ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Dynamic and Energy-Efficient Management of Programmable Network Infrastructure Utilizing Reinforcement Learning Techniques

DIPLOMA THESIS
by
Georgia Bousmpoukea

Επιβλέπων : Συμεών Παπαβασιλείου
Καθηγητής ΕΜΠ

Αθήνα, Ιούνιος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Dynamic and Energy-Efficient Management of Programmable Network Infrastructure Utilizing Reinforcement Learning Techniques

DIPLOMA THESIS
by
Georgia Bousmpoukea

Επιβλέπων : Συμεών Παπαβασιλείου
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Ιουνίου 2025.

.....	.....	.....
Συμεών Παπαβασιλείου	Ελένη Στάη	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Επ. Καθηγήτρια Ε.Μ.Π.	Επ. Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2025

(Υπογραφή)

.....

Μπουσμπουκέα Γεωργία
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο Διαχωρισμός Δικτύου αποτελεί μία σημαντική τεχνολογία στις σύγχρονες τηλεπικοινωνίες, ιδιαίτερα στο πλαίσιο των δικτύων 5G και πέρα, καθώς επιτρέπει την αποσύνθεση του Δικτύου Πρόσβασης Ραδιοσυχνοτήτων (RAN) σε πολλαπλές εικονικοποιημένες υπομονάδες, καθεμία με διαφορετικές και ετερογενείς απαιτήσεις. Αυτή η προσέγγιση, ειδικά υπό την αρχιτεκτονική του κατανεμημένου O-RAN (Open Radio Access Network), εξασφαλίζει την αποδοτική λειτουργία διάφορων υπηρεσιών παράλληλα. Η εικονικοποίηση του δικτύου επιτρέπει την ενσωμάτωση προηγμένων μηχανισμών βελτιστοποίησης για τη διαχείριση των πόρων, τη μεγιστοποίηση της απόδοσης και τον έλεγχο της κατανάλωσης ενέργειας.

Στο πλαίσιο αυτό, η παρούσα διπλωματική εργασία προτείνει μια λύση βασισμένη σε τεχνικές Ενισχυτικής Μάθησης. Ο πράκτορας που αναπτύχθηκε είναι υπεύθυνος για τον δυναμικό έλεγχο αποδοχής και τοποθέτησης αιτημάτων δικτύου και για τη βέλτιστη διάσπαση των Εικονικοποιημένων Λειτουργιών Δικτύου (Virtual Network Functions - VNFs) κάθε αιτήματος μεταξύ στις μονάδες του O-RAN. Το προτεινόμενο μοντέλο εκπαιδεύεται να λαμβάνει αποφάσεις με στόχο την ελαχιστοποίηση της ενεργειακής κατανάλωσης και τη βελτιστοποίηση της χρήσης πόρων, προσαρμοζόμενο σε μεταβαλλόμενα μοτίβα δημιουργίας αιτημάτων. Τα πειραματικά αποτελέσματα καταδεικνύουν τη δυνατότητα του πράκτορα να λαμβάνει λογικές και βέλτιστες αποφάσεις, προσαρμοσμένες στις δεδομένες συνθήκες του περιβάλλοντος. Παράλληλα, η σημασία της δυναμικής προσέγγισης επιβεβαιώνεται μέσω σύγκρισης με στατικό πράκτορα, ο οποίος υστερεί στις αξιολογούμενες μετρικές απόδοσης.

Λέξεις Κλειδιά - Διαχωρισμός Δικτύου, Δίκτυο ORAN, Ενισχυτική Μάθηση, Εικονικοποιημένη Συνάρτηση Δικτύου, Διάσπαση Λειτουργιών

Abstract

Network Splitting is a critical concept in modern telecommunications, particularly within the context of 5G and beyond, where it enables the decomposition of the Radio Access Network (RAN) into multiple virtualized subunits, each with distinct and heterogeneous requirements. This approach, especially under the disaggregated O-RAN (Open Radio Access Network) architecture, ensures the efficient operation of diverse services in parallel. The virtualization of the network allows for dynamic splitting of the functions of a slice between the network nodes, enabling the integration of advanced optimization mechanisms to manage resources effectively, maximize performance, and control power consumption.

In this context, the present thesis proposes a solution based on Reinforcement Learning techniques. The developed agent is responsible for the dynamic admission control and placement of network slices, as well as the optimal splitting of Virtual Network Functions (VNFs) of each slice across the nodes of the O-RAN architecture. The proposed model is trained to make decisions with the aim of minimizing energy consumption and optimizing resource utilization, adapting to varying patterns of slice request generation. Experimental results demonstrate the agent's ability to make rational and optimal decisions, tailored to the given environmental conditions. Additionally, the importance of the dynamic approach is confirmed through comparison with a static agent, which underperforms in the evaluated performance metrics.

Keywords— Network Splitting, O-RAN, Reinforcement Learning, Virtual Network Function, Function Splitting

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Συμεών Παπαβασιλείου, για την ευκαιρία που μου έδωσε να εκπονήσω την διπλωματική μου εργασία στο εργαστήριο NETMODE (Network Management & Optimal Design Laboratory) καθώς και για την εμπιστοσύνη που μου έδειξε για να ασχοληθώ με αυτό το ιδιαίτερα ενδιαφέρον θέμα. Ακόμη, ευχαριστώ θερμά την κα. Ελένη Στάη, που με καθοδήγησε καθ' όλη τη διάρκεια της διπλωματικής μου και αποτέλεσε για μένα ακαδημαϊκό και προσωπικό πρότυπο. Θα ήθελα επιπλέον να ευχαριστήσω τον Νικόλα Φρυγανιώτη και τον Γεώργιο Γκοτζιά για την στενή συνεργασία μας και την πολύτιμη βοήθειά τους.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένειά μου, που με στήριξε καθ' όλη τη διάρκεια των σπουδών μου και χωρίς εκείνη δεν θα είχα καταφέρει να φτάσω ως εδώ. Ευχαριστώ επίσης τους φίλους μου, με τους οποίους εξελιχθήκαμε μαζί, στηρίξαμε ο ένας τον άλλον και περάσαμε πολλές στιγμές που, πέρα από την παρέα και την ψυχαγωγία, έπαιξαν σημαντικό ρόλο στη διαμόρφωση της προσωπικότητάς μου.

Γεωργία Μπουσμπουκέα, Ιούνιος 2025

Contents

List of Figures

List of Tables

Chapter 1

Εκτεταμένη Ελληνική Περίληψη

1.1 Εισαγωγή

Η εξέλιξη των κινητών δικτύων από το 5G στο 6G, σε συνδυασμό με τις αυξανόμενες απαιτήσεις για υψηλότερους ρυθμούς δεδομένων, αυξημένη χωρητικότητα και υποστήριξη πολυμέσων και υπηρεσιών επαυξημένης πραγματικότητας, απαιτεί την ανάπτυξη προηγμένων μηχανισμών για αποτελεσματική διαχείριση της κίνησης του δικτύου και ελαχιστοποίηση της κατανάλωσης ενέργειας. Η τεχνολογία του Τεμαχισμού Δικτύου (Network Slicing) αποτελεί βασικό εργαλείο, που χωρίζει το δίκτυο σε πολλαπλά εικονικά υποδίκτυα, γνωστά ως «slices», το καθένα προσαρμοσμένο ώστε να καλύπτει τις διαφορετικές ανάγκες συγκεκριμένων εφαρμογών, παρόχων υπηρεσιών και χρηστών. Κάθε slice λειτουργεί ως ξεχωριστό, απομονωμένο εικονικό δίκτυο, με τους δικούς του αφιερωμένους πόρους, πολιτικές και χαρακτηριστικά απόδοσης. Αυτή η διαίρεση του δικτύου είναι ουσιαστική για την ορθολογική κατανομή των πόρων και την αποτελεσματική λειτουργία διαφορετικών υπηρεσιών παράλληλα. Στην παρούσα διπλωματική, τα slices μοντελοποιούνται με την προσέγγιση Αλυσίδας Λειτουργιών (Service Function Chain- SFC), αναπαριστώντας τα ως μια διατεταγμένη ακολουθία Εικονικών Λειτουργιών Δικτύου (Virtual Network Functions- VNFs). Τα VNFs είναι δικτυακές λειτουργίες βασισμένες σε λογισμικό που εκτελούνται σε εικονικοποιημένο υλικό αντί για αποκλειστικές φυσικές συσκευές. Το μοντέλο Αλυσίδας Λειτουργιών καθορίζει τη συγκεκριμένη σειρά με την οποία αυτά τα VNFs εκτελούνται εντός ενός slice, ώστε να εκπληρώνονται οι απαιτήσεις της υπηρεσίας.

Μία σημαντική τεχνική στη διαχείριση των slices είναι η Διάσπαση Λειτουργιών (Function Splitting), η οποία περιλαμβάνει τη κατανομή των VNFs ενός slice σε πολλούς κόμβους του δικτύου. Η Διάσπαση Λειτουργιών επιτρέπει να εκτελούνται μέρη της Αλυσίδας Λειτουργιών κοντά στον χρήστη για τη μείωση της καθυστέρησης, ενώ άλλα μέρη μπορούν να κεντρικοποιηθούν για τη βελτιστοποίηση κατανάλωσης υπολογιστικών πόρων και ενέργειας.

Η διπλωματική αυτή εστιάζει στον τομέα του RAN, συγκεκριμένα στο πλαίσιο της αρχιτεκτονικής O-RAN, με στόχο την ενίσχυση της ανοιχτότητας και διαλειτουργικότητας. Όπως εισάγεται από την O-RAN Alliance, το RAN διαχωρίζεται σε λογικούς κόμβους, ανάλογα με τη θέση τους στο κυψελοειδές δίκτυο, συγκεκριμένα Radio Unit (RU), Distributed Unit (DU) και Centralized Unit (CU) [1]. Οι DUs συνήθως βρίσκονται στο άκρο του δικτύου (Edge), σε κοντινή απόσταση από τα RUs, για τη μείωση της καθυστέρησης των χρηστών.

Αντίθετα, οι CUs βρίσκονται στον πυρήνα του δικτύου και χαρακτηρίζονται από μεγαλύτερη υπολογιστική ισχύ σε σχέση με τους DUs. Για την ικανοποίηση των slices εντός της αρχιτεκτονικής O-RAN και την εκτέλεση της Διάσπασης Λειτουργιών, η Αλυσίδα Λειτουργιών που αντιστοιχεί σε κάθε slice πρέπει ουσιαστικά να διαιρεθεί ανάμεσα στους τρεις αυτούς κόμβους, με τρόπο που να βελτιστοποιεί την ανταλλαγή μεταξύ κατανάλωσης ισχύος και καθυστέρησης, λαμβάνοντας υπόψη και το ποσοστό αποδοχής slices.

Στην παρούσα διπλωματική, το πρόβλημα του Τεμαχισμού Δικτύου και Διάσπασης Λειτουργιών προσεγγίζεται μέσω Ενισχυτικής Μάθησης, προσφέροντας μια δυναμική και προσαρμοστική λύση για την αποδοχή slices και την τοποθέτηση VNFs. Ο πράκτορας εκπαιδεύεται με τον αλγόριθμο Προσεγγιστικής Βελτιστοποίησης Πολιτικής (Proximal Policy Optimization - PPO), με στόχο τη διαχείριση της ισορροπίας μεταξύ των εσόδων από την αποδοχή slices και του κόστους κατανάλωσης ισχύος που σχετίζεται με τα edge clouds και τους δικτυακούς συνδέσμους. Μέσω μιας σειράς πειραμάτων, αξιολογείται η απόδοση του πράκτορα με διάφορες μετρικές. Εξετάζονται διεξοδικά η επίγνωση κατάστασης του πράκτορα, οι δυνατότητες γενίκευσής του, καθώς και η σημασία της δυναμικής λήψης αποφάσεων, προσφέροντας χρήσιμες γνώσεις για την αποτελεσματικότητα των λύσεων Ενισχυτικής Μάθησης σε σύνθετες εργασίες διαχείρισης δικτύων.

1.2 Βασικές Αρχές RAN και ORAN Αρχιτεκτονικών

Το Δίκτυο Πρόσβασης Ραδιοσυχνοτήτων (Radio Access Network - RAN) αποτελεί βασικό στοιχείο ενός ασύρματου δικτύου, καθώς επιτελεί τη σύνδεση μεταξύ του εξοπλισμού του χρήστη με το ευρύτερο δίκτυο μέσω ραδιοσύνδεσης. Καθ' όλη την εξέλιξή του από δίκτυα πρώτης (1G) μέχρι και πέμπτης (5G) γενιάς, αυτή η σύνδεση παραμένει ο κύριος σκοπός του RAN, με την αρχιτεκτονική του να προσαρμόζεται στις αυξανόμενες απαιτήσεις υψηλής χωρητικότητας, μαζικής συνδεσιμότητας, χαμηλού κόστους και εξοικονόμησης ισχύος, ώστε να παρέχει υπηρεσίες χαμηλής καθυστέρησης και υψηλής αξιοπιστίας.

Παρά την πρόοδο στην τυποποίηση των δικτυακών πρωτοκόλλων και στη διασφάλιση διαλειτουργικότητας, το RAN παραμένει το τελευταίο ιδιόκτητο τμήμα του δικτύου. Συγκεκριμένοι προμηθευτές, δηλαδή, κυριαρχούν έναντι ανοικτών προτύπων. Αυτή η ιδιόκτητη φύση του RAN απασχολεί τη βιομηχανία, ιδιαίτερα καθώς νέες αρχιτεκτονικές, όπως το Open RAN (O-RAN) στοχεύουν να καταστήσουν αυτό το τμήμα πιο προσιτό και διαλειτουργικό, φέρνοντας επανάσταση στον τρόπο σχεδιασμού, ανάπτυξης και διαχείρισης των κινητών δικτύων [2].

1.2.1 Συστατικά του RAN

Σε όλα τα στάδια εξέλιξης του RAN, τα φυσικά συστατικά του κατηγοριοποιούνται ανάλογα με τον ρόλο τους στις ακόλουθες κατηγορίες:

• Εξοπλισμός Χρήστη: Συσκευές με μόντεμ ευρυζωνικής πρόσβασης, ικανά να μεταδίδουν και να λαμβάνουν ασύρματα σήματα (δεδομένα, φωνή, σήματα ελέγχου).

• Κεραίες: στοιχεία του φυσικού επιπέδου του RAN, συνήθως τοποθετημένες στον ή κοντά στον σταθμό βάσης. Λειτουργούν ως η πρώτη διεπαφή μεταξύ του εξοπλισμού τελικού χρήστη

και του RAN, μεταδίδοντας και λαμβάνοντας ηλεκτρομαγνητικά σήματα στον αέρα.

• Σταθμοί Βάσης: Οι σταθμοί βάσης επεξεργάζονται τα ακατέργαστα σήματα που συλλέγονται από τις κεραίες και τα μετατρέπουν σε μορφή που μπορεί να μεταδοθεί μέσω του δικτύου.

• Δίκτυο Μεταφοράς: Το Δίκτυο Μεταφοράς είναι υπεύθυνο για τη μεταφορά δεδομένων και σημάτων ελέγχου μεταξύ των διαφόρων στοιχείων του δικτύου. Συνδέει το RAN, το Βασικό Δίκτυο (Core Network) και άλλα κρίσιμα στοιχεία του δικτύου, εξασφαλίζοντας αποτελεσματική και αξιόπιστη μετάδοση δεδομένων.

1.2.2 Εξέλιξη του RAN

Η συνεχής εξέλιξη των τεχνολογιών κινητής επικοινωνίας αντικατοπτρίζεται και στο RAN. Κάθε διαδοχική γενιά, από 1G μέχρι και το 5G, εισάγει καινοτόμα αρχιτεκτονικά πρότυπα, σημαντικές βελτιώσεις στο υλικό και βελτιωμένο λογισμικό, βελτιώνοντας την συνολική απόδοση, η οποία υπολογίζεται από μετρικές όπως η αυξημένη ταχύτητα μετάδοσης δεδομένων, η ενισχυμένη αξιοπιστία, η βελτιωμένη επεκτασιμότητα και η μεγαλύτερη ενεργειακή αποδοτικότητα. Οι ραγδαίες αυτές αλλαγές στις τεχνολογίες του RAN καθοδηγούνται από τις αυξανόμενες απαιτήσεις σε χωρητικότητα, οι οποίες προκύπτουν από την εκθετική αύξηση στην κίνηση δεδομένων τις τελευταίες δεκαετίες.

Συγκεκριμένα, από τις αναλογικές τεχνολογίες του 1G, τα δίκτυα 2G σηματοδοτούν την μετάβαση στις ψηφιακές ασύρματες επικοινωνίες. Η αρχιτεκτονική του RAN στα δίκτυα 2G υιοθέτησε την προσέγγιση του Κατανεμημένου RAN (DRAN), όπου οι Σταθμοί Βάσης (BSs) κατανέμονται σε όλη την περιοχή κάλυψης, με κάθε σταθμό να εξυπηρετεί ένα συγκεκριμένο κελί του κυψελωτού δικτύου και να διαχειρίζεται τις δικές του λειτουργίες επεξεργασίας και ελέγχου. Αυτός ο σταθμός βάσης διαιρείται στους Σταθμούς- Πομπού Βάσης (Base Transceiver Station- BTS), που διαχειρίζονταν τα ασύρματα κανάλια επικοινωνίας και το σύστημα κεραιών, και στον Ελεγκτή Σταθμού Βάσης (Base Station Controller- BSC), που έλεγχε πολλαπλούς BTSs και συνδεόταν με το core network. Καθώς τα δίκτυα 2G ήταν κυρίως φωνοκεντρικά, το δίκτυο μεταφοράς ήταν circuit switching, απαιτώντας μια αποκλειστική φυσική σύνδεση για όλη τη διάρκεια μιας συνεδρίας επικοινωνίας [3].

Η μετάβαση στην τεχνολογία 3G εισήγαγε σημαντικές αρχιτεκτονικές αλλαγές, με γνώμονα την αυξανόμενη ανάγκη για υποστήριξη υπηρεσιών πολυμέσων. Ο NodeB, ο οποίος αντικατέστησε τον BTS, ήταν υπεύθυνος για την παροχή της ραδιοδιεπαφής που επέτρεπε στο εξοπλισμό χρήστη να συνδεθεί με το δίκτυο. Ωστόσο, σε αντίθεση με την αρχιτεκτονική 2G, ένας εξωτερικός κεντρικός Ελεγκτής Ραδιοδικτύου (RNC) ανέλαβε τις περισσότερες λειτουργίες ελέγχου και διαχείρισης υψηλότερου επιπέδου, όπως η διαχείριση ραδιο-πόρων και των μεταβιβάσεων μεταξύ κεραίων. Αυτός ο ελεγκτής λειτουργούσε ως ενδιάμεσος, ανάμεσα στους πολλαπλούς NodeBs και το core network, εξελίσσοντας την λογική του RAN από την κατανεμημένη της μορφή. Συγχρόνως, η κυριαρχία των δεδομένων έναντι της φωνής, οδήγησε σε μια υβριδική αρχιτεκτονική δικτύου μεταφοράς, που συνδυάζει circuit-switched τεχνολογία για τη φωνή και packet-switched τεχνολογία για τα δεδομένα, επιτρέποντας πιο αποδοτική χρήση των πόρων του δικτύου [4].

Η μετάβαση στο 4G/LTE (Long-Term Evolution) σηματοδότησε μια σημαντική πρόοδο στην αρχιτεκτονική των κινητών δικτύων, ώστε να ανταποκριθεί στην εκθετική αύξηση της

κίνησης δεδομένων, που δεν μπορούσε να καλύψει το παραδοσιακό Κατανεμημένο RAN με την τοπική, κυτταρική προσέγγιση του. Μια εξελιγμένη εκδοχή του Σταθμού Βάσης, ο eNodeB, ενσωμάτωσε τις λειτουργίες του παραδοσιακού NodeB και του RNC σε μια ενιαία, ολοκληρωμένη οντότητα, καθιστώντας την αρχιτεκτονική πιο αποδοτική. Ο eNodeB χωρίζεται σε δύο βασικά χαρακτηριστικά: (α) Απομακρυσμένες Κεφαλές/Μονάδες Ραδιοεπικοινωνίας (RRHs/RRUs), που αναλαμβάνουν λειτουργίες όπως η ενίσχυση, το φιλτράρισμα, η μετάδοση και η λήψη σημάτων προς/από τις κεραίες και (β) Μονάδες Βασικής Ζώνης (BBUs), υπεύθυνες για λειτουργίες όπως διαμόρφωση/ αποδιαμόρφωση και κωδικοποίηση/αποκωδικοποίηση, καθώς ακόμη λειτουργούν ως διεπαφή μεταξύ core network και των RRUs [4]. Επιπρόσθετα, στα δίκτυα 4G/LTE, το δίκτυο μεταφοράς μεταβλήθηκε πλήρως σε μοντέλο μετάδοσης μέσω πακέτων, ευθυγραμμισμένο με τη δεδομενοκεντρική φύση του δικτύου.

Η αυξανόμενη κυκλοφορία των δεδομένων, οδήγησε σε μια εξελιγμένη αρχιτεκτονική, το Κεντρικοποιημένο RAN (Centralised RAN- CRAN), το οποίο συγκέντρωσε τα BBUs σε μια ενιαία τοποθεσία (π.χ. data center), που ονομάζεται BBU Hotel, για την εξυπηρέτηση πολλών κεραιών μιας συγκεκριμένης περιοχής, στοχεύοντας στη μείωση του κόστους εγκατάστασης και συντήρησης. Τα BBU Hotel επικοινωνούν με τα RRHs μέσω συνδέσεων fronthaul και μπορούν να διαμοιραστούν μεταξύ πολλών τοποθεσιών. Αυτό το μοντέλο επιτρέπει την επεξεργασία μεγαλύτερου όγκου κυκλοφορίας, ενισχύοντας την ασφάλεια του δικτύου και διευκολύνοντας την επέκταση όποτε το απαιτεί η ζήτηση [4].

Συνολικά, ενώ η αρχιτεκτονική CRAN και οι τεχνολογίες 4G ενίσχυσαν την αποδοτικότητα του δικτύου, το υποκείμενο υλικό και λογισμικό παρέμειναν σε μεγάλο βαθμό ιδιόκτητα ανά προμηθευτή. Πολλοί προμηθευτές χρησιμοποιούσαν ιδιόκτητες διεπαφές μεταξύ των BBUs και των RRHs, συνδυασμένες με προσαρμοσμένες υλοποιήσεις των πρωτοκόλλων front-haul. Ως αποτέλεσμα, το υλικό συχνά έπρεπε να παρέχεται από έναν και μόνο προμηθευτή για να διασφαλιστεί η συμβατότητα και η απρόσκοπτη ενσωμάτωση, περιορίζοντας έτσι την ευελιξία και τη διαλειτουργικότητα της υποδομής του δικτύου.

Η μεταγενέστερη εξάπλωση του 5G έφερε σημαντική πρόοδο στην αρχιτεκτονική του RAN, προκειμένου να ανταποκριθεί στις απαιτήσεις για υπερυψηλές ταχύτητες δεδομένων, χαμηλή καθυστέρηση και μαζική συνδεσιμότητα συσκευών, με τεχνολογίες όπως Massive MIMO και Beamforming. Ο σταθμός βάσης, γνωστός ως gNodeB, διαχωρίζει τα συστατικά του λογικά και φυσικά σε μια πιο αποσυγκεντρωμένη αρχιτεκτονική. Ειδικότερα, αποτελείται από τις εξής επιμέρους μονάδες:

• Ραδιο-Μονάδα (Radio Unit- RU): Αποτελεί το τμήμα του δικτύου που διαχειρίζεται τις λειτουργίες φυσικού επιπέδου, επικεντρωμένο στη μετάδοση και λήψη ραδιοσημάτων. Είναι κομμάτι υλικού, εγκατεστημένο κοντά στην κεραία ή ενσωματωμένο σε αυτήν.

• Κατανεμημένη Μονάδα (Distributed Unit- DU): Αποτελεί συνδυασμό λογισμικού και φυσικών τεχνολογιών και είναι υπεύθυνη για λειτουργίες πραγματικού χρόνου με πολύ χαμηλές απαιτήσεις καθυστέρησης. Τοποθετούνται στην "άκρη" του δικτύου (edge data centers), κοντά στις Ραδιο-Μονάδες.

• Κεντρικοποιημένη Μονάδα (Centralised Unit- CU): Είναι μονάδα λογισμικού που φιλοξενείται σε κάποια πλατφόρμα στο νέφος και αναλμβάνει λειτουργίες μη πραγματικού χρόνου. Αυτή η κεντρικοποίηση επιτρέπει την δυναμική κατανομή πόρων μεταξύ πολλαπλών DUs.

Αυτός ο διαχωρισμός απαιτεί φυσικά ένα διαφορετικό μοντέλο δικτύου μεταφοράς: το Fronthaul (FH) συνδέει την RU με τη DU, το Midhaul τη DU με τη CU και το Backhaul τη CU με το Core Network.

Η συνολική διανομή των λειτουργιών ανάμεσα στις τρεις μονάδες της RAN—CU, DU και RU—συμβάλλει στη μείωση των απαιτήσεων σε εύρος ζώνης και καθυστέρηση για κάθε μονάδα. Επιπλέον, επιτυγχάνεται καλύτερη αξιοποίηση των πόρων και πιο αποτελεσματική διαχείριση σε όλο το δίκτυο, ενισχύοντας την αποδοτικότητα και την επεκτασιμότητα της αρχιτεκτονικής. Συγκεκριμένα, οι λειτουργίες της BBU στο 4G μοιράστηκαν μεταξύ της DU και της CU, δύο ξεχωριστές και γεωγραφικά διαχωρισμένες μονάδες. Έτσι, λειτουργίες διαφορετικών επιπέδων πρωτοκόλλων, και, άρα, με διαφορετικές χρονικές απαιτήσεις, εκτελούνται σε διαφορετικές μονάδες. Συγχρόνως, στις DUs μεταφέρονται περίπλοκες λειτουργίες επεξεργασίας, μειώνοντας το κόστος των RUs και επιτρέποντας τον κεντρικό συντονισμό πολλών RUs από μία DU. Αυτό επιτρέπει την καλύτερη εκμετάλλευση της χωρικής διαθεσιμότητας, τον περιορισμό των παρεμβολών μεταξύ συστημάτων και τη δυναμική επιλογή της καταλληλότερης κεραίας για την καλύτερη εξυπηρέτηση του χρήστη, ενισχύοντας την κάλυψη και την αποδοτικότητα.

Στο 5G, η αρχιτεκτονική μεταβαίνει στο Cloud RAN, όπου η επεξεργασία της βασικής ζώνης γίνεται με κεντρικοποιημένο τρόπο από λογισμικό σε κέντρα δεδομένων στο νέφος. Αυτή η προσέγγιση διευκολύνει την επεκτασιμότητα και την δυναμική κατανομή πόρων, επιτρέποντας στο δίκτυο να προσαρμόζεται στις μεταβαλλόμενες απαιτήσεις κίνησης. Ωστόσο, αν και η διεπαφή μεταξύ των RUs και των DUs ή CUs στα υπάρχοντα RAN συστήματα βασίζεται σε μερικώς τυποποιημένες προδιαγραφές, η πρακτική εφαρμογή αυτών των διεπαφών συχνά περιλαμβάνει ιδιόκτητες παραλλαγές, με αποτέλεσμα να περιορίζεται η διαλειτουργικότητα μεταξύ διαφορετικών προμηθευτών. Επιπλέον, το λογισμικό που έχει σχεδιαστεί για τον συγκεκριμένο εξοπλισμό είναι συνήθως ασύμβατο με υλικό από άλλους κατασκευαστές. Αυτή η εξάρτηση από ιδιόκτητες, ειδικά διαμορφωμένες υλοποιήσεις των διεπαφών δημιουργεί ένα σημαντικό εμπόδιο στη διαλειτουργικότητα και ενισχύει το φαινόμενο του vendor lock-in.

1.2.3 Βασικές Τεχνολογίες

1.2.3.1 Εικονικοποίηση Λειτουργιών Δικτύου

Η Εικονικοποίηση Λειτουργιών Δικτύου (Network Function Virtualization- NFV) αντικαθιστά τις παραδοσιακές δικτυακές λειτουργίες που εκτελούνται σε εξειδικευμένο υλικό, όπως firewalls, load balancers και routers, με λογισμικό που μπορεί να εκτελείται σε COTS servers σε cloud πλατφόρμες. Αυτή η μετάβαση επιτρέπει την ευέλικτη και δυναμική διαχείριση των υπολογιστικών, αποθηκευτικών και δικτυακών πόρων, χωρίς την ανάγκη αναβάθμισης του φυσικού εξοπλισμού. Οι εικονικές δικτυακές λειτουργίες (VNFs) μπορούν να μετακινηθούν ή να αναπτυχθούν δυναμικά σε διάφορα σημεία του δικτύου, ανάλογα με τις ανάγκες των υπηρεσιών, συμβάλλοντας στην καλύτερη αξιοποίηση των πόρων και τη βελτιστοποίηση της απόδοσης. Τα σχετικά VNFs συνδέονται, σχηματίζοντας αλυσίδες λειτουργιών υπηρεσιών (Service Function Chains - SFC), οι οποίες επιτρέπουν την εφαρμογή συγκεκριμένων πολιτικών διαχείρισης σε διαφορετικές ροές δεδομένων.

1.2.3.2 Λειτουργικός Διαχωρισμός

Η στοίβα πρωτοκόλλων του RAN στο 5G περιλαμβάνει πολλά επίπεδα που διαχειρίζονται τη μετάδοση και την επεξεργασία δεδομένων από το φυσικό επίπεδο έως τη σηματοδοσία ελέγχου. Το φυσικό επίπεδο (PHY) αναλαμβάνει τη ραδιοεπικοινωνία, ενώ το MAC και το RLC διαχειρίζονται τον καταμερισμό πόρων και τη μεταφορά δεδομένων αντίστοιχα. Το PDCP προσφέρει λειτουργίες όπως συμπίεση επικεφαλίδων και κρυπτογράφηση, ενώ το RRC ελέγχει τη διαχείριση συνδέσεων. Το SDAP εξασφαλίζει την ποιότητα υπηρεσίας (QoS). Αυτά τα επίπεδα οργανώνονται σε τρεις γενικές κατηγορίες: το φυσικό επίπεδο (L1), το επίπεδο ζεύξης δεδομένων (L2) και το επίπεδο δικτύου (L3).

Καθώς η κίνηση στα 5G RAN αυξάνεται, οι παραδοσιακές διεπαφές fronthaul αντιμετωπίζουν σημαντικές προκλήσεις χωρητικότητας. Μια λύση είναι η μεταφορά περισσότερων λειτουργιών επεξεργασίας στις RUs, αλλά παραμένουν ερωτήματα σχετικά με το ποιες λειτουργίες και κατά πόσο πρέπει να εκτελούνται τοπικά. Ο 3GPP έχει καθορίσει διάφορους λειτουργικούς διαχωρισμούς (functional splits) μεταξύ RU, DU και CU, οι οποίοι καθορίζουν πόση επεξεργασία πραγματοποιείται τοπικά στη RU και πόση μεταφέρεται στη DU, επηρεάζοντας άμεσα τις απαιτήσεις του fronthaul. Οι κύριες επιλογές περιλαμβάνουν το Option 8, όπου η DU εκτελεί όλες τις λειτουργίες βασικής ζώνης και η RU διατηρεί μόνο RF λειτουργίες, και τα Options 7.1 - 7.3, όπου όλο και περισσότερες λειτουργίες μεταφέρονται στη RU για μείωση των απαιτήσεων εύρους ζώνης στο fronthaul. Η αύξηση των λειτουργιών επεξεργασίας στη RU μειώνει το φόρτο του fronthaul και τη χρονική καθυστέρηση, αλλά αυξάνει την πολυπλοκότητα και το κόστος της RU. Η επιλογή του βέλτιστου λειτουργικού διαχωρισμού εξαρτάται από τις απαιτήσεις QoS, την υποδομή μεταφοράς δεδομένων και την πυκνότητα χρηστών στην περιοχή εξυπηρέτησης [5].

Στις σύγχρονες RAN αρχιτεκτονικές, χρησιμοποιείται δυναμικός διαχωρισμός, όπου η κατανομή των λειτουργιών προσαρμόζεται σε πραγματικό χρόνο, βελτιστοποιώντας την απόδοση του δικτύου με βάση το φόρτο, την καθυστέρηση και τους διαθέσιμους πόρους.

Τα βασικά πλεονεκτήματα του NFV περιλαμβάνουν τη δυναμική κατανομή πόρων, την ευέλικτη διαχείριση, την ταχύτερη κλιμάκωση υπηρεσιών, τη μείωση κόστους και τη μείωση της ενεργειακής κατανάλωσης για τους Χειριστές δικτύων κινητής τηλεφωνίας [6]. Ωστόσο, η εικονικοποίηση των λειτουργιών εισάγει νέες προκλήσεις στη διαχείριση και την ασφάλεια, καθώς απαιτεί προηγμένες λύσεις για την προστασία των VNFs από πιθανά κενά ασφαλείας και για την εξασφάλιση της αποδοτικής λειτουργίας του δικτύου. Παρόλα αυτά, το NFV αποτελεί μία καινοτόμο και αποδοτική προσέγγιση στη διαχείριση σύγχρονων τηλεπικοινωνιακών δικτύων, προσφέροντας ευελιξία, επεκτασιμότητα και οικονομική αποδοτικότητα [7].

1.2.3.3 Δικτυακός Τεμαχισμός

Ο Δικτυακός Τεμαχισμός (Network Slicing) αφορά τη δημιουργία πολλαπλών εικονικών υποδικτύων, αναφερόμενων ως τεμάχια, πάνω σε μια κοινή φυσική υποδομή, επιτρέποντας την εξατομικευμένη κατανομή δικτυακών πόρων σε διαφορετικές εφαρμογές. Κάθε τεμάχιο λειτουργεί ως ανεξάρτητο εικονικό δίκτυο, με δικούς του πόρους, πολιτικές και χαρακτηριστικά απόδοσης, όπως χαμηλή καθυστέρηση, υψηλό εύρος ζώνης ή μαζική συνδεσιμότητα, προσαρμοσμένο στις απαιτήσεις της εκάστοτε υπηρεσίας του. Αυτή η εικονικοποίηση είναι δυνατή μέσω της ενσωμάτωσης τεχνικών από το Software Defined Networking (SDN) και το NFV. Ανάλογα με τις απαιτήσεις της κάθε εφαρμογής, δημιουργούνται συγκεκριμένα

τεμάχια, εξυπηρετώντας ένα φάσμα περιπτώσεων χρήσης, όπως εφαρμογές υψηλού εύρους ζώνης (eMBB slice), επικοινωνίες με εξαιρετικά χαμηλή καθυστέρηση (URLLC slice), διαχείριση IoT συσκευών (mMTC slice).

