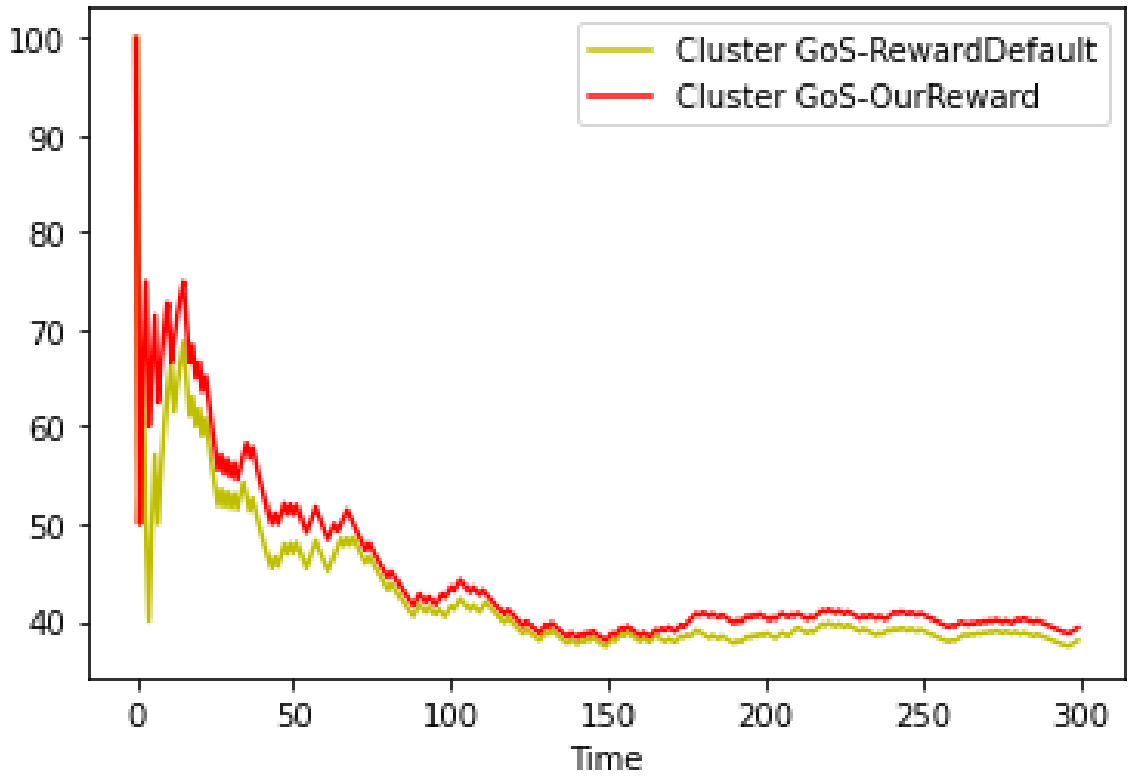

Figure 8. Σύγκριση DQN με διαφορετική υλοποίηση Reward

Στο Figure 8, παρατηρούμε τη σύγκριση του αλγόριθμου DQN με διαφορετικό τρόπο υπολογισμού της ανταμοιβής (reward). Η αρχική μας προσέγγιση για τον υπολογισμό της ανταμοιβής (default reward) ήταν να λαμβάνει την τιμή 1 το reward σε κάθε κλήση της συνάρτησης στην περίπτωση που γινόταν σωστή ανάθεση των υπολογιστικών πόρων και 0 στην αντίθετη περίπτωση. Εμείς προτείνουμε μια διαφορετική προσέγγιση για τον υπολογισμό της ανταμοιβής με βάση την εξίσωση (3), η οποία ορίζει καλύτερα τον τρόπο υπολογισμού της ανταμοιβής σε σχέση με την αρχική προσέγγιση.

Έτσι παρατηρούμε ότι, με τη διαφορά που εφαρμόσαμε στην ανταμοιβή, η δική μας υλοποίηση πετυχαίνει ακόμα καλύτερα αποτελέσματα της τάξεως του 2-3% από τον κλασικό DQN σε επίπεδο διαχείρισης δικτύου.

Πίνακας 1. Συγκριτικός πίνακας με 1 cluster και 3 Fog Nodes

	Cluster Grade of Services	Resources Utilization	Cloud contention
DQN-Our Reward	52%	20%	48%
DQN-Reward Default	49%	18%	51%
Best Fit	20%	19%	80%
First Fit	19%	21%	81%

Ο συγκεκριμένος Πίνακας 1 αναφέρεται στο συγκριτικό πείραμα που εκτελέσαμε όπου αξιολογήθηκαν διάφορες μέθοδοι διαχείρισης υπολογιστικών πόρων για ένα σύστημα με ένα cluster και τρία Fog Nodes.

