



AI-Driven Orchestration and Management of Cloud-Native 6G Networks

by

Sokratis Christakis

Supervisors:

Athanasios Korakis, Professor, University of Thessaly
Serge Fdida, Professor, Sorbonne University

Committee:

Athanasios Korakis, Professor, University of Thessaly (Supervisor)
Serge Fdida, Professor, Sorbonne University (Supervisor)
Antonios Argyriou, Professor, University of Thessaly
Fotios Plessas, Professor, University of Thessaly
Paris Flegkas, Assistant Professor, University of Thessaly
Jim Kurose, Professor, University of Massachusetts Amherst
Thi-Mai-Trang Nguyen, Professor, Université Sorbonne Paris Nord

October 2026

UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AI-Driven Orchestration and Management of Cloud-Native 6G Networks

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Sokratis Christakis

October 2026

UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Orchestration et gestion intelligentes des réseaux 6G cloud-natifs fondées sur l'intelligence artificielle

Thèse présentée pour l'obtention du diplôme de Doctorat

Sokratis Christakis

October 2026

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ενορχήστρωση και Διαχείριση Cloud-Native Ασυρμάτων Δικτύων Έκτης Γενιάς με Τεχνητή Νοημοσύνη

Διατριβή η οποία υποβλήθηκε για τη μερική εκπλήρωση
των υποχρεώσεων απόκτησης του Διδακτορικού Διπλώματος

Σωκράτης Χριστάκης

October 2026





AI-Driven Orchestration and Management of Cloud-Native 6G Networks

PhD Dissertation

Sokratis Christakis

Advisory committee

Athanasios Korakis, Professor, University of Thessaly (Supervisor)

Serge Fdida, Professor, Sorbonne University (Supervisor)

Examination committee

Antonios Argyriou, Professor, University of Thessaly

Fotios Plessas, Professor, University of Thessaly

Paris Flegkas, Assistant Professor, University of Thessaly

Jim Kurose, Professor, University of Massachusetts Amherst

Thi-Mai-Trang Nguyen, Professor, Université Sorbonne Paris Nord

October 2026

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this PH.D. dissertation, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The declarant



Sokratis Christakis

Acknowledgments

This thesis marks the end of a long and rewarding journey between Volos and Paris, and it would not have been possible without the people who walked parts of it with me.

I am deeply grateful to my supervisors, Professor Thanasis Korakis of the University of Thessaly and Professor Serge Fdida of Sorbonne Université. Professor Korakis welcomed me into NITLab and gave me my first real contact with experimental research. His trust allowed me to work on real testbeds from the very beginning, and his guidance taught me to look at every problem from the point of view of a system that has to work in practice. Professor Fdida welcomed me to LIP6 and made my time in Paris both productive and inspiring. His clear vision of where networking research is heading, his openness to new directions and his steady encouragement shaped the way I think as a researcher. Through him I became part of a wide European research community, which broadened the scope of this thesis well beyond what I had imagined when I started. I am especially thankful for the trust he showed in me, for the time he always found for our discussions despite his many responsibilities, and for the example he set of how to combine scientific rigor with a genuinely collaborative spirit.

I would also like to sincerely thank the members of the examination committee for the time and care they devoted to reading this thesis, for their valuable comments, and for the stimulating discussion during the defense.

A special thank you goes to Dr. Nikos Makris, who has been present at every stage of this work. From the first experiments to the final chapters, his technical insight, his ideas and his patience had a direct impact on the quality of this thesis. Beyond research, he was always available with advice and support whenever I needed it, and I am truly thankful for that.

At NITLab, my first thanks go to Dimitrios Kefalas, with whom I shared this entire journey. We started together, went through the joint PhD side by side, and shared the long days in the lab, the deadlines and the moves between Volos and Paris. I am also grateful to Theodoros Tsourdinis for his collaboration, his help and his friendship throughout these years.

I would also like to thank Dimitris Kanavaris, Stergios Lantzos, Efraim Pavlidis, Marialena Lefkopoulou, Katerina Kyriakou, Giorgos Labis, Alexandros Stoltidis, Ippokratis Koukoulis and many more colleagues at NITLab for the good collaboration and the good moments inside and outside the lab, as well as the postdoctoral researchers Virgilios Passas, Ilias Syrigos, Kostas Chounos, Kostas Choumas and Apostolis Apostolaras for their advice and support. I also thank Stavroula Maglavera and the NITLab administrative team, Katerina Arvaniti, Sissy Magouritsa, Olina Stergiopoulou and Stergios Avlogiaris. I am also grateful to my colleagues at LIP6, Hassane Rahich, Frederic Vaissade, Taha Mohsen Zaidi and Bilel Zaghoudi, who made Paris feel like a second home.

I owe a special thank you to Emilie Mespoulhes at LIP6. From my very first days in Paris she guided me through every administrative step of the joint PhD, from registration and paperwork to finding my way in a new city, always with patience and kindness.

Above all, I want to thank my family my parents, Alexios Christakis, Aliki Eftaxia and my brother, Charilaos Christakis, for their unconditional love and support. They believed in me at every step, and none of this would have been possible without them.

Finally, I want to thank my friends and all the people who have been meaningful to me along the way: Panagiotis Kostoulas, Vera Lekka, Konstantinos Polizos, Vasilis Lignos, Pavlos Eggonidis, Ioannis Tselios, Alexandros Palatzidis, Fotis Tolis and many more.

PhD Dissertation

AI-Driven Orchestration and Management of Cloud-Native 6G Networks

Sokratis Christakis

Abstract

The sixth generation of mobile networks is set to redefine the boundaries of wireless connectivity, enabling applications that range from holographic telepresence and pervasive AI-driven services to seamless integration of terrestrial, aerial, and satellite access into a unified network fabric. Realizing this vision demands not only advances at the Radio Access Network (RAN), but a fundamental rethinking of how the entire network infrastructure is designed, deployed, and managed. 6G envisions core networks that are intelligent, energy-efficient, and capable of adapting autonomously to heterogeneous and ever-changing demands, requirements that push beyond the capabilities of current deployments and call for entirely new orchestration mechanisms and management paradigms. Addressing these challenges requires not only advances in core network design, but also the experimental infrastructure and tools needed to develop and validate them at scale.

In this thesis, we begin by establishing an empirical foundation for 5G/6G core network management through a systematic real-world evaluation of four open-source UPF implementations. By characterizing their performance and resource trade-offs under identical real 5G traffic conditions, we reveal insights that theoretical analysis cannot capture and that are essential for any subsequent management or orchestration effort.

Building on this, we investigate how cloud-native principles can be applied to dynamically scale core network functions in response to real-time traffic demand. We design and evaluate a reactive scaling mechanism for Kubernetes-based 5G deployments, demonstrating that elastic core network management concretely outperforms static provisioning in both network performance and resource efficiency.

We then advance toward the proactive and sustainable orchestration that 6G demands. By integrating traffic prediction with energy-aware deployment optimization across hetero-

geneous network function implementations, we show that intelligent orchestration can simultaneously satisfy performance requirements and reduce energy consumption, demonstrating that sustainability in the 6G core is an achievable design objective rather than an aspirational one.

The fourth direction addresses one of the defining features of the 6G connectivity vision: the seamless integration of heterogeneous access technologies into a unified network fabric. 6G envisions users simultaneously exploiting multiple radio access paths, from cellular to Wi-Fi to satellite, with the network dynamically coordinating traffic across them. We experimentally investigate how this vision can be realized in practice, evaluating real-world multi-access integration approaches and multipath transport protocols over real-radio channels, revealing fundamental trade-offs that have direct implications for the design of heterogeneous access management in beyond-5G and 6G networks.

Finally, we address the accessibility of the experimental research infrastructure that underpins all of these advances. Large-scale research platforms are essential for validating 6G research, yet their complexity creates steep learning curves that limit accessibility and slow the research cycle. We present slAIces, an LLM-based platform that combines Retrieval-Augmented Generation and fine-tuning to assist researchers in navigating and operating the SLICES-RI infrastructure, significantly lowering the barrier for new users and enabling end-to-end experiment automation through natural language alone.

Together, these contributions form a coherent research program toward intelligent, sustainable, and accessible 5G/6G core network management — from empirical characterization and elastic orchestration, to heterogeneous access and AI-driven experimentation.

Keywords: 5G; 6G; Core Network Management; User Plane Function; Cloud-Native Orchestration; Energy Efficiency; Machine Learning; ATSSS; MPTCP; MPQUIC; Non-3GPP Access; Large Language Models; RAG; SLICES-RI; OpenAirInterface.

Διδακτορική Διατριβή

Ενορχήστρωση και Διαχείριση Cloud-Native Ασυρμάτων Δικτύων Έκτης Γενιάς με Τεχνητή Νοημοσύνη

Σωκράτης Χριστάκης

Περίληψη

Η έκτη γενιά κινητών δικτύων (6G) αναμένεται να επαναπροσδιορίσει τα όρια της ασύρματης συνδεσιμότητας, υποστηρίζοντας εφαρμογές που εκτείνονται από την ολογραφική τηλεπαρουσία (holographic telepresence) και τις διάχυτες υπηρεσίες τεχνητής νοημοσύνης (pervasive AI-driven services) έως την απρόσκοπτη ενοποίηση επίγειων, εναέριων και δορυφορικών τεχνολογιών πρόσβασης σε ένα ενιαίο δικτυακό περιβάλλον. Η υλοποίηση αυτού του οράματος απαιτεί όχι μόνο σημαντικές εξελίξεις στο Δίκτυο Ραδιοπρόσβασης (Radio Access Network — RAN), αλλά και έναν θεμελιώδη επανασχεδιασμό του τρόπου με τον οποίο σχεδιάζονται, αναπτύσσονται και διαχειρίζονται οι συνολικές δικτυακές υποδομές. Τα δίκτυα κορμού (Core Networks) του 6G προβλέπεται να είναι ευφυή, ενεργειακά αποδοτικά και ικανά να προσαρμόζονται αυτόνομα σε ετερογενείς και διαρκώς μεταβαλλόμενες απαιτήσεις, χαρακτηριστικά που υπερβαίνουν τις δυνατότητες των σημερινών δικτύων και καθιστούν αναγκαία την ανάπτυξη νέων μηχανισμών ενορχήστρωσης (orchestration) και διαχείρισης (management). Η αντιμετώπιση αυτών των προκλήσεων προϋποθέτει όχι μόνο καινοτομία στον σχεδιασμό των δικτύων κορμού, αλλά και την ύπαρξη πειραματικών υποδομών και εργαλείων που θα επιτρέψουν την ανάπτυξη και αξιολόγησή τους σε μεγάλη κλίμακα.

Στην παρούσα διατριβή, αρχικά θεμελιώνουμε μία εμπειρική βάση για τη διαχείριση δικτύων 5G/6G μέσω μιας συστηματικής αξιολόγησης τεσσάρων υλοποιήσεων ανοικτού κώδικα της Συνάρτησης Επιπέδου Χρήστη (User Plane Function — UPF) σε πραγματικές συνθήκες λειτουργίας δικτύου 5G. Μέσα από τη συγκριτική ανάλυση της απόδοσης και της κατανάλωσης υπολογιστικών πόρων υπό ταυτόσημες συνθήκες πραγματικής κυκλοφορίας, αναδεικνύονται κρίσιμοι συμβιβασμοί απόδοσης και πόρων (performance-resource trade-offs) που δεν μπορούν να αποτυπωθούν επαρκώς μέσω θεωρητικής ανάλυσης και οι οποίοι αποτελούν βασική προϋπόθεση για κάθε μεταγενέστερο μηχανισμό ενορχήστρωσης και δια-

χείρισης συναρτήσεων δικτύου (Network Functions).

Στη συνέχεια, διερευνούμε την εφαρμογή αρχών cloud-native στη δυναμική κλιμάκωση (dynamic-scaling) δικτυακών συναρτήσεων με βάση τη ζήτηση κυκλοφορίας σε πραγματικό χρόνο. Σχεδιάζουμε και αξιολογούμε έναν αντιδραστικό μηχανισμό κλιμάκωσης (reactive scaling mechanism) για αναπτύξεις 5G βασισμένες στο Kubernetes, αποδεικνύοντας ότι η ελαστική διαχείριση του δικτύου κορμού (elastic core network management) υπερτερεί ουσιαστικά της στατικής δέσμευσης πόρων (static resource provisioning) τόσο ως προς την απόδοση του δικτύου όσο και ως προς την αποδοτικότητα χρήσης υπολογιστικών πόρων.

Έπειτα, προχωρούμε προς την προδραστική και βιώσιμη ενορχήστρωση (proactive and sustainable orchestration) που απαιτούν τα μελλοντικά δίκτυα 6G. Μέσω της ενσωμάτωσης μηχανισμών πρόβλεψης δικτυακής κίνησης (traffic prediction) και ενεργειακά αποδοτικής βελτιστοποίησης τοποθέτησης και κλιμάκωσης (energy-aware placement and scaling) ετερογενών υλοποιήσεων UPF, δείχνουμε ότι η ευφυής ενορχήστρωση μπορεί να ικανοποιεί ταυτόχρονα τις απαιτήσεις απόδοσης και να μειώνει την κατανάλωση ενέργειας έως και κατά 51%. Το αποτέλεσμα αυτό αποδεικνύει ότι η βιωσιμότητα των δικτύων κορμού 6G (6G Core Networks) μπορεί να αποτελέσει ρεαλιστικό και εφικτό στόχο σχεδιασμού.

Η τέταρτη ερευνητική κατεύθυνση εστιάζει σε ένα από τα πλέον χαρακτηριστικά στοιχεία του οράματος συνδεσιμότητας του 6G: την απρόσκοπτη ενοποίηση πολλαπλών τεχνολογιών πρόσβασης (multi-access integration) σε ένα ενιαίο δικτυακό περιβάλλον. Τα δίκτυα 6G αναμένεται να επιτρέπουν στους χρήστες την ταυτόχρονη αξιοποίηση πολλαπλών τεχνολογιών πρόσβασης, όπως κυψελοειδή δίκτυα (cellular networks), Wi-Fi και δορυφορικές επικοινωνίες (satellite communications), με το δίκτυο να κατανέμει δυναμικά την κίνηση μεταξύ αυτών των διαδρομών. Στο πλαίσιο αυτό, διερευνούμε πειραματικά τρόπους πρακτικής υλοποίησης της ενοποίησης πολλαπλών τεχνολογιών πρόσβασης (multi-access integration), αξιολογώντας διαφορετικές προσεγγίσεις ενοποίησης πρόσβασης (access integration) και πρωτόκολλα μεταφοράς πολλαπλών διαδρομών (multipath transport protocols) σε πραγματικά ασύρματα κανάλια. Η ανάλυση αναδεικνύει θεμελιώδεις συμβιβασμούς (trade-offs) που έχουν άμεσες επιπτώσεις στον σχεδιασμό μηχανισμών διαχείρισης ετερογενούς πρόσβασης (heterogeneous access management) για δίκτυα πέραν του 5G και του 6G.

Τέλος, εξετάζουμε το ζήτημα της προσβασιμότητας των ερευνητικών πειραματικών υποδομών που υποστηρίζουν τις παραπάνω εξελίξεις. Παρότι οι μεγάλης κλίμακας ερευνητικές πλατφόρμες είναι απαραίτητες για την αξιολόγηση τεχνολογιών 6G, η αυξημένη πολυπλο-

κότητά τους δημιουργεί σημαντικά εμπόδια για νέους ερευνητές και επιβραδύνει τον κύκλο ανάπτυξης και πειραματισμού. Για την αντιμετώπιση αυτής της πρόκλησης παρουσιάζουμε το slAIces, μία πλατφόρμα βασισμένη σε Μεγάλα Γλωσσικά Μοντέλα (Large Language Models — LLMs), η οποία συνδυάζει Ανάκτηση-Επαυξημένη Παραγωγή (Retrieval-Augmented Generation — RAG) και fine-tuning προκειμένου να υποστηρίζει τους ερευνητές στη χρήση και διαχείριση της ερευνητικής υποδομής SLICES-RI. Η πλατφόρμα μειώνει σημαντικά την καμπύλη εκμάθησης για νέους χρήστες και επιτρέπει την πλήρη αυτοματοποίηση πειραμάτων μέσω φυσικής γλώσσας. Συνολικά, οι συνεισφορές της παρούσας διατριβής συνθέτουν ένα συνεκτικό ερευνητικό πλαίσιο για την ευφυή (intelligent), βιώσιμη (sustainable) και προσβάσιμη (accessible) διαχείριση δικτύων 5G/6G — από την εμπειρική αξιολόγηση και την ελαστική ενορχήστρωση (elastic orchestration), έως την ετερογενή πρόσβαση (heterogeneous access) και την έρευνα καθοδηγούμενη από τεχνητή νοημοσύνη (AI-driven experimentation).

Λέξεις-κλειδιά: 5G; 6G; Διαχείριση Core Network; User Plane Function (UPF); Cloud-Native Orchestration; Ενεργειακή Αποδοτικότητα; Machine Learning (ML); ATSSS (Access Traffic Steering, Switching and Splitting); MPTCP (Multipath TCP); MPQUIC (Multipath QUIC); Non-3GPP Access; Large Language Models (LLMs); Retrieval-Augmented Generation (RAG); SLICES-RI; OpenAirInterface.

Thèse de Doctorat

Orchestration et gestion intelligentes des réseaux 6G cloud-natifs fondées sur l'intelligence artificielle

Sokratis Christakis

Résumé

La sixième génération de réseaux mobiles (6G) devrait redéfinir les limites de la connectivité sans fil, en supportant des applications allant de la téléprésence holographique (holographic telepresence) et des services omniprésents basés sur l'intelligence artificielle (pervasive AI-driven services) jusqu'à l'intégration transparente des technologies d'accès terrestres, aériennes et satellitaires au sein d'un environnement réseau unifié. La concrétisation de cette vision nécessite non seulement des avancées significatives au niveau du Réseau d'Accès Radio (Radio Access Network — RAN), mais aussi une refonte fondamentale de la façon dont les infrastructures réseau sont conçues, déployées et gérées. Les réseaux cœur (Core Networks) de la 6G sont appelés à être intelligents, économes en énergie et capables de s'adapter de manière autonome à des exigences hétérogènes et en constante évolution — des caractéristiques qui dépassent les capacités des réseaux actuels et qui rendent nécessaire le développement de nouveaux mécanismes d'orchestration (orchestration) et de gestion (management). Relever ces défis suppose non seulement une innovation dans la conception des réseaux cœur, mais aussi l'existence d'infrastructures expérimentales et d'outils permettant leur développement et leur évaluation à grande échelle. Dans la présente thèse, nous établissons d'abord une base empirique pour la gestion des réseaux 5G/6G à travers une évaluation systématique de quatre implémentations open-source de la Fonction de Plan Utilisateur (User Plane Function — UPF) dans des conditions réelles de fonctionnement d'un réseau 5G. Par l'analyse comparative des performances et de la consommation de ressources de calcul dans des conditions identiques de trafic réel, nous mettons en évidence des compromis critiques entre performance et ressources (performance-resource trade-offs) qui ne peuvent être suffisamment appréhendés par une analyse théorique, et qui constituent un prérequis fondamental pour tout mécanisme ultérieur d'orchestration et de gestion des fonctions réseau

(Network Functions). Nous explorons ensuite l'application des principes cloud-native à la mise à l'échelle dynamique (scaling) des fonctions réseau (Network Functions) en fonction de la demande de trafic en temps réel. Nous concevons et évaluons un mécanisme de mise à l'échelle réactif (reactive scaling mechanism) pour des déploiements 5G basés sur Kubernetes, en démontrant que la gestion élastique du réseau cœur (elastic core network management) surpasse substantiellement l'allocation statique de ressources (static resource provisioning), tant en termes de performances réseau que d'efficacité d'utilisation des ressources de calcul. Nous progressons ensuite vers l'orchestration proactive et durable (proactive and sustainable orchestration) qu'exigent les futurs réseaux 6G. En intégrant des mécanismes de prédiction du trafic réseau (traffic prediction) et une optimisation du placement et de la mise à l'échelle tenant compte de l'énergie (energy-aware placement and scaling) pour des implémentations hétérogènes d'UPF, nous montrons qu'une orchestration intelligente peut simultanément satisfaire les exigences de performance et réduire la consommation énergétique. Ce résultat démontre que la durabilité des réseaux cœur 6G (6G Core Networks) peut constituer un objectif de conception réaliste et atteignable. Le quatrième axe de recherche se concentre sur l'un des éléments les plus caractéristiques de la vision de connectivité 6G : l'intégration transparente de multiples technologies d'accès (multi-access integration) au sein d'un environnement réseau unifié. Les réseaux 6G devraient permettre aux utilisateurs d'exploiter simultanément plusieurs technologies d'accès, telles que les réseaux cellulaires (cellular networks), le Wi-Fi et les communications par satellite (satellite communications), le réseau distribuant dynamiquement le trafic entre ces différents chemins. Dans ce cadre, nous explorons expérimentalement des modalités d'implémentation pratique de l'intégration multi-accès (multi-access integration), en évaluant différentes approches d'intégration d'accès (access integration) et des protocoles de transport multichemin (multipath transport protocols) sur des canaux sans fil réels. L'analyse met en évidence des compromis fondamentaux (trade-offs) ayant des implications directes sur la conception de mécanismes de gestion d'accès hétérogène (heterogeneous access management) pour les réseaux au-delà de la 5G et de la 6G. Enfin, nous abordons la question de l'accessibilité des infrastructures expérimentales de recherche qui soutiennent les développements précités. Bien que les plateformes de recherche à grande échelle soient indispensables à l'évaluation des technologies 6G, leur complexité croissante crée des obstacles significatifs pour les nouveaux chercheurs et ralentit le cycle de développement et d'expérimentation. Pour répondre à ce défi, nous présentons slAIces, une

plateforme basée sur des Grands Modèles de Langage (Large Language Models — LLMs), qui combine la Génération Augmentée par Récupération (Retrieval-Augmented Generation — RAG) et le fine-tuning afin d'assister les chercheurs dans l'utilisation et la gestion de l'infrastructure de recherche SLICES-RI. La plateforme réduit considérablement la courbe d'apprentissage pour les nouveaux utilisateurs et permet une automatisation complète des expériences via le langage naturel. Dans l'ensemble, les contributions de la présente thèse forment un cadre de recherche cohérent pour la gestion intelligente (intelligent), durable (sustainable) et accessible (accessible) des réseaux 5G/6G — de l'évaluation empirique et de l'orchestration élastique (elastic orchestration), jusqu'à l'accès hétérogène (heterogeneous access) et l'expérimentation guidée par l'intelligence artificielle (AI-driven experimentation).

Mots-clés : 5G ; 6G ; gestion du cœur de réseau ; fonction du plan utilisateur (User Plane Function, UPF) ; orchestration cloud-native (Cloud-Native Orchestration) ; efficacité énergétique ; apprentissage automatique (Machine Learning) ; ATSSS (Access Traffic Steering, Switching and Splitting) ; MPTCP (Multipath TCP) ; MPQUIC (Multipath QUIC) ; accès non-3GPP (Non-3GPP Access) ; grands modèles de langage (Large Language Models, LLMs) ; RAG (Retrieval-Augmented Generation) ; SLICES-RI ; OpenAirInterface.

List of Publications

Journals

[J1] Sokratis Christakis, Nikos Makris, Serge Fdida, Thanasis Korakis. **Machine Learning–Driven Energy-Aware Scaling of Heterogeneous User Plane Functions in Beyond-5G Core Networks**. Computer Networks: The International Journal of Computer and Telecommunications Networking, Elsevier, 2026. (Submitted)

Conferences

[C1] S. Christakis, T. Tsourdinis, N. Makris, T. Korakis and S. Fdida, **Evaluation of User Plane Function Implementations in Real-World 5G Networks**, IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WK-SHPS), Vancouver, BC, Canada, 2024. [1]

[C2] S. Christakis, N. Makris, T. Korakis and S. Fdida, **On the Automated Scaling of User Plane Function for 5G: An Experimental Evaluation**, 2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Antwerp, Belgium, 2024. [2]

[C3] S. Christakis, N. Makris, S. Fdida and T. Korakis, **Dynamic AI-Based UPF Switching: Adapting for Performance/Energy-Efficiency in 5G Networks**, GLOBECOM 2025 - 2025 IEEE Global Communications Conference, Taipei, Taiwan, 2025. [3]

[C4] D. Kefalas, S. Christakis, S. Fdida, N. Makris, I. Syrigos, V. Passas, and T. Korakis, **slAIces: an LLM Chatbot for Simplifying Experiments with the SLICES-RI**, 2024 IFIP Networking Conference (IFIP Networking), Thessaloniki, Greece, 2024. [4]

[C5] S. Christakis, D. Kefalas, N. Makris, S. Fdida and T. Korakis, **Toward Multi-RAT 6G Networks: Real-Radio Evaluation of Multipath Transport over a 5G ATSSS Testbed**, IEEE Future Networks World Forum (FNWF), 2026, Lisbon, Portugal.

Table of contents

Acknowledgments **xi**

Abstract **xiii**

Περίληψη **xv**

Résumé **xix**

List of Publications **xxiii**

Table of contents **xxv**

List of figures **xxxi**

List of tables **xxxiii**

Abbreviations **xxxv**

1 Introduction **1**

1.1 Introduction to 6G Networks 1

1.1.1 6G Usage Scenarios 2

1.1.2 6G Key Enabling Technologies 3

1.2 Motivation: Challenges in 6G Core Network Management 4

1.2.1 Core Network Performance Under Growing Traffic Demands . . . 5

1.2.2 Flexible, Intelligent and Sustainable Resource Management 5

1.2.3 Heterogeneous and Multi-Access Connectivity 6

1.2.4 Accessibility of Large-Scale Research Infrastructures 6

1.3 Thesis Contributions 7

1.4 Thesis Structure 10

2 Background **13**

2.1 Mobile Network Generations 14

2.1.1 Introduction to Cellular Networks 14

2.1.2 Introduction to 4G/LTE 15

2.1.2.1 E-UTRAN 15

2.1.2.2 Evolved Packet Core (EPC) 16

2.1.3 Introduction to 5G networks 18

2.1.3.1 5G Key Enablers 18

2.1.3.2 NG-RAN (Radio Access Network) 19

2.1.3.3 5G Core (5GC) 20

2.1.3.4 UPFs: Design and key Implementations 23

2.2 Multi-Access Networking and Traffic Steering 25

2.2.1 3GPP and Non-3GPP Access Integration 25

2.2.2 ATSSS: Access Traffic Steering, Switching, and Splitting 26

2.3 Enabling Technologies and Tools for Cloud-Native Networks 27

2.3.1 SDN and NFV Principles 27

2.3.2 Virtual Machines vs Containers 28

2.3.3 Docker and Docker Compose 29

2.3.4 Kubernetes and Service Orchestration 30

2.3.4.1 Architecture. 31

2.3.4.2 Core Abstractions. 31

2.3.4.3 Networking Plugins. 32

2.3.4.4 Scaling. 33

2.3.4.5 Prometheus 33

2.3.4.6 Grafana 34

2.3.5 Cloud-Native Network Functions (CNFs) 34

2.4 Machine Learning for Network Management 34

2.4.1 ML in Beyond-5G and 6G Networks 35

2.4.2 Model Architectures 35

2.4.2.1 Gated Recurrent Unit (GRU) 35

2.4.2.2 Long Short-Term Memory (LSTM) 36

2.4.2.3 CNN-LSTM 36

2.4.2.4 XGBoost . . . 37
2.5 Large Language Models . . . 37
2.5.1 Transformer Architecture . . . 38
2.5.2 Prompting . . . 38
2.5.3 Fine-Tuning . . . 38
2.5.4 Retrieval-Augmented Generation . . . 38
2.6 Experimental Platforms . . . 39
2.6.1 SLICES-RI Infrastructure . . . 39
2.6.2 OpenAirInterface . . . 40
2.6.3 Free5GC . . . 41

3 Evaluation of UPF Implementations in Real-World 5G Networks 43
3.1 Introduction . . . 43
3.2 Related Work . . . 44
3.3 System Architecture . . . 46
3.3.1 SPGWU-Based UPF . . . 47
3.3.2 VPP-Based UPF . . . 48
3.3.3 P4-Switch-Based UPF . . . 49
3.3.4 SmartNIC-P4 UPF . . . 50
3.4 Experimental Evaluation . . . 51
3.4.1 SPGWU-UPF, VPP-UPF, P4-SWITCH-UPF (LIP6/CNRS) . . . 51
3.4.2 SmartNIC P4-Card UPF (NITOS) . . . 54
3.5 Conclusions & Future Work . . . 56

4 Reactive UPF Scaling in cloud-native 5G networks 57
4.1 Introduction . . . 57
4.2 Related Work . . . 58
4.3 System Architecture . . . 60
4.4 Experimental Findings . . . 65
4.5 Conclusion & Future Work . . . 70

5 Proactive and Energy-Aware Orchestration of Heterogeneous UPFs 71
5.1 Introduction . . . 71
5.2 Related Work . . . 73

5.2.1 Energy Efficiency in Softwarized 5G/6G Core Networks 73
5.2.2 Dynamic Placement and Scaling of Core Functions 74
5.2.3 Machine Learning for Proactive Network Orchestration 75
5.2.4 UPF Implementation Heterogeneity: Performance & Energy Trade-offs 75
5.2.5 Positioning of the Proposed Framework 76
5.3 System Architecture 77
5.3.1 UPF Technologies 77
5.3.2 Dataset & Preprocessing 77
5.3.3 Machine Learning 78
5.3.3.1 Plain LSTM 80
5.3.3.2 GRU 80
5.3.3.3 XGBoost 80
5.3.3.4 CNN-LSTM 80
5.3.4 End-to-end Architecture & Functionality 81
5.3.4.1 Mathematical Model 83
5.4 Experimental Setup & Evaluation 86
5.4.1 Experimental Setup 86
5.4.2 Experimental Scenarios 87
5.4.3 Machine Learning Evaluation 88
5.4.4 Experimental Results 90
5.4.4.1 Single SPGWU 90
5.4.4.2 Single VPP 91
5.4.4.3 Multiple SPGWUs 91
5.4.4.4 Multiple VPPs 93
5.4.4.5 Our Framework 95
5.4.5 Discussion 97
5.5 Conclusion 101

6 Multi-RAT Integration and Evaluation in a Real Beyond-5G Network 103
6.1 Introduction 103
6.2 Related Work 105
6.3 System Architecture 106
6.3.1 Core-level Integration: N3IWF and ATSSS 106

6.3.2 RAN-level Integration: Wi-Fi-DU . . . 107
6.4 Experimental Setup & Results . . . 108
6.4.1 Single-Path Baselines: Wi-Fi via N3IWF and 5G NR via OAI gNB 109
6.4.2 MPTCP . . . 111
6.4.3 MPQUIC . . . 112
6.4.4 Wi-Fi-DU . . . 114
6.5 Conclusions . . . 116

7 LLM-Based Automation of Network Experimentation 117
7.1 Introduction . . . 117
7.2 Related Work . . . 119
7.3 Architecture & LLM Configuration . . . 120
7.3.1 Data acquisition . . . 121
7.3.2 Data augmentation . . . 121
7.3.3 In-context learning & Fine Tuning . . . 122
7.3.3.1 In-context learning via a RAG system . . . 122
7.3.3.2 Fine Tuning (FT) . . . 123
7.4 Experimental Evaluation . . . 125
7.4.1 Datasets . . . 125
7.4.1.1 Data size & Vocabulary uniqueness . . . 125
7.4.1.2 Data comprehensiveness . . . 125
7.4.1.3 Retriever Analysis . . . 126
7.4.2 Performance Evaluation - In-context Learning . . . 128
7.4.3 Performance Evaluation - Fine Tuned Model . . . 129
7.5 Toward Automated Experiment Execution . . . 131
7.5.1 Implementation . . . 131
7.5.2 Controller . . . 134
7.5.3 Performing an Experiment . . . 134
7.6 Conclusion & Future Work . . . 135

8 Conclusions and Future Work 137
8.1 Summary of Contributions . . . 137
8.2 Perspectives for Future Work . . . 140

8.3 Lessons Learned and Outlook 142

References **143**

List of figures

2.1 Cellular Networks: Cell Density in Urban vs Rural Areas 15
2.2 4G/LTE Architecture 16
2.3 5G RAN protocol stack and functional split across CU, DU and RU 20
2.4 5G Core Service-based Architecture (SBA) 23
2.5 Container vs VM Architecture 29

3.1 Experimental Setup: Softwarized UPFs vs P4-Switch-based UPF 46
3.2 P4-Switch-based UPF Experimental Architecture. 50
3.3 SmartNIC-P4 UPF Experimental Architecture. 51
3.4 Throughput Evaluation: Uplink and Downlink. 52
3.5 Latency Evaluation: RTT & Jitter. 53
3.6 Packet Loss Evaluation: Uplink and Downlink. 53
3.7 CPU Consumption for each UPF. 54
3.8 Throughput, RTT & Jitter for SmartNIC-P4 UPF 55
3.9 Packet Loss & CPU Consumption 55

4.1 Monolithic vs Dynamic Algorithm Architecture 61
4.2 Data representation: Active Connections per Hour 62
4.3 Data representation: Throughput per Hour 63
4.4 UPF CPU Allocation Comparison for UDP and TCP 67
4.5 UPF Packet Reception for the UDP scenario 68
4.6 UPF Packet Reception for the TCP scenario 69

5.1 Overall System Architecture 81
5.2 Experimental traffic dataset: Total throughput over Time. 87
5.3 Comparison & Evaluation of different prediction models. 91

5.4 Single SPGWU-UPF performance: Throughput and CPU usage over time. . 92
5.5 Single VPP-UPF: Throughput and CPU utilization over time. 93
5.6 Multiple SPGWU deployment: per-instance throughput and aggregate demand. 94
5.7 Multiple SPGWU deployment: per-instance CPU utilization and energy consumption. 95
5.8 Multiple VPP deployment: per-instance throughput and aggregate demand. 96
5.9 Multiple VPP deployment: per-instance CPU utilization and energy consumption. 97
5.10 Proposed framework: per-node throughput and total aggregate throughput over time. 98
5.11 Proposed framework: per-node CPU utilization and energy consumption. . 99

6.1 ATSSS System Architecture utilizing different multi-path transport protocols 107
6.2 RAN-Level integration of Wi-Fi into 5G 108
6.3 N3IWUE DL Throughput over Time (UDP vs TCP) 110
6.4 5G-NR UE DL Throughput over Time (UDP vs TCP) 110
6.5 MPTCP DL Aggregated Throughput (Wi-Fi & 5G) 111
6.6 MPTCP DL Throughput Before and After gNB Crash 112
6.7 PicoQUIC Single-Path DL Throughput per RAT 113
6.8 MPQUIC DL Aggregated Throughput (Wi-Fi & 5G) 113
6.9 MPQUIC DL Throughput Before and After gNB Crash 114
6.10 Wi-Fi-DU Single-Path Downlink Throughput (UDP) 115
6.11 Wi-Fi-DU Aggregated Downlink Throughput (5G NR + Wi-Fi, UDP) . . . 115

7.1 System Architecture 124
7.2 Data Comparison: Vocabulary & Entities Uniqueness 126
7.3 Retriever Similarity Percentage 127
7.4 Percentage of Correctly Answered Questions 128
7.5 Raw GPT-3.5 Model vs Fine-tuned GPT-3.5 Model: Percentage of Correctly Answered Questions 130
7.6 Framework Overview. 132
7.7 SLICES Documentation LLM Pipeline. 133
7.8 Post-5G LLM Pipeline. 134

List of tables

3.1 LIP6 Experimental Setup 47
3.2 NITOS Experimental Setup 48

4.1 Master Specifications 60
4.2 Worker1 & Worker2 Specifications 62

5.1 Hyperparameters of evaluated prediction models 79
5.2 Testbed Hardware and Software Specification 82
5.3 Regression Coefficients of the CPU Utilization Model ($\text{CPU}_u = a_u d^2 + b_u d + c_u$) 84
5.4 Prediction Accuracy of Evaluated Models (3-Step-Ahead Forecasting) . . . 89

6.1 SLICES-RI Experimental Testbed Hardware Configuration 109

7.1 Question Responses per Model & Dataset 129

Abbreviations

5G	Fifth Generation
5GC	5G Core Network
5QI	5G QoS Identifier
6G	Sixth Generation
ACK	Acknowledgement
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
AP	Access Point
API	Application Programming Interface
AR	Augmented Reality
ASIC	Application-Specific Integrated Circuit
ATSSS	Access Traffic Steering, Switching, and Splitting
AUSF	Authentication Server Function
B5G	Beyond 5G
BS	Base Station
CN	Core Network
CNCF	Cloud Native Computing Foundation
CNF	Cloud-Native Network Function
CNI	Container Network Interface
CNN	Convolutional Neural Network
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRD	Custom Resource Definition
CU	Central Unit
CUPS	Control and User Plane Separation

DL	Downlink
DN	Data Network
DNS	Domain Name System
DPDK	Data Plane Development Kit
DU	Distributed Unit
EAP	Extensible Authentication Protocol
eBPF	Extended Berkeley Packet Filter
eMBB	Enhanced Mobile Broadband
EPC	Evolved Packet Core
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
FAR	Forwarding Action Rule
FT	Fine-Tuning
gNB	Next Generation NodeB
GPT	Generative Pre-trained Transformer
GRE	Generic Routing Encapsulation
gRPC	Google Remote Procedure Call
GRU	Gated Recurrent Unit
GSM	Global System for Mobile Communications
GTP-U	GPRS Tunnelling Protocol User Plane
HAPS	High Altitude Platform Station
HARQ	Hybrid Automatic Repeat Request
HPA	Horizontal Pod Autoscaler
HPC	High-Performance Computing
HSS	Home Subscriber Server
HTTP	HyperText Transfer Protocol
ICL	In-Context Learning
IETF	Internet Engineering Task Force
IMT	International Mobile Telecommunications
IoT	Internet of Things
IP	Internet Protocol
ISAC	Integrated Sensing and Communication
ITU	International Telecommunication Union

ITU-R	ITU Radiocommunication Sector
JSON	JavaScript Object Notation
LEO	Low Earth Orbit
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSTM	Long Short-Term Memory
LTE	Long Term Evolution
LWAAP	LTE-WLAN Aggregation Adaptation Protocol
MAC	Medium Access Control
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MEC	Mobile Edge Computing
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MME	Mobility Management Entity
mmWave	Millimeter Wave
mMTC	Massive Machine-Type Communications
MPQUIC	Multi-Path QUIC
MPTCP	Multi-Path Transmission Control Protocol
N3IWF	Non-3GPP Interworking Function
N3IWUE	Non-3GPP Interworking User Equipment
NAS	Non-Access Stratum
NEF	Network Exposure Function
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NG-RAN	Next Generation Radio Access Network
NIC	Network Interface Card
NITOS	Network Implementation Testbed using Open Source Platforms
NLP	Natural Language Processing
NR	New Radio
NRF	Network Repository Function
NSSF	Network Slice Selection Function

NTN	Non-Terrestrial Network
NWDAF	Network Data Analytics Function
OAI	OpenAirInterface
OFDM	Orthogonal Frequency-Division Multiplexing
O-RAN	Open Radio Access Network
P4	Programming Protocol-independent Packet Processors
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PDR	Packet Detection Rule
PDU	Protocol Data Unit
PFCP	Packet Forwarding Control Protocol
PGW	Packet Data Network Gateway
PHY	Physical Layer
QoS	Quality of Service
QUIC	Quick UDP Internet Connections
RACK	Recent ACKnowledgement
RAG	Retrieval-Augmented Generation
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technology
RBAC	Role-Based Access Control
REST	Representational State Transfer
RI	Research Infrastructure
RIC	RAN Intelligent Controller
RIS	Reconfigurable Intelligent Surface
RLC	Radio Link Control
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RNTI	Radio Network Temporary Identifier
RRC	Radio Resource Control
RRM	Radio Resource Management

RTT	Round-Trip Time
RU	Radio Unit
SBA	Service-Based Architecture
SDAP	Service Data Adaptation Protocol
SDN	Software-Defined Networking
SGW	Serving Gateway
SLA	Service Level Agreement
SLICES	Scientific Large-scale Infrastructure for Computing/Communication Experimental Studies
SMF	Session Management Function
SPGWU	Serving and PDN Gateway User Plane
SR-IOV	Single Root I/O Virtualization
TCP	Transmission Control Protocol
THz	Terahertz
TNGF	Trusted Non-3GPP Gateway Function
TNAP	Trusted Non-3GPP Access Point
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
UERANSIM	UE and RAN Simulator
UL	Uplink
UL CL	Uplink Classifier
UMTS	Universal Mobile Telecommunications System
UPF	User Plane Function
URR	Usage Reporting Rule
URLLC	Ultra-Reliable Low-Latency Communications
USRP	Universal Software Radio Peripheral
VM	Virtual Machine
VNF	Virtualized Network Function
VPA	Vertical Pod Autoscaler

VPP	Vector Packet Processing
VR	Virtual Reality
VXLAN	Virtual Extensible Local Area Network
Wi-Fi DU	Wi-Fi Distributed Unit
WLAN	Wireless Local Area Network
XDP	eXpress Data Path
XGBoost	Extreme Gradient Boosting
XR	Extended Reality
YAML	YAML Ain’t Markup Language

Chapter 1

Introduction

1.1 Introduction to 6G Networks

The ongoing digital transformation of modern society is driving an unprecedented demand for advanced wireless connectivity. Emerging applications, including holographic telepresence, autonomous vehicles, industrial automation, extended reality (XR), large-scale digital twins, and pervasive Internet of Things (IoT) deployments, are significantly expanding the capabilities required from communication networks. These applications impose very strict performance requirements, such as terabit-per-second data rates, end-to-end latency below one millisecond, sub-centimeter positioning accuracy, and the ability to support millions of devices per square kilometer. Although fifth-generation (5G) networks have introduced substantial technological advancements, they were not designed to simultaneously meet this full spectrum of demands. As the number of connected devices and the volume of data-intensive services continue to grow exponentially, the limitations of 5G systems are becoming increasingly evident.

On the other hand, sixth-generation (6G) networks are currently under development to meet these growing demands. Aiming for commercial launch around 2030 under the ITU-R IMT-2030 framework [5], 6G is not simply an upgrade to existing systems but a complete rethinking of how mobile networks are built. The goal is to deliver a single, unified platform where communication, computing, sensing, and intelligence all work together as built-in features rather than separate add-ons. In parallel, global standardization efforts are progressing rapidly, with key organizations such as ITU-R, 3GPP, and regional initiatives across Europe, Asia, and North America actively contributing to the definition of the 6G vision and its as-

sociated technical requirements [6].

1.1.1 6G Usage Scenarios

The ITU-R IMT-2030 framework identifies six major usage scenarios that collectively define the scope of 6G [5, 7], reflecting a vision far broader than the three use case categories of 5G.

- **Immersive Communication.** As already stated, immersive communication builds on 5G's AR and VR to deliver richer, multisensory experiences through holographic telepresence and advanced XR. Supporting these experiences requires sustained data rates approaching 1 Tbps and end-to-end latency below 1 ms, which are far beyond the capabilities of current networks.

- **Hyper-Reliable and Low-Latency Communication.** Building on the URLLC concept of 5G, this use case targets latencies below 100 µs and packet error rates as low as 10^{-9} for mission-critical applications such as remote surgery, autonomous industrial control, and the tactile internet, where microsecond delays or a single lost packet can have life-critical consequences.

- **Massive Communication.** The continued proliferation of IoT devices, smart manufacturing sensors, and connected infrastructure requires 6G to support device densities of up to 10^7 per km^2. This demands entirely new access paradigms capable of managing massive simultaneous connections with minimal overhead and energy consumption.

- **Ubiquitous Connectivity.** 6G aims to eliminate coverage gaps by seamlessly integrating terrestrial and non-terrestrial access — including Low Earth Orbit (LEO) satellites, High Altitude Platform Stations (HAPS), and UAV relays — into a unified network. This extends reliable connectivity to remote and underserved areas currently beyond the reach of terrestrial infrastructure.

- **Integrated Sensing and Communication.** 6G networks will natively support environmental sensing as a service, using wireless signals simultaneously for data transmission and physical world perception. Applications include real-time localization, gesture recognition, object detection, environmental mapping, and autonomous nav-

igationm directly enabling a new class of context-aware services that are impossible with communication-only networks.