Ο Δικτυακός Τεμαχισμός προσφέρει μεγάλη προσαρμοστικότητα, επιτρέποντας την δυναμική ανακατανομή των πόρων σε πραγματικό χρόνο, αυξάνοντας την αποδοτικότητα και την κλιμάκωση του δικτύου. Διασφαλίζει, έτσι, ότι οι τηλεπικοινωνιακοί πάροχοι μπορούν να υποστηρίζουν ένα ευρύ φάσμα εφαρμογών, από streaming βίντεο μέχρι αυτόνομα οχήματα και IoT, με βέλτιστη απόδοση και αποτελεσματική διαχείριση των δικτυακών πόρων. Ως αποτέλεσμα, έχει εξελιχθεί σε θεμελιώδη τεχνολογία των σύγχρονων 5G δικτύων [8].

1.2.4 Εικονικοποιημένο RAN

Το Εικονικοποιημένο RAN (Virtualized RAN- vRAN) εφαρμόζει τις αρχές του NFV στο RAN, εικονικοποιώντας και συγκεντρώνοντας συστατικά επεξεργασίας, όπως η Μονάδα Βασικής Ζώνης (BBU), οι gNB/eNodeB, η DU και η CU, αποδεσμεύοντάς τα από εξειδικευμένο υλικό. Αυτή η κεντρικοποιημένη αρχιτεκτονική επιτρέπει στους παρόχους να αξιοποιούν τον Δικτυακό Τεμαχισμό για δυναμική κατανομή πόρων και ευέλικτη προσαρμογή του δικτύου στις απαιτήσεις των εφαρμογών. Η ανάγκη για χειροκίνητη διαχείριση του δικτύου εξαλείφεται, καθώς ο έλεγχος των VNFs γίνεται με κεντρικό τρόπο μέσω εργαλείων ενορχήστρωσης. Επομένως, το vRAN δίνει τη δυνατότητα στους παρόχους να προσαρμόζουν τον Λειτουργικό Διαχωρισμό του δικτύου με βάση τη χωρητικότητα του fronthaul και τις ανάγκες QoS των εκάστοτε εφαρμογών, επιτρέποντας τη βέλτιστη υποστήριξη διαφορετικών υπηρεσιών[5, 9].

Παρά τα οφέλη της ευελιξίας και της κλιμάκωσης, το vRAN εξακολουθεί να αντιμετωπίζει προκλήσεις, κυρίως λόγω της εξάρτησής του από ιδιόκτητες διεπαφές μεταξύ των radio και core δικτυακών στοιχείων, γεγονός που περιορίζει τη διαλειτουργικότητα και εγκλωβίζει τους παρόχους σε συγκεκριμένους προμηθευτές RAN.

Table 1.1: Different Radio Access Network Generations.

Architecture	Baseband Hardware	Baseband Software	Radio Hardware (RRU)	BBU/RRU Interface	Interability
CRAN (Centralized)	Proprietary Technology	Proprietary Software	Proprietary Hardware	Proprietary Interface	Radio + BBU (HW+SW single vendor)
V-RAN	COTS	Proprietary Software	Proprietary Hardware	Proprietary Interface	Radio + BBU (SW single vendor)
O-RAN	COTS	Software Open Interface	COTS	Open Interface	Radio + BBU (HW+SW various vendors)

1.2.5 ORAN

Το Open Radio Access Network (O-RAN) είναι μια πρωτοβουλία της βιομηχανίας τηλεπικοινωνιακών παρόχων, που στοχεύει στην αναδιαμόρφωση των παραδοσιακών RAN αρχιτεκτονικών, προωθώντας την ανοικτότητα, τη διαλειτουργικότητα και την ευελιξία. Βασίζεται στις αρχές της εικονικοποίησης, της τυποποίησης και της χρήσης ανοικτών προτύπων, επιτρέποντας στους παρόχους να αξιοποιούν εξοπλισμό και λογισμικό από πολλαπλούς προμηθευτές. Σε αντίθεση με τις παραδοσιακές μονολιθικές αρχιτεκτονικές RAN, όπου το υλικό και το λογισμικό είναι στενά συνδεδεμένα και παρέχονται από έναν μόνο κατασκευαστή, το O-RAN διαχωρίζει τα στοιχεία του δικτύου και τα συνδέει μέσω ανοικτών διεπαφών. Αυτό μειώνει την εξάρτηση των παρόχων από συγκεκριμένους προμηθευτές (vendor lock-in) και ενισχύει τον ανταγωνισμό και την καινοτομία στη βιομηχανία των τηλεπικοινωνιών. Συγχρόνως, δίνεται έμφαση στην ενσωμάτωση AI και ML στη RAN αρχιτεκτονική, καθώς το δίκτυο γίνεται πιο πυκνό και απαιτεί υποστήριξη εφαρμογών υψηλής έντασης δεδομένων, οι παραδοσιακές χειροκίνητες μέθοδοι διαχείρισης δεν επαρκούν. Το O-RAN προωθεί αυτοδιαχειριζόμενα, αυτοθεραπευόμενα και αυτοδιαμορφούμενα δίκτυα, μειώνοντας το κόστος υποδομής (το RAN αντιστοιχεί σε 65%-70% του συνολικού CapEx) και βελτιώνοντας την εμπειρία των χρηστών.

Παρά τις προσπάθειες των παρόχων κινητής τηλεφωνίας να προωθήσουν τη διαλειτουργικότητα στο RAN, ξεκινώντας από το 2000, η αγορά παρέμεινε στα χέρια λίγων μεγάλων κατασκευαστών, ενισχύοντας το vendor lock-in και περιορίζοντας τον ανταγωνισμό. Η κυριαρχία του LTE επιδείνωσε αυτή τη συγκέντρωση, αναγκάζοντας τους παρόχους να αναζητήσουν νέες προσεγγίσεις που θα τους προσέφεραν μεγαλύτερη ελευθερία και ευελιξία. Ο Συνασπισμός O-RAN (O-RAN Alliance), που ιδρύθηκε το 2018 από πέντε κορυφαίους παρόχους παγκοσμίως, αποτέλεσε την πρώτη επιτυχημένη πρωτοβουλία για την ανάπτυξη ενός ανοιχτού και διαλειτουργικού RAN, βασισμένου σε τυποποιημένες διεπαφές και white-box υλικό [2]. Η O-RAN Alliance έχει πλέον καθορίσει τεχνικές προδιαγραφές που ορίζουν ανοικτές διεπαφές για τη σύνδεση των στοιχείων του O-RAN, προωθώντας τη διαλειτουργικότητα. Δηλαδή. ένας Near-RT RIC από έναν κατασκευαστή μπορεί να συνεργάζεται με σταθμούς βάσης από διαφορετικό κατασκευαστή, ενώ οι CUs, DUs και RUs μπορούν να λειτουργούν μεταξύ τους ανεξάρτητα από τον προμηθευτή τους.

Με χαμηλότερο κόστος, λιγότερα εμπόδια εισόδου στην αγορά και αυξημένη αποδοτικότητα, η επιτυχής υιοθέτηση του O-RAN δεν μετασχηματίζει μόνο το RAN, αλλά και ολόκληρη τη βιομηχανία τηλεπικοινωνιών, επιταχύνοντας τις τεχνολογικές εξελίξεις και βελτιώνοντας τις υπηρεσίες για τους παρόχους και τους τελικούς χρήστες.

1.2.5.1 Αρχιτεκτονική του ORAN

• O-Cloud
Το O-Cloud είναι η βασική υποδομή νέφους του O-RAN, παρέχοντας το υπολογιστικό περιβάλλον για τη φιλοξενία εικονικοποιημένων και containerized δικτυακών λειτουργιών. Βασίζεται σε COTS servers, υποστηρίζει ανοιχτές και διαλειτουργικές διεπαφές, και επιτρέπει στους παρόχους να υλοποιούν vRAN λειτουργίες με μεγαλύτερη επεκτασιμότητα και ευελιξία [2].

• Service Management and Orchestration

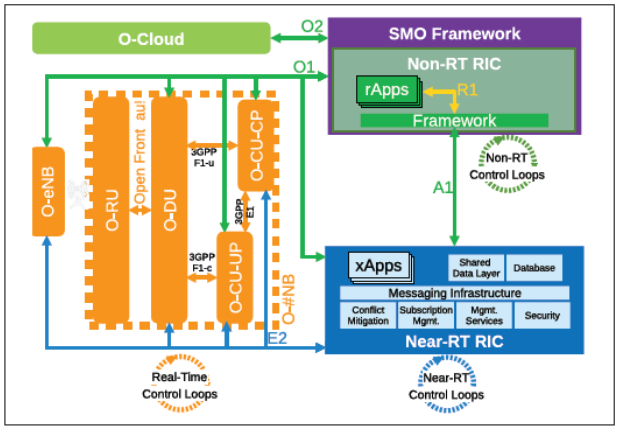


Figure 1.1: Αρχιτεκτονική του O-RAN [1].

Το Service Management and Orchestration (SMO) πλαίσιο αποτελεί το κεντρικό σύστημα διαχείρισης και ορχήστρωσης στο O-RAN, επιβλέποντας τη λειτουργία, ασφάλεια και απόδοση του δικτύου. Χρησιμοποιώντας την O2 διεπαφή, επιτρέπει τη δυναμική διαχείριση των O-Cloud πόρων και, μέσω της O1 διεπαφής, επιβλέπει και παρακολουθεί την απόδοση όλων των συστατικών της αρχιτεκτονικής, εκτός από την O-RU. Με αυτή τη δομή, το SMO βελτιώνει την αποδοτικότητα, την ευελιξία και τον έλεγχο του O-RAN, εξασφαλίζοντας συνεχή προσαρμογή στις απαιτήσεις του δικτύου [2].

• Non-RT RAN Intelligent Controller (Non-RT RIC)
Το Non-RT RIC είναι ενσωματωμένο στο SMO. Ο κύριος στόχος του είναι να υποστηρίξει τη βελτιστοποίηση του RAN μέσω έξυπνης καθοδήγησης και πολιτικών διαχείρισης. Συλλέγει δεδομένα από το δίκτυο, όπως μετρικές απόδοσης και πρότυπα κυκλοφορίας, και τα αναλύει χρησιμοποιώντας προηγμένους αλγορίθμους, συμπεριλαμβανομένων μοντέλων μηχανικής μάθησης. Στη συνέχεια, δημιουργεί στρατηγικές και πληροφορίες βελτιστοποίησης, τις οποίες αποστέλλει στον Near-RT RIC για εκτέλεση σε πραγματικό χρόνο. Οι πάροχοι κινητής τηλεφωνίας έχουν τη δυνατότητα να αναπτύσσουν και να διαχειρίζονται αλγορίθμους για το Non-RT RIC, προσαρμόζοντας τη λειτουργία του RAN σύμφωνα με τις πολιτικές και τους στόχους τους. Οι διαδικασίες διαχείρισης δεδομένων διοχετεύνται στις O1/O2 διεπαφές για εκτέλεση, ενώ η καθοδήγηση μέσω πολιτικής και οι πληροφορίες εμπλουτισμού αποστέλλονται στον Near-RT RIC μέσω της A1 διεπαφής. Η A1 διεπαφή συγχρόνως επιτρέπει και βασική ανατροφοδότηση από το Near-RT RIC στο Non-RT RIC, μέσω αναφορών μετρήσεων λειτουργίας και αποτελεσμάτων εφαρμοσμένων πολιτικών. [2, 10].

• Near-RT RAN Intelligent Controller (Near-RT RIC)
Το Near-RT RAN Intelligent Controller (Near-RT RIC) είναι μια λογική λειτουργία που επιτρέπει τη βελτιστοποίηση, τον έλεγχο και την παρακολούθηση των O-CU και O-DU σε σχεδόν πραγματικό χρόνο. Λειτουργεί βάσει πολιτικών και κανόνων υψηλού επιπέδου που παράγονται από το Non-RT RIC και μεταφράζονται σε άμεσα εφαρμόσιμες εντολές για τους E2 κόμβους μέσω της E2 διεπαφής [11]. Το Near-RT RIC είναι υπεύθυνο λειτουργίες, όπως ισορροπία φορτίου ανά χρήστη, διαχείριση ραδιοδικτυακών πόρων, ανίχνευση και μείωση παρεμβολών, διαχείριση QoS, διαχείριση συνδεσιμότητας και έλεγχο ομαλής μεταπομπής. Για τη λήψη αποφάσεων, χρησιμοποιεί μια βάση δεδομένων (R-NIB), η οποία καταγράφει την

κατάσταση του δικτύου σε σχεδόν πραγματικό χρόνο. Τα δεδομένα αυτά χρησιμοποιούνται για εκπαίδευση AI/ML μοντέλων στο Non-RT RIC [10].

• O-RU

Η O-RU είναι το στοιχείο του O-RAN που συνδέεται με τις κεραίες και διαχειρίζεται τη μετάδοση και λήψη ραδιοσημάτων, εκτελώντας φιλτράρισμα, ενίσχυση και beamforming πριν στείλει τα δεδομένα στην O-DU μέσω του fronthaul [10].

• O-DU

Η O-DU αναλαμβάνει την επεξεργασία των δεδομένων και της σηματοδότησης στο επίπεδο πάνω από το φυσικό, και συνεργάζεται στενά με την O-CU για την υλοποίηση ελέγχου και εργασιών χρήστη [10, 12]. Επικοινωνεί με την O-RU μέσω της Open Fronthaul διεπαφής. Η O-RAN Alliance έχει επιλέξει τη διαμόρφωση "7-2x" ως το βέλτιστο σημείο ισορροπίας, μεταξύ απλοποίησης της O-RU και μείωση της συμφόρησης στο fronthaul δίκτυο .

• O-CU

Η O-CU χωρίζεται σε O-CU-CP (Control Plane) και O-CU-UP (User Plane), διαχωρίζοντας τις λειτουργίες σηματοδοσίας από την επεξεργασία δεδομένων χρηστών. Η O-CU-CP διαχειρίζεται τη συνδεσιμότητα, τη διαχείριση κινητικότητας και τη ρύθμιση των ραδιοπόρων, ενώ ασχολείται και με τη συμπίεση επικεφαλίδων και την κρυπτογράφηση των σηματοδοτικών μηνυμάτων. Από την άλλη, η O-CU-UP αναλαμβάνει τη δρομολόγηση των δεδομένων μεταξύ του UE και του Core Network, εξασφαλίζοντας τη σωστή μετάδοση των πακέτων και την εφαρμογή των QoS πολιτικών. Ο διαχωρισμός της O-CU προσφέρει μεγαλύτερη ευελιξία και επεκτασιμότητα, καθώς το user plane γίνεται πιο τυποποιημένο και οικονομικό στη διαχείριση, ενώ το control plane παρέχει καλύτερο έλεγχο και διαχείριση των ραδιοδικτυακών πόρων. Μέσω τεχνολογιών μηχανικής μάθησης, η O-CU βελτιστοποιεί τη λειτουργία του δικτύου.

Οι O-CU και O-DU ανήκουν στους E2 κόμβους, οι οποίοι συνδέονται με το Near-RT RIC μέσω της E2 διεπαφής, επιτρέποντας δυναμική διαχείριση και προσαρμογή των ραδιοπόρων του RAN σε πραγματικό χρόνο [11].

Table 1.2: Αντιστοίχιση Συστατικών με Διεπαφές.

Components	**Interface**
SMO and E2 nodes	O1 Interface
Non-RT RIC and near-RT RIC	A1 Interface
Near-RT RIC and E2 nodes	E2 Interface
Non-RT RIC and rApps	R1 Interface
O-CU-CP and O-CU-UP	E1 Interface
O-CU and O-DU	F1 Interface
O-DU and O-RU	Open FrontHaul
SMO and Cloud Platform (O-Cloud)	O2 Interface

1.2.5.2 Υπηρεσίες Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης

Η ενσωμάτωση AI/ML μοντέλων στο O-RAN είναι θεμελιώδες στοιχείο της αρχιτεκτονικής του, επιτρέποντας αυτοματοποιημένο έλεγχο πόρων, ανίχνευση ανωμαλιών και ταξινόμηση κίνησης. Το ML training host είναι το δίκτυο που δημιουργεί και εκπαιδεύει AI/ML μοντέλα εκτός σύνδεσης, ενώ το ML inference host εκτελεί τα μοντέλα και μπορεί να πραγματοποιεί online εκπαίδευση. Τα εκπαιδευμένα μοντέλα χρησιμοποιούνται για αποφάσεις δικτύου, επηρεάζοντας τη διαχείριση διαμόρφωσης μέσω της O1 διεπαφής, τη διαχείριση πολιτικών μέσω της A1 και τον έλεγχο O-CU/O-DU/O-RU μέσω της E2 διεπαφής [13, 10].

Υπάρχουν τρεις βασικές στρατηγικές ανάπτυξης AI/ML στο O-RAN:

- Το Non-RT RIC αναλαμβάνει και τους δύο ρόλους training και inference host. Σε αυτή τη διαμόρφωση, ολόκληρη η διαδικασία μηχανικής μάθησης, που περιλαμβάνει τη δημιουργία μοντέλων, διαχείριση κύκλου ζωής και την παροχή δεδομένων, πραγματοποιείται εντός του SMO. Οι ενέργειες που εκτελούνται σε αυτή τη διαμόρφωση περιλαμβάνουν:
  - Πολιτική για το Near-RT RIC, η οποία μεταφέρεται μέσω της A1 διεπαφής.
  - Διαμόρφωση παραμέτρων για τα O-CU/O-DU/O-RU, η οποία εφαρμόζεται μέσω της O1 διεπαφής.
- Το Non-RT RIC λειτουργεί ως ML training host, ενώ το Near-RT RIC αναλαμβάνει το inference host. Αυτή η προσέγγιση χρησιμοποιεί τις διεπαφές O1 και O2 για τη δημιουργία και συντήρηση μοντέλων ML. Οι ενέργειες περιλαμβάνουν:
  - Πληροφορίες Πρόβλεψης για τις εσωτερικές διαδικασίες του Near-RT RIC, με την A1 διεπαφή να διευκολύνει την ανταλλαγή δεδομένων μεταξύ Non-RT και Near-RT RICs.
  - Διαμόρφωση παραμέτρων O-CU/O-DU/O-RU, χρησιμοποιώντας την E2 διεπαφή για συλλογή δεδομένων και επιβολή πολιτικών.
- Το Non-RT RIC λειτουργεί ως training host, ενώ το inference host μεταφέρεται απευθείας στο O-CU ή το O-DU. Αυτή η μέθοδος επεκτείνει τη κατανεμημένη ευφυΐα (distributed intelligence) στα στοιχεία του RAN, εκμεταλλευόμενη την εγγύτητα στις πηγές δεδομένων για άμεση λήψη αποφάσεων και έλεγχο σε πραγματικό χρόνο [10].

1.2.5.3 Open-v-RAN

Το Open RAN συνεπάγεται τον διαχωρισμό του υλικού από το λογισμικό, επιτρέποντας τη διαλειτουργικότητα μεταξύ προμηθευτών, αλλά δεν σημαίνει απαραίτητα ότι το hardware ή το λογισμικό είναι ανοιχτού κώδικα. Οι διεπαφές μεταξύ των διαφορετικών στοιχείων του δικτύου είναι ανοιχτές, αλλά το υλικό μπορεί να παραμείνει ιδιόκτητο, αρκεί να συμμορφώνεται με τις Open RAN προδιαγραφές. Παράλληλα, υπάρχει ευελιξία στην ανάπτυξη των RU, DU και CU, καθώς ορισμένες μπορεί να περιλαμβάνουν ενσωματωμένες δικτυακές λειτουργίες στο υλικό, ενώ άλλες μπορεί να έχουν διαχωρισμένο λογισμικό και υλικό.

Το Open vRAN συνδυάζει τις έννοιες της εικονικοποίησης και του Open RAN. Αναφέρεται στη χρήση τεχνολογιών NFV για την ανάπτυξη εικονικοποιημένων Open RAN αρχιτεκτονικών, όπου οι O-DU και O-CU υλοποιούνται ως λογισμικό (software-based network functions) [9, 14].

1.3 Ενισχυτική Μάθηση

Η Ενισχυτική Μάθηση είναι ένας τομέας της Μηχανικής Μάθησης που επικεντρώνεται στη μάθηση με στόχο, μέσω αλληλεπιδράσεων με ένα περιβάλλον. Βασικό στοιχείο αποτελεί ένας πράκτορας, ο οποίος παίρνει αποφάσεις και μαθαίνει να επιτυγχάνει έναν συγκεκριμένο στόχο αλληλεπιδρώντας με το περιβάλλον, λαμβάνοντας ανατροφοδότηση με τη μορφή ανταμοιβών και βελτιώνοντας τη συμπεριφορά του με στόχο την μεγιστοποίηση των αθροιστικών ανταμοιβών με την πάροδο του χρόνου. Αποκτάει, δηλαδή, γνώσεις και δεξιότητες μέσω άμεσης εμπειρίας, αντίθετα από άλλες προσεγγίσεις μηχανικής μάθησης, όπου υπάρχει διαθέσιμο κάποιο σύνολο δεδομένων.

1.3.1 Συστατικά Ενισχυτικής Μάθησης

Εκτός από τον πράκτορα, ένα σύστημα ενισχυτικής μάθησης έχει τρία υποστοιχεία: ένα περιβάλλον και, ενδεχομένως, μοντέλο για αυτό, μία πολιτική και μία συνάρτηση αξίας.

- **Περιβάλλον**

Το περιβάλλον λειτουργεί ως το εξωτερικό σύστημα, με το οποίο ο πράκτορας αλληλεπιδρά, παρέχοντας ανατροφοδότηση στις ενέργειες του πράκτορα. Αυτή η ανατροφοδότηση επιτρέπει στον πράκτορα να βελτιώσει σταδιακά τη συμπεριφορά του προκειμένου να επιτύχει έναν καθορισμένο στόχο. Περιλαμβάνει τον Χώρο Ενεργειών (Action Space- A), που αντιπροσωπεύει το σύνολο όλων των πιθανών ενεργειών που μπορεί να εκτελέσει ο πράκτορας, τον Χώρο Καταστάσεων (State Space- S), που περιλαμβάνει όλες τις πιθανές διαμορφώσεις του περιβάλλοντος που μπορεί να αντιληφθεί ο πράκτορας και ένα Σήμα Ανταμοιβής (Reward Signal), το οποίο καθορίζει τον στόχο ποσοτικοποιώντας την καταλληλότητα μιας συγκεκριμένης κατάστασης ή ενέργειας. Σε κάθε χρονική στιγμή, το περιβάλλον παρέχει μια ανταμοιβή στον πράκτορα με βάση την πιο πρόσφατη ενέργειά του και την αντίστοιχη κατάσταση που προέκυψε. Ο στόχος του πράκτορα είναι να μεγιστοποιήσει την αθροιστική ανταμοιβή, γνωστή ως απόδοση, με την πάροδο του χρόνου. Η αθροιστική ανταμοιβή ορίζεται ως κάποια συνάρτηση της ακολουθίας των ανταμοιβών. Στην πιο απλή περίπτωση, είναι το άθροισμα των επιμέρους ανταμοιβών:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T, \tag{1.1}$$

όπου το T είναι το τελικό βήμα σε επεισοδιακά προβλήματα, που αντιστοιχεί σε μια τερματική κατάσταση. Η εκπτωτική αθροιστική ανταμοιβή ορίζεται ως:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T, \tag{1.2}$$

όπου $0 \leq \gamma \leq 1$ λέγεται παράγοντας έκπτωσης και καθορίζει κατά πόσο θα έχουν αξία οι μελλοντικές ανταμοιβές για αυτό το βήμα.

Εναλλακτικά, για συμπερίληψη των συνεχών προβλημάτων, ο τύπος γίνεται:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}, \quad \text{συμπεριλαμβανομένης της πιθανότητας } T = \infty \text{ or } \gamma = 1. \quad (1.3)$$

Σε πολλά προβλήματα ενισχυτικής μάθησης είναι διαθέσιμο ή μπορεί να κατασκευαστεί ένα μοντέλο του περιβάλλοντος. Ένα τέτοιο μοντέλο επιτρέπει στον πράκτορα να κάνει υποθέσεις για τη δυναμική του περιβάλλοντος, προβλέποντας πώς το περιβάλλον θα αντιδράσει σε ένα δεδομένο ζεύγος κατάστασης-ενέργειας. Το μοντέλο ορίζεται επίσημα με τη χρήση της συνάρτησης μετάβασης κατάστασης $p(s'|s, a)$, που καθορίζει την πιθανότητα μετάβασης σε μια επόμενη κατάσταση s' δεδομένης της τρέχουσας κατάστασης s και της ενέργειας a, και της συνάρτησης ανταμοιβής $R(s, a, s')$, που προσδιορίζει την ανταμοιβή κατά αυτήν τη μετάβαση. Αυτή η προσέγγιση, που είναι γνωστή ως μέθοδος βασισμένη στο μοντέλο, εξαρτάται από την ρητή αναπαράσταση της συμπεριφοράς του περιβάλλοντος για την καθοδήγηση της διαδικασίας λήψης αποφάσεων. Αντίθετα, οι μέθοδοι χωρίς μοντέλο παρακάμπτουν την ανάγκη για ένα τέτοιο μοντέλο και βασίζονται αποκλειστικά στην εκμάθηση μέσω δοκιμής και λάθους και συχνά υπερέχουν σε σενάρια όπου η κατασκευή ενός αρκετά ακριβούς μοντέλου είναι μη πρακτική ή υπολογιστικά ακριβή.

Συνολικά, ο ρόλος του περιβάλλοντος είναι καθοριστικός, καθώς δεν παρέχει μόνο το πλαίσιο εντός του οποίου δρα ο πράκτορας, αλλά καθορίζει επίσης την πολυπλοκότητα εκμάθησης. Παράγοντες όπως η στοχαστικότητα των μεταβάσεων καταστάσεων και η διάσταση των χώρων κατάστασης και ενέργειας επηρεάζουν σημαντικά την επιλογή αλγορίθμων και τη γενική απόδοση του πράκτορα.

• Πολιτική

Η πολιτική καθορίζει τη στρατηγική του πράκτορα για την επιλογή ενεργειών σε κάθε δεδομένο χρόνο. Τυπικά, πρόκειται για μια αντιστοίχιση από τις αντιληπτές καταστάσεις του περιβάλλοντος στις ενέργειες που θα πρέπει να εκτελούνται σε αυτές τις καταστάσεις. Αντιπροσωπεύει τον μηχανισμό λήψης αποφάσεων του πράκτορα, ενσωματώνοντας τις γνώσεις που αποκτήθηκαν κατά τη διάρκεια της εκπαίδευσης. Ανάλογα με τον αλγόριθμο ενίσχυσης μάθησης που έχει επιλεγεί, ο πράκτορας μπορεί να αλλάξει την πολιτική του ως αποτέλεσμα της εμπειρίας του. Επομένως, η πολιτική είναι συνήθως δυναμική.

Η πολιτική μπορεί να πάρει διάφορες μορφές, με απλούστερη έναν πίνακας αναζητήσεων ή μια απλή συνάρτηση. Σε πιο περίπλοκες καταστάσεις, μπορεί να περιλαμβάνει εκτεταμένους υπολογισμούς, όπως ένα νευρωνικό δίκτυο. Μπορεί να είναι στοχαστική, επιτρέποντας τη χρήση τυχαιότητας στην επιλογή ενεργειών. Μια στοχαστική πολιτική συχνά αναπαρίσταται ως π_t, όπου $\pi_t(a|s)$ είναι η πιθανότητα επιλογής της ενέργειας a τη στιγμή t, δεδομένης της τρέχουσας κατάστασης s.

• Συνάρτηση Αξίας

Ενώ το σήμα ανταμοιβής παρέχει άμεση ανατροφοδότηση για την καταλληλότητα συγκεκριμένων ενεργειών ή καταστάσεων, οι συναρτήσεις αξίας επεκτείνουν αυτή την έννοια εκτιμώντας το μακροπρόθεσμο όφελος καταστάσεων ή ζευγών κατάστασης-ενέργειας. Οι

συναρτήσεις αξίας καθοδηγούν τη διαδικασία λήψης αποφάσεων του πράκτορα. Αντί να επιλέγει ενέργειες που αποφέρουν την υψηλότερη άμεση ανταμοιβή, ο πράκτορας στοχεύει στην επιλογή ενεργειών που οδηγούν σε καταστάσεις με την υψηλότερη εκτιμώμενη αξία, μεγιστοποιώντας έτσι την αναμενόμενη απόδοση μακροπρόθεσμα.

Οι συναρτήσεις αξίας ορίζονται σε σχέση με συγκεκριμένες πολιτικές. Ανεπίσημα, η αξία μιας κατάστασης s σύμφωνα με μια πολιτική π, $v_\pi(s)$, είναι η αναμενόμενη απόδοση ξεκινώντας από την κατάσταση s και ακολουθώντας την πολιτική π στη συνέχεια. Ομοίως, η συνάρτηση αξίας-ενέργειας για την πολιτική π αναπαρίσταται ως $q_\pi(s, a)$.

Οι συναρτήσεις αξίας v_π και q_π εκτιμούνται συνεχώς από τις ακολουθίες παρατηρήσεων του πράκτορα κατά τη διάρκεια της αλληλεπίδρασής του με το περιβάλλον, σε αντίθεση με τις ανταμοιβές, οι οποίες παρέχονται άμεσα από το περιβάλλον. Αυτή η διαδικασία εκτίμησης περιλαμβάνει τη συγκέντρωση και την εξαγωγή πληροφοριών που συλλέγονται με την πάροδο του χρόνου. Για παράδειγμα, εάν ένας πράκτορας ακολουθεί την πολιτική π και διατηρεί για κάθε κατάσταση που συναντά έναν μέσο όρο των πραγματικών αποδόσεων που ακολούθησαν εκείνη την κατάσταση, τότε ο μέσος όρος θα συγκλίνει στην αξία της κατάστασης, v_π, καθώς ο αριθμός των φορών που συναντάται η κατάσταση τείνει στο άπειρο. Αν διατηρηθούν ξεχωριστοί μέσοι όροι για κάθε ενέργεια που εκτελείται σε μια κατάσταση, τότε αυτοί οι μέσοι όροι θα συγκλίνουν αντίστοιχα στις τιμές $q_\pi(s, a)$ (μέθοδος εκτίμησης Monte Carlo).

1.3.2 Marcov Decision Process (MDP)

Σε ένα πρόβλημα ενισχυτικής μάθησης το τρέχον σήμα κατάστασης λέγεται μαρκοβιανό όταν περιλαμβάνει όλη την πληροφορία που χρειάζεται για την λήψη απόφασης, χωρίς να απαιτούνται ιστορικά δεδομένα. Δηλαδή, αν για μια αλληλουχία διακριτών βημάτων , t = 0, 1, 2, 3, ..., σε κάθε βήμα t ο πράκτορας λαμβάνει την κατάσταση του περιβάλλοντος $S_t \in S$ και επιλέγει ενέργεια $A_t \in A(S_t)$, λαμβάνοντας ανταμοιβή $R_{t+1} \in R$ και μεταβαίνοντας στην κατάσταση S_{t+1}, ισχύει:

$$p(s', r|s, a) = P\{R_{t+1} = r, S_{t+1} = s'|S_t, A_t\}, \forall r, s', S_t, A_t. \tag{1.4}$$

Ένα πρόβλημα που ικανοποιεί την μαρκοβιανή ιδιότητα καλείται Markov Decision Process(MDP).

1.3.3 Εξισώσεις Bellman

Για ένα MDP η συνάρτηση αξίας ορίζεται ως:

$$\begin{aligned}
v_\pi(s) &= E_\pi[G_t|S_t = s] \\
&= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\middle| S_t = s\right] \\
&= E_\pi\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2}\middle| S_t = s\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2}\middle| S_{t+1} = s'\right]\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right], \quad \forall s, s' \in S, a \in A(s), r \in R \text{ [15].}
\end{aligned} \tag{1.5}$$

Η τελευταία μορφή της εξίσωσης καλείται Εξίσωση Bellman και εκφράζει τη σχέση μεταξύ της αξίας μιας κατάστασης και της αξίας των επόμενων, υπολογίζοντας όλες τις δυνατές επιλογές και ζυγίζοντας καθεμία με την πιθανότητα εμφάνισής της. Ομοίως για τη συνάρτηση αξίας-ενέργειας:

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a][15]. \tag{1.6}$$

Στόχος στην ενισχυτική μάθηση είναι να βρεθεί μια πολιτική που μεγιστοποιεί την αθροιστική ανταμοιβή μακροπρόθεσμα. Σε πεπερασμένα MDP, η βέλτιστη πολιτική μπορεί να προσδιοριστεί, μέσω των συναρτήσεων αξίας. Μια πολιτική π θεωρείται καλύτερη ή ίση με μια άλλη π' αν ισχύει ότι $v_\pi(s) \geq v_{\pi'}(s), \forall s \in S$. Αποδεικνύεται ότι υπάρχει πάντα τουλάχιστον μία πολιτική που υπερέχει από τις υπόλοιπες, η οποία συμβολίζεται ως $\pi*$ και έχει τη βέλτιστη συνάρτηση αξίας, v*:

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in S. \tag{1.7}$$

Επομένως, ορίζεται η Εξίσωση Βελτιστοποίησης Bellman ως:

$$\begin{aligned} v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\ &= \max_a E[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \ [15]. \end{aligned} \tag{1.8}$$

Σε πεπερασμένα MDP, η εξίσωση αυτή έχει μοναδική λύση ανεξάρτητα από την πολιτική, και αν είναι γνωστή η δυναμική του περιβάλλοντος, p, θεωρητικά μπορεί να υπολογιστεί, δρώντας με άπληστο (greedy) τρόπο ως προς την v_*.

1.3.4 Εξερεύνηση vs Εκμετάλλευση

Το δίλημμα εξερεύνησης-εκμετάλλευσης είναι μια θεμελιώδης πρόκληση στην ενισχυτική μάθηση, καθώς απαιτεί επιλογή ανάμεσα στην εκμετάλλευση γνωστών πληροφοριών για άμεσες ανταμοιβές και την εξερεύνηση νέων επιλογών για μελλοντική βελτίωση. Η σωστή ισορροπία είναι κρίσιμη: η υπερβολική εκμετάλλευση μπορεί να αφήσει να ξεφύγουν καλύτερες επιλογές, με κίνδυνο παγίδευσης σε τοπικό ελάχιστο, ενώ η υπερβολική εξερεύνηση σπαταλά πόρους. Ο πράκτορας πρέπει να εξισορροπεί, προσαρμόζοντας τη στρατηγική του στο δυναμικό περιβάλλον, ώστε να πετυχαίνει μακροπρόθεσμη επιτυχία.

1.3.5 Δυναμικός Προγραμματισμός

Ο όρος "δυναμικός προγραμματισμός" στην Ενισχυτική Μάθηση περιγράφει μια ομάδα αλγορίθμων που, επιλύοντας τις Εξισώσεις Βελτιστοποίησης Bellman, υπολογίζουν τη βέλτιστη πολιτική σε ένα MDP με γνωστό μοντέλο περιβάλλοντος. Από αυτούς, η Επανάληψη Πολιτικής και η Επανάληψη Αξίας είναι δύο βασικές προσεγγίσεις που υιοθετούν διαφορετικές στρατηγικές για την εύρεση της βέλτιστης πολιτικής.