Συγκεκριμένα, παρουσιάζει στοιχεία για:

1. Δείκτης ποιότητας υπηρεσίας συστοιχίας (Cluster GoS)
2. Δείκτης αξιοποίησης πόρων (Resource Utilization)
3. Δείκτης φόρτου στο cloud (Cloud contention)

Για κάθε μέθοδο, συγκρίνουμε τις επιδόσεις του DQN με την δικιά μας υλοποίηση ανταμοιβής, του βασικού DQN με την απλή μέθοδο υπολογισμού της ανταμοιβής και των μεθόδων best Fit και first Fit. Με αυτόν τον τρόπο παρέχετε μια ανάλυση των αλγορίθμων με τα βασικά στοιχεία σύγκρισης των μεθόδων.

4.3 Συνέπειες και συμβολή

Τα αποτελέσματα του πειράματος έδειξαν ότι το προτεινόμενο μοντέλο κατάφερε να αξιοποιήσει αποτελεσματικά τους περιορισμένους πόρους της άκρης, με τον δείκτη resource utilization να κυμαίνεται γύρω στο 30-50%.

Επίσης, πέτυχε ποσοστό cluster GoS, με περίπου το 20-25% των αιτημάτων να εκτελούνται τοπικά στους κόμβους των edge clusters και το cloud contention στο 75-80% όμως στα αιτήματα υψηλής προτεραιότητας πέτυχε ποσοστό της τάξεως του 86-87%. Με βάση αυτά τα αποτελέσματα, μπορούμε να πούμε πως επιτεύχθηκαν οι στόχοι βελτιστοποίησης των δεικτών ποιότητας υπηρεσίας.

Το προτεινόμενο μοντέλο προσφέρει σημαντικά πλεονεκτήματα σε σχέση με άλλες προσεγγίσεις, όπως οι πιο απλές και στατικές μεθόδους του best-fit και first-fit. Αυτό που το ξεχωρίζει είναι ότι του επιτρέπει να μαθαίνει πολύ πιο γρήγορα και αποτελεσματικά από την εμπειρία, σε σχέση με άλλες τεχνικές πιο «επιφανειακής» ενισχυμένης μάθησης, κάνοντάς τον σε διαρκή διαδικασία βελτιστοποίησης μέσω της επαναλαμβανομένης εκπαίδευσης του.

Επιπλέον, ο διαχωρισμός σε διαφορετικά clusters που συντονίζονται από τους EC εξασφαλίζει μεγαλύτερη ομοιογένεια και αποτελεσματικότητα στην κατανομή των υπολογιστικών πόρων. Ωστόσο, θα μπορούσαν να γίνουν ορισμένες βελτιώσεις που θα έφερναν καλύτερα αποτελέσματα και μεγαλύτερη ευελιξία.

Συγκεκριμένα, ένας τομέας βελτίωσης θα μπορούσε να είναι η εφαρμογή αλγορίθμων βαθιάς μάθησης όπως τα αναδρομικά νευρωνικά δίκτυα (RNN) ή οι μηχανές υποστήριξης διανύσματος (SVMs). Αυτοί οι αλγόριθμοι θα επέτρεπαν στον EC να θυμάται προηγούμενες καταστάσεις και να λαμβάνει υπόψη τη σειρά των γεγονότων, οδηγώντας σε ακόμα καλύτερες αποφάσεις διαχείρισης των πόρων.

Το προτεινόμενο μοντέλο όμως αποτελεί μια καλή αρχή για τη βελτιστοποίηση του δικτυακού κόψιμου και τη βελτίωση της ποιότητας υπηρεσίας στο IoT. Ωστόσο, μελλοντικά θα μπορούσαν να εξεταστούν πιο προηγμένες τεχνικές. Η περαιτέρω έρευνα και ανάπτυξη σε αυτούς τους τομείς θα οδηγήσει σε ακόμα πιο αποτελεσματικές και έξυπνες λύσεις διαχείρισης πόρων στο IoT.

4.4 Περιορισμοί

Στο παρόν υποκεφάλαιο θα αναφερθούν ορισμένοι περιορισμοί που συναντήθηκαν κατά τη διεξαγωγή του πειράματος και της έρευνας γενικότερα. Ένας περιορισμός ήταν ότι το πείραμα διεξήχθη σε ένα σχετικά μικρό, δοκιμαστικού χαρακτήρα περιβάλλον. Η διεξαγωγή του σε μεγαλύτερη κλίμακα και σε πραγματικές συνθήκες πιθανώς να έφερνε στο φως επιπρόσθετους παράγοντες και προβλήματα που θα χρειαζόντουσαν λύση.