- **AI and Communication.** Artificial intelligence is embedded as a native capability of 6G, enabling networks that are self-configuring, self-optimizing, and self-healing.

1.1.2 6G Key Enabling Technologies

Realizing the 6G use cases requires a set of enabling technologies that go fundamentally beyond the capabilities of 5G.

- **Terahertz (THz) Spectrum.** 6G exploits the terahertz frequency bands, spanning roughly 0.1 to 10 THz, to access contiguous bandwidths of tens of GHz, enabling peak data rates of up to 1 Tbps required for immersive applications [8]. However, THz frequencies introduce severe propagation challenges including high path loss and extreme blockage sensitivity, requiring novel antenna designs, advanced beamforming, and tight integration with RIS to maintain viable links. Nonetheless, early experimental demonstrations have begun validating THz feasibility, with recent trials achieving data rates exceeding 100 Gbps over THz links [9].

- **Reconfigurable Intelligent Surfaces (RIS).** RIS are programmable meta-surfaces composed of large arrays of passive elements that dynamically control the amplitude, phase, and polarization of incident electromagnetic waves [10]. By intelligently shaping the wireless propagation environment, RIS creates virtual line-of-sight paths, mitigates blockage, and enhances both coverage and sensing performance, without active power amplification, making it a fundamentally energy-efficient infrastructure component.

- **AI-Native Architecture.** Unlike 5G where AI is an external optimization add-on, 6G embeds machine learning natively into all layers of the network, from physical-layer signal processing and beamforming to core network orchestration and resource management [11]. This enables zero-touch autonomous operation, where AI agents dynamically reconfigure network parameters, predict demand, and recover from faults without human intervention.

- **Integrated Sensing and Communication (ISAC) Technology.** ISAC merges communication and sensing hardware, spectrum, and signal processing into a unified frame-

work [12]. ISAC-enabled base stations simultaneously transmit data and emit sensing waveforms, processing reflected signals to extract positional and environmental information. Combined with massive MIMO and RIS, ISAC enables fine-grained 3D environmental mapping and object tracking at scales not achievable with dedicated sensing infrastructure.

- **Non-Terrestrial Networks (NTN).** 6G natively integrates terrestrial and non-terrestrial infrastructure into a unified architecture [13]. NTN components — LEO satellites, HAPS, and UAV relays — extend coverage to areas beyond terrestrial reach, provide resilient backup connectivity, and enable aerial edge computing. Tight integration with the 6G core, including seamless handover between terrestrial and satellite access, is a key architectural challenge currently being addressed in standardization.

- **Energy Efficiency.** Energy efficiency is a first-class design objective in 6G, targeting a substantial reduction in energy consumption compared to 5G. The densification required by THz deployments and the proliferation of connected devices risk making 6G significantly more energy-intensive, making sustainability a central design constraint. This is addressed through AI-driven energy management, coordinated base station sleep modes, RIS-assisted passive coverage, and energy harvesting at edge nodes.

1.2 Motivation: Challenges in 6G Core Network Management

The ambitious vision of 6G sets requirements that go far beyond what current network infrastructure can deliver. Meeting these requirements demands not only advances in RAN, but a fundamental transformation of how core networks are designed, deployed, and operated. The ITU-R IMT-2030 framework explicitly elevates network intelligence, heterogeneity, energy efficiency, and ubiquitous connectivity to first-class design objectives, reflecting a broad consensus that future network management must be radically more adaptive, efficient, and autonomous. The following sections identify the principal challenges that motivate the research presented in this thesis.

1.2.1 Core Network Performance Under Growing Traffic Demands

Global mobile data traffic is projected to grow at a compound annual rate exceeding 20% through 2030 [14], driven by extended reality, autonomous systems, industrial automation, and massive IoT. This growth significantly increases the operational demands on core network user plane functions, as their throughput, latency, and resource efficiency strongly influence end-user service quality.

A wide spectrum of user plane implementations has emerged in response, ranging from software-based solutions on commodity hardware to hardware-accelerated approaches leveraging programmable ASICs and SmartNICs. Each architecture offers a different combination of performance, flexibility, energy consumption, and deployment complexity. However, in the absence of systematic real-world comparative evaluations, network operators and researchers lack the empirical grounding needed to make informed deployment decisions. The gap between theoretical performance claims and actual behavior under realistic traffic conditions remains largely uncharacterized, making it difficult to identify which solutions genuinely meet the performance and sustainability demands of 6G networks.

1.2.2 Flexible, Intelligent and Sustainable Resource Management

Mobile network traffic fluctuates significantly across time, yet network functions are typically provisioned statically for worst-case loads. This mismatch is both resource wasteful and environmentally unsustainable as over-provisioned resources consume energy regardless of actual demand, while under-provisioned deployments degrade service quality during peaks.

Cloud-native architectures offer dynamic scaling as a solution, but translating this capability into effective network function management is non-trivial. Network functions exhibit complex, non-linear relationships between traffic load, resource utilization, and performance that generic autoscaling mechanisms fail to capture accurately. Purely reactive approaches are also inherently limited because by the time congestion is detected, service quality has already deteriorated. In addition, the IMT-2030 framework targets a huge improvement in energy efficiency per bit compared to 5G, and leading standardization bodies have identified energy-aware management as a critical 6G requirement. Designing orchestration mechanisms that can anticipate future demand, adapt resource configurations proactively, and jointly optimize for both performance and energy efficiency remains one of the central open challenges on the path toward the intelligent and sustainable core networks that 6G envisions.

1.2.3 Heterogeneous and Multi-Access Connectivity

As already discussed, the 6G vision of ubiquitous connectivity extends mobile networks beyond conventional cellular infrastructure, natively integrating complementary access technologies such as Wi-Fi and satellite communications into a unified core. However, integrating heterogeneous access technologies introduces significant complexity at both architectural and operational levels. Different access technologies operate under fundamentally different latency, throughput, and reliability profiles, and the core network must manage traffic from all of them simultaneously and transparently. The 3GPP ATSSS mechanism addresses this by enabling traffic to be steered, switched, and split across simultaneous access paths using multipath transport protocols. However, the practical performance implications of multi-access convergence, including protocol overhead, path switching impact on session continuity, and interactions between different access technologies under real traffic, remain an open research challenge as heterogeneous access becomes a core aspect of 6G.

1.2.4 Accessibility of Large-Scale Research Infrastructures

Advancing the 6G research agenda requires experimental validation on real, large-scale infrastructure. Large-scale federated research platforms that integrate heterogeneous testbeds, programmable radio resources, and cloud computing across multiple institutions have emerged as essential instruments for the 6G research community. However, the richness of such platforms is inseparable from their complexity and as a result, conducting experiments requires deep familiarity with platform-specific toolchains, resource management workflows, and configuration formats, creating a steep learning curve that limits accessibility and slows the translation of research ideas into experimental results.

The emergence of large language models as general-purpose reasoning and knowledge retrieval tools offers a compelling opportunity to lower this barrier. LLMs can assist users in navigating complex technical domains, answering domain-specific questions, and translating high-level intent into executable actions. However, applying these capabilities reliably to the specialized context of large-scale network experimentation requires adapting general-purpose models to domain-specific knowledge, which remains an open and practically important research challenge.

Together, these challenges span a full lifecycle of 6G core network research: from the empiri-

cal evaluation of core network functions and their intelligent orchestration, to the integration of heterogeneous access technologies and the accessibility of the experimental infrastructure needed to validate such advances. Addressing them collectively motivates the research presented in this thesis.

1.3 Thesis Contributions

This thesis addresses the challenges outlined in Section 1.2 through five research directions, guided by two overarching research questions:

1. *How can 6G core networks be empirically evaluated, intelligently orchestrated and extended to support heterogeneous access in a sustainable and adaptive manner?*

2. *How can large-scale experimental research infrastructures be made accessible and automated to accelerate 6G research and validation?*

To answer these questions, the thesis addresses the problem across five research directions:

1. Real-World Performance Evaluation of UPF Implementations

Research Questions:

- How do different existing UPF implementations compare in terms of performance and resource efficiency when evaluated in a real 5G network?
- What are the performance, resource, and deployment trade-offs between software-based and hardware-accelerated UPF solutions under realistic 5G traffic conditions?
- How can real-world UPF evaluations guide deployment decisions in operational future 6G networks?

Contribution (Chapter 3): In Chapter 3, we design and conduct a systematic comparative evaluation of four open-source UPF implementations deployed within a real end-to-end 5G network built on OpenAirInterface on the NITOS and LIP6 testbeds of SLICES-RI. All implementations are evaluated simultaneously across throughput, latency, packet loss, and CPU utilization under identical traffic conditions, revealing the fundamental performance and resource trade-offs between kernel-based, DPDK-accelerated, and hardware-offloaded packet

processing architectures. The resulting empirical profiles form the quantitative foundation on which all subsequent dynamic management and orchestration mechanisms are built.

2. Reactive Horizontal Scaling of UPF Instances in Kubernetes

Research Questions:

- How can UPF instances be dynamically scaled in a cloud-native 5G network to adapt capacity to varying traffic demands?
- How can real-time network performance metrics drive more effective UPF scaling decisions?
- What are the benefits of dynamic UPF scaling over static deployment in terms of resource efficiency and congestion prevention?

Contribution (Chapter 4): In Chapter 4, we designed, implemented, and evaluated a reactive UPF scaling mechanism for cloud-native 5G deployments orchestrated by Kubernetes. Driven by real-time data-plane performance metrics, the mechanism deploys additional UPF instances when demand approaches capacity and releases them when load subsides. Evaluated on the NITOS testbed with a real-world telecommunications dataset, the dynamic mechanism significantly outperforms a monolithic static deployment in CPU efficiency and packet reception, preventing congestion events that a fixed single-UPF configuration cannot avoid.

3. Proactive and Energy-Aware Orchestration of Heterogeneous UPFs

Research Questions:

- How can core network orchestration move beyond reactive scaling toward proactive, anticipatory management that prevents congestion before it occurs?
- How can ML-based traffic forecasting be leveraged to anticipate future demand and guide UPF deployment decisions ahead of congestion?
- How can the complementary energy-performance trade-offs between heterogeneous UPF implementations be exploited to jointly satisfy throughput requirements and minimize energy consumption?

Contribution (Chapter 5): In Chapter 5, we proposed a proactive and energy-aware UPF selection and scaling framework that jointly optimizes the number and type of UPF instances at each decision interval. A CNN-LSTM model, selected through systematic benchmarking

against LSTM, GRU, and XGBoost baselines, achieves a MAPE of 5.74% for 3-step-ahead throughput forecasting on a real 5G production dataset. Based on polynomial CPU and linear power models derived from empirical UPF profiling, an optimization mechanism selects at each interval the deployment configuration that minimizes energy consumption while satisfying predicted demand. Evaluated across five deployment scenarios on a real Kubernetes-based SLICES-RI testbed, the framework achieves an average packet loss of only 1.5% while reducing total energy consumption by 51% compared to the static all-VPP deployment — the only baseline otherwise capable of meeting equivalent throughput requirements.

4. Multi-RAT Integration and Evaluation in a Real Beyond-5G Network

Research Questions:

- The 6G vision of ubiquitous connectivity requires seamless integration of heterogeneous access technologies into a unified core network, yet real-world experimental evaluations of such integration remain scarce. How do different non-3GPP integration approaches compare in a real beyond-5G environment, and what are their practical implications for core network operation?
- How do MPTCP and MPQUIC-based traffic splitting under ATSSS compare in terms of aggregated throughput, resilience, and failover behavior over real heterogeneous radio channels?
- What are the fundamental trade-offs between standards-compliant core-level integration and RAN-level multi-access approaches in terms of performance, resilience, and architectural complexity?

Contribution (Chapter 6): In Chapter 6, we present a real-world evaluation of multipath transport over a 5G ATSSS testbed, comparing two core-level integration approaches via the N3IWF — one using MPTCP and one using MPQUIC as the multipath layer — against a RAN-level Wi-Fi-DU integration that treats Wi-Fi as a non-3GPP DU managed directly by the CU. All experiments are conducted over real over-the-air 5G NR and Wi-Fi channels on the NITOS testbed. MPTCP achieves near-optimal aggregation with robust failover recovery, while MPQUIC delivers higher aggregated throughput and instantaneous path migration upon failure. The RAN-level Wi-Fi-DU achieves the highest throughput but at the cost of 3GPP non-compliance and loss of path redundancy. These results reveal a fundamental trade-off between standards compliance, aggregation performance, and resilience that has

direct implications for the design of heterogeneous access management in beyond-5G and 6G deployments.

5. LLM-Based Automation of Network Experimentation

Research Questions:

- Large-scale experimental research infrastructures are growing in complexity and diversity, creating steep learning curves that hinder researcher onboarding. How can AI-driven tools effectively reduce this barrier and simplify access to such platforms?
- How can domain-specific knowledge be effectively integrated into LLMs to provide accurate and reliable guidance for network experimentation in large-scale experimentation platforms?
- How can natural language interfaces be extended to fully automate the experiment lifecycle, from intent description to deployed experiment?

Contribution (Chapter 7): In Chapter 7, we present slAIces, an LLM-based system designed to lower the barrier for new users accessing the SLICES-RI infrastructure by providing domain-aware assistance over its documentation. Through systematic evaluation, we show that enriching the knowledge base improves response accuracy and reliability, while adaptation techniques further enhance performance in domain-specific queries. We then extend slAIces into a fully automated framework for 5G experimentation, enabling the translation of natural language intent into structured experiment configurations and supporting automated deployment on the SLICES-RI infrastructure.

The cumulative result of this thesis is a comprehensive and empirically grounded research program spanning UPF evaluation, reactive scaling, proactive energy-aware orchestration, non-3GPP multi-access integration, and AI-assisted experimentation. Each contribution has been validated on real SLICES-RI testbed infrastructure under realistic conditions, and each builds upon and motivates the next, forming a coherent investigation into intelligent, sustainable, and accessible 5G/6G core network management.

1.4 Thesis Structure

The rest of this thesis is organized as follows:

- **Chapter 1 – Introduction:** Presents the motivation behind this research, tracing the evolution of mobile networks toward 6G and identifying the core challenges that define the thesis research questions. Summarizes the contributions of each chapter.

- **Chapter 2 – Background:** Provides the technical foundations necessary to understand the contributions of this thesis. It traces the evolution from early cellular networks through 4G/LTE to 5G, introduces the UPF and its open-source implementations, and covers the enabling technologies for cloud-native deployments. It further presents multi-access networking and ATSSS, machine learning techniques for network management, large language models for automated experimentation, and the experimental platforms used throughout this thesis.

- **Chapter 3 – Evaluation of UPF Implementations in Real-World 5G Networks:** Establishes the empirical foundation of the thesis through a comprehensive benchmarking study of four UPF key implementations deployed in a real-world end-to-end 5G network. The evaluation covers throughput, latency, packet loss, and CPU utilization under identical traffic conditions, revealing the fundamental performance and resource trade-offs between implementations that motivate the subsequent chapters.

- **Chapter 4 – Reactive UPF Scaling in Cloud-Native 5G Networks:** Building on the observation that software-based UPFs can be dynamically managed unlike their hardware counterparts, this chapter proposes a reactive scaling mechanism for UPF instances within a Kubernetes cluster. The system monitors real-time network performance metrics and adjusts the number of active UPF instances to prevent congestion while minimizing resource waste, and is evaluated on real testbed infrastructure under dynamic traffic conditions.

- **Chapter 5 – Proactive and Energy-Aware Orchestration of Heterogeneous UPFs:** Extends the scaling framework of Chapter 4 in two directions: from reactive to proactive, using a CNN-LSTM model to forecast traffic demand three steps ahead; and from homogeneous to heterogeneous, jointly selecting the number and type of UPF instances to deploy at each decision interval. The proposed framework exploits the complementary energy profiles of SPGWU and VPP to minimize energy consumption while satisfying throughput requirements.

- **Chapter 6 – Multi-RAT Integration and Evaluation in a Real Beyond-5G Network:** Investigates the integration of non-3GPP access technologies into the 5G Core and evaluates multipath transport protocols under real over-the-air conditions. It compares core-level integration via the N3IWF using both MPTCP and MPQUIC against a RAN-level Wi-Fi-DU approach, revealing a fundamental trade-off between standards compliance, aggregation performance, and path resilience that has direct implications for heterogeneous access management in beyond-5G and 6G networks.

- **Chapter 7 – LLM-Based Automation of Network Experimentation:** Addresses the usability of the experimental infrastructure that underpins the entire thesis. It presents slAIces, an LLM-based chatbot that leverages RAG and fine-tuning to lower the barrier for new users accessing SLICES-RI, and extends it into a fully automated experiment execution platform integrating a multi-pipeline architecture and a gRPC-based deployment Controller.

- **Chapter 8 – Conclusions and Future Work:** Provides the contributions of the thesis by revisiting each research question and answering it conclusively. Presents the main findings as a unified body of evidence, discusses limitations and open challenges, and outlines possible directions for future research in the context of 6G core network management.

Chapter 2

Background

This chapter presents the essential information regarding the technical foundations deemed necessary to support the contributions of this thesis. Section 2.1 begins by tracing the evolution of wireless networks from their inception to 5th-generation (5G) networks, delving into the technologies that defined each generation as well as their architecture. It is also extended by an introduction to the role of the UPF and the most prominent implementations, analyzing how they operate and their limitations. Next, in Section 2.2, we analyze the effort to integrate non-3GPP technologies (e.g., Wi-Fi, Satellite Communications) into wireless networks, as well as the key network functions that play a decisive role in this process. In Section 2.3 we introduce the key technologies and tools that have facilitated the transition of wireless networks from hardware-defined to software-defined architectures, as well as their deployment in cloud-native environments. In addition, in Section 2.4, we analyze the advent of Machine Learning models and basic functionalities as well as their expected role in future cellular networks. Furthermore, in Section 2.5, we provide a brief introduction to LLMs and how they have helped end-users leverage their capabilities to reduce the learning curve when conducting 5G experimentation. Finally, in Section 2.6 we list the experimentation platforms and testbeds used throughout this thesis, as our primary concern was that the research we are promoting could be replicated and validated in real-world environments.

2.1 Mobile Network Generations

2.1.1 Introduction to Cellular Networks

Prior to the advent of cellular communications, information exchange relied entirely on wired infrastructure and mostly on coaxial cables, telephone lines, and fixed terminals physically attached to a network point. While sufficient for that period in time, a fundamental problem arose that now defines our daily lives: user mobility was impossible. The advent of mobile communications in the late 1970s, and more specifically in Japan in 1979, led to the launch of the first commercial wireless network, enabling connection to a network without the use of cables but through wireless interfaces. This event laid the foundation for the first generation of wireless networks (1G), and subsequently became the basis upon which subsequent wireless networks from 1G through the anticipated 6G were built, which, in turn, redefined the limits of speed, capacity, and mass user connectivity.

As the name cellular implies, the basic principle behind a cellular network is the distribution of a geographic space into cells (Figure 2.1), where each cell is served by one base station that is able to manage radio communication with every wireless device that is within its area of coverage. Cells are not uniform in all cases, meaning that their size and density can vary based on the population, terrain and of course traffic demand. For example, we can see that in densely populated urban areas, where many users rely on the services of a specific cellular network and there may be obstacles hindering signal propagation, operators are required to use smaller cells, such as picocells and microcells, in order to deal with the signal obstruction and elevated load and congestion. In contrast, in rural areas where traffic is lower and there may be fewer physical obstacles, operators have the option of using larger cells, such as macrocells, which can cover a wider geographic area taking advantage of the fact that users are not concentrated but scattered. This flexible, demand-driven topology is what allows cellular networks to scale from a single city to global coverage using a unified architectural model.

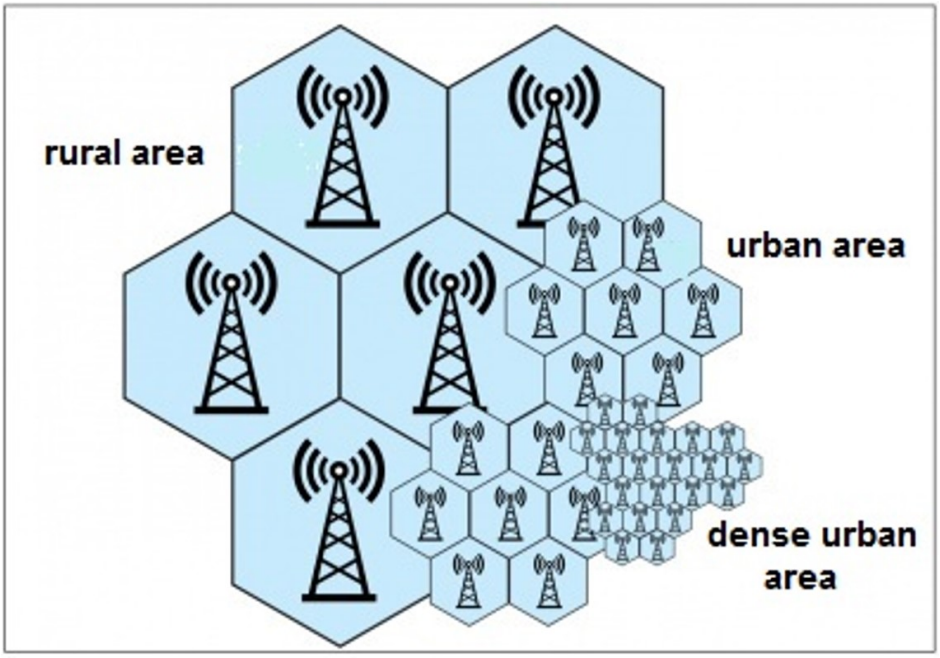


Figure 2.1: Cellular Networks: Cell Density in Urban vs Rural Areas

2.1.2 Introduction to 4G/LTE

The advent of 4G networks, which was largely based on Long-Term Evolution (LTE), is widely regarded as a fundamental technological development, that laid the foundation for next-generation mobile networks, such as 5G and 6G. LTE is often mentioned in the same context as 4G, both in academia and in industry, as they share the same architecture. However, LTE initially did not meet 4G standards in terms of speeds and user experience, which it subsequently managed to achieve with LTE-Advanced.

The 4G/LTE system, illustrated in Figure 2.2 is composed of two main components: the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and the Evolved Packet Core (EPC).

2.1.2.1 E-UTRAN

E-UTRAN is the radio access network of 4G/LTE networks and consists of the base station, which in the context of 4G is called evolved Node B (eNodeB), and the wireless user device, which is called User Equipment (UE). The eNodeB is configured to handle wireless communication between itself and UEs and is exclusively managed by the network operator. In 4G networks, eNodeBs were designed so that all control and radio protocol stack func-

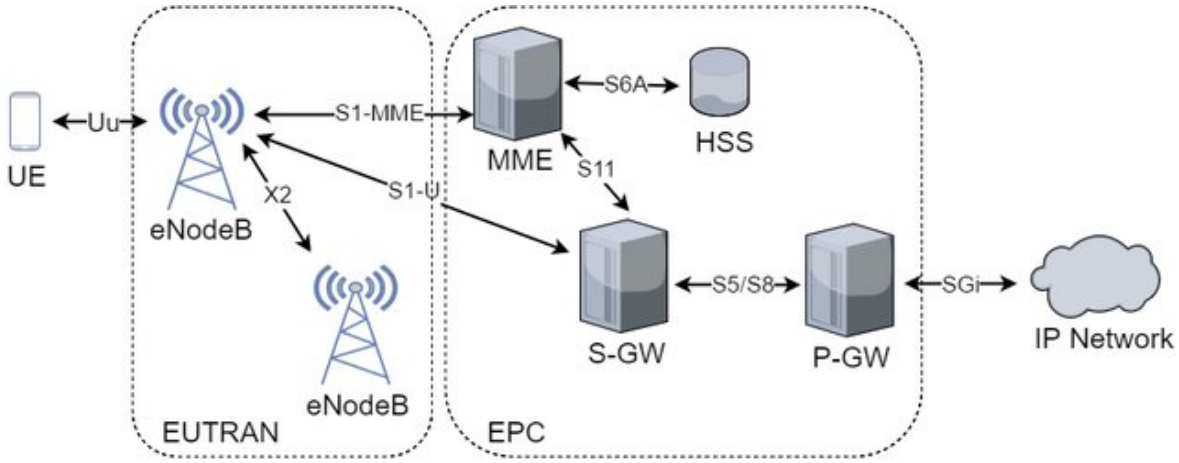


Figure 2.2: 4G/LTE Architecture

tions are integrated within the eNodeB itself, without being split across different network components, something that, as we will see later, changed in 5G.

As expected, the eNodeB is enabled to perform several critical functions to ensure efficient and reliable communication between the network and UEs:

- Radio Resource Management (RRM), which is responsible for the allocation and optimization of radio resources among users.
- Packet scheduling and link adaptation, enabling efficient data transmission by dynamically adjusting transmission parameters based on channel conditions.
- Mobility management and handover control, ensuring seamless connectivity when users move between cells.
- Header compression, encryption, and integrity protection, which enhance transmission efficiency while ensuring data security and confidentiality.

2.1.2.2 Evolved Packet Core (EPC)

The Evolved Packet Core (EPC) is the core network architecture of 4G/LTE systems, standardized by the 3GPP as part of the LTE specification. It is mainly responsible for control signaling, mobility management, session management, user data forwarding, policy enforcement, charging, user authentication and many more. These functionalities are handled by a set of key network functions, which can be broadly categorized into two planes: the control plane, responsible for signaling and management tasks, and the user plane, responsible for the actual forwarding of user data traffic.

- Mobility Management Entity (MME): It is responsible for many functionalities, primarily related to control messaging between the core network and the RAN. Some of these include user authentication, security, and registration/deregistration in the network. Additionally, it monitors the movement of UEs and determines which eNodeB is the most suitable based on the UE's coordinates and area coverage. The latter functionality in which the core network makes a decision to transfer a UE to a different eNodeB, is called handover.

- Serving Gateway (SGW): The SGW is the network function responsible for establishing and maintaining the user data path between the eNodeB and the PGW. As the intermediary node between these two entities, SGW is inherently designed to support handover procedures.

- Packet Data Network Gateway (PGW): The PGW acts as the gateway between the EPC and external data networks, receiving packets from the SGW and forwarding them to the destination network, such as the Internet. Beyond data forwarding, it is also responsible for IP address allocation to UEs, as well as defining and enforcing packet filtering and QoS rules.

- Home Subscriber Server (HSS): It is the database in which user data regarding user profiles and identities, as well as authentication data used when users enter the network, is stored.

- Policy and Charging Rules Function (PCRF): It implements policies regarding QoS by determining how data traffic is handled on a per-user and per-service basis, while also managing charging and billing.

In practice, the SGW and PGW are often deployed as a single combined network function known as the Serving and Packet Data Network Gateway (SPGW). As traffic volumes grew, the need to scale the user plane independently of the control plane became apparent, since dedicating the same resources to both planes is inherently inefficient. This led to the introduction of Control and User Plane Separation (CUPS) in LTE (Subsection 2.1.3.1), which split the SPGW into the SPGW-U, handling user data forwarding, and the SPGW-C, handling session control. This separation directly inspired the 5G architecture, where the SPGW-U evolved into the UPF and the SPGW-C's functions were absorbed by the SMF.

2.1.3 Introduction to 5G networks

While 4G was designed to provide very high speeds, low latency and improved user experience, 5G networks have extended these capabilities to both the Core Network (CN) and the Radio Access Network (RAN) to a much greater extent and have set new standards and capabilities that include Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC) and Massive Machine Type Communications (mMTC) [15], [16]. In order to make the transition from 4G networks to 5G, there were many factors that played a key role, both functional and architectural. Some of these key factors are summarized below:

2.1.3.1 5G Key Enablers

Massive MIMO. Massive MIMO is one of the most fundamental technologies in 5G RAN and determines that a base station can have dozens or even hundreds of antennas that use beamforming, different frequency bands and can serve many users simultaneously, thereby increasing the network's capacity and efficiency.

Millimeter Wave Spectrum. 5G expands the available frequencies in the mmWave spectrum, which range from 24 to 100 GHz, and provides much wider frequency bands that now extend from the MHz range to the GHz range. However, as wave frequencies increase, the signal waves are more susceptible to interference and their range decreases, forcing operators to deploy dense small-cell base stations to deal with coverage.

Network Slicing. Network Slicing allows the physical 5G network to be partitioned into logical networks so that different network services can be assigned to different slices based on their requirements. For example, an eMBB slice can be dedicated to bandwidth-demanding applications such as video streaming, while a separate URLLC slice with low-latency serves time-critical applications such as autonomous vehicles, both running concurrently on the same physical infrastructure.

Mobile Edge Computing (MEC). MEC is a technology that enables data storage and processing at the edge of the network. This way, 5G networks are able to process data close to the user, reducing latency and processing times, which is very useful in applications that require real-time responses. A typical example of such a configuration is the deployment of

the UPF to the edge so that the data processing happens there and does not need to go through the cloud.

Control and User Plane Separation (CUPS). As we saw earlier, in the early stages of 4G networks, the control plane and the user plane were closely integrated within the EPC, which limited flexibility in traffic routing and management. 5G networks were designed to support CUPS so that these two planes can be completely separated, allowing the control plane to manage signaling, connection, and user mobility, etc., while the user plane exclusively forwards user-plane data. This separation allows the 5G UPF to be deployed independently at different points in the network, either near the Edge or the Cloud, offering tremendous flexibility to the network with a direct impact on throughput and latency.

2.1.3.2 NG-RAN (Radio Access Network)

The 5G system introduces a redesigned architecture consisting of the Next Generation Radio Access Network (NG-RAN) and the 5G Core (5GC), with a strong emphasis on flexibility, modularity, and cloud-native operation. As illustrated in Figure 2.3, the 5G RAN protocol stack comprises six distinct layers, each performing a specific set of functions. At the bottom of this hierarchy sits the Physical Layer (PHY), which handles directly the transmission and reception of radio signals. Directly above it, the Medium Access Control (MAC) layer oversees scheduling operations and channel access management. The Radio Link Control (RLC) layer, which sits right above MAC, is responsible for data segmentation and guarantees reliable packet delivery. Above RLC, the Packet Data Convergence Protocol (PDCP) layer handles security mechanisms and header compression, and is shared by both the control and the user plane. From this point, the stack splits depending on the plane: on the control plane, the Radio Resource Control (RRC) layer manages connection establishment, mobility procedures and the configuration of radio resources, while on the user plane, the Service Data Adaptation Protocol (SDAP) layer enforces QoS policies across radio bearers. The NG-RAN architecture is built around next-generation base stations referred to as gNBs (Next-Generation NodeBs), along with the UEs they serve. In contrast to 4G networks, where the eNB constituted a monolithic entity responsible for the full protocol stack, the 5G gNB adopts a functionally disaggregated architecture comprising three separate units: the Radio Unit (RU), the Distributed Unit (DU), and the Central Unit (CU). Physically located at the an-

tenna site, the RU is responsible for radio frequency (RF) processing and the lower segment of the PHY layer. The DU, which is typically deployed in close proximity to or collocated with the RU, handles the upper PHY, MAC, and RLC layers. The CU, on the other hand, implements the higher-layer functions (i.e. PDCP, RRC, and SDAP) and maintains a direct interface with the 5G Core network. In most recent 5G architectures the CU unit can be further divided into two components: the CU-CP (Control Plane), which manages RRC and the control-plane functions of PDCP, and the CU-UP (User Plane), which manages SDAP alongside the user-plane functions of PDCP. This disaggregation can be directly related to the CUPS-based separation already discussed in the context of the core network, and aims for the same reasons to completely separate the user plane from the control plane in the 5G RAN, providing the flexibility to deploy network functions at the edge or in the cloud based on performance goals and efficient network management.

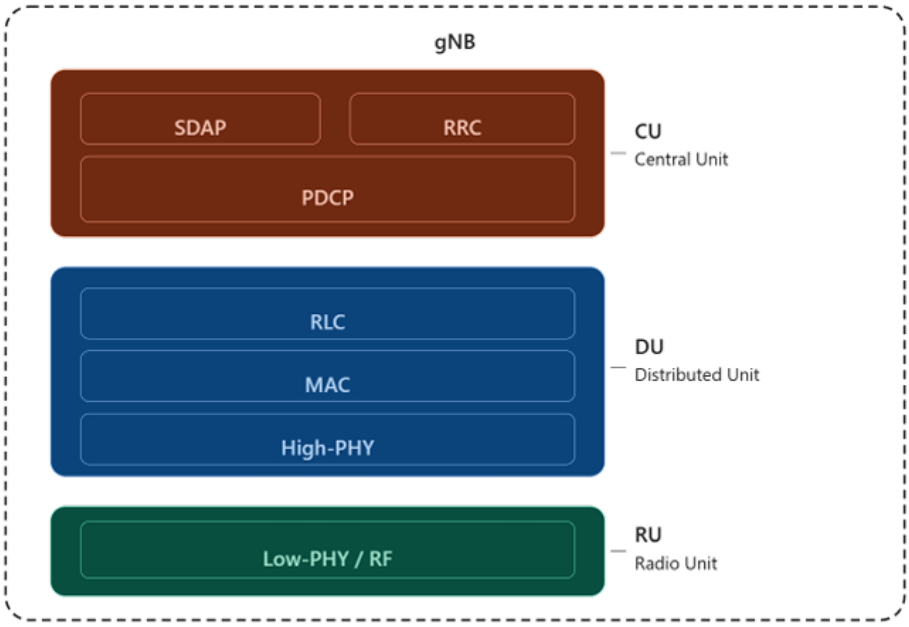


Figure 2.3: 5G RAN protocol stack and functional split across CU, DU and RU

2.1.3.3 5G Core (5GC)

The 5G Core (Figure 2.4) adopts a Service-Based Architecture (SBA), where network functions interact through standardized service-based interfaces using web technologies such as HTTP/2 and RESTful APIs. Each Network Function exposes its services to other func-

tions via a service registry, enabling dynamic discovery and communication. This architecture enhances flexibility, scalability, and interoperability by allowing network functions to be independently developed, deployed, and updated. SBA also facilitates the integration of third-party services and supports automation and orchestration in cloud-native environments.

Key 5G Core network functions include:

- **Access and Mobility Management Function (AMF):** The AMF is the evolution of the 4G MME network function and, as such, serves as the primary access point for UEs to the network. It manages registration/deregistration, authentication, and mobility, and is capable of communicating directly with the gNB through the N2 interface and with the UE through the N1 interface, while also communicating with the rest of the core network functions through the SBI (Service-Based Interface) via RESTful APIs. One key difference from MME is that AMF no longer handles session management; instead, SMF does.

- **Session Management Function (SMF):** The SMF performs the functions of the 4G PGW-C and part of the MME. More specifically, it is responsible for managing PDU sessions (i.e. establishment, modification, and release), which in a nutshell are encrypted communication channels between UEs and UPFs. In addition, it assigns IP addresses to UEs, enforces QoS and charging policies received from the PCF and fully controls the UPF instances via the N4 interface using the PFCP protocol.

- **User Plane Function (UPF):** The UPF is the sole data-plane function of the 5G Core, responsible for packet routing and forwarding between the RAN and external data networks. It enforces QoS policies, performs traffic inspection, and reports usage statistics to the SMF. As previously discussed, its placement within the network directly influences the latency and throughput experienced by end users.

- **Unified Data Management (UDM):** The UDM stores subscriber profiles, authentication credentials, and session-related data. It generates authentication vectors used by the AUSF and provides subscription information to the AMF and SMF during registration and session establishment.

- **Policy Control Function (PCF):** The PCF, whose functionality is retrieved from PCRF, defines and distributes policy rules governing session parameters, QoS flows, and net-

work slice selection. It supplies these rules to both the SMF, which applies them at the UPF, and the AMF, which enforces access and mobility policies.

- **Network Repository Function (NRF):** The NRF serves as the service registry of the 5G Core, maintaining a record of available network function instances along with their capabilities. Network functions register with the NRF upon initialization and subsequently query it to discover and select appropriate peer functions. Through this mechanism, network functions can dynamically discover and communicate with one another in a more loosely-coupled manner, which is a fundamental principle of the SBA.

- **Authentication Server Function (AUSF):** The AUSF is responsible for UE authentication on behalf of the AMF. It interacts with the UDM to obtain authentication vectors and executes authentication procedures such as 5G-AKA or EAP-AKA, thereby establishing a security context used by the AMF to secure communications with the UE.

- **Unified Data Repository (UDR):** The UDR serves as the central data storage function of the 5G Core, providing a structured repository for subscriber, policy, and application data. It acts as the backend database for the UDM, PCF, and NEF, which access and manage their respective data. The separation of storage (UDR) from data management logic (e.g., UDM and PCF) enables each function to scale independently.

- **Network Slice Selection Function (NSSF):** The NSSF supports the AMF in selecting the appropriate network slice for a UE based on subscription information, requested slice identifiers (S-NSSAIs) and operator policies. As already discussed, network slicing enables the 5G infrastructure to be partitioned into multiple logical networks, each provisioned to support specific service types such as eMBB, URLLC, and mMTC.

- **Network Exposure Function (NEF):** The NEF provides a secure and controlled interface that enables authorized third-party applications and edge computing platforms to access 5G Core capabilities. It exposes services such as traffic influence, QoS management, and event monitoring, translating external API requests into internal 5GC procedures while enforcing access control and data privacy policies.

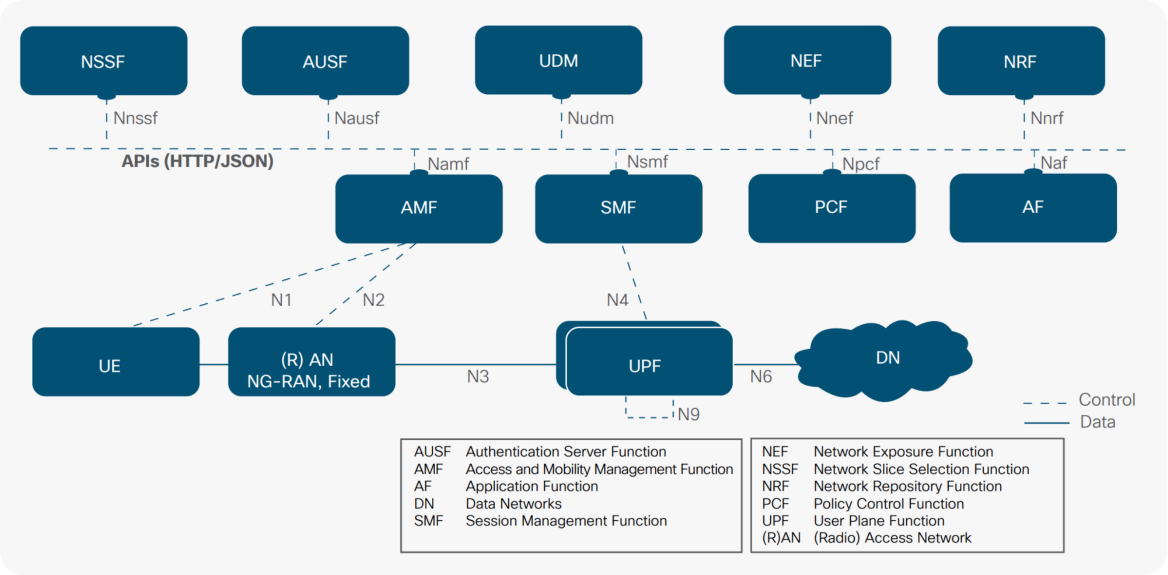


Figure 2.4: 5G Core Service-based Architecture (SBA)

2.1.3.4 UPFs: Design and key Implementations

UPF is the central data-plane component of the 5G Core Network, responsible for all packet processing and forwarding between the UE and external data networks. As a result, the UPF is the performance bottleneck of the 5G Core due to the fact that its throughput, latency, and resource efficiency directly determine the quality of service experienced by end users. In that regard, several UPF implementations have emerged from both the open-source community and academic research, each reflecting a different set of design trade-offs between performance, flexibility, and hardware requirements.

SPGWU (Serving and PDN Gateway User Plane): As already stated, SPGWU is a software-based UPF implementation rooted in the 4G/LTE, adapted for 5G deployments. It operates within the standard Linux kernel networking stack and relies on conventional socket-based packet processing. In that regard, SPGWU offers ease of deployment and high compatibility, making it a widely used baseline deployment in research fields. However, its reliance on kernel-space processing introduces higher latency and limits its throughput under high traffic loads, making it unsuitable for demanding scenarios.

VPP-UPF (Vector Packet Processor): The VPP-UPF is built on the Vector Packet Processing framework developed by Cisco and contributed to the FD.io open-source project. VPP operates in user space and uses the Data Plane Development Kit (DPDK) to bypass

the Linux kernel network stack entirely, processing packets in batches (vectors) to maximize CPU cache efficiency and minimize per-packet overhead. This design yields significantly higher throughput and lower latency compared to kernel-based solutions. Its main trade-off is elevated CPU consumption due to VPP's polling-based mechanism, which continuously occupies CPU cores even when no traffic is generated at all.

P4-Switch UPF: P4-based UPF implementations leverage programmable data-plane hardware (typically high-speed P4 switches) to perform packet forwarding at line rate. P4 is a domain-specific language for expressing packet processing pipelines in programmable ASICs, enabling throughput in the range of tens to hundreds of Gbps with deterministic, sub-microsecond latency. When deployed as a UPF, a P4 switch offloads all data-plane processing to hardware while retaining programmability through the P4 pipeline, achieving a combination of performance and flexibility that software solutions cannot match. On the other hand, P4-switches come at a very high cost and are very complex in terms of deployment and management.

SmartNIC-P4 UPF: SmartNIC-based UPF implementations represent a hybrid approach, by using P4-programmable Smart Network Interface Cards to offload packet processing from the host CPU. Unlike full P4 switches, SmartNICs are embedded in commodity servers and leverage Single-Root I/O Virtualization (SR-IOV) to expose virtual interfaces directly to user space and allow near-hardware-speed packet processing while maintaining the server-based deployment model familiar in cloud environments. SmartNIC-P4 UPFs offer a compelling middle ground: higher throughput than software-only solutions and lower deployment costs than dedicated P4 switches, at the expense of more constrained programmability and limited processing capacity compared to full P4 hardware.

eBPF-UPF: eBPF-based UPF implementations leverage the extended Berkeley Packet Filter (eBPF) framework within the Linux kernel to implement programmable, high-performance packet processing directly in kernel space. By attaching eBPF programs to networking hooks (e.g. XDP or TC layers), packet processing logic can be executed at an early stage in the kernel networking stack, significantly reducing overhead compared to traditional kernel-based approaches. This design enables flexible and efficient data-plane programmability while maintaining tight integration with the operating system, making it well-suited for cloud-native deployments. However, although eBPF-UPFs achieve improved performance over conven-

tional kernel implementations such as SPGWU, they cannot reach their full potential without additional hardware offloading through specialized NICs or FPGAs.

2.2 Multi-Access Networking and Traffic Steering

2.2.1 3GPP and Non-3GPP Access Integration

Mobile networks have traditionally operated over 3GPP-standardized radio technologies, including GSM, UMTS, LTE, and more recently, 5G New Radio. However, a significant portion of mobile data traffic is offloaded to non-3GPP access technologies, most notably Wi-Fi (IEEE 802.11), which operates in unlicensed spectrum and is widely deployed in indoor and dense urban environments, but also increasingly through satellite communications, which extend connectivity to remote and underserved areas beyond the reach of terrestrial infrastructure. Bringing all these access domains together — licensed cellular, unlicensed Wi-Fi, and satellite — under a unified 5G Core has been a long-standing objective of 3GPP, with Releases 15 and 16 providing the most complete realization of this vision to date [17, 18].

The 5G architecture distinguishes between two categories of non-3GPP access. Trusted non-3GPP access refers to operator-controlled and secure networks, such as managed Wi-Fi deployments, whereas untrusted non-3GPP access refers to external networks outside the operator's control, such as public hotspots or residential routers. Each category requires a dedicated interworking function for integration with the 5G Core.

For untrusted non-3GPP access, the Non-3GPP Interworking Function (N3IWF) acts as a secure gateway between the UE and the 5G Core. The UE establishes an IPsec tunnel over the untrusted Wi-Fi network, which is terminated at the N3IWF. The N3IWF then forwards the traffic to the 5G Core over the N2 and N3 interfaces, in a manner analogous to a gNB. Authentication is performed end-to-end via the AMF and AUSF, ensuring that subscriber credentials are never exposed to the untrusted access network.