Θα χρησιμοποιούνται κεφαλαία $V_\pi(), Q_\pi()$ για τις εκτιμήσεις των συναρτήσεων αξίας και ενέργειας- αξίας, ενώ τα $v_\pi(), q_\pi()$ θα συμβολίζουν τις πραγματικές συναρτήσεις υπό την πολιτική π, $v_*(), q_*()$ αν είναι βέλτιστες για όλες τις πολιτικές.

1.3.5.1 Επαναληπτική Αξιολόγηση Πολιτικής

Η Επαναληπτική Αξιολόγηση Πολιτικής είναι μια μέθοδος που εκτιμά τη συνάρτηση αξίας $V_\pi()$ για μια δεδομένη πολιτική π, μέσω επαναληπτικών ενημερώσεων βάσει της εξίσωσης Bellman. Ξεκινώντας από αυθαίρετες τιμές αξίας για κάθε κατάσταση, τις ενημερώνει διαδοχικά, με βάση τις πιθανότητες ενεργειών, μετάβασης και τις αναμενόμενες ανταμοιβές:

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} p(s', r|s, a)[r + \gamma V_k(s')] \tag{1.9}$$

Αυτό επαναλαμβάνεται μέχρι τη σύγκλιση σε σταθερές (με βάση κάποιο κατώφλι) τιμές, κάτι που είναι εγγυημένο για πεπερασμένο MDP [4]. Αυτή η διαδικασία βελτιώνει τις αρχικές εκτιμήσεις με βάση άλλες εκτιμήσεις (bootstrapping).

1.3.5.2 Επανάληψη Πολιτικής

Η Επανάληψη Πολιτικής υπολογίζει τη βέλτιστη πολιτική σε ένα MDP μέσω εναλλασσόμενων βημάτων. Αρχικά, για μια δεδομένη πολιτική, υπολογίζεται η συνάρτηση αξίας $v_\pi(s)$ μέσω Επαναληπτικής Αξιολόγησης Πολιτικής, μέχρι να συγκλίνει. Στη συνέχεια, η πολιτική βελτιώνεται επιλέγοντας, για κάθε κατάσταση, την ενέργεια που μεγιστοποιεί την αναμενόμενη απόδοση βάσει των τρεχουσών εκτιμήσεων:

$$\pi'(s) = \arg\max_a \sum_{s'} p(s'|s, a)(r + \gamma v_\pi(s')). \tag{1.10}$$

Η διαδικασία αυτή επαναλαμβάνεται έως ότου η πολιτική σταθεροποιηθεί και θεωρηθεί βέλτιστη. Σε πεπερασμένα MDP, η σύγκλιση είναι εγγυημένη [4].

1.3.5.3 Επανάληψη Αξίας

Η Επανάληψη Αξίας αποτελεί εναλλακτική λύση στην Επανάληψη Πολιτικής για την εύρεση της βέλτιστης πολιτικής. Σε αυτή τη μέθοδο, η συνάρτηση αξίας κάθε κατάστασης ενημερώνεται επαναληπτικά με βάση την Εξίσωση Βελτιστοποίησης Bellman, χωρίς αναμονή για πλήρη αξιολόγηση της πολιτικής:

$$V_{k+1}(s) = \max_a \sum_{s'} p(s'|s, a)(r + \gamma V_k(s')). \tag{1.11}$$

. Όταν επιτευχθεί η σύγκλιση, η βέλτιστη πολιτική προκύπτει επιλέγοντας, για κάθε κατάσταση, την ενέργεια που μεγιστοποιεί την αναμενόμενη απόδοση:

$$\pi_*(s) = \arg\max_a \sum_{s'} p(s'|s, a)(r + \gamma V(s')). \tag{1.12}$$

Αυτή η προσέγγιση είναι πιο αποδοτική υπολογιστικά, καθώς ενημερώνει ταυτόχρονα τόσο τη συνάρτηση αξίας όσο και την πολιτική.

1.3.5.4 Περιορισμοί

Ένα μεγάλο μειονέκτημα των μεθόδων Δυναμικού Προγραμματισμού είναι ότι απαιτούν πέρασμα από ολόκληρο το σύνολο καταστάσεων του MDP και βασίζονται σε ένα τέλειο μοντέλο, κάτι που σπάνια ισχύει. Επιπλέον, η συνάρτηση αξίας-ενέργειας υπολογίζεται ξεχωριστά για κάθε ακολουθία χωρίς δυνατότητα γενίκευσης. Επομένως, στην πράξη συνήθως οι συναρτήσεις αυτές προσεγγίζονται είτε με γραμμικό είτε με μη-γραμμικό τρόπο (π.χ. νευρωνικά δίκτυα).

1.3.6 Μάθηση Χρονικών Διαφορών

1.3.6.1 Γενικά

Η Μάθηση Χρονικών Διαφορών (Temporal Difference- TD) αποτελεί μια μέθοδο πρόβλεψης (εκτίμηση συνάρτησης αξίας) και ελέγχου (βελτίωση πολιτικής) που δεν απαιτεί πλήρη γνώση του περιβάλλοντος, όπως οι μέθοδοι δυναμικού προγραμματισμού. Βασίζεται σε δείγματα εμπειριών του πράκτορα από αλληλεπίδραση με το περιβάλλον του.

Συγκεκριμένα, η πιο απλή εκδοχή είναι η μέθοδος TD(0), που ενημερώνει σταδιακά την εκτίμηση της συνάρτησης αξίας κάθε κατάστασης με βάση τον κανόνα:

$$V(S_t) \leftarrow V(S_t) + a[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)], \tag{1.13}$$

όπου a είναι ο ρυθμός μάθησης και η τιμή $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ είναι το TD σφάλμα, που αντιπροσωπεύει τη διαφορά μεταξύ της τρέχουσας εκτίμησης και της ενημερωμένης με βάση την παρατηρούμενη μετάβαση. Η TD(0) αποδεικνύεται ότι συγκλίνει στην πραγματική συνάρτηση αξίας για κάθε δεδομένη πολιτική [4].

1.3.6.2 Εντός Πολιτικής vs. Εκτός Πολιτικής

Σε προβλήματα ελέγχου, ο στόχος είναι η βελτιστοποίηση της πολιτικής μέσω της εκμάθησης της συνάρτησης αξίας- ενέργειας (Q-function). Οι μέθοδοι TD, και όχι μόνο, χωρίζονται σε εντός και εκτός πολιτικής (on-policy και off-policy), ανάλογα με το πώς ενημερώνεται η Q-function. Στην περίπτωση on-policy (π.χ., Sarsa), η ίδια πολιτική που χρησιμοποιείται για την αλληλεπίδραση με το περιβάλλον ενημερώνει και τη Q-function.

Αντίθετα, στο off-policy (π.χ., Q-learning) η Q-function ενημερώνεται ώστε να προσεγγίζει την βέλτιστη Q ανεξάρτητα από την πολιτική που ακολουθείται. Δηλαδή, ο Sarsa ενημερώνει τη Q με βάση την επόμενη ενέργεια της τρέχουσας πολιτικής, ενώ το Q-learning χρησιμοποιεί το μέγιστο των Q τιμών της επόμενης κατάστασης για την ενημέρωση.

Algorithm 1: Αλγόριθμος Sarsa για επεισοδιακά προβλήματα

Input: Χώρος καταστάσεων S, Χώρος ενεργειών A, Ρυθμός μάθησης α, Παράγοντας έκπτωσης γ, Ρυθμός εξερεύνησης ϵ

Αρχικοποίηση $Q(s,a)$, $\forall s \in S$, $a \in A(s)$ αυθαίρετα, και $Q(\text{τερματική-κατάσταση}, \cdot) = 0$

Επανάληψη (για κάθε επεισόδιο):

    Αρχικοποίηση S
    Επανάληψη
        Επιλογή A από το S χρησιμοποιώντας την πολιτική που προκύπτει από το Q (π.χ., ϵ-greedy)
        Εκτέλεση ενέργειας A, παρατήρηση R, S'
        Επιλογή A' από το S' χρησιμοποιώντας την πολιτική που προκύπτει από το Q (π.χ., ϵ-greedy)
        Ενημέρωση $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
        $S \leftarrow S'$; $A \leftarrow A'$

Μέχρι να είναι η κατάσταση S **τερματική**

Algorithm 2: Αλγόριθμος Q-learning για επεισοδιακά προβλήματα

Input: Χώρος καταστάσεων S, Χώρος ενεργειών A, Ρυθμός μάθησης α, Παράγοντας έκπτωσης γ, Ρυθμός εξερεύνησης ϵ

Αρχικοποίηση $Q(s,a)$, $\forall s \in S$, $a \in A(s)$ αυθαίρετα, και $Q(\text{τερματική-κατάσταση}, \cdot) = 0$

Επανάληψη (για κάθε επεισόδιο):

    Αρχικοποίηση το S
    Επανάληψη
        Επιλογή A από το S χρησιμοποιώντας την πολιτική που προκύπτει από το Q (π.χ., ϵ-greedy)
        Εκτέλεση ενέργειας A, παρατήρηση R, S'
        Επιλογή A' από το S' χρησιμοποιώντας την πολιτική που προκύπτει από το Q (π.χ., ϵ-greedy)
        Ενημέρωση $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$

Μέχρι να είναι η κατάσταση S **τερματική**

1.3.7 Βελτιστοποίηση Πολιτικής vs. Q-learning

Οι μέθοδοι που έχουν αναλυθεί μέχρι τώρα (DP, TD) είναι σαν σε μορφή πίνακα, δηλαδή οι συναρτήσεις αξίας V και Q έχουν μία τιμή για κάθε κατάσταση ή ζεύγος κατάστασης-ενέργειας. Αυτή η προσέγγιση λειτουργεί μόνο σε περιβάλλοντα με μικρό αριθμό καταστάσεων και ενεργειών και δεν γενικεύεται για συνεχείς χώρους καταστάσεων ή ενεργειών. Συνεπώς,

για καλύτερη γενίκευση σε περιβάλλοντα χωρίς μοντέλο, χρησιμοποιούνται προσεγγίσεις για αυτές τις συναρτήσεις (function approximation), όπου η πολιτική ή η συνάρτηση αξίας-ενέργειας αναπαρίστανται ως παραμετροποιημένες συναρτήσεις με μια διανυσματική παράμετρο, η οποία προσαρμόζεται μέσω της εκπαίδευσης για τη μεγιστοποίηση της αναμενόμενης απόδοσης. Αυτή η προσέγγιση αποτελεί τη βάση της Βαθιάς Ενισχυτικής Μάθησης (Deep Reinforcement Learning-DRL), που ενσωματώνει νευρωνικά δίκτυα σε προβλήματα ενισχυτικής μάθησης.

1.3.7.1 Μέθοδοι Κλίσης Πολιτικής (Βελτιστοποίηση Πολιτικής)

Σε αυτή την προσέγγιση, η πολιτική αναπαρίσταται ρητά ως $\pi_\theta(a|s)$, όπου $\boldsymbol{\theta}$ είναι το διάνυσμα παραμέτρων (π.χ., τα βάρη ενός νευρωνικού δικτύου). Ο στόχος είναι να ενημερωθούν οι παράμετροι $\boldsymbol{\theta}$ έτσι ώστε να καταστήσουν την π_θ βέλτιστη πολιτική, η οποία θα μεγιστοποιεί την αναμενόμενη αθροιστική ανταμοιβή. Επομένως, η αντικειμενική συνάρτηση για την βελτιστοποίηση της πολιτικής σε επεισοδιακά προβλήματα ορίζεται ως:

$$J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{E}[R(\tau)], \tag{1.14}$$

όπου $R(\tau)$ αναπαριστά την απόδοση από την τροχιά τ. Η πολιτική βελτιστοποιείται μέσω ανόδου κλίσης (gradient ascent):

$$\theta_{k+1} \leftarrow \theta_k + a\nabla_\theta J(\pi_\theta)|\pi_{\theta_k}, \tag{1.15}$$

όπου $\boldsymbol{\theta_k}$ είναι η τιμή της παραμέτρου στην επανάληψη k και a είναι ο ρυθμός εκμάθησης. Η διαίσθηση πίσω από αυτή τη διαδικασία είναι ότι η κλίση της αντικειμενικής συνάρτησης δείχνει τη κατεύθυνση στην οποία πρέπει να μετακινηθεί η $\boldsymbol{\theta}$, ώστε να αυξηθεί το $\pi_{\theta_t}(a|s)$ όσο πιο γρήγορα γίνεται.

Αποδεικνύεται ότι ([16]):

$$\nabla_\theta J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{E}[\sum_{t=0}^{T} \nabla_\theta log\pi_\theta(a_t|s_t)\Phi_t], \tag{1.16}$$

όπου ο πρώτος όρος δείχνει πώς αλλάζει η πιθανότητα επιλογής της ενέργειας a_t στην s_t ως προς τη θ και ο δεύτερος χρησιμεύει ως βάρος για την ενημέρωση. Υποθέτοντας ότι η πολιτική αναπαρίσταται με τρόπο που επιτρέπει τον υπολογισμό της κλίσης της, η μέση τιμή υπολογίζεται εκτελώντας την πολιτική στο περιβάλλον για να συλλεχθεί ένα σύνολο τροχιών D και λαμβάνοντας τον μέσο όρο:

$$\nabla_\theta J(\pi_\theta) = \frac{1}{|D|}\sum_{\tau\in D}\sum_{t=0}^{T} \nabla_\theta log\pi_\theta(a_t|s_t)\Phi_t. \tag{1.17}$$

Η συνάρτηση Φ_t μπορεί να πάρει διαφορετικές μορφές, όπως η συνολική απόδοση του επεισοδίου, $R(\tau)$, η απόδοση από το τρέχον βήμα και ύστερα, $\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})$ και η Συνάρτηση Πλεονεκτήματος, $A_{\pi_\theta}(s_t, a_t) = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t)$. Η τιμή $A_{\pi_\theta}(s_t, a_t)$ δείχνει πόσο καλύτερο ή χειρότερο είναι να επιλεγεί η a στην s συγκριτικά με την επιλογή που υπαγορεύει η πολιτική. Η Φ_t μπορεί ακόμη να είναι η $\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t)$. Η baseline function $b(s_t)$ χρησιμοποιείται για μείωση της διασποράς και συνήθως είναι μια προσέγγιση της συνάρτησης αξίας, $V_\phi(s_t)$. Όλες αυτές οι διαφορετικές μορφές δεν εισάγουν επιπλέον προκατάληψη στον υπολογισμό της κλίσης [15] και η χρήση τους εξαρτάται από τον επιλεγμένο αλγόριθμο.

1.3.7.2 Μέθοδοι Βασισμένες στην Αξία (Q-learning)

Οι μέθοδοι αυτής της κατηγορίας μαθαίνουν μια προσέγγιση $Q_\theta(s,a)$ για τη βέλτιστη συνάρτηση αξίας-ενέργειας, $q^*(s,a)$. Επεκτείνουν την έννοια του Q-learning από την ενότητα TD, ενσωματώνοντας την προσέγγιση συναρτήσεων. Αυτή η βελτιστοποίηση πραγματοποιείται off-policy, που σημαίνει ότι κάθε ενημέρωση μπορεί να χρησιμοποιεί δεδομένα που συλλέγονται σε οποιοδήποτε σημείο της εκπαίδευσης, ανεξάρτητα από το πώς ο πράκτορας εξερευνά το περιβάλλον όταν συλλέγονται τα δεδομένα. Η συνάρτηση σφάλματος είναι το Μέσο Τετραγωνικό Σφάλμα TD:

$$L(\theta) = \underset{s,a\sim p()}{E}([r_{t+1} + \gamma \max_a Q_\theta(s_{t+1},a) - Q_\theta(s_t,a_t))^2], \tag{1.18}$$

όπου γ είναι ο παράγοντας έκπτωσης και $p(s,a)$ είναι η πολιτική συμπεριφοράς (μια κατανομή πιθανοτήτων στις καταστάσεις και ενέργειες μέσω της οποίας συλλέγονται οι εμπειρίες, π.χ. $\epsilon-$greedy). Οι παράμετροι $\boldsymbol{\theta}$ ενημερώνονται μέσω κατάβασης κλίσης (gradient descent). Ο στόχος παραμένει να βρεθεί η βέλτιστη πολιτική, η οποία επιτυγχάνεται μέσω της σχέσης μεταξύ q^* και π^*:

$$a(s) = \arg\max_a Q_\theta(s,a). \tag{1.19}$$

1.3.7.3 Σύγκριση

Οι Μέθοδοι Βελτιστοποίησης Πολιτικής και Βασισμένες στην Αξία έχουν ξεχωριστά πλεονεκτήματα. Η δεύτερη κατηγορία παράγει αποκλειστικά ντετερμινιστικές πολιτικές και δυσκολεύεται να επεκταθεί σε συνεχείς χώρους ενεργειών, καθώς το νευρωνικό δίκτυο πρέπει να υπολογίζει αξίες για κάθε ενέργεια. Αντίθετα, οι Μέθοδοι Βελτιστοποίησης Πολιτικής ενημερώνουν απευθείας την πολιτική μέσω της κλίσης της και, παρά τη δυνατότητα εμφάνισης υψηλής διακύμανσης, αποδίδουν καλύτερα σε περιβάλλοντα υψηλών διαστάσεων ή συνεχούς χώρου ενεργειών.

1.3.8 Αλγόριθμοι Βαθιάς Ενισχυτικής Μάθησης

1.3.8.1 Μέθοδοι Δράστη-Κριτή

Οι μέθοδοι Δράστη-Κριτή (Actor-Critic) είναι μια κατηγορία αλγορίθμων που συνδυάζουν τα πλεονεκτήματα των value-based και policy-based προσεγγίσεων, χρησιμοποιώντας δύο νευρωνικά δίκτυα: τον Δράστη, που καθορίζει την πολιτική και επιλέγει ενέργειες και παραμετροποιείται από τις μεταβλητές $\boldsymbol{\theta}$, και τον Κριτή, που παραμετροποιείται από τις $\boldsymbol{w}$ και είναι υπεύθυνος για την αξιολόγηση των ενεργειών, εκτιμώντας την αναμενόμενη απόδοση και παρέχοντας ανατροφοδότηση. Αυτή η συνεργασία εξασφαλίζει πιο σταθερή και ευέλικτη σύγκλιση, αφού μειώνει την υψηλή διακύμανση που συνήθως έχουν οι policy-based μέθοδοι.

Ειδικότερα, ο δράστης βελτιστοποιεί την αντικειμενική συνάρτηση $J(\theta)$, που αντιπροσωπεύει την αναμενόμενη απόδοση με την πολιτική π_θ, μέσω ανόδου κλίσης:

$$\theta_{t+1} \leftarrow \theta_t + a\nabla_\theta J(\theta)|\theta_t, \tag{1.20}$$

όπου:

$$\nabla_\theta J(\theta) = \underset{\tau\sim\pi_\theta}{E}[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t)(R_{t+1} - V(s_t)]. \tag{1.21}$$

Το R_{t+1} συμβολίζει την αθροιστική ανταμοιβή από αυτό το βήμα μέχρι το τέλος του επεισοδίου και προσεγγίζεται ως:

$$R_{t+1} \approx r_{t+1} + \gamma V(s_{t+1}). \tag{1.22}$$

Συγχρόνως, ο κριτής λαμβάνει τις τιμές r_t, s_{t+1} από το περιβάλλον και στοχεύει στην ελαχιστοποίηση του TD error:

$$\delta_t = r_t + V_w(s_{t+1}) - V_w(s_t). \tag{1.23}$$

Επομένως η συνάρτηση σφάλματος είναι:

$$J(w) = (r_t + \gamma V_w(s_{t+1}) - V_w(s_t))^2 \tag{1.24}$$

και την ελαχιστοποιεί μέσω καθόδου κλίσης:

$$w_{t+1} \leftarrow w_t - a_V \nabla_w J(w). \tag{1.25}$$

Οι εκτιμήσεις του κριτή περνάνε στον δράστη, οπότε η τελική μορφή της συνάρτησης βελτιστοποίησης του δράστη είναι:

$$\nabla_\theta J(\theta) = \underset{\tau \sim \pi_\theta}{E}[\sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_w(s_{t+1}) - V_w(s_t)]. \tag{1.26}$$

Διαισθητικά, όταν ο κριτής υπολογίζει θετικό TD error – δηλαδή η αξία της τρέχουσας κατάστασης είναι μεγαλύτερη από αυτήν που εκτιμήθηκε – ο δράστης ενημερώνει τις παραμέτρους του ώστε να αυξήσει την πιθανότητα επιλογής της ενέργειας αυτής. Αντίθετα, ένα αρνητικό TD error αποθαρρύνει την επιλογή της ίδιας ενέργειας στο μέλλον.

1.3.8.1.1 Προσεγγιστική Βελτιστοποίηση Πολιτικής

Η Προσεγγιστική Βελτιστοποίηση Πολιτικής (Proximal Policy Optimization- PPO) είναι ένας on-policy, χωρίς μοντέλο αλγόριθμος Δράστη-Κριτή. Έχει σχεδιαστεί για να βελτιώσει τη σταθερότητα και την αποδοτικότητα των ενημερώσεων πολιτικής, περιορίζοντας μεγάλες αλλαγές πολιτικής που θα μπορούσαν να αποσταθεροποιήσουν την εκπαίδευση. Υπάρχουν δύο κύριες παραλλαγές: PPO-Clip και PPO-Penalty, που χρησιμοποιούν διαφορετικούς μηχανισμούς για τον περιορισμό του μεγέθους των ενημερώσεων πολιτικής. Η πιο διαδεδομένη υλοποίηση είναι η PPO-Clip. Στο PPO-Clip, ο ρόλος του Δράστη παραμένει ο ίδιος, όπως περιγράφηκε προηγουμένως. Ωστόσο, ο Κριτής στοχεύει στη μεγιστοποίηση μιας τροποποιημένου αντικειμενικής συνάρτησης:

$$L^{CLIP}(\theta) = \underset{\tau \sim \pi_{\theta_{old}}}{E}[\min(r(\theta)\hat{A}^{\pi_{\theta_{old}}}(s,a), clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_{old}}}(s,a))], \tag{1.27}$$

όπου:

- $\pi_{\theta_{old}}$ είναι η παλιά πολιτική (αυτή που χρησιμοποιήθηκε για τη συλλογή των δεδομένων), ενώ π_θ είναι η νέα πολιτική που βελτιστοποιείται,
- $r(\theta)$ είναι ο λόγος πιθανοτήτων μεταξύ της νέας και της παλιάς πολιτικής, που ποσοτικοποιεί τη διαφορά τους:

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{1.28}$$

- ϵ είναι μια μικρή υπερπαράμετρος που καθορίζει πόσο μπορεί να αποκλίνει η νέα πολιτική από την παλιά,

- Η συνάρτηση clip περιορίζει τον λόγο πιθανοτήτων, διασφαλίζοντας ότι παραμένει εντός του διαστήματος $[1-\epsilon, 1+\epsilon]$,

- $\hat{A}^{\pi_{\theta_{old}}}(s,a)$ είναι η εκτίμηση του πλεονεκτήματος (advantage estimate). Ο απλούστερος τρόπος για να εκτιμηθεί είναι μέσω του TD error:

$$\hat{A}^{\pi_{\theta_{old}}}(s,a) \approx \delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t). \tag{1.29}$$

  Ωστόσο, αυτή η ενημέρωση μιας μόνο μετάβασης μπορεί να οδηγήσει σε υψηλή διακύμανση στις ενημερώσεις πολιτικής. Για να το αντιμετωπίσει αυτό, ο πράκτορας χρησιμοποιεί τη Γενικευμένη Εκτίμηση Πλεονεκτήματος (Generalized Advantage Estimation - GAE). Η GAE εξομαλύνει και σταθεροποιεί την εκτίμηση του πλεονεκτήματος ενσωματώνοντας πολλαπλές μελλοντικές ανταμοιβές, χρησιμοποιώντας ένα εκθετικά σταθμισμένο άθροισμα των TD errors:

$$\hat{A}^{\pi_{\theta_{old}}}(s_t,a_t) \approx \sum_{l=0}^{T-1} (\gamma\lambda)^l \delta_{t+l} \tag{1.30}$$

Ο όρος $r(\theta)\hat{A}^{\pi_{\theta_k}}()$ ενθαρρύνει την πολιτική να αυξήσει την πιθανότητα των ενεργειών με θετικά πλεονεκτήματα (καλές ενέργειες) και να μειώσει την πιθανότητα των ενεργειών με αρνητικά πλεονεκτήματα. Ταυτόχρονα, η συνάρτηση clip διασφαλίζει ότι ο λόγος πιθανοτήτων δεν αποκλίνει υπερβολικά από το 1, αποτρέποντας πολύ μεγάλες ενημερώσεις στην πολιτική σε ένα μόνο βήμα. Αυτός είναι ο βασικός μηχανισμός που καθιστά το PPO σταθερό.

Ο πλήρης αλγόριθμος PPO-Clip περιλαμβάνει τα εξής βήματα [17]:

Algorithm 3: Αλγόριθμος PPO

Συλλογή συνόλου διαδρομών D_k εκτελώντας την πολιτική π_{θ_k} στο περιβάλλον.

Υπολογισμός των εκτιμήσεων πλεονεκτήματος $\hat{A}^{\pi_{\theta_k}}(s,a)$ χρησιμοποιώντας GAE (ή άλλη μέθοδο) με βάση την τρέχουσα συνάρτηση αξίας $V_{w_k}(s)$.

Ενημέρωση του δικτύου πολιτικής μέσω ανόδου κλίσης:

Για Ν επαναλήψεις:

Ενημέρωση της πολιτικής:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta L^{CLIP}(\theta)|_{\theta_k},$$

όπου ο περικομμένος στόχος είναι:

$$L^{CLIP}(\theta_k) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T-1} \min\Big(r(\theta)\hat{A}^{\pi_{\theta_k}}(s_t,a_t), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_k}}(s_t,a_t)\Big)$$

Ενημέρωση του δικτύου κριτή μέσω καθόδου κλίσης:

$$w_{t+1} \leftarrow w_t - \alpha_V \nabla_w J(w_t),$$

όπου η συνάρτηση απώλειας του κριτή δίνεται από:

$$J(w) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T-1} \left(r_{t+1} + \gamma V_w(s_{t+1}) - V_w(s_t)\right)^2$$

1.4 Διαμόρφωση του Προβλήματος

Σε αυτήν την ενότητα παρουσιάζεται η διαμόρφωση του προβλήματος, που αφορά τις προκλήσεις ελέγχου αποδοχής slices και τοποθέτησης Εικονικοποιημένων Λειτουργιών Δικτύου (VNF) στην αρχιτεκτονική του O-RAN, αξιοποιώντας τη Βαθιά Ενισχυτική Μάθηση (Deep Reinforcement Learning - DRL). Στόχος είναι η μοντελοποίηση ενός ρεαλιστικού περιβάλλοντος O-RAN και η ανάπτυξη ενός πράκτορα RL, ο οποίος θα λαμβάνει αποδοτικά αποφάσεις σχετικά με την αποδοχή, την τοποθέτηση και τη διάσπαση των εισερχόμενων αιτημάτων slices.

1.4.1 Μοντέλο Δικτύου

Υποτίθεται μια τυπική αρχιτεκτονική βασισμένη στο O-RAN, όπου τα Edge Clouds (ECs) συνδέονται μέσω συνδέσμων Fronthaul (FH) με την ραδιομονάδα (RU), προκειμένου να υποστηρίζουν εφαρμογές με αυστηρές απαιτήσεις χαμηλής καθυστέρησης. Υποτίθεται μια απλοποιημένη μορφή όπου κάθε EC περιλαμβάνει ένα μόνο Distributed Unit (DU). Ένα μοναδικό Regional Cloud (RC), συνδεδεμένο μέσω συνδέσμων Midhaul (MH) με τα ECs, λειτουργεί ως κεντρικός υπολογιστικός πόρος, υπεύθυνος για επεξεργασία υψηλού επιπέδου και συντονισμό μεταξύ πολλών ECs. Οι FH σύνδεσμοι εξασφαλίζουν χαμηλή καθυστέρηση

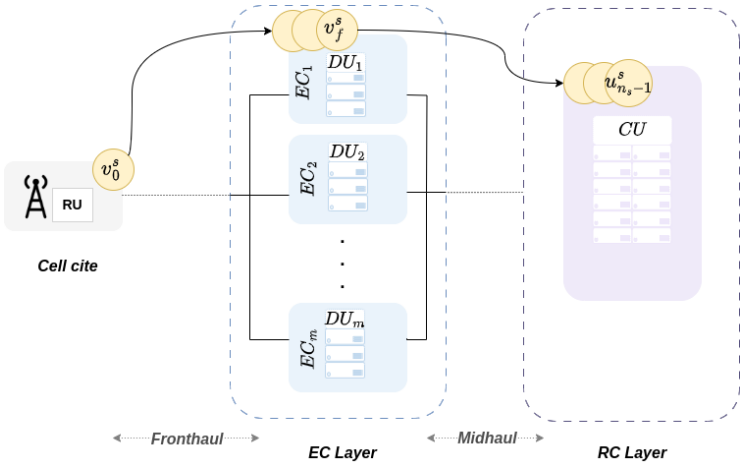


Figure 1.2: Μοντέλο Δικτύου.

επικοινωνίας μεταξύ RU και ECs, ενώ οι MH σύνδεσμοι παρέχουν υψηλής χωρητικότητας σύνδεση μεταξύ ECs και RC. Τα ECs και το RC χαρακτηρίζονται από τη συνολική υπολογιστική τους ικανότητα, μετρημένη σε πυρήνες CPU, ενώ κάθε σύνδεσμος περιγράφεται από την χωρητικότητα εύρους ζώνης και την καθυστέρηση που επιβάλλει.

Οι παράμετροι που σχετίζονται με τη διατύπωση του περιβάλλοντος παρουσιάζονται στον Πίνακα **??**.

Σύμβολο	**Περιγραφή**
E	Σύνολο ECs
F	Σύνολο διαθέσιμων VNFs
F_s	Ταξινομημένο υποσύνολο VNFs που αποτελεί την αλυσίδα λειτουργιών για το slice s
$f \in F_s$	Δείκτης VNF στην αλυσίδα λειτουργιών του slice s
n_s	Πλήθος VNFs του slice s
t_s	Χρόνος άφιξης του slice s
ht_s	Χρόνος διατήρησης του slice s
pr_s	Τιμή προτεραιότητας του slice s
$R(t)$	Σύνολο slices που έχουν αφιχθεί μέχρι το χρόνο t
$c_{s,f}$	Απαίτηση σε CPU του VNF f του slice s
$b_{s,f}$	Απαιτήσεις εύρους ζώνης μεταξύ δύο διαδοχικών VNFs $f-1, f$ του slice s
$D_{max,s}$	Απαίτηση για end-to-end καθυστέρηση του slice s
$C_e(t)$	Χρήση CPU στο cloud $e \in \mathcal{E}$
$B_{FH,e}(t)$ $(B_{MH,e}(t))$	Χρήση εύρους ζώνης μεταξύ RU (RC) και EC $e \in \mathcal{E}$
CE_e	Συνολική υπολογιστική ικανότητα του EC e για κάθε χρονική στιγμή t
CR	Συνολική υπολογιστική ικανότητα του RC
$CB_{F,e}$ $(CB_{M,e})$	Συνολικό εύρος ζώνης των συνδέσμων FH (MH) που σχετίζονται με $e \in \mathcal{E}$
$\delta_{r,e}$, $\delta_{e,\mathcal{R}}$	Παράμετροι καθυστέρησης
P^{max}	Μέγιστη κατανάλωση ισχύος ενός EC
γ	Αναλογία της καταναλισκόμενης ισχύος σε αδρανή server ως προς P_{max}
P_{net}^{max}	Μέγιστη κατανάλωση ισχύος των δικτυακών συνδέσμων
$P_{net,e}^{fix}$	Σταθερή κατανάλωση ισχύος σύνδεσης δικτύου μεταξύ RU και $e \in \mathcal{E}$

Table 1.3: Περιγραφή των παραμέτρων του προβλήματος.

1.4.2 Μοντέλο Αιτήματος Slice

Το δίκτυο λαμβάνει αιτήματα για slices με τη μορφή Αλυσίδας Λειτουργιών (Service Function Chain- SFC), τα οποία καθορίζουν τη σειριακή σειρά επεξεργασίας των VNFs, με δείκτες $v_f \in F$ (όπου F είναι το σύνολο των διαθέσιμων VNFs) εντός του slice. Με άλλα λόγια, ένα slice αντιστοιχεί σε ένα διατεταγμένο υποσύνολο του F, το οποίο σημειώνεται ως $F_s = \{{v_0}^s, {v_1}^s, ..., {v_{n_s-1}}^s\} \subseteq F$. Είναι σημαντικό να σημειωθεί ότι τα VNFs με δείκτες 0 και 1 είναι σταθερά σε όλα τα αιτήματα για slices, με το πρώτο υποχρεωτικά τοποθετημένο στο RU και το δεύτερο στο DU. Επιπλέον, κάθε slice s χαρακτηρίζεται από συγκεκριμένες απαιτήσεις σε υπολογιστικούς και δικτυακούς πόρους. Αυτές ορίζονται ανά VNF και, συγκεκριμένα, για κάθε VNF ${v_f}^s \in F_s$ το ${c_f}^s$ αντιπροσωπεύει τη ζήτηση σε πυρήνες CPU και το ${b_f}^s$ το απαιτούμενο εύρος ζώνης για τη μεταφορά δεδομένων μεταξύ των διαδοχικών VNFs ${v_f}^s, {v_{f+1}}^s$ μέσω ενός δικτυακού συνδέσμου. Επιπλέον, κάθε αίτημα s φτάνει σε συγκεκριμένο χρονικό διάστημα t_s εντός ενός περιορισμένου χρονικού ορίζοντα H. Το αίτημα περιλαμβάνει επίσης χρόνο διατήρησης ht_s, που δείχνει τη διάρκεια κατά την οποία το slice παραμένει ενεργό μετά την αποδοχή του. Επίσης, ορίζεται μια απαίτηση για καθυστέρηση από άκρο σε άκρο $D_{max,s}$ και μια τιμή προτεραιότητας pr_s για κάθε αίτημα slice, που αντιπροσωπεύουν το επίπεδο ανοχής στην καθυστέρηση και τη σχετική σημασία του αντίστοιχα.

Οι βασικές μεταβλητές απόφασης είναι οι δείκτες αποδοχής ή απόρριψης ενός slice και η τοποθέτηση των VNFs σε περίπτωση αποδοχής. Για αυτόν τον σκοπό, ορίζεται η δυαδική μεταβλητή $X_s(t)$, η οποία ισούται με 1 αν ένα αίτημα slice έχει γίνει αποδεκτό στο χρόνο t ή νωρίτερα, και 0 διαφορετικά. Για την τοποθέτηση των VNFs, ορίζονται οι δυαδικές μεταβλητές $x^e_{s,f}(t)$, $y_{s,f}(t), \forall e \in \mathcal{E}, f \in F_s, s \in R(t)$, όπως ακολούθως:

$$x^e_{s,f}(t)/y_{s,f}(t) = \begin{cases} 1, & f \in F_s \text{ τοποθετείται στο EC } e \in \mathcal{E}/RC \text{ στο χρόνο } t, \\ 0, & \text{διαφορετικά.} \end{cases} \tag{1.31}$$

Επιπλέον, μια επιπρόσθετη μεταβλητή $sr_s(t)$ δείχνει αν ένα ενεργό (δηλαδή μη ληγμένο) slice s έχει γίνει αποδεκτό στο χρόνο t:

$$sr_s(t) = \begin{cases} X_s(t), & t_s \leq t < (t_s + ht_s), \\ 0, & \text{διαφορετικά.} \end{cases} \tag{1.32}$$

Η τοποθέτηση των slices υπόκειται φυσικά σε αρκετούς περιορισμούς, όπως οι χωρητικότητες των στοιχείων του δικτύου και οι απαιτήσεις καθυστέρησης που συνδέονται με κάθε slice [18]. Οι περιορισμοί αυτοί έχουν εκφραστεί μαθηματικά στο Κεφάλαιο 5.3.