Επίσης, η υλοποίηση του μοντέλου έγινε σε εικονικό περιβάλλον, χωρίς να υπάρχει πραγματική δικτύωση και επικοινωνία μεταξύ των διαφόρων συστατικών στοιχείων. Η υλοποίησή του σε πραγματικό δίκτυο θα μπορούσε να φέρει στο φως και άλλους περιορισμούς όπως τα προβλήματα επικοινωνίας των διαφορών συσκευών και καθυστερήσεις στην ανάθεση των υπολογιστικών πόρων θα μπορούσαν να αποβούν μοιραίες.

Ένας ακόμα περιορισμός που θα μπορούσε να προκύψει είναι ότι δεν λαμβάνεται υπόψη η προτεραιότητα των διεργασιών κατά την ανάθεση τους σε edge clusters ή cloud. Συγκεκριμένα, στο προτεινόμενο μοντέλο που περιγράφεται οι διεργασίες χαρακτηρίζονται μόνο από τους

παραμέτρους φόρτου εργασίας (c) και χρόνου εκτέλεσης (h), δεν γίνεται λόγος για την προτεραιότητά τους.

Αυτό σημαίνει ότι ο EC δεν λαμβάνει υπόψη το βαθμό προτεραιότητας μιας διεργασίας δηλαδή θα πρέπει να έχουμε και μια μεταβλητή για το ποσό υψηλή προτεραιότητα έχει μια διεργασία, στο δικό μας μοντέλο βλέπουμε μόνο τον φόρτο εργασίας και χρόνο εκτέλεσης της κάθε διεργασίας. Αυτό θα μπορούσε να θεωρηθεί ως ένας περιορισμός του μοντέλου, αφού καινούργιες διεργασίες με μεγαλύτερη προτεραιότητα μπορεί τελικά να παραπεθνούν στο cloud, γιατί δεν υπάρχουν αρκετοί υπολογιστικοί πόροι στο cluster αφού θα έχουν ανατεθεί σε άλλες διεργασίες .

4.4.1 Περιορισμοί στο πείραμα

Ένας περιορισμός του μοντέλου όσο αναφορά το πείραμα είναι ότι δεν υπάρχει η δυνατότητα μετακίνησης μιας διεργασίας από ένα edge cluster σε άλλο, κατά τη διάρκεια της εκτέλεσής του. Συγκεκριμένα, όταν ο EC αναθέτει μια διεργασία για εκτέλεση σε ένα συγκεκριμένο edge cluster, αυτή η διεργασία παραμένει να εκτελείται εκεί μέχρι να ολοκληρωθεί. Δεν υπάρχει η δυνατότητα, κατά τη διάρκεια εκτέλεσής της, να μεταφερθεί σε κάποιο άλλο edge cluster, ακόμα και αν αυτό θα ήταν πιο αποδοτικό, λόγω της κατάστασης που επικρατεί στο δίκτυο. Αυτός είναι ένας περιορισμός του μοντέλου που θα μπορούσε να βελτιωθεί σε μελλοντικές εργασίες.

Ένας ακόμη περιορισμός του πειράματος είναι ότι οι υπολογιστικοί πόροι των edge clusters θεωρούνται σταθεροί και δεν μπορούν να προσαρμοστούν δυναμικά. Συγκεκριμένα, κατά την έναρξη του πειράματος, οι πόροι κάθε edge cluster ορίζονται σε συγκεκριμένη τιμή (π.χ. 10 πόροι ανά cluster) και αυτή η τιμή παραμένει σταθερή σε όλη τη διάρκεια του πειράματος, ενώ θα μπορούσε να αυξομειώνεται οι πόροι στα FNs ανάλογα κάθε φορά πόσο φόρτο εργασίας υπάρχει, μειώνοντας έτσι και το ενεργειακό κόστος. Τέλος, παρά την καλή απόδοση του μοντέλου σύμφωνα με τα αποτελέσματα του πειράματος, η περαιτέρω βελτιστοποίηση του θα μπορούσε να αναδείξει και άλλους τομείς βελτίωσης.

Κεφάλαιο 5: Συμπεράσματα

Στο πέμπτο και τελευταίο κεφάλαιο της διπλωματικής εργασίας, παρουσιάζονται τα συμπεράσματα και τις προτάσεις που προέκυψαν από τον διεξοδικό αναλυτικό έλεγχο, την εφαρμογή του μοντέλου και το πείραμα που διεξήχθη.

Το κεφάλαιο ολοκληρώνεται με τις προτάσεις για μελλοντική έρευνα. Πώς μπορούν τα ευρήματα αυτής της εργασίας να χρησιμεύσουν ως βήματα για περαιτέρω βελτιώσεις. Ποιες νέες προκλήσεις παρουσιάζονται και πώς μπορούν να αντιμετωπιστούν.