For trusted non-3GPP access, the Trusted Non-3GPP Gateway Function (TNGF) provides a similar function without the need for IPsec encapsulation. The UE connects to a Trusted Non-3GPP Access Point (TNAP), which forwards traffic to the TNGF. The TNGF then interfaces with the 5G Core over the N2 and N3 interfaces, providing equivalent connectivity to the N3IWF but with lower latency and reduced overhead due to the inherent trust in the access network.

In both cases, the 5G Core treats the UE as a standard subscriber regardless of whether it connects via NR, LTE, or Wi-Fi. Mobility management is handled by the AMF, session control by the SMF, and user-plane forwarding by the UPF, rendering the underlying access technology transparent to the core network. This architectural structure enables seamless multi-access operation and forms the basis for ATSSS functionality that we analyze below.

2.2.2 ATSSS: Access Traffic Steering, Switching, and Splitting

Access Traffic Steering, Switching, and Splitting (ATSSS) is a feature introduced in 3GPP Release 16 that allows a single UE to simultaneously use multiple access networks under the coordinated control of the 5G Core. Unlike earlier multi-access approaches, which relied on application-layer mechanisms or manual configuration, ATSSS is fully integrated into the 5G session management framework. In particular, the SMF configures ATSSS rules during PDU session establishment, while both the UPF and the UE enforce these rules transparently to applications.

ATSSS defines three main traffic management functions:

- **Steering:** assigns new traffic flows to a preferred access network based on operator policies and real-time network conditions. For example, latency-sensitive traffic may be directed to the 5G NR path, while best-effort traffic is routed over Wi-Fi.
- **Switching:** transfers an ongoing flow from one access network to another when the primary path degrades below a predefined threshold, ensuring session continuity without disrupting the transport-layer connection.
- **Splitting:** distributes a single flow across multiple access networks simultaneously, aggregating available bandwidth to improve throughput and/or resilience beyond what a single path can provide. Splitting is typically implemented using either a Multi-Path TCP (MPTCP) proxy or Multipath QUIC (MPQUIC), both of which are co-located with the UPF. MPTCP-based splitting is transparent to applications and applicable to generic TCP traffic, whereas MP-QUIC operates end-to-end without a proxy for applications that natively support the QUIC protocol.

The UPF serves as the network-side anchor for all ATSSS operations. It terminates GTP-U tunnels from both gNB and non-3GPP access paths (e.g., N3IWF or TNGF), applies the

configured steering, switching, and splitting rules on a per-packet basis, and, in the case of MPTCP-based operation, schedules subflows across multiple access links. On the UE side, an ATSSS functional layer above the IP stack enforces corresponding policies for uplink traffic. The SMF ensures coordination between both endpoints by distributing consistent ATSSS rule sets to the UPF via N4/PFCP and to the UE during PDU session establishment.

2.3 Enabling Technologies and Tools for Cloud-Native Networks

2.3.1 SDN and NFV Principles

Prior to the softwarization era, network infrastructures were based on purpose-built, vendor-specific hardware appliances such as routers, firewalls, and load balancers, each running proprietary software tightly coupled to dedicated hardware. This architecture resulted in high deployment costs, limited flexibility, and slow reconfiguration and scaling capabilities. To address these limitations, two key paradigms emerged: Software-Defined Networking (SDN) and Network Function Virtualization (NFV).

Software-Defined Networking (SDN) decouples the control plane, which determines how traffic is handled, from the data plane, which is responsible for packet forwarding based on those decisions. In a typical SDN architecture, a logically centralized controller maintains a global view of the network and programs forwarding behavior in data-plane devices through standardized southbound interfaces such as OpenFlow. This separation enables network behavior to be expressed in software and dynamically reconfigured without modifications to physical infrastructure, allowing operators to deploy and update traffic policies programmatically.

Network Function Virtualization (NFV), introduced by ETSI in 2012, addresses the complementary challenge of hardware dependency in traditional network functions. NFV proposes the implementation of network functions—including firewalls, deep packet inspection systems, and mobile core components—as software instances running on commodity x86 servers. The ETSI NFV architecture is composed of three main layers: the NFV Infrastructure (NFVI), which provides the underlying compute, storage, and networking resources; the Virtualized Network Functions (VNFs), which implement the network functions in software;

and the Management and Orchestration (MANO) framework, which manages the lifecycle of VNFs, including instantiation, scaling, and termination.

Together, SDN and NFV form the foundational pillars of modern cloud-native network architectures. SDN enables programmable and centralized traffic control, while NFV enables hardware-independent deployment of network functions. Their combined use provides the flexibility, scalability, and cost efficiency required for modern mobile networks and forms the basis of the 5G Core architecture and the dynamic UPF management explored in this thesis.

2.3.2 Virtual Machines vs Containers

The two most dominant virtualization technologies (Figure 2.5) that are widely used for softwarized network functions are virtual machines (VMs) and containers and the choice between them gives some unique tradeoffs in terms of performance, scalability, and operational complexity.

VMs are set to emulate full hardware using a hypervisor (e.g., KVM, VMware ESXi) and run a complete guest operating system, providing strong isolation, as each VM has its own kernel and resources, making them suitable for secure multi-tenant environments. However, this design leads to high overhead due to the size of the images, high memory usage, slower boot times (seconds to minutes), and higher communication latency due to full virtualization.

On the other hand, containers take a different approach by virtualizing at the OS level using Linux namespaces and cgroups, meaning that they share the host kernel rather than running a full guest operating system. This makes them significantly lighter, as they start in milliseconds, consume fewer resources, and deliver near-native performance, which is why they have become the best choice for cloud-native network functions that require fast scaling and frequent updates. The main tradeoff is isolation, since containers share the host kernel, a failed container could potentially affect others running on the same host, something that is not a concern with VMs. However, in practice this is largely addressed through orchestrating frameworks like Kubernetes, where security mechanisms such as network policies and RBAC are used to contain and isolate potential failures within the cluster.

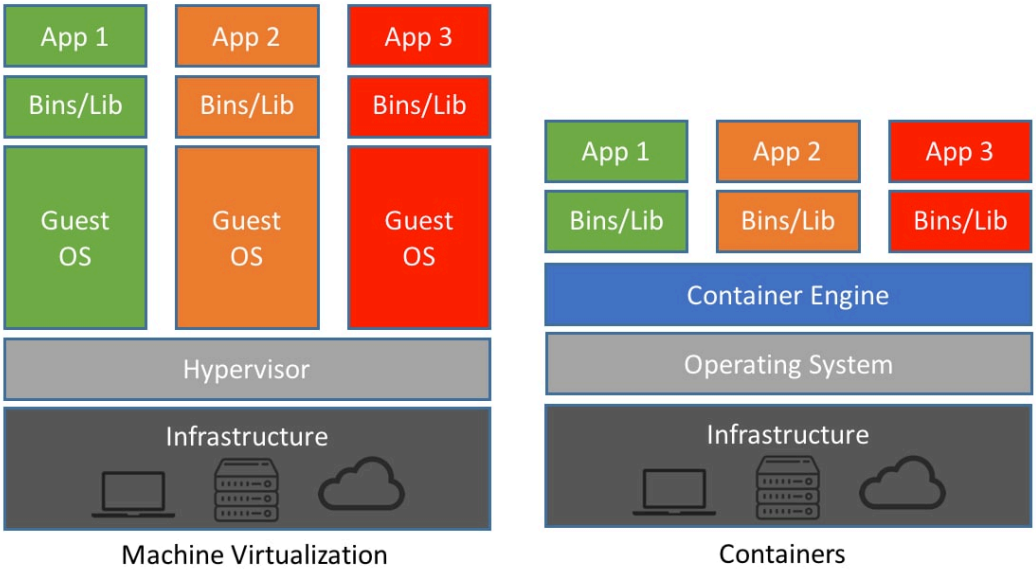


Figure 2.5: Container vs VM Architecture

2.3.3 Docker and Docker Compose

Docker [19] is the most widely adopted container runtime and has played a key role in standardizing the container ecosystem. It consists of three main components: the Docker daemon, which manages the lifecycle of containers on a host; the Docker command-line interface (CLI), which provides the user-facing interface for building, running, and managing containers; and the Docker image registry (most commonly Docker Hub), which enables the distribution of pre-built images and the publication of custom images.

Images and Containers. A Docker image is a read-only, layered filesystem constructed from a `Dockerfile`, which specifies a base image and a sequence of build instructions such as copying files, installing dependencies, and configuring environment variables. Each instruction in the Dockerfile creates a new filesystem layer, which is cached by Docker to optimize subsequent builds by reusing unchanged layers. When a container is instantiated from an image, a thin writable layer is added on top of the immutable image layers, while the underlying image remains unchanged. This copy-on-write mechanism allows multiple containers to share the same base layers in memory and on disk, significantly reducing storage and memory overhead in deployments running multiple instances of the same service.

Networking. Docker provides several networking modes that define how containers communicate with each other and with external networks. In bridge mode, which is the default configuration, Docker creates a virtual bridge on the host and connects each container to it via a virtual Ethernet interface, giving each container its own private IP address. Containers on the same bridge network can talk to each other directly, while external communication is handled through routing and port mapping. In host mode, which is widely used in many of our deployments, the container shares the host's network namespace entirely, bypassing network virtualization and achieving native network performance. This makes it particularly relevant for latency-sensitive applications such as the UPF. Finally, in overlay mode, Docker enables containers running on different hosts to communicate through encapsulated virtual networks, which forms the basis for distributed container networking in orchestrated environments.

Docker Compose. Docker Compose [20] is a tool for defining and managing multi-container applications using a declarative YAML configuration file. A Compose file describes all services that form an application, including their images, environment variables, port mappings, volumes, networks, dependencies, and other runtime parameters. The `docker compose up` command parses this configuration and deploys all services, automatically creating the required networks and volumes while respecting service dependencies. In the context of 5G Core deployments, Docker Compose is commonly used to instantiate a complete OAI-CN5G stack, including components such as the AMF, SMF, UPF, UDM, AUSF, NRF, and supporting services, using a single command. This makes it particularly suitable for development, testing, and small-scale experimental environments. For large-scale production deployments across multiple hosts, Docker Compose is typically replaced by Kubernetes, which is discussed in the following section.

2.3.4 Kubernetes and Service Orchestration

In scenarios where containerized deployments exceed the capacity of a single host, a dedicated orchestration platform becomes necessary to automate scheduling, scaling, networking, and lifecycle management across a cluster of machines. Kubernetes [21], originally developed by Google and donated to the Cloud Native Computing Foundation (CNCF) in 2015, has become the de facto standard in this domain. It abstracts a heterogeneous set of physical or virtual machines into a unified compute fabric and exposes a declarative API through which

users define the desired state of their applications. The Kubernetes control plane continuously reconciles the actual cluster state with this desired state.

2.3.4.1 Architecture.

A Kubernetes cluster is composed of two logical planes: the control plane and the data plane.

Control Plane: The control plane acts as the brain of the cluster and consists of the following core components:

- **API Server:** The central management interface through which all interactions with the cluster are performed via a RESTful API.
- **etcd:** A distributed key-value store that maintains the complete cluster state, including both desired and observed states of all objects.
- **Scheduler:** Responsible for assigning newly created Pods to suitable nodes based on resource availability, affinity rules, taints, and tolerations.
- **Controller Manager:** Runs multiple control loops that continuously compare the desired state with the observed state and take corrective actions when needed.

Data Plane: The data plane consists of worker nodes, which are responsible for running the actual application workloads and each one utilizes the following components:

- **kubelet:** An agent that communicates with the API Server, ensures that containers defined in Pod specifications are running correctly, and reports their status.
- **kube-proxy:** Maintains network rules and enables Service abstractions by routing traffic to the appropriate Pod endpoints.
- **Container Runtime:** Typically *containerd*, *docker* or *CRI-O*, responsible for pulling images and running containers as instructed by the kubelet.

2.3.4.2 Core Abstractions.

Kubernetes organizes workloads through a set of hierarchical abstractions, each serving a distinct purpose in managing containerized applications:

- **Pod:** The smallest deployable unit in Kubernetes, consisting of one or more containers that share the same network namespace and storage volumes, meaning they are always scheduled and run together on the same host. Pods are inherently ephemeral and can be created, terminated, or replaced at any time.
- **Deployment:** Defines the desired number of identical Pod replicas and manages their entire lifecycle, including rolling updates and rollback strategies. It is the standard abstraction for stateless workloads, which covers most 5G Core network functions.
- **StatefulSet:** Provides each Pod with a stable network identity and persistent storage that survives restarts and rescheduling, making it suitable for applications that require persistent state.
- **DaemonSet:** Ensures that exactly one Pod runs on every node in the cluster, making it well suited for infrastructure-level services such as monitoring agents or networking components that need to be present across the entire cluster.
- **Service:** Provides a stable network endpoint for a dynamically changing set of Pods, using label selectors to group them and exposing them through a fixed virtual IP address and DNS name, while automatically distributing incoming traffic across all available replicas.
- **ConfigMaps and Secrets:** Used to decouple configuration data and sensitive information, such as credentials or certificates, from the container images themselves, injecting them into Pods at runtime either as environment variables or as mounted files.

2.3.4.3 Networking Plugins.

Kubernetes relies on Container Network Interface (CNI) plugins to manage Pod networking, determining how Pods are assigned IP addresses and how traffic flows between them across nodes. Two CNI plugins are particularly relevant in 5G core deployments.

Flannel is a lightweight overlay network plugin that assigns each node in the cluster a dedicated subnet and encapsulates inter-node traffic using VXLAN, creating a flat, cluster-wide network in which every Pod can reach any other Pod regardless of which node it runs on. Flannel is designed for simplicity: it requires minimal configuration and integrates transparently with the Kubernetes networking model, making it a common default choice for clusters

where straightforward Pod-to-Pod connectivity across nodes is the primary requirement. In 5G core deployments, Flannel provides the baseline cluster network over which control-plane traffic between network functions such as the AMF, SMF, UDM, UPF and NRF flows.

Multus is a meta-CNI plugin that enables a Pod to be attached to multiple network interfaces simultaneously, going beyond the single interface that standard CNI plugins provide. Rather than replacing the primary CNI, Multus acts as a coordinator: it calls the primary CNI plugin to set up the default interface and then invokes one or more additional CNI plugins to attach secondary interfaces as specified in the Pod's annotations. This capability is essential for 5G core network functions that must separate different traffic types onto distinct network paths. The UPF, for example, requires at minimum three separate interfaces: one for the N4 control interface toward the SMF, one for the N3 data interface toward the gNB, and one for the N6 interface toward the external data network. Without Multus, such multi-interface configurations are not achievable within a standard Kubernetes deployment.

2.3.4.4 Scaling.

Kubernetes supports two primary forms of autoscaling. The Horizontal Pod Autoscaler (HPA), which adjusts the number of Pod replicas based on observed metrics such as CPU utilization, memory usage, or custom application-level metrics exposed through the Metrics API and the Vertical Pod Autoscaler (VPA), which in contrast, modifies the resource requests and limits of individual containers according to their observed usage, thereby improving resource efficiency without changing the number of replicas. For network function deployments, the HPA is generally preferred, as increasing the number of replicas directly translates into higher packet-processing capacity and higher flexibility on distributing replicas across the network.

2.3.4.5 Prometheus

Prometheus is an open-source monitoring and alerting toolkit that has become the standard for metrics collection in Kubernetes environments. It works by periodically scraping metrics endpoints exposed by running services and storing the collected data in a time-series database. In the context of 5G deployments, Prometheus can be used to monitor the health and performance of core network functions, tracking things like packet throughput, session counts, and resource utilization over time and many more. It also supports a query language called PromQL, which allows operators to define custom alerting rules and dig into specific

metrics when troubleshooting.

2.3.4.6 Grafana

Grafana is an open-source visualization platform that is widely deployed alongside Prometheus and provides the interface through which operators can actually visualize the data retrieved from Prometheus. It connects to Prometheus as a data source and allows users to build interactive dashboards that display real-time and historical metrics.

2.3.5 Cloud-Native Network Functions (CNFs)

A Cloud-Native Network Function (CNF) is a network function implemented as containerized microservices, designed to be deployed and managed through orchestration platforms such as Kubernetes. CNFs extend the concept of VNFs, which are monolithic applications running inside virtual machines with little architectural adaptation for cloud environments. Unlike VNFs, CNFs are decomposed into loosely coupled microservices that communicate via APIs, allowing each component to be independently deployed, scaled, or updated. Being typically stateless, with persistent data stored externally, they can scale horizontally and recover from failures quickly. They also expose health and readiness probes that allow Kubernetes to automatically detect and respond to failures, and are configured through declarative manifests rather than manual intervention. In the 5G Core, all network functions — AMF, SMF, UPF, UDM, PCF, AUSF, and NRF — can be deployed as CNFs, each running as an independent containerized service managed by the orchestration platform. Among these, the UPF, which is the most demanding due to its data-plane role, can be instantiated, scaled or terminated by Kubernetes based on traffic demand.

2.4 Machine Learning for Network Management

Machine Learning (ML) is a subfield of artificial intelligence concerned with building models that learn patterns from data rather than following explicitly programmed rules. Instead of manually defining system behavior, ML models discover structure by processing large volumes of historical observations and adjusting their parameters accordingly. This approach has grown substantially in recent years, driven by the increasing availability of data,

advances in computational hardware, and the development of model architectures capable of capturing complex relationships.

2.4.1 ML in Beyond-5G and 6G Networks

The emergence of beyond-5G and 6G networks has elevated ML from an optional optimization layer to a core architectural principle, with intelligence embedded natively into the network rather than applied on top of it. In 5G, this is partially realized through the Network Data Analytics Function (NWDAF), which provides standardized analytics and predictive capabilities within the core network, while 6G systems are expected to take this further by embedding AI/ML across all network layers.

A key application of ML in this context is time-series forecasting of network metrics such as throughput, latency, and session load. Accurate prediction of traffic demand enables proactive resource management and reduces reliance on reactive scaling mechanisms. The models presented in this thesis are evaluated in this setting, specifically for predicting traffic load to support dynamic UPF scaling in a cloud-native 5G Core.

2.4.2 Model Architectures

This section presents the four model architectures used in this thesis in order to mainly predict future throughput demands, covering both deep learning-based sequence models and a gradient boosting approach.

2.4.2.1 Gated Recurrent Unit (GRU)

The GRU is recurrent architecture and processes a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ by maintaining a hidden state $\mathbf{h}_t$ that summarizes past information. It uses two gates to control information flow: the reset gate $\mathbf{r}_t$, which determines how much past information is discarded when forming the candidate state, and the update gate $\mathbf{z}_t$, which balances the contribution of the previous hidden state and the new candidate $\tilde{\mathbf{h}}_t$.

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2.1}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2.2}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{2.3}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{2.4}$$

The GRU is a computationally efficient alternative to the LSTM, achieving comparable performance on many sequential tasks while using fewer parameters, which makes it well suited for scenarios where training data or computational resources are limited.

2.4.2.2 Long Short-Term Memory (LSTM)

The LSTM extends the basic recurrent architecture by introducing a dedicated memory cell $\mathbf{c}_t$, which allows the model to retain information over longer time horizons. Three gates govern the flow of information: the forget gate $\mathbf{f}_t$ decides what past memory to discard, the input gate $\mathbf{i}_t$ controls what new information is written to the cell, and the output gate $\mathbf{o}_t$ determines what part of the memory is exposed as the hidden state.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.5}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.6}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.7}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.8}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.9}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.10}$$

The explicit memory cell makes the LSTM particularly effective at capturing long-range dependencies in sequential data, which is especially relevant in network traffic forecasting where patterns may span across multiple time scales.

2.4.2.3 CNN-LSTM

The CNN-LSTM architecture combines convolutional feature extraction with sequential modeling. Given an input sequence $\mathbf{x} \in \mathbb{R}^{T \times F}$, a convolutional layer first extracts local temporal patterns across the feature dimension:

$$z_j^{(i)} = \text{ReLU}\left(\sum_{f=1}^{F} \mathbf{w}_j^{(f)} * \mathbf{x}^{(f)} + b_j\right) \tag{2.11}$$

where $\mathbf{w}_j^{(f)}$ is the j-th filter applied to feature f, and b_j is the bias term. The convolutional layer captures short-term local patterns, which are then passed to an LSTM to model longer-term temporal dependencies.

By combining the local pattern extraction of CNNs with the sequential memory of LSTMs, this architecture is able to exploit both short-term structure and long-term trends in the input sequence, which can be advantageous when the traffic signal contains multi-scale temporal patterns.

2.4.2.4 XGBoost

XGBoost is a gradient boosting method that builds an ensemble of regression trees over hand-crafted features derived from past observations. The feature vector at time t is constructed as:

$$\phi_t = [x_{t-1}, \ldots, x_{t-p}, \mu_{t,w}, \sigma_{t,w}, \text{ToD}_t, \text{DoW}_t] \tag{2.12}$$

where x_{t-i} are lagged values capturing short-term history, $\mu_{t,w}$ and $\sigma_{t,w}$ are the rolling mean and standard deviation over a window w, and ToD_t and DoW_t encode temporal context. The final prediction is obtained by summing the outputs of all M trees:

$$\hat{y}_t = \sum_{m=1}^{M} f_m(\phi_t) \tag{2.13}$$

where f_m denotes the m-th regression tree in the ensemble. Unlike the deep learning models above, XGBoost does not learn representations from raw sequences but instead relies on explicitly engineered features, making it a strong and interpretable baseline that is particularly effective when the relevant temporal patterns can be captured through hand-crafted statistics.

2.5 Large Language Models

Large Language Models (LLMs) are large-scale neural networks trained on vast text corpora to predict token sequences. At scale, they exhibit emergent capabilities such as reason-

ing, code generation, and in-context learning. These properties make them a strong candidate for automating 5G experimentation workflows in large-scale infrastructures, where the complexity of the system creates a high barrier to entry for new users.

2.5.1 Transformer Architecture

The Transformer is the architectural foundation of modern LLMs and allows for processing sequences in parallel using self-attention, which allows the model to capture relationships between any two tokens directly, regardless of their distance. Stacked layers progressively build more abstract representations, and decoder-only variants trained via next-token prediction form the basis of most modern LLMs.

2.5.2 Prompting

Prompting is the primary way of interacting with an LLM, where the model is given a natural language description of a task and is expected to produce the appropriate response without any changes to its parameters. This makes it a lightweight and flexible approach for directing model behavior, which in the context of this thesis is used to translate high-level experimental intents into concrete configurations and commands within SLICES-RI experimental infrastructure.

2.5.3 Fine-Tuning

Although pre-trained LLMs are capable of handling a wide range of tasks through prompting alone, their performance on domain-specific tasks can be significantly improved by adapting their parameters to relevant data, a process known as fine-tuning. This allows the model to internalize domain-specific knowledge and terminology, which is particularly relevant when targeting specialized workflows such as 5G network management and experimentation.

2.5.4 Retrieval-Augmented Generation

A core limitation of LLMs is that their knowledge is fixed at training time, making them prone to generating outdated or incorrect information when queried about evolving systems. RAG addresses this by retrieving relevant documents from an external knowledge source at inference time and incorporating them into the model's context before generating a response.

This is especially valuable in a dynamic infrastructure like SLICES-RI, where the model needs access to current documentation, new experiments and system state to provide accurate and grounded responses.

2.6 Experimental Platforms

A core concept of this thesis is that all experiments are conducted in realistic, reproducible, and fully controllable environments. To this end, the experiments presented here are built on top of two complementary open experimental ecosystems: the SLICES-RI large-scale research infrastructure, which provides the distributed hardware resources needed for real-world experimentation, and open-source 5G software implementations, which offer full control and transparency over the entire network stack. Together, these platforms ensure that the results obtained throughout this thesis are grounded in realistic 5G deployments while remaining reproducible and independent of proprietary components.

2.6.1 SLICES-RI Infrastructure

SLICES-RI (Scientific Large-scale Infrastructure for Computing/Communication Experimental Studies — Research Infrastructure) is a European research infrastructure [22] that federates heterogeneous experimental testbeds across 16 countries under a unified access and management framework. It entered the ESFRI roadmap in 2021, making it the first infrastructure of its kind specifically dedicated to supporting research in digital sciences and ICT, an area that had previously lacked a dedicated European-scale facility. SLICES-RI is designed as a fully programmable and virtualized, remotely accessible platform that brings together advanced computing, storage, and networking resources interconnected by dedicated high-speed links. It covers a broad research scope including networking protocols, radio technologies, cloud and edge computing, IoT, NFV, and 5G/6G experimentation, with a strong emphasis on reproducibility and fair data management throughout the full research lifecycle. The infrastructure is organized around a set of Blueprints, which are replicable combinations of software, hardware, and methodology designed to enable standardized experimentation across partner sites. Within this infrastructure, two testbeds are used in this thesis:

NITOS. NITOS is a large-scale heterogeneous wireless and wired experimentation facility hosted at the University of Thessaly, Greece [23] and is remotely accessible to researchers around the world. The testbed comprises over 100 operational nodes spread across three distinct deployments: an outdoor testbed at the exterior of the University campus supporting heterogeneous wireless technologies including 5G-NR, Wi-Fi, WiMAX, and LTE, an indoor RF-isolated testbed consisting of Icarus nodes deployed in a controlled environment that eliminates external interference, and an office testbed located at the CERTH facilities in Volos. The testbed also includes Software-Defined Radio (SDR) components based on USRP devices, which allows researchers to program physical layer features and conduct dedicated PHY-layer or cross-layer research. Resource management is handled through the SLICES-RI Portal and allows users to reserve slices of the testbed, including nodes, access points, base stations, and frequency spectrum, ensuring isolation between concurrent experiments.

OneLab / LIP6. OneLab is a federated experimental facility that provides a single point of access to a large collection of heterogeneous testbeds spread across multiple European institutions. It was established in 2014 and is administered from the LIP6 laboratory at Sorbonne Université in Paris. The federation brings together diverse platforms including wireless testbeds, IoT facilities, cognitive radio testbeds, and internet overlay networks, collectively providing hundreds of nodes for experimentation.

2.6.2 OpenAirInterface

OpenAirInterface (OAI) is an open-source implementation of 3GPP cellular standards developed and maintained by the OpenAirInterface Software Alliance [24]. It is one of the most mature and widely used open-source platforms in the cellular research community, providing both a full 5G NR/LTE radio access network stack (OAI-RAN) and a complete 5G Core implementation (OAI-CN5G). The 5G Core includes all major network functions and is fully compliant with 3GPP Release 15.

OAI-CN5G supports multiple UPF backends, including a standard SPGWU-based implementation and a high-performance VPP-based dataplane, which can be selected depending on the performance requirements of the experiment. All network functions are containerized using Docker and can be orchestrated via Docker Compose for single-node deployments or Kubernetes for distributed multi-node scenarios. On the radio access side, OAI supports both

real SDR hardware and integration with UERANSIM [25], a software-based UE and gNB simulator that enables fully emulated end-to-end 5G experiments without the need for physical radio equipment. This flexibility makes OAI suitable for a wide range of experimental scenarios, from controlled software-only setups to realistic deployments involving actual radio hardware.

2.6.3 Free5GC

Free5GC is an open-source 5G Core implementation developed at National Yang Ming Chiao Tung University in Taiwan [26]. Written entirely in Go, it implements all major 3GPP Release 15 network functions following a strict service-based architecture where each function operates as an independent process communicating through HTTP/2-based RESTful APIs, closely mirroring the architectural principles defined by 3GPP.

Beyond the standard core functions, Free5GC implements the N3IWF and TNGF, which enable UEs to connect to the 5G Core via untrusted and trusted non-3GPP access networks respectively, such as Wi-Fi or satellite links. These functions also support ATSSS, allowing a UE to simultaneously use multiple access networks and steer traffic across them based on policies enforced at the UPF. This makes Free5GC a particularly relevant platform for research involving heterogeneous access integration, which is directly relevant to the work presented in this thesis.

Chapter 3

Evaluation of UPF Implementations in Real-World 5G Networks

3.1 Introduction

As established in Chapter 2, the User Plane Function is the data-plane heart of the 5G Core, responsible for forwarding every packet exchanged between a UE and an external network. Its performance directly determines the throughput, latency, and reliability experienced by end users, making it the most performance-critical component of the 5GC. The UPF can be realized through fundamentally different packet processing architectures that span from software-based solutions such as SPGWU and VPP-UPF, which run on commodity servers, to hardware-accelerated approaches such as P4 switches and SmartNICs, which offload processing to dedicated chips. Each design choice carries distinct trade-offs in throughput, latency, CPU/energy consumption, cost and deployment complexity. Despite the availability of these implementations, a systematic, real-world comparative evaluation conducted within a fully operational 5G network had not been established in the literature. Existing studies either evaluated UPF solutions in isolation, relied on simulation environments, or focused on a single performance metric without considering the broader operational picture. Before designing any dynamic management or optimization mechanism, it is essential to first establish a clear picture of how each UPF implementation performs under real network conditions — how much traffic it can handle, where it struggles, and what it costs in terms of computational resources, deployment and maintenance. Without this empirical foundation, any subsequent control or orchestration logic would rest on assumptions rather than evidence. This is where

the research presented in this thesis begins.

To this end, this chapter presents a comprehensive benchmarking study of four distinct UPF implementations — SPGWU, VPP-UPF, P4-Switch, and SmartNIC-P4 — deployed in a real end-to-end 5G network built on the OpenAirInterface framework and evaluated on the NITOS and LIP6 testbeds of the SLICES-RI infrastructure. Each implementation is deployed under identical network conditions and subjected to the same traffic scenarios, allowing direct, fair comparison across throughput, latency, packet loss, and CPU utilization. The insights gained from this evaluation form the empirical foundation on which the dynamic or proactive UPF management mechanisms developed in subsequent chapters are built.

The rest of this chapter is organized as follows. Section 3.2 reviews related work on UPF implementations and performance evaluation. Section 3.3 describes the system architecture and experimental setup, while in Section 3.4 we present the experimental results. Finally, Section 3.5 draws conclusions and directions for future work.

3.2 Related Work

5G introduces key changes to the core network by adopting a modular and cloud-based approach. A notable shift is the move from a traditional telecom monolithic setup (and closed-in vendor-specific hardware in many cases) to a Service-Based Architecture, which offers more flexibility and opportunities for network enhancements. Such innovation enables the network operators to deploy the network functions even as microservices in matching modern cloud-native technologies like Docker and Kubernetes [27]. In this way the network functions can be easily managed, scaled [28], and migrated [29] to cover the needs of the end-users. Furthermore, the softwarization of the network functions has opened the horizon for further novelties in the telecom architecture such as the Control and User plane separation [30]. This in turn brings new innovations mainly on the UPF side, which have the sole purpose of increasing bandwidth, reducing latency, adding flexibility and improving the Quality of Service. Multiple Access Edge Computing takes advantage of such CUPS architectures by placing the user plane part of the UPF at the edge of the network, while the control plane side can run in the cloud [31]. Although MEC has enabled low-latency access to 5G networks suitable for ultra-reliable low-latency communication features, the bottleneck of performance still persists in the RAN configuration but also in the UPF implementation.

Focusing on the UPF implementations, several solutions are utilizing the Data Plane Development Kit (DPDK), a technology that contains a set of libraries and drivers for fast packet processing. One key advancement of the DPDK technology is that it is designed to bypass the traditional kernel-based network stack, allowing for fast packet processing directly in user space. By skipping the kernel's network stack, DPDK can achieve lower latency and higher throughput for packet processing [32]. To achieve high-performance UPF in [33] the authors proposed optimizations, including the use of DPDK for high throughput requirements. Their DPDK-based UPF solution achieved a performance of up to 40Gbps in a simulation environment. In [34], the authors highlighted the challenges of using DPDK in edge computing, particularly its CPU-intensive polling mechanism. They introduced an alternative in-kernel 5G UPF solution aligned with 3GPP Release 16, suitable for MEC deployments. This solution taps into the extended version of the Berkeley Packet Filter (eBPF)/eXpress Data Path (XDP), a kernel-based technology, showcasing its potential as a viable alternative to traditional DPDK methods. eBPF is a revolutionary technology within the Linux kernel that allows the execution of user-defined programs in a safe manner without changing kernel source code or loading modules [35]. XDP is a framework within the Linux kernel that leverages eBPF for high-performance packet processing directly at the device driver level before the kernel network stack gets involved [36] [37].

Several works also employ hardware solutions with the most promising one being the P4-driven UPF. P4 switches, especially when implemented in hardware, can process packets at a line rate, while maintaining the programmability of the flows, ensuring that the high throughput and low latency requirements of 5G networks are met. The authors in [38] showcase the performance benefits of using a P4 switch as a UPF and the implementation challenges. Additionally, in [39] the paper compares modern packet processing acceleration technologies, including P4, and their respective latency performance. The results highlight the advantages of hardware-based P4 implementations in terms of reduced tail latency.

In this work, we progress beyond existing literature by deploying a cloud-native end-to-end 5G network, integrating all the open-source UPF solutions in the deployed network, and evaluating them in a real-world setup. Specifically, we integrate all the softwarized UPFs, by starting with the legacy SPGWU, then moving to a DPDK-based solution utilizing Vector Packet Processing (VPP) UPF, and finally utilizing a P4-Switch to function as a UPF. Moreover, as an alternative to DPDK-based solutions, we use the virtual interfaces provided

by P4-SmartNICs through the single-root input/output virtualization function. This capability allows real-time processing of packets in user-space. On top, we deploy the UPF function and compare its performance with the rest of the solutions. Finally, we evaluate all the employed solutions in terms of throughput, latency, packet loss, and CPU consumption.

3.3 System Architecture

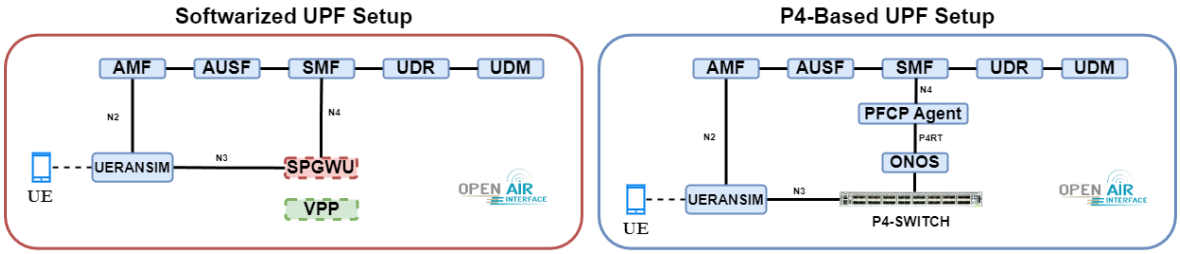


Figure 3.1: Experimental Setup: Softwarized UPFs vs P4-Switch-based UPF

Our overall setup consists of two separate end-to-end 5G architectures; the softwarized UPF setup and the P4-Switch UPF setup. Fig. 3.1 illustrates both architectures. Both setups rely on OpenAirInterface's core network for the corresponding control network functions such as AMF, AUSF, SMF, UDR, and UDM. The way they differ is on the UPF implementation, since in the softwarized setup, the UPF is either spgwu-tiny or VPP, which are both software-based UPF solutions. The SmartNIC-UPF can be described by the softwarized setup as well, as the OAI architecture is the same, with the only exception that it uses the SRIOV interface provided by the SmartNIC-P4 card for the GPRS Tunneling Protocol (GTP) session. In contrast, the P4-Switch setup uses a hardware P4 switch to operate as a UPF. All network functions are deployed as microservices using Docker container runtime. To connect the end-users to the network and in order to perform stress-testing to each UPF technology we utilized UERANSIM. Each setup is deployed in real-world machines at our premises at LIP6-Sorbonne University and the NITOS testbed in Greece, which are both part of SLICES-RI. The deployment specifications are summarized in Tables 3.1, 3.2. Below we analyze the respective UPF solutions that we integrated along with the various additional technologies. All implementations and each setup can be reproduced by following the instructions and

deploying the codes. [1]

Table 3.1: LIP6 Experimental Setup

System	Description
CPU	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz
Cores	64
RAM	240GB
Operating System	Ubuntu 22.04.2 LTS
Linux Header Image	5.19.0-1022-lowlatency
MTU	1500
5G-Core Network	OpenAirInterface
UPF Types	SPGWU, VPP, SD-Fabric
P4 Switch	Wedge100BF-32QS
QFSP Cable	Mellanox 100Gb QFSP28
5G-RAN	UERANSIM
5G-UE	UERANSIM

3.3.1 SPGWU-Based UPF

The Serving and Packet Data Network Gateway User plane is a core element of 4G/LTE networks and is responsible for routing and forwarding user data packets. Based on 3GPP cellular specifications, OpenAirInterface has developed ‘oai-spgwu-tiny’, which is an improved version of SPGW-U and initially was compatible only with 4G/LTE networks. The advent of 5G has triggered OAI to adapt its operation and integrate it into 5G networks as well. However, in many cases, this monolithic technical approach does not satisfy the high demands of 5G networks, and as a result, other UPF technologies are being considered.

[1] Information about each setup available at: `https://github.com/schristakis/UPF-Evaluation`

Table 3.2: NITOS Experimental Setup

System	Description
CPU	11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz
Cores	16
RAM	32GB
Operating System	Ubuntu 18.04.6 LTS
Linux Header Image	4.15.0-213-generic
MTU	1500
SmartNIC Type	Netronome Agilio CX 25GbE
5G-Core Network	OpenAirInterface
UPF Types	SmartNIC-P4 UPF
5G-RAN	UERANSIM
5G-UE	UERANSIM

3.3.2 VPP-Based UPF

Vector Packet Processing is a high-speed packet-processing framework that can operate on a variety of CPUs. As already mentioned, VPP differs from traditional packet processing as it processes multiple packets at once using a technique called vector processing, which utilizes the SIMD capabilities of modern processors to handle several packets in parallel. This approach increases the data plane performance, achieving high throughput and low latency that are comparable to, and sometimes surpass, specialized hardware solutions. VPP also leverages technologies like the Data Plane Development Kit, which allows direct access to network interface cards (NICS), bypassing the operating system's networking stack, and reducing additional overhead. Furthermore, it allows the distribution of the packet processing workload across numerous CPU cores to optimize hardware usage and scale in overload conditions.

When compared to typical monolithic SPGWU systems in 5G networks, VPP provides numerous additional benefits. As a result, it has been utilized for User Plane Function solutions in 5G networks, to manage high-speed data traffic and low response times. Its high-performance packet processing capabilities, make it particularly suitable for the data-intensive

and dynamic nature of 5G services. Additionally, VPP's adaptability allows for the rapid deployment of network slices, supporting the diverse and evolving use cases of 5G, from enhanced mobile broadband (eMBB) to ultra-reliable low latency communications (URLLC) and massive machine type communications (mMTC).

To integrate this solution we relied on the OpenAirInterface's implementation of VPP-UPF. This implementation utilizes a VPP User Plane Gateway (UPG) [40] that relies on the FD.io VPP plugin [41]. The user-plane functionalities comply with the 15th release of 3GPP.

3.3.3 P4-Switch-Based UPF

P4 is a programming language that enables packet switches to modify the way packets are processed, offering the flexibility to adapt to changing protocols and services, a key feature of 5G networks. The use of P4 upgrades packet switches from the limitations of fixed-mode hardware, allowing them to operate using reconfigurable rules based on the network's requirements. The ability to transform static hardware devices into real-time programmable devices inspired their usage and incorporation into UPF operations. This integration brings significant advantages as we can combine speed and fast hardware processing with programming flexibility.

Our P4-switch setup, which can be observed in Fig. 3.2, consists of several components, regarding Open Network Operating System (ONOS), management switches, Stratum and the Packet Forwarding Control Protocol (PFCP) Agent. ONOS is the network controller that manages the P4-Switch-based User Plane Function (P4-Switch UPF). It provides a platform for network configuration and administration, allowing for dynamic programming of network devices and real-time modifications to network policies. ONOS allows us to operate and control the setup, regardless of the underlying hardware, which is crucial for easy management and deployment. Furthermore, we utilize Stratum, a software that indicates the physical switches on how to handle the data that are being forwarded. We could consider Stratum as the component that translates the instructions received from ONOS, into specific, actionable commands for the hardware switch. PFCP Agent was developed to allow the control plane (i.e., the SMF) to communicate with and manage P4-programmable switches. More specifically, it is deployed to manage the communication between the SMF and the P4-Switch UPF functions, in order to successfully establish the PFCP session. As depicted in Fig. 3.2, ONOS, and OpenAirInterface functions are deployed on the server, whereas the UPF and Stratum are

executed within the P4 switch itself. The server and the switch are interconnected and linked to a management switch, which serves as a centralized unit for administration and control. The *"p4rtc_server"* is the IP address that the P4-Switch UPF application (UP4-APP) is running on the ONOS controller. PFCP Agent communicates with ONOS (UP4-APP) via this IP, over the management network. For the user-plane network, we connected a QFSP cable from our server to the P4 port. The user-plane interface is the *"access_ip"* from the side of the P4 switch playing the role of the N3 interface. From the server side, the interface plays the role of the external data network (N6 interface).

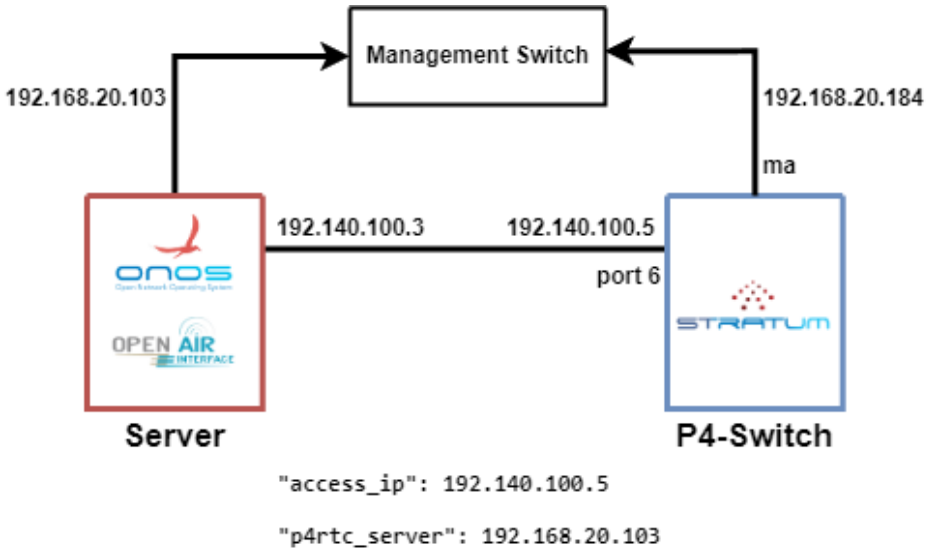


Figure 3.2: P4-Switch-based UPF Experimental Architecture.

3.3.4 SmartNIC-P4 UPF

Our setup for the SmartNIC-based SPGWU consists of two powerful machines provided by NITOS testbed. One is used to deploy the RAN functions and the other (which incorporates the SmartNIC-P4 card) is utilized to deploy the core functions of our 5G network. In order to provide low-latency access to the network functions, we use the virtual interfaces that are deployed along with the SRIOV function for the card and provide access directly to the packets written to the network in user-space. The card also supports programmable flows, which in our case are set to transfer the packets received on the physical port to the virtual port on top of which the UPF is instantiated. The general functionality remains the same as

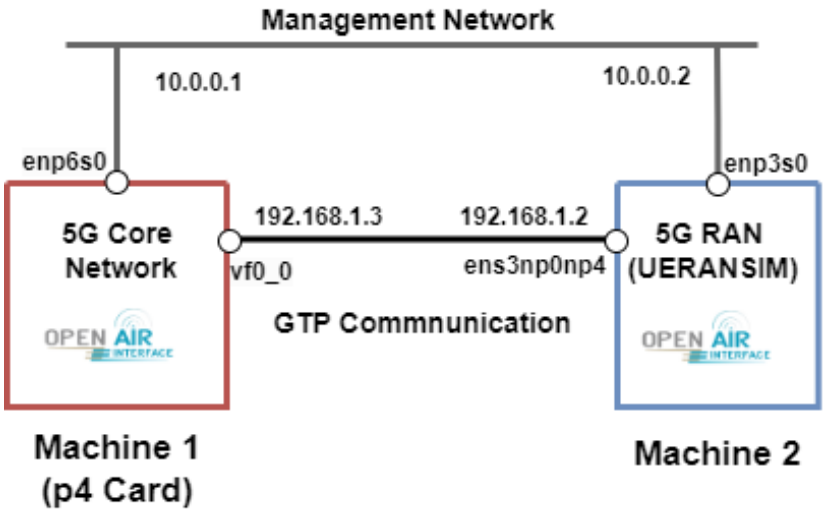


Figure 3.3: SmartNIC-P4 UPF Experimental Architecture.

the one of legacy SPGWU, with the only exception that the GTP traffic is forced to go through the SmartNIC-P4 card's interface, to get the best performance possible (Fig. 3.3).