1.4.3 Κόστος

Τα συνολικά έσοδα που συσσωρεύονται στο σύστημα σε κάθε χρονικό βήμα t ορίζονται ως:

$$ReV(t) = \sum_{s \in R(t)} sr_s(t) \cdot pr_s. \tag{1.33}$$

Το κόστος του δικτύου προέρχεται από την κατανάλωση ισχύος των ECs και των δικτυακών συνδέσμων που χρησιμοποιούνται για την ανάπτυξη των slices. Για το σχεδιασμό

του κόστους κατανάλωσης ισχύος των συνδέσμων FH και MH, ορίζονται δύο επιπλέον τύποι μεταβλητών για να καθορίσουν τη χρήση τους. Έστω $u_e(t)$ μια δυαδική μεταβλητή που εκφράζει εάν ο σύνδεσμος FH που συνδέει το EC $e \in \mathcal{E}$ με το RU χρησιμοποιείται στο t. Αυτή η μεταβλητή τίθεται σε 1 μόνο αν τουλάχιστον ένα slice έχει το VNF v_1 τοποθετημένο στο EC e, δηλαδή,

$$u_e(t) \geq x^e_{s,1}(t), \ \ \forall s \in R(t), \forall e \in \mathcal{E}. \tag{1.34}$$

Έστω επίσης $v_e(t)$ μια δυαδική μεταβλητή που εκφράζει αν ο σύνδεσμος MH μέσω του EC $e \in \mathcal{E}$ χρησιμοποιείται στο t. Αυτή η μεταβλητή τίθεται σε 1 αν για δύο διαδοχικά VNFs οποιουδήποτε slice, το ένα τοποθετείται στο EC e και το άλλο στο RC, δηλαδή,

$$v_e(t) \geq x^e_{s,f}(t) y_{s,f+1}(t), \ \ \forall s \in R(t), \forall f \in F_s \setminus \{n_s - 1\}, \forall e \in \mathcal{E}. \tag{1.35}$$

Για να εκφραστεί το κόστος κατανάλωσης ισχύος των ECs, χρησιμοποιείται το μοντέλο αποδοτικής τοποθέτησης VNF από το [19], δηλαδή:

$$PC^{EC}(t) = \sum_{e \in \mathcal{E}} \left(u_e(t) \gamma P^{max} + (1 - \gamma) \frac{C_e(t)}{CE_e} P^{max} \right). \tag{1.36}$$

Για τους συνδέσμους που συνδέουν το RU με τα ECs:

$$PC^{RU-E}(t) = \sum_{e \in \mathcal{E}} \left(u_e(t) P^{fix}_{net,e} + \frac{B_{FH,e}(t)}{CB_{F,e}} P^{max}_{net} \right). \tag{1.37}$$

Τέλος, στην περίπτωση των συνδέσμων που συνδέουν τα ECs με το RC, εκφράζεται ως:

$$PC^{E-R}(t) = \sum_{e \in \mathcal{E}} \left(v_e(t) P^{fix}_{net,e} + \frac{B_{MH,e}(t)}{CB_{M,e}} P^{max}_{net} \right). \tag{1.38}$$

1.4.4 MDP

Στη συνέχεια, για την ανάπτυξη της λύσης με Ενισχυτική Μάθηση πρέπει να οριστεί επίσημα το Markov Decision Process (MDP).

Στο προτεινόμενο πλαίσιο, τα αιτήματα για slices παρουσιάζονται στον agent, ο οποίος στη συνέχεια αποφασίζει αν θα αποδεχτεί κάθε ένα από αυτά με βάση τη σειρά άφιξής τους. Κάθε βήμα απόφασης του RL αντιστοιχεί σε ένα αίτημα slice. Σε περιπτώσεις όπου πολλά αιτήματα φτάνουν στο ίδιο χρονικό βήμα, η επιλογή της σειράς επίλυσης γίνεται αυθαίρετα. Για αποφυγή σύγχυσης, ο δείκτης k δηλώνει το βήμα απόφασης του RL για αποδοχή ή απόρριψη ενός slice που αντιστοιχεί σε ένα αίτημα, ενώ t_k δηλώνει το χρονικό διάστημα του ελέγχου κατά το οποίο λαμβάνει χώρα η απόφαση k. Πολλαπλά βήματα απόφασης του RL, με δείκτες $k, k+1, ..., m$, μπορεί να αναφέρονται στο ίδιο χρονικό διάστημα $t_k = t_{k+1} = ... = t_m$ όταν φτάνουν ταυτόχρονα πολλαπλά αιτήματα slices. Επιπλέον, τα slices που δεν έγιναν αποδεκτά επανεισάγονται στον πράκτορα στο επόμενο χρονικό βήμα, εφόσον ο χρόνος διατήρησής τους δεν έχει λήξει. Σε αυτή την περίπτωση, αυτά τα αιτήματα αντιμετωπίζονται ως νέα αιτήματα slices, με τον χρόνο διατήρησης να μειώνεται κατάλληλα. Ο πράκτορας συνεχίζει να επεξεργάζεται αιτήματα slices έως ότου δεν απομείνουν ενεργά μη αποδεκτά slices ή έως το τέλος του επεισοδίου, που εκτείνεται σε H χρονικά βήματα.

1.4.4.1 Κατάσταση

Σε κάθε βήμα απόφασης, ο πράκτορας χρειάζεται επαρκείς πληροφορίες τόσο για την τρέχουσα κατάσταση του δικτύου όσο και για το επερχόμενο αίτημα slice, πάνω στο οποίο θα βασιστεί η απόφαση. Επομένως, η κατάσταση του πράκτορα σε κάθε βήμα k, όταν επεξεργάζεται το αίτημα slice s, αναπαρίσταται από την ακόλουθη πλειάδα:

$$(\mathbf{AC}_k, \mathbf{AB}_k, \mathbf{AT}_k, SI_k, t_k),$$

όπου:

- $\mathbf{AC}_k = [AC_1(k), ..., AC_{|E|}(k), AC_{RC}(k)]$, δηλαδή η διαθέσιμη χωρητικότητα όλων των ECs και του RC,
- $\mathbf{AB}_k = [AB_{FH_1}(k), ..., AB_{FH_{|E|}}(k), AB_{MH_1}(k), ..., AB_{MH_{|E|}}(k)]$, δηλαδή η διαθέσιμη χωρητικότητα εύρους ζώνης όλων των δικτυακών συνδέσμων FH και MH,
- $\mathbf{AT}_k = [AT_{EC_1}(k), ..., AT_{EC_{|E|}}(k), AT_{MH_1}(k), ..., AT_{MH_{|E|}}(k)]$, δηλαδή ο χρόνος που απομένει για κάθε EC ή σύνδεσμο MH μέχρι να απενεργοποιηθεί σύμφωνα με την τρέχουσα διαμόρφωση του δικτύου,
- $SI_k = (pr_s, D_{max,s}, ht_s, c_{s,1}, ..., c_{s,n_s-1}, b_{s,1}, ..., b_{s,n_s-1})$, δηλαδή όλες οι απαραίτητες πληροφορίες του slice που πρόκειται να επεξεργαστεί,
- t_k: το τρέχον χρονικό διάστημα του ορίζοντα ελέγχου.

1.4.4.2 Ενέργεια

Ο πράκτορας αποφασίζει από κοινού αν θα αποδεχτεί το slice, ποιο EC θα το εξυπηρετήσει σε περίπτωση αποδοχής, και τον αριθμό των VNFs του slice που θα τοποθετηθούν σε αυτό το EC (δηλαδή τη διάσπαση λειτουργιών). Συγκεκριμένα, ο πράκτορας λαμβάνει την απόφαση (e, v), όπου $e \in \{1, ..., |E|\}$ αναπαριστά το δείκτη του επιλεγμένου EC, και $v \in \{1, ..., n_s - 1\}$ αντιστοιχεί στον αριθμό VNFs που θα τοποθετηθούν στο edge (με το v_0 να ανατίθεται πάντα στο RU). Για να ευθυγραμμιστεί με το δίκτυο Δράστη, το ζεύγος ενεργειών απεικονίζεται σε μονοδιάστατο χώρο μέσω της συνάρτησης $f(e, v) = (e - 1)(n_s - 1) + v$. Επομένως, για κάθε αίτημα slice s που επεξεργάζεται στο t_k (με $t_s \leq t_k$), ο πράκτορας λαμβάνει ενέργεια a_k, από την οποία το (e_k, v_k) υπολογίζεται μέσω της αντίστροφης απεικόνισης της $f(e, v)$. Η ενέργεια $a_k = 0$ αντιστοιχεί σε απόρριψη του slice στο χρόνο t_k. Σε αυτή την περίπτωση, ο χρόνος διατήρησής του μειώνεται κατά 1. Αν ο ενημερωμένος χρόνος διατήρησης φτάσει στο μηδέν, το slice απορρίπτεται οριστικά από το δίκτυο, διαφορετικά το slice επανεισάγεται στον πράκτορα στο επόμενο χρονικό διάστημα του ορίζοντα, όπου λαμβάνεται νέα ανεξάρτητη απόφαση.

1.4.4.3 Ανταμοιβή

Ορίζονται οι ακόλουθες τιμές:

- $ReV_s = pr_s \cdot ht_s$ είναι τα συνολικά έσοδα που προκύπτουν από την αποδοχή του slice,
- PC_s^{EC} είναι η κατανάλωση ισχύος στο επιλεγμένο EC:

$$PC_s^{EC} = \max\{ht_s - AT_{EC,e_k}(k), 0\}\gamma P^{max} + (1-\gamma)\frac{\sum_{f=1}^{v_k} c_{s,f}}{CE_{e_k}} P^{max} ht_s, \tag{1.39}$$

όπου AT_{EC,e_k} λαμβάνεται από την τρέχουσα κατάσταση s_k,

- PC_s^{RU-E} είναι η κατανάλωση ισχύος του χρησιμοποιούμενου συνδέσμου FH:

$$PC_s^{RU-E} = \max\{ht_s - AT_{EC,e_k}(k), 0\}P_{net,e}^{fix} + \frac{b_{s,1}}{CB_{F,e}} P_{net}^{max} ht_s, \tag{1.40}$$

- PC_s^{E-R} είναι η κατανάλωση ισχύος του χρησιμοποιούμενου συνδέσμου MH:

$$PC_s^{E-R} = \mathbf{1}_{\{v_k < n_s - 1\}} \left[\max\{ht_s - AT_{MH,e_k}(k), 0\}P_{net,e}^{fix} + \frac{b_{s,v_k+1}}{CB_{M,e}} P_{net}^{max} ht_s\right], \tag{1.41}$$

όπου $\mathbf{1}_{\{v_k < n_s - 1\}} = 1$ αν $v_k < n_s - 1$ και 0 διαφορετικά (το slice τοποθετείται ολόκληρο στο EC, χωρίς χρήση MH συνδέσμου). Οι παραπάνω εξισώσεις για το κόστος κατανάλωσης ισχύος βασίζονται στο μοντέλο αποδοτικής τοποθέτησης VNF από [19].

Το κόστος αδράνειας ισχύος του EC ανατίθεται στο πρώτο slice που τοποθετείται στο EC ή σε αυτό που επεκτείνει τον ενεργό χρόνο λειτουργίας του, συνεπώς κι οι τιμές AT_x.

Ο πράκτορας πρέπει να ενθαρρύνει ενέργειες που θα οδηγήσουν σε υψηλότερη ανταμοιβή, ενώ παράλληλα να επιδιώκει χαμηλότερο κόστος κατανάλωσης ισχύος. Έτσι, η συνάρτηση ανταμοιβής για κάθε κατάσταση s_k, λαμβάνοντας ενέργεια a_k για το slice s διατυπώνεται ως εξής:

$$R(s_k, a_k) = \begin{cases} ReV_s - PC_s, & \text{αν το slice } s \text{ γίνει αποδεκτό στο βήμα } k, \\ 0, & \text{αν το slice } s \text{ απορριφθεί στο βήμα } k, \end{cases} \tag{1.42}$$

όπου $PC_s = PC_s^{EC} + PC_s^{RU-E} + PC_s^{E-R}$.

1.4.4.4 Συνάρτηση Μετάβασης

Μια μετάβαση πραγματοποιείται μετά από κάθε ενέργεια a_k για να τροποποιήσει την κατάσταση του δικτύου σύμφωνα με την ενέργεια. Οι μεταβάσεις ορίζονται ανά βήμα απόφασης RL για κάθε αίτημα slice και όχι ανά χρονικό διάστημα. Συνεπώς, πολλαπλές ενημερώσεις μπορεί να συμβούν εντός ενός χρονικού διαστήματος του ορίζοντα ελέγχου, ανάλογα με τον αριθμό των αιτημάτων slices που φτάνουν στο αντίστοιχο χρονικό διάστημα. Μετά από κάθε απόφαση, η κατάσταση του δικτύου — που περιλαμβάνει τις χωρητικότητες των κόμβων και των συνδέσμων — ενημερώνεται ώστε να αντικατοπτρίζει την αποδοχή του slice, αν αυτή έχει γίνει, και να προσαρμόζει τους χρόνους διατήρησης όλων των ενεργών slices.

1.4.5 Λύση DRL

Η προτεινόμενη λύση για το πρόβλημα της διαδικτυακής αποδοχής slices και κατανομής λειτουργιών, όπως ορίζεται από το παραπάνω MDP, είναι ο αλγόριθμος PPO, όπως αναλυτικά περιγράφεται στην παράγραφο 1.3.8.1.1. Ο αλγόριθμος αυτός ταιριάζει με τον διακριτό χώρο ενεργειών του προβλήματος. Χρησιμοποιείται μια τροποποιημένη παραλλαγή του PPO, γνωστή ως Maskable PPO, όπως υλοποιείται στη βιβλιοθήκη Python Stable Baselines3 [20]. Αυτή η παραλλαγή ενσωματώνει την τεχνική του action masking για να εξαιρεί τις ενέργειες που παραβιάζουν περιορισμούς σε κάθε χρονικό βήμα, απλοποιώντας έτσι τη διαδικασία λήψης αποφάσεων για τον agent. Οι ενέργειες θεωρούνται παραβιαστικές αν δεν ικανοποιούν τους περιορισμούς χωρητικότητας και καθυστέρησης.

1.5 Πειραματικά Αποτελέσματα

Σε αυτό το σημείο, οι δυνατότητες του αναπτυγμένου πράκτορα θα παρουσιαστούν μέσω μιας σειράς πειραμάτων, αναλύοντας τη συμπεριφορά του μοντέλου κατά τη διάρκεια της εκπαίδευσης και αξιολογώντας την απόδοσή του σε πολλές μετρικές.

1.5.1 Μετρικές Αξιολόγησης

Διαφορετικές Μετρικές Αξιολόγησης ενσωματώνονται για να εκτιμήσουν την απόδοση του μοντέλου, καθεμία από τις οποίες παρέχει μια διαφορετική οπτική στις δυνατότητές του. Αυτές είναι:

• **Αντικειμενική Αξία:** Αντιπροσωπεύει το συνολικό κέρδος που λαμβάνει ο πράκτορας σε κάθε βήμα. Ορίζεται από τα συνολικά έσοδα που προκύπτουν από την αποδοχή του slice (εξίσωση 1.46), μειωμένα από την κατανάλωση ισχύος που προκαλείται στην επακόλουθη κατάσταση του δικτύου (εξισώσεις 1.49-1.51). Αυτή η μετρική αντικατοπτρίζει την ισορροπία μεταξύ της ελαχιστοποίησης της κατανάλωσης ισχύος και της μεγιστοποίησης της συνολικής ανταμοιβής, είτε αυξάνοντας τον λόγο αποδοχής είτε δίνοντας προτεραιότητα σε αιτήματα slices με υψηλότερη προτεραιότητα.

• **Λόγος Αποδοχής:** Μετρά τον λόγο των αποδεκτών slices, μέχρι ένα συγκεκριμένο βήμα χρόνου, προς το υποσύνολο των ενεργών slices, το οποίο περιλαμβάνει τα αιτήματα που είναι σε εξέλιξη και δεν έχουν λήξει. Αντικατοπτρίζει την αποτελεσματικότητα του συστήματος στην αντιμετώπιση των εισερχόμενων απαιτήσεων των slices εν μέσω των υφιστάμενων δεσμεύσεων.

• **Αποδοτικότητα Ισχύος:** Είναι ο λόγος του αθροίσματος των αποδεκτών slices προς τη συνολική κατανάλωση ισχύος των υποδομών υπολογιστικού και δικτυακού εξοπλισμού του υποκείμενου δικτύου. Παρέχει μια εικόνα του ποσοστού της συνολικής κατανάλωσης ισχύος που αντιστοιχεί σε κάθε αποδεκτό slice.

Για την παρουσίαση των αποτελεσμάτων, χρησιμοποιείται η σωρευτική μέση τιμή αυτών των μετρικών. Υπολογίζεται με την μέση τιμή των τιμών κατά τη διάρκεια των επεισοδίων δοκιμών για κάθε βήμα χρόνου, ακολουθούμενη από τη σωρευτική άθροιση και, στη συνέχεια, χρησιμοποιούνται αυτές οι σωρευτικές τιμές για τον υπολογισμό των αντίστοιχων λόγων.

1.5.2 Εκπαίδευση - Κανονική Λειτουργία

Στον πίνακα 1.4 παρατίθενται οι τιμές για τις παραμέτρους της υποδομής του δικτύου και των αιτημάτων slices. Εξετάζονται δύο τύποι slices με διαφορετικές απαιτήσεις, δηλαδή URLLC και eMBB.

Παράμετρος	**Τιμή**
Πλήθος ECs	3
Χωρητικότητα EC, RC	$64, 256$ πυρήνες
Χωρητικότητα συνδέσμου Fronthaul, καθυστέρηση	2Gbps, 4ms
Χωρητικότητα συνδέσμου Midhaul, καθυστέρηση	4Gbps, 8ms
γ	0.8
P^{max}, P^{max}_{net}	200 W
P^{fix}_{net}	160 W
Πλήθος Αιτημάτων	20
Μήκος αιτήματος SFC	8 VNFs
Απαιτούμενα CPU cores ανά VNF	$\in \{2, 4, 8\}$
Χρόνος κράτησης αιτήματος	$\min(U\{3, 6\}, 12 - t_s)$
Απαιτούμενο εύρος ζώνης για URLLC και eMBB	100, 200 Mbps
Απαιτούμενη καθυστέρηση για URLLC και eMBB	25, 50 ms
Κανονικοποιημένη προτεραιότητα URLLC και eMBB	3, 2.4 ανά time-slot
Ορίζοντας Βελτιστοποίησης H	12 χρονικά βήματα

Table 1.4: Παράμετροι Προσομοίωσης.

Αρχικά, η λειτουργία του πράκτορα θα παρουσιαστεί σε σενάριο όπου τα αιτήματα για slices παράγονται σύμφωνα με μια κανονική κατανομή κατά τη διάρκεια του ορίζοντα H, συγκεκριμένα $\mathcal{N}\left(\frac{H}{2} - 1, 0.9\right)$.

Ο πράκτορας εκπαιδεύεται για 3500 επεισόδια, και η πρόοδος της διαδικασίας εκπαίδευσης μπορεί να παρακολουθηθεί μέσω του TensorBoard. Όπως απεικονίζεται στο Σχήμα 1.4, το σφάλμα, το οποίο ορίζεται ως ο μέσος όρος του σφάλματος εκπαίδευσης και το σφάλμα αξίας των δικτύων του Δράστη και του Κριτή αντίστοιχα, μειώνεται με την πάροδο του χρόνου και σταθεροποιείται περίπου στο βήμα 90,000. Το βήμα αυτό αντιστοιχεί περίπου στο επεισόδιο 3000, μετά το οποίο η συμπεριφορά του πράκτορα φαίνεται να σταθεροποιείται.

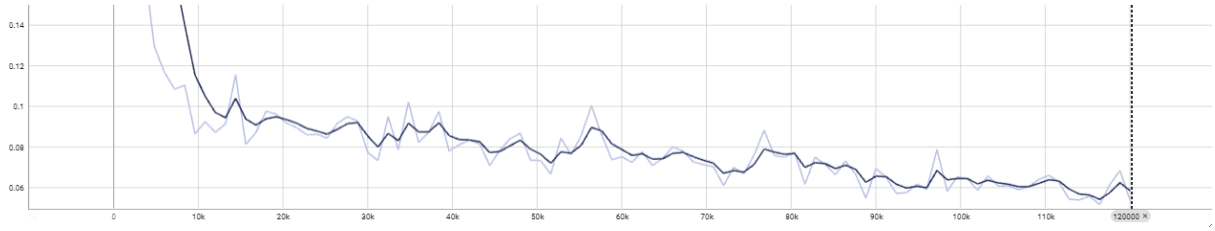
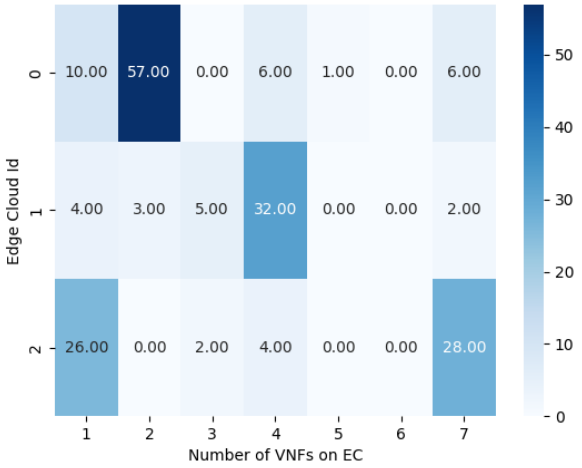


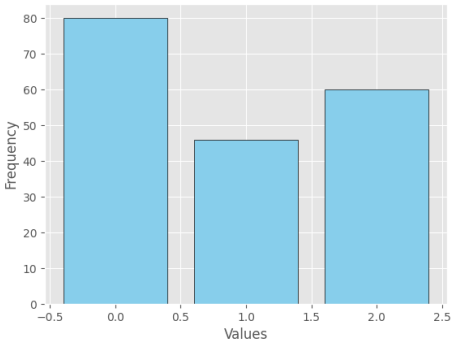
Figure 1.3: Σφάλμα κατά την εκπαίδευση.

Για την περαιτέρω κατανόηση της διαδικασίας λήψης αποφάσεων του πράκτορα, οι αποφάσεις που λαμβάνονται κατά τη διάρκεια μιας σειράς 10 επεισοδίων δοκιμών παρουσιάζονται: στην Εικόνα 1.5 (α) απεικονίζεται ένας θερμικός χάρτης με τη συχνότητα κάθε ζεύγους EC

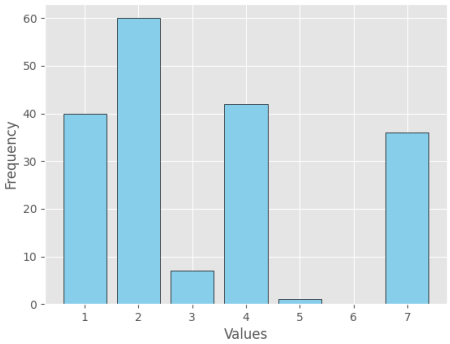
id-αριθμού VNFs, ενώ στις Εικόνες 1.5 (β-γ) η συχνότητα για κάθε επιμέρους απόφαση απεικονίζεται με μεγαλύτερη λεπτομέρεια.



(a) Θερμικός χάρτης με τιμές για κάθε ζεύγος αποφάσεων, EC, VNF.



(b) Απόφαση για EC.



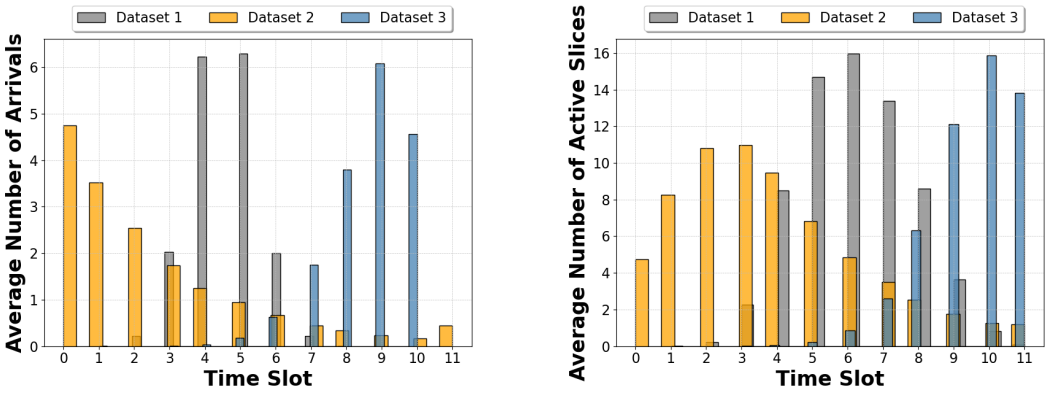
(c) Απόφαση για τον αριθμό των VNFs στο EC.

Figure 1.4: Αποφάσεις του πράκτορα.

Τα αποτελέσματα μπορούν πλέον να αναλυθούν. Καταρχάς, παρατηρείται μια ανισορροπία στην επιλογή των ECs, με προτίμηση στον πρώτο EC, κάτι που συμβαδίζει με τον στόχο της ελαχιστοποίησης της κατανάλωσης ισχύος, καθώς είναι πιο αποδοτικό να παραμένουν ανενεργά όσο το δυνατόν περισσότερα ECs, συγκεντρώνοντας τα αιτήματα στον ελάχιστο απαραίτητο αριθμό. Ο πράκτορας επίσης φαίνεται να προτιμά τη διάσπαση των slices σε μικρότερο αριθμό VNFs, γεγονός που δικαιολογείται από την προσπάθεια μείωσης της κατανάλωσης ισχύος, δεδομένου ότι τα VNFs που τοποθετούνται στο RC δεν προσθέτουν τόσο στον υπολογιστικό φόρτο, αλλά μόνο στην κατανάλωση της σύνδεσης Midhaul (MH) όταν η ενεργή περίοδός της επεκτείνεται. Επιπλέον, η επιλογή μη διάσπασης (τοποθέτηση και των 7 VNFs σε ένα μόνο EC) είναι συχνή. Αντιστοιχεί σε περιπτώσεις όπου το slice χωρά πλήρως στον επιλεγμένο EC και είτε έχει χαμηλό φορτίο και χρόνο διατήρησης, είτε το αίτημα γίνεται δεκτό κοντά στο τέλος του ορίζοντα ελέγχου, οπότε δεν αναμένονται επιπλέον αιτήματα. Σε

αυτές τις περιπτώσεις, ο πράκτορας αποφεύγει τη διάσπαση για να μειώσει την κατανάλωση ισχύος στη σύνδεση MH. Τέλος, οι αποφάσεις μη διάσπασης συνήθως σχετίζονται με τον τρίτο EC, υποδεικνύοντας προσπάθεια διατήρησης ικανού αποθηκευτικού χώρου στα άλλα ECs για τη συγκέντρωση των επερχόμενων προς διάσπαση slices, μειώνοντας εκ νέου το κόστος κατανάλωσης ισχύος της σύνδεσης MH. Συνολικά, ο πράκτορας παρουσιάζει την αναμενόμενη συμπεριφορά.

1.5.3 Αξιολόγηση υπό διαφορετικές κατανομές αφίξεων



(a) Μέσος αριθμός αφίξεων.

(b) Μέσος αριθμός ενεργών slices.

Figure 1.5: Πληροφορίες συνόλου δεδομένων.

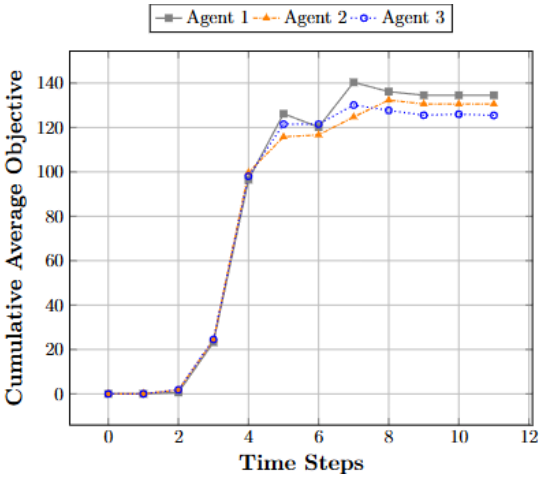
Στο πρώτο σύνολο πειραμάτων δημιουργήθηκαν τρία σύνολα δεδομένων με διαφορετικούς ρυθμούς αφίξεων ανά χρονικό βήμα στον ορίζοντα ελέγχου. Τα μοτίβα αφίξεων παράγονται σύμφωνα με τρεις διακριτές κατανομές — Κανονική, Εκθετική και Beta — όπως απεικονίζονται στην Εικόνα 1.5. Για κάθε κατανομή, χρησιμοποιήθηκαν 1000 σενάρια για εκπαίδευση και 10 για δοκιμή. Σκοπός του πειράματος είναι η αξιολόγηση της ικανότητας μάθησης του πράκτορα σε διαφορετικά μοτίβα αφίξεων.

Ο πράκτορας εκπαιδεύτηκε σε κάθε σύνολο ξεχωριστά (Πράκτορας 1 στο Dataset 1, Πράκτορας 2 στο Dataset 2, Πράκτορας 3 στο Dataset 3) και στη συνέχεια αξιολογήθηκε σε όλα τα τρία σύνολα, όπως φαίνεται στην Εικόνα 1.6. Ως μετρική αξιολόγησης χρησιμοποιήθηκε η Αντικειμενική Αξία, όπως ορίζεται στην ενότητα 1.5.1.

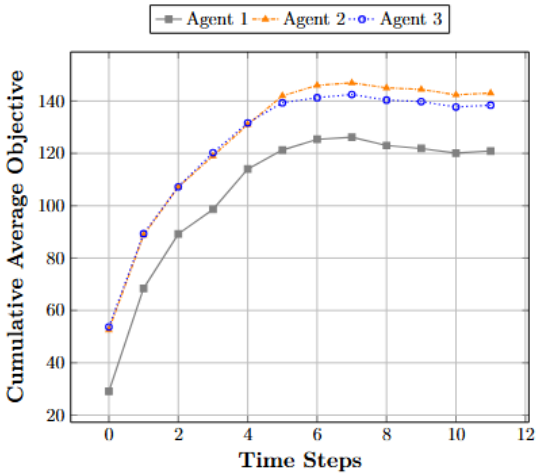
Όπως φαίνεται στην Εικόνα 1.7, ο πράκτορας αποδίδει καλύτερα στο dataset στο οποίο εκπαιδεύτηκε, αποδεικνύοντας την ικανότητά του να μαθαίνει και να βελτιστοποιεί τις αποφάσεις του ανάλογα με το μοτίβο αφίξεων. Ωστόσο, οι αποκλίσεις μεταξύ των αποδόσεων σε διαφορετικά datasets είναι μικρές, υποδεικνύοντας ισχυρή ικανότητα γενίκευσης.

1.5.4 Σύγκριση με πράκτορα στατικής διάσπασης

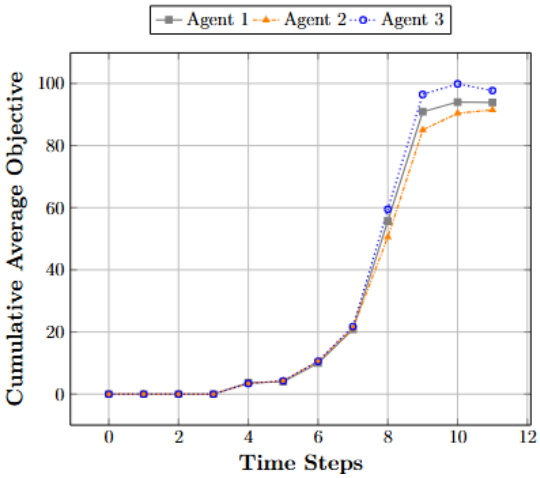
Το δεύτερο σύνολο πειραμάτων έχει σχεδιαστεί για να αξιολογήσει την ικανότητα του πράκτορα να λαμβάνει βέλτιστες αποφάσεις διάσπασης. Για τον σκοπό αυτό, ο πράκτορας



(a) Σωρευτική μέση αντικειμενική αξία στο σύνολο δεδομένων δοκιμής 1.



(b) Σωρευτική μέση αντικειμενική αξία στο σύνολο δεδομένων δοκιμής 2.

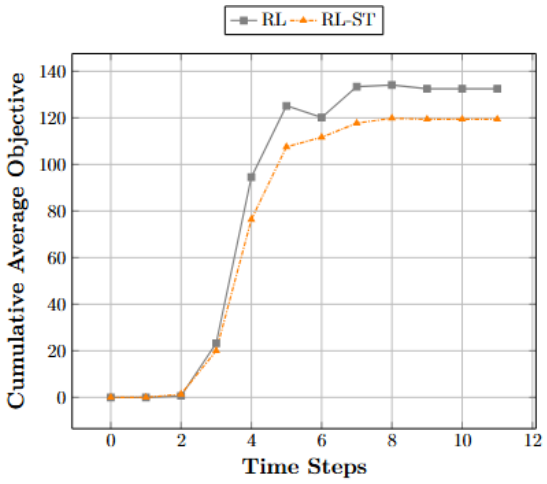


(c) Σωρευτική μέση αντικειμενική αξία στο σύνολο δεδομένων δοκιμής 3.

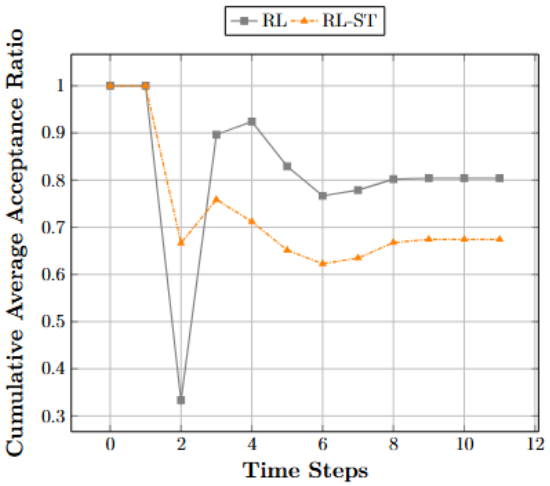
Figure 1.6: Συγκριτική αξιολόγηση υπό μεταβαλλόμενα μοτίβα άφιξης.