Η εφαρμογή τεχνικών ενισχυμένης μάθησης από τον EC αποτέλεσε έναν από τους παράγοντες που συνέβαλαν καθοριστικά στην αύξηση της αποδοτικότητας του μοντέλου. Συγκεκριμένα, η αξιοποίηση αλγορίθμων βαθιάς ενισχυμένης μάθησης, όπως ο Deep Q-Network, επέτρεψε στον EC να μαθαίνει δυναμικά από την εμπειρία και να προσαρμόζει συνεχώς τις αποφάσεις του, οδηγώντας σε σημαντική βελτίωση των επιδόσεων σε σχέση με άλλες προσεγγίσεις. Η ενισχυμένη μάθηση αποτέλεσε έτσι καταλυτικό στοιχείο για την επιτυχία του προτεινόμενου μοντέλου.

Η εφαρμογή του προτεινόμενου μοντέλου διαχείρισης πόρων σε ένα πραγματικό B5G δίκτυο αποτελεί σημαντική κατεύθυνση για μελλοντική έρευνα και βελτιστοποίηση του μοντέλου όπως αναφέρθηκε. Ένας σημαντικός παράγοντας είναι η αξιολόγησή του σε ένα πραγματικό περιβάλλον που θα επέτρεπε τη μελέτη της συμπεριφοράς και των επιδόσεων του κάτω υπό πραγματικές συνθήκες δικτύου. Κάτι τέτοιο θα εξασφάλιζε τη λήψη υπόψη παραγόντων όπως οι πραγματικές συνθήκες υποδομής ,οι διακυμάνσεις στην ποιότητα σύνδεσης, τα προβλήματα διασύνδεσης, τα οποίοι δεν μπορούν να προσομοιωθούν με ακρίβεια. Επιπλέον, θα διευκόλυνε το πείραμα και θα μπορούσε να πέτυχει ακόμα καλύτερα αποτελέσματα, η συλλογή μεγάλου όγκου δεδομένων για την εκπαίδευση. Συνολικά, η εφαρμογή του και σε άλλα δίκτυα πιο περιπλοκά θα οδηγούσε σε σημαντική βελτίωση και εξέλιξή του.

Επίσης μια σημαντική προοπτική για περαιτέρω δοκιμή και βελτιστοποίηση του προτεινόμενου μοντέλου είναι η εφαρμογή του υπό διαφορετικά σενάρια φόρτου εργασίας. Συγκεκριμένα, η δοκιμή του μοντέλου κάτω από διαφορετικές συνθήκες φόρτου, με διαφορετικούς συνδυασμούς απαιτήσεων εφαρμογών και αιτημάτων υπηρεσιών, θα μπορούσε να φέρει στο φως περιοχές βελτίωσης, εμείς δημιουργήσαμε διάφορες συνθήκες αλλά ποτέ δεν είναι αρκετές.

Επιπλέον, η προσομοίωση έκτακτων συνθηκών φόρτου, όπως αιφνίδιες αυξήσεις ή μειώσεις του φόρτου εργασίας, θα βοηθούσε στη βελτιστοποίηση της ικανότητας του μοντέλου να ανταποκρίνεται σε δυναμικά μεταβαλλόμενες συνθήκες, προσφέροντας ακόμα υψηλότερη απόδοση.

Ένας ακόμα σημαντικός παράγοντας που θα μπορούσε να βελτιώσει την απόδοση του μοντέλου είναι να ληφθεί υπόψη η κινητικότητα των συσκευών και των χρηστών στο δίκτυο. Σε αντίθεση με το υποκείμενο μοντέλο, όπου θεωρήθηκε ότι οι συσκευές και οι χρήστες βρίσκονται σε σταθερή τοποθεσία εντός της περιοχής κάλυψης ενός συγκεκριμένου cluster, στην πραγματικότητα αυτό δεν ισχύει. Καθώς οι χρήστες και οι συσκευές μετακινούνται, ενδέχεται να αλλάζει η γεωγραφική τοποθεσία τους και να χρειάζονται να μεταβαίνουν σε διαφορετικό cluster ώστε να γίνεται πιο γρηγορά η επεξεργασία της διεργασίας τους.

Επομένως, αν ο EC μπορούσε να λαμβάνει υπόψη την κινητικότητα των συσκευών και να ενημερώνει δυναμικά τις αποφάσεις του ανάλογα με την τρέχουσα τοποθεσία τους, θα μπορούσε να βελτιώσει περαιτέρω την αποδοτικότητα και την ποιότητα των υπηρεσιών. Αυτό αποτελεί μια πρόκληση που άξιζε να εξεταστεί σε μελλοντική έρευνα για την εξέλιξη του μοντέλου.