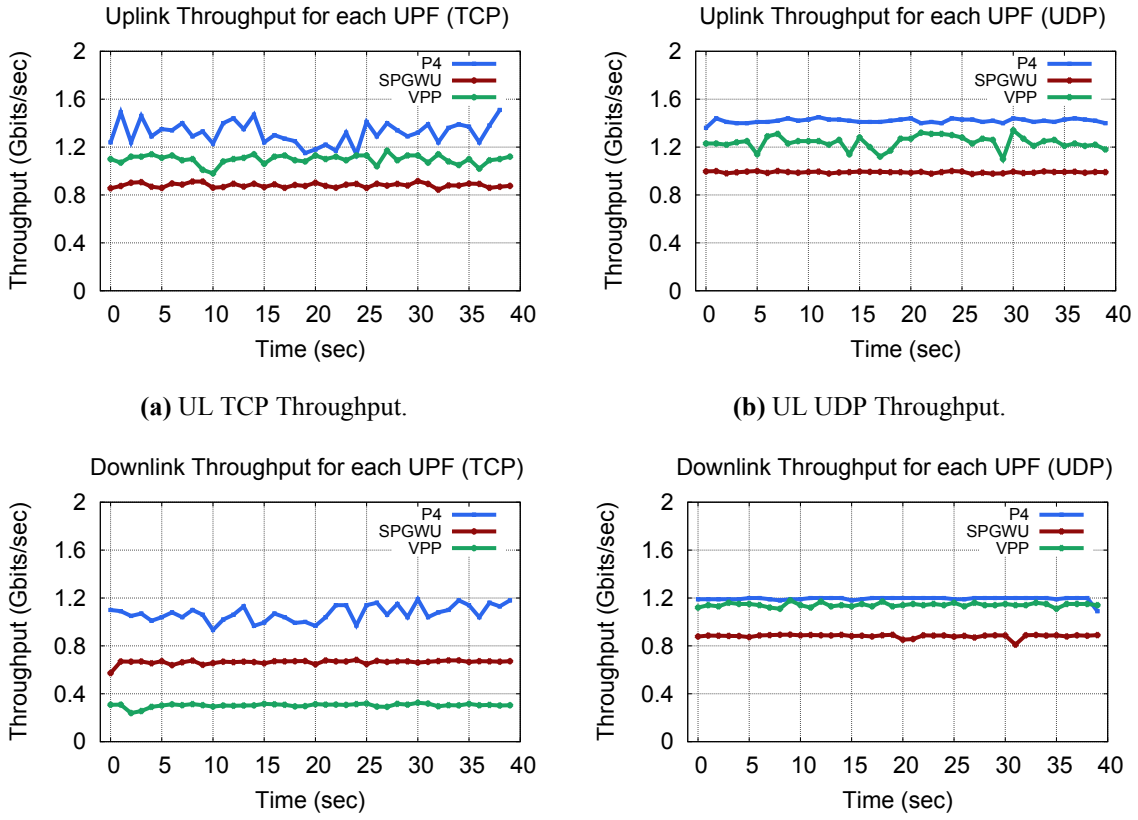
3.4 Experimental Evaluation

To evaluate the performance of the various UPF types, we initially measured the throughput for both Uplink (UL) and Downlink (DL) traffic. Furthermore, we tested the performance of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols, by utilizing a traffic generator tool (iperf) on the side of the UERANSIM and the external-data network and vice-versa. For a better and structured analysis, we divide the experimental findings into two subsections. Subsection 3.4.1 describes the experiments that have been conducted in the real-world servers at our premises at LIP6-Sorbonne University, whereas Subsection 3.4.2 is about our experiments in NITOS Testbed.

3.4.1 SPGWU-UPF, VPP-UPF, P4-SWITCH-UPF (LIP6/CNRS)

Fig. 3.4 illustrates all the throughput measurements for every UPF type. The P4-switch UPF achieved the highest throughput among the other solutions, having almost a 2-times higher throughput than the monolithic SPGWU. However, the VPP-based UPF being a software network function and indirectly a cheaper solution compared to the P4-switch UPF,

reached relatively good performance, surpassing the SPGWU and approaching the performance of the P4-switch UPF. However, in the downlink TCP scenario, the VPP-UPF performed very poorly in comparison with the other conducted experiments.



(a) UL TCP Throughput.

(b) UL UDP Throughput.

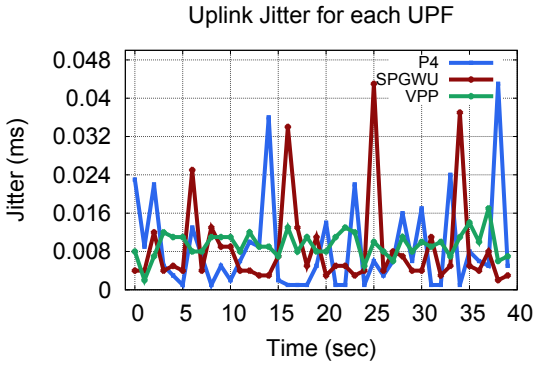
(c) DL TCP Throughput.

(d) DL UDP Throughput.

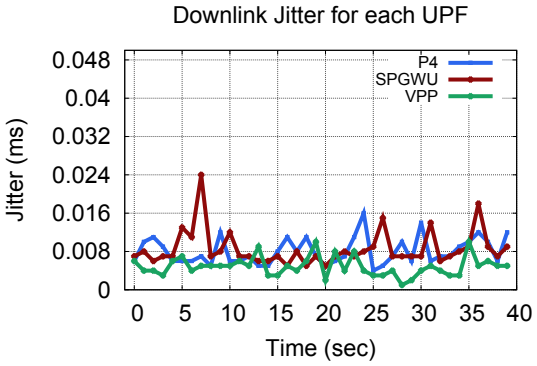
Figure 3.4: Throughput Evaluation: Uplink and Downlink.

Additionally, the VPP-UPF offered the smoothest experience to the end-users in terms of latency. This can be observed in Fig. 3.5, which aggregates all the latency measurements as Jitter in both UL and DL and as the average round-trip time (RTT). In the UL Jitter (Fig. 3.5a) both SPGWU and P4-UPF had high spikes, in contrast with VPP which had more smooth performance. The low-jitter performance was also maintained in the DL jitter (Fig. 3.5b) and this also impacted the RTT (Fig. 3.5c), with the VPP-UPF achieving the lowest average RTT value. Nonetheless, the jitter values of SPGWU and P4-switch UPF are not prohibitive.

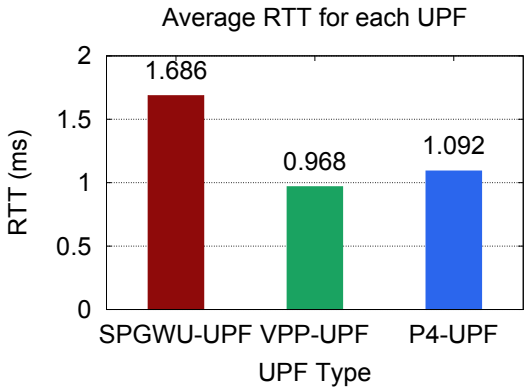
Nevertheless, in terms of packet loss, the SPGWU had the lowest percentage in the UL traffic while the P4-switch UPF had the best performance in DL traffic, achieving only 0.3%. This is illustrated in Fig. 3.6a and 3.6b respectively. It is worth mentioning that all the metrics were observed at the same time. This means that the packet loss is mainly affected by the



(a) UL Jitter.



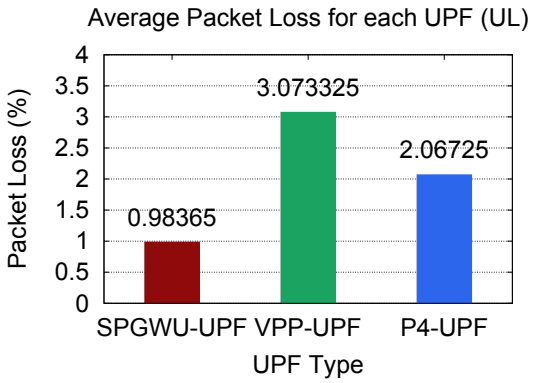
(b) DL Jitter.



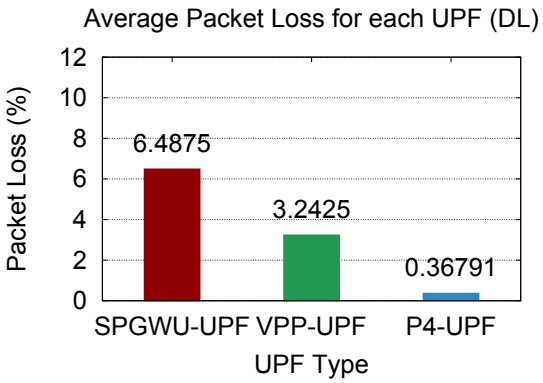
(c) Average RTT.

Figure 3.5: Latency Evaluation: RTT & Jitter.

maximum throughput that we use to stress the links just before they break.



(a) Average UL Packet Loss.



(b) Average DL Packet Loss.

Figure 3.6: Packet Loss Evaluation: Uplink and Downlink.

Finally, we measured the CPU consumption for the network functions only, since the P4-switch UPF offloads the packet processing to the chips. This means the chip itself is handling packet processing at line-rate speeds without involving the CPU. The CPU is typically only

involved during the initial configuration or for control plane tasks. We specifically measured the CPU consumption for the SPGWU and VPP-UPF during UL UDP traffic. The results can be observed in Fig. 3.7. The VPP-UPF used the most CPU cores during the traffic reaching almost 160%. Furthermore, before and after the traffic, it occupied the CPU at 100% while being IDLE. This happens since the DPDK solutions such as VPP, often employ a polling mechanism rather than an interrupt-driven mechanism. This means VPP continuously checks for new packets to process, even if none are available. Polling can lead to higher CPU usage compared to interrupt-driven systems, especially when there's no traffic. In contrast with VPP-UPF, SPGWU had zero CPU consumption when IDLE in terms of traffic, and kept a relatively low CPU consumption percentage of 90% during traffic.

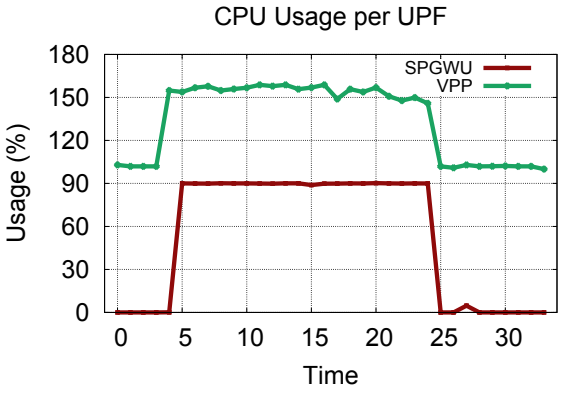
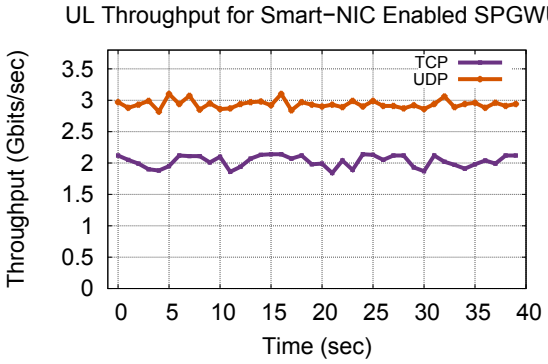


Figure 3.7: CPU Consumption for each UPF.

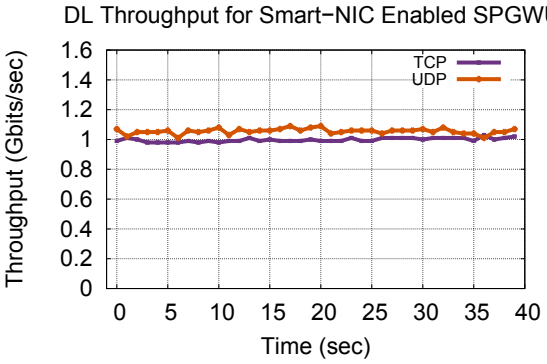
3.4.2 SmartNIC P4-Card UPF (NITOS)

Testing the SmartNIC-P4 UPF gave us the most promising results in terms of Uplink throughput. As depicted in Fig. 3.8a, the uplink throughput exceeded 3Gbps and 2Gbps in UDP and TCP, respectively, outperforming any previous implementation. The downlink throughput (Fig. 3.8b) for both UDP and TCP is slightly lower than the one in the P4-Switch, but still one of the highest. In terms of RTT (Fig. 3.8c) the SmartNIC-P4 UPF showcased the highest value in comparison to the other implementations, while the jitter (Fig. 3.8d) seems to be similar and slightly improved in comparison to the one of the legacy SPGWU.

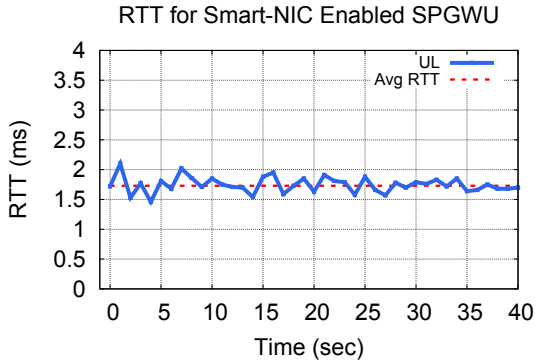
In the context of packet loss (Fig. 3.9a), it had the highest value in UL (exceeding 9%), whereas in DL, it had a relatively low percentage (1.9%). Finally, the CPU allocation (Fig.



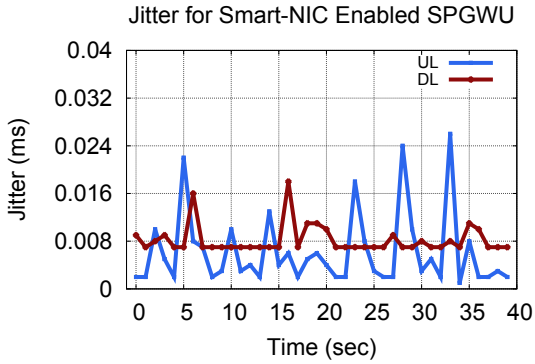
(a) UL Throughput for SmartNIC-P4 UPF



(b) DL Throughput for SmartNIC-P4 UPF



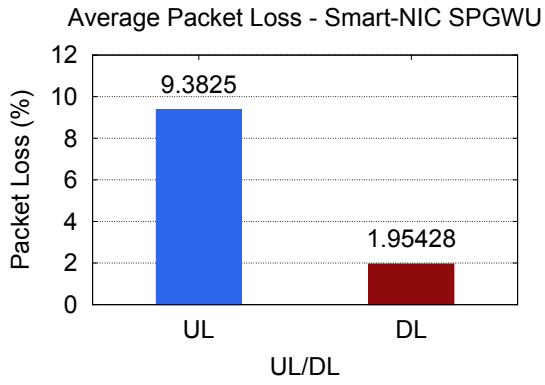
(c) RTT for SmartNIC-P4 UPF



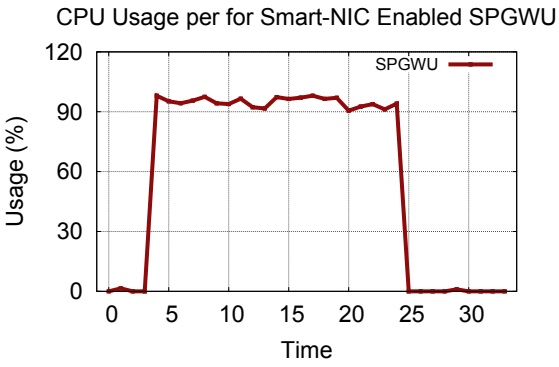
(d) Jitter for SmartNIC-P4 UPF

Figure 3.8: Throughput, RTT & Jitter for SmartNIC-P4 UPF

3.9b) was a bit higher than the one in legacy SPGWU-UPF, but lower than the one of the VPP-UPF.



(a) Average Packet Loss for SmartNIC-P4 UPF



(b) CPU Consumption for SmartNIC-P4 UPF

Figure 3.9: Packet Loss & CPU Consumption

Overall, we can conclude that the P4-switch UPF had an elite performance since it achieved high throughput, generally low latency values, and the lowest DL packet loss percentage.

Additionally to these benefits, it doesn't occupy the CPU since it offloads all the data-plane packet processing to the chips. However, the VPP-UPF is a very promising solution since it's way cheaper than the P4-switch UPF, it has smooth performance on the latency, and also achieves high throughput rates, with the only drawback being the high CPU consumption. Finally, the SmartNIC-P4 UPF reached the highest throughput speeds in the UL scenario, and very good ones in the DL scenario (almost like the p4-switch). The jitter metrics showed a slight improvement in comparison with the default SPGWU, while the RTT and packet loss values were very high. The P4-switch UPF, although running on a 100 Gbps switch, appears to be limited by its UPF implementation rather than by the switch hardware, while the VPP solution is limited by the general-purpose CPU cores available for traffic handling.

3.5 Conclusions & Future Work

In this chapter, we studied and experimentally evaluated various UPF open-source solutions that rely on the legacy SPGWU, on the DPDK technology, and on the P4 language. Our experimental evaluation was under real-world 5G implementation using OpenAirInterface. Our findings revealed that the P4-based (P4-switch & SmartNIC-P4) UPFs generally showcased superior throughput and packet loss performance. This is a direct result of the hardware's ability to perform tasks faster and with less overhead, making it ideal for specialized functions that require rapid execution. However, the VPP-UPF presents a cost-effective alternative with competitive latency metrics. This study underscores the importance of selecting the appropriate UPF implementation while balancing performance, cost, and operational considerations.

As networks continue to evolve, further research and innovations in UPF implementations will be crucial in shaping the future of mobile communications. In the future, we foresee evaluating additional softwarized UPF solutions, with the most promising one being the eBPF/XDP-based UPF. In addition, our current findings may not fully reflect UPF performance on commercial platforms due to differences in network conditions and diverse cloud architectures. To provide a more thorough comprehension, future research will be held to offer a more detailed evaluation across varied deployment scenarios.

Chapter 4

Reactive UPF Scaling in cloud-native 5G networks

4.1 Introduction

As established in Chapter 3, while hardware-accelerated UPF solutions such as P4 switches offer superior raw performance, they are inherently fixed in capacity and cannot be scaled dynamically. Software-based implementations, on the other hand, present a compelling opportunity: by deploying them as cloud-native functions within a Kubernetes cluster, their capacity can be adjusted in real time through horizontal scaling by deploying additional instances under load and releasing them when demand subsides. Building on the cloud-native foundations described in Chapter 2, platforms such as Kubernetes [21], RedHat OpenShift, and OpenStack [42] have served as the basis for purpose-built network management frameworks such as ONF AETHER, ONF CORD, and Google Nephio [43], aimed exclusively at the orchestration of core network functions. These tools further unlock capabilities such as live migration and multi-datacenter scheduling, and enable indirect management of broader network aspects such as edge UPF placement for MEC and network slicing.

In this chapter, we build upon this cloud native approach for the 5G Core Network, and design, implement and evaluate a scheme for the dynamic scaling of the UPF functions at the edge of the network. We employ an experimentally driven approach, deploying the OpenAir-Interface platform as a cloud-native 5G Core Network and UERANSIM to emulate the gNB and the UEs. The scaling process is managed through metrics collected in real-time from the network operation, which reflect the load and energy efficiency aspects of the network. Our

contributions are summarized as follows:

- **Efficient Resource Management:** Leveraging Kubernetes and cloud-native deployment of the 5G core enables dynamic resource allocation, leading to improved utilization of UPF resources.
- **Adaptive Congestion Mitigation:** Scaling the UPF based on runtime metrics allows the system to dynamically handle congestion and maintain stable performance levels.
- **Scalability and Flexibility:** The proposed approach supports scalable and adaptable deployments, allowing the network to efficiently respond to varying traffic conditions.
- **Realistic Experimental Validation:** Evaluating the system under real network conditions ensures the practical relevance and reliability of the obtained results.

The rest of this Chapter is structured as follows. In Section 4.2, details are given on research and studies related to this work. In sections 4.3 and 4.4, we provide details on the system architecture and our experimental findings. Finally, in Section 4.5, we conclude and discuss some possible future work.

4.2 Related Work

With the advent of cloud-native computing and networking, several works have focused on building functionality on top of new features that frameworks around this ecosystem can provide. One of such features is the support of auto-scaling, as a means to accommodate the demand for the services. Scaling is broken down into three different approaches: 1) Horizontal Scaling, where more replicas of the service are spawned, and traffic is balanced between them, 2) Vertical Scaling, where more/fewer resources are applied to the running service to meet the demand (adding/removing CPU cores/memory), and 3) Cluster Auto-scaling, where the cluster hosting the services offers more/less resources. Several of these features are implemented in well-established frameworks used for cloud-native research and industrial environments. The most notable ones that support all three types of scaling are Kubernetes (K8s), ETSI Open Source MANO [44], and Hashicorp Nomad [45]. Different works exploit such functionality towards defining new algorithms/policies, as well as services that can auto-scale to meet the service demand. In [46], authors investigate how the Mobility Management

Entity shall auto-scale in 4G networks, towards meeting high swarms of users entering the cell. Authors in [47] provide a vendor-agnostic system architecture that enables the elasticity of cloud-native services utilizing automated scaling and resource provisioning.

As such decisions rely on metrics collected directly from the platforms hosting the services, they can be easily annotated with timestamps and in turn, create a time-series of metrics. Such time-series can be fed to Neural Networks, to predict their future values, and hence proactively apply decisions for the network. Authors in [48], [49], [50] employ such an approach, with differentiating factors the models used to train and predict the future values. In [51] and [52], authors employ a similar approach for auto-scaling the Access and Mobility Management Function (AMF) entity of the 5G network. An analysis of the data, training models, accuracy of predictions, and benefits of scaling are provided.

Nevertheless, the majority of such works only address the control plane part of the telecommunications network (5G AMF/4G LTE) or apply to stateless services. In this work, we progress beyond existing state-of-the-art and propose the scaling process for the 5G User Plane Function, that is in charge of breaking out the traffic to Data Networks. Works [53] and [54] suggest a proactive solution, which relies on deploying UPFs utilizing machine learning techniques. Our work focuses on deploying UPFs dynamically and only when necessary, by observing real-time metrics. Authors in [55] suggest a well-structured way of scaling VPP-based UPFs based on VPP rate, which represents the number of packets that can be processed simultaneously. In [56], authors explore the benefits that are brought by the reconfiguration of the UPF component in relationship to the different connections with the rest of the 5G network entities. The authors of [28] apply deep reinforcement learning to scale UPF instances, while [57] analyzes threshold-based UPF scaling through a queueing model. In addition, in [58], the authors suggest a 4G/LTE-based model that scales Data Plane resources based on Bit rate Aware Auto Scaling (BAAS). In this work, we strictly follow an experimentally driven approach (beyond simulations) applied in real networks, to validate the architecture and quantify the benefits of the proposed approach. The custom-built auto-scaler mitigates the high triggering times of such tools.

4.3 System Architecture

In this section, we will provide a detailed overview of the key components and the system configuration. To start with, we employ a Kubernetes cluster consisting of one master node and 2 worker nodes, which helps us manage our 5G network infrastructure easily, ensuring scalability and efficient resource management. For the nodes to communicate seamlessly in our cluster, we deploy Flannel overlay management network. As noted, the 5G core network components operate as independent functions, benefiting from SBA's capacity to allow operation in a fully distributed manner. Having this ability, Kubernetes leverages Docker containers to deploy our 5G Network in a micro-services environment. Multus CNI is also deployed to increase network versatility by enabling multiple network interfaces within a single pod.

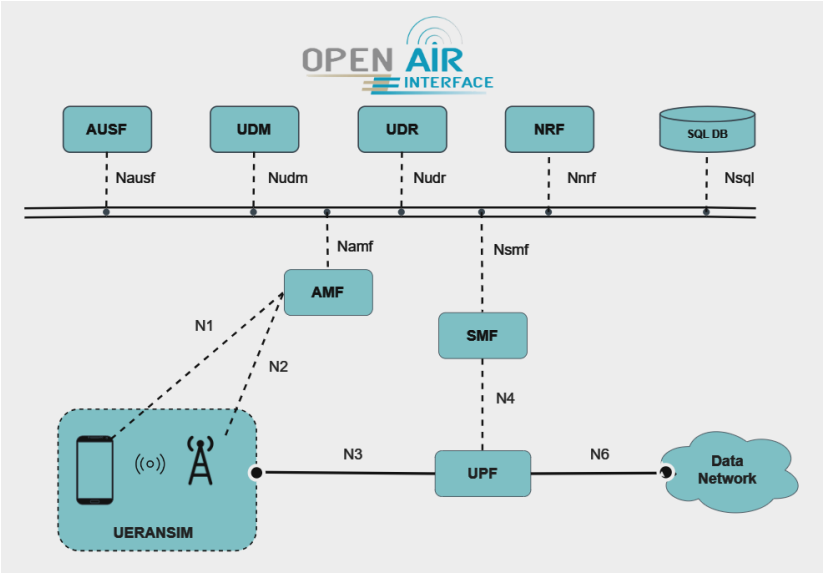
It is worth mentioning that our system operates within the real experimental infrastructure provided by NITOS, which provides a controlled and reliable environment for testing our 5G Network, confirming that our work is applicable to real-world scenarios and environments. The detailed specifications of each cluster node can be retrieved from Tables 4.1,4.2.

Table 4.1: Master Specifications

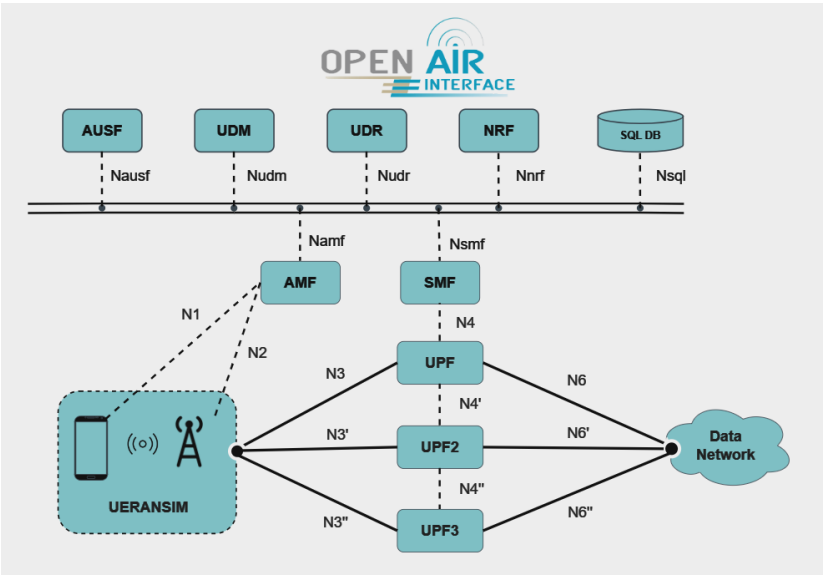
Node / Specs	**Master Node**
CPU	Intel(R) Core(TM) i7-8700 CPU
Cores	12
RAM	16GB
Operating System	Ubuntu 18.04.6 LTS
Docker Version	19.03.6
Kubernetes version	v1.20.0
Tasks Executed	5G-Core Functions, UERANSIM, Prometheus/Grafana

As mentioned earlier, our 5G Network deployment lies in the OpenAirInterface platform and in that matter, we deploy the necessary 5G core components required for this work, including the NRF, UDR, UDM, AUSF, AMF, SMF and the UPF. For the Radio Access Network part, we deploy UERANSIM in order to simulate the behavior of 5G RAN, giving us the ability to perform advanced 5G experiments. When the UE successfully connects to the core network, a PDU session is created between the UE and the UPF, allowing data traffic. This configuration can be observed in Figure 4.1a and depicts a monolithic 5G network architecture. In our work, we extend this setup (Figure 4.1b) by incorporating multiple UPFs,

which are responsible for managing the user data traffic. More specifically the 5G model that we deploy is capable of supporting multiple UPFs, while keeping the number of the other network functions the same (1 AMF, 1 SMF etc.). Each UPF is associated with a unique Data Network Name (DNN), which plays a vital role, as it allows us to astutely connect the UEs to the specific UPF that matches the conditions it necessitates. In a state where a UPF is experiencing high load, nearing its capacity, and the link is at the brink of degradation, we utilize the DNNs to reroute User Equipment (UE) to a different UPF that has the capacity and resources to maintain the link's stability and prevent a potential network breakdown.



(a) Monolithic Architecture



(b) Dynamic Algorithm Architecture

Figure 4.1: Monolithic vs Dynamic Algorithm Architecture

Table 4.2: Worker1 & Worker2 Specifications

Node / Specs	Worker1 & Worker 2
CPU	Intel Core Processor (Skylake, IBRS)
Cores	4
RAM	8GB
Operating System	Ubuntu 18.04.6 LTS
Docker Version	19.03.6
Kubernetes Version	v1.20.0
Tasks Executed	UPF2 , UPF3, Prometheus/Grafana

In addition, we employ Prometheus [59] and Grafana [60] to efficiently monitor and visualise our cluster's performance. Prometheus acts as data tracker, collecting metrics (CPU consumption, packet reception etc.) through NodeExported from the Kubernetes cluster, to provide end-to-end monitoring. These metrics are then passed to Grafana, which serves as a visualization and data extraction tool. Grafana generates sophisticated graphs and dashboards while giving the ability to store and process the collected data for thorough evaluation.

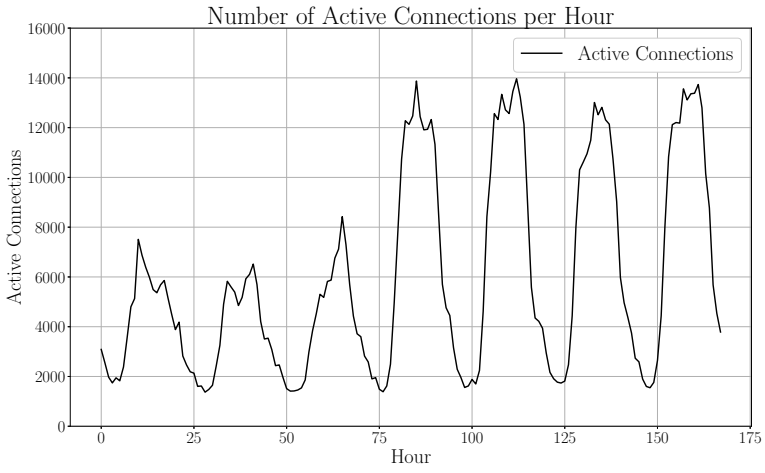


Figure 4.2: Data representation: Active Connections per Hour

In order to emulate a real-world scenario, we have used an online available dataset [61]. The datasets provided, contain information about telecommunication activities in Milan and the Province of Trentino. Due to the extensive size of the dataset, we performed an extraction, obtaining a subset of 170 data points that describe the active connections per hour (Figure 4.2). This subset contains a wide range of the amount of connections per hour, spanning from

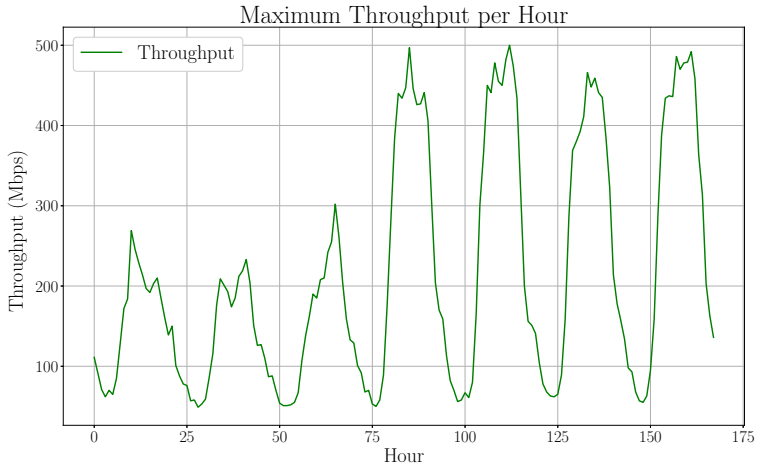


Figure 4.3: Data representation: Throughput per Hour

low to high values, in order to offer a thorough representation of real world scenarios. To be able to utilize the data in our experiments, we converted it to throughput data. We assume that the number of connections correlate to the load and the usage of the network. To elaborate further, we spotted the highest value in the data that reflects the maximum number of active connections and mapped that value to the maximum Mbps (i.e. 500Mbps) that our system can handle before the PDU session (tunnel) drops. Then, we scaled the rest of the values with respect to the highest value. The produced dataset used for the experiments represents the total throughput generated per hour and can be seen in Figure 4.3.

As noted earlier, the main idea of this work is the design and implementation of a dynamic auto-scaling algorithm. The algorithm starts by deploying the monolithic 5G architecture within the cluster, initially featuring a single UPF. Subsequently, it reads a sequence of throughput values from the dataset and initiates traffic using iperf for 20 seconds per value. Each throughput value is split evenly across the UEs, which direct their traffic toward the data network through the active UPF. The algorithm uses a dedicated monitoring thread to continuously observe the CPU consumption of all active UPF instances. While all instances are monitored for observability and cluster health purposes, scaling decisions are driven by the CPU of the primary UPF instance, which consistently reflects the most stressed point in the deployment and serves as the main scaling signal. Two predefined thresholds are enforced, $LimitA$ (67%) and $LimitB$ (87%), chosen empirically to represent moderate and high load conditions respectively, providing sufficient headroom for a new instance to become operational before the primary instance reaches saturation. Scale-out decisions are governed as

Algorithm 1 Dynamic UPF Scaling Algorithm

Initialize:
- Deploy a single UPF (primary) and set $LimitA = 67\%$, $LimitB = 87\%$
- Initialize variables for tracking active UPFs and UE assignments

function MonitorCPU
  Initialize a thread for continuous CPU monitoring
  while Algorithm is running **do**
    - Measure CPU consumption of all active UPF instances
    - Track $CPU_{primary}$ of the primary UPF instance
    Sleep(short duration)
  end while
end function

MonitorCPU

repeat
  - Read next throughput value from dataset and generate traffic via iperf
  - $CPU_{primary} \leftarrow$ CPU of the primary UPF instance
  if active UPFs $= 1$ **and** $CPU_{primary} > LimitB$ **then**
    - Deploy 2 new UPFs on available nodes
    - Redirect a portion of UEs to new UPFs to reduce primary load
  else if active UPFs $= 1$ **and** $CPU_{primary} > LimitA$ **then**
    - Deploy 1 new UPF on an available node
    - Redirect a portion of UEs to the new UPF to reduce primary load
  else if active UPFs $= 2$ **and** $CPU_{primary} >$ either threshold **then**
    - Deploy 1 new UPF on an available node (maximum of 3 total)
    - Redirect a portion of UEs to the new UPF to reduce primary load
  end if
  if $CPU_{primary} < LimitA$ **and** active UPFs > 1 **then**
    - Reassign all UEs to the original UPF
    - Terminate all additional UPF instances
  else if $LimitA \leq CPU_{primary} < LimitB$ **and** active UPFs $= 3$ **then**
    - Redirect UEs to reduce load across remaining two UPFs
    - Terminate the third UPF instance
  end if
  if active UPFs $= 3$ **and** aggregate demand exceeds combined capacity **then**
    - Maintain maximum deployment
    - Continue traffic redistribution to minimize per-instance load
  end if
until End of dataset

follows. If the CPU of the primary UPF exceeds $LimitA$ and only one UPF is running, the algorithm deploys a second UPF on an available worker node and redirects a portion of the UEs to it. If CPU consumption exceeds $LimitB$ and only one UPF is active, the algorithm directly deploys two additional instances, redistributing UEs across all three to reduce load on the primary instance. If two UPFs are already active and the primary instance rises above either threshold, one additional instance is deployed to reach the maximum of three, with UEs

redistributed to balance the load. Each newly deployed UPF uses a unique DNN, enabling on-demand UE reassignment. The maximum number of active UPFs is bounded by the number of available nodes, which in this setup is three. Scale-in decisions are triggered when the CPU of the primary UPF falls below the active threshold. If it falls below $LimitA$ and more than one UPF is running, all UEs are reassigned to the original UPF and the additional instances are terminated, regardless of how many instances are currently active. If it falls between $LimitA$ and $LimitB$ and three UPFs are active, UEs are redistributed across two instances and the third is terminated. Since scaling decisions are driven exclusively by the primary UPF, saturation of secondary instances is prevented in practice by the traffic redistribution mechanism, which ensures that each deployed instance handles at most a proportional share of the total demand. In the event that the aggregate demand exceeds the combined capacity of all three instances, the system operates at maximum deployment and relies on the traffic distribution to minimize per-instance load, with any residual congestion accepted as an operational boundary of this proof-of-concept implementation. It is important to note that while some packet loss may occur during the reassignment process, it is significantly lower than the loss experienced when the primary UPF's CPU approaches saturation. The high-level concept of the algorithm is depicted in Algorithm 1.

4.4 Experimental Findings

This section delves into the experimental part of our research, where we bridge the gap between theory and practice. It is worth mentioning that each experiment was conducted multiple times to ensure the consistency of the algorithm. The experiments are held under two categories: the monolithic (default) architecture and the architecture enhanced by the dynamic scaling algorithm. Within each category, we focus on two scenarios, one involving the UDP and the other using the TCP. Employing both UDP and TCP protocols in our experiments ensures a comprehensive evaluation of our architecture. It allows us to explore how our architectural decisions adapt to the unique characteristics of each protocol and perform a comparative analysis. In the first series of experiments, we examine the monolithic architecture. We run iperf commands for all the throughput values from our dataset, for both UDP and TCP scenarios independently, where each iperf command lasts 20 seconds. Subsequently, we perform the exact same sequence of experiments, while also deploying the

dynamic algorithm, in order to validate its ability to meet the evolving demands in contrast with the monolithic scenario.

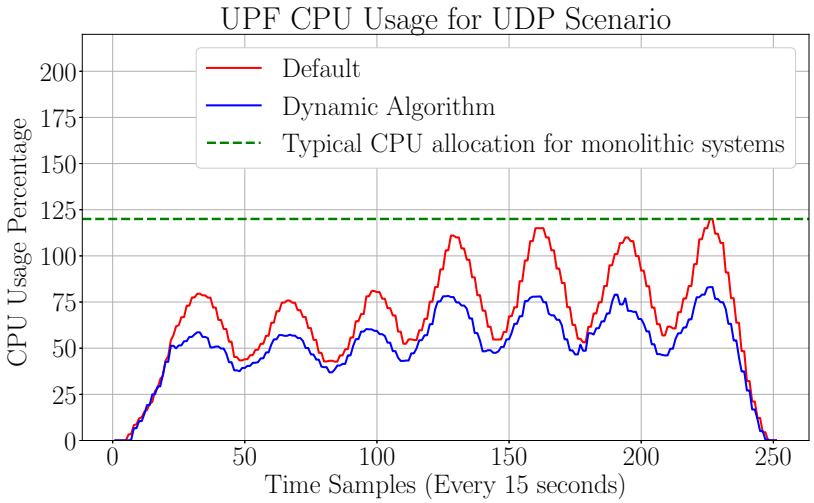
After conducting our experiments, we proceeded to extract and analyze data, generating graphs using Prometheus and Grafana. Figures 4.4a and 4.4b illustrate a comparative analysis of CPU allocation in the monolithic and dynamic architectures for both UDP and TCP scenarios. Our observations reveal that, in both protocol scenarios, the CPU usage in the monolithic architecture occasionally exceeds 100%, signifying intensive resource utilization across multiple CPU cores. Specifically, in the TCP scenario, CPU usage spikes to 140%, while in the UDP scenario, it reaches 120%. Notably, TCP exhibits higher peaks in CPU allocation, due to the fact that when the link (i.e. PDU session) and the UPF CPU are stressed, it triggers re-transmissions, resulting to additional network load. In contrast, when our scaling algorithm operates, we observe (Figures 4.4a & 4.4b) major differences in CPU allocation.

Peak CPU usage reaches around 83% for UDP and approximately 74% for TCP. These numbers represent a marked improvement compared to the monolithic approach, where CPU usage exceeds 100% in high throughput values. In the dynamic setup, CPU usage stays within manageable levels, preventing resource congestion and performance issues. This means that the dynamic scaling achieve better CPU allocation, resulting in a smoother and more reliable network operation.

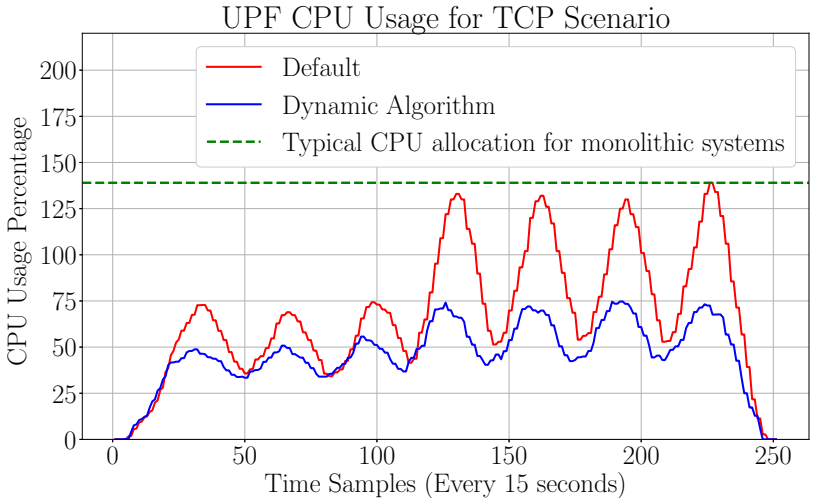
Figure 4.5 & Figure 4.6 present a visual comparison of the UPF packet reception rate (within a 5-minute window) for both the monolithic and dynamic architectures in the context of UDP and TCP scenarios. In order to retrieve this metric, we utilized the Prometheus container-network-receive-packets-total query and specified the interface value so it can track packets that come specifically from the PDU session (i.e. tun0 interface).

The generated graphs reveal substantial differences in packet reception rates. The points where the extra UPFs are deployed can be easily identified by examining the respective UPF lines. In the monolithic architecture, the packet reception rate reaches 45,000 and approximately 34,000 packets for UDP and TCP, respectively.