συγκρίνεται με μια στατική έκδοση, που αναφέρεται ως "RL-ST". Αυτή η έκδοση πραγματοποιεί τον διαχωρισμό των slices με ντετερμινιστικό τρόπο: συγκεκριμένα, τα eMBB slices χωρίζονται στο δεύτερο VNF (v_1), ενώ τα URLLC στην μέση (v_3). Η απόφαση του πράκτορα περιορίζεται έτσι στην επιλογή του EC. Και οι δύο πράκτορες εκπαιδεύονται και αξιολογούνται σε σύνολα δεδομένων που προκύπτουν από την κατανομή του Dataset 1.

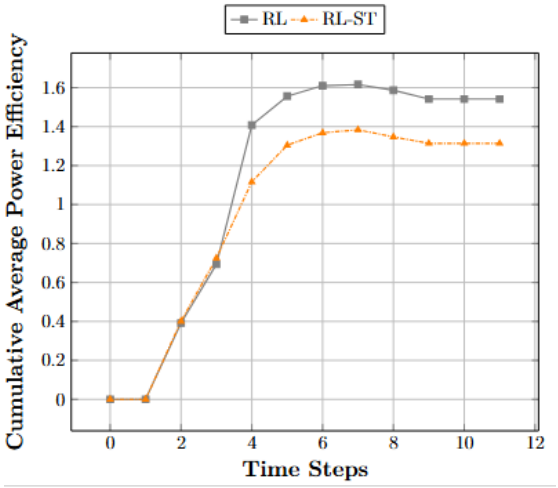
Η Εικόνα 1.7 παρουσιάζει την απόδοση των δύο μοντέλων, όπως αξιολογήθηκε με τις μετρικές που ορίζονται στην Ενότητα 1.5.1. Η μέθοδος "RL-ST", η οποία στερείται προσαρμοστικότητας, οδηγεί σε υποβέλτιστη διαχείριση φόρτου και αυξημένο αριθμό απορρίψεων, προκαλώντας σημαντική υποβάθμιση της απόδοσης σε όλες τις μετρικές που αξιολογήθηκαν. Το αποτέλεσμα αυτό αναδεικνύει τη ζωτική σημασία της δυναμικής λήψης αποφάσεων του προτεινόμενου μοντέλου RL για την αποτελεσματική διαχείριση των εισερχόμενων αιτημάτων. Το μοντέλο αυτό παρουσιάζει ισχυρή επίγνωση της κατάστασης (state awareness), επιτρέποντάς του να προσαρμόζεται σε μεταβαλλόμενες συνθήκες του δικτύου και να βελτιστοποιεί την απόδοση.



(a) Σωρευτική μέση αντικειμενική αξία.



(b) Σωρευτικός μέσος λόγος αποδοχής.



(c) Σωρευτική μέση αποδοτικότητα ισχύος.

Figure 1.7: Συγκριτική αξιολόγηση δυναμικού έναντι στατικού διαχωρισμού λειτουργιών.

1.6 Επίλογος

Η έλευση του 5G και, τελικά, του 6G έχει οδηγήσει σε αυξανόμενες απαιτήσεις πόρων δικτύου από πολλούς οργανισμούς ή άτομα, με ετερογενείς απαιτήσεις για τις υπηρεσίες που ζητούνται. Η εικονικοποίηση των λειτουργιών του δικτύου και η εξέλιξη του τμήματος RAN του δικτύου στην αποσυναρμολογημένη αρχιτεκτονική O-RAN επιτρέπουν την ενσωμάτωση ευφυών μηχανισμών για τη διαχείριση της κυκλοφορίας του δικτύου.

Η παρούσα διπλωματική εργασία παρουσιάζει μια καινοτόμο λύση στο πρόβλημα της δι-

αχείρισης των εισερχόμενων αιτημάτων slice σε ένα δίκτυο O-RAN χρησιμοποιώντας μια προσέγγιση Ενισχυτικής Μάθησης. Ο κύριος στόχος της εργασίας ήταν να αντιμετωπιστεί το πρόβλημα της δυναμικής λήψης αποφάσεων για ικανοποίηση αιτημάτων slices και για την τοποθέτηση Εικονικοποιημένων Λειτουργιών Δικτύου (VNF Placement) υπό μεταβαλλόμενες δικτυακές συνθήκες. Λαμβάνοντας υπόψη τη διαδικτυακή φύση του προβλήματος, η αντίστοιχη εξαντλητική λύση που θα προέκυπτε από διαδικασία βελτιστοποίησης θα είχε περιορισμένη χρησιμότητα. Η προτεινόμενη λύση αξιοποιεί τον αλγόριθμο Προσεγγιστικής Βελτιστοποίησης Πολιτικής για να εκπαιδεύσει έναν πράκτορα ενισχυτικής μάθησης, ο οποίος είναι ικανός να λαμβάνει δυναμικές αποφάσεις, λαμβάνοντας υπόψη την τρέχουσα κατάσταση του δικτύου ανά πάσα στιγμή. Ο πράκτορας στοχεύει στη βελτιστοποίηση της ισορροπίας μεταξύ των εσόδων από την αποδοχή των slices και του κόστους κατανάλωσης ισχύος στα edge clouds και στους επικοινωνιακούς συνδέσμους, που είναι κρίσιμα για τη βελτίωση της ενεργειακής αποδοτικότητας στα σύγχρονα ασύρματα δίκτυα.

Μέσω μιας σειράς πειραματικών αξιολογήσεων, η απόδοση του προτεινόμενου πράκτορα αξιολογήθηκε σε αρκετές σημαντικές μετρικές. Αυτά τα πειράματα ανέδειξαν τη σημασία της δυναμικής λήψης αποφάσεων για την επίτευξη αποτελεσματικής διαχείρισης των πόρων και την ικανότητα του πράκτορα να λαμβάνει εύστοχες αποφάσεις, προσαρμοζόμενος στις συνθήκες του δικτύου. Ο πράκτορας παρουσίασε αξιόπιστη απόδοση σε διαφορετικά σενάρια εισερχόμενης κίνησης, αποδεικνύοντας την ικανότητα του μοντέλου να ικανοποιεί έναν υψηλό αριθμό εισερχόμενων αιτημάτων, ενώ ταυτόχρονα διατηρεί χαμηλά κόστη κατανάλωσης ισχύος.

Μελλοντική Επέκταση

Παρόλο που η παρούσα διπλωματική εργασία παρέχει μια ισχυρή βάση για το προαναφερθέν πρόβλημα, υπάρχουν αρκετές πιθανές επεκτάσεις για μελλοντική επέκταση. Πρώτον, είναι κρίσιμο να αξιολογηθεί ο πράκτορας με ένα πραγματικό σύνολο δεδομένων αιτημάτων slices και VNFs. Ένα τέτοιο σύνολο δεδομένων, το οποίο θα περιλαμβάνει τα απαιτούμενα χαρακτηριστικά χωρητικότητας, εύρους ζώνης και καθυστερήσεων για κάθε VNF εντός κάθε slice, δεν έχει αναπτυχθεί ακόμη.

Δεύτερον, η δοκιμή του πράκτορα σε πραγματικά, μεγάλης κλίμακας περιβάλλοντα με περισσότερα RUs, ECs και RCs θα ήταν ιδιαίτερα ωφέλιμη. Επιπλέον, η αφαίρεση της υπόθεσης του ενός DU ανά EC, που έγινε στην παρούσα εργασία, θα παρείχε μια πιο ρεαλιστική αξιολόγηση της απόδοσης του πράκτορα.

Τέλος, αυτή η εργασία θα μπορούσε να βελτιωθεί περαιτέρω με την ενσωμάτωση ενός μηχανισμού για την επανατοποθέτηση slices, όπου ήδη δεκτά slices θα μετακινούνται δυναμικά μεταξύ των κόμβων του δικτύου για τη μείωση της κατανάλωσης ισχύος.

Συνοψίζοντας, καθώς τα δίκτυα συνεχίζουν να αυξάνονται σε πολυπλοκότητα και κλίμακα, λύσεις όπως αυτή που προτείνεται στην παρούσα διπλωματική εργασία θα διαδραματίσουν κρίσιμο ρόλο στην εξασφάλιση της αποδοτικής χρήσης των πόρων, μειώνοντας ταυτόχρονα τον περιβαλλοντικό αντίκτυπο μέσω της εξοικονόμησης ισχύος.

Chapter 2

Introduction

The evolution of mobile networks from 5G to 6G, coupled with the growing demands for higher data rates, enhanced capacity and support for multimedia and augmented reality (AR) services, necessitates the development of advanced mechanisms to efficiently manage network traffic and minimize power consumption. Network Slicing is a pivotal technology that partitions the network into multiple virtual subnetworks, referred to as "slices", each tailored to meet the diverse needs of specific applications, service providers, and individual users. Each slice operates as a distinct, isolated virtual network, with its own dedicated resources, policies, and performance characteristics, such as low latency (Ultra Reliable Low-Latency Communication - URLLC slices) or enhanced mobile broadband (Enhanced Mobile Broadband - eMBB). This segmentation of the network is essential to optimize the allocation of resources and ensure the efficient operation of diverse services in parallel. In this thesis, the network slices are modeled using the Service Function Chain (SFC) approach, representing them as an ordered sequence of Virtual Network Functions (VNFs). VNFs are software-based network functions—such as firewalls, load balancers, or traffic optimizers—that run on virtualized hardware rather than dedicated physical devices. The SFC model defines the specific order in which these VNFs are executed within a slice to fulfill its service requirements.

An important technique in managing slices is function splitting, which involves distributing the VNFs of a slice across multiple network nodes. Function splitting allows parts of the service chain to be executed closer to the user to reduce latency, while other parts can be centralized to optimize computational resources and power consumption.

The focus of this work is on the RAN domain, specifically within the context of an O-RAN architecture, aiming to enhance openness and interoperability. As introduced by the O-RAN Alliance, the RAN component is now composed of disaggregated logical nodes, each defined by its position within the cellular network, namely Radio Unit (RU), Distributed Unit(DU) and Centralized Unit (CU) [1]. DUs are typically deployed at the network Edge, in close proximity to the RUs, to minimize user delays. CUs, on the other hand, are located within the core of the network and are characterized by a greater computational capacity compared to DUs. To deploy slices within the O-RAN architecture and perform Function Splitting, the SFC corresponding to each slice should essentially be split between the three nodes, in a way that optimizes the trade-off between power consumption and user delays with slice admittance ratio.

When formulated as an optimization problem, this challenge involves several binary variables, rendering it a complex, NP-hard problem. In this thesis, a Reinforcement Learning (RL) approach is employed to address the Network Slicing problem, offering a dynamic, adaptive solution to slice admission and function splitting. The developed agent learns to make decisions by interacting with the network environment, adjusting its policies to optimize the deployment of network slices. The agent is trained using the Proximal Policy Optimization (PPO) algorithm, with the objective of balancing the revenues from slice acceptance against the power consumption costs associated with edge clouds and communication links. Through a series of experiments, the performance of the agent is evaluated using various metrics. The agent's state-awareness, generalization capabilities, and the significance of dynamic decision-making are thoroughly examined, providing insights into the effectiveness of RL-based solutions for complex network management tasks.

Chapter 3

Foundations of RAN and O-RAN Architectures

3.1 Radio Access Network(RAN) Overview

A Radio Access Network (RAN) is a critical component of a wireless telecommunications system, serving as the bridge that connects user equipment (UE), such as cellphones, computers, or remotely operated machines, to the broader network via a radio link. The RAN facilitates communication between user devices and the mobile carrier's core network, which oversees key functions like managing subscriber data, location tracking, and more. It includes both physical infrastructure, such as base stations (cell towers or small cells) and logical components, which ensure efficient management and optimization of data, voice, and control signal transmissions. While RAN technology has evolved significantly, from 1G to 5G, its fundamental purpose remains the same: enabling user devices to access the network. While mobile networks evolve over the years, RAN architecture is redesigned to meet the increasing demands of high capacity, massive connectivity, reduced costs, and energy efficiency, and to realize communication with ultra-low latency and ultra-high reliability.

Despite advancements in standardization and interoperability, the RAN is often the last remaining segment of the network that is largely proprietary, meaning it is still dominated by vendor-specific solutions rather than open standards. This proprietary nature is a key focus of industry discussions, particularly as newer concepts like Open RAN (O-RAN) aim to make this segment more accessible and interoperable, revolutionizing how mobile networks are designed, deployed, and managed, with a strong emphasis on flexibility, scalability, cost and efficiency [2].

3.1.1 Cellular Network

The Radio Access Network (RAN) constitutes the radio component of a cellular network. A cellular network is organized into geographical areas known as cells, each served by one or more fixed-location radio transceivers, commonly referred to as cell sites or base stations. While a single transceiver can cover a cell, the typical setup involves three

transceivers per cell site to enhance coverage and efficiency. These base stations are responsible for providing network coverage within each cell, enabling the transmission of voice, data, internet access, and other types of content via radio waves. When multiple cells are interconnected, they collectively form a wide-area radio coverage that allows seamless communication across a large geographical region. To prevent interference and ensure reliable service, each cell operates on a unique set of frequencies that differ from those of neighboring cells. This technique, known as frequency reuse, guarantees sufficient bandwidth and avoids signal overlap, even when multiple cells are in proximity. The cellular network design supports mobility, enabling devices to maintain continuous communication while transitioning between cells during active transmission. This is achieved through coordination between base stations, ensuring that connections are handed over seamlessly from one cell to another without interruption. As a result, the network facilitates real-time communication between mobile devices, fixed transceivers, and other network endpoints, regardless of device movement [21].

3.1.2 RAN Components

Understanding the components of RAN provides valuable insight into its architecture and how it manages the wireless connection between User Equipment (UE) and the Core Network. These components can be broadly categorized based on their specific roles:

• User Equipment: User equipment refers to any device equipped with mobile broadband modems capable of transmitting and receiving wireless signals. Examples include smartphones, tablets, laptops, and IoT (Internet of Things) devices. These devices serve as the endpoints of the network, interacting with the RAN to send and receive data, voice, or control signals.

• Antennas: Antennas are the physical layer components of the RAN infrastructure, typically located at or near the base station. They act as the first interface between user equipment and the RAN, transmitting and receiving electromagnetic signals over the air. Antennas started as passive components, meaning they only handled the electromagnetic interface, while the associated base station performed the active tasks of signal processing, but evolved into active systems, integrating radio frequency (RF) components like amplifiers and filters (MIMO/Beamforming antennas).

• Base Stations: Base stations play a central role in connecting user equipment to the core network. They process the raw signals captured by the antennas and convert them into a format that can be transmitted over the network.

• Transport Network: The Transport Network is responsible for carrying user data, signaling, and control information between different network components. It connects the Radio Access Network (RAN), the Core Network, and other critical network elements, ensuring efficient and reliable data transmission.

3.1.3 RAN Evolution

The evolution of the Radio Access Network (RAN) reflects the continuous advancements in mobile communication technology, spanning from the first generation (1G) to the fifth generation (5G). Each successive generation has introduced innovative architectural paradigms, significant hardware improvements, and enhanced software functionalities, all designed to achieve substantial performance gains. These advancements include increased data transmission speeds, enhanced reliability, improved scalability, and greater energy efficiency. These rapid changes of RAN technologies are driven by the escalating capacity requirements resulting from the exponential growth in global data traffic over the past decades.

3.1.3.1 1G: Analog RAN

The first generation of cellular networks (1G) utilized analog radio technology.

3.1.3.2 2G: DRAN

With the advent of 2G, mobile networks transitioned to digital wireless technology. The RAN architecture in 2G adopted a Distributed RAN (DRAN) approach, where Base Stations (BSs) were distributed across the coverage area, each serving a specific cell. All RAN components were co-located at each base station site, which handled its own processing and control functions. The architecture was built around:

• Base Transceiver Stations (BTS): Managed the wireless communication channels for voice and limited data and the allocation of radio resources. Along with other components, BTS contained the antenna system and transceivers (TRX), which converted the digital signals (mainly voice or data) into radio frequency (RF) signals and vice versa.

• Base Station Controllers (BSC): Controlled multiple BTSs, handling tasks such as call setup, frequency allocation, and mobility management. The BSC was connected to the Mobile Switching Center (MSC) in the Core Network, which managed call routing, handovers, and other network services.

Since 2G networks were primarily voice-centric, the transport network was circuit-switched, requiring a dedicated physical connection for the duration of a communication session [3].

3.1.3.3 3G

The transition to 3G introduced significant architectural changes, driven by the growing demand for higher data rates, improved capacity and support for multimedia services. The NodeB, which replaced the BTS as the base station, was responsible for providing the radio interface that allowed UE to connect to the network. However, unlike the 2G architecture, most control and higher-level management functions, like radio-resource and handover management, were offloaded to an external Radio Network Controller (RNC), which replaced the BSC. The RNC acted as an intermediary between multiple NodeBs

and the core network, handling resource allocation, mobility management, and handover coordination. This structure was an extension of the DRAN architecture [4].

With the increasing dominance of data over voice, 3G networks adopted a hybrid transport network architecture that combined circuit-switched technology for voice with packet-switched technology for data. Packet switching allowed for more efficient use of network resources, enabling dynamic data transmission and supporting services such as mobile internet and video streaming [4].

3.1.3.4 4G/ LTE: DRAN to CRAN

The transition to 4G/LTE (Long-Term Evolution) marked a significant advancement in mobile network architecture to meet the exponential growth of data-driven applications. The demands for higher data rates, lower latency, and improved reliability revealed the limitations of the traditional Distributed RAN with its localized, cell-based approach.

A more advanced version of the 3G NodeB, known as the eNodeB (evolved NodeB), was introduced. It combines the functions of the traditional NodeB and the Radio Network Controller (RNC) into a single, integrated entity. This reduced latency and made the architecture more efficient. The architecture of the eNodeB is disaggregated into two key components: (a)the Remote Radio Heads/Units (RRHs/ RRUs), which handle radio frequency (RF) tasks, including amplification, filtering, transmission and reception of signals to/from the antennas, and (b) Baseband Units (BBUs), responsible for baseband signal processing, including tasks like modulation/ demodulation, coding/ decoding. They serve as the interface between the core/backhaul and the radios in the RAN [4]. To deliver an optimal subscriber experience, eNBs utilize the X2 interface for intercommunication and coordination, particularly in managing resource allocation. This interface plays a pivotal role in facilitating seamless handovers, mitigating interference, and optimizing resource distribution among neighboring eNBs. Despite its critical functionality, the reliance on the X2 interface has led to challenges related to vendor lock-in. Different RAN vendors often developed proprietary implementations of the X2 interface, creating interoperability barriers and making it challenging for Mobile Network Operators to have multiple RAN vendors in a single location [22].

As traffic demand increased, the amount of radio equipment required to serve a coverage area increased and evolved into a model where the BBUs are all concentrated in a single location (e.g. data center), referred to as BBU Hotel, to serve multiple antennas in a certain region. They are connected to the RRHs at the cell sites via fronthaul links and can be shared among multiple cell sites. This format is also known as CRAN (Centralised RAN) and it aims to reduce installation and maintenance costs while operating more effciently, enabling the processing of a higher volume of traffc, as well as increasing network security and facilitating system expansion whenever demand requires it [4].

At the same time, to increase spectral efficiency and support higher data rates, MIMO technology (Multiple Input, Multiple Output) became a core component of 4G. MIMO enables the simultaneous transmission of multiple data streams using multiple antennas at both the transmitter and the receiver. This improves capacity and throughput without requiring additional spectrum.

With 4G/LTE, the transport network fully transitioned to a packet-switched architecture, aligning with the data-centric nature of the network.

Overall, while CRAN architectures and 4G technologies enhanced network efficiency, the underlying hardware and software remained largely vendor-specific. Many vendors used proprietary interfaces between BBUs and RRHs, coupled with custom implementations of front-haul protocols. Consequently, critical components such as BBUs, radios, and antennas often had to be procured from a single vendor to guarantee compatibility and seamless integration, thereby limiting the flexibility and interoperability of the network infrastructure.

3.1.3.5 5G: C-RAN

The introduction of 5G brought significant advancements to RAN architecture, in order to meet the demands of ultra-high-speed data, low latency, and massive device connectivity. In this context, cutting-edge radio technologies are utilized. These include Massive MIMO, which employs a large number of antennas to enable simultaneous communication with multiple users, enhancing capacity, and Beamforming, which leverages multiple antennas to control the direction of the transmitted or received signals, focusing the signal on specific users instead of broadcasting in all directions.

The 5G Base Station, referred to as gNodeB (Next Generation NodeB), unlike its predecessors, adopts a disaggregated architecture, meaning its components can be be physically or logically separated. This provides greater flexibility for resource allocation to individual components and enhances resource utilization and management. Specifically, the gNodeB is divided into the following components:

• Radio Unit (RU): Manages the RF (Radio Frequency) layer and handles tasks such as the conversion of signals between analog and digital formats. It is a hardware component located at cell site, close to the antenna or integrated with it. The RU is considered the radio portion of the RAN and is managed by the Distributed Units (DUs).

• Distributed Unit (DU): A mixture of software-based or physical technologies that handle real-time tasks, that require low latency like signal processing, scheduling, and radio resource management. They should be at edge data centers, not be far from the RUs. The normal distance is between 1 km and 20 km, interconnected by fiber optic. Several RUs may be connected to the DUs.

• Centralized Unit (CU): Software-based units hosted in a cloud-based platform or regional data center that handle non-real-time tasks, like mobility management and higher-level protocols. They can be centralized in a way that allows them to oversee multiple DUs.

Technically, the BBU functions of 4G are disaggregated into the DU and the CU, which are distinct and geographically separated units. This separation was implemented to ensure that processing tasks associated with different protocol layers are performed in physically distinct units. The reason for this design lies in the differing timing requirements of these functions. The DU is responsible for executing time-critical functions that require real-time processing, making its proximity to the RUs essential for maintaining low latency. In contrast, the CU handles non-time-critical functions, allowing it to be centralized and positioned farther from the RUs. This centralization facilitates resource pooling, enabling dynamic resource sharing across multiple DUs, thereby improving overall network efficiency

and scalability [4].

On the other hand, the separation of DUs and RUs in the 5G architecture is driven by three key factors: (1) cost reduction: Simplifying the RUs by offloading complex processing to the DUs reduces their intelligence and, consequently, their cost; (2) enhanced Management of MIMO/massive MIMO and Beamforming: centralized management of multiple RUs by a DU enables effective coordination of antennas, optimizing the use of spatial diversity and mitigating interference between systems. This coordination improves coverage and allows the DU to dynamically select the most suitable antenna to serve a user; (3) resource sharing: a single DU can allocate its computational resources across multiple RUs, enhancing efficiency and flexibility in resource utilization [4].

Overall, distributing processing functions among the three RAN units—CU, DU, and RU—not only reduces transport bandwidth and latency demands at each unit but also optimizes resource utilization and management across the network.

The 5G split architecture requires an additional transport network solution, dividing it into three distinct segments, reflecting how data flows between different components of the RAN: Fronthaul (FH): connects the RU to the DU, Midhaul (MH): connects the DU to the CU, and Backhaul (BH): connects the CU to the Core Network.

In the 5G, the architecture transitions to Cloud RAN, where baseband processing becomes software-defined, allowing baseband functions to be centralized in data centers or cloud-like environments. This cloud-native, software-driven approach facilitates dynamic resource allocation, enabling the network to adapt to fluctuating traffic demands, while delivering a highly scalable, flexible, and cost-efficient solution [4]. Despite this shift, the hardware executing these functions typically remains specialized telecom equipment. Additionally, even though the interface between RUs and DUs or CUs in existing RAN systems is based on partially standardized specifications, the practical implementation of these interfaces often includes proprietary variations, resulting in limited multivendor interoperability. In other words, if an operator employs a radio unit from Vendor A, it generally must also use the baseband equipment from the same vendor. Additionally, the software designed for the baseband hardware is typically incompatible with hardware from other vendors. This reliance on proprietary, vendor-specific implementations of interface specifications creates a significant barrier to interoperability and reinforces vendor lock-in [9].

3.2 Key Technologies

3.2.1 Network Function Virtualisation (NFV)

The main objective of Network Function Virtualisation (NFV) is virtualizing network functions like proxies, load balancers, firewalls, routers or any other network function traditionally running in proprietary hardware appliances. These hardware-driven functions have evolved into software-based functions and no longer require special proprietary hardware to perform; they can instead be deployed on top of Commercial Off-The-Shelf servers (COTS) on a cloud-based platform, without the need for installation of a new entity or hardware device. All hardware resources of these servers such as computing, storage and networking devices are monitored as a common resource pool. These Virtual Network

Functions (VNFs) can be migrated or instantiated in various locations in the network based on the functionality. Additionally, appropriate VNFs can be efficiently chained such that service-oriented functions/policies can be enforced on the traffic from a specific service/application, giving rise to a service function chain (SFC). For example, traffic from a video streaming service may pass through VNFs for firewall (to ensure security), traffic optimizer (to adjust video quality based on network conditions), and load balancer (to distribute traffic across servers).

Virtualization technologies allowed for dynamic resource allocation, agile and cost-effective network management (applications can be upgraded easily or swapped altogether), efficient scaling of network functions, and the introduction of new services without the need for extensive hardware upgrades [6]. By virtualizing network functions, NFV also reduces power consumption and lowers the operational and maintenance costs for MNOs. However, the deployment and management of VNFs introduce new complexities, particularly in terms of manageability and security. These challenges make the deployment of VNF instances non-trivial, requiring robust solutions to ensure efficient operation and protection against potential vulnerabilities [7].

3.2.2 Functional Splits

3.2.2.1 RAN Protocol Stack

The RAN protocol stack refers to the set of protocols that govern the operation and communication between different components of the RAN. It defines how data is processed, transmitted, and managed in the RAN from the physical layer all the way up to the application layer. The RAN Protocol Stack in 5G typically consists of the following layers:

• Physical Layer (PHY)
The lowest layer, responsible for the actual transmission and reception of radio signals over the air interface. It handles tasks like modulation, demodulation, coding, and decoding of signals, as well as error correction. In modern architectures it is divided into Low-PHY (handles tasks that are closely tied to the RF) and High-PHY(performs computationally intensive baseband processing tasks, including channel coding, modulation).

• Medium Access Control (MAC) Layer
The MAC layer is responsible for controlling access to the radio medium and scheduling the transmission of data. It handles tasks like resource allocation, scheduling, and multiplexing of data from different users over the same radio channel.

• Radio Link Control (RLC) Layer
The RLC layer is responsible for the segmentation and reassembly of data packets, as well as error correction and flow control. It ensures reliable transmission of data over the air interface by handling retransmissions in case of errors.

• Packet Data Convergence Protocol (PDCP) Layer
The PDCP layer is responsible for header compression (to reduce the size of data packets), ciphering (encryption of data), and packet sequence number management. It also handles

user plane data and control plane signaling, including the RRC (Radio Resource Control) protocol.

• Radio Resource Control (RRC) Layer
The RRC layer is responsible for managing the control plane signaling between the UE and the RAN. It handles tasks like connection establishment, handover management, and power control, while it also controls the configuration of the RAN.

• Service Data Adaptation Protocol (SDAP) Layer
The SDAP layer is responsible for managing the quality of service (QoS) for user plane data. It ensures that data flows are classified and mapped to appropriate QoS parameters, ensuring the required performance for each service (e.g., voice, video, or data). It resides between the RLC and PDCP layers within the user plane.

The PHY, MAC, and RLC layers are involved in tasks that are very delay-sensitive (of the order of a few milliseconds), while the higher layer PDCP and RRC protocols undertake tasks that are more delay tolerant and can take tens of milliseconds.

These layers can be further condensed to broader categories: the network layer (L3), which includes RRC layer, the data link layer (L2), which include PDCP, SDAP (for user plane), RLC and MAC layers, and the physical layer (L1), which includes the PHY layer.

3.2.2.2 Functional Splits

While moving lower in the protocol stack, both processing demands and transport overhead increase significantly [23]. In the context of 5G RAN, mobile traffic is growing at such a rapid pace that it is quickly surpassing the capabilities of conventional fronthaul interfaces. To address this limitation, one potential solution is to offload more processing functions to RUs, enabling them to process observed signals more extensively before forwarding them to the DU via the fronthaul network. However, critical questions remain regarding the optimal number of processing functions that should be localized at the RUs and to what extent this can alleviate the immense capacity demands placed on the fronthaul links [5].

To address these challenges, the 3rd Generation Partnership Project (3GPP) has defined various Functional Splits that determine how RAN functions are distributed across the RU, DU, and CU. These splits are categorized based on the layer of the protocol stack at which the functions are divided. The functional split will determine the allocation of processing functions between the RU and the DU. 3GPP standard consists of eight main Functional Split options, including suboptions, shown in Fig. 2.1.

Some examples in more detail:

• Option 8 (RF/PHY split): all the baseband (L1/L2/L3) signal processing functions are centralized at the DU, whereas only the RF functions are left in the RU site. Using this option, the maximum resource sharing can be obtained, but the required fronthaul data rate is the highest compared to other functional split options.

• Option 7.1 (low PHY split): only RF functions such as the iFFT/FFT and beamforming port expansion/reduction are left in the RU.

Figure 3.1: Signal processing functions in the protocol stack and different Functional Split options provided by 3GPP [5].

• Option 7.2 (low PHY/high PHY split): the RU additionally includes Low-PHY processing (resource element demapping, channel estimation, and diversity combiner functions in the uplink and precoding and the resource element mapping functions in the downlink), while DU handles High-PHY, MAC, and RLC functions.

• Option 7.3 (High PHY split): further includes in the RU modulation/demodulation, equalization, and inverse discrete Fourier transform functions in the uplink, and layer mapping in the downlink [5].

As the split progresses to higher layers, more processing functions are retained locally within the RU. This means more extensive signal processing performed before transmitting signals over the fronthaul network to the DU. This approach reduces the data rate and latency demands on the fronthaul link. However, increasing the processing responsibilities of the RU also raises its configuration complexity. This added complexity may restrict the deployment of simpler, low-cost, and energy-efficient RUs.

The decision on how to distribute the baseband processing functions depends on several key considerations, such as the Quality of Service (QoS) requirements of the supported services (e.g., low/high throughput, low/high latency, real-time/non-real-time applications) and the characteristics of the available transport network infrastructure (e.g., optical fiber, wireless networks, including microwave, millimeter waves, or free-space optics). Additionally, the Functional Split must accommodate specific load demands and user densities within a given geographical service area. The choice of Functional Split must be carefully considered, taking into account some cost-effective and technical trade-offs between the required fronthaul capacity, latency, and the degree of centralization of signal processing functions [5].

In modern RAN architectures, the full potential of Functional Splits is realized through dynamic splitting, where the placement of network functions is adaptively adjusted based on real-time feedback from the network. This approach enables the system to optimize performance in response to changing conditions, such as traffic load, latency requirements, and resource availability.

3.2.3 Network Slicing

Network slicing is a technology that facilitates the creation of multiple virtual networks, referred to as "slices" on top of a shared physical infrastructure. These slices can be implemented across both the core network and the RAN. In a virtualized network environment, the physical infrastructure of the 5G network serves as a secondary component, while the logical components—abstracted representations of the underlying physical infrastructure—become the primary focus. These logical components are designed to address specific requirements based on diverse use cases. This virtualization is possible by embedding techniques of Software Defined Networks (SDN) and NFV.

The sub-components of Network Slicing involve slice creation, which enables the development of 5G network slices; slice isolation, which isolates the different type of slices from one other, ensuring that issues in one slice (e.g., congestion or failures) do not affect others; and slice management, which manages the overall process from slice creation to slice delivery to the use case.

The concept of Network Slicing allows for the provision of smaller, dedicated network slices instead of offering the entire network's capabilities. Each slice operates as an independent virtual network with its own resources, policies, and performance characteristics, such as low latency, ultra-high bandwidth, or enhanced mobile broadband. These attributes enable slices to address a diverse range of use cases effectively. For instance, a low-latency slice is ideal for real-time streaming applications, while an ultra-high bandwidth slice is better suited for streaming high-definition videos. Each slice is customized with specific configurations for parameters such as bandwidth, latency, reliability, and security, ensuring it meets the unique requirements of the corresponding use case. Each use case is served by the slice that best suits its service requirements. Different use cases are:

- eMBB Slice: Handles high-bandwidth applications like video streaming, AR/VR, or file downloads.
- URLLC Slice: Supports ultra-low-latency and high-reliability applications like autonomous vehicles or industrial robotics.
- mMTC Slice: Manages IoT applications with massive device connectivity and low data rates, such as smart meters or environmental sensors.

It is evident that 5G networks, designed to accommodate a broad spectrum of applications with vastly differing requirements, rely on Network Slicing to deliver tailored services for each use case over a shared physical infrastructure. A critical advantage of Network Slicing is its inherent adaptability. Logical networks can dynamically reconfigure themselves in response to changing demands, reallocating resources to optimize performance. This dynamic flexibility enhances resource efficiency and scalability, enabling operators to support a diverse range of complex use cases effectively. As a result, network slicing has become a fundamental feature of modern 5G networks, ensuring they can meet the diverse and evolving needs of users and applications [8].

3.3 vRAN

Virtualized RAN refers to the application of NFV principles on the RAN. vRAN decouples the software from hardware by virtualizing (implementing as software modules) baseband processing components: baseband unit, gNB/eNodeB, DU and CU, which in a traditional RAN architecture were performed by hardware-specific elements (e.g. FPGAs, ASICs) [24].

With vRAN, processing functions can be centralized in a highly flexible manner, facilitating network scalability. This centralization provides operators the ability to leverage these pooled VNFs and dynamically allocate different resources through network slicing to create the architecture on-demand and fit the needs of the applications [9]. At the same time, centralization allows for enhanced visibility and control over network functions, enabling more efficient management. Advanced orchestration tools further streamline operations by automating the provisioning and configuration of network resources, thereby minimizing the need for manual intervention and reducing operational complexity.

The flexibility of vRAN also allows mobile operators to leverage different Functional Splits dynamically, tailoring configurations based on available network resources and specific operational needs. As each Functional Split has distinct fronthaul capacity and latency requirements, this adaptability enables operators to support a wide range of QoS configurations for various services, such as high data rates for broadband applications or low latency for real-time use cases. This ability to adjust Functional Splits flexibly empowers operators to respond effectively to varying traffic demands and user densities across different geographical regions. This ensures optimal resource allocation and consistent service quality [5].

Nonetheless, the primary drawback of vRAN is that it traditionally uses proprietary interfaces between radios and other network elements, thereby locking the operator into a single RAN vendor.

Table 3.1: Different Radio Access Network Generations.

Architecture	Baseband Hardware	Baseband Software	Radio Hardware (RRU)	BBU/RRU Interface	Interability
CRAN (Centralized)	Proprietary Technology	Proprietary Software	Proprietary Hardware	Proprietary Interface	Radio + BBU (HW+SW single vendor)
V-RAN	COTS	Proprietary Software	Proprietary Hardware	Proprietary Interface	Radio + BBU (SW single vendor)
O-RAN	COTS	Software Open Interface	COTS	Open Interface	Radio + BBU (HW+SW various vendors)

3.4 ORAN

Open Radio Access Network (O-RAN) is an industry-driven initiative and framework designed to revolutionize traditional RAN architectures by promoting openness, interoperability, and flexibility. It embraces the principles of virtualization, standardization, and openness to unlock the full potential of mobile networks. By redefining the conventional vendor-specific, monolithic RAN equipment, O-RAN disaggregates network components into interoperable units connected through open interfaces, leveraging open hardware and software solutions. This approach empowers network operators to mix and match components from multiple vendors, fostering competition, driving innovation, and reducing vendor lock-in [2].