Συμπερασματικά, η παρούσα διπλωματική εργασία απέδειξε ότι η εφαρμογή κατάλληλων μοντέλων διαχείρισης πόρων στην άκρη του δικτύου και η βελτιστοποίηση τους μέσω τεχνικών ενισχυμένης μάθησης είναι κρίσιμης σημασίας, για την υποστήριξη των μελλοντικών έξυπνων συστημάτων και την ικανοποίηση των σύνθετων απαιτήσεων τους σε ποιότητα υπηρεσιών.

Το προτεινόμενο μοντέλο κατάφερε να διαχειριστεί με επιτυχία τους περιορισμένους πόρους στην άκρη του δικτύου, βελτιστοποιώντας τη χρήση τους και εξασφαλίζοντας την ποιοτική υπηρεσιών εξυπηρετώντας τις συσκευές IoT με μεγάλη επιτυχία. Η έρευνα αυτή συνέβαλε στην κατανόηση αυτών των συστημάτων και παρέχει τη βάση για περαιτέρω βελτιώσεις που θα οδηγήσουν σε ένα μελλοντικό έξυπνο και βιώσιμο περιβάλλον.

Παραρτήματα

Στο πλαίσιο της υλοποίησης της διπλωματικής εργασίας, χρησιμοποιήθηκε το λογισμικό Python 3.9, με το περιβάλλον ανάπτυξης Spyder για την ανάλυση και την εκτέλεση των πειραμάτων. Ο υπολογιστής που χρησιμοποιήθηκε εξοπλίζεται με μια κάρτα γραφικών NVIDIA GeForce GTX 1660 Super και έναν επεξεργαστή AMD Ryzen 2600X.

Παράρτημα κώδικα

Υλοποίηση του DQN:

```
class DQNAgent:
    def __init__(self, state_size, action_size):
        # Initialize the agent's attributes
        self.state_size = state_size  # Size of the state space
        self.action_size = action_size  # Size of the action space
        self.memory = []  # List to store experiences for replay
        self.gamma = 0.95  # Discount factor for Q-learning
        self.epsilon = 1.0  # Initial exploration rate
        self.epsilon_min = 0.01  # Minimum exploration rate
        self.epsilon_decay = 0.995  # Decay rate for exploration
        self.learning_rate = 0.001  # Learning rate for neural network training
        self.model = self._build_model()  # Create the Q-network model

    def _build_model(self):
        # Build the neural network model for Q-learning
        model = Sequential()
        # Input layer with 24 neurons and ReLU activation
        model.add(Dense(24, input_dim=self.state_size, activation='relu'))
        # Hidden layer with 24 neurons and ReLU activation
        model.add(Dense(24, activation='relu'))
        # Output layer with linear activation to predict Q-values
        model.add(Dense(self.action_size, activation='linear'))
        # Compile the model with mean squared error loss and Adam optimizer
        model.compile(loss='mse', optimizer=Adam(lr=self.learning_rate))
        return model

    def remember(self, state, action, reward, next_state, done):
        # Store an experience tuple into the replay memory
        self.memory.append((state, action, reward, next_state, done))

    def act(self, state):
        # Choose an action based on epsilon-greedy policy
        if np.random.rand() <= self.epsilon:
            # Explore: choose a random action
            return random.randrange(self.action_size)
        # Exploit: choose the action with maximum predicted Q-value
        act_values = self.model.predict(state.reshape(1, -1))
        return np.argmax(act_values[0])

    def replay(self, batch_size):
        # Train the Q-network using a minibatch of experiences
        minibatch = random.sample(self.memory, batch_size)
        for state, action, reward, next_state, done in minibatch:
            target = reward
            if not done:
                # Compute the target Q-value using Bellman equation
                target = (reward + self.gamma * np.amax(self.model.predict(next_state.reshape(1, -1))[0]))
            # Get current Q-values
            target_f = self.model.predict(state.reshape(1, -1))
            # Update the Q-value for the chosen action
            target_f[0][action] = target
            # Fit the model to the new Q-values
            self.model.fit(state.reshape(1, -1), target_f, epochs=1, verbose=0)
        # Decay the exploration rate
        if self.epsilon > self.epsilon_min:
            self.epsilon *= self.epsilon_decay
```