In the UDP scenario, the rate of received packets exceeds that of the TCP scenario. This difference can be explained by UDP's inherent characteristics, which include lower overhead and fewer data reliability measures. This makes UDP transmit packets at a higher rate compared to TCP, which ensures controlled and reliable data delivery. This highlights how the protocol directly affects the rate of packet reception, giving insights into the trade-offs be-



(a) Monolithic vs. Algorithm for the UDP scenario

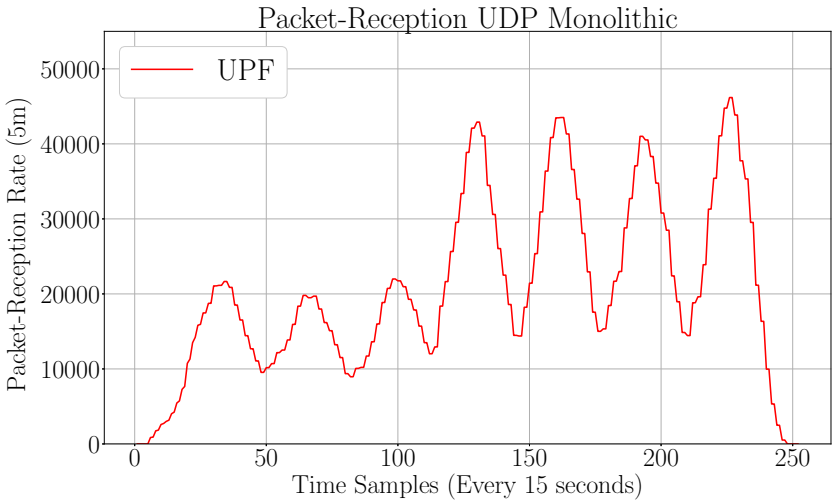


(b) Monolithic vs. Algorithm for the TCP scenario

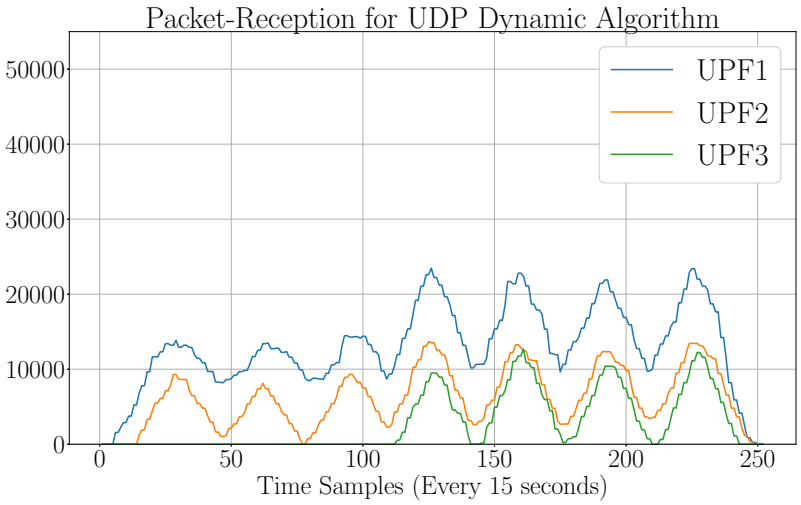
Figure 4.4: UPF CPU Allocation Comparison for UDP and TCP

tween speed and reliability in network communication. However, when the dynamic mechanism operates, we observe notable differences in peak packet reception rates. In this scenario, maximum peak values reach approximately 22,500 packets for UDP and around 18,500 for TCP. In both UDP and TCP scenarios, during the initial phase, represented by the first three peaks, packets are efficiently forwarded to the two active UPFs. However, as network demands maximize (in the final four peaks), we observe that three active UPFs are deployed to manage the increased traffic load, dividing the network traffic.

Overall, the initial approach of directing all traffic through a single UPF, pushing it to the brink of overload, underscored the limitations of a monolithic system. However, the dynamic



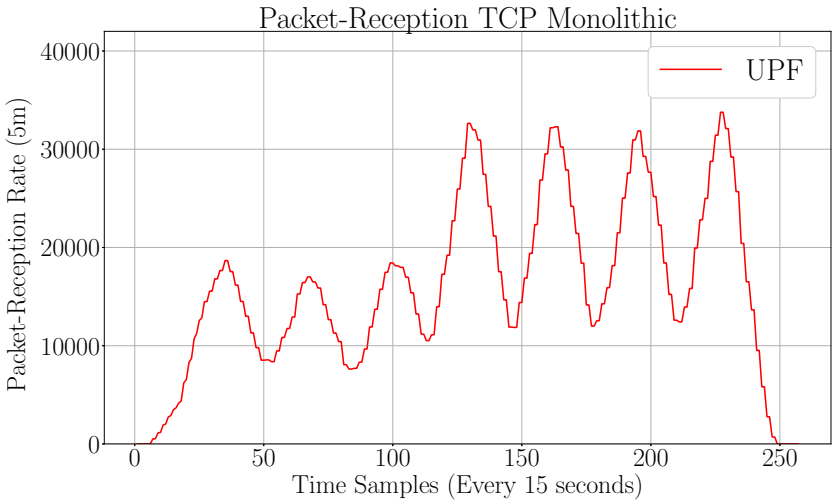
(a) Monolithic Packet Reception for the UDP scenario



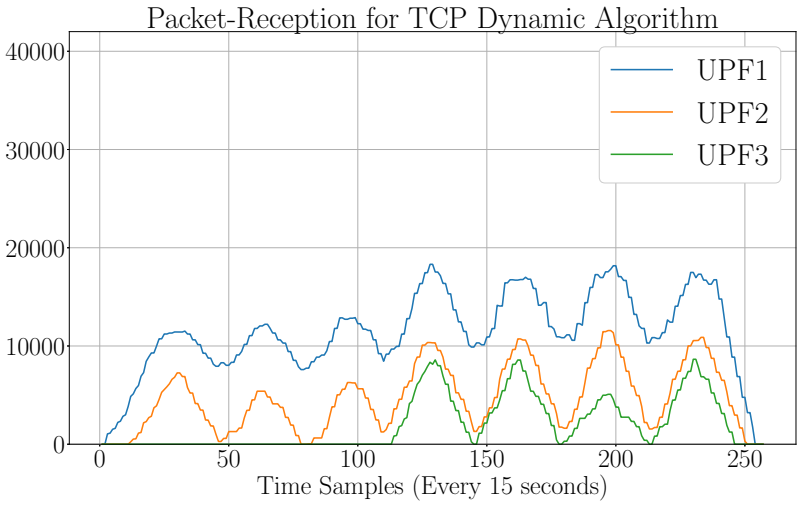
(b) Dynamic Algorithm Packet Reception for the UDP scenario

Figure 4.5: UPF Packet Reception for the UDP scenario

scaling algorithm's ability to evenly distribute traffic, while keeping the link uncongested, suggests a significant improvement. By efficiently managing resources and deploying additional UPFs as needed, the dynamic mechanism prevented network congestion and ensured reliable performance. Furthermore, the observed CPU allocation emphasizes the advantages of dynamic scaling further, showcasing its ability to maintain optimal performance, while efficiently managing CPU resources. Projecting the CPU consumption directly to the energy consumption of the platform, it is evident that lower CPU utilization leads to higher energy efficiency. Arguably, when the CPU allocation reaches full capacity, high temperature and bad scheduling can occur, leading to higher overall energy consumption compared to distributing



(a) Monolithic Packet Reception for the TCP scenario



(b) Dynamic Algorithm Packet Reception for the TCP scenario

Figure 4.6: UPF Packet Reception for the TCP scenario

the workload to other CPUs. For the cases of monolithic deployment, resources need to be over-provisioned during their initial deployment to meet the demand (provisioning for the peak of the demand). Therefore, an automatic scaling process can provide clear benefits for the case of the overall energy efficiency, allowing provisioning of lower-utilized UPF deployments only when needed. It's worth noting that the algorithm can be easily deployed and employed across various machines, as it doesn't necessitate the use of advanced tools. In addition, it can be adjusted to support more than 3 UPFs; however, mindlessly deploying new UPFs can lead to wasted resources, as our goal is to reduce CPU usage to levels that don't affect network performance, not zero it.

4.5 Conclusion & Future Work

In this work, we have deployed a 5G network in a real-world micro-services environment with the use of Kubernetes and NITOS testbed. Subsequently, we have designed and implemented a dynamic algorithm, as opposed to the traditional monolithic approach, to incorporate additional UPF functions based on real-time metrics, that reflect the network conditions. Extensive experiments using a real-world telecommunications dataset have been employed to evaluate our approach. By analyzing the generated data and graphs, we have seen obvious improvements in the CPU allocation percentage, as well as in the network congestion part. All in all, the dynamic mechanism demonstrated its efficiency, by optimizing resource management, reducing costs, and enhancing network reliability.

Despite these results, this work should be viewed as a proof-of-concept demonstration of dynamic UPF scaling in a real Kubernetes deployment rather than a production-ready solution, and several limitations are worth acknowledging. First, the reactive nature of the mechanism means that scaling decisions are triggered by observed degradation rather than anticipated demand, leaving the system vulnerable to congestion under bursty traffic where the time to instantiate a new UPF may be insufficient to prevent service degradation. Second, CPU utilization is used as the sole scaling signal, which is a proxy metric that does not directly reflect service quality and is particularly unsuitable for DPDK-based UPF implementations such as VPP, which maintain high CPU occupancy regardless of traffic load. Third, the threshold values of $LimitA$ and $LimitB$ are empirically chosen and fixed, with no sensitivity analysis or adaptive mechanism to adjust them based on changing network conditions or UPF implementation type. Finally, the evaluation is conducted with a maximum of three UPF instances bounded by the available worker nodes, and the behavior of the algorithm under larger deployments remains unexplored. Addressing these limitations defines the research directions pursued in Chapter 5, where a proactive, energy-aware, and heterogeneous orchestration framework replaces reactive threshold-based decisions with ML-driven demand forecasting, moving from a proof-of-concept toward a principled and deployable 6G core network management system.

Chapter 5

Proactive and Energy-Aware Orchestration of Heterogeneous UPFs

5.1 Introduction

Chapter 4 demonstrated that horizontal scaling of software-based UPF instances within a Kubernetes cluster is an effective mechanism for adapting core network capacity to changing traffic demand. However, that work operated under two simplifying assumptions that limit its applicability in real deployments. First, it treated all UPF instances as homogeneous and performed scaling of a single implementation type without considering that different implementations carry fundamentally different performance and energy profiles. Second, its scaling decisions were reactive: triggered by observed degradation rather than anticipated demand, leaving the system vulnerable to transient congestion during the interval between detection and remediation.

These limitations point to a deeper opportunity. As established through the empirical profiling conducted in Chapter 3, SPGWU and VPP-based UPFs exhibit fundamentally different resource and energy profiles: SPGWU consumes modest CPU and power under low-to-moderate load, making it an efficient choice when traffic demand is not very demanding, while VPP sustains significantly higher throughput by leveraging DPDK-based vector packet processing, at the cost of consistently elevated energy consumption regardless of load. Neither implementation alone is optimal across all traffic conditions as SPGWU saturates under high demand, while VPP wastes energy during low-demand periods. A framework that exploits this complementarity, dynamically selecting not only *how many* instances to deploy

but also *which type* at each decision interval, and doing so proactively based on predicted future demand rather than observed degradation, can simultaneously satisfy throughput requirements and minimize energy consumption; two objectives that static deployments and the reactive scaling approach of Chapter 4 cannot jointly achieve.

To address these limitations, this chapter proposes a proactive and energy-aware UPF selection and scaling framework that explicitly exploits the complementary performance characteristics of SPGWU- and VPP-based UPFs. The framework integrates a CNN-LSTM-based traffic prediction model to anticipate future throughput demand over a 3-step-ahead horizon, enabling proactive reconfiguration decisions before network congestion occurs. Based on UPF-specific polynomial models of CPU utilization and power consumption derived from empirical profiling, the framework selects at each decision interval the number and type of UPF instances to deploy across a Kubernetes cluster, minimizing total energy consumption while satisfying predicted traffic demand within the available node budget.

The framework is implemented and evaluated on SLICES-RI, where all 5G core and RAN components are containerized and orchestrated through Kubernetes across a multi-node cluster using OpenAirInterface and UERANSIM. We conduct a systematic experimental comparison across five deployment scenarios under identical dynamic traffic conditions. Results show that the proposed framework is the only configuration that successfully satisfies all throughput demands, achieving an average packet loss of only 1.5%, while reducing total energy consumption by 51% compared to the static all-VPP deployment.

The main contributions of this chapter are summarized as follows:

- **Empirical Profiling and Modeling:** We provide comprehensive performance and energy profiles for two representative User Plane Function implementations, SPGWU and VPP. From these, we derive polynomial CPU utilization and linear power consumption models that accurately capture their distinct non-linear behaviors under real traffic.

- **Predictive Traffic Modeling:** We propose a CNN-LSTM-based traffic prediction model with a 3-step-ahead horizon. Through systematic benchmarking against LSTM, GRU, and XGBoost baselines, our model achieves a Mean Absolute Error (MAE) of 77.95 Mbps and a Mean Absolute Percentage Error (MAPE) of 5.74% on a real-world 5G production dataset.

- **Energy-Aware Deployment Optimization:** We formulate an optimization mechanism that dynamically determines the optimal number and type of UPF instances. By exploiting the complementary energy-performance trade-offs of these two implementations, the framework is able to minimize energy consumption while meeting the predicted demand at the same time.

- **Experimental Validation:** In order to make sure that our proposed framework is tested under real-world conditions we evaluate it on a Kubernetes-based real-world 5G testbed. Results demonstrate 51% energy savings compared to static all-VPP deployments with an average packet loss of 1.5%, validating the framework's ability for sustainable 5G/6G core operations.

The remainder of this chapter is organized as follows. Section 5.2 focuses on related work relative with energy efficiency in softwarized 5G/6G networks, dynamic core function placement, ML-based proactive orchestration, and UPF implementation heterogeneity. Section 5.3 describes the full system architecture and provides all the insights of the functionalities we built. Section 5.4 presents the experimental setup and evaluation results and finally Section 5.5 concludes the chapter and outlines directions for future work.

5.2 Related Work

5.2.1 Energy Efficiency in Softwarized 5G/6G Core Networks

As already discussed in Chapter 1, energy efficiency has emerged as a defining challenge in 5G and beyond-5G softwarized networks, and is increasingly recognized as a primary design objective for 6G systems. The ITU identifies energy efficiency as one of the key performance indicators for IMT-2030, targeting a substantial improvement in energy efficiency per bit compared to IMT-2020 [16]. In softwarized networks, where core functions are implemented as cloud-native services on general-purpose hardware, the virtualization of network functions introduces significant computational overhead, leading to higher power consumption compared to dedicated hardware solutions and making energy-aware orchestration an urgent practical concern.

Several works have addressed infrastructure-level energy optimization in this context. Al-Quzweeni *et al.* formulate a mixed-integer linear programming model to minimize total

power consumption of VNFs in 5G cores through optimal VM and server placement, demonstrating significant energy savings over static deployments [62]. Moosavi *et al.* investigate joint routing and VNF placement strategies in SDN/NFV-enabled networks, showing improved energy efficiency under hybrid cloud deployments [63]. Fernández-Fernández *et al.* explore energy-aware routing in SDN-based 5G systems, with special focus on trade-offs between reduced power usage and performance metrics like latency and link utilization [64]. Broader studies on this topic further underline that energy-aware orchestration, traffic-aware control, and dynamic resource scaling are key enablers for sustainable softwarized core networks [65, 66].

However, there is a major deficiency in the above achievements. The proposed energy-aware schemes work either at the infrastructure level or at the VNF abstraction level, assuming energy homogeneity of all the network services. This means that the complex non-linear relation of traffic loads, CPU usage, and power consumption on the individual UPFs is overlooked by all these schemes, which turns out to be crucial in our research.

5.2.2 Dynamic Placement and Scaling of Core Functions

The dynamic orchestration techniques seek to match the provisioning of network function deployments to the traffic requirements. In this regard, Taleb *et al.* propose the use of the "Follow Me Cloud" where the network functions would be relocated nearer to the mobile user to decrease latency and support mobility [67]. On the other hand, Ren *et al.* have shown how the VNFs placement and routing problems can be solved through joint optimization which performs better compared to independent approaches [68]. In Kubernetes, while horizontal pod autoscaler offers a solution, the limitation lies on the linear threshold model based on the static CPU utilization [69]. At the 3GPP standardization level, Uplink Classifier (ULCL) and branching point mechanisms defined in TS 23.501 enable multi-UPF traffic steering, but their configuration remains largely static and does not incorporate energy awareness [70]. Moreover, reinforcement learning techniques have been used to solve network slicing and autoscaling problems considering latency, energy consumption, and instantiation cost [71]. Threshold-based UPF autoscaling is also considered using queueing-theoretic models to understand its stability, blocking probability, and delay [57].

The key limitation shared by all these approaches is their reactive nature — scaling decisions are triggered by observed conditions rather than anticipated demand. This leaves the

system perpetually one step behind traffic fluctuations, incurring either network violations during scale-up or energy waste during over-provisioned idle periods. Furthermore, none of these works account for the heterogeneous energy-performance characteristics of different UPF implementations when making deployment decisions.

5.2.3 Machine Learning for Proactive Network Orchestration

Machine learning has increasingly been leveraged for proactive network management, enabling forecasting and predictive resource allocation ahead of traffic fluctuations. Alawe *et al.* demonstrate that LSTM networks can accurately predict 5G core load demands, improving orchestration responsiveness compared to reactive threshold mechanisms [72]. Similar frameworks have been proposed for slice-aware resource orchestration, where predictive models guide dynamic allocation of physical resources under heterogeneous service demands [73]. ITU-T Recommendation Y.3172 formalizes how ML pipelines integrate with network management systems to enable automated, data-driven decisions in IMT-2020 and beyond-5G networks. Passas *et al.* explore deep learning-based autoscaling in cloud-native 5G cores, showing improved load balancing and admission control on a real Kubernetes testbed [52]. Nguyen *et al.* apply deep reinforcement learning to UPF scaling as a sequential decision problem, dynamically adjusting instance count to match predicted traffic states [28].

The common limitation across these works is their treatment of UPFs as homogeneous entities — scaling decisions adjust the *number* of instances but not the *type*. As a result, the energy-performance heterogeneity of different UPF implementations is entirely ignored, preventing the system from exploiting the complementary characteristics of available implementations to minimize energy consumption while satisfying performance requirements.

5.2.4 UPF Implementation Heterogeneity: Performance & Energy Trade-offs

The User Plane Function is central to 5G core performance and energy efficiency, yet the implications of UPF implementation heterogeneity for orchestration decisions remain largely underexplored. Software-based UPFs differ significantly in architecture and resource demands. VPP-based UPFs leverage user-space packet processing and vectorization to maximize throughput, while SPGWU-based UPFs offer lower complexity and reduced overhead

at moderate traffic loads [1]. P4-switch and SmartNIC-based UPFs provide hardware acceleration but at the cost of reduced deployment flexibility. Empirical evaluations reveal distinct CPU utilization and throughput profiles across UPF types, which directly impact energy consumption and scaling decisions [34, 39, 74]. Rischke *et al.* compare UPF acceleration technologies under URLLC traffic conditions, demonstrating that tail latency and energy behavior differ substantially across implementations [39]. In Chapter 3, we conducted a direct evaluation of multiple UPF implementations in a real-world 5G network deployment, confirming that identical scaling policies applied to different UPF types yield substantially different energy costs and throughput outcomes.These benchmarking results establish a clear imperative: orchestration frameworks must explicitly incorporate UPF-implementation-specific performance and energy models when selecting both the number and type of instances to deploy. However, existing benchmarking studies stop short of integrating these models into a live orchestration system — leaving a direct bridge between empirical UPF evaluation and real-time deployment decisions as an open research problem.

5.2.5 Positioning of the Proposed Framework

The survey above reveals that existing work falls into three categories, each addressing only part of the problem. Infrastructure-level energy optimization works [62, 63, 64] minimize power consumption at the server or VNF abstraction level but ignore the heterogeneous energy behavior of specific UPF implementations. ML-based proactive orchestration works [28, 52, 72] enable predictive scaling but treat UPFs as homogeneous entities, adjusting only instance count without considering implementation type or energy implications. UPF benchmarking works [1, 39] characterize the heterogeneous performance and energy profiles of different implementations but stop short of integrating these profiles into a live orchestration decision system.

The proposed framework is, to the best of our knowledge, the first to jointly address all three dimensions: it explicitly models the heterogeneous CPU utilization and power consumption behavior of SPGWU and VPP UPF implementations through empirical profiling, integrates a CNN-LSTM-based traffic prediction model for proactive 3-step-ahead demand forecasting, and formulates an energy-aware deployment optimization that jointly determines both the number and type of UPF instances at each decision interval. This joint consideration of traffic prediction, implementation heterogeneity, and UPF-level energy modeling enables

proactive, high-performance, and energy-efficient orchestration in cloud-native 5G core networks — bridging the gap between empirical UPF characterization and real-time autonomous deployment decisions, and establishing a foundation for sustainable beyond-5G and 6G core network operation.

5.3 System Architecture

5.3.1 UPF Technologies

As described in Chapter 2 and empirically profiled in Chapter 3, the two software-based UPF implementations considered in this work, SPGWU and VPP, exhibit fundamentally different performance and energy characteristics that make them complementary rather than interchangeable.

The SPGWU-based UPF operates within the standard Linux kernel networking stack, resulting in moderate CPU utilization and energy consumption at low-to-moderate traffic loads. However, its throughput is bounded by kernel-space processing overhead, and beyond a saturation point, additional traffic yields disproportionate increases in resource consumption without corresponding throughput gains.

The VPP-based UPF bypasses the kernel entirely through DPDK and processes packets in vectors on dedicated polling CPU cores, sustaining significantly higher throughput than SPGWU. The trade-off is a persistently high CPU and energy footprint regardless of traffic volume, as the polling mechanism consumes resources even in the absence of traffic.

These contrasting profiles, SPGWU efficient at low load and VPP necessary at high load, form the basis of the heterogeneous deployment strategy proposed in this chapter.

5.3.2 Dataset & Preprocessing

For the purposes of this work, we utilized a 5G production dataset [75] provided by an Irish mobile operator. The dataset contains key performance indicators (KPIs) collected from a live 5G network under different mobility experiments and application usage patterns. It includes multiple context and radio-related metrics such as channel conditions, cell statistics, and throughput measurements. In this study, we focus on the uplink throughput time series, selecting a trace of more than *6,000* samples representing the actual uplink traffic demand

observed in the network.

Throughput scaling. To ensure that the traffic trace matches the operational range of our experimental testbed, we apply a uniform scaling procedure. The aggregate throughput capacity of the testbed reaches up to 4.5 Gbps under full VPP deployment across all five worker nodes. The scaling is performed as:

$$x_{\text{scaled}} = x_{\text{original}} \times \frac{C}{x_{\max}}, \tag{5.1}$$

where x_{original} denotes the original throughput values, $x_{\max}$ is the maximum throughput observed in the dataset, and C is the target maximum throughput. This step maintains the temporal behavior of the original traffic trace while adjusting its scale to match the capacity constraints of our testbed.

Smoothing and anomaly mitigation. Before feeding the data to the model for training, we make sure that the scaled throughput time series is smoothed to reduce short-term fluctuations and mitigate abnormal spikes that could negatively affect our model's decisions. In order to achieve that, we employ the Savitzky–Golay filter [76] with a window half-size of *m* and polynomial degree $k = 3$, which performs local polynomial regression over a sliding window and smooths values while preserving important structural characteristics such as peaks and valleys. For a window centered at sample index i, the filter computes the smoothed value as:

$$y_{\text{smooth},i} = \sum_{j=-m}^{m} c_j \, y_{i+j}, \tag{5.2}$$

where c_j are pre-computed coefficients derived from the least-squares polynomial fit of degree k. Following smoothing, all sequences are normalized using Min-Max scaling:

$$\tilde{y}_i = \frac{y_{\text{smooth},i} - y_{\min}}{y_{\max} - y_{\min}}, \tag{5.3}$$

where $y_{\min}$ and $y_{\max}$ are computed over the training set only and applied to both training and test sets to prevent data leakage. Model outputs are inverse-transformed to recover throughput values in Mbps for evaluation.

5.3.3 Machine Learning

As part of the proposed system architecture, different machine learning models are assessed for short-term uplink throughput prediction using historical network measurements, allowing the identification and selection of the most suitable model for the framework. For

Table 5.1: Hyperparameters of evaluated prediction models

Parameter	LSTM	GRU	XGB	CNN-LSTM
Window	30	30	30	30
Features	1	1	1	1
Horizon	3	3	3	3
Layer 1	LSTM(100)	GRU(100)	–	Conv1D(64,3)
Layer 2	Drop(0.2)	Drop(0.2)	–	MaxPool(2)
Layer 3	LSTM(50)	GRU(50)	–	LSTM(100)
Layer 4	Drop(0.2)	Drop(0.2)	–	Drop(0.2)
Layer 5	Dense(3)	Dense(3)	–	Dense(32)
Layer 6	–	–	–	Dense(3)
Normaliz.	MinMax	MinMax	MinMax	MinMax
Optimizer	Adam	Adam	–	Adam
LR	0.001	0.001	0.05	0.001
Loss	MAE	MAE	–	MAE
Batch	16	16	–	16
Epochs	50	50	–	50
Early stop	p=10	p=10	–	p=10
LR sched.	f=0.5	f=0.5	–	f=0.5
Estimators	–	–	50	–
Max depth	–	–	8	–
Subsample	–	–	0.8	–
Col sample	–	–	0.8	–

all experiments, the same preprocessing pipeline is used, while differing only in their prediction module to ensure fairness. Four different types of prediction methods are considered—XGBoost, LSTM, GRU, and CNN-LSTM—that range from relatively straightforward approaches to more sophisticated models, allowing us to determine which approach is suitable for the problem under study. The training configuration of all evaluated models are summarized in Table 5.1

5.3.3.1 Plain LSTM

The first candidate of this evaluation is a standard LSTM [77] architecture, characterized by maintaining its state via gated units; hence, it is capable of capturing both short-term variations and long-term trends in the time series. In the current implementation, a fixed window of historical throughput is presented to LSTM to generate multi-step ahead predictions. Despite being rather strong, LSTMs require increased computational efforts due to their architecture.

5.3.3.2 GRU

The Gated Recurrent Unit (GRU) [78] is integrated in our evaluation as a computationally efficient alternative to LSTM. GRU simplifies the recurrent structure by merging the input and forget gates, reducing the number of trainable parameters while preserving the ability to learn temporal correlations. In the proposed architecture, GRU utilizes the same throughput window as the LSTM and is also fixed to generate 3-step ahead predictions. This design is considered in order to provide a possible trade-off between prediction accuracy and computational efficiency and assess how suitable a GRU model could be for near real-time deployment scenarios.

5.3.3.3 XGBoost

XGBoost [79] is adopted as a non-sequential, tree-based regression model to serve as a lightweight baseline. XGBoost does not natively model temporal dependencies, but converts the throughput time series into a supervised learning format using lag-based features. In order to achieve that XGBoost utilizes separate models that are trained for each prediction horizon step. This architecture, in contrast with all the other ML-Model candidates, provides extremely fast training and inference, but relies heavily on feature engineering and it is more unlikely to capture complex temporal dynamics compared to recurrent models.

5.3.3.4 CNN-LSTM

Finally in our evaluation we make use of a CNN-LSTM [80] architecture which is capable of combining convolutional and recurrent components to jointly capture local temporal patterns and long-term dependencies. In this design, one-dimensional convolutional layers first extract short-term throughput patterns and fluctuations from the input sequence. The

resulting feature maps are then passed to an LSTM layer that models long-range temporal relationships. This hybrid architecture is particularly effective in handling bursty throughput behavior. However this comes at the cost of additional computational complexity compared to plain recurrent models.

5.3.4 End-to-end Architecture & Functionality

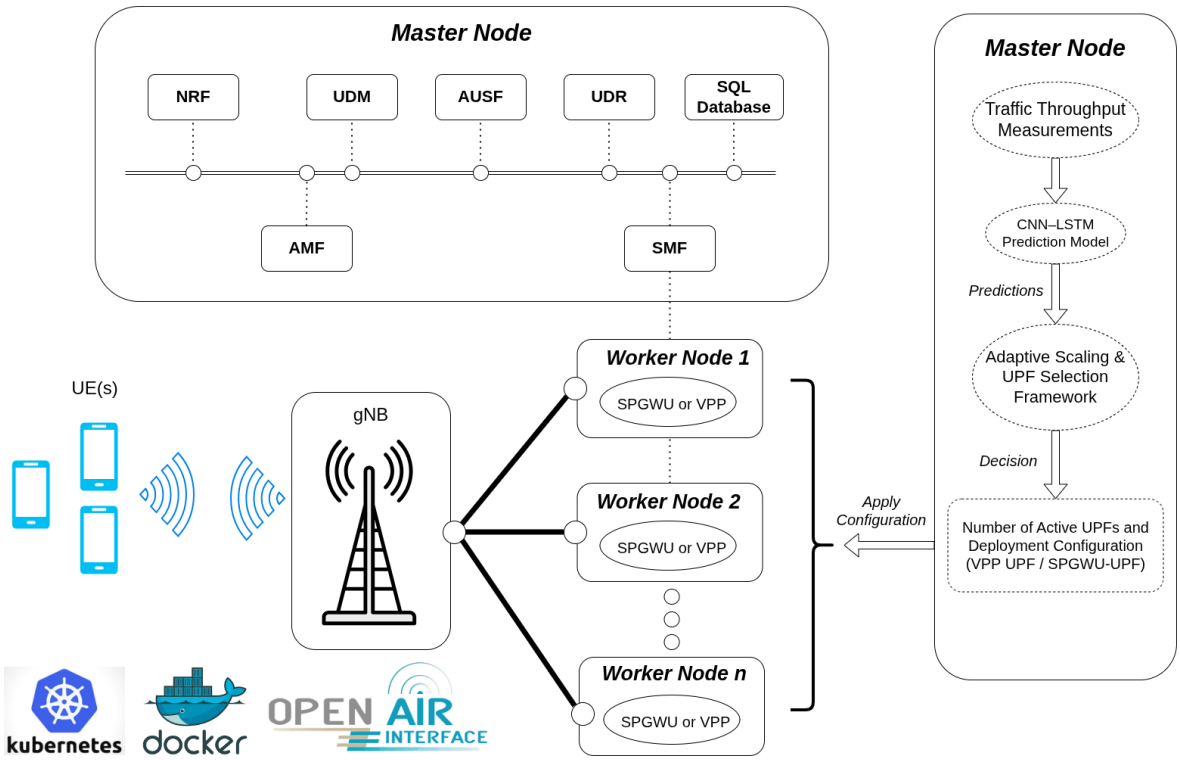


Figure 5.1: Overall System Architecture

The proposed framework is implemented on a real-world experimental testbed illustrated in Figure 5.1. The testbed consists of a complete 5G Standalone (SA) core network comprising the UDR, UDM, NRF, SQL database, AUSF, AMF, SMF, and UPF instances, with UERANSIM used to emulate both the gNB and multiple UEs. All 5G core components and emulated RAN entities are deployed as Docker containers and orchestrated through Kubernetes across a multi-node cluster consisting of one master node and five worker nodes. The complete hardware and software specification of the testbed is summarized in Table 5.2.

The master node hosts the complete 5G core network, the CNN-LSTM traffic prediction model, and the energy-aware UPF selection framework. At each decision interval of $\Delta t = 10$ seconds — aligned with the traffic application intervals used in the experimental evaluation — the master node executes the following sequential operations: (i) collects the latest uplink

Table 5.2: Testbed Hardware and Software Specification

Component	**Master Node**	**Worker Nodes**
CPU model	Intel Core i7-8700 (Coffee Lake)	Intel Core i7-12700K (Alder Lake, KVM)
vCPUs	12 (6C/12T, HT)	4 vCPUs
Base freq.	3.20 GHz	3.60 GHz
Max turbo	4.60 GHz	5.00 GHz
L2 cache	1.5 MiB	16 MiB
L3 cache	12 MiB	32 MiB
Node type	Bare metal	KVM virtual machine
OS	Ubuntu 22.04.5 LTS	Ubuntu 22.04.5 LTS
Kernel	6.8.0-106-generic	5.15.0-164-generic
Kubernetes	v1.31.8	
Docker	v28.1.1 (Engine)	
OAI CN5G	v2.1.0	
OAI UPF-VPP	oai-upf-vpp:latest†	
UERANSIM	rohankharade/ueransim:latest†	
†Docker images pulled on June 2025.		

throughput measurements from the running UPF instances; (ii) feeds the most recent $W = 30$ samples into the CNN-LSTM model to generate a 3-step-ahead demand prediction $\hat{D}(t)$; (iii) solves the energy-aware deployment optimization described in Section 5.3.4.1 to determine the optimal number and type of UPF instances; and (iv) applies the selected configuration by instantiating or terminating UPF containers on the worker nodes via the Kubernetes API. The five worker nodes are exclusively dedicated to hosting UPF instances. At any given decision interval, each worker node may be idle (no UPF deployed), running an SPGWU-based UPF, or running a VPP-based UPF, depending on the traffic prediction and the energy-aware

configuration selected by the framework. This architecture enables the system to dynamically transition between deployment configurations in response to predicted traffic demand, exploiting the complementary energy-performance characteristics of the two UPF implementations.

5.3.4.1 Mathematical Model

We consider a 5G core network deployed on a cluster of N worker nodes, where each node can host at most one UPF instance. At each decision interval t, the framework determines the number and type of UPFs to deploy based on the predicted traffic demand.

Let $D(t)$ denote the measured aggregate traffic demand at time t, and let $\hat{D}(t)$ denote the predicted demand obtained from a machine learning model trained on historical traffic data. The predicted demand is used by the framework to determine the UPF deployment configuration, while the measured demand is used post-hoc to evaluate the realized CPU utilization and energy consumption.

Two UPF implementations are available, SPGWU and VPP, denoted by the set $\mathcal{U} = \{\text{SPGWU}, \text{VPP}\}$. Each UPF type $u \in \mathcal{U}$ is characterized by a maximum throughput capacity C_u. Let $x_u(t)$ represent the number of deployed UPFs of type u at time t. Since each worker node can host at most one UPF instance, the deployment is constrained by the number of available nodes. Additionally, the aggregate processing capacity of the deployed UPFs must satisfy the predicted traffic demand:

$$\sum_{u \in \mathcal{U}} x_u(t) \leq N, \quad \sum_{u \in \mathcal{U}} x_u(t) C_u \geq \hat{D}(t). \tag{5.4}$$

For each candidate deployment configuration, the framework distributes the predicted traffic demand among the active UPF instances in order to evaluate the expected CPU utilization and energy consumption. Traffic is distributed proportionally to the processing capacity contributed by each UPF type, so that all active instances operate at the same fraction of their respective saturation throughput. This capacity-proportional policy maximizes the headroom of every instance against prediction error, and renders the per-instance load — and hence the expected CPU utilization via (5.7) — deterministic for each candidate configuration, allowing exact evaluation of the optimization objective. The predicted traffic handled by UPF type u is therefore:

$$D_u^{\text{pred}}(t) = \hat{D}(t)\frac{x_u(t)C_u}{\sum_{v\in\mathcal{U}} x_v(t)C_v}. \tag{5.5}$$

The predicted traffic processed by each individual UPF instance of type u is then given by:

$$d_u^{\text{pred}}(t) = \frac{D_u^{\text{pred}}(t)}{x_u(t)}. \tag{5.6}$$

Through extensive experimental profiling, we characterize the CPU utilization behavior of each UPF implementation under controlled uplink traffic conditions. For each UPF type, uplink throughput was incremented in steps of 5 Mbps starting from zero up to the maximum sustainable throughput of each implementation — approximately 500 Mbps for SPGWU and 900 Mbps for VPP. At each throughput level, the instantaneous CPU utilization was continuously monitored and recorded. This procedure was repeated across multiple independent runs to account for measurement variance, and polynomial regression was applied to the aggregated CPU-throughput measurements to derive a smooth and representative utilization profile for each implementation. A quadratic polynomial was selected based on goodness-of-fit analysis — both SPGWU and VPP exhibit a convex, non-linear CPU-throughput relationship that a linear model cannot capture accurately, while higher-degree polynomials did not yield statistically significant improvement. For each UPF type $u \in \mathcal{U}$, the expected CPU utilization is modeled as:

$$\text{CPU}_u^{\text{exp}}(t) = a_u\big(d_u^{\text{pred}}(t)\big)^2 + b_u\, d_u^{\text{pred}}(t) + c_u, \tag{5.7}$$

where a_u, b_u, and c_u are regression coefficients derived from the profiling measurements and reported in Table 5.3. The model is applied within the measured operational throughput range and clamped to the physically observed CPU bounds during the optimization process.

Table 5.3: Regression Coefficients of the CPU Utilization Model ($\text{CPU}_u = a_u d^2 + b_u d + c_u$)

UPF Type	a_u	b_u	c_u
SPGWU	1.18×10^{-4}	0.127	3.01
VPP	1.00×10^{-6}	0.045	105.56

The coefficients reflect the fundamentally different processing architectures of the two implementations. The SPGWU baseline offset $c_{\text{SPGWU}} = 3.01\%$ confirms its negligible idle

CPU overhead under interrupt-driven kernel processing. In contrast, $c_{\text{VPP}} = 105.56\%$ quantifies the continuous polling cost of VPP regardless of traffic load — the primary driver of its higher energy consumption at low and moderate throughput levels.

In contrast, the *actual CPU utilization* of each UPF instance is measured in real time during operation using system monitoring tools. Let $\text{CPU}_u^{\text{act}}(t)$ denote the instantaneous measured CPU usage of a UPF instance of type u. This measured CPU is used post-hoc to evaluate the realized power and energy consumption independently to assess the performance of the proposed framework.

The instantaneous power consumption of a node is modeled as a linear function of CPU utilization [81]. Since CPU utilization is reported in per-core units as measured by the `top` utility, its value ranges over $[0, 400]\%$ on the 4-vCPU worker nodes — exceeding 100% for the multi-threaded VPP implementation — and is therefore normalized by the total core capacity of the node:

$$P_u(t) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}}) \frac{\text{CPU}_u(t)}{400}, \tag{5.8}$$

where P_{idle} and P_{max} denote the idle and full-load power consumption of a worker node. As the workers are 4-vCPU virtual machines whose power cannot be metered directly, these parameters are set to $P_{\text{idle}} = 15$ W and $P_{\text{max}} = 126$ W, corresponding to the idle and full-load power attributed to each virtual machine, consistent with published per-core power characterizations of the host's Intel Alder Lake platform [82, 83].

Based on this model, the framework estimates the expected instantaneous power consumption of each UPF instance as:

$$P_u^{\text{exp}}(t) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}}) \frac{\text{CPU}_u^{\text{exp}}(t)}{400}. \tag{5.9}$$

Similarly, the realized power consumption is computed by applying the same model to the CPU utilization measured during operation:

$$P_u^{\text{act}}(t) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}}) \frac{\text{CPU}_u^{\text{act}}(t)}{400}. \tag{5.10}$$

The total energy consumption over the experiment duration $[t_0, t_1]$ is then computed as:

$$E^{\text{exp}} = \sum_{u \in \mathcal{U}} \int_{t_0}^{t_1} x_u(t)\, P_u^{\text{exp}}(t)\, dt, \tag{5.11}$$

$$E^{\text{act}} = \sum_{u \in \mathcal{U}} \int_{t_0}^{t_1} x_u(t)\, P_u^{\text{act}}(t)\, dt. \tag{5.12}$$

Worker nodes hosting no UPF instance are assumed to be paused by the framework and therefore contribute no energy in this accounting, while every worker hosting a UPF is charged its full modelled power, including its idle component. In the static baselines, all workers host a UPF throughout the experiment and are charged accordingly. Likewise, since the master node executes an identical control plane in every scenario, it is charged only the marginal energy of the CNN-LSTM inference and the scaling agent in the proposed framework's scenario

Finally, the framework selects the number and type of UPF instances that minimize the expected energy consumption while satisfying the predicted traffic demand and cluster resource constraints:

$$\begin{aligned} \min_{\{x_u(t)\}} \quad & E^{\text{exp}} \\ \text{s.t.} \quad & \sum_{u \in \mathcal{U}} x_u(t) C_u \geq \hat{D}(t), \\ & \sum_{u \in \mathcal{U}} x_u(t) \leq N, \\ & x_u(t) \in \mathbb{Z}^+, \quad \forall u \in \mathcal{U}. \end{aligned} \tag{5.13}$$

In this formulation, the framework uses the instantaneous expected energy $E^{\text{exp}}(t)$ to select the most appropriate UPF configuration and deployment at each decision interval, while E^{act} is computed post-hoc to evaluate the realized energy consumption and assess both the performance and the efficiency of the deployed configuration.

5.4 Experimental Setup & Evaluation

5.4.1 Experimental Setup

A controlled experimental setup was designed to systematically evaluate all scenarios under identical, varying uplink traffic demands. From the complete 5G production dataset described in Section 5.3.2, a representative subset of 230 throughput values was extracted to drive the physical testbed experiments. This subset was selected to keep the experiments temporally manageable while preserving a wide range of traffic conditions, spanning from 500 Mbps to 3,200 Mbps, as illustrated in Fig. 5.2. Uplink traffic was generated using `iperf3`, with each throughput value applied for a $\Delta t = 10$-second interval, resulting in a total experiment duration of approximately 2,300 seconds per scenario. The generated traffic originates at the emulated UEs and is transmitted through the established PDU session, traversing the

deployed UPF(s) before reaching the data network. This end-to-end path allows direct observation of each UPF deployment strategy in terms of throughput, packet loss, CPU utilization, and energy consumption.

As previously mentioned, all experiments were deployed on the SLICES-RI testbed [22]. The cluster comprises one master node and five worker nodes. The master node hosts the complete OAI 5G core network, the CNN-LSTM prediction model, and the energy-aware UPF selection framework, while the five worker nodes are exclusively dedicated to hosting UPF instances.

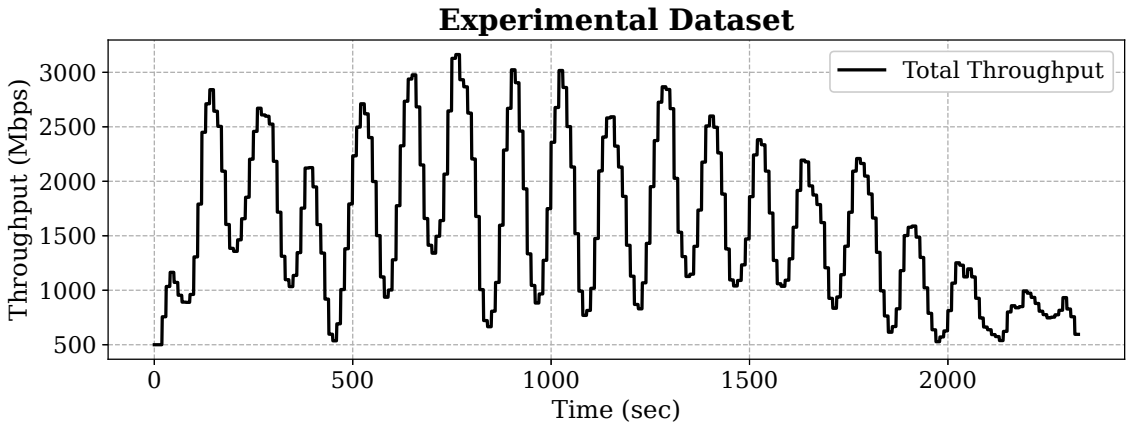


Figure 5.2: Experimental traffic dataset: Total throughput over Time.

5.4.2 Experimental Scenarios

To evaluate the performance and energy efficiency of different UPF deployment strategies, five experimental scenarios are considered. All scenarios are conducted on the same cluster and under identical traffic conditions to ensure a fair comparison. The evaluated scenarios are as follows:

1. **Single SPGWU Deployment:** A single SPGWU-based UPF is deployed on one worker node and is responsible for handling the entire traffic demand. This scenario serves as a baseline to assess the performance limitations and energy behavior of a kernel-based UPF under increasing throughput requirements.

2. **Single VPP Deployment:** A single OAI VPP-based UPF is deployed on one worker node under the same traffic profile. This scenario enables a direct comparison between SPGWU- and VPP-based UPFs operating in isolation under identical network demands.

3. **Multiple SPGWU Deployments:** SPGWU-based UPF instances are deployed across all five worker nodes using a greedy traffic distribution strategy: traffic is forwarded to an additional UPF instance only when the CPU utilization of existing active instances reaches the saturation threshold of 95%. This scenario evaluates the scalability and cumulative energy consumption of a homogeneous kernel-based deployment.

4. **Multiple VPP Deployments:** Five OAI VPP-based UPF instances are deployed across all worker nodes following the same greedy allocation strategy. This scenario evaluates the scalability, performance stability, and energy implications of a homogeneous high-performance deployment under maximum cluster utilization.

5. **Proposed Optimization Framework:** The proposed energy-aware framework is enabled. At each decision interval, the CNN-LSTM model generates a 3-step-ahead demand prediction, and the optimization mechanism determines both the number and type of UPF instances to deploy — selecting between SPGWU- and VPP-based UPFs — to satisfy predicted demand while minimizing energy consumption. This scenario is deployed in order to observe and evaluate the adaptive capability of the proposed approach.