The primary objective of the O-RAN architecture is to transition RAN networks toward a more open, intelligent, and adaptable design while maintaining compliance with 3GPP standards and extending their capabilities. This open and modular approach provides the flexibility needed to meet the diverse and evolving requirements of 5G applications.

3.4.1 Evolution

During the early 2000s, Nokia, Samsung, NEC, and LG collaborated to introduce the Open Base Station Architecture Initiative (OBSAI), which aimed to standardize communication between Radio and Baseband units, namely the fronthaul network. This initiative sought to establish a uniform set of rules and protocols, facilitating seamless interoperability between Remote Radio Units (RRUs) and Baseband Units (BBUs) from different manufacturers. By reducing reliance on proprietary solutions, OBSAI aimed to promote vendor-neutral systems and foster a more competitive ecosystem [25]. However, the adoption of OBSAI faced resistance from certain vendors, who formed a competing alliance to develop an alternative standard, the Common Public Radio Interface (CPRI). The coexistence of these competing standards fragmented the market, undermining the goal of universal interoperability. This represents one of the earliest significant attempts at standardizing Open RAN principles, although it ultimately failed to achieve widespread industry acceptance.

In the years that followed, Mobile Network Operators (MNOs) continued to advocate for interoperability within the cellular network infrastructure, promoting initiatives to standardize components of base stations. Despite these efforts, adoption remained limited, with proprietary solutions continuing to dominate the industry.

The advent and widespread adoption of LTE further consolidated the market, leading to a significant reduction in the number of vendors and resulting in market dominance by a few key players. This limited the choices available to operators, creating a landscape of vendor lock-in and reduced competition. In response, operators began seeking ways to disrupt these monopolies by "opening" the RAN. The goal was to break the reliance on proprietary systems, foster competition, and create opportunities for new vendors to enter the market. This drive for openness and flexibility laid the foundation for modern Open RAN initiatives, aiming to redefine the traditional RAN ecosystem and enable a more democratic, diverse, and competitive vendor landscape.

The first successful initiative to realize the vision of an open and interoperable RAN was

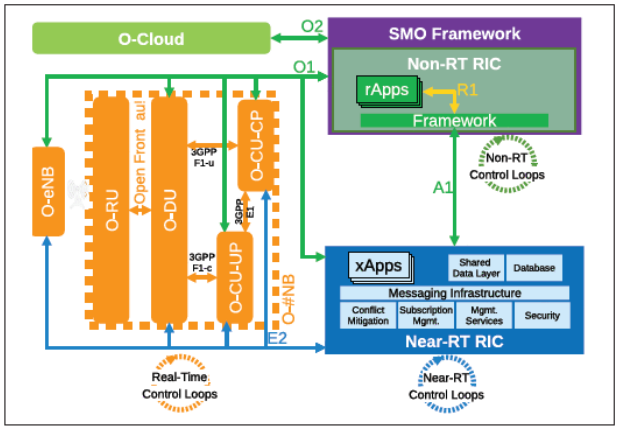


Figure 3.2: O-RAN Architecture Overview [1].

the O-RAN Alliance, established in 2018 by five leading operators: AT&T, China Mobile, Deutsche Telekom, NTT DOCOMO, and Orange. Open RAN (O-RAN) advocates for the adoption of open and standardized interfaces to eliminate vendor lock-in and foster a more competitive and dynamic market. By enabling seamless interoperability between cellular network equipment from multiple vendors, O-RAN leverages white-box servers and standardized hardware, moving away from the proprietary, custom-built equipment traditionally used in base stations [2].

If implemented successfully, O-RAN has the potential to drive a transformative wave of innovation in the RAN domain by significantly lowering market entry barriers for new players. This could lead to a more diverse and competitive vendor landscape, ultimately benefiting operators and end-users through improved efficiency, reduced costs, and accelerated technological advancements.

3.4.2 Architecture

Figure 3.2 illustrates the components and control loops within the O-RAN architecture. These control loops represent closed-loop autonomous action and feedback mechanisms designed to ensure normal operation and optimize network performance. The first loop is the Real-time O-DU Scheduler control loop, responsible for real-time processing tasks such as radio scheduling, beamforming, and other time-sensitive operations. It operates with a timing constraint of less than 10 milliseconds, qualifying it as a real-time control loop. The Near-real-time Control Loop operates on a time scale between 10 milliseconds and 1 second and is managed by the Near-Real-Time RAN Intelligent Controller (Near-RT RIC). Finally, the Non-real-time Control Loop is associated with the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) and operates on a time scale exceeding 1 second. It focuses on long-term network optimization and policy management, leveraging historical data and advanced analytics to enhance performance over time.

These control loops operate in parallel and are designed to function independently or interact, depending on the specific use case [11].

3.4.2.1 Components

• O-Cloud

The O-Cloud is a cloud computing platform composed of physical infrastructure nodes that adhere to O-RAN specifications, enabling the hosting of essential O-RAN functions such as Near-RT RIC, O-CU-CP, O-CU-UP, and O-DU, along with supporting software components like operating systems, virtual machine monitors, and container runtimes, as well as management and orchestration tools. This platform provides the foundational infrastructure for virtualized and containerized network functions required by O-RAN. It is built on COTS servers augmented with hardware accelerators, such as field-programmable gate arrays (FPGAs) and GPUs, and includes networking infrastructure that supports O-RAN software decoupled from hardware across various layers. By adhering to open and interoperable interfaces, the O-Cloud ensures compatibility with O-RAN components and uses hardware acceleration for computationally intensive tasks, such as fast Fourier transforms and forward error correction. Virtualized Network Functions (VNFs), including eNBs and gNBs, are deployed within the O-Cloud in virtualized RAN (vRAN) scenarios, enabling enhanced scalability and flexibility.

O-RAN can also be implemented through alternatives to the O-Cloud, such as Physical Network Functions (PNFs) or server chassis and racks in shared cloud environments. These alternatives provide varying levels of openness and flexibility, with the choice of deployment depending on the operator's specific needs and infrastructure capabilities. Among these options, the O-Cloud stands out as the most open and flexible solution [2].

• Service Management and Orchestration

The SMO framework serves as a unified platform for the management and orchestration of various RAN components, addressing both traditional network management tasks and those specific to O-RAN's open architecture. Its primary responsibilities include overseeing fault, configuration, accounting, performance, and security (FCAPS) functions for O-RAN network elements. It also handles long-term network planning and optimization of the RAN. Through the O2 interface, the SMO manages and orchestrates O-Cloud resources, performing tasks such as resource discovery, scaling, FCAPS operations, software management, and CRUD operations on O-Cloud resources.

• Non-RT RAN Intelligent Controller (Non-RT RIC)

The Non-RT RAN Intelligent Controller, integrated within the Service Management and Orchestration (SMO) framework, focuses on long-term RAN optimization and management tasks that operate over extended timescales, typically from seconds to minutes or longer. These tasks do not require immediate responses and include policy computation, machine learning model management (e.g. training), and radio resource management, within this temporal scope. The primary goal of Non-RT RIC is to support intelligent RAN optimization by providing policy-based guidance, model management, and enrichment information to the near-RT RIC function so that the RAN can be optimized. The Non-RT RIC collects data from the network, such as performance metrics and traffic patterns, and applies advanced algorithms, including ML models, to analyze this data. It generates actionable insights and optimization strategies, which are then communicated to the Near-Real-Time RIC for execution in real-time environments. Additionally, it inter-

acts with the SMO to ensure cohesive network-wide orchestration, contributing to unified management across the RAN ecosystem.

Modular third-party software applications, referred to as rApps, operate within the Non-RT RIC to provide value-added services such as policy enforcement through the O1 and O2 interfaces, drive of A1 interface, network slicing, and generation of additional contextual or operational information that is shared with other rApps or components of the O-RAN architecture to improve network performance. These rApps can be updated, added, or replaced without disrupting the system, allowing for flexible enhancements to RAN operations.

Operators develop and own the core algorithms of the Non-RT RIC, enabling customization of RAN behavior through models optimized for specific policies and objectives. Data management tasks initiated by the Non-RT RIC are converted into the O1/O2 interface for execution, while contextual and enrichment information is shared with the Near-RT RIC via the A1 interface to the near-RT RIC [2, 10].

• Near-RT RAN Intelligent Controller (Near-RT RIC)
The Near-RT RAN Intelligent Controller is a logical function designed to enable near-real-time optimization, control, and monitoring of O-CU and O-DU nodes within timescales of 10 milliseconds to 1 second. It operates based on policies (high-level objectives or rules) generated by the Non-RT RIC, which are translated into actionable directives for the RAN. The Near-RT RIC is responsible for a variety of critical tasks, including per-user equipment (UE) controlled load balancing, radio resource management, interference detection and mitigation, quality of service (QoS) management, connectivity management, and seamless handover control. The Near-RT RIC utilizes a Radio-Network Information Base (R-NIB) database, which captures the near-real-time state of the network and provides valuable data to train AI/ML models within the Non-RT RIC. These models are then used by the Near-RT RIC to optimize radio resource management for subscribers. Additionally, the Near-RT RIC supports third-party modular micro-service-based applications, called xApps, which run on the RIC platform and extend its functionality. These xApps can receive data and telemetry from the RAN and send back control, enabling custom logic to implement tailored QoS policies for specific applications, thus improving the overall performance and adaptability of the RAN [10].

• O-RU
The O-RU is directly connected to the antennas and is responsible for transmitting and receiving radio signals over the air. It interfaces with the O-DU through the fronthaul interface. The primary functions of the O-RU include RF signal processing, such as filtering, amplification, digital conversion, and beamforming [10].

• O-DU
The O-DU processes data and signaling above the low-level physical layer and interacts with the O-CU (Central Unit) for control and user plane tasks. It is responsible for functions such as segmentation, reassembly, and retransmission of data, as well as multiplexing of data packets and encoding/decoding [10].

• O-CU

In the O-RAN architecture, the O-CU (O-RAN Central Unit) is split into two logical nodes: O-CU-CP (Control Plane) and O-CU-UP (User Plane). The O-CU-CP manages signaling tasks such as connection setup, mobility management, and radio resource configuration, as well as control functions like header compression and ciphering for signaling messages. These tasks govern how the network interacts with user devices (UEs) and other network components. The O-CU-UP, on the other hand, handles the data traffic between the UE and the core network, processing user data packets, ensuring they are delivered to the appropriate destination, and implementing QoS policies. It also manages user plane-specific encryption and compression [11].

The decoupling of the O-CU into separate control and user planes offers several benefits. One key advantage is that it allows the user plane to become more standardized, as most of the variability lies in the control plane. This enables easier scaling and more cost-effective solutions for the user plane. Another benefit is the enhanced control functionality, which leads to increased efficiency and better radio resource management. These control functions leverage analytics and data-driven approaches, including advanced machine learning (ML) and artificial intelligence (AI) tools, to optimize network performance [11].

O-CU and O-DU are collectively referred to as E2 nodes, which are defined as any logical RAN function or entity that supports the E2 interface and interacts with the Near-RT RIC. One near-RT RIC may be connected through transport functions to one or multiple E2 nodes, although each E2 node may be connected to a single near-RT RIC.

3.4.2.2 Services and Interfaces

The O-RAN Alliance has introduced technical specifications that define open interfaces connecting various components of the O-RAN architecture. Standardizing these interfaces is crucial for breaking vendor lock-in within the RAN. For example, it allows a near-RT RIC from one vendor to interact with base stations from another vendor, or enables interoperability between CUs, DUs, and RUs from different manufacturers. These open interfaces facilitate the deployment of the O-RAN architecture across different network locations, such as the cloud, edge, or cell sites, and support multiple configurations. This flexibility allows operators to select the most suitable setup for their specific needs, fostering a more open and competitive market.

• O1 Interface
The O1 interface is a logical interface used to perform management services within the O-RAN architecture. It is responsible for managing the life-cycle of O-RAN components, which includes tasks such as startup, configuration, fault tolerance, and heartbeat services. Additionally, the O1 interface handles performance assurance, trace collection, and Key Performance Indicators (KPIs) reporting, as well as software and file management under the FCAPS framework [2]. All O-RAN network functions, except for the O-RU, are expected to support the O1 interface when interfacing with the SMO.

• A1 Interface
The A1 interface is a logical northbound (higher-level control influencing lower-level operations) interface that connects the non-RT RIC with the near-RT RIC in the O-RAN architecture. Its primary purpose is to enable the non-RT RIC to provide policy-based

Table 3.2: Interface Mapping Between Components.

Components	**Interface**
SMO and E2 nodes	O1 Interface
Non-RT RIC and near-RT RIC	A1 Interface
Near-RT RIC and E2 nodes	E2 Interface
Non-RT RIC and rApps	R1 Interface
O-CU-CP and O-CU-UP	E1 Interface
O-CU and O-DU	F1 Interface
O-DU and O-RU	Open FrontHaul
SMO and Cloud Platform (O-Cloud)	O2 Interface

guidance, AI/ML model management, and enrichment information, such as context data for machine learning models that may not be directly accessible to the near-RT RIC from network function data. This information is used to optimize specific RAN functions. Additionally, the A1 interface allows for basic feedback from the near-RT RIC to the non-RT RIC, reporting operational metrics or the outcomes of policies applied [10].

• E2 Interface

The interfaces used by near-RT RIC, namely A1 and E2, are also used by the xApps to provide value-added services and enhance the capabilities of the near-RT RIC. The xApps hosted by Near-RT RIC use the E2 interface to collect near real-time information (on a UE basis or a cell basis). The Near-RT RIC control over the E2 nodes is steered via the policies and the data provided via A1 from the Non-RT RIC.

The near-RT RIC utilizes the E2 interface to gather near-real-time information from E2 nodes, either periodically or in response to predefined trigger events. This data is then used to support Radio Resource Management (RRM) by feeding it into the near-RT RIC. Additionally, the E2 interface allows the near-RT RIC to send configuration commands directly to the CU/DU [11].

• R1 Interface

The non-RT RIC hosts the R1 termination, which enables rApps to interface with the non-RT RIC. This setup allows rApps to access data management and exposure services, as well as AI/ML functionalities. Additionally, rApps can interact with the A1, O1, and O2 interfaces through this connection [10].

• O2 Interface

The O2 interface is a collection of services and associated interfaces between the O-Cloud and the SMO. These services are organized into two logical groups: Infrastructure Management Services, which are responsible for deploying and managing cloud infrastructure, and Deployment Management Services, which handle the lifecycle management of virtualized or containerized deployments on cloud infrastructure.[10].

• Open Fronthaul Interface

O-RAN's open fronthaul is a logical interface that standardizes communication between the O-DU and the O-RU, enabling multi-vendor interoperability. To choose the appropriate functional split one must consider the inherent trade-off between keeping the O-RU as simple as possible to reduce costs by centralizing functions in CU and distributing functions toward the RU to alleviate congestion on the fronthaul network. O-RAN Alliance has selected a "7-2x" functional split.

The Open Fronthaul Interface is structured across multiple logical planes: the lower-layer split control plane (LLS-CP), which manages signaling and configuration for controlling the O-RU; the LLS user plane (LLS-UP), which carries user data traffic between the O-DU and O-RU; the synchronization plane (S-plane), which ensures precise timing synchronization between the O-DU and O-RU; and the management plane (M-plane), which oversees O-RU configuration, fault management, and performance monitoring [12, 26].

3.4.2.2.1 Artificial Intelligence and Machine Learning Services The use of AI/ML models in next-generation RANs is a cornerstone of O-RAN's architecture. These models enable advanced functionalities such as zero-touch and automated resource control, anomaly detection, and traffic classification. Within the O-RAN framework, an ML training host is a network function responsible for building and training ML models offline, while an ML inference host is the network function tasked with executing these models and/or performing online training. ML models are typically integrated into larger decision-making solutions hosted by the actor network. This network is ultimately responsible for implementing decisions or actions, which may include configuration management changes via the O1 interface, policy management through the A1 interface, and control or policy adjustments for O-eNB (O-CU/O-DU/O-RU) components over the E2 interface. The deployment of ML hosts determines the specific interface and control parameters used [13, 10]. Three deployment scenarios are typically considered:

- The Non-RT RIC assumes both roles of ML training and inference host. Here, the entire ML process, including model creation, lifecycle management, and data provisioning, is conducted within the SMO. Actions in this deployment include:
  - A policy for the near-RT RIC, which is transferred through the policy service of the A1 interface,
  - An O-CU/O-DU/O-RU configuration parameter, which is enforced using the 01 interface.
- The Non-RT RIC works as the ML training host, while the near-RT RIC serves as the inference host. This setup uses both the O1 and O2 interfaces for model creation and maintenance. Actions include:
  - Forecasting information for near-RT RIC's internal mechanisms (with A1's enrichment data service facilitating data exchange between Non-RT and near-RT RICs),
  - Configuring O-CU/O-DU/O-RU parameters, leveraging the E2 interface for data collection and policy enforcement.

- The Non-RT RIC acts as the ML training host, with ML inference hosted directly on the O-CU or O-DU. This approach extends the distributed intelligence to the RAN components, utilizing their proximity to the data sources for real-time decision-making and control [10].

3.4.2.3 Open Software

Open SW in the context of Open RAN refers to the use of open-source code and collaborative development efforts to create software that enables the operation of RAN functions. This software can be contributed by individuals, companies, and organizations within the Open RAN ecosystem. The goal is to create a more flexible, interoperable, and cost-effective RAN solution by enabling contributions from a broad set of players. One example of open software initiative is the O-RAN Software Community (OSC) which is a partnership between the O-RAN Alliance and the Linux Foundation to support the software development for an Open RAN solution. OSC supports the development of software for RAN functions, including Near-RT RIC, Non-RT RIC, O-CU, O-DU, and other critical components of the Open RAN architecture [9].

3.4.3 Security in ORAN

The adoption of Open RAN introduces significant opportunities for flexibility, efficiency, and innovation in next-generation cellular networks. However, it also brings unprecedented security challenges. The disaggregated and distributed nature of O-RAN, combined with the integration of open interfaces, custom control logic, and AI/ML components, significantly expands the attack surface, creating vulnerabilities for malicious exploitation.

Specific risks include threats against open-source code and the potential for adversaries to compromise AI/ML systems by injecting misleading data into datasets used for offline training. Furthermore, O-RAN inherits security risks from virtualized and cloud-based deployments, such as supply chain threats and the complexities of managing a multi-party ecosystem involving operators, cloud providers, and system integrators. The ORAN system could also be attacked on the Open Fronthaul 7.2x split interface, which is not encrypted on the control plane, because of the challenging timing requirements that encryption would introduce. This introduces man-in-the-middle attacks, in which the attacker impersonates the DU (or RU), and compromises user data or configurations in either of the two endpoints. Despite these challenges, O-RAN's openness also offers advantages, such as increased visibility into RAN operations, enabling operators to maintain greater control over their networks. The virtualized architecture facilitates rapid deployment of security patches, automated testing, and comprehensive oversight of vendors and software components [2].

To address these challenges, a risk-based approach is essential. The O-RAN Security Focus Group (SFG) is working to mitigate risks by adhering to 3GPP security standards and industry best practices. This proactive approach ensures that O-RAN can meet the high-security expectations of 5G networks while balancing the benefits of openness and innovation [27].

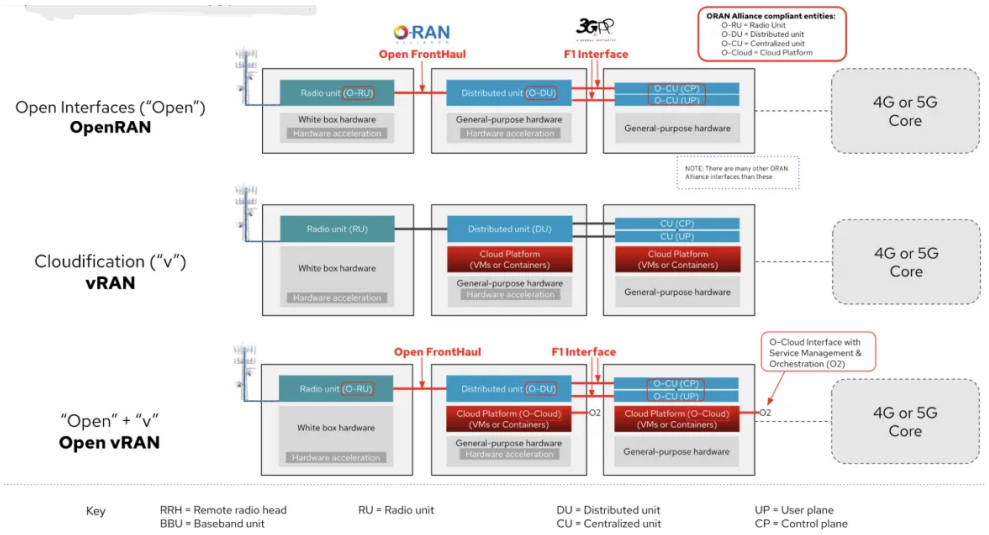


Figure 3.3: Open and Virtualized Models [28].

3.4.4 Open-v-RAN

Open RAN automatically implies disaggregation of hardware and software. This enables vendor interoperability but does not imply open hardware nor open software [9]. The interfaces between different parts of the network are open, but the hardware itself might not always be open. That means operators can still use proprietary hardware as long as it complies with the open interfaces defined by Open RAN. At the same time, operators have options when it comes to the deployment of RU, DU and CU. Some of that equipment may have network functions built into the hardware, while others may have disaggregated software/hardware functions [14].

Open vRAN (Virtualized Radio Access Network) is a term that combines the concepts of virtualization and Open RAN. It refers to the use of virtualization technologies (NFV) in the deployment of Open RAN architectures, deploying O-DU and O-CU as software-based network functions.

3.4.5 Conclusion

The O-RAN framework represents a transformative approach to the evolution of the RAN ecosystem, leveraging open hardware, software, and interfaces to foster interoperability and innovation. By enabling a multivendor ecosystem, O-RAN allows operators to avoid vendor lock-in, adopt "best of breed" solutions, and accelerate the deployment of advanced network services. This flexibility is critical for meeting the diverse and rigorous requirements of 5G and beyond, particularly in vertical industries with varying demands for performance, capacity, and latency.

The O-RAN Alliance emphasizes integrating artificial intelligence and machine learning into the RAN architecture to address the complexity of modern networks. As networks densify and support data-intensive applications such as augmented reality, IoT, and autonomous systems, traditional manual methods of network management are no longer viable. The manual human interventions required in 2G, 3G, and 4G networks cannot

keep up with the scale and scope of what needs to be achieved in multiple spectrum bands with highly sliced and multiservice networks for 5G and beyond. Instead, O-RAN envisions self-driving, self-healing, and auto-configuring networks that leverage learning-based technologies to optimize resource allocation, reduce capital expenditures (RAN approximately 65%-70% of total network CapEx [29]), and enhance user experiences. Interoperable open interfaces are central to the O-RAN architecture, enabling seamless data collection and configuration changes required for AI/ML-driven automation. This intelligence, embedded at every layer of the RAN, ensures that networks can dynamically adapt to their environment and application contexts in real-time.

O-RAN promotes a competitive market and creates opportunities for smaller vendors, traditionally alien to large-scale RAN deployments, achieving nonetheless the economies of scale needed to compete with major vendors remains a challenge [14].

In conclusion, O-RAN represents a pivotal shift toward intelligent, open, and flexible RAN architectures that are critical for realizing the full potential of 5G and beyond. By embracing AI/ML and fostering a competitive ecosystem, O-RAN is poised to drive innovation, reduce costs, and enable the next generation of wireless networks.

Chapter 4

Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning (ML) that focuses on goal-oriented learning through interactions with an environment. Unlike supervised and unsupervised learning, which rely on labeled and unlabeled datasets respectively, RL involves an agent (learner and decision-maker) that interacts with an environment (outside of the agent's control) and learns to achieve a specific objective by sequentially performing actions, receiving feedback in the form of rewards, and refining its behavior to maximize cumulative rewards over time. This learning paradigm allows RL agents to acquire knowledge and skills through direct experience, distinguishing it from other machine learning approaches.

The success of RL is evident in its application to a wide range of real-world problems, particularly those involving complex, sequential decision-making processes. In robotics, for instance, RL has proven to be a transformative tool, enabling robots to autonomously learn intricate tasks through interaction with their environment. This approach contrasts with traditional robotics, which relies on pre-programmed instructions, by empowering robots to optimize their behavior through trial-and-error learning. Such capabilities are particularly advantageous in unstructured and dynamic environments where predefined rules may be inadequate or infeasible. Beyond robotics, RL has been successfully applied in domains such as gaming, recommendation systems, and autonomous vehicles.

The motivation for employing RL lies in its distinctive learning framework, which is particularly suited to problems requiring a series of interdependent decisions. Each action taken by an RL agent directly influences the subsequent state of the system, making it highly effective for scenarios that involve long-term planning and optimization. Furthermore, RL agents are capable of adapting to dynamic environments by continuously learning from new experiences, a feature that differentiates them from traditional static models. RL is also well-suited for tasks where labeled data is scarce, expensive to obtain, or difficult to define.

4.1 Elements of Reinforcement Learning

Beyond the agent, a reinforcement learning system can have three main subelements: an environment and optionally its model, a policy, and a value function.

4.1.1 Environment

The environment in an RL task serves as the external system with which the agent interacts, providing feedback in response to the agent's actions. This feedback enables the agent to iteratively refine its behavior in order to achieve a specified goal. The environment thus defines the context within which the agent operates. This includes three key components:

• **Action Space** A**:** The action space represents the set of all possible actions that the agent can take in a given environment. Actions are the means by which the agent interacts with the environment and attempts to influence its state.

• **State Space** S**:** The state space encompasses all possible configurations or observations of the environment that the agent can perceive. A state provides the necessary information for the agent to make decisions. Thus, the agent's decision-making process is fundamentally a function of the state of the environment.

• **Reward Signal:** The reward signal defines the goal of the RL task by quantifying the desirability of a particular state or action. At each timestep, the environment provides a reward to the agent based on its most recent action and the resulting state. The agent's objective is to maximize the cumulative reward, also known as the return, over time.

The return/cumulative reward is defined as some specific function of the reward sequence. In the simplest case, the return is the sum of the rewards:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T, \tag{4.1}$$

where T is a final time step in episodic tasks, corresponding to a terminal state. The discounted return is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T, \tag{4.2}$$

where $0 \leq \gamma \leq 1$ is called the discount rate and determines the present value of future rewards. For continuing tasks, the formula is altered to be:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{4.3}$$

Alternatively, these formulas can be unified in:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}, \quad \text{including the possibility that } T = \infty \text{ or } \gamma = 1. \tag{4.4}$$

Together, these components form the foundation of any RL problem, which can be reduced to the exchange of three critical signals between the agent and the environment: one signal to represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (the states), and one signal to define the agent's goal (the rewards).

In many RL tasks, a model of the environment is available or can be constructed. Such a model allows the agent to make inferences about the environment's dynamics, predicting

how the environment will respond to a given state-action pair. Specifically, the model provides the next state and the reward resulting from a particular action taken in a given state. The model is formally defined by the use of the state transition function $p(s'|s, a)$, which defines the probability of transitioning to a next state s' given the current state s and action a, and the reward function $R(s, a, s')$, that specifies the reward received when transitioning from state s' to s after taking action a. This approach, known as a model-based method, relies on the explicit representation of the environment's behavior to guide decision-making.

Conversely, model-free methods bypass the need for such a model and rely solely on trial-and-error learning. These methods do not attempt to predict how the environment will change in response to a single action, focusing instead on learning directly from interactions. While model-based methods can be advantageous in structured environments where accurate models can be constructed, model-free methods often excel in scenarios where constructing a sufficiently accurate model is impractical or computationally expensive.

Overall, the environment's role in RL is pivotal, as it not only provides the framework within which the agent operates, but also determines the complexity and feasibility of learning. Factors such as the stochasticity of state transitions and the dimensionality of the state and action spaces significantly influence the choice of RL algorithms and the general performance of the agent.

4.1.2 Policy

In an RL task, the policy defines the agent's strategy for selecting actions at any given time. Formally, it is a mapping from the perceived states of the environment to the actions to be taken when in those states. It represents the agent's decision-making mechanism, encapsulating the knowledge gained during training. Depending on the chosen reinforcement learning algorithm, the agent may change its policy as a result of its experience. It is, thus, usually dynamic.

The policy can take various forms depending on the complexity of the RL task and the algorithm used. In simple cases, it may be represented as a lookup table or a straightforward function. In more complex scenarios, it may involve extensive computation, such as a search process or a neural network. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior.

It may be stochastic, meaning that it assigns probabilities to each possible action from a state, allowing for randomness in action selection. A stochastic policy is often represented as π_t, where $\pi_t(a|s)$ is the probability of selecting action a at time t, given that the current state is s.

4.1.3 Value Functions

Whereas the reward signal provides immediate feedback on the desirability of specific actions or states, value functions extend this concept by estimating the long-term benefit of states or state-action pairs. Value functions are crucial in reinforcement learning as they guide the agent's decision-making process. Instead of selecting actions that yield the highest immediate reward, the agent aims to choose actions that lead to states with the highest estimated value, thereby maximizing expected return in the long run.

Value functions are defined with respect to particular policies. Informally, the value of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter. The function v_π is called the state-value function for policy π. Similarly, the action-value function for policy π is represented as $q_\pi(s, a)$.

The value functions v_π and q_π must be estimated and re-estimated from the sequences of observations an agent makes over its entire interaction with the environment, unlike rewards, which are directly provided by the environment. This estimation process involves aggregating and refining information gathered over time. For example, if an agent follows policy π and maintains an average for each state encountered of the actual returns that have followed that state, then the average will converge to the state's value, $v_\pi(s)$, as the number of times that state is encountered approaches infinity. If separate averages are kept for each action taken in a state, then these averages will similarly converge to the action values, $q_\pi(s, a)$ (Monte Carlo estimation method).

Efficiently estimating value functions is a cornerstone of reinforcement learning. The ability to accurately estimate value functions is fundamental to an agent's capacity to learn optimal policies and achieve long-term success in its environment.

4.2 Markov Decision Process (MDP)

In a more formal way, the interaction between agent and environment takes place in a sequence of discrete time steps, t = 0, 1, 2, 3, ... (using discrete time). At each time step t, the agent receives some representation of the environment's state, $S_t \in S$, where S is the set of possible states, and on that basis selects an action, $A_t \in A(S_t)$, where $A(S_t)$ is the set of actions available in state S_t. One time step later, as a consequence of its action, the agent receives a numerical reward, $R_{t+1} \in R$, and transitions to a new state S_{t+1}.

In order to effectively interact with the environment, the agent must be able to sense the state of the environment to some extent and must be able to take actions that affect the state. The agent also must have a goal or goals relating to the state of the environment. Nonetheless, the state signal should not be expected to inform the agent of everything about the environment. A state signal that encapsulates all relevant information is said to be Markov or to have the Markov property. An environment is considered to satisfy the Markov property if its state signal sufficiently summarizes all necessary past information, allowing accurate predictions of future states without the need for additional historical data.

To simplify the mathematical formulation, it is assumed that there is a finite number of states and reward values. In the most general, causal case, the response of the environment at time $t + 1$ to the action taken at t may depend on the entire history of events that have occurred earlier. In this case the dynamics can be defined only by specifying the complete probability distribution:

$$P\{R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, ..., S_t, A_t\}, \forall r, s', \text{ and all possible values of } S_i, R_i. \quad (4.5)$$

If the state signal has the Markov property, on the other hand, then the environment's response at $t + 1$ depends only on the state and action representations at t, in which case the environment's dynamics can be defined by:

$$p(s', r|s, a) = P\{R_{t+1} = r, S_{t+1} = s' | S_t, A_t\}, \forall r, s', S_t, A_t. \quad (4.6)$$

When the environment exhibits the Markov property, this one-step dynamic model enables the prediction of the next state and the expected next reward based only on the current state and action. The Markov property is important in reinforcement learning because decisions and values are assumed to be a function only of the current state.

A reinforcement learning task that satisfies the Markov property is called a Markov decision process, or MDP. If the state and action spaces are finite, then it is called a finite MDP. A specific finite MDP is defined by its state and action sets and by the one-step dynamics of the environment. Given any state and action s and a, the probability of each possible pair of next state and reward, s', r, is denoted:

$$p(s', r|s, a) = P\{R_{t+1} = r, S_{t+1} = s'|S_t = s, A_t = a\}. \tag{4.7}$$

This information allows for the computation of other important quantities, such as the state-transition probabilities:

$$p(s'|s, a) = P\{S_{t+1} = s'|S_t = s, A_t = a\} = \sum_{r \in R} p(s', r|s, a). \tag{4.8}$$

4.3 Bellman Equations

For an MDP the value functions can be defined formally as:

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] \\ &= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \Big| S_t = s\right] \\ &= E_\pi\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \Big| S_t = s\right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \Big| S_{t+1} = s'\right]\right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right], \quad \forall s, s' \in S, a \in A(s), r \in R \text{ [15].} \end{aligned} \tag{4.9}$$

For each triple s, a, s', its probability, $\pi(a|s)p(s', r|s, a)$ is computed. The quantity in brackets is weighted by that probability, then the sum over all possibilities is taken to get an expected value. This is the Bellman equation for v_π. It expresses a relationship between the value of a state and the values of its successor states, by averaging over all the possibilities and then weighting each by its probability of occurring.

Similarly for action value function q:

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a][15]. \tag{4.10}$$

4.3.1 Bellman Optimality Equations

The objective of solving a reinforcement learning task is to identify a policy that maximizes the cumulative reward over the long term. In the context of finite MDPs, the optimal policy can be formally defined.

Value functions provide a mechanism to establish a partial ordering among policies. A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. Mathematically, this relationship is expressed as $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s), \forall s \in S$. There is always at least one policy that is better than or equal to all other policies. This is an optimal policy. While there may be multiple optimal policies, they are collectively denoted as π_*. They share the same state-value function, called the optimal state-value function, denoted v_*, and defined as:

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in S. \tag{4.11}$$

The Bellman optimality equation is subsequently defined as:

$$\begin{aligned} v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\ &= \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \text{ [15].} \end{aligned} \tag{4.12}$$

This equation conveys that the value of a state under an optimal policy must equal the expected return for the best action available from that state.

Similarly, the optimal action-value function $q_*(s, a)$ is defined:

$$q_*(s,a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')][15]. \tag{4.13}$$

For finite MDPs, the Bellman optimality equation has a unique solution independent of the policy. If the dynamics of the environment, p, are known, then it is theoretically possible to compute this solution.