Υλοποίηση του περιβάλλον:

```python
class Environment:
    def __init__(self, num_nodes_per_cluster, num_resources_per_node):
        # Number of nodes per cluster
        self.num_nodes_per_cluster = num_nodes_per_cluster
        # Number of resources available per node
        self.num_resources_per_node = num_resources_per_node
        # Total observation space considering all resources and processes in all clusters
        self.observation_space = self.num_nodes_per_cluster * (self.num_resources_per_node + 2)
        # Action space is the number of edge clusters as an action is choosing a cluster
        self.action_space = self.num_nodes_per_cluster
        # State of the environment (resources status and processes)
        self.state = None
        # Keep track of the previous resource usage
        self.previous_resource_usage = 0
        # List to keep track of all ongoing processes
        self.processes = []
        # Resources array indicating the usage of each resource in each cluster
        self.resources = np.zeros((self.observation_space,), dtype=bool)
        # Dictionary to keep process names
        self.process_names = {}
        # Store previous states of processes
        self.previous_processes = []

    def reset(self):
        # Reset environment state to the initial state
        self.state = np.zeros(self.observation_space)
        self.processes = []
        self.resources = np.zeros((self.num_nodes_per_cluster, self.num_resources_per_node), dtype=bool)
        self.process_names = {}
        self.previous_processes = []

    def step(self, action, resources_needed, process_name, time):
        # Simulate the environment's reaction to an action

        # Calculate total resources needed
        resources_needed *= time

        # Check if resources can be allocated in the chosen cluster
        if np.sum(self.resources[action]) + resources_needed <= self.num_resources_per_node:
            # Get indices of available resources in the chosen cluster
            available_resources = np.where(~self.resources[action])[0]
            num_resources_needed = min(resources_needed, len(available_resources))
            resources_to_allocate = available_resources[:num_resources_needed]

            # Allocate the resources and save the process details
            for resource in resources_to_allocate:
                self.resources[action][resource] = True
                self.processes.append((action, resource, process_name, resources_needed, time))

        # Update the environment state
        self.state = self._update_state()
        done = False

        return self.state, done

    def _update_state(self):
        # Generate the current state based on resource usage and processes
        state = np.zeros(self.observation_space)
        state[:self.num_nodes_per_cluster * self.num_resources_per_node] = self.resources.flatten()
        for process_cluster, resource_idx, _, _, _ in self.processes:
            state[self.num_nodes_per_cluster * self.num_resources_per_node + process_cluster] = 1

        return state
```

```
def save_previous_processes(self):
    # Store the current state of processes for future reference
    self.previous_processes.append((copy.deepcopy(self.processes), copy.deepcopy(self.process_names)))

def render(self):
    # Print the current state of the environment (for visualization/debugging)
    print("Resources:")
    for cluster_idx in range(self.num_nodes_per_cluster):
        print(f"Cluster {cluster_idx + 1}: ", end="")
        for resource_idx in range(self.num_resources_per_node):
            if self.resources[cluster_idx][resource_idx]:
                print("[1]", end=" ")
            else:
                print("[0]", end=" ")
        print()

    print("Processes:")
    process_counts = {}
    for process_cluster, _, process_name, _, _ in self.processes:
        if process_name not in process_counts:
            process_counts[process_name] = 0
        process_counts[process_name] += 1

    for process_name, count in process_counts.items():
        print(f"(Name: {process_name}) - {count} processes")

def check_process(self):
    # Check every processes and free up resources based on time
    unique_process_names = set()
    for processes, _ in self.previous_processes:
        for _, _, process_name, resources_needed, time in processes:
            resources_needed *= time
            if resources_needed > 0:
                unique_process_names.add(process_name)
    for process_name in unique_process_names:
        resources_used = self.get_resources_used(process_name)
        for i in range(resources_used):
            self.remove_process(process_name)

def remove_process(self, process_name):
    # Remove a process and free its resources
    freed_resources = []
    new_processes = []
    resources_to_free = 1

    for process_cluster, resource_idx, name, resources_needed, time in self.processes:
        if name == process_name and resources_to_free > 0:
            if self.resources[process_cluster][resource_idx]:
                self.resources[process_cluster][resource_idx] = False
                freed_resources.append((process_cluster, resource_idx))
                resources_to_free -= 1
        else:
            new_processes.append((process_cluster, resource_idx, name, resources_needed, time))

    self.processes = new_processes
    self.state = self._update_state()
```

```
def get_resources_used(self, process_name):
    # Get the number of resources used by a particular process
    resources_used = 0
    for process_cluster, _, name, resources_needed, time in self.processes:
        if name == process_name:
            resources_used = int(resources_needed/time)
    return resources_used

def calculate_reward(self, cloud_avoidance, resource_utilization):

    # Weights for each factor
    alpha = 0.6  # Weight for resource utilization
    beta = 0.4   # Weight for cloud avoidance

    # Calculate total reward
    reward = alpha * resource_utilization + beta * cloud_avoidance

    return reward
```