5.4.3 Machine Learning Evaluation

Table 5.4 summarizes the prediction accuracy of all evaluated models under the 3-step-ahead forecasting setting, and Fig. 5.3 illustrates the actual versus predicted throughput for each model. The 3-step-ahead horizon is selected because it provides the orchestration layer sufficient time to execute UPF selection and apply the required Kubernetes deployment changes before the predicted traffic demand materializes — a reaction window that single-step-ahead prediction cannot provide without forcing the system into a largely reactive operating mode.

For the UPF selection task, MAE and MAPE are adopted as the primary model selection criteria. MAE penalizes absolute prediction errors uniformly, which maps directly to

Table 5.4: Prediction Accuracy of Evaluated Models (3-Step-Ahead Forecasting)

Model	**RMSE (Mbps)**	**MAE (Mbps)**	**MAPE (%)**	**Inference time (ms)**
LSTM	107.86	79.77	5.99	20-30
GRU	106.29	82.05	6.30	10-20
XGBoost	155.18	118.48	8.84	3-35
CNN-LSTM	106.53	**77.95**	**5.74**	25-35

over-provisioning or under-provisioning costs in UPF deployment decisions. MAPE captures relative errors, which is particularly relevant under the wide dynamic range of traffic conditions in this dataset. RMSE, which disproportionately penalizes large errors, is reported for completeness. Under these criteria, CNN-LSTM achieves the best overall performance with a MAE of 77.95 Mbps and MAPE of 5.74%, and is therefore selected as the prediction model for the proposed framework.

As shown in Fig. 5.3d, the CNN-LSTM model provides the closest alignment with actual throughput, particularly during rapid fluctuations and burst events characteristic of real 5G networks. As already discussed, the front-end extracts short-term local patterns where at the same time the LSTM layer captures the long-range temporal dependencies, making this hybrid architecture a well-suited solution for the bursty traffic observed in the dataset.

In the case of the plain LSTM (Fig. 5.3a we can see it demonstrates competitive performance compared to CNN-LSTM with RMSE of 107.86 Mbps, MAE of 79.77 Mbps, and MAPE of 5.99%, showing as well its ability to capture temporal dependencies for the 3-step ahead prediction. However, we can see that it exhibits slightly larger deviations during bursty periods compared to CNN-LSTM, indicating reduced performance when it comes to abrupt throughput transitions.

The GRU model (Fig. 5.3b) achieves RMSE of 106.29 Mbps which is lower compared to both CNN-LSTM and Plain LSTM but its MAE of 82.05 Mbps and MAPE of 6.30%, which correspond to the key metrics in this work, are higher than both. The simplified gating mechanism of GRU makes it harder to capture throughput variations over multiple future steps, particularly under rapidly changing network conditions where the model tends to lag

behind sudden transitions.

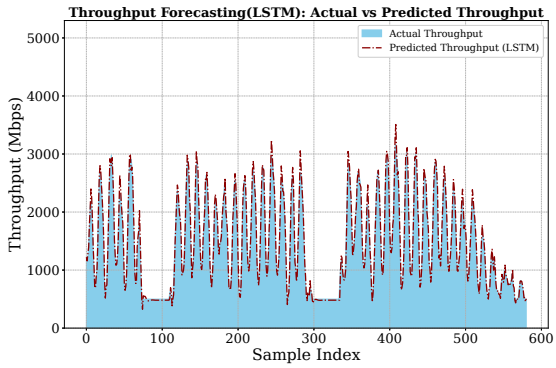
Finally, XGBoost (Fig. 5.3c), as expected, exhibits the weakest performance under the multi-step predictions, with RMSE of 155.18 Mbps, MAE of 118.48 Mbps, and MAPE of 8.84%. Its reliance on fixed lag-based features prevents effective capture of complex non-linear temporal dependencies, leading to large deviations, especially during bursty intervals. On one hand, XGBoost offers the fastest execution time compared to the other models, but on the other hand, the higher prediction errors make it unsuitable for driving UPF deployment decisions, where inaccurate forecasts directly affect the efficiency of the framework, leading to suboptimal and wasteful resource allocation.

Given that prediction errors affect directly into UPF deployment decisions, where under-estimation leads to congestion and overestimation to unnecessary energy consumption the CNN-LSTM architecture was selected not only for achieving the lowest MAE of 77.95 Mbps and MAPE of 5.74%, but also because its convolutional front-end explicitly captures local burst patterns before the LSTM layer models long-range temporal dependencies, making it a better solution in cases of bursty traffic where misleading forecasts carry direct operational consequences.

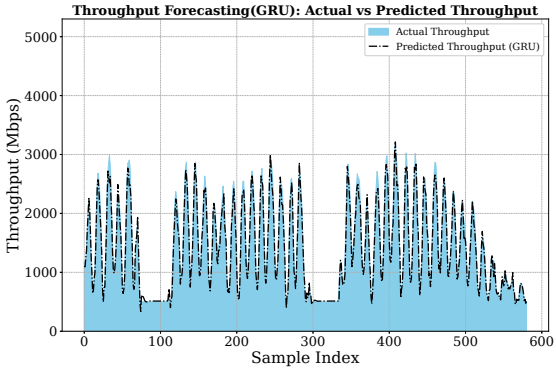
5.4.4 Experimental Results

5.4.4.1 Single SPGWU

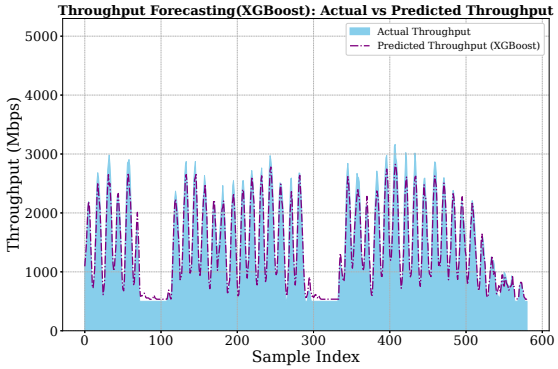
In the first experiment, a single SPGWU instance was deployed in the cluster to serve the whole traffic demand. The results retrieved show that a single SPGWU-based UPF was completely unable to sustain high throughput levels, with the maximum achievable throughput limited to approximately 500 Mbps. Due to the fact that most of the values in the dataset was more than this limit, the system was severely saturated, leading to an average packet loss of 60.63%. CPU utilization remained close to 100% at all times due to the high traffic demand, indicating that the SPGWU-based UPF was continuously saturated and as a consequence, the requested throughput could not be achieved, making us mark this experimental deployment as unsuccessful in terms of network performance.



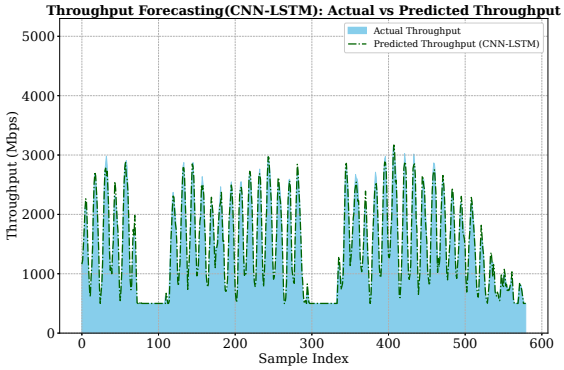
(a) LSTM: Actual vs Predicted Throughput



(b) GRU: Actual vs Predicted Throughput



(c) XGBoost: Actual vs Predicted Throughput



(d) CNN-LSTM: Actual vs Predicted Throughput

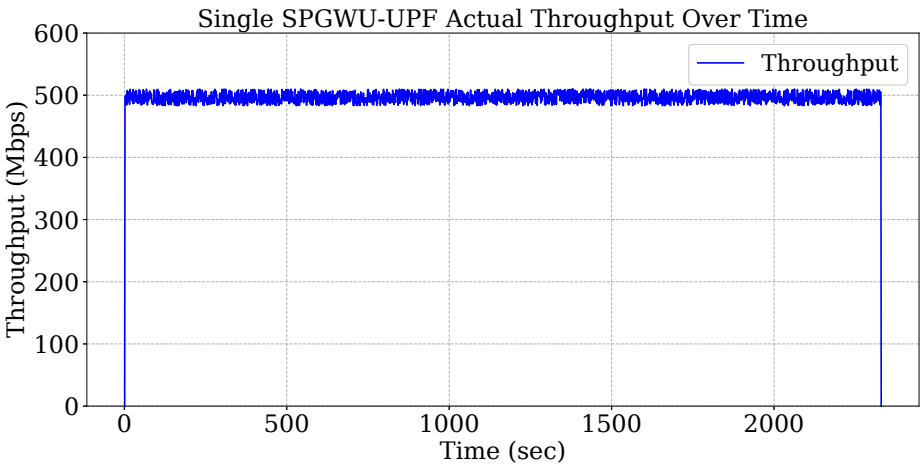
Figure 5.3: Comparison & Evaluation of different prediction models.

5.4.4.2 Single VPP

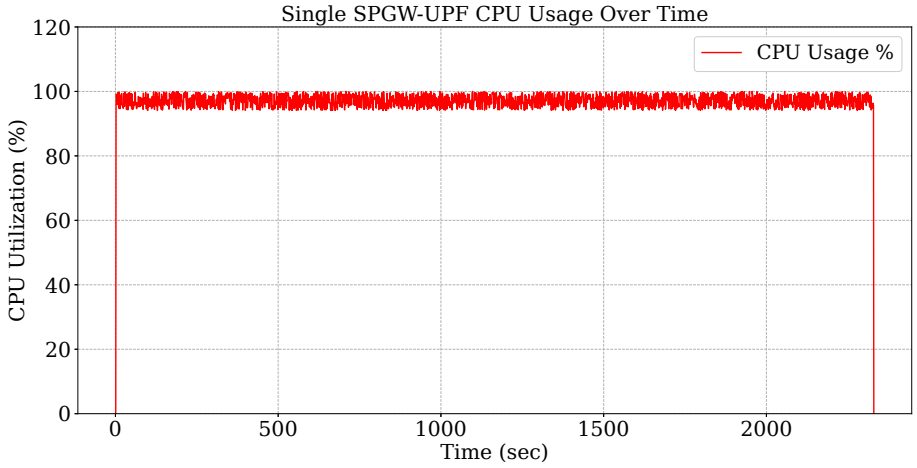
The second experiment as already stated uses a single VPP-based UPF under the exact same traffic conditions. When comparing these two cases we can observe that VPP demonstrated improved performance, achieving a maximum throughput of nearly 900 Mbps and reducing the average packet loss to 33.77%. However, this higher-performance, came at the cost of significantly higher CPU utilization, which exceeded 140% and was always over 100% even when no packets are transmitted due to the polling mechanism that VPP implementations utilize. In either case, even if VPP was able to accommodate low traffic demands more effectively than SPGWU, it still failed to meet the throughput requirements across the entirety of the dataset, and thus the experiment was also deemed unsuccessful.

5.4.4.3 Multiple SPGWUs

In the third conducted experiment we deployed SPGWU-based instances across all five worker nodes using the greedy traffic distribution strategy defined in Section 5.4.2. As shown



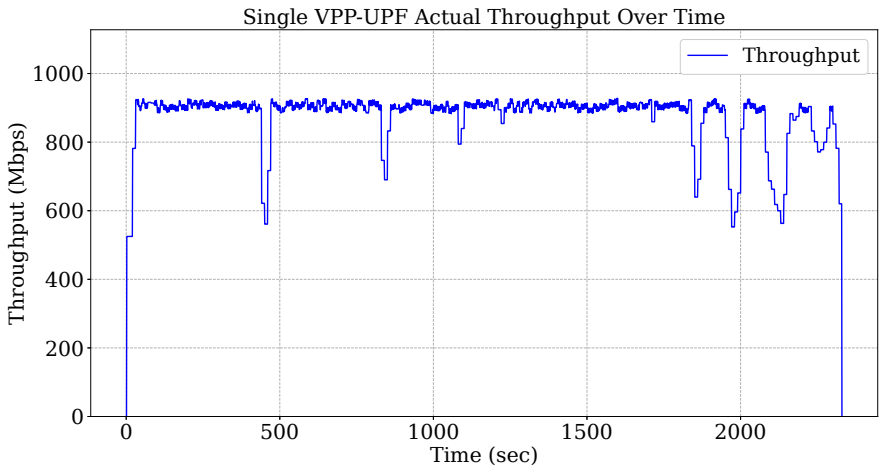
(a) Throughput over time.



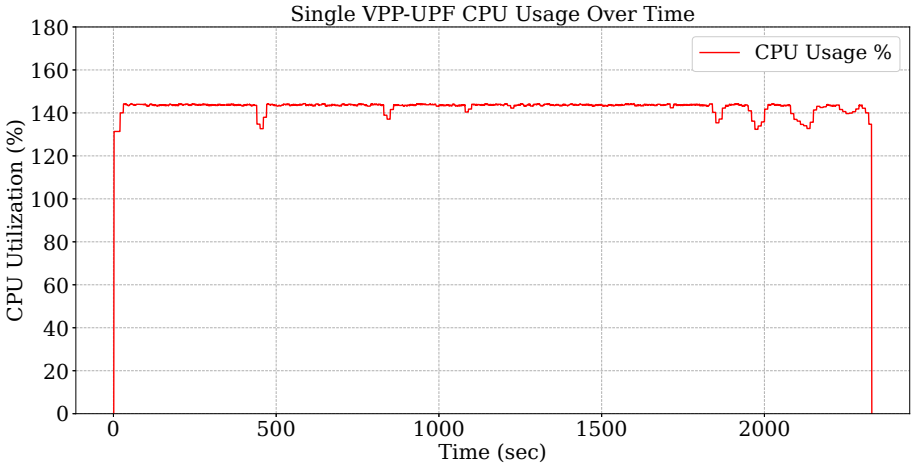
(b) CPU utilization over time.

Figure 5.4: Single SPGWU-UPF performance: Throughput and CPU usage over time.

in Fig. 5.6, this configuration utilized the combined processing capacity of the five instances and satisfied a broader range of the throughput demands. However, throughput values that exceeded 2,400 Mbps could not be fully met, with packet loss at high-demand intervals exceeding 28%. In terms of CPU utilization Fig. 5.7 shows that the utilization remained low during low traffic periods, but when the demand was at its peak all five instances approached 100% CPU utilization. Despite full resource utilization, throughput limitations persisted indicating that the architectural constraints of kernel-based packet processing cannot be overcome through horizontal scaling of homogeneous SPGWU-based UPFs. The total energy consumption for this specific experiment reached 368kJ and was calculated by adding the aggregated energy across all five SPGWU instances.



(a) Throughput over time.



(b) CPU utilization over time.

Figure 5.5: Single VPP-UPF: Throughput and CPU utilization over time.

5.4.4.4 Multiple VPPs

The fourth experiment is similar to the third experiment but in this case we deployed five VPP-based UPFs (instead of SPGWU-based UPFs) across all worker nodes following the same greedy allocation strategy. As shown in Fig. 5.8, this configuration successfully managed to meet all requested throughput values across the full range of the experimental dataset, achieving zero congestion with average packet loss below 1%. The combined processing capacity of five VPP instances was sufficient to handle even the highest traffic demands.

However, this performance came at a substantial energy cost (Fig. 5.9) where we see that each VPP-UPF consumed at least 100% CPU utilization even during idle or low-load periods

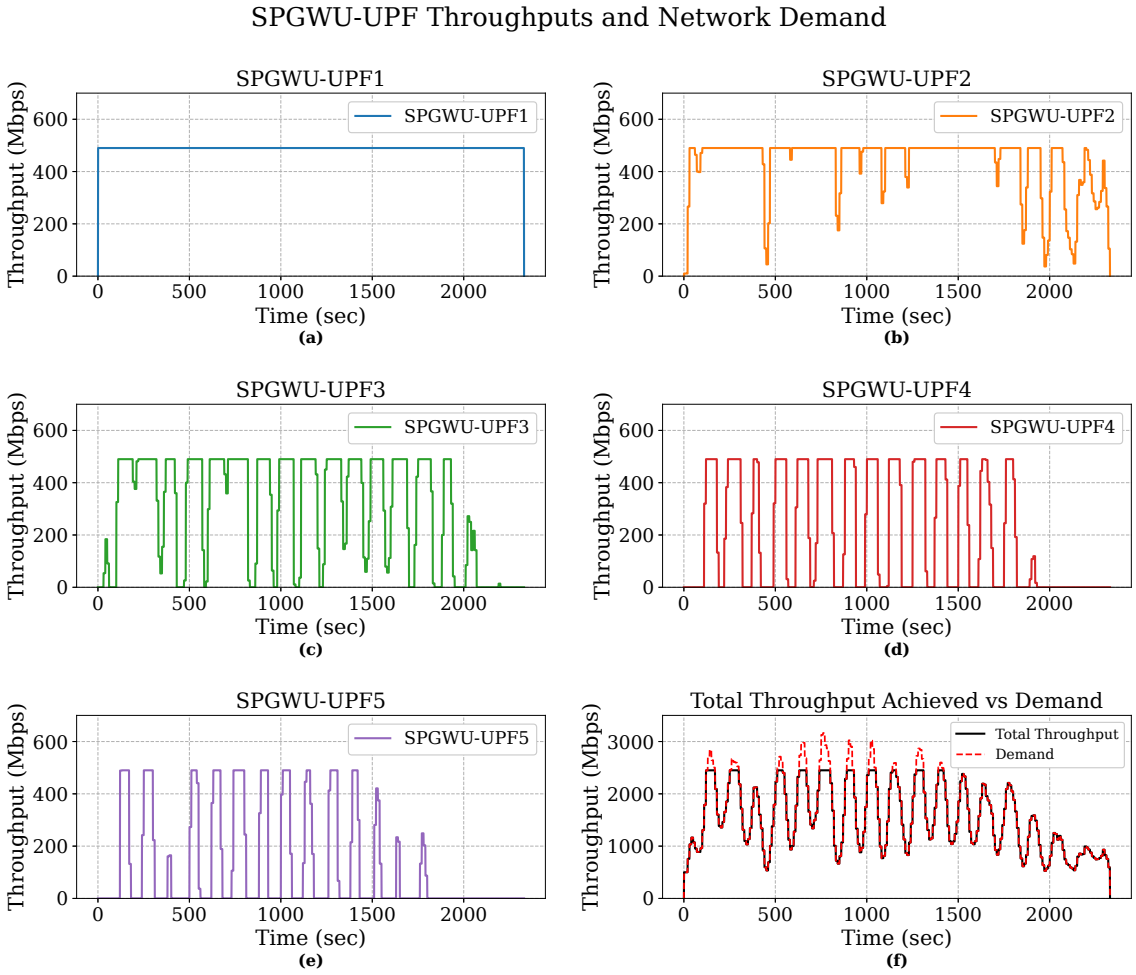


Figure 5.6: Multiple SPGWU deployment: per-instance throughput and aggregate demand.

due to the polling mechanism. During the peak values the CPU utilization reached approximately 150% and as a consequence, the total energy consumption reached 562 kJ, which is nearly 1.5 times that of the multiple SPGWU scenario. These results demonstrate the fundamental inefficiency of a static all-VPP deployment because while it satisfies performance requirements, it does so at the cost of continuous and unavoidable energy consumption across all nodes regardless of the nature of the instantaneous traffic load.

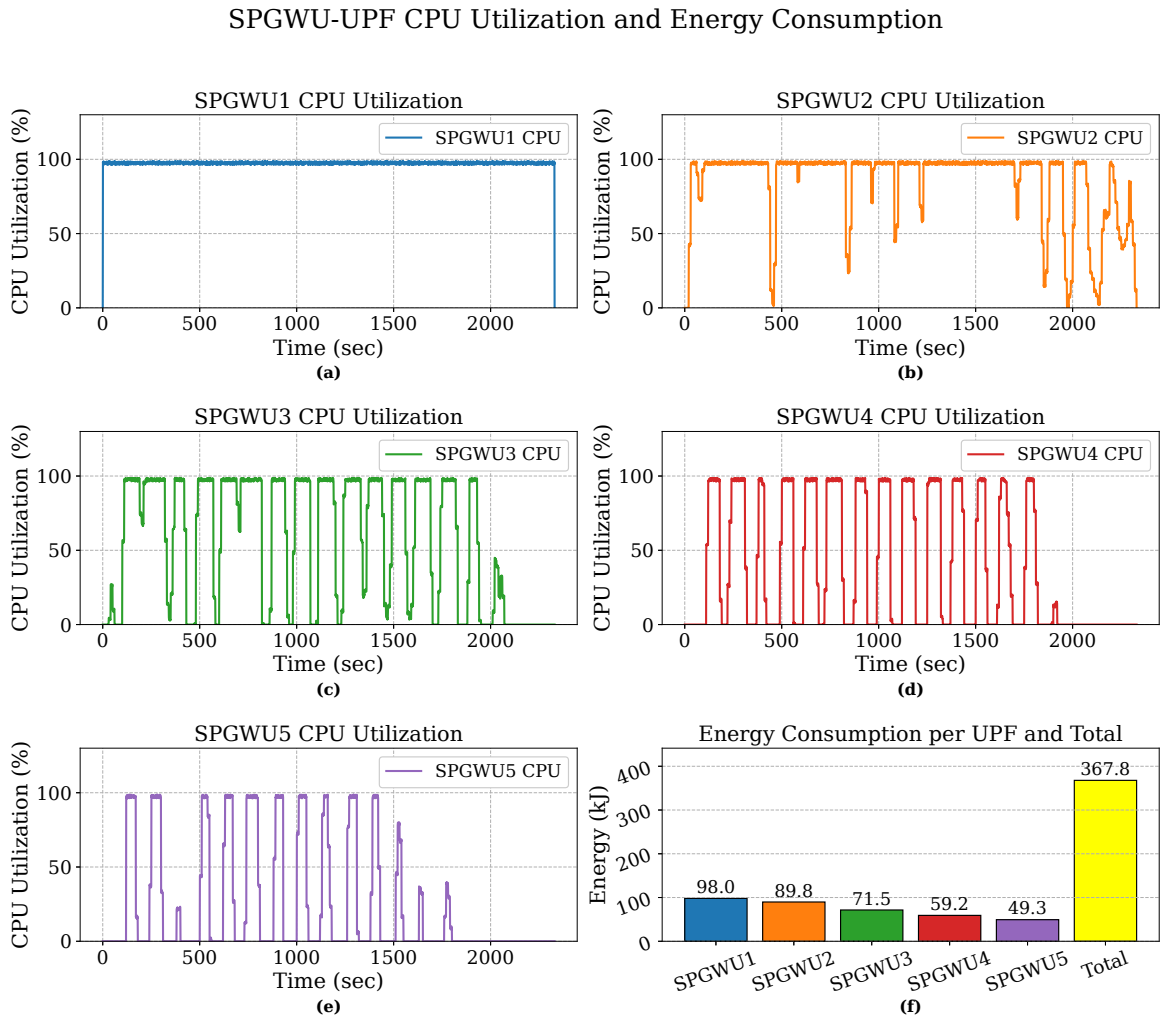


Figure 5.7: Multiple SPGWU deployment: per-instance CPU utilization and energy consumption.

5.4.4.5 Our Framework

The proposed energy-aware framework was evaluated under identical traffic conditions. At each 10-second decision interval, the CNN-LSTM model generated a 3-step-ahead demand prediction, and the optimization mechanism selected the number and type of UPF instances to deploy across the five worker nodes. As shown in Fig. 5.10, worker nodes dynamically transitioned between three states — idle (no UPF deployed), SPGWU deployment, and VPP deployment — in response to predicted traffic demand. This adaptive behavior allowed the framework to deploy lightweight SPGWU instances during low-to-moderate traffic periods and switch to VPP instances only when high throughput demands were anticipated, exploiting the complementary energy-performance characteristics of both implementations.

The framework successfully satisfied all throughput demands across the full experimen-

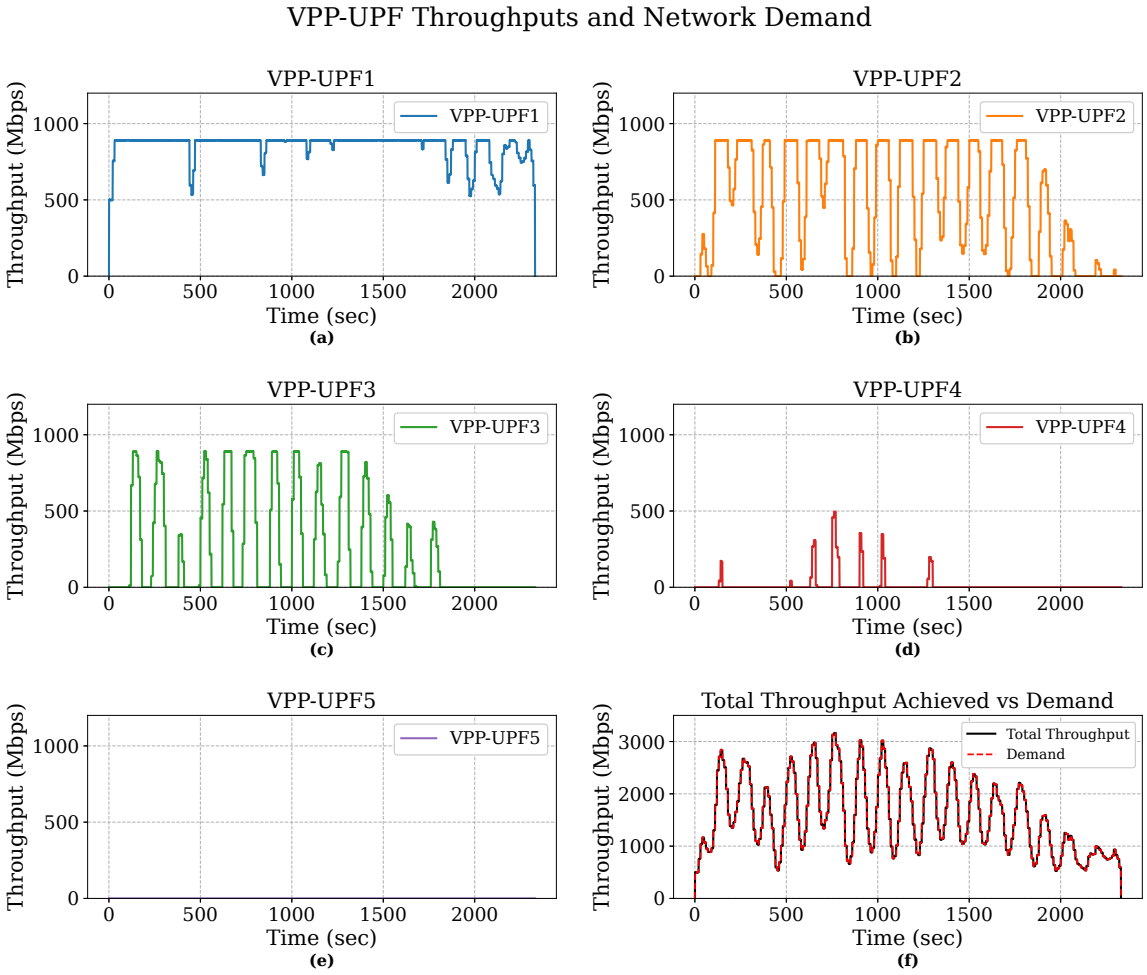


Figure 5.8: Multiple VPP deployment: per-instance throughput and aggregate demand.

tal range, achieving an average packet loss of only 1.5%. As shown in Fig. 5.11, the total energy consumption was reduced to 272 kJ, corresponding to an energy saving of approximately 51% compared to the static all-VPP deployment — the only baseline configuration capable of satisfying equivalent throughput requirements. This figure includes the computational overhead of the CNN-LSTM inference and the scaling agent running on the master node, which is reflected in the master node energy bar in Fig. 5.11.

It is worth noting that the convexity of the CPU–throughput behavior of the UPF implementations directly shapes the optimizer's decisions. Since the CPU cost of a given load grows faster than linearly at higher per-instance throughput, serving a demand with more, lightly loaded instances can incur lower total CPU cost than saturating fewer instances. The optimizer weighs this gain against the idle power of each additional active node, so the se-

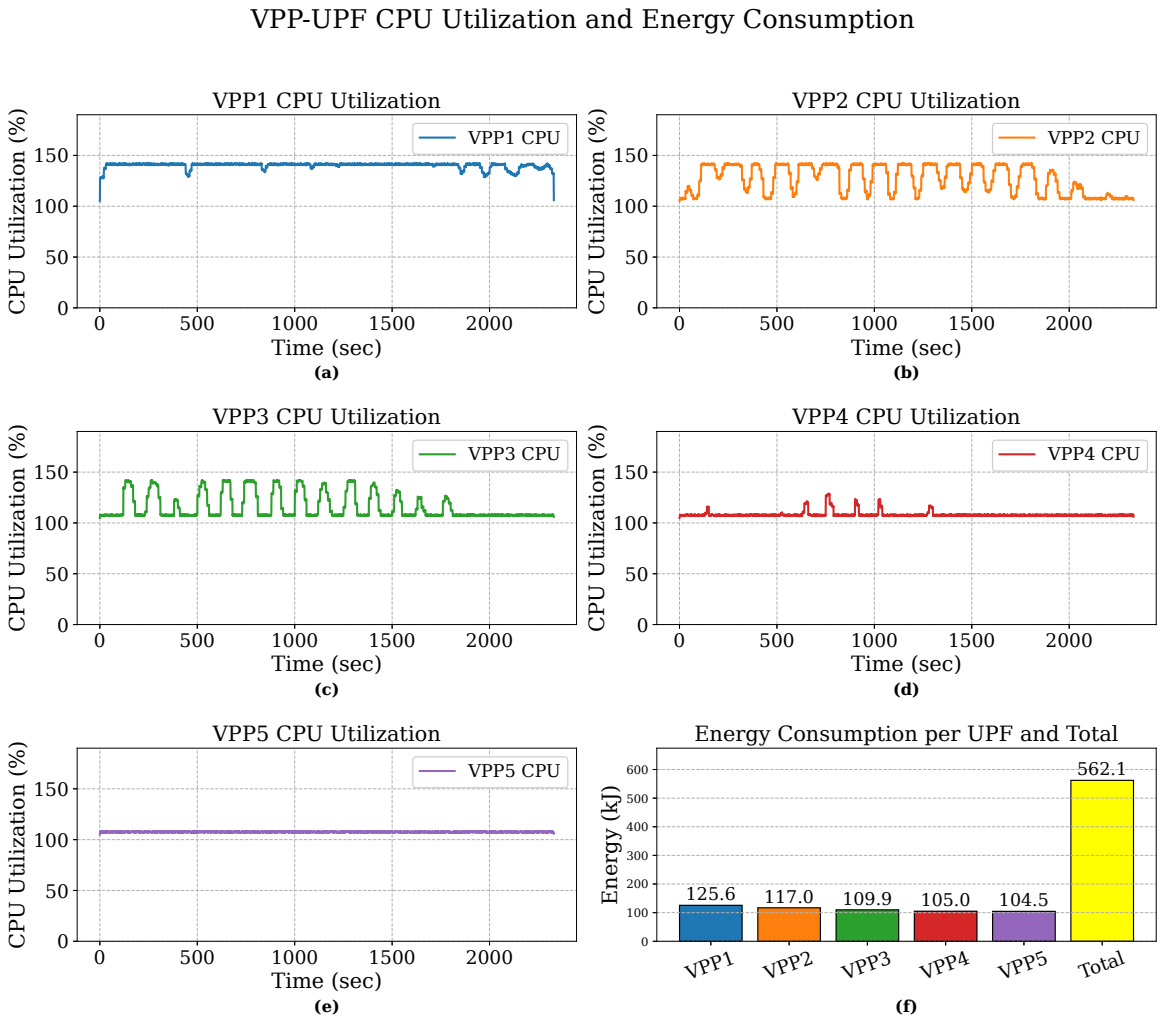


Figure 5.9: Multiple VPP deployment: per-instance CPU utilization and energy consumption.

lected instance count reflects a genuine energy trade-off rather than simply the minimum number of instances required to cover the demand.

5.4.5 Discussion

The experimental results reveal key findings that collectively validate the design rationale of the proposed framework.

Neither UPF type alone is sufficient. Single-UPF deployments confirm that SPGWU and VPP each address only part of the performance-energy space. SPGWU shows more efficient energy consumption at low loads but becomes saturated as soon as throughput values exceed 500Mbps, while on the other hand VPP sustains higher throughput albeit at the cost of continuous energy waste through polling regardless of actual traffic. Nevertheless none of

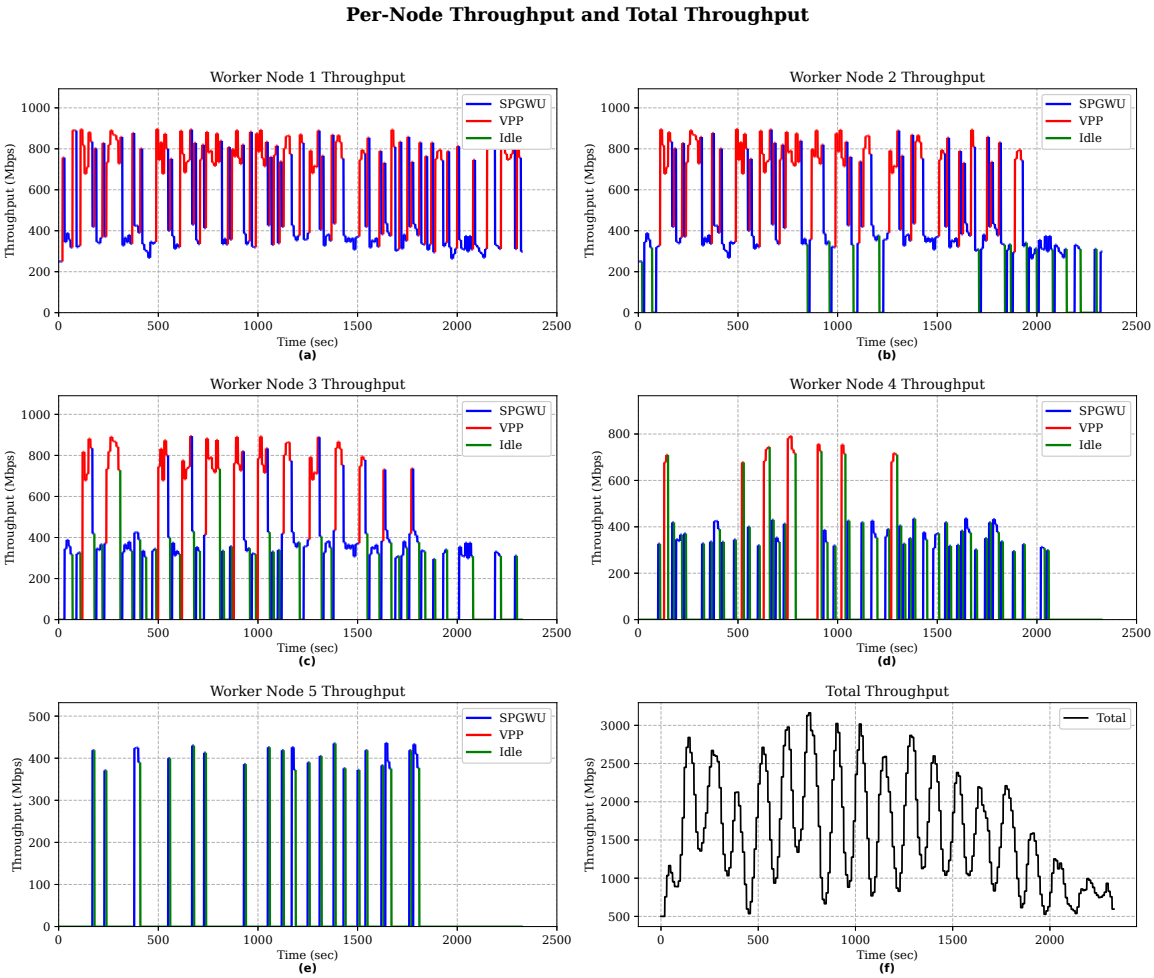


Figure 5.10: Proposed framework: per-node throughput and total aggregate throughput over time.

these implementations could sustain the demands when considered in isolation and this fundamental asymmetry between them makes heterogeneous orchestration not merely beneficial but necessary.

Horizontal scaling exposes architectural limits. As expected, deploying multiple homogeneous UPF instances improves aggregate capacity but does not resolve all the underlying problems. In the case of multiple SPGWU-based UPFs, it is evident that they cannot overcome the kernel-space throughput ceiling through horizontal scaling alone. On the other hand, the deployment of multiple VPP-based UPFs can successfully satisfy the performance requirements but at the cost of nearly 1.5 times the baseline energy, marking the deployment unsustainable for energy-conscious next-generation networks.

Proactive heterogeneous orchestration reconciles performance and efficiency. The proposed framework achieves the performance of the all-VPP baseline with approximately 51% less energy by dynamically selecting the appropriate UPF type at each decision interval

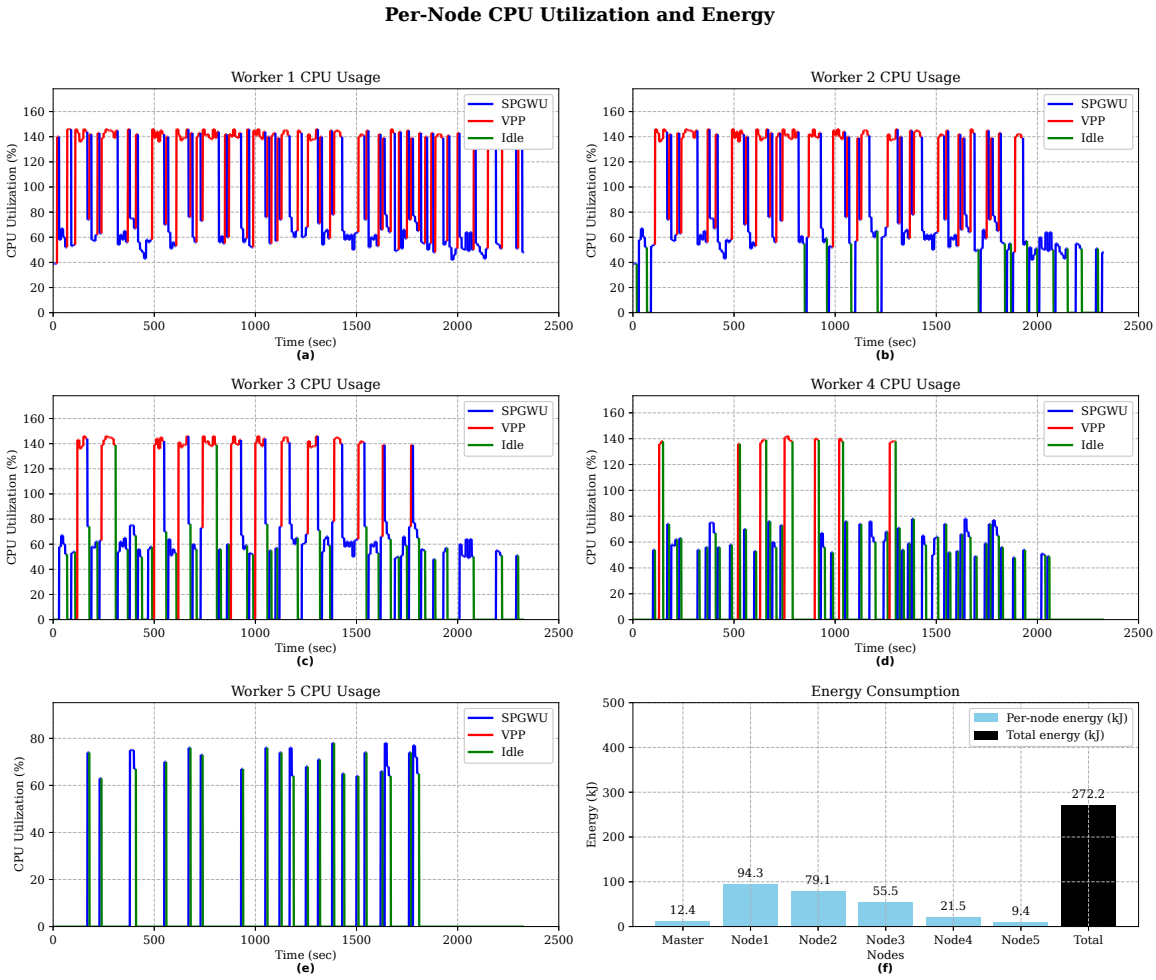


Figure 5.11: Proposed framework: per-node CPU utilization and energy consumption.

based on predicted demand. The CNN-LSTM prediction model is central to this outcome — by anticipating demand, the framework avoids both reactive over-provisioning and delayed scale-up that would cause network violations. These results suggest that the energy-performance trade-off in software UPF deployments is not a fixed constraint but a configurable operating point that can be continuously optimized through intelligent, prediction-driven orchestration — a property that becomes increasingly important as traffic volumes and energy costs grow toward 6G deployment scales.

Worst-case behavior under maximum demand. It is important to acknowledge the boundary conditions of the proposed framework. When traffic demand approaches the maximum testbed capacity of 4.5 Gbps — requiring all five worker nodes to simultaneously run VPP instances to satisfy throughput requirements — the framework converges to the same configuration as the static all-VPP deployment. In this worst-case scenario, the energy consumption of the proposed framework is equivalent to that of the all-VPP baseline, as

no energy-saving heterogeneous configuration exists that can satisfy the demand within the node budget. This behavior is an inherent property of the optimization problem: when the predicted demand exceeds the capacity achievable by any mixed SPGWU-VPP configuration, the optimizer correctly selects all-VPP as the only feasible solution. The 51% energy saving reported in this work therefore represents the average-case gain over the full experimental traffic range of 500 Mbps to 3,200 Mbps, where demand varies dynamically and the framework can exploit low-to-moderate traffic periods to deploy lighter SPGWU instances. The practical implication is that the energy efficiency gains of the proposed framework scale with traffic heterogeneity — the greater the variation between peak and average demand, the larger the achievable energy savings.

Toward a unified adaptive UPF. The complementary energy-performance behavior exploited by the proposed framework raises a deeper architectural question: whether a single UPF implementation could intrinsically combine low idle CPU overhead with high-throughput packet processing, reducing the need for heterogeneous instance-level orchestration. A natural candidate is the class of eBPF/XDP-based UPF implementations , which intercept packets prior to socket buffer allocation through kernel-resident XDP hooks, avoiding the continuous poll-mode CPU occupancy of DPDK-based architectures and demonstrating sustained throughput exceeding that of VPP-based solutions under native XDP operation [34]. However, native XDP requires dedicated physical NIC driver support — available on Mellanox ConnectX and Intel i40e class adapters [34] — which is absent in the virtualized network interfaces that constitute the standard I/O model across cloud-native 5G deployments, including OpenStack-based NFV infrastructures and public cloud environments on AWS, GCP, and Azure, where paravirtualized virtio interfaces are the norm. In these environments, eBPF/XDP falls back to generic mode, reintroducing kernel stack traversal overhead and reducing its throughput ceiling to levels comparable to kernel-based UPF implementations. A more hardware-agnostic path exists within VPP itself, where the FD.io framework natively exposes an adaptive RX mode that initializes in interrupt-driven operation and transitions autonomously to polling when packet arrival rates exceed an internal threshold — mirroring the SPGWU–VPP complementarity exploited in this work within a single instance, without N4 session disruption [84]. Cisco's production VPP-based 5G UPF extends this through its Variable Relinquishing of CPU Cycles (VRCC) mechanism, introducing micro-sleep intervals in VPP worker threads inversely proportional to observed traffic

load, engaging CPU power-saving states during low-demand periods with a measured latency overhead of only 0.3–0.4 ms [85]. Critically, both mechanisms remain reactive — driven by instantaneous packet counts rather than anticipated demand — exposing the same fundamental lag that motivates the proactive approach of this work. The natural extension is therefore an ML-driven adaptive VPP-UPF in which the ML prediction model drives RX mode selection proactively ahead of traffic transitions, collapsing heterogeneous multi-instance orchestration into a single self-adapting instance without PDU session disruption and establishing a principled foundation for intra-instance energy optimization in beyond-5G core networks.