Once the optimal state-value function v_*, is obtained, determining an optimal policy becomes straightforward. For each state s, there will be one or more actions at which the maximum is obtained in the Bellman optimality equation. Any policy that assigns non-zero probability only to these actions is an optimal policy. In essence, any policy that is greedy (picks the best action with one-step look ahead) with respect to the optimal evaluation function v_* qualifies as an optimal policy. The significance of the optimal value function lies in its ability to transform the optimal expected long-term return into a locally and immediately available quantity for each state, since it already takes into account the reward consequences of all possible future behavior.

Many reinforcement learning methods can be understood as approximations to solving the Bellman optimality equation. These methods typically rely on actual experienced transitions as substitutes for the expected transitions, p in scenarios where the environment dynamics are not explicitly known.

4.4 Exploration vs. Exploitation

The exploration-exploitation dilemma represents a fundamental challenge in reinforcement learning and decision-making processes. At its core, this dilemma arises from the tension between two competing strategies: exploitation, which involves leveraging known

information to maximize immediate rewards, and exploration, which involves seeking new information to improve future decision-making. Striking an optimal balance between these two strategies is critical for an agent to achieve long-term success.

Exploitation focuses on selecting actions that have previously yielded high rewards, relying on the agent's current knowledge to make efficient and effective decisions. This approach is essential for maximizing short-term gains, as it prioritizes actions with proven outcomes. However, excessive exploitation can lead to suboptimal behavior, as the agent may overlook potentially superior actions that have not yet been thoroughly evaluated.

On the other hand, exploration emphasizes the discovery of new actions. This is particularly important in stochastic environments, where the true value of an action can only be estimated through repeated trials. Exploration allows the agent to refine its understanding of the environment and uncover actions that may yield higher rewards in the long run. However, excessive exploration can be inefficient, as it may involve sacrificing immediate rewards for uncertain future benefits.

The agent cannot simultaneously explore and exploit. Every action selection inherently involves a choice between exploiting known rewards or exploring new possibilities. This trade-off is further complicated in dynamic or uncertain environments, where the optimal balance may shift over time as the agent gains more information. Ultimately, the challenge for a reinforcement learning agent is to navigate this trade-off effectively, ensuring that it neither becomes overly conservative by exploiting suboptimal actions nor wastes resources on excessive exploration.

4.5 Dynamic Programming Methods

The term dynamic programming (DP) in RL is used to describe a collection of algorithms that can be used to compute optimal policies, by solving the Bellman Optimality equations, assuming a perfect model of the environment represented as an MDP. Among them, Policy Iteration and Value Iteration are two fundamental approaches for determining the optimal policy, each employing a distinct strategy to achieve this objective.

Capital $V_\pi(), Q_\pi()$ are used to denote the estimates of the value, action-value functions under π, while $v_\pi(), q_\pi()$ symbolize the actual functions under π, $v_*(), q_*()$ if they are optimal over all policies.

4.5.1 Iterative Policy Evaluation

Iterative Policy Evaluation is a method used to calculate the value function for a given policy π, it is thus a "prediction" problem. Specifically, it estimates the state-value function $V_\pi(s)$, which gives the expected return (or value) of each state s when following policy π. The method iteratively refines its estimate of the value function until it converges to the true value function.

The process of Iterative Policy Evaluation is:

Algorithm 4: Iterative Policy Evaluation

Input: State space S, Action space A, Policy π, Discount factor γ, Transition function p, Convergence threshold θ

Output: Updated value function $V(s)$

1 **Initialization:** Start with an arbitrary value function $V(s)$, often initialized to 0 for all states.

2 **Iterative Update:** For each state $s \in S$, update $V(s)$ using the Bellman equation for v_π as an update rule:

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} p(s', r|s, a)[r + \gamma V_k(s')]$$

This update is done for all states in each iteration k. Here, $V_k(s)$ is the value of the state at iteration k, and $V_{k+1}(s)$ is the updated value after one more iteration.

3 **Convergence:** Repeat the update step until the values stabilize (i.e., the difference between consecutive updates is below a small threshold θ).

Iterative Policy Evaluation progressively improves the estimates of how good each state is under a policy by "bootstrapping" from its previous estimates. Initially, the estimates may be poor, but as the algorithm iterates, the value of each state becomes more accurate as it incorporates the expected rewards and the values of future states. At convergence, the value function $V_\pi(s)$ will satisfy the Bellman equation, meaning it represents the true value of each state under the policy π. Convergence is guaranteed for a finite MDP [15].

Since Iterative Policy Evaluation updates the value function for each state in each iteration, the computational complexity depends on the number of states and the number of iterations required for convergence.

4.5.2 Policy Improvement

The reason for computing the value function for a policy is to facilitate the identification of better policies. When evaluating a deterministic policy π, the question arises whether to modify the policy by selecting a different action $a \neq \pi(s)$ for some state s. The key consideration is whether this change would yield a better or worse outcome. One approach to addressing this question is to assess the consequences of selecting action a in s and thereafter follow the existing policy, π [15].

Let π and π' be any pair of deterministic policies such that, $\forall s \in S$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s). \tag{4.14}$$

Then the policy π' must be as good as, or better than, π. That is, the expected return under π' is greater than or equal to that under π for all $s \in S$:

$$v_{\pi'}(s) \geq v_\pi(s). \tag{4.15}$$

To refine the policy towards optimality, all possible state transitions and available actions should be evaluated. At each state, the action that maximizes the expected return,

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Figure 4.1: Policy Iteration [15].

as defined by $q_\pi(s, a)$ should be selected. This leads to the formulation of a new greedy policy, π', given by

$$\pi'(s) = \arg\max_a q_\pi(s, a) \tag{4.16}$$

The process of making a new policy π' that improves on an original policy π, by making it greedy with respect to the value function of the original policy, is called policy improvement. These extend to stochastic policies as well.

4.5.3 Policy Iteration

Policy Iteration is a method aiming to compute the optimal policy π_*, given an MDP (control problem).

Policy Iteration alternates between two steps: policy evaluation and policy improvement. It progressively improves the policy by using its current estimate of the value function:

Algorithm 5: Policy Iteration

Input: State space S, Action space A, Discount factor γ, Initial policy π, Transition function $p()$

Output: Optimal policy π^*

1 **Policy Evaluation:** For a given policy π, compute the state-value function $v_\pi(s)$. This is typically done using Iterative Policy Evaluation until convergence.

2 **Policy Improvement:** Once the value function $v_\pi(s)$ is computed, improve the policy by acting greedily with respect to the current value function using a one-step look-ahead. The new policy π' is updated as:

$$\pi'(s) = \arg\max_a \sum_{s'} p(s'|s, a)(r + \gamma v_\pi(s'))$$

This means that the new policy chooses actions that maximize the expected return based on the current value estimates.

3 **Repeat:** Alternate between policy evaluation and policy improvement until the policy stabilizes (i.e., the policy does not change anymore). Once this happens, the policy is optimal.

Convergence is guaranteed for a finite MDP [15].

4.5.4 Value Iteration

Value Iteration is an alternative to Policy Iteration for finding the optimal policy.

In value iteration, the optimal state value function is computed by iteratively updating the estimate. Instead of waiting for the policy evaluation to fully converge, Value Iteration updates the value function in every iteration and simultaneously improves the policy:

Algorithm 6: Value Iteration

Input: State space S, Action space A, Discount factor γ, Initial value function $V_0(s)$, Transition function $p()$

Output: Optimal policy π^* and optimal value function V_{π^*}

1 **Value Update:** Instead of fully evaluating the policy, the value function is updated directly using the Bellman Optimality Equation:

$$V_{k+1}(s) = \max_a \sum_{s'} p(s'|s,a)(r + \gamma V_k(s'))$$

This update simultaneously improves the value function and the policy.

2 **Repeat:** The value function is updated iteratively for all states until convergence. Once the value function has converged, the optimal policy can be derived by acting greedily with respect to the optimal value function:

$$\pi_*(s) = \arg\max_a \sum_{s'} p(s'|s,a)(r + \gamma V(s'))$$

Value Iteration is more computationally efficient than Policy Iteration, since it simultaneously updates value and policy and does not involve exhaustive policy evaluation.

4.5.5 Generalized Policy Iteration

The term generalized policy iteration (GPI) is used to refer to the general idea of letting policy evaluation and policy improvement processes interact, independent of the granularity and other details of the two processes. Both Policy and Value Iteration are GPI algorithms, but differ in the mechanics of their updates.

4.5.6 Limitations of DP Methods

A major drawback to the DP methods is that they involve operations over the entire state set of the MDP, that is, they require sweeps of the state set. They also assume a perfect model of the MDP, which in reality, is nearly never the case. In practice, this basic approach is totally impractical, because the action-value function is estimated separately for each sequence, without any generalization. Instead, it is common to use a function approximator to estimate the action-value function, which is either a linear function approximator, or a non-linear function approximator, such as a neural network (DRL).

4.6 Types of Reinforcement Learning

4.6.1 Temporal Difference Learning

Temporal Difference (TD) Learning includes a class of methods used for both for prediction (estimate value functions) and control (optimize policy). Unlike DP methods, they do not assume complete knowledge of the environment, that is full distribution of all possible successors. Instead, TD methods are model-free and require only raw experience—sample sequences of states, actions, and rewards from actual or simulated interaction with an environment. Similarly to DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap).

In prediction problem, TD learning updates value estimates incrementally after each step. The simplest TD method for prediction, known as TD(0), is defined by the following update rule:

$$V(S_t) \leftarrow V(S_t) + a[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)], \tag{4.17}$$

where a is the learning rate and the value $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is the TD error, that represents the difference between the current value estimate and the updated estimate based on observed transitions. For any fixed policy π, TD(0) has been proved to converge to the real value function v_π [15].

4.6.2 On-Policy vs. Off-Policy Learning

Since in control problems the goal is to optimize the policy, the control methods learn an action-value function rather than a state-value function. TD control methods are categorized into on-policy and off-policy methods. This distinction is not exclusive to TD learning and primarily refers to how the data used for learning is generated and how the learning process interacts with the policy being optimized. To clarify this distinction, it is essential to differentiate between the behavior policy and the update policy:

• **Behavior Policy:** The behavior policy is the policy an agent follows when choosing which action to take in the environment at each time step, it generates actions and determines how the agent interacts with the environment. For example in an ε-greedy stochastic policy, the agent selects the action with the highest estimated value with probability 1-ε and another random action is chosen with probability ε.

• **Update Policy:** The update policy is how the agent updates the Q-function. Specifically, it dictates how the agent derives the state-action pairs which are used to calculate the difference between the actual q-value and current predicted Q-value, also known as the TD-error. The TD-error is then used to update the Q-function.

On-policy algorithms attempt to improve upon the current behavior policy that is used to make decisions and therefore these algorithms learn the value of the policy carried out by the agent, Q_π. Off-policy algorithms learn the value of the optimal policy, $Q*$, and can improve upon a policy that is different from the behavior policy.

For example, Sarsa is an on-policy TD control algorithm that uses the update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + a[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]. \tag{4.18}$$

Sarsa converges to an optimal policy and action-value function as long as all state–action pairs are visited an infinite number of times and the policy converges in the limit to the greedy policy [15].

Algorithm 7: Sarsa Algorithm for episodic tasks

Input: State space S, Action space A, Learning rate α, Discount factor γ, Exploration rate ϵ

Initialize $Q(s,a)$, $\forall s \in S$, $a \in A(s)$ arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
Repeat (for each episode):
Repeat
    Initialize S
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Repeat
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
        Update $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
        Set $S \leftarrow S'$; $A \leftarrow A'$
Until S is terminal

In contrast, Q-learning is an off-policy TD control algorithm, with one-step update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + a[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \tag{4.19}$$

In this case, the learned action-value function, Q, directly approximates q_*, the optimal action-value function, independent of the policy being followed. The policy still has an effect in that it determines which state–action pairs are visited and updated. All that is required for correct convergence is that all pairs continue to be updated.

Algorithm 8: Q-learning Algorithm for episodic tasks

Input: State space S, Action space A, Learning rate α, Discount factor γ, Exploration rate ϵ

Initialize $Q(s,a)$, $\forall s \in S$, $a \in A(s)$ arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
Repeat (for each episode):
Repeat
    Initialize S
    Repeat
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        Update $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        Set $S \leftarrow S'$
Until S is terminal

4.6.3 Policy Optimization vs. Q-learning

The methods analyzed so far (DP, TD) are tabular, meaning that estimates of value functions V, Q, are represented as a table with one entry for each state or for each state–action pair. This is limited to tasks with small numbers of states and actions and does not generalize well if the action or state spaces are continuous. To achieve better generalization in model-free environments, function approximation is leveraged. Methods of this family represent the approximate policy or action-value function not as a table, but as a parameterized function with a learnable parameter vector, whom goal is to optimize in order to achieve maximum expected return. This forms the foundation of Deep Reinforcement Learning, which integrates neural networks into RL tasks.

4.6.3.1 Policy Gradient Methods (Policy Optimization)

In this case, a policy is explicitly represented as $\pi_\theta(a|s)$, where $\boldsymbol{\theta} \in \mathcal{R}^n$ are the parameters of the policy (e.g., weights of a neural network). The goal is to update $\boldsymbol{\theta}$ to values that make π_θ the optimal policy, which will maximize the expected cumulative reward. Therefore, the objective function for policy optimization in episodic tasks is:

$$J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{E}[R(\tau)], \tag{4.20}$$

where $R(\tau)$ denotes the return from trajectory τ. Because $\boldsymbol{\theta}$ will change, $\boldsymbol{\theta_k}$ denotes $\boldsymbol{\theta}$ at iteration k. The policy is optimized by gradient ascent:

$$\theta_{k+1} \leftarrow \theta_k + a\nabla_\theta J(\pi_\theta)|\pi_{\theta_k}. \tag{4.21}$$

The intuition behind this is that the gradient of the optimization function points in the direction in which to move $\boldsymbol{\theta}$ so as to increase the value of $\pi_{\theta_t}(a|s)$ the fastest. For example, if for some state s, action $a*$ is optimal, $\boldsymbol{\theta}$ should be updated so as to increase the probability $\pi_\theta(a*|s)$, reducing at the same time the probability for other actions in s. Learning rate a is used to control the magnitude of this update.

It is proven ([16]) that:

$$\nabla_\theta J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{E}[\sum_{t=0}^{T} \nabla_\theta log\pi_\theta(a_t|s_t)\Phi_t], \tag{4.22}$$

where the first term shows how the probability of choosing a_t in s_t changes with respect to θ and the second is used as a weight for the update. Assuming that the policy is represented in a way which allows the calculation of its gradient, the mean value is calculated by running the policy in the environment to collect a trajectory dataset D and taking the sample mean:

$$\nabla_\theta J(\pi_\theta) = \frac{1}{|D|}\sum_{\tau\in D}\sum_{t=0}^{T} \nabla_\theta log\pi_\theta(a_t|s_t)\Phi_t. \tag{4.23}$$

Depending on the specific policy optimization method, the function Φ_t can take different forms, that do not invite bias in the gradient ([15]). The most usual ones are:

• Total return of episode, $R(\tau)$. It has high variance due to the long-term dependencies. This causes large fluctuations in the gradient estimates. As a result, each policy update can be very noisy, which can slow down convergence and make training unstable.

• $\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})$: In the previous case, $R(\tau)$ included all rewards obtained in τ, but in reality, rewards obtained before taking an action have no bearing on how good that action was: only rewards that come after. This modified function is used to take that into consideration.

• $\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t)$, where $b(s_t)$ is called baseline function and is used to reduce the variance. Usually, the value function is used as baseline, $b(s_t) = v_\pi(s_t)$. In practice, $v_\pi(s_t)$ cannot be computed exactly, so it has to be approximated. This is usually done with a neural network, $V_\phi(s_t)$, which is updated concurrently with the policy (so that the value network always approximates the value function of the most recent policy). The goal for this network is to minimize a loss function, measuring the accuracy of V_ϕ. In the simplest case, as loss function can be used the MSE:

$$L(\phi) = \underset{s_t, \hat{R}_t \sim \pi_k}{E} [(V_\phi(s_t) - \hat{R}_t)^2]). \tag{4.24}$$

For learning V_ϕ, gradient descent is used to minimize the loss function:

$$\phi_{k+1} \leftarrow \phi_k - a_l \nabla L(\phi)|\phi_k. \tag{4.25}$$

• The Advantage Function $A_{\pi_\theta}(s_t, a_t) = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t)$. The Advantage function describes how much better or worse taking action a in state s is compared to acting according to the policy.

4.6.3.2 Value Based Methods (Q-learning)

Methods in this family learn an approximator $Q_\theta(s, a)$ for the optimal action-value function, $q^*(s, a)$. They extend the concept of Q-learning algorithm of TD section, incorporating function approximation. This optimization is performed off-policy, which means that each update can use data collected at any point during training, regardless of how the agent was choosing to explore the environment when the data was obtained. The loss function is the Mean Square TD Error:

$$L(\theta) = \underset{s,a \sim p()}{E} ([r_{t+1} + \gamma \max_a Q_\theta(s_{t+1}, a) - Q_\theta(s_t, a_t))^2], \tag{4.26}$$

where γ is the discount factor and $p(s, a)$ is the behavior policy (a probability distribution over states and actions through which experience is collected), e.g. $\epsilon-$greedy. Parameters $\boldsymbol{\theta}$ are updated through gradient descent.

The goal remains to find the optimal policy, which is obtained via the connection between q^* and π^*: the actions taken by the Q-learning agent are given by:

$$a(s) = \arg\max_a Q_\theta(s, a). \tag{4.27}$$

4.6.3.3 Comparison

Value-based and policy-based RL have both specific advantages and corresponding use cases.

One limitation of Q-learning is that it inherently produces a deterministic policy based on the learned action-value function. This means that it can not learn stochastic policies, which can be useful in most environments, while at the same time it does not incorporate an explicit exploration mechanism. Another drawback is that, since the neural network approximating the Q-function must output a value for each possible action, there is no straightforward way to extend Q-learning to continuous action spaces. Policy gradient also prevails in that it follows gradients with respect to the policy itself, which means direct improvement of the policy. By contrast, in Q-Learning the improvement is performed on the estimates of the values of actions, which only implicitly improves the policy. This in practice is less efficient.

Policy gradient methods may suffer from high gradient variance but, overall, tend to outperform value-based methods, particularly in high-dimensional or continuous action spaces.

4.7 Algorithms in Deep Reinforcement Learning

To effectively illustrate the differences between policy gradient methods and value-based methods in reinforcement learning, it is essential to examine specific algorithms that embody these approaches. While both families of methods aim to optimize an agent's decision-making process, they differ fundamentally in their underlying principles, update mechanisms, and suitability for various types of tasks.

4.7.1 Deep Q-learning Network (DQN)

DQN is a model-free, off-policy algorithm, that belongs in the class of Q-learning methods. It learns to approximate the optimal action-value function $Q_*(s, a)$, using a neural network with parameters $\boldsymbol{\theta}$. The neural network takes the current state s as input and outputs a Q-value for each possible action. Similar to other value-based algorithms, DQN employs several key techniques to enhance learning stability and efficiency:

• **Experience Replay:** The agent maintains a replay buffer to store experiences (consisting of state, action, reward, and next state tuples). Learning directly from consecutive samples is inefficient due to strong temporal correlations, which can lead to overfitting and increased variance in updates. To mitigate these issues, experiences are sampled randomly from the replay buffer during training. This process helps to break temporal dependencies and ensures that the network learns from a more diverse distribution of past experiences.

• **Target Network:** DQN uses a separate target network with the same architecture as the main Q-network, but with frozen parameters for a fixed number of steps, to stabilize the learning process. Periodically, the target network is updated by copying the weights from the main network.

The DQN algorithm, after proper state representation and Neural Network construction, involves the following steps [30]:

Algorithm 9: DQN Algorithm

1 **Store experience** from interaction with the environment in the replay buffer.

2 **Repeat until convergence**:

3 **Sample mini-batches** of experiences from the replay buffer to update the neural network weights by performing (mini-batch) gradient descent on the loss function:

$$L(\theta_i) = \mathbb{E}_{s,a\sim p()}\left[\left(r_{t+1} + \gamma \max_a Q_{\theta^{i-1}}(s_{t+1}, a) - Q_{\theta^i}(s_t, a_t)\right)^2\right].$$

Here the state-action value of the target $((r_{t+1} + \gamma \max_a Q_{\theta_{i-1}}(s_{t+1}, a))$ is computed based on the target network, which uses the parameters from the previous iteration $\boldsymbol{\theta_{i-1}}$. Using the same network would cause instability in training, as it would simultaneously modify the parameters used to generate its own target values. The parameter update at step k is:

$$\theta^i_{k+1} \leftarrow \theta^i_k - \alpha \nabla_\theta L(\theta)|_{\theta^i_k}.$$

4 **Balance exploration and exploitation**: Select actions either greedily based on the current policy or stochastically to encourage exploration.

5 **Update target network**: If needed (every some steps), update the target network by either copying parameters from the main network:

$$\theta^{i-1} \leftarrow \theta^i$$

or by Polyak averaging:

$$\theta^{i-1} \leftarrow p\theta^{i-1} + (1-p)\theta^i,$$

where p is a hyperparameter between 0 and 1.

4.7.2 Actor-Critic Methods

Actor-Critic (AC) methods are a class of reinforcement learning algorithms that leverages the advantages of both value-based and policy-based methods. It introduces two separate components—the Actor and the Critic—to effectively optimize the agent's behavior:

• **Actor:** The actor is a neural network parameterized by $\boldsymbol{\theta}$, that embodies the policy function. It is responsible for selecting actions based on the current state of the environment.

• **Critic:** The critic is a neural network parameterized by $\boldsymbol{w}$, that evaluates the chosen by the actor action by estimating the expected return (e.g., using a Q-value or a state-value function). It provides feedback to the actor, guiding it toward better policy updates over time. This feedback reduces the high variance that policy gradient methods typically suffer from.

The integration of these two components enables actor-critic methods to combine the stability of value-based approaches with the flexibility of policy-based methods, particularly in environments with continuous action spaces. The critic's role is pivotal, as it provides a stable and efficient training signal to the actor, leading to more robust convergence compared to purely policy-based or value-based methods.

Specifically, the actor optimizes the objective function $J(\theta)$, that represents the expected return under π_θ. The policy parameters are updated using gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + a\nabla_\theta J(\theta)|\theta_t, \tag{4.28}$$

where the policy gradient is given by:

$$\nabla_\theta J(\theta) = \underset{\tau\sim\pi_\theta}{E}[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t)(R_{t+1} - V(s_t)]. \tag{4.29}$$

Here, R_{t+1} denotes the total reward from that step until the end of the episode (second case of baselines functions, as defined in 4.6.3.1 section) (this constitutes the action-value $Q(s_t, a_t)$, so the second term of the above relationship is in fact the advantage $A(s_t, a_t)$). However, since computing R_{t+1} requires access to the complete trajectory, the agent computes an estimate, which in the simplest form is:

$$R_{t+1} \approx r_{t+1} + \gamma V(s_{t+1}). \tag{4.30}$$

Simultaneously, the critic receives the values of r_t, s_{t+1} from the environment and aims to minimize the TD error:

$$\delta_t = r_t + V_w(s_{t+1}) - V_w(s_t), \tag{4.31}$$

so the critic's loss function is defined as:

$$J(w) = (r_t + \gamma V_w(s_{t+1}) - V_w(s_t))^2, \tag{4.32}$$

and its parameters are updated using gradient descent:

$$w_{t+1} \leftarrow w_t - a_V \nabla_w J(w). \tag{4.33}$$

The critic then gives this TD error to the actor, so the actor's gradient takes its final form:

$$\nabla_\theta J(\theta) = \underset{\tau\sim\pi_\theta}{E}[\sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_w(s_{t+1}) - V_w(s_t)]. \tag{4.34}$$

Intuitively, if the critic computes a positive TD error, indicating that the value of the current state has been underestimated, the parameters θ of the actor are updated in the direction where the probability of action a_t in s_t increases. Conversely, a negative TD error discourages the actor from favoring the same action in the future.

4.7.2.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a model-free, on-policy reinforcement learning algorithm that falls under the Actor-Critic framework. It is designed to improve the stability and efficiency of policy updates by constraining large policy changes, which can otherwise destabilize training. There are two main variants: PPO-Clip and PPO-Penalty, that use different mechanisms to limit the step size of policy updates. The PPO-Clip variant is the most widely used implementation of PPO.

In PPO-Clip the role of the critic remains the same, as described before. The actor, however, aims to maximize a modified objective:

$$L^{CLIP}(\theta) = \underset{\tau \sim \pi_{\theta_{old}}}{E}[\min(r(\theta)\hat{A}^{\pi_{\theta_{old}}}(s,a), clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_{old}}}(s,a))], \tag{4.35}$$

where:

- $\pi_{\theta_{old}}$ denotes the old policy (the one used to collect the data) and π_θ denotes the new policy (the one being optimized),
- $r(\theta)$ is the probability ratio between new and old policy, which measures the difference between them:
$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{4.36}$$
- ϵ is a (small) hyperparameter which roughly says how far away the new policy is allowed to go from the old one,
- the clip function clips the probability ratio, ensuring that it stays within the interval $[1-\epsilon, 1+\epsilon]$,
- $\hat{A}^{\pi_{\theta_{old}}}(s,a)$ is the advantage estimate. The simplest way would be to estimate the advantage with the TD error:
$$\hat{A}^{\pi_{\theta_{old}}}(s,a) \approx \delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t). \tag{4.37}$$
However, this one-step transition would lead to high variance in policy updates. Thus, the agent leverages Generalized Advantage Estimation (GAE). GAE smooths and stabilizes advantage estimation by incorporating multiple future rewards using an exponentially weighted sum of TD errors:
$$\hat{A}^{\pi_{\theta_{old}}}(s_t, a_t) \approx \sum_{l=0}^{T-1} (\gamma\lambda)^l \delta_{t+l} \tag{4.38}$$

The term $r(\theta)\hat{A}^{\pi_{\theta_k}}()$ encourages the policy to increase the probability of actions with positive advantages (good actions) and decrease the probability of actions with negative advantages (bad actions). At the same time, the clip function ensures that the probability ratio does not deviate too far from 1, preventing overly large updates to the policy in a single step. This is the key mechanism that makes PPO stable.

So the full PPO-Clip algorithm, after proper state representation and Neural Network construction for actor and critic, includes the following steps [17]:

Algorithm 10: PPO Algorithm

Collect a set of trajectories D_k by running policy π_{θ_k} in the environment.

Compute advantage estimates $\hat{A}^{\pi_{\theta_k}}(s,a)$ using GAE (or some other method) based on the current value function $V_{w_k}(s)$.

Update the policy network via gradient ascent:

Repeat for N iterations:

Perform policy update:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta L^{CLIP}(\theta)|_{\theta_k},$$

where the clipped objective is:

$$L^{CLIP}(\theta_k) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T-1} \min\Big(r(\theta)\hat{A}^{\pi_{\theta_k}}(s_t,a_t), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_k}}(s_t,a_t)\Big)$$

Update the critic network via gradient descent:

$$w_{t+1} \leftarrow w_t - \alpha_V \nabla_w J(w_t),$$

where the critic loss is given by:

$$J(w) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T-1} \left(r_{t+1} + \gamma V_w(s_{t+1}) - V_w(s_t)\right)^2$$

Chapter 5

Problem Formulation

This chapter introduces the problem formulation, addressing the challenges of slice admission control and Virtual Network Function (VNF) placement within the physical architecture of O-RAN, utilizing Deep Reinforcement Learning (DRL). The goal is to model a realistic O-RAN environment and develop an RL agent, that will efficiently make decisions regarding the admission, placement, and splitting of incoming slice requests.

5.1 Network Model

A standard O-RAN-based architecture is assumed, wherein Edge Clouds (ECs) are connected via FH links to the radio unit (RU), to support applications that require low-latency. A simplified version, where each EC contains a single DU, is assumed. A single Regional Cloud (RC), connected via MH links to the ECs, functions as a centralized computing resource, responsible for higher-level processing and coordination across multiple ECs. The FH links facilitate low-latency communication between the RU and the ECs, while MH links provide high-bandwidth connectivity between ECs and the RC. Both ECs and the RC are characterized by their total computing capacities, measured in CPU cores, and every link is characterized by its bandwidth capacity and the delay it imposes.

The parameters associated with the environment formulation are introduced in Table 5.1.

5.2 Slice Request Model

The network receives slice requests in the form of Service Function Chain (SFC), which specify the sequential order in which VNFs, indexed by $v_f \in F$ (where F is a set of available VNFs), are processed within the network slice. In other words, a network slice corresponds to an ordered subset of F denoted as $F_s = \{{v_0}^s, {v_1}^s, ..., {v_{n_s-1}}^s\} \subseteq F$. It is important to note that VNFs with indices 0 and 1 are fixed across all slice requests, with the former being mandatorily deployed at the RU and the latter at a DU. Additionally, each network slice s is characterized by specific computational and networking resource requirements. These are defined per VNF and, in particular, for each VNF ${v_f}^s \in F_s$ let ${c_f}^s$ stand for its demand

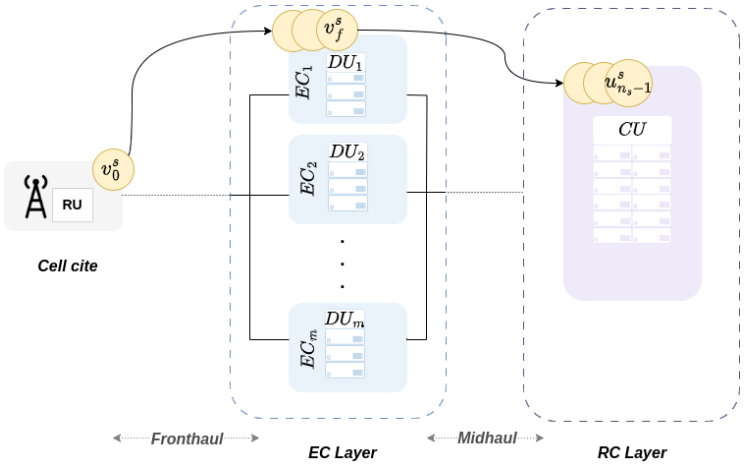


Figure 5.1: Network Model.

in CPU cores and ${b_f}^s$ the required bandwidth for the data transfer between consecutive VNFs ${v_f}^s, {v_{f+1}}^s$ over a network link. Furthermore, each request s arrives at a specific time-slot t_s within a limited time horizon H. The request also includes a holding time ht_s, indicating the duration for which the slice remains active once requested. Moreover, an end-to-end delay requirement $D_{max,s}$ and a priority value pr_s are defined for each slice request, reflecting its tolerance level for delay and its relative importance, respectively.

The main decision variables are the indicators of whether or not a slice is accepted and where the VNFs should be placed if the slice is accepted. To this end, the binary $X_s(t)$ is defined, that equals 1 if a slice request has been admitted at time t or earlier, and 0 otherwise. For the VNFs placement, the binary variables $x^e_{s,f}(t)$, $y_{s,f}(t), \forall e \in \mathcal{E}, f \in F_s, s \in R(t)$ are defined as follows:

$$x^e_{s,f}(t)/y_{s,f}(t) = \begin{cases} 1, & f \in F_s \text{ is placed on EC } e \in \mathcal{E}/RC \text{ at t,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

Furthermore, an additional variable $sr_s(t)$ indicates if an active (i.e., not expired) slice s is admitted at time t:

$$sr_s(t) = \begin{cases} X_s(t), & t_s \leq t < (t_s + ht_s), \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

5.3 Constraints

The placement of slices is, of course, subject to several constraints, including the capacities of the network elements and the delay requirements associated with each slice [18].