Υλοποίηση της Main:

```
# Number of nodes per cluster
num_nodes_per_cluster = 5
# Number of resources available per node
num_resources_per_node = 10
# Maximum time a resource can be used
max_time_uses = 5
# Total episodes for the simulation
num_times = 300
# Size of the batch to train the agent
batch_size = 32
# Maximum processes per time
max_process_per_time = 10

# Initialize the environment with the specified nodes and resources per node
env = Environment(num_nodes_per_cluster, num_resources_per_node)
# Initialize the DQN agent with the observation and action spaces from the environment
agent = DQNAgent(env.observation_space, env.action_space)
# Get the initial state (resources)
state = env.resources.flatten()
# Reset the environment to its initial state
env.reset()

# Counters to track total requests, served requests and false allocated on cloud
total_requests_received = 0
total_requests_served = 0
allocated_at_cloud = 0

# Lists to store metrics for visualization
resource_utilization_list = []
cloud_contention_list = []
cluster_GoS_list = []
allocated_at_cloud_list = []
```

```
for episode in range(num_times):
    max_process = random.randint(1, max_process_per_time)
    for process in range(max_process):

        # Counter to track resource usage in the current episode
        total_resource_usage = 0
        # Randomly decide the resources needed for the current episode
        resources_needed = random.randint(1, num_resources_per_node)
        # Randomly decide the time duration for which the resources are needed
        time = random.randint(1, max_time_uses)
        # Get the action (node) recommended by the agent for the current state
        action = agent.act(state)
        # Initially, assume resources are not allocated
        allocated = False

        # Increment the total requests counter
        total_requests_received += 1
        # Calculate total resources used across all nodes
        for cluster_idx in range(num_nodes_per_cluster):
            total_resource_usage += np.sum(env.resources[cluster_idx])

        # Check if the resources can be allocated in the chosen node
        if np.sum(env.resources[action]) + (resources_needed * time) <= num_resources_per_node:
            total_requests_served += 1  # Increment the served requests counter
            allocated = True

        # If resources are allocated, update the environment with the allocation
        if allocated:
            process_name = f"Process {random.randint(0,999)}"
            next_state, done = env.step(action, resources_needed, process_name, time)
            # If resources are not allocated, continue with the current state
        else:
            next_state = state
            done = False

        if allocated == False and resources_needed * time < num_resources_per_node:
            allocated_at_cloud += 1

        # Train the agent if enough experiences are collected
        if len(agent.memory) > batch_size:
            agent.replay(batch_size)



    # Calculate and store metrics for the current time
    cluster_GoS = (total_requests_served / total_requests_received) * 100
    cluster_GoS_list.append(cluster_GoS)

    cloud_contention = (1 - (total_requests_served / total_requests_received)) * 100
    cloud_contention_list.append(cloud_contention)

    resource_utilization = total_resource_usage / (num_nodes_per_cluster * num_resources_per_node) * 100
    resource_utilization_list.append(resource_utilization)

    cluster_but_cloud_process = int ((allocated_at_cloud/total_requests_received) * 100)
```

```
# Smooth the resource utilization data for better visualization
smoothed_resource_utilization = smooth_data(resource_utilization_list)

# Plot the metrics
fig, ax = plt.subplots()
ax.plot(smoothed_resource_utilization, color='b', label='Resource Utilization')
ax.plot(cluster_GoS_list, color='g', label='Cluster GoS')
ax.plot(cloud_contention_list, color='r', label='Cloud Contention')
ax.set_xlabel('Time')
ax.legend()

fig, ax = plt.subplots()
ax.plot(allocated_at_cloud_list, color='g', label='Process False Served by Cloud')
ax.set_xlabel('Time')
ax.legend()
```

Βιβλιογραφία

[1] «What is 6G? How is it different from 5G?,» [Ηλεκτρονικό].

[2] a. Y. Y. Almuthanna Nassar, «Deep Reinforcement Learning for Adaptive Network Slicing in 5G for Intelligent Vehicular Systems and Smart Cities,» p. 13, 2020.

[3] Y. Y. Z. S. &. L. W. Deren Li, «From digital Earth to smart Earth,» 2014.

[4] R. V. S. Z. &. L. L. Pan Wang, «Introduction: Advances in IoT research and applications,» 2016.

[5] J. E. Louis Coetzee, «The Internet of Things - promise for the future? An introduction,» 2011.

[6] N. S. G. Deepti Sehrawat, «Smart Sensors: Analysis of Different Types of IoT Sensors,» 2019.

[7] P. Š. D. G.-D. L. P. S Nižetić, «Internet of Things (IoT): Opportunities, issues and challenges towards,» 2019.

[8] X. Z. A. X. L. TAIHUI LI, «An End-to-End Network Slicing Algorithm Based,» 2020.

[9] A. B. O. A. P. Q. M. A. M. H. S Messaoud, «Deep federated Q-learning-based network slicing for industrial IoT,» 2020.

[10] A. B. Y. P. S Dawaliby, «Distributed Network Slicing in Large Scale IoT Based on Coalitional Multi-Game Theory,» 2019.

[11] G. B. M Singh, «Quality of service (qos) in internet of things,» 2018.