5.5 Conclusion

In this chapter, we proposed an energy-aware UPF selection and scaling framework for cloud-native 5G core networks that jointly considers traffic demand, UPF-specific performance characteristics, and power consumption. By combining empirical UPF profiling, CNN-LSTM-based proactive traffic prediction, and energy-aware deployment optimization, the framework dynamically selects both the number and type of UPF instances at each decision interval. Evaluated on a real-world Kubernetes-based testbed using OpenAirInterface and UERANSIM across five deployment scenarios, the framework achieved 51% energy savings compared to the only baseline capable of satisfying equivalent throughput requirements, while maintaining an average packet loss of only 1.5%. These results demonstrate that energy-efficient 5G core operation requires intelligent, adaptive orchestration mechanisms that jointly model traffic dynamics and UPF-specific behavior — rather than static provisioning or reactive scaling alone. As energy efficiency becomes a primary design concern for beyond-5G and 6G systems, the proposed framework provides a practical foundation for sustainable user plane orchestration in softwarized core networks.

Several directions for future work emerge from this study. Incorporating latency as a second optimization objective alongside energy would better address the multi-dimensional SLA requirements of 5G network slicing. A particularly interesting direction is the design of an ML-driven adaptive VPP-UPF, in which the ML prediction model drives VPP RX mode selection proactively ahead of traffic transitions — collapsing heterogeneous multi-instance orchestration into a single self-adapting instance without PDU session disruption. Finally, profiling and integrating additional UPF implementations — including eBPF/XDP-

based solutions where native XDP driver support is available — would further generalize the heterogeneous orchestration approach beyond what we evaluated here.

Chapter 6

Multi-RAT Integration and Evaluation in a Real Beyond-5G Network

6.1 Introduction

The contributions of Chapters 3 through 5 addressed the characterization and orchestration of the User Plane Function under the assumption of a single, homogeneous access technology. In practice, however, the 6G vision of ubiquitous connectivity demands that the core network simultaneously manage traffic arriving from fundamentally different radio access technologies, including cellular 5G NR, unlicensed Wi-Fi, and in future deployments, satellite links. This multi-access paradigm introduces a new dimension of complexity at the UPF: it must now converge heterogeneous radio paths, enforce traffic steering and splitting policies across them, and maintain session continuity under real-world path failures. How this integration is realized at the architectural level, and how different multipath transport protocols perform over real radio channels, are questions that neither the orchestration frameworks of the previous chapters nor the existing simulation-based literature can answer. This chapter addresses that gap directly.

As already defined by the ITU-R IMT-2030 framework [5], 6G introduces six service categories to address the demands of beyond-5G applications, among which Ubiquitous Connectivity envisions a converged fabric that seamlessly integrates terrestrial and non-terrestrial access technologies into a unified network infrastructure. In this paradigm, User Equipment is expected to simultaneously exploit different Radio Access Technologies while the network dynamically steers and splits traffic across them to optimize throughput, latency, and

resilience. This initiative began in the 4G era with the introduction of the LTE-WLAN Aggregation Adaptation Protocol [86] and was progressively enhanced in 5G: Release 15 [17] defined the N3IWF to connect untrusted non-3GPP access networks to the 5G Core, and Release 16 [70] introduced ATSSS, enabling 5G networks to concurrently manage 3GPP and non-3GPP access through a unified core. The ATSSS specification designates MPTCP [87] and MPQUIC [88] as the two transport-layer solutions for exploiting path diversity at the connection level.

However, core-level integration of non-3GPP access introduces several limitations. Since ATSSS operates at the flow level, it lacks direct visibility of RAN conditions such as link quality and RAT load, leading to traffic steering and splitting decisions that are made without knowledge of the underlying radio environment [89]. An alternative approach is to integrate non-3GPP access directly at the RAN level, leveraging the disaggregated 5G RAN architecture. By deploying a Wi-Fi DU within the O-RAN disaggregated architecture [90,91], the CU can jointly manage both 5G NR and Wi-Fi at the PDCP layer, enabling packet-level traffic splitting with unified scheduling and without requiring IPsec tunnels, N3IWF, or transport-layer multipath support at the UE.

Despite the growing interest in non-3GPP integration, real-world evaluations over actual radio environments remain limited, with existing ATSSS studies relying primarily on simulations [92, 93]. This chapter addresses that gap through two distinct real-radio testbed implementations: a core-level integration based on free5GC [26] with an OAI [24] gNB and a Wi-Fi access point connected via the N3IWF, enabling evaluation of both MPTCP and MPQUIC under ATSSS; and a RAN-level integration based on OAI 5GCN and O-RAN, where a custom Wi-Fi-DU is integrated directly as a non-3GPP DU.

The main contributions of this chapter are the following:

- A real-radio deployment and comparison of core-level and RAN-level non-3GPP integration approaches over a real 5G testbed, with no simulation or channel emulation involved.

- The first experimental evaluation of both ATSSS-specified multipath transport protocols — MPTCP and MPQUIC — over real heterogeneous 5G NR and Wi-Fi channels, including live gNB failover experiments.

- A quantitative comparison of core-level multipath (MPTCP, MPQUIC) against RAN-

level packet splitting (Wi-Fi-DU), highlighting the trade-offs between standards compliance, aggregation performance, and path redundancy.

- Identification and resolution of a fundamental incompatibility between Picoquic's Recent ACKnowledgement (RACK)-based loss detection and the N3IWF tunneled path, increasing single-path Wi-Fi QUIC throughput from 86 Mbps to 193 Mbps.

The rest of this chapter is organized as follows. Section 6.2 reviews related work. Section 6.3 describes the system architecture. Section 6.4 presents the experimental results. Finally, Section 6.5 concludes and outlines future directions

6.2 Related Work

A lot of research has been conducted on how non-3GPP access technologies can be integrated into the cellular networks. This research has three basic axes, namely the integration of non-3GPP access technologies at both the core and RAN levels, and the multipath transport protocols along with the corresponding scheduling decisions.

The first initiative of Wi-Fi integration at the cellular networks happened at 3GPP Release 13 (LTE/4G), which introduced LTE-WLAN Aggregation (LWA) at the PDCP layer and LTE-WLAN Radio Level Integration with IPsec Tunnel (LWIP) at the IP layer. Several works explored this direction. Authors in [94] developed a 4G LWIP-based prototype with a COTS UE, evaluating the link aggregation strategies under varying Wi-Fi loads. On the contrary, paper [95] implemented a PDCP-level LWA system integration, demonstrating the throughput gains over the standalone LTE. In the 5G era, authors in [96] discuss the potential and challenges of the 5G Cloud-based RAN (C-RAN) to support multiple RATs. Based on the aforementioned, our prior works [90] and [91] extended this direction into 5G/O-RAN networks, integrating the Wi-Fi as a non-3GPP DU managed by the CU with xApp-driven traffic steering on the Near-RT RIC.

From the 3GPP scope, the non-3GPP integration has moved to the CN part in the 5G era. As already mentioned, Release 15 extended the prior LWIP functionality and introduced the N3IWF, and Release 16 defined the ATSSS framework using MPTCP or MPQUIC (Release 18) as transport-layer enablers. The first implementation of traffic steering in the CN was reported in [97], in which the integration of MPTCP-based traffic steering inside the User

Plane Function (UPF) was proposed. Following this, authors in [98] presented the first real-world testbed implementation, deploying the N3IWF and 5G CN to support multi-access connectivity via the ATSSS, focusing on MPTCP only. Different works present the efficiency of the N3IWF and the ATSSS scheduling by deploying networks with multiple RATs such as 5G NR, Wi-Fi, and LiFi as shown in [99] and [100]. All of the prior works utilize MPTCP as the multipath ATSSS layer and evaluate its performance.

None of the existing works aims to integrate MPQUIC as the multipath ATSSS layer. An interesting research comparing the two multipath protocols was conducted in [101], where both MPQUIC and MPTCP were tested over real UEs, Wi-Fi and LTE networks, showing that both protocols exhibit similar performance under most traffic patterns. The authors report that MPTCP achieves significantly higher throughput than MPQUIC in high-bandwidth scenarios.

To the best of our knowledge, no prior works have experimentally compared core-level ATSSS (using both MPTCP and MPQUIC) against RAN-level packet steering (Wi-Fi-DU) over real over-the-air 5G NR and Wi-Fi channels. Our work bridges this gap through a unified real-radio evaluation of all three approaches under live heterogeneous RATs.

6.3 System Architecture

6.3.1 Core-level Integration: N3IWF and ATSSS

The core-level integration architecture is illustrated in Fig. 6.1. The 5G core is realised with free5GC, hosting the standard control-plane functions Access and Mobility Function (AMF), Session Management Function (SMF), Authentication Server Function (AUSF), Unified Data Management (UDM), Unified Data Repository (UDR) alongside a UPF that anchors both downlink paths. Two distinct radio flows reach the UPF through separate interfaces. The 3GPP access path follows the standard NR user-plane flow, where the gNB forwards user-plane traffic to the UPF via the N3 interface and vice-versa. The non-3GPP Wi-Fi path is handled through the N3IWF, the 3GPP-defined gateway for untrusted non-3GPP access defined in Release 15, which terminates an IKEv2/IPsec tunnel from the UE and connects to the UPF via the N3 interface. Through this tunnel, the N3IWF handles UE registration by translating N3IWUE's control-plane messages into Non-Access Stratum (NAS) signaling relayed to the AMF, enabling the UE to register with the 5GCN and establish a Protocol Data Unit

(PDU) session with the UPF. 5G NR and Wi-Fi paths converge at the UPF, which terminates two separate PDU sessions, effectively creating two distinct slices toward the same UPF instance. The UPF exposes the combined path diversity to the transport layer through the ATSSS framework, with traffic steering and splitting rules provisioned by the SMF. To enable multipath transport, an MPTCP proxy server and an MPQUIC server are deployed within the UPF, which receives downlink traffic from the Data Network (DN) and splits it across the two available paths toward the UE, acting as the network-side termination points for MPTCP and MPQUIC respectively.

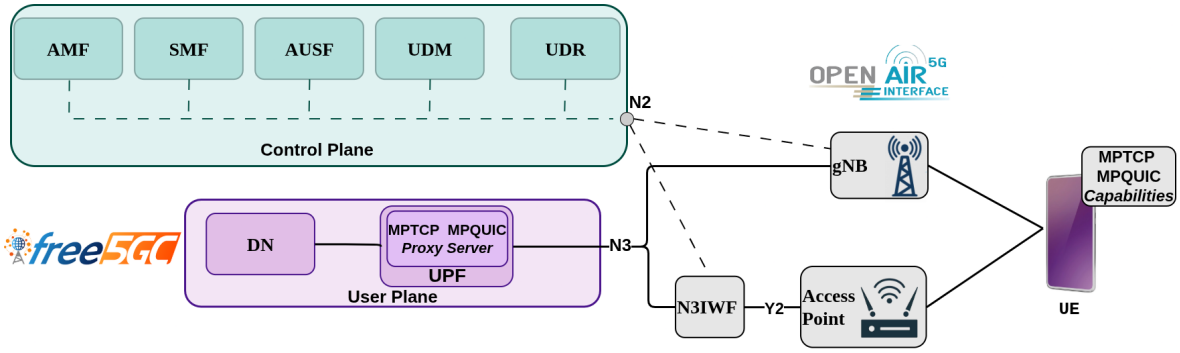


Figure 6.1: ATSSS System Architecture utilizing different multi-path transport protocols

6.3.2 RAN-level Integration: Wi-Fi-DU

As an alternative to core-level non-3GPP integration, we consider integrating Wi-Fi directly into the MAC layer of the 5G NR protocol stack within the DU of the disaggregated 5G RAN, following the approach proposed in [91]. In the 3GPP Option 2 split [102], the CU is responsible for the SDAP and PDCP layers, while the DU handles the RLC, MAC and PHY. In our architecture, the Wi-Fi DU replaces the NR MAC and PHY layers with the IEEE 802.11 MAC and PHY, which allows the CU to manage both RATs in a unified way, without the need for N3IWF, IPsec tunnels or multipath transport-layer support at the UE side. When a UE registers with the network, the Wi-Fi DU maps its 5G Radio Network Temporary Identifier (RNTI) to Wi-Fi MAC address based on predefined rules, enabling correct identification and routing of packets toward the UE across both RATs. In contrast to our prior implementation in [91] that employed a custom F1-over-IP protocol over TCP, in this work the F1 interface is operating over a GPRS Tunnelling Protocol – User plane (GTP-U) tunnel, aligning the Wi-Fi DU integration with the same transport used by the 5G NR DU. In this

architecture, the control-plane of the UE is managed exclusively through the 5G link, with the Wi-Fi DU serving only as a user-plane path.

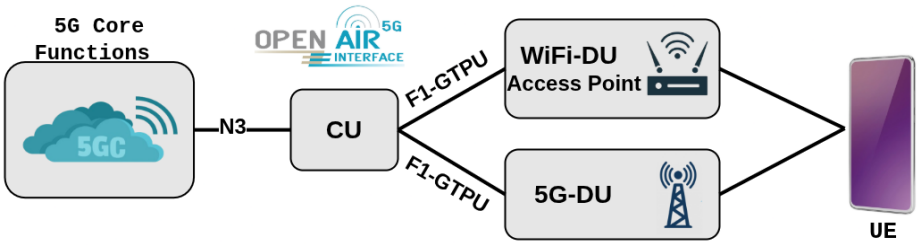


Figure 6.2: RAN-Level integration of Wi-Fi into 5G

Regarding the downlink path, the PDCP packets are forwarded from the CU to the Wi-Fi DU, which strips the PDCP headers, maps the RNTI to the corresponding Wi-Fi MAC address using the predefined mapping table, encapsulates the packet in the 802.11 MAC header, and transmits the frame over the air. For the uplink, the inverse process is performed at the Wi-Fi DU level, with the PDCP sequence number management handled centrally at the CU component, ensuring correct packet ordering regardless of the originating RAT. The traffic steering is implemented in a scheduler that operates between the PDCP and RLC layers at the CU, which dynamically distributes downlink packets across the two DUs. In our experiments, we deploy an xApp on the O-RAN Near-RT RAN Intelligent Controller (RIC) that enforces a 5G-first policy: traffic is initially forwarded to the 5G DU up to its known maximum capacity, and any excess demand is offloaded to the Wi-Fi DU.

6.4 Experimental Setup & Results

The experiments are conducted on the NITOS testbed, part of the SLICES-RI research infrastructure, utilizing four dedicated nodes to deploy the Beyond 5G networks presented in Section 6.3. The *Core Server* node hosts the 5GCN, deploying the free5GC CN for the core-level integration, and the OAI 5G CN for the RAN-level integration. The *RAN Server* node is responsible for the 5G NR radio components, deploying a monolithic gNB in the core-level integration and the disaggregated CU alongside the 5G NR DU in the RAN-level integration. The *WLAN Server* hosts the Wi-Fi AP and, depending on the integration approach, the N3IWF or the Wi-Fi-DU. Regarding the radio hardware, the 5G NR over-the-air channel is established through a USRP B210 on the *RAN Server*, while the Wi-Fi link operates over Wi-Fi 5 (IEEE 802.11ac) utilizing the Atheros QCA988x NIC with the ath10k_pci driver

Table 6.1: SLICES-RI Experimental Testbed Hardware Configuration

Node	Role	Hardware (CPU, RAM)	Radio
Core Server	free5GC & OAI Core	i7-8700, 16 GB	–
RAN Server	gNB & CU, 5G DU	i7-8700, 16 GB	USRP B210, 1x1 SISO
WLAN Server	AP, N3IWF & AP, Wi-Fi-DU	i7-8700, 16 GB	Atheros QCA988x 3x3 MIMO
UE	Multi-RAT UE	i7-3770, 8 GB	Quectel RM510Q-GL + Atheros QCA988x

and hostapd on *WLAN Server*. The multi-homed UE is equipped with the Quectel RM510Q-GL 5G dongle and an additional Atheros QCA988x operating as a standard 802.11ac station, providing simultaneous connectivity over both RATs Table 6.1 summarises the hardware and software components of the testbed.

The real-radio evaluation was broken down to five experimental scenarios: single-path Wi-Fi and 5G NR baselines, MPTCP and MPQUIC core-level aggregation, each with a live gNB failover experiment, and RAN-level packet splitting via the Wi-Fi-DU. Although the figures depict 40-second intervals for readability, all experiments were conducted over extended durations and repeated 10 times. The figures present measurements from the run that align closely with the average across all 10 runs. The consistency observed across both extended durations and repeated trials confirms that the reported results are stable and not artifacts of short measurement windows or run-to-run variability. Throughput across all scenarios is measured using iPerf3, with the server instance running on the DN and the client on the UE, reporting per-second throughput over the duration of each experiment.

6.4.1 Single-Path Baselines: Wi-Fi via N3IWF and 5G NR via OAI gNB

We first establish the single-path capacity of each RAT independently. In the Wi-Fi baseline, the UE connects exclusively through the N3IWF to the free5GC core, with the Wi-Fi path exposed to the UE as the `gretun-id-2-1` Generic Routing Encapsulation (GRE) tunnel interface established over the IKEv2/IPsec session. As shown in Figure 6.3, TCP downlink throughput stabilises at a mean of **214 Mbps** while UDP achieves **461 Mbps**, reflecting the effective capacity ceiling of the real Wi-Fi radio link as seen end-to-end through the 5G core. The gap between TCP and UDP is consistent with the overhead introduced by the TCP congestion control mechanisms operating through the IPsec tunnel at the N3IWF. In the 5G NR baseline, the UE connects exclusively through the OAI gNB over the real NR air interface,

with the 5G dongle exposed as the `wwan0` network interface. As shown in Figure 6.4, TCP achieves a mean of **60 Mbps** and UDP **124 Mbps**, with greater short-term variability compared to the Wi-Fi path due to the sensitivity of the NR air interface to scheduling decisions and real-world channel dynamics. The resulting path asymmetry, Wi-Fi at 214 Mbps versus 5G NR at 60 Mbps (TCP), is an inherent characteristic of the real radio environment and has direct implications for multipath aggregation efficiency, as discussed in the following subsections.

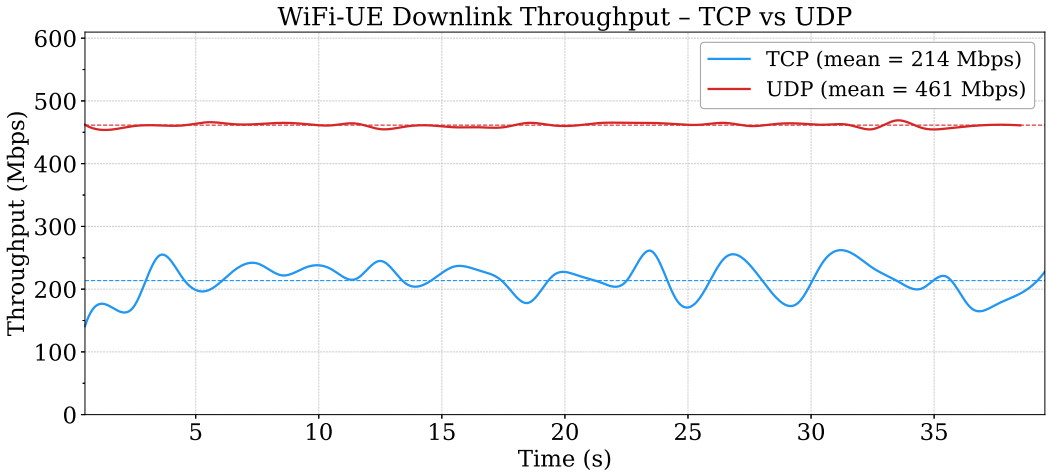


Figure 6.3: N3IWUE DL Throughput over Time (UDP vs TCP)

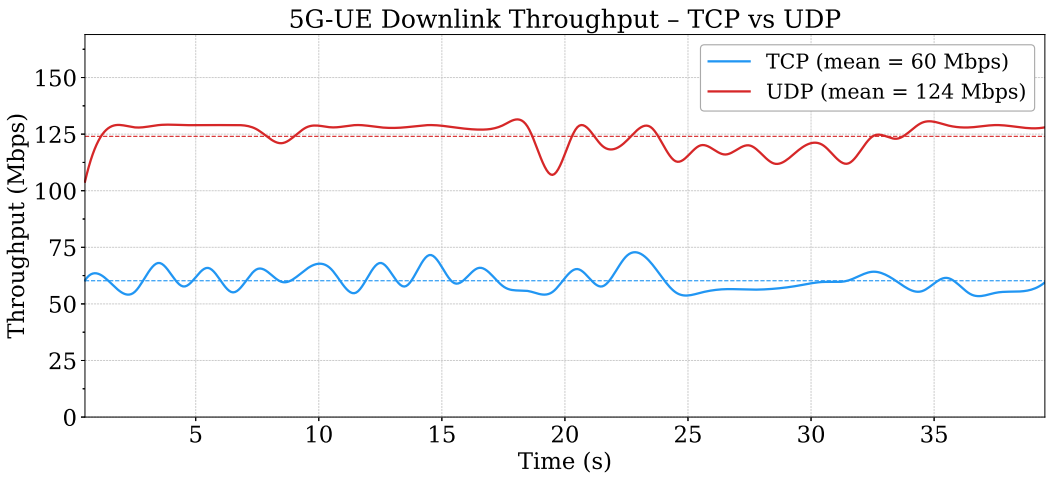


Figure 6.4: 5G-NR UE DL Throughput over Time (UDP vs TCP)

6.4.2 MPTCP

The MPTCP kernel module is installed on both the UE and the MPTCP proxy server deployed within the UPF on the Core Server. The two UE interfaces `wwan0` and `gretun-id-2-1` are configured as separate MPTCP subflows, with downlink traffic originating from the DN, passed to the UPF where the MPTCP proxy server splits it across both paths, and delivered to the UE simultaneously. The UE establishes two subflows over the air: one via the real Wi-Fi link through the N3IWF, and one via the live 5G NR interface. Fig. 6.5 shows the aggregated downlink throughput, which achieves a mean of **267 Mbps**. While MPTCP successfully aggregates both real radio paths, the achieved throughput of **267 Mbps** closely approaches the arithmetic sum of the individual TCP path capacities, indicating that MPTCP exploits nearly all of the available combined capacity under these conditions. Residual losses arise from subflow coupling effects, out-of-order packet delivery under real channel conditions, and receiver-side reordering overhead driven by the bandwidth asymmetry between the two radio interfaces. The oscillations visible in Fig. 6.5 reflect the dynamic interaction between the two subflows' congestion windows as they react to the real radio channel conditions.

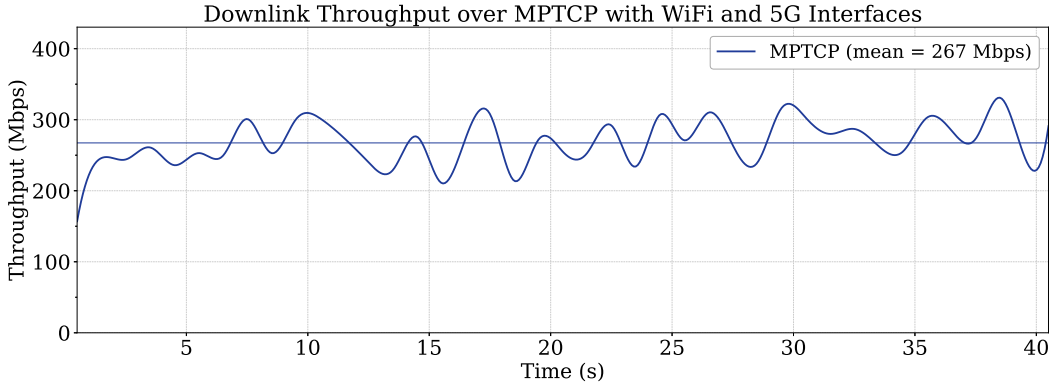


Figure 6.5: MPTCP DL Aggregated Throughput (Wi-Fi & 5G)

To evaluate resilience under real radio failure conditions, Fig. 6.6 presents an experiment in which the live 5G NR gNB is deliberately crashed at approximately t=20 s, causing a real radio path failure. Prior to the failure, the aggregated throughput operates at a mean of **275 Mbps**. Following the crash, MPTCP detects the path failure and reroutes all traffic through the surviving Wi-Fi subflow, recovering to a mean of **216 Mbps** within approximately 2 seconds as the stack retransmits in-flight data and the congestion window reopens.

The post-failover throughput of 216 Mbps closely matches the single-path Wi-Fi baseline, confirming clean path isolation and minimal residual overhead once the failed subflow is removed.

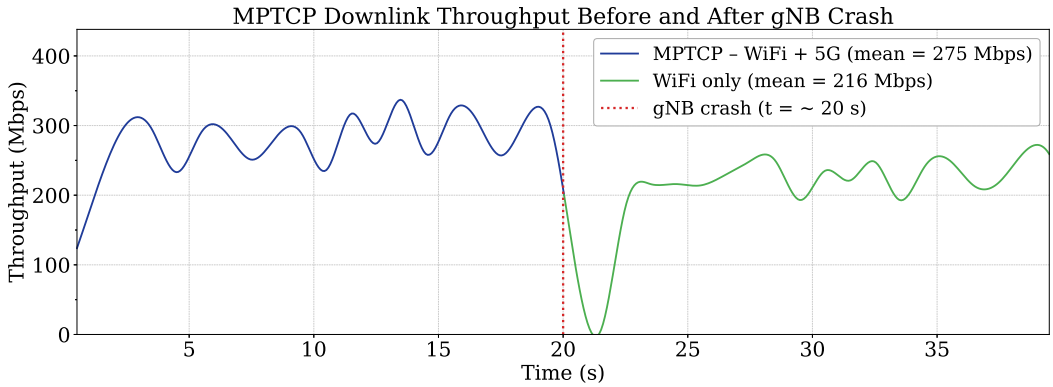


Figure 6.6: MPTCP DL Throughput Before and After gNB Crash

6.4.3 MPQUIC

In this scenario, we evaluate Multipath QUIC using the Picoquic implementation by Private Octopus [103], grounded in the formal IETF QUIC specification [104] and extended with multipath support [88]. The UE is configured to run the QUIC capabilities with multipath extensions enabled, with `wwan0` and `gretun-id-2-1` mapped to separate QUIC paths. Downlink traffic originates from the DN, is passed to the UPF where the MPQUIC server schedules packets across both paths, and is delivered to the UE simultaneously over the live Wi-Fi link through the N3IWF and the 5G NR interface via the gNB. A key implementation challenge arose from the interaction between Picoquic's RACK-based loss detection and the N3IWF tunneled path: GRE over IPsec encapsulation at the N3IWF introduces per-packet processing jitter on the ACK return path, causing ACKs to arrive at the server in bursts with large inter-arrival gaps, triggering Picoquic's `delta_seq` threshold for packets still in flight. Qlog analysis confirmed that over 85% of retransmissions were spurious, repeatedly collapsing the congestion window—a problem absent on the 3GPP path due to the gNB's deterministic scheduling. We addressed this by increasing the `delta_seq` parameter, and raising the default flow control window resulting in an increase of the single-path Wi-Fi QUIC throughput from **86 Mbps** to **193 Mbps**.

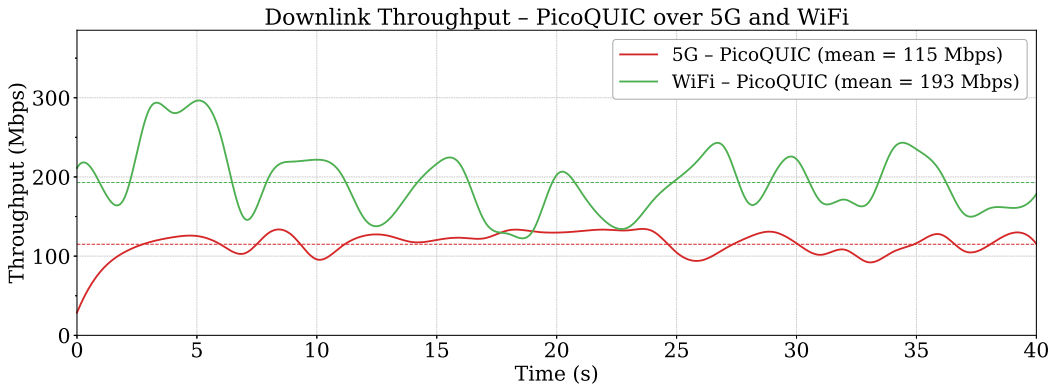


Figure 6.7: PicoQUIC Single-Path DL Throughput per RAT

Fig. 6.7 shows the single-path QUIC throughput for each interface independently, yielding a mean of **193 Mbps** on the Wi-Fi path and **115 Mbps** on the 5G path, establishing the per-path QUIC baseline prior to multipath activation. Fig. 6.8 shows the aggregated downlink throughput, which achieves a mean of **317 Mbps**, exceeding both the MPTCP aggregate and the arithmetic sum of the individual TCP baselines. This aggregation result is consistent with MPQUIC's connection-level path scheduling, which distributes load across paths more aggressively than MPTCP's coupled subflow congestion control under asymmetric channel conditions. The individual path measurements yield a mean of 115 Mbps on the 5G path and 193 Mbps on the Wi-Fi path, confirming that both radio interfaces are actively utilised.

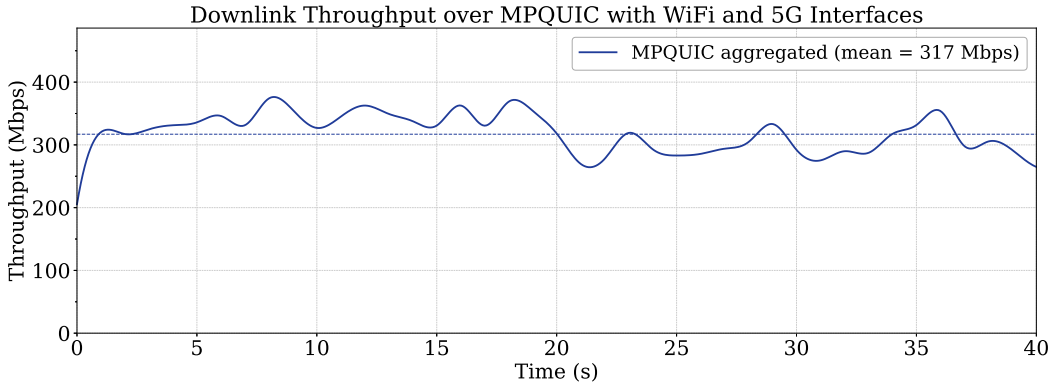


Figure 6.8: MPQUIC DL Aggregated Throughput (Wi-Fi & 5G)

Resilience under failure is assessed in Fig. 6.9, where the gNB is crashed at t=20 s. Prior to the failure, the aggregated mean throughput is approx. **308 Mbps**. After the crash, MPQUIC

migrates all traffic to the surviving Wi-Fi path, recovering to a mean of **181 Mbps**. In contrast to MPTCP, which exhibited a recovery period of approximately 2 seconds following the gNB failure, MPQUIC shows no measurable throughput drop upon path loss. Since both paths share a single QUIC connection, traffic is immediately redirected to the surviving Wi-Fi path without subflow teardown or retransmission timeout, as the Wi-Fi path remains continuously active and its congestion state is preserved throughout the failover event.

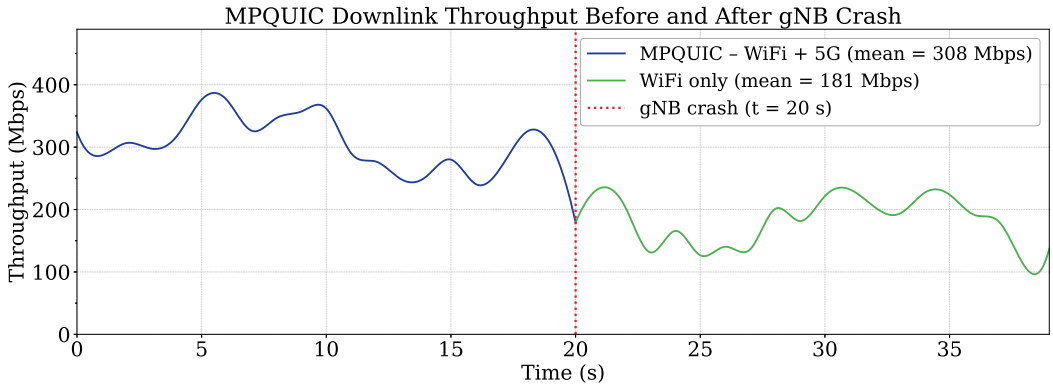


Figure 6.9: MPQUIC DL Throughput Before and After gNB Crash

6.4.4 Wi-Fi-DU

Finally, we evaluate the RAN-level integration approach described in Section 6.3.2, where Wi-Fi is treated as a non-3GPP DU managed by the CU at the PDCP layer, bypassing the N3IWF and IPsec encapsulation entirely. The experimental deployment follows the same distributed hardware setup as the previous architecture. However, in this case, the RAN is disaggregated using the F1-split (CU/DU architecture). The CU and the 5G NR DU are co-located on the RAN Server, while the Wi-Fi DU is deployed on the WLAN Server alongside the Wi-Fi Access Point. Fig. 6.10 shows the single-path Wi-Fi DU UDP throughput, which achieves a mean of **487 Mbps**, reflecting the raw capacity of the Wi-Fi radio link at the PDCP layer without the overhead present in the core-level integration scenarios. Fig. 6.11 presents the aggregated downlink throughput when the CU scheduler simultaneously exploits both the 5G NR DU and the Wi-Fi DU under the 5G-first xApp policy. The combined throughput achieves a mean of 606 Mbps, demonstrating that RAN-level steering enables near-complete aggregation of the available path capacities without imposing any transport-layer constraints

on the UE. However, this approach is not compliant with 3GPP specifications, as it repurposes the DU interface in a non-standards compliant manner. In addition, the architecture introduces a single point of failure at the base station, as Wi-Fi DU is one of the disaggregated components of the 5G BS, meaning a failure of the CU results in complete loss of connectivity on both paths, while a failure at the DU implies the de-registration of the UE at the 5G NR-level, eliminating the redundancy that ATSSS core-level integration is specifically designed to provide.

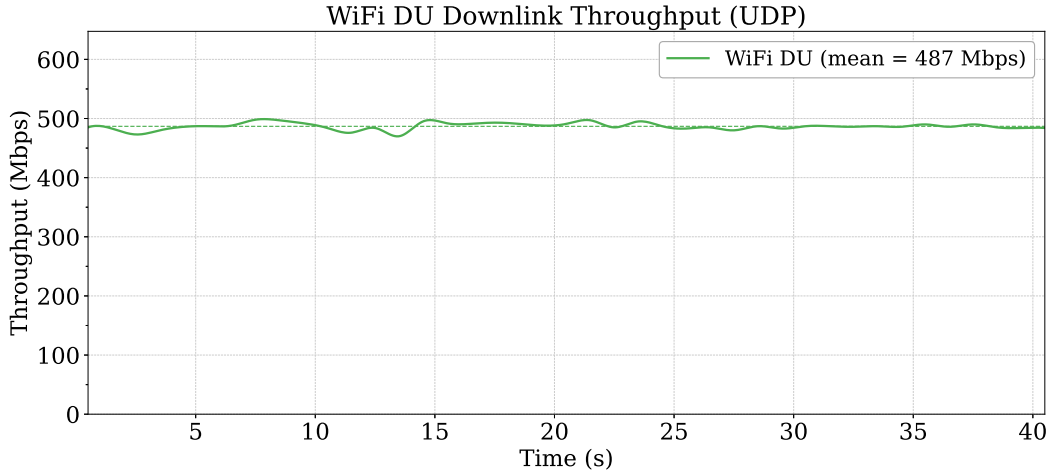


Figure 6.10: Wi-Fi-DU Single-Path Downlink Throughput (UDP)

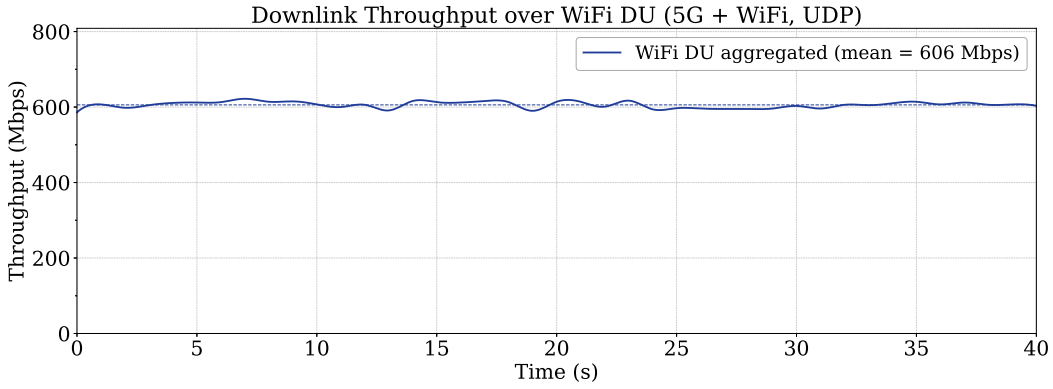


Figure 6.11: Wi-Fi-DU Aggregated Downlink Throughput (5G NR + Wi-Fi, UDP)

6.5 Conclusions

In this chapter, we presented a real-world evaluation of multipath transport protocols over a 5G ATSSS testbed, comparing the core-level integration via the N3IWF against the RAN-level packet splitting via a custom Wi-Fi-DU, entirely over real over-the-air 5G NR and Wi-Fi channels. MPTCP achieved near-optimal aggregation efficiency of 267 Mbps with robust failover recovery within 2 seconds. MPQUIC, after resolving a fundamental incompatibility between Picoquic's RACK-based loss detection and the N3IWF tunneled path, achieved an aggregation of 317 Mbps with instantaneous path migration upon gNB failure. The RAN-level Wi-Fi-DU achieved the highest throughput of 606 Mbps, but at the cost of 3GPP non-compliance and loss of path redundancy. These results highlight a fundamental trade-off between standards compliance, aggregation performance, and resilience. As future work, we plan to integrate satellite communications as a third non-3GPP access type via emulation, and to incorporate AI-based traffic steering into the ATSSS framework by integrating the Network Data Analytics Function (NWDAF), replacing the static SMF-provisioned steering rules with data-driven policies that adapt to the time-varying conditions of real multi-RAT deployments.

Chapter 7

LLM-Based Automation of Network Experimentation

7.1 Introduction

The contributions presented in Chapters 3 through 6 focus on different aspects of 5G Core and beyond-5G system design, ranging from performance evaluation and dynamic resource management to advanced architectural and service-level extensions. A common thread across these works is the reliance on real experimental infrastructure — the SLICES-RI testbeds — to validate proposed solutions under realistic conditions. However, accessing and operating such large-scale research infrastructure is itself a non-trivial challenge, particularly for researchers who are not yet familiar with the methodologies, resource configurations, and toolchains required to get started. This chapter addresses that challenge directly, shifting focus from core network and architectural contributions to the usability of the experimentation platform itself.

Research Infrastructures (RIs) have emerged as key components for validating research outcomes in domain sciences. Such instruments need to keep up with the most recent advances in technology, offering bleeding-edge resources to researchers, and integrating the most advanced features that accelerate the materialization of new ideas to research outcomes. As such, SLICES [22], [105] is one of the first RIs defined as a scientific instrument in the Digital Infrastructures (DI) domain, able to offer cutting-edge resources for experimentation to researchers of the field, while providing them with structured experiment lifecycle management from the design phase, to the deployment, evaluation, archiving and post-experiment

analysis. The resources offered by SLICES are defined based on thought experiments, spanning a wide range of programmable infrastructure, from Open-RAN and beyond 5G networks to edge and High-Performance Computing (HPC) and IoT.

However, retrieving documentation needed for bootstrapping the experimentation process, as well as identifying the appropriate resources in such complex environments can be a very tedious task, especially for new users of the RI that are not yet familiar with the specific methodologies that need to be followed in order to get started with the experimentation process. To this aim, tools that increase the user-friendly interface for on-boarding users with the facilities are needed to increase the attractiveness and overall usability of the RI. Generative AI and Large Language Models (LLMs) have emerged as tools that can simplify to a large extent the creation of new human interpretable content based on existing knowledge that they have access to. Based on their training inputs, such models are able to create content by combining knowledge from multiple sources, including even automatic code generation, that can be used in the context of user support for RIs. In this chapter, we present slAIces, an LLM-based chatbot for lowering the barrier for new users experimenting with the SLICES-RI. slAIces uses the well-established Generative Pre-trained Transformer (GPT) 4 model for generating output for accessing the infrastructure, including selecting the most appropriate nodes depending on the type of experiment that the users need to conduct, while receives as input the documentation of each SLICES-RI testbed (incl. hardware capabilities, supported types of experiments, etc.). This chapter presents the design choices, lessons learned and recommendations from the development and the model training process, and a framework evaluation based on the inputs provided from the online documentation sites of the different SLICES-RI nodes.

The rest of this chapter is organized as follows. Section 7.2 reviews related work on LLM-based domain-specific assistants and RAG systems. Section 7.3 presents the overall architecture and LLM configuration of slAIces. Section 7.4 evaluates the framework under different training inputs, highlighting the importance of data augmentation and fine-tuning. Section 7.5 presents the extension of slAIces toward fully automated experiment execution, including the multi-pipeline architecture and the deployment Controller. Finally, Section 7.6 concludes and outlines future directions.

7.2 Related Work

A researcher, with no familiarity with the SLICES infrastructure, can benefit from the capabilities provided by the slAIces chatbot. Indeed, it can significantly reduce the time it takes to learn the experimental platform and help the experimenter unlock the full potential of SLICES-RI much more rapidly than traditional methods would allow.

When designing an LLM-based chatbot, it is important to consider how the LLM model will acquire special-purpose knowledge. Authors of paper [106], present insights on how LLMs can be used to integrate and apply domain-specific knowledge with the usage of in-context learning (ICL), providing the techniques, and challenges of ICL. One widespread way of integrating an external source of knowledge into LLM, based on ICL and structuring the retrieval information part, is through Retrieval-Augmented Generation (RAG) systems. Paper [107] provides a brief analysis of the architecture of the different components and the advancements of RAG systems, analyzing different sophisticated models such as Naive RAG, Advanced RAG, and Modular RAG. It also proposes methods for evaluating a RAG system, many of which were utilized to evaluate our proposed system in section 7.4. Paper [108], explores a different way to construct domain-specific LLMs via Fine Tuning (FT). The authors implement a comprehensive approach for the FT process, analyzing various aspects such as dataset selection, preprocessing, model choice, particularly focusing building LLMs for the financial field.

Authors of [109] explore applying an LLM model using the domain-specific knowledge of surface engineering, having as a dataset a list of scientific papers of the field, using the indexing method. This work highlights the importance of integrating advanced AI tools with expert knowledge, to enhance accuracy and relevance in a specialized field (such as the surface-engineering) oriented queries. Exploring the integration of LLMs into specialized domains, it is worth mentioning the work [110], which provides LLM-powered agent assistance into the video editing workflow, aiming to simplify complex editing tasks such as automatically generating descriptions for footage, enabling the LLM to execute editing tasks based on user objectives and others. The survey in [111] explores the use of LLMs in developing autonomous agents, noting a departure from previous research focused on agents with limited knowledge in isolated environments. It presents a detailed examination of the architecture, diverse applications, methodologies for assessment and challenges of LLM-based autonomous agents, thus providing very important details on the design and implementation of slAIces.

The paper [112] delves into the evolution and challenges faced by Virtual Intelligent Assistants (VIAs) such as Amazon Alexa and Google Assistant, which have gained prevalence in recent years. It explores new research directions focused on understanding multi-modal contexts, and personalizing assistant services to provide insights for researchers and practitioners while emphasizing the pivotal role of LLMs in enhancing these strategies.

In this chapter, we pioneer the development of an advanced RAG system using the special-purpose data sourced from the SLICES RI documentation, enriched with a strategic usage of GPT4 and combining both ICL and FT methods. This system aims to provide an improved user experience for experimenters of the SLICES RI infrastructure to reduce the learning curve for new users, increase the research productivity and assisting in experimental design, planning, and automated experimental deployment.