First, there exist aggregated computing capacity limitations for each EC and the RC, expressed as,

Notation	Description
E	Set of ECs
F	A set of available VNFs
F_s	An ordered subset of VNFs consisting a service chain for slice s
$f \in F_s$	Index of a VNF in the service chain of slice s
n_s	Number of VNFs of slice s
t_s	Arrival time of slice s
ht_s	Holding time of slice s
pr_s	Priority value of slice s
$R(t)$	Set of slices arrived up to time t
$c_{s,f}$	CPU requirement of VNF f of slice s
$b_{s,f}$	Bandwidth requirements for two successive VNFs $f-1, f$ of slice s
$D_{max,s}$	End-to-end delay requirement of slice s
$C_e(t)$	CPU utilization on cloud $e \in \mathcal{E}$
$B_{FH,e}(t)$ $(B_{MH,e}(t))$	Bandwidth utilization between the RU (RC) and the EC $e \in \mathcal{E}$
CE_e	Total possible computing capacity of EC e for every time t
CR	Total possible computing capacity of the RC
$CB_{F,e}$ $(CB_{M,e})$	Total bandwidth of FH (MH) link related to $e \in \mathcal{E}$
$\delta_{r,e}$, $\delta_{e,\mathcal{R}}$	Delay parameters
P^{max}	Maximum power consumption of an EC
γ	Proportion of the consumed power of an idle server with respect to P_{max}
P_{net}^{max}	Maximum power consumption of network links
$P_{net,e}^{fix}$	Fixed power consumption of a network link between RU and $e \in \mathcal{E}$

Table 5.1: Description of problem parameters.

$$\sum_{s \in R(t)} \sum_{f \in F_s} x_{s,f}^e(t) c_{s,f} \leq CE_e, \ \forall e \in \mathcal{E}, \tag{5.3}$$

$$\sum_{s \in R(t)} \sum_{f \in F_s} y_{s,f}(t) c_{s,f} \leq CR. \tag{5.4}$$

Second, there exists bandwidth constraints over transmission links expressed as follows. Note that a slice uses a link between the RU and an EC, if its second VNF (i.e., with index 1) is placed in this EC.

$$\sum_{s \in R(t)} x_{s,1}^e(t) b_{s,0} \leq CB_{F,e}, \ \forall e \in \mathcal{E}, \tag{5.5}$$

$$\sum_{s \in R(t)} \sum_{f \in F_s \setminus \{n_s - 1\}} x_{s,f}^e(t) y_{s,f+1}(t) b_{s,f} \leq CB_{M,e}, \ \forall e \in \mathcal{E}. \tag{5.6}$$

Third, a VNF f of an admitted slice s is deployed at a unique physical resource, i.e.,

$$\sum_{e\in\mathcal{E}} x^e_{s,f}(t) + y_{s,f}(t) = sr_s(t),\ \forall s \in R(t), \forall f \in F_s \setminus \{0\}. \tag{5.7}$$

Fourth, the following constraints should be imposed on the order/way of the VNFs deployment:

As by convention, for every admitted slice s, its first VNF (v_0) is placed in the RU, thus it cannot be placed in an EC or the RC, i.e.,

$$\sum_{\forall e\in\mathcal{E}} x^e_{s,0}(t) = 0, \quad y_{s,0}(t) = 0,\ \forall s \in R(t). \tag{5.8}$$

Moreover, the second VNF v_1 of an admitted slice should be placed in an EC, which is guaranteed if it is not placed in the RC:

$$y_{s,1}(t) = 0,\ \forall s \in R(t). \tag{5.9}$$

In addition, under the assumptions of service chaining and collocation, if for an admitted slice s, a VNF f is placed in an EC, the VNFs preceding f in the service chain, i.e., those with IDs $1, \ldots, f-1$, should be also placed in the same EC. Similarly, if a VNF is placed in the RC, its successive VNFs in the service chain of the slice should be also placed in the RC. These constraints are expressed mathematically as follows:

$$x^e_{s,f}(t) \le x^e_{s,f-1}(t),\ \forall f \in F_s \setminus \{0,1\}, \forall s \in R(t), \forall e \in \mathcal{E}, \tag{5.10}$$

$$y_{s,f}(t) \le y_{s,f+1}(t),\ \forall f \in F_s \setminus \{n_s - 1\}, \forall s \in R(t). \tag{5.11}$$

Fifth, the delay of a slice is determined by the delays introduced by the FH and MH links and the maximum delay requirement of a slice s, $D_{max,s}$ is imposed by:

$$\sum_{e\in\mathcal{E}} x^e_{s,1}(t)\delta_{r,e} + \sum_{f\in F_s\setminus\{n_s-1\}} \sum_{e\in\mathcal{E}} x^e_{s,f}(t) y_{s,f+1}(t)\delta_{e,R} \le D_{\max,s},\ \forall s \in R(t). \tag{5.12}$$

Sixth, in this work, reallocation of VNFs for accepted slices is not allowed, which is imposed by:

$$x^e_{s,f}(t) \ge sr_s(t) \cdot x^e_{s.f}(t-1), \forall s \in R(t), \forall f \in F_s, \forall e \in \mathcal{E}, \tag{5.13}$$

$$y_{s,f}(t) \ge sr_s(t) \cdot y_{s,f}(t-1), \forall s \in R(t), \forall f \in F_s. \tag{5.14}$$

Finally, if a slice is admitted, then it retains its admission status until its holding time expires, i.e.,

$$X_s(t) \ge X_s(t-1),\ \forall s \in R(t). \tag{5.15}$$

5.4 Cost

The total revenue accumulated in the system at every timestep t is defined as:

$$ReV(t) = \sum_{s \in R(t)} sr_s(t) \cdot pr_s. \tag{5.16}$$

The cost of the network is derived from the power consumption of the ECs and the network links utilized for the slice deployment. To design the power consumption costs for the FH and MH links, two additional types of variables should be defined to determine their utilization. Let $u_e(t)$ be a binary variable expressing whether the FH link connecting the EC $e \in \mathcal{E}$ with the RU is utilized at t. This variable is set only if at least one slice has its VNF v_1 placed on the EC e, i.e.,

$$u_e(t) \geq x^e_{s,1}(t), \ \ \forall s \in R(t), \forall e \in \mathcal{E}. \tag{5.17}$$

Let $v_e(t)$ be a binary variable expressing whether the MH link through the EC $e \in \mathcal{E}$ is utilized at t . This variable is set if for any two successive VNFs of any slice, one is placed in the EC e and the other on the RC, i.e.,

$$v_e(t) \geq x^e_{s,f}(t) y_{s,f+1}(t), \ \ \forall s \in R(t), \forall f \in F_s \setminus \{n_s - 1\}, \forall e \in \mathcal{E}. \tag{5.18}$$

To express the power consumption cost of the ECs, the model of a power-efficient VNF placement from [19] is used, i.e.:

$$PC^{EC}(t) = \sum_{e \in \mathcal{E}} \left(u_e(t) \gamma P^{max} + (1 - \gamma) \frac{C_e(t)}{CE_e} P^{max} \right). \tag{5.19}$$

For the links that connect the RU with ECs:

$$PC^{RU-E}(t) = \sum_{e \in \mathcal{E}} \left(u_e(t) P^{fix}_{net,e} + \frac{B_{FH,e}(t)}{CB_{F,e}} P^{max}_{net} \right). \tag{5.20}$$

Finally, in case of links that connect ECs with the RC, it can be written as

$$PC^{E-R}(t) = \sum_{e \in \mathcal{E}} \left(v_e(t) P^{fix}_{net,e} + \frac{B_{MH,e}(t)}{CB_{M,e}} P^{max}_{net} \right). \tag{5.21}$$

5.5 MDP

Next, in order to develop the RL solution, the MDP must be formally defined.

In the proposed framework, slice requests are presented to the agent, which then determines whether to accept each one of them based on the order of their arrival times. Each RL decision step corresponds to one slice request. In instances where multiple slice requests arrive within the same time-slot, ties are resolved arbitrarily. To avoid confusion, the index k denotes the RL-based decision step for accepting or not a slice corresponding to a request, whereas t_k denotes the time slot of the control window at which the RL decision

k occurs. Multiple RL decision steps, indexed as $k, k+1, ..., m$ may refer to the same time slot $t_k = t_{k+1} = ... = t_m$ in the control window when concurrent slice request arrivals take place. Furthermore, slices that were not accepted are reintroduced to the agent at the next time slot, provided their holding time has not expired. In this case, these requests are treated as new slice requests, with their holding times appropriately reduced. The agent continues to process slice requests until there are no active slices left not-admitted or until the end of the episode, which spans H time-steps.

5.5.1 State

At each decision step, the agent requires sufficient information regarding both the current network state and the forthcoming slice request, upon which the decision will be made. Therefore, the state of the agent at each step k, when slice request s is processed, is represented by the following tuple:

$$(\mathbf{AC}_k, \mathbf{AB}_k, \mathbf{AT}_k, SI_k, t_k),$$

where:

- $\mathbf{AC}_k = [AC_1(k), ..., AC_{|E|}(k), AC_{RC}(k)]$, i.e. the available capacity of all ECs and of the RC,
- $\mathbf{AB}_k = [AB_{FH_1}(k), ..., AB_{FH_{|E|}}(k), AB_{MH_1}(k), ..., AB_{MH_{|E|}}(k)$, i.e. the available bandwidth capacity of all network links, FH and MH,
- $\mathbf{AT}_k = [AT_{EC_1}(k), ..., AT_{EC_{|E|}}(k), AT_{MH_1}(k), ..., AT_{MH_{|E|}}(k)]$, i.e. the remaining times that each EC or each MH link will remain active according to the current network configuration,
- $SI_k = (pr_s, D_{max,s}, ht_s, c_{s,1}, ..., c_{s,n_s-1}, b_{s,1}, ..., b_{s,n_s-1})$, i.e. all the necessary information of the slice to be processed,
- t_k: the current time slot of the control horizon.

5.5.2 Action

The agent will jointly decide whether to accept the slice, which EC will serve it if accepted, and the number of VNFs of the slice that will be placed on this EC (i.e. the function splitting). Specifically, the agent makes the decision (e, v), where $e \in \{1, ..., |E|\}$ represents the index of the chosen EC, and $v \in \{1, ..., n_s - 1\}$ corresponds to the number of VNFs to be placed on the edge (with v_0 always assigned to the RU). To align with the actor network, this action pair is mapped to the one-dimensional space through the function $f(e, v) = (e-1)(n_s - 1) + v$. Therefore, for every slice request s processed at t_k (with $t_s \leq t_k$) the agent takes action a_k, from which (e_k, v_k) is calculated by the inverse mapping of $f(e, v)$. The action $a_k = 0$ corresponds to slice rejection at time t_k. In this case, its holding time is decreased by 1. If the updated holding time reaches zero, the slice is permanently rejected from the network, otherwise the slice will be reintroduced to the agent at the next time slot of the horizon, at which point a new independent decision will be made.

5.5.3 Reward

The following values are defined:

- $ReV_s = pr_s \cdot ht_s$ is the total revenue obtained by the acceptance of slice,

- PC_s^{EC} is the power consumption on the chosen EC:

$$\begin{aligned} PC_s^{EC} =& \max\{ht_s - AT_{EC,e_k}(k), 0\}\gamma P^{max} \\ &+ (1-\gamma)\frac{\sum_{f=1}^{v_k} c_{s,f}}{CE_{e_k}} P^{max} ht_s, \end{aligned} \tag{5.22}$$

where AT_{EC,e_k} is obtained from the current state s_k,

- PC_s^{RU-E} is the power consumption of the used FH link:

$$\begin{aligned} PC_s^{RU-E} =& \max\{ht_s - AT_{EC,e_k}(k), 0\}P_{net,e}^{fix} \\ &+ \frac{b_{s,1}}{CB_{F,e}} P_{net}^{max} ht_s, \end{aligned} \tag{5.23}$$

- PC_s^{E-R} is the power consumption of the used MH link:

$$\begin{aligned} PC_s^{E-R} =& \mathbf{1}_{\{v_k < n_s - 1\}} \Big[\max\{ht_s - AT_{MH,e_k}(k), 0\}P_{net,e}^{fix} \\ &+ \frac{b_{s,v_k+1}}{CB_{M,e}} P_{net}^{max} ht_s \Big], \end{aligned} \tag{5.24}$$

where $\mathbf{1}_{\{v_k < n_s - 1\}} = 1$ if $v_k < n_s - 1$ and 0 otherwise (the slice is entirely placed on EC, no MH link is used). The above equations for the power consumption cost are based on the model of a power-efficient VNF placement from [19].

The EC's idle power cost is assigned to the first slice placed on the EC, or to the one extending the EC's active time, thus the use of the AT_x values.

The agent needs to encourage actions that will result in higher reward, while simultaneously aiming for lower power consumption cost. Thus, the reward function for every state s_k, taking action a_k for slice s is formulated as:

$$R(s_k, a_k) = \begin{cases} ReVs - PC_s, & \text{if slice } s \text{ is accepted at step } k, \\ 0, & \text{if slice } s \text{ is rejected at step } k, \end{cases} \tag{5.25}$$

where $PC_s = PC_s^{EC} + PC_s^{RU-E} + PC_s^{E-R}$.

5.5.4 Transition Function

A transition takes place after every action a_k to modify the network state according to the action. The transitions are defined per RL decision step for each slice request and not per time slot. Consequently, multiple updates may occur within a single time-slot of the control horizon, depending on the number of slice requests that arrive during the corresponding time-slot. Following each decision, the network state—comprising the capacities of nodes and links—is updated to reflect the admission of the slice, if applicable, and to adjust the holding times of all active slices.

5.6 DRL Solution

The proposed solution to the online slice admission and function splitting problem, as formulated by the above MDP, is the PPO algorithm, as detailed in paragraph 4.7.2.1. This algorithm is aligned with the discrete action space of the problem. A modified variant of PPO, referred to as Maskable PPO, is employed, as implemented in the Python library Stable Baselines3 [20]. This variant incorporates action masking to exclude actions that violate constraints at each time step, thereby simplifying the decision-making process for the agent. Actions are considered violating if they fail to satisfy the capacity and delay requirements constraints, defined in section 5.3.

Chapter 6

Experimental Results

In this chapter, the potential of the developed RL agent will be demonstrated through a series of experiments, detailing the behavior of the model during training and evaluating its performance across multiple metrics.

6.1 Evaluation Metrics

Different evaluation metrics are incorporated to assess the performance of the model, each providing a different perspective on its capabilities. These are:

• **Objective Value:** It represents the total gain received by the agent at every step. It is defined by the total revenue obtained by the slice admittance (equation 5.16), reduced by the power consumption incurred in the resulting network state (equations 5.19-5.21). This metric reflects the trade-off between minimizing power consumption and maximizing overall reward, either by increasing the acceptance ratio or by prioritizing slice requests with higher priority.

• **Acceptance Ratio:** It measures the ratio of admitted slices, by a certain time step, to the active slice subset, which includes ongoing requests yet to expire. It reflects the system's efficacy in handling incoming slice demands midst existing deployment commitments.

• **Power Efficiency:** It is the ratio of the sum of accepted slices over the total power consumption of the compute and network counterparts of the substrate network. It gives an insight into the fraction of total power consumption corresponding to each accepted slice.

To demonstrate the results, the cumulative average value of these metrics is used. This is calculated by averaging the values across test episodes for each time-step, followed by taking the cumulative sum, and then using these cumulative values to compute the corresponding ratios.

6.2 Training- Normal Operation

In the table 6.1 the values for the network infrastructure and slice requests parameters are presented. Two types of slices with different requirements are considered, namely URLLC and eMBB.

Parameter	**Value**
Number of ECs	3
EC, RC capacity	$64, 256$ cores
Fronthaul link capacity, delay	2Gbps, 4ms
Midhaul link capacity, delay	4Gbps, 8ms
γ	0.8
P^{max}, P^{max}_{net}	200 W
P^{fix}_{net}	160 W
Number of Requests	20
SFC request length	8 VNFs
CPU cores demand per VNF	$\in \{2, 4, 8\}$
Request holding time	$\min(U\{3, 6\}, 12 - t_s)$
URLLC and eMBB bandwidth requirement	100, 200 Mbps
URLLC and eMBB delay requirement	25, 50 ms
URLLC and eMBB normalized priority	3, 2.4 per time-slot
Optimization Horizon H	12 time steps

Table 6.1: Simulation Parameters.

Initially, the agent's operation will be demonstrated under a scenario where slice requests are generated according to a normal distribution over the horizon H, specifically $\mathcal{N}\left(\frac{H}{2} - 1, 0.9\right)$.

The agent is trained over 3500 episodes, and the progression of the training process can be monitored using TensorBoard. As illustrated in Figure 6.1, the loss, defined as the average of the training loss and the value loss of the actor and critic networks, respectively, decreases over time, stabilizing at around step 90,000. This step corresponds to approximately episode 3000, after which the agent's behavior appears to stabilize.

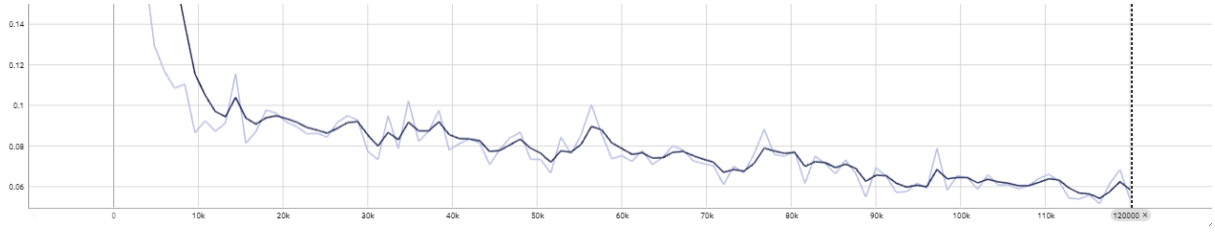
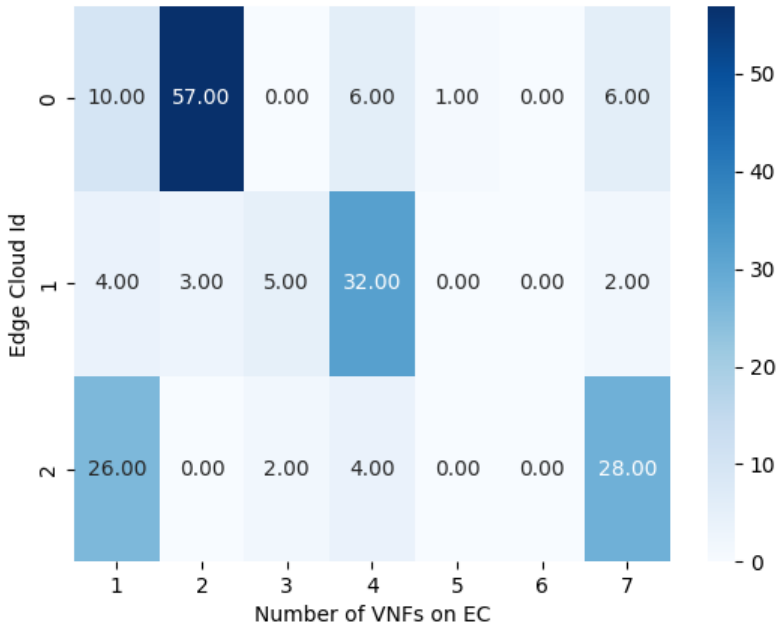
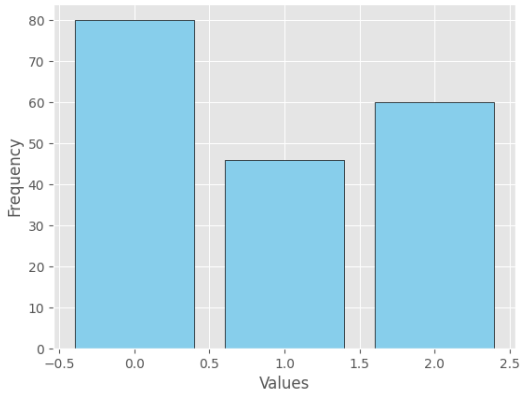


Figure 6.1: Loss during training.

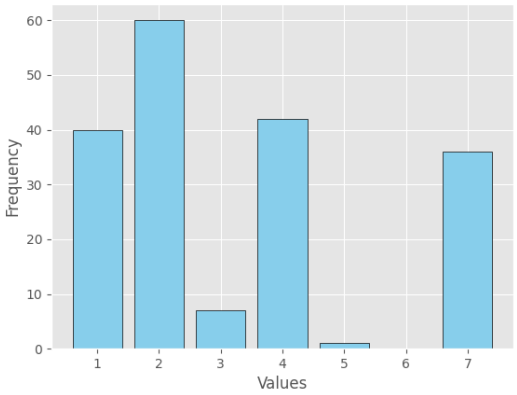
To further enhance understanding of the agent's decision-making process, the decisions made by the agent during a series of 10 test episodes are visualized. Figure 6.2 (a) illustrates a heatmap with the frequency of every EC id- number of VNFs pair and in the figures 6.2 (b-c) the frequency for every separate decision is more clearly depicted.



(a) Heatmap with the the values for each pair of decisions, EC, VNF.



(b) Decision for EC.



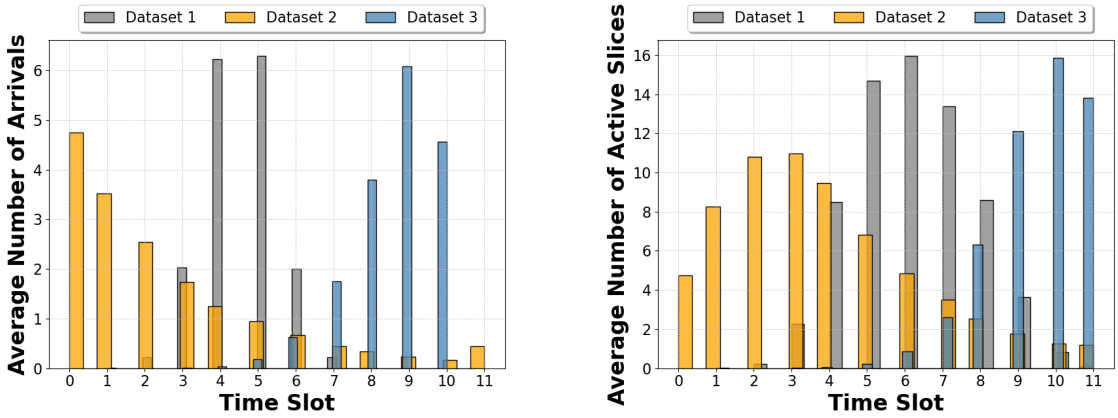
(c) Decision for number of VNFs on EC.

Figure 6.2: Decisions of the agent.

These results can now be analyzed. First of all, a degree of imbalance in the selection of ECs is evident, with a preference for the first EC. This aligns with the goal of minimum power consumption (defined by the term of the reward function), as it is advantageous to keep as many ECs idle as possible, thereby concentrating the requests on the minimal number of ECs required. Similarly, the agent appears to favor splitting the slice on a

lower number of VNFs. This is also justified by the attempt to minimize power consumption, given that VNFs deployed on the RC do not contribute to the total computational consumption (they only increase the MH link consumption, if the slice extends its active period). Additionally, the choice of no splitting (i.e., assigning all 7 VNFs to a single EC) seems to be frequent. This corresponds to cases where the slices can be fully accommodated within the selected EC and either have a low total load and holding time or they are accepted near the end of the control horizon, when no further requests are expected. In these situations, the agent prefers to avoid splitting, thereby avoiding the power consumption associated with the MH link. Finally, it is worth noting that the no-splitting decisions are usually associated with the third EC, indicating an effort to reserve sufficient capacity in the other ECs to concentrate the upcoming split slices, thereby minimizing again the MH link power consumption cost. Overall, the agent exhibits the expected behavior.

6.3 Evaluation on different arrival distributions



(a) Average number of arrivals.

(b) Average number of active slices.

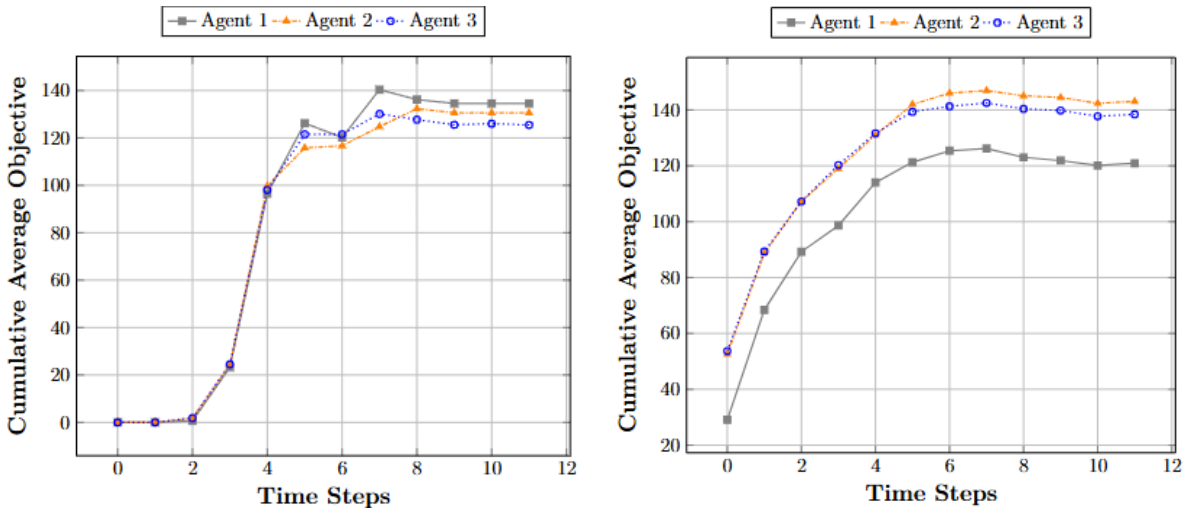
Figure 6.3: Dataset information.

In the first set of experiments, 3 datasets are created with varying arrival rates per time slot in the control horizon. Specifically, arrival patterns are generated according to three distinct distributions, visualized in Figure 6.3: a Normal distribution, $\mathcal{N}\left(\frac{H}{2}-1, 0.9\right)$, an Exponential distribution, $\text{Exp}\left(\frac{4}{H}\right)$, and a Beta distribution, $\text{Beta}\left(H-2, \frac{H}{6}\right)$ rescaled to the horizon interval and grouped by time slot. For each distribution, $1,000$ scenarios are used for the training dataset and 10 for the test set. This experiment targets to evaluate the agent's learning ability by testing its behavior on diverse slice arrival distributions.

The agent is trained on one of the datasets at a time. In particular, Agent 1 represents the model trained on Dataset 1, Agent 2 on Dataset 2, and Agent 3 on Dataset 3. These agents are then evaluated across all three datasets, as illustrated in Figure 6.5(a). The objective value, as defined in section 6.1, is used as evaluation metric.

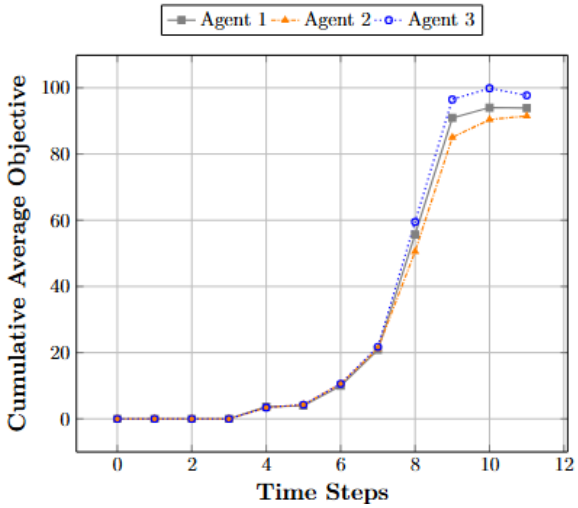
In each case, as observed in Figure 6.4, the agent performs best on the dataset it was trained on, demonstrating its ability to learn the arrival patterns and optimize decision-making for each scenario. However, the observed divergences are small, indicating strong

generalization capabilities.



(a) Cumulative average objective value on test dataset 1.

(b) Cumulative average objective value on test dataset 2.

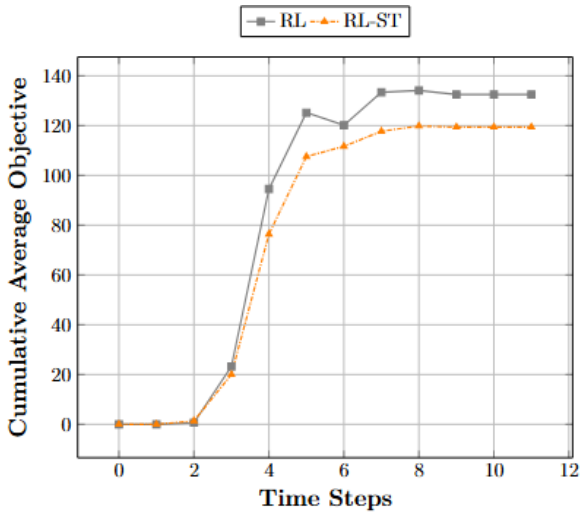


(c) Cumulative average objective value on test dataset 3.

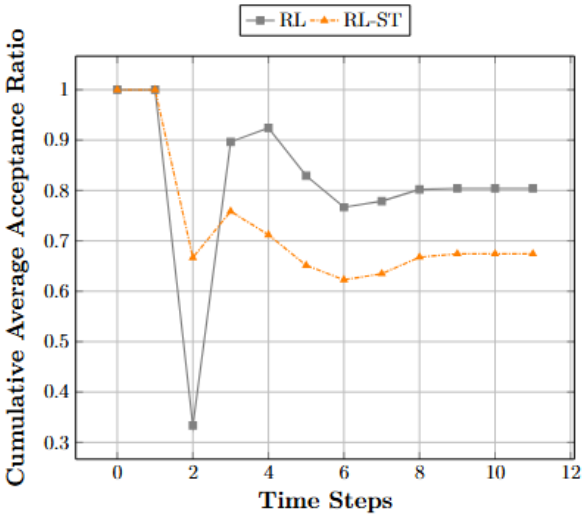
Figure 6.4: Comparative evaluation under varying arrival patterns.

6.4 Comparison with static splitting agent

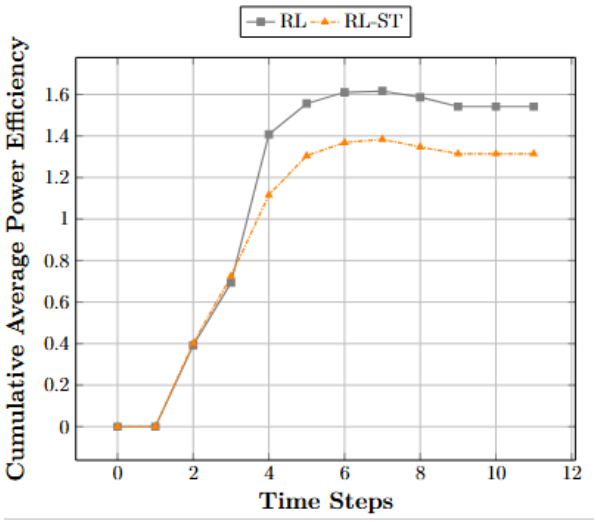
The second set of experiments is designed to assess the agent's ability to take the optimal splitting decision. To this end, the agent is compared against a static version, denoted by "RL-ST". This version splits deterministically the slices. In particular, the emBB slices are split in their second VNF (v_1) and the URLLC ones are split in the middle (v_3). The decision of the agent is thus limited to the selection of the EC. The two agents are trained and tested on sets sampled by the distribution on Dataset 1.



(a) Cumulative average objective value.



(b) Cumulative average acceptance ratio.



(c) Cumulative average power efficiency.

Figure 6.5: Comparative evaluation dynamic vs static splitting.

Figure 6.5 illustrates the performance of the two models, as evaluated using the metrics defined in Section 6.1. The "RL-ST" method, which lacks adaptability, results in suboptimal load management and an increased number of rejections, leading to a noticeable deterioration in performance across all evaluated metrics. This outcome highlights the critical importance of dynamic decision-making of the proposed RL model in effectively managing incoming requests. This model exhibits strong state awareness, enabling it to

adapt to varying network conditions and optimize performance.

Chapter 7

Epilogue

The advent of 5G and eventually 6G has given rise to growing demands in network resources from multiple organizations or individuals, with heterogeneous requirements for the demanded services. The virtualization of the network functions and the evolution of the RAN part of the network into the disaggregated O-RAN architecture allow for the incorporation of intelligent mechanisms to manage the network traffic.

This thesis presents an innovative solution to the challenge of managing incoming slice requests in an O-RAN network using a Reinforcement Learning approach. The primary focus of the work was on addressing the dynamic decision-making problem in Network Slicing and VNF Placement in varying network conditions. Taking into account the on-line nature of the problem, the solution from the corresponding exhaustive optimization task would be of little value. The proposed solution leverages the Proximal Policy Optimization (PPO) algorithm to train a reinforcement learning agent capable of making dynamic decisions given the current network state at any time. The agent aims to optimize the trade-off between revenue from slice acceptance and the power consumption costs at edge clouds and communication links, both of which are critical for improving energy efficiency in modern wireless networks.

Through a series of experimental evaluations, the performance of the proposed RL agent was assessed across several key metrics. These experiments highlighted the significance of dynamic decision-making in achieving effective resource management and the ability of the agent to make insightful decisions, adapting to the network conditions. The agent showed robust performance across different network scenarios, illustrating the model's ability to satisfy a high number of incoming requests, while simultaneously maintaining low power consumption costs.

Future Work

While this thesis offers a solid foundation for the aforementioned problem, several potential extensions for future work can be identified. First, it is crucial to evaluate the agent on a real dataset of slice requests and VNFs. Such a dataset, which includes the required capacity, bandwidth, and delay specifications for each VNF within every slice, has yet to be developed.

Second, testing the agent in real-world, large-scale environments with more RUs, ECs and RCs, would be highly beneficial. Additionally, relaxing the assumption of one DU per EC, which was made in the current work, would provide a more realistic evaluation of the agent's performance.

Finally, this work could be further improved by incorporating a mechanism for slice reallocation, where already deployed slices are dynamically moved across the network nodes in order to minimize power consumption.

In sum, as networks continue to grow in complexity and scale, solutions like the one proposed in this thesis will play a crucial role in ensuring that resources are used efficiently, while simultaneously minimizing environmental impact through energy conservation.

Bibliography

[1] O.-R. ALLIANCE, "O-ran architecture overview," 2024.

[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[3] G. Heine and H. Sagkob, *GSM Networks: Protocols, Terminology, and Implementation*. Artech House, 1998.

[4] J. L. Frauendorf and É. A. de Souza, *The Architectural and Technological Revolution of 5G*. Cham, Switzerland: Springer, 2023.

[5] M. A. Hasabelnaby, M. Obeed, A. Chaaban, M. Saif, and M. J. Hossain, "From centralized ran to open ran: A survey on the evolution of distributed antenna systems," *arXiv preprint arXiv:2411.12166*, Nov 2024. [cs.ET].

[6] F. Giannone, K. Kondepu, H. Gupta, F. Civerchia, P. Castoldi, A. Antony Franklin, and L. Valcarenghi, "Impact of virtualization technologies on virtualized ran midhaul latency budget: A quantitative experimental evaluation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 604–607, 2019.

[7] P. v. Anvith, N. Gunavathi, B. Malarkodi, and B. Rebekka, "A survey on network functions virtualization for telecom paradigm," in *2019 TEQIP III Sponsored International Conference on Microwave Integrated Circuits, Photonics and Wireless Networks (IMICPW)*, pp. 302–306, 2019.

[8] P. Subedi, A. Alsadoon, P. W. C. Prasad, *et al.*, "Network slicing: A next generation 5g perspective," *Journal of Wireless Communications and Networking*, vol. 2021, p. 102, 2021.

[9] G. Americas, "Transition toward open and interoperable networks," 2025. White Paper.

[10] A. Garcia-Saavedra and X. Costa-Pérez, "O-ran: Disrupting the virtualized ran ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, 2021.

[11] O.-R. Alliance, "O-ran: Towards an open and smart ran," 2018.

[12] O. Alliance, "O-ran fronthaul working group. control, user and synchronization plane specification (o-ranwg4-cus. 0-v03. 00)," *Tech. Spec., Apr*, 2020.

[13] O. Alliance, "O-ran working group 2. ai/mlworkflow description and requirements (o-ran. wg2. aiml-v01. 02)," tech. rep., Tech. Rep, 2021.

[14] G. Americas, "Understanding open ran," 2020.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. n/a, second edition, in progress ed., 2014. © 2014, 2015.

[16] OpenAI, "Spinning up in deep rl: Introduction to reinforcement learning," 2025. Accessed: 2025-01-29.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] G. B. Giorgos Gotzias, Nikolaos Fryganiotis *et al.*, "Towards slice admission control and split in o-ran using reinforcement learning," *Institute of Communication and Computer Systems, School of Electrical & Computer Engineering, National Technical University of Athens (NTUA)*, 2025.

[19] A. Varasteh, B. Madiwalar, *et al.*, "Holu: Power-Aware and Delay-Constrained VNF Placement and Chaining," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1524–1539, 2021.

[20] A. G. A. Raffin, A. Hill *et al.*, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[21] Qualcomm, "Fundamentals of wireless signals and cellular networks," October 2019.

[22] P. Wireless, "Openran 101 series - role of ric," 2025.

[23] Ericsson, "Evaluating processing options in ran: Purpose-built vs. cloud ran," September 2020.

[24] I. A. Alimi, A. L. Teixeira, and P. P. Monteiro, "Toward an efficient c-ran optical fronthaul for the future networks: A tutorial on technologies, requirements, challenges, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, 2018.

[25] O. B. S. A. I. (OBSAI), *OBSAI System Reference Document, Version 2.0*. OBSAI, 2006.

[26] O.-R. Alliance and E. T. C. M. S. G. (MSG), "Publicly available specification (pas): O-ran fronthaul management plane specification v12.01," 2025.

[27] A. Akman, S. Gu, J. Huang, I. Wong, H. Akhtar, S. Poretsky, J. Balas, Q. Sun, B. Yargicoglu-Sahin, Y. Xie, J. Voigt, G. Hannak, B. Raghothaman, S. Nanba, K. Tang, J. Chou, Y. Koral, B. Yang, Y. Liu, J. Song, S. Singh, S. Narayanan, L. Ong, R. Jana, and A. Buldorini, "O-RAN Minimum Viable Plan and Acceleration towards

Commercialization," *RS Open Journal on Innovative Communication Technologies*, vol. 4, nov 6 2023. https://rs-ojict.pubpub.org/pub/gsry0u68.

[28] R. Hat, "Open ran and o-ran: A brief," 2025.

[29] Ericsson, "Ericsson mobility report," November 2022.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.