[12] M. Y. W. N. Irfan Awan, «Modelling QoS in IoT Applications,» 2014.

[13] M. S.-H. C.-H. L. I. L. Mohammad Aazam, «MeFoRE: QoE based Resource Estimation at Fog to Enhance QoS in IoT,» 2016.

[14] M. S. M. U. H. D. M. Z. z. a. R. A. Qaisar Shaheen, «A Lightweight Location-Aware Fog Framework (LAFF) for QoS in Internet of Things Paradigm,» 2020.

[15] X. C. X. Ren Duan, «A QoS Architecture for IOT,» 2011.

[16] J. M. S. T. Melrose Roderick, «Implementing the Deep Q-Network,» 2015.

[17] N. B. N. S. O Anschel, «Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning,» 2017.

[18] A. G. D. S. H Van Hasselt, «Deep Reinforcement Learning with Double Q-Learning,» 2016.

[19] M. D. P. P. A. S. J. L. D. N. O. D. H. P. DC Nguyen, «6G Internet of Things: A Comprehensive Survey,» 2021.

[20] S. Y. M. P. H Xiang, «A realization of fog-RAN slicing via deep reinforcement learning,» 2020.

[21] K. S. G. I. A. Z. LM Ang, «Deployment of IoV for smart cities: Applications, architecture, and challenges,» 2018.

[22] Y. X. H. S. R. B. Y Sun, «Internet of Things Services for Small Towns,» 2014.

[23] S. S. P. M. S Painuly, «Future Trends and Challenges in Next Generation Smart Application of 5G-IoT,» 2021.

[24] P. Pirinen, «A Brief Overview of 5G Research Activities,» 2014.

[25] M. H. A Čolaković, «Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues,» 2018.

[26] Y. S. X. L. Z. M. C. W. M Peng, «Recent Advances in Cloud Radio Access Networks: System Architectures, Key Techniques, and Open Issues,» 2016.

[27] A. S. S. H. U. J. M. G. Emiliano Sisinni, «Industrial Internet of Things: Challenges, Opportunities, and Directions,» 2018.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

EC: Edge Controller

FN: Fog Node

IoT: Internet of Things

QoS: Quality of Service

MDP: Markov Decision Process

DQN: Deep Q-Network

DRL: Deep Reinforcement Learning

F-RAN: Fog Radio Access Network

C-RAN: Cloud Radio Access Network

BBU: Baseband Unit

RRU: Remote Radio Unit

NFV: Network Function Virtualization

SDN: Software Defined Networking

URLLC: Ultra-Reliable Low Latency Communications

VANET: Vehicular Ad hoc Networks

AI: Artificial Intelligence

CNN: Convolutional Neural Network

LSTM: Long Short-Term Memory

RNN: Recurrent Neural Network

SVM: Support Vector Machine

DNN: Deep Neural Network

Απόδοση ξενόγλωσσων όρων

IoT - Διαδίκτυο των Πραγμάτων (Internet of Things)

QoS - Ποιότητα Υπηρεσίας (Quality of Service)

MDP - Διαδικασία Λήψης Αποφάσεων με Μάρκοφ (Markov Decision Process)

DQN - Βαθύ Δίκτυο Q (Deep Q-Network)

DRL - Ενισχυμένη Μάθηση με Βάθος (Deep Reinforcement Learning)

F-RAN - Δίκτυο Πρόσβασης Ραδιοσυχνοτήτων με Ομίχλη (Fog Radio Access Network)

C-RAN - Συγκεντρωτικό Δίκτυο Πρόσβασης Ραδιοσυχνοτήτων (Cloud Radio Access Network)

BBU - Μονάδα Βασικού Ζώνης (Baseband Unit)

RRU - Μονάδα Μακρινής Ραδιοσυχνότητας (Remote Radio Unit)

VM - Εικονική Μηχανή (Virtual Machine)

NFV - Εικονικοποίηση Δικτυακών Λειτουργιών (Network Function Virtualization)

SDN - Λογισμικού Οριζόμενο Δίκτυο (Software Defined Networking)

URLLC - Υψηλής Αξιοπιστίας Χαμηλής Καθυστέρησης Επικοινωνίες (Ultra-Reliable Low Latency Communications)

VANET - Ad Hoc Δίκτυα Οχημάτων (Vehicular Ad hoc Networks)

AI - Τεχνητή Νοημοσύνη (Artificial Intelligence)

CNN - Συνεργατικά Νευρωνικά Δίκτυα (Convolutional Neural Network)

LSTM - Μακροχρόνια Μνήμη Σύντομης Διάρκειας (Long Short-Term Memory)

RNN - Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Network)

SVM - Διανύσματα Υποστήριξης (Support Vector Machine)

DNN - Βαθιά Νευρωνικά Δίκτυα (Deep Neural Network)