7.3 Architecture & LLM Configuration

To design SlAIces, we employed a structured framework development strategy, which consists of four key stages: data preparation, designing and implementing a RAG system, LLM fine-tuning, and user querying via an appropriate environment. The overall integrated architecture can be observed in Figure 7.1. The initial stage involves acquiring SLICES data from diverse websites using the web scraper described in subsection 7.3.1. Subsequently, raw data is appropriately enhanced by enriching both the textual content and the code content. This enhancement process involves leveraging the online GPT-4 model, which is manually utilized to upload our data for further enrichment. In addition, our domain expertise is leveraged to supplement this process. The RAG system consists of a pipeline of different functions that are responsible: 1) for breaking and storing the data in suitable data structures (indexing), 2) retrieving the most similar information to the user query from the data (retrieval), 3) constructing a prompt that contains the retrieved information as long as the user query (prompting), 4) using an LLM to get the response of the user query (LLM-Output). Following these, we proceed with deploying the chatbot service onto one of our SLICES servers. In order to provide a user-friendly interface, we have established a link between the SLICES server (housing the deployed chatbot) and a Discord server. This integration allows users to interact with the chatbot directly through the Discord server, simplifying the process and providing a more intuitive interface for the end user. The server is also integrated to the online

portal, to provide a conversational experience to the users that want to conduct experiments within the SLICES infrastructure. The basic concepts and functionalities of our system are further explained in the subsections below.

7.3.1 Data acquisition

The data that will feed our model are sourced primarily from the official website of SLICES-RI [113] as well as the links within its contents that point to educational and training material, such as tutorials for students and researchers to become familiar with various frameworks and technologies that each integrated testbed of SLICES-RI is equipped.

To be able to automatically retrieve the required information from the aforementioned documentation websites, a web scraper was developed, designed to operate in two stages. In the first stage, all the links and URLs inside the documentation pages are tracked recursively from the root website, with the help of the *urllib* library. After gathering all the referral URLs on the visited websites, duplicate URLs are removed to create a unique set of URLs. Subsequently, a manual review and evaluation of the content of the web pages is performed, to evaluate its relevance and value to be included in the dataset. In the second stage of the web scraper, the web page data is retrieved and converted from HTML format to text through the *Beautiful Soup* library. Given the variation in website designs among the SLICES-RI testbeds, the code was adapted according to each website-specific design to extract the information correctly. At this stage, the data from each website is stored in distinct files, allowing a modular approach to removing/adding new information to the model.

7.3.2 Data augmentation

This step is mandatory to make our system efficient. Although the retrieved data from the documentation sites are interpretable by humans, they can be less beneficial for LLMs. LLMs, in contrast to humans, struggle to understand complexity and ambiguity since they lack real-world understanding.

The learning process for humans versus the one for LLMs significantly differs, especially in tutorials. A human can approach tutorials already having a basic understanding of the technology and the ontology of the various entities and components used in them. Foundational knowledge makes it easier for humans to make new connections among the different parts of information as long as they can easily grasp new concepts. Tutorials mainly point to hands-on

experience, not providing depth in theory. In addition, the reader of the tutorials can refer to other sources besides the tutorial and its own references. However, none of these applies to LLMs, since LLMs lack ontological understanding. They realize text based on patterns observed using statistical correlations in their training data and not using relationships to convey meaning. LLMs also suffer from Referencing Limitations since their knowledge is restricted only to their training and In-context Learning (ICL) dataset, without external post-training resources.

Our raw dataset, provided by the documentations, presents gaps and deficiencies for appropriate LLM training, with respect to clarity, depth, or specificity of different information. To address such vulnerabilities, we enriched our dataset with comprehensive vocabulary as well as with an extensive and precise description of the theory (e.g. component's architecture, terminology, etc.). We employed the popular ChatGPT AI system [114] based on GPT4 model [115] to perform this enhancement. Following that, we restructured the generated text, into smaller paragraphs with titles emphasizing the context of their body. We validated rigorously, and thoroughly checked the data enhancement to ensure that we did not add vague information and pollute our data with noise, since ChatGPT can make mistakes.

7.3.3 In-context learning & Fine Tuning

A diverse amount of publicly available internet text data, such as websites, Wikipedia, Common Crawl, articles, as well as scientific papers, have been used to train GPT. This extensive training leads LLMs to identify patterns observed using statistical correlations, and identify grammar, syntax, and semantics, achieving a significant amount of general knowledge. Although GPT has tremendous performance in general-purpose knowledge, it looks very vulnerable to specific domains, such as providing assistance to SLICES RI infrastructure, where domain-specific knowledge and contextual understanding are crucial. For an LLM model to be updated with special-purpose data, there are two popular methods: in-context learning via a RAG system and fine-tuning.

7.3.3.1 In-context learning via a RAG system

The first approach, in-context learning, feeds the LLM with special-purpose data and by providing extra information in the prompt, without any parameter updates or fine-tuning. This method utilizes a RAG system to insert context into the input prompt, allowing the LLM to

generate responses by leveraging its reasoning capabilities. Our RAG implementation follows the Naive RAG architecture as described in [107] and has three necessary axes: Data Ingesting, data indexing and data retrieving.

Data ingestion refers to how the contents of files in different formats (CSV, JSON, XML, PDFs, etc.) can be converted into data structures. Subsequently, Data Indexing is responsible for organizing, storing, and efficiently accessing the converted data. Several proposed index schemes can be used in different use-case scenarios. A node in the data Indexing world, represents a chunk of source data from a document, regardless of its type. There are multiple index schemes available, such as List Index, Vector Store Index, Tree Index and the keyword index. The data retriever is responsible to identify the most similar data nodes of the indexed dataset to include its content to the prompt. Our system utilized the open-source library LlamaIndex [116] to manage the data ingestion, indexing and retrieving procedures of slAIces implementation.

7.3.3.2 Fine Tuning (FT)

Fine-tuning (FT) is another strategy to integrate specific-domain knowledge into an LLM than ICL. The strategy mainly focuses on training a pre-trained model using domain-specific datasets to incorporate data from specialized fields. Using a pre-trained model offers considerable advantages compared to an LLM that is exclusively trained with our private data. One of the most important ones is efficiency, since training the model from scratch requires significant computational resources, time, and an enormous dataset to learn language understanding. FT changes the model's weight through additional training, improving performance on domain-specific tasks. During the FT process, great attention was given to the parameters that will be modified so that they maintain the performance of our model and not to become overfit.

The strategy of FT is not as straightforward as that of In-context learning. Since the FT process is designed to adjust the model's parameters using examples of input-output pairs, we have to change the format in which the data are fed to the model. Instead of giving the information in raw text as in Section 7.3.3.1, all information intended for training should be converted into a query-answer format to be used as a prompt. Following this, our dataset should be structured in a prompt-completion format as described in the Fine-tuning Guide [117]. We decided to fine tune the model with a portion of the dataset that LLM had difficulty

responding when using only ICL. OpenAI did not support fine-tuning for GPT-4 at that time, since it was at an experimental stage at the time of this study (2024). For this work, FT used GPT-3.5-turbo-0125 limiting the model's capabilities compared it to the GPT4.

ICL and FT are two complementary techniques that can enhance a model's capabilities on distinct levels without being mutually exclusive. Our chatbot assistant is of considerable complexity, primarily because it must deliver not just theoretical knowledge but also detailed instructions and advanced information across a multitude of configurations, console outputs, commands, architectural concepts and many more. To deal with this issue, it is critical to integrate both FT and ICL into our development process. With ICL we are capable of providing the model with all the information, including even the slightest detail. On the other hand, FT is inevitable, since it helps the model to deeply understand our specific dataset, enabling it to interpret the meanings of different data types and follow advanced procedures.

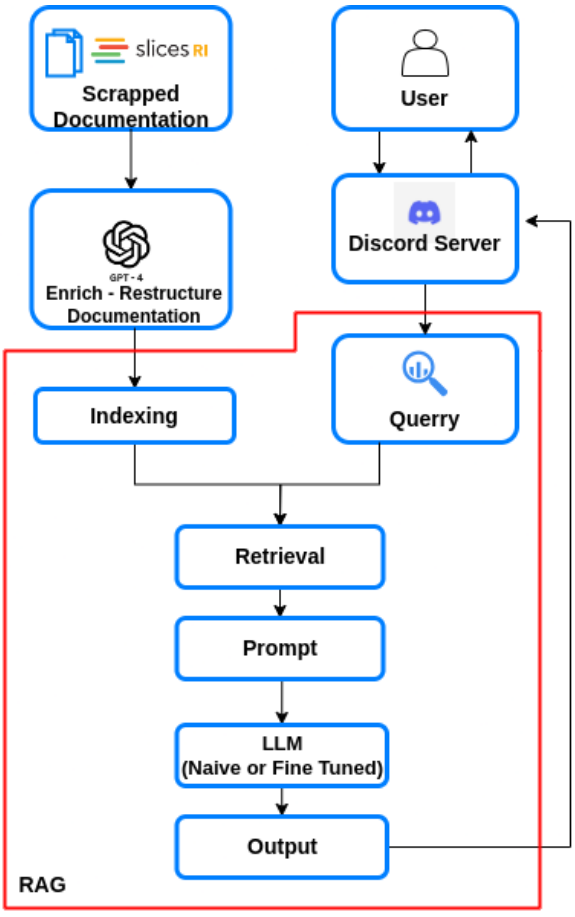


Figure 7.1: System Architecture

7.4 Experimental Evaluation

In Section 7.4.1, we describe our evaluation methodology between the generated dataset after the enrichment process (see Section 7.3.2), and the raw one. Then, in Section 7.4.2 we examine the system performance by leveraging multiple choice and open-ended questions using the original and the enhanced data as long as the fine-tuned LLM (Section 7.4.3).

7.4.1 Datasets

To compare the enriched dataset against the raw one, we followed the popular dataset evaluation method described in [118], which analyzes critical LLM dataset characteristics such as data size, comprehensiveness, and how easily we can retrieve data from it.

7.4.1.1 Data size & Vocabulary uniqueness

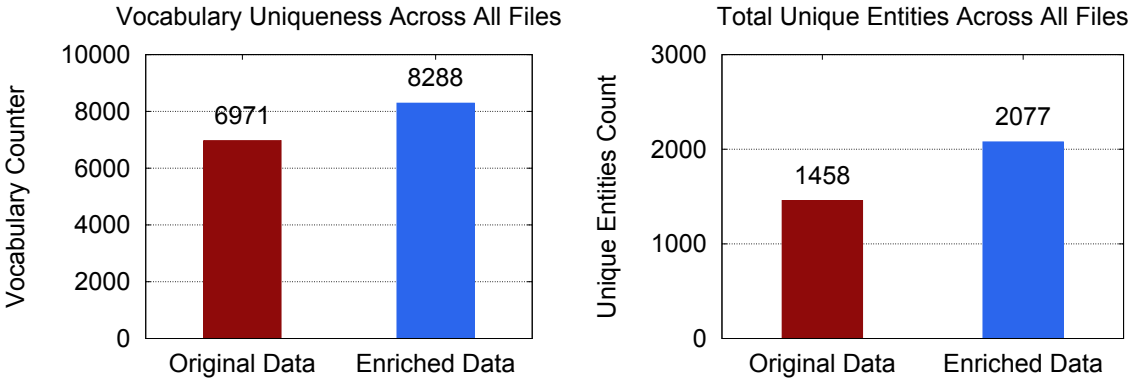
As already mentioned, LLMs are based on identifying patterns observed using statistical correlations. The enhanced dataset is obviously larger than the raw data. Although, presenting only its size would not support the efficiency of our proposed dataset. That is why we focused on the vocabulary uniqueness, which defends that the diversity and the richness of the dataset not only offer a large quantity of text to the indexing mechanism of RAG, but also exposes it to a wide variety of words, phrases, and linguistic structures. This plays a crucial role, especially in the ICL and the way the RAG system will provide the necessary information to the model query.

The enriched data size is approximately 20% bigger than the raw one. Figure 7.2a, illustrates that the enriched data have 18.9% more uniqueness regarding the vocabulary, implying a highly efficient enrichment process without leading to an overload dataset, since the additional text is just a lexical restructuring of meaning and content of the raw data.

7.4.1.2 Data comprehensiveness

This lexical restructuring, alongside with the necessary explanation provided to some contexts of the original dataset (such as a more brief explanation of programming commands, more detailed technology architectural analysis, and theory), has led to a clearer ontological existence of entities in the enhanced dataset. Such an example of an entity could be the 5G components, different kind of testbed equipment and others. Entities have a crucial role

in a dataset's comprehensiveness since they reflect real-world complexity into it. To validate the enrichment process with respect to a more structured and well-written enhancement, we utilized the biggest pre-trained model of SpaCy library, en_core_web_lg [119], which is a popular large English model designed for general-purpose Natural Language Processing (NLP) tasks, to identify entities.



(a) Overall Vocabulary Uniqueness Across All Files

(b) Total Unique Entities Across All Files

Figure 7.2: Data Comparison: Vocabulary & Entities Uniqueness

Following the results of the vocabulary analysis, we continued the evaluation of our enhanced dataset to its entity dimension, comparing it to the original one. As unique entities, we identify each distinct entity that the model was able to recognize in our datasets, counting each entity only once, regardless of how many times it appears across the data. This ensures that even if an entity is observed multiple times, it is included as a single entry in the final set of identified entities. Figure 7.2b, illustrates a 42.5% increase in the total unique entities of the enriched data identified by the entity model. It is worth mentioning that the model fed with the original data, could not understand some key components of SLICES tutorials, such as the F1 interface, which is an important interface in the distributed 5G RAN architecture. These results indicate that the descriptive power and utility of the enhancement lead to an ML model being able to identify domain-specific terms and concepts.

7.4.1.3 Retriever Analysis

To assess our RAG implementation, we need to find suitable metrics, such as the similarity metric, that describe the process of finding the index's nodes that match the user queries. The proposed RAG system, implemented with the usage of the Llama library is based on the following strategy: Initially, the retriever is responsible for fetching the most relevant context

from the available index nodes based on the query's degree of similarity and the node's content. The topology of the nodes could be one of those described in 7.3.3.1. Following that, the NodePostProcessor accepts a set of retrieved nodes as input and then applies filtering, augmentation, and replacement transformations. In our implementation, we used the Vector Store Indexing topology and the NodePostProcessor, to perform filtering based on the similarity metric. We set as a lower limit the value at 0.8, to support the model being fed with a lot of data with relevant content when we have a high degree of similarity, and with less when data's content is not so relevant. Finally, there is the response synthesizer, which takes the transformed set of nodes from the NodePostProcessor and, based on their content, forms the information that our system will provide to LLM (such as GPT-4 in our case) to generate the response.

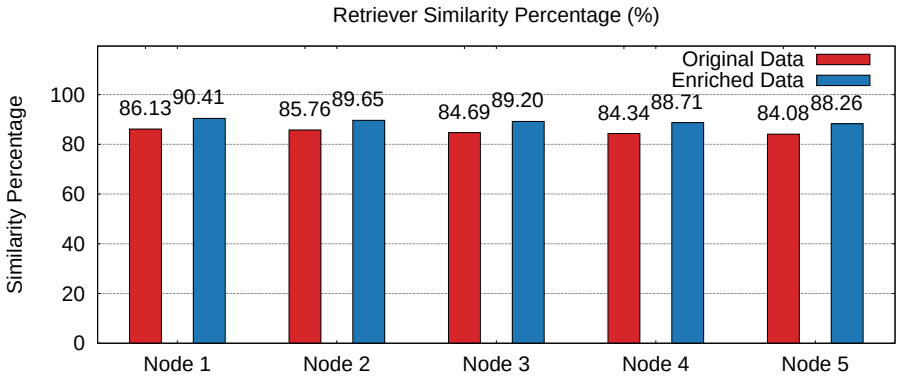


Figure 7.3: Retriever Similarity Percentage

We used the retrieving process to evaluate the impact of our enhanced dataset over the raw one. As already mentioned, the retriever is able to identify nodes of the indexed data with similar content to the user query. Figure 7.3 illustrates the average similarity scores from the ith nodes obtained by the retriever for the open-ended questions described in Section 7.4.2. As depicted, the retriever in the case of enhanced data can recognize a higher similarity degree of the top_k nodes (5 in our case), having an average of 5% higher similarity over the raw data.

7.4.2 Performance Evaluation - In-context Learning

To further assess the chatbot's performance before and after data augmentation, we have also generated 25 multiple-choice questions and open-ended questions from our datasets, which can be found in [120]. It is important to mention that each question was formulated across a diverse range of data sources and was structured to have answers that had small deviations from the correct one, in order to increase the level of difficulty. Subsequently, the questions were first presented to the RAG system with the original dataset, and then to the same model indexed with the enhanced dataset. The RAG system was prompted the same way for both the raw and the enhanced dataset, outlining the role of RAG as a chatbot assistant, as well as generating responses based only on the information provided to it. As depicted in Figure 7.4, the correctly-answered questions percentage by the model with the original-data was 76%, in contrast to the enhanced-data model that achieved 96%. This difference suggests a huge performance improvement, as it underscores the importance of data augmentation in enhancing the chatbot's understanding and response accuracy.

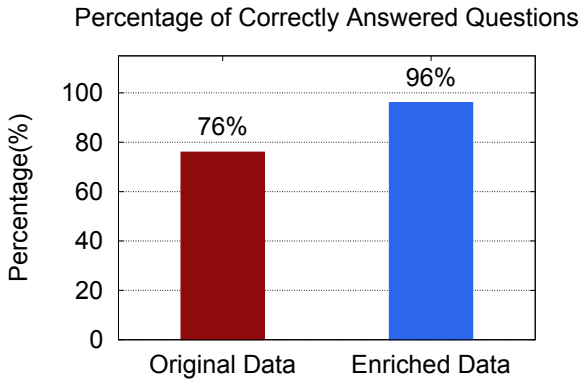


Figure 7.4: Percentage of Correctly Answered Questions

Since there is no deterministic way to evaluate the validity of an open-ended response from LLMs formed from multiple data sources, we adopted the human validation method. Using our domain expertise and dataset familiarity, we checked the generated responses in terms of validity, closeness, clarity and descriptiveness for both datasets (original & enriched).

Regarding the open-ended answers, the model that uses the enhanced data provided responses with more precision and included more information compared to the model using the

Table 7.1: Question Responses per Model & Dataset

Question	**Which nodes of the NITOS testbed should I use to deploy the 5G RAN using USRPs according to 5G Blueprint?**
Original Data with GPT-4 Model	The given context does not provide specific details about the nodes of NITOS that can deploy 5G RAN using USRPs according to the 5G Blueprint. Detailed information regarding this would be required to provide a precise answer.
Enhanced Data with GPT-4 Model	The NITOS testbed nodes that are capable of deploying the 5G RAN using USRPs, according to the 5G Blueprint, are: node055, node056, node059, node060, node067, node068, node071, node072. These nodes are equipped with USRP B210 models, which are connected via USB 3.0. The USRP B210 devices are particularly significant for 5G research due to their wide frequency range and high bandwidth capabilities, enabling researchers to simulate and analyze various aspects of 5G networks.
Enhanced Data - GPT-3.5	To deploy the 5G RAN using USRPs according to the 5G Blueprint in the NITOS testbed, you should utilize the nodes equipped with USRP N210 and USRP B210 devices. Here are the specific nodes that you should use for this purpose: USRP N210 Nodes: node052, node064, node080 USRP B210 Nodes (Connected via USB 3.0): node055, node056, node059, node060, node067, node068, node071, node072.
Enhanced Data - Fine-Tuned GPT-3.5	The NITOS testbed encompasses specific nodes equipped to facilitate the deployment of 5G RAN experiments utilizing USRP devices. According to the 5G Blueprint, the following NITOS nodes are suitable for this purpose: node055, node056, node059, node060, node067, node068, node071, node072. These nodes are instrumental in realizing advanced 5G RAN scenarios, enabling researchers to leverage the versatile capabilities of USRP B210 models for comprehensive and cutting-edge experiments.

raw dataset. When it came to modifying code and configuration files, both models achieved the expected outcomes, with the responses from the RAG system using the enriched data being more solid, especially in the reasoning part. For example, as depicted in Table 7.1, when asking the same question (*Which nodes of the NITOS [23] testbed should I use to deploy the 5G RAN using USRPs according to 5G Blueprint [121]?*) to the GPT-4 model, the chatbot did not respond in the case of raw data, in contrast to the enhanced one, which answered correctly. At this point we note that the same information was present in both datasets, but the GPT-4 model could not successfully identify or interpret it to produce good responses.

7.4.3 Performance Evaluation - Fine Tuned Model

As already mentioned in Section 7.3.3.2, we fine-tuned the LLM model used in our RAG system. The fine-tuning was carried out to describe general strategies that we want our model to adopt, as well as to address specific weaknesses that we observed during the development of ICL, giving the model the desired functionality and behavior. At the time of conducting

this study, no publicly available version of GPT-4 was available for fine-tuning. To perform fine-tuning, we limited our research to the available model of GPT-3.5. We followed a similar evaluation strategy as in 7.4.2, where we exposed our system to the same questions (multiple-choice and open-ended).

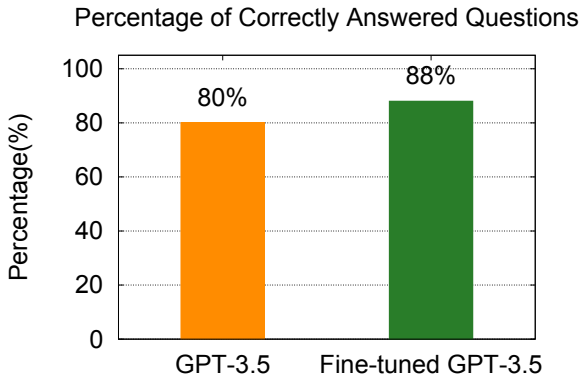


Figure 7.5: Raw GPT-3.5 Model vs Fine-tuned GPT-3.5 Model: Percentage of Correctly Answered Questions

Figure 7.5 shows the percentage of correct answers brought by our RAG system, when utilizing either the standard GPT-3.5 model or the fine-tuned GPT-3.5 Turbo, in combination with the enhanced dataset. As expected, the fine-tuned model achieved 88% correct answers, which suggests a great improvement to the default model, that achieved around 80%. This highlights the necessity of fine-tuning an LLM, in order to fully exploit the model's potential and achieve better performance, especially in the case of specific-domain data. It is worth noting that the fine-tuned GPT-3.5-turbo model does not perform better than the default GPT-4, giving even greater margins of correct answers to a fine-tuned GPT-4 model.

In the case of open-ended responses, we have observed that the fine-tuned GPT-3.5 model was able to answer sophisticated and complex questions with quality reasoning, in contrast to the GPT-3.5 (with enhanced data).

An example of such differences can be seen in Table 7.1. The correct answer should only include nodes equipped with USRP B210s and not those with USRP N210s as illustrated in [121]. The GPT-3.5 provided all the nodes equipped with USRPs, even though not all USRP models are suitable for deploying the 5G Blueprint. Fine-Tuned GPT-3.5 managed

to provide only the nodes suitable for conducting the experiment (similar answer to GPT-4 Model).

7.5 Toward Automated Experiment Execution

The work presented in Sections 7.3 through 7.4 represents the initial version of slAIces, developed at a time when LLM-based assistants for specialized technical domains were still an emerging research area and the primary challenge was demonstrating that domain-specific knowledge could be reliably integrated into general-purpose models. Since then, the rapid maturation of LLM capabilities and the emergence of production-grade deployment frameworks have opened the door to a more ambitious objective: not merely assisting researchers in understanding the infrastructure, but fully automating the experiment lifecycle. Building upon the validated foundation of the initial slAIces, we extend the framework from a purely informational assistant into a platform capable of fully automating experiment execution, pursuing two objectives: answering general user queries about the SLICES infrastructure, and converting natural language instructions into executable experiments compatible with the SLICES-RI Post-5G blueprint format.

7.5.1 Implementation

Figure 7.6 provides a high-level overview of the extended implementation. Upon connecting to the SLICES portal, the user is presented with two interaction modes: a **Chat** interface for conversational interaction with the chatbot, and a **Configuration** tab that tracks and displays the current state of the experiment blueprint throughout the session. These two modes operate in parallel, with the Chat interface driving configuration changes that are continuously reflected in the Configuration tab.

In the Chat interface, prompts are forwarded to a Dedicated Preprocessor LLM (DPLLM), which analyzes both the prompt content and the full conversation history to infer the user's intent and route the request to the appropriate pipeline. This intent classification step is critical: without it, a single monolithic pipeline would either over-respond to simple queries or under-serve complex configuration requests. The DPLLM routes each request to one of the following pipelines:

- **SLICES Documentation:** Handles general queries about the SLICES infrastructure,

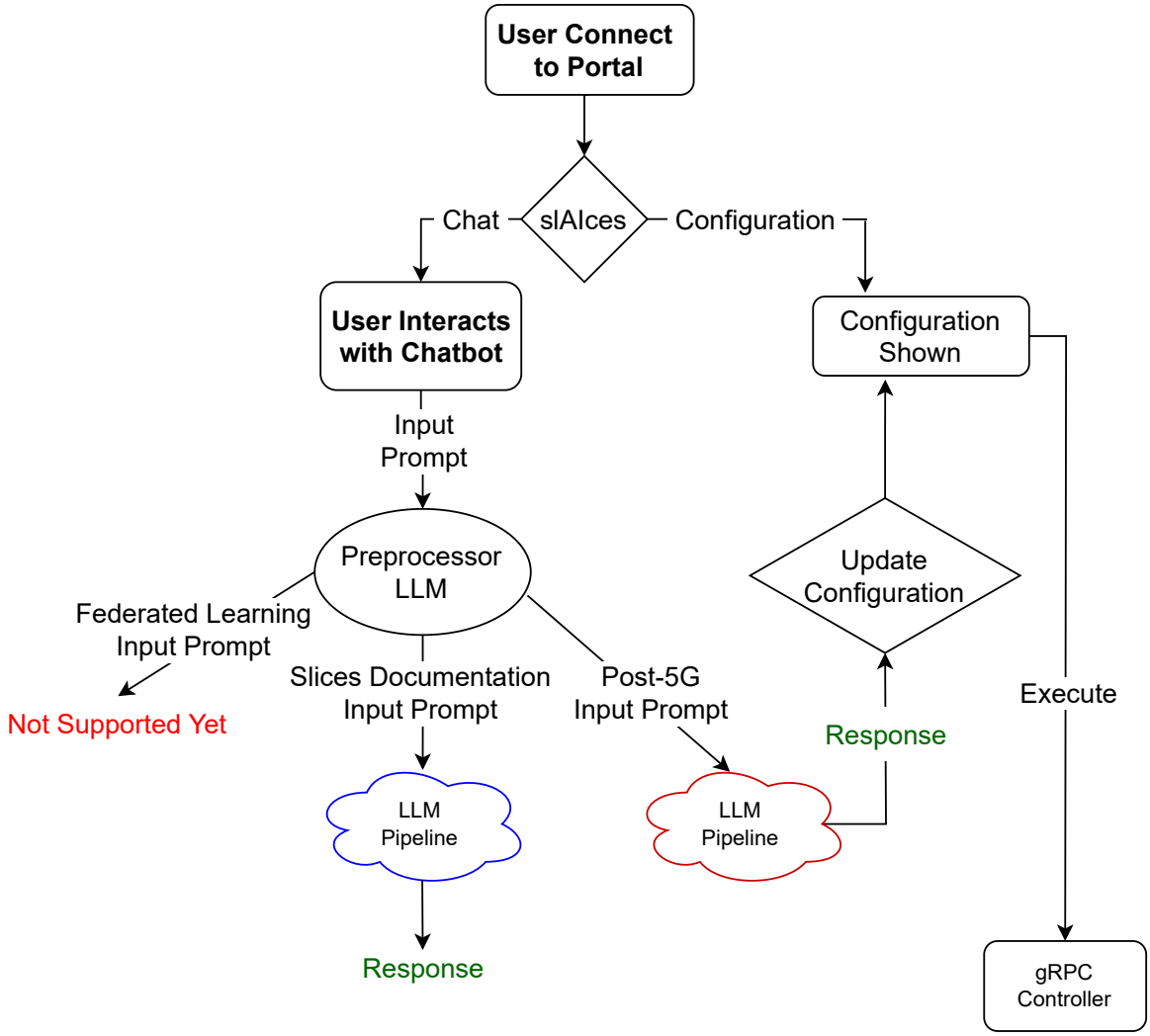


Figure 7.6: Framework Overview.

including questions about available resources, testbed capabilities, supported experiment types, and blueprint theory. No changes are made to the experiment configuration.

- **Post-5G:** Handles experiment configuration requests expressed in natural language, generating both a text explanation for the user and a JSON patch applied to the active blueprint. The updated configuration is immediately reflected in the Configuration tab.

Once the user is satisfied with the configuration, pressing the Execute button in the Configuration tab triggers the **gRPC Controller**. The Controller performs a final validation of critical experimental variables and once all checks pass, forwards the finalized configuration to the SLICES-RI infrastructure for deployment.

SLICES Documentation Pipeline. Figure 7.7 illustrates the pipeline for handling general queries about the SLICES infrastructure. A RAG system retrieves relevant context from the

official SLICES-RI documentation based on the input prompt, and the LLM integrates this context alongside the conversation history to generate an accurate, up-to-date response without requiring model retraining.

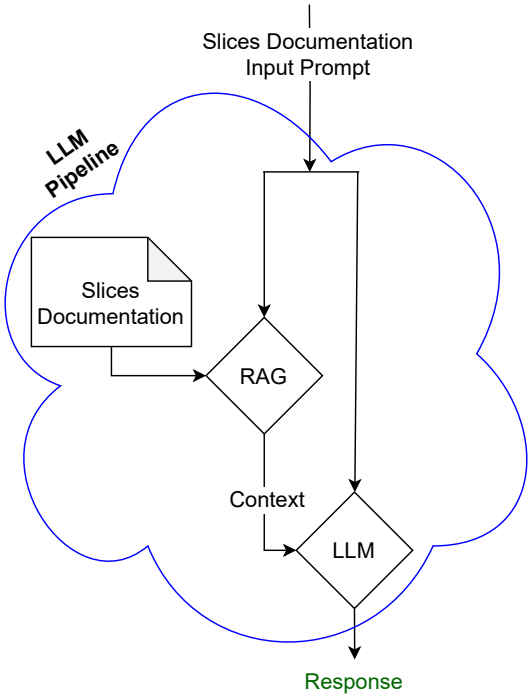


Figure 7.7: SLICES Documentation LLM Pipeline.

Post-5G Pipeline. Figure 7.8 illustrates the pipeline for handling experiment configuration requests. A SLICES Blueprint provides a structured and abstract definition of an experiment, specifying the desired infrastructure components, services, and topology, implemented in practice as a JSON configuration file. The LLM takes the user prompt, conversation history, current experiment configuration, and RAG-retrieved documentation as inputs, and produces two outputs: a text explanation delivered to the user in the Chat interface, and a JSON patch that encodes the specific changes to be applied to the active Post-5G blueprint. The JSON patch is passed to an Update Configuration step, which merges the changes into the current configuration state and reflects them immediately in the Configuration tab. This loop — prompt, pipeline response, configuration update, user review — can be repeated iteratively until the desired experiment setup is achieved.

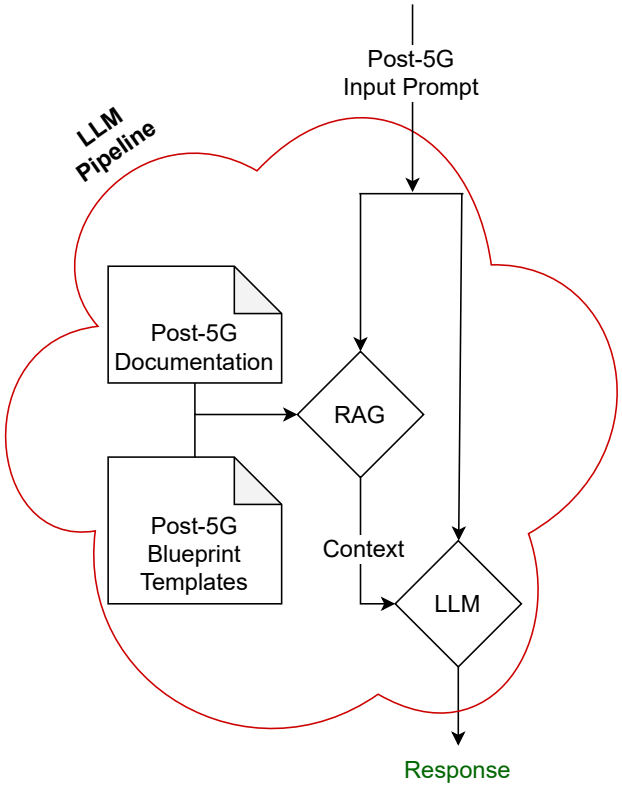


Figure 7.8: Post-5G LLM Pipeline.

7.5.2 Controller

Once the experiment configuration is complete, the experimenter can instruct the chatbot to trigger execution via the gRPC Controller, deployed as a service. The Controller applies the configuration generated by the LLM and performs a final validation of critical experimental parameters — resource selection, experiment configuration— before deployment. If any parameters are missing or inconsistent, the Controller notifies the experimenter and prompts further interaction with the LLM to resolve them. Once all validation checks pass, the Controller forwards the finalized configuration to the SLICES-RI infrastructure and deploys the experiment.

7.5.3 Performing an Experiment

The experimenter begins by asking general questions about available blueprints and infrastructure capabilities through the Chat interface. Based on this information, they progres-

sively define the desired experiment using natural language instructions, with the chatbot incrementally refining the blueprint configuration after each exchange. At any point, the experimenter can switch to the Configuration tab to inspect the current state of the blueprint and verify that all parameters are correctly set. Once satisfied, they press the Execute button, which triggers the gRPC Controller and initiates the deployment process on the SLICES-RI infrastructure.

7.6 Conclusion & Future Work

In this chapter, we introduced slAIces, a sophisticated chatbot assistant designed to facilitate and enhance the user experience within SLICES-RI. We presented our methodology and developed a RAG system employing both in-context learning and fine-tuning, two complementary methods for integrating domain-specific knowledge into an LLM. Through rigorous evaluation, we demonstrated that data augmentation and fine-tuning significantly improve response quality, achieving accuracy of 96% and 88% in correctly answering multiple-choice questions with GPT-4 and fine-tuned GPT-3.5, respectively.

Building on these results, Section 7.5 presented an extended framework that moves slAIces beyond information retrieval into automated experiment execution. The addition of a multi-pipeline architecture, a Dedicated Preprocessor LLM, and a deployment Controller enables experimenters to design, configure, and deploy Beyond-5G experiments through natural language interaction alone, without directly engaging with the underlying infrastructure.

Future work will focus on extending the closed-loop capabilities of the system by incorporating valid experiment results collected from the SLICES-RI infrastructure back into the training dataset, allowing the chatbot to continuously adapt to the evolving needs of the research community. Additionally, further pipelines covering federated learning blueprints and additional testbed types are planned, broadening the scope of experiments that can be automated through the framework.

Chapter 8

Conclusions and Future Work

This thesis investigated the evaluation, orchestration, and intelligent management of cloud-native 5G core networks, addressing challenges that are central to the realization of the 6G vision. The contributions spanned across five research directions, each grounded in real-world experimentation on the SLICES-RI infrastructure, and collectively advance the path toward sustainable, adaptive, and accessible 6G core network management.

8.1 Summary of Contributions

Real-World Performance Evaluation of UPF Implementations

The thesis began by addressing the absence of systematic, real-world comparative evaluation of UPF implementations. Four architecturally distinct open-source solutions — SPGWU, VPP-UPF, P4-Switch, and SmartNIC-P4 — were deployed within a fully operational end-to-end 5G network built on OpenAirInterface on the NITOS and LIP6 testbeds of SLICES-RI, and evaluated simultaneously across throughput, latency, packet loss, and CPU utilization under identical traffic conditions.

The evaluation established a clear empirical picture of the trade-off landscape. P4-Switch UPF achieved elite performance in both throughput and packet loss, offloading all data-plane processing to dedicated ASIC hardware and consuming negligible host CPU resources, confirming hardware acceleration as the performance ceiling for UPF deployments. VPP-UPF emerged as the most promising software-only alternative: it achieved high throughput and competitive latency at substantially lower cost than hardware solutions, though at the expense of continuous CPU occupancy due to DPDK polling — a behavior that directly motivated the

energy-aware orchestration of Chapter 5. SmartNIC-P4 reached the highest uplink throughput of any tested implementation but exhibited unexpectedly high RTT and packet loss under real GTP-U traffic, revealing implementation-specific constraints that only manifest in real 5G conditions. SPGWU provided the lightest resource footprint at low load but saturated early, confirming its role as a baseline solution rather than a production-grade option under high demand.

These empirical profiles constitute the quantitative foundation on which all subsequent contributions are built.

Reactive Horizontal Scaling of UPF Instances in Kubernetes

Building on the observation that hardware UPF solutions are inherently static while software-based implementations can be elastically managed, Chapter 4 designed, implemented, and evaluated a reactive UPF scaling mechanism for cloud-native 5G deployments orchestrated by Kubernetes. Driven by real-time data-plane CPU signals, the mechanism deploys additional UPF instances when demand approaches capacity and releases them when load subsides.

Evaluated on the NITOS testbed with a real-world telecommunications dataset, the dynamic mechanism demonstrated significant improvements in CPU allocation efficiency and packet reception compared to a static monolithic deployment, preventing congestion events that fixed provisioning cannot avoid. This work validated the feasibility of Kubernetes-driven UPF lifecycle management and identified the inherent limitation of purely reactive, homogeneous scaling as well as its inability to anticipate demand or account for the energy implications of different UPF implementations, directly motivating the proactive and heterogeneous framework of Chapter 5.

Proactive and Energy-Aware Orchestration of Heterogeneous UPFs

Chapter 5 proposed a proactive and energy-aware UPF selection and scaling framework that jointly optimizes both the number and type of UPF instances at each decision interval, addressing the two limitations identified in Chapter 4. The framework integrates three components: empirical profiling of SPGWU and VPP-UPF to derive polynomial CPU utilization and linear power consumption models that accurately capture their non-linear behaviors under real traffic; a CNN-LSTM model for 3-step-ahead throughput forecasting, se-

lected through systematic benchmarking against LSTM, GRU, and XGBoost baselines; and an energy-aware optimization mechanism that uses the forecast and the derived models to select the deployment configuration that minimizes total energy consumption while satisfying predicted demand within the available node budget.

Evaluated across five deployment scenarios on a real Kubernetes-based SLICES-RI testbed, the framework was the only configuration to satisfy all throughput demands, achieving an average packet loss of only 1.5% while reducing total energy consumption by 51% compared to the static all-VPP deployment — the only baseline otherwise capable of meeting equivalent throughput requirements. These results demonstrated that the energy-performance trade-off in software UPF deployments is not a fixed constraint but a continuously optimizable operating point, providing a practical and empirically validated foundation for sustainable user plane orchestration in beyond-5G and 6G core networks.

Non-3GPP Access Integration and ATSSS Evaluation

Chapter 6 extended the scope of the thesis to the multi-access convergence challenge at the heart of the 6G ubiquitous connectivity vision. Two distinct approaches to integrating non-3GPP access into the 5G Core were evaluated entirely over real over-the-air 5G NR and Wi-Fi radio channels on the NITOS testbed. Core-level integration via the N3IWF was used to evaluate both ATSSS-specified multipath transport protocols: MPTCP achieved 267 Mbps aggregated throughput with robust failover recovery within 2 seconds, while MPQUIC, after resolving a fundamental incompatibility between Picoquic's RACK-based loss detection and the N3IWF tunneled path, achieved 317 Mbps with instantaneous path migration upon gNB failure. RAN-level integration via a custom Wi-Fi-DU managed at the PDCP layer achieved the highest throughput of 606 Mbps, but at the cost of 3GPP non-compliance and loss of path redundancy.

These results exposed a fundamental trade-off between standards compliance, aggregation performance, and resilience, with direct implications for the design of heterogeneous access management in beyond-5G and 6G networks.

LLM-Based Automation of Network Experimentation

Chapter 7 shifted focus from network function management to the usability of the experimental infrastructure that underpins the entire thesis. slAIces, an LLM-based chatbot

grounded in SLICES-RI documentation through RAG and fine-tuning, was developed and evaluated as a tool for lowering the barrier for new users accessing SLICES-RI. Systematic evaluation demonstrated that structured data augmentation of the knowledge base raises answer accuracy from 76% to 96%, confirming that the structural gap between human-oriented documentation and LLM statistical pattern recognition requires explicit bridging. Fine-tuning of GPT-3.5 further improved domain-specific performance from 80% to 88%, with the two techniques addressing complementary failure modes: RAG provides factual grounding while fine-tuning instills procedural domain reasoning.

The framework was further extended into a fully automated experiment execution platform integrating a Dedicated Preprocessor LLM for intent classification, a multi-pipeline architecture covering SLICES documentation queries and Post-5G blueprint generation, and a gRPC-based deployment Controller that validates and deploys experiments onto the SLICES-RI infrastructure, directly enabling experimenters to design, configure, and deploy Beyond-5G experiments through natural language alone.

8.2 Perspectives for Future Work

The contributions of this thesis open several directions for future investigation, both as direct extensions of the presented works and as broader research avenues toward 6G core network management.

Multi-Objective and Latency-Aware UPF Orchestration

The contributions of Chapters 3, 4, and 5 collectively establish a progressive research arc spanning from empirical UPF evaluation, to reactive scaling, to proactive energy-aware orchestration that naturally points toward a unified, multi-objective UPF management framework. A direct extension is the evaluation of new UPF architectures that will emerge alongside 6G standardization, expanding the empirical landscape needed to guide deployment and orchestration decisions in future networks. Beyond benchmarking, incorporating latency as an additional optimization objective alongside energy consumption would better address the multi-dimensional SLA requirements of 5G/6G network slices, particularly for URLLC services where energy savings must not come at the cost of latency degradation. A particularly compelling direction is the design of an ML-driven adaptive VPP-UPF, in which the predic-

tion model drives VPP RX mode selection proactively ahead of traffic transitions, collapsing heterogeneous multi-instance orchestration into a single self-adapting instance without PDU session disruption. Taken together, these directions point toward a fully autonomous UPF management plane capable of jointly optimizing performance, energy, and latency across diverse deployment scenarios.

Dynamic Orchestration Under Multi-Access Conditions

The evaluation of Chapter 6 characterized the performance and resilience trade-offs of multi-access integration and ATSSS, but did not address how the additional variability introduced by heterogeneous radio paths affects dynamic UPF management. A natural extension is the integration of satellite communications as a third non-3GPP access type, expanding the evaluation to a truly ubiquitous connectivity scenario that reflects the 6G vision. Beyond access diversity, replacing the static SMF-provisioned ATSSS steering rules with AI-driven policies — leveraging the Network Data Analytics Function (NWDAF) to adapt to the time-varying conditions of real multi-RAT deployments — represents a compelling direction toward intelligent, closed-loop multi-access management. Connecting these advances to the energy-aware orchestration framework of Chapter 5, by designing a unified management plane that jointly optimizes UPF deployment and ATSSS steering decisions under multi-access conditions, would represent a significant step toward the holistic 6G core management vision outlined in this thesis.

Toward Closed-Loop LLM-Assisted Experimentation

The slAIces framework of Chapter 7 demonstrated that LLMs can effectively lower the barrier to SLICES-RI access and automate experiment execution. Future work should pursue a fully closed-loop system in which experiment results collected from the infrastructure are automatically validated and fed back into the knowledge base, enabling the chatbot to continuously improve its guidance based on the accumulated experimental record of the research community. In addition, extending the multi-pipeline architecture to cover additional experiment types and exploring more capable fine-tuned models as they become available would further broaden the scope and improve the response quality of the system.

8.3 Lessons Learned and Outlook

Reflecting on the research conducted in this thesis, several key lessons emerge that have broader implications for the 6G research community.

Real-world validation is indispensable. A consistent finding across all chapters is that behaviors observed in real, end-to-end deployments differ meaningfully from what simulation or isolated benchmarking would predict. The SmartNIC-P4 UPF's unexpectedly high RTT under real GTP-U traffic, VPP's load-independent CPU occupancy due to DPDK polling, and Picoquic's fundamental incompatibility with the N3IWF tunneled path are all phenomena that only manifest under full-stack, real network conditions. This reinforces not only the importance of large-scale experimental infrastructures as essential instruments for credible 6G research, but also the need to conduct experiments under real network conditions rather than controlled or emulated environments.

Heterogeneity is both a challenge and an opportunity. The most significant gains in performance and energy efficiency in this thesis emerged from exploiting the differences between UPF implementations rather than treating them as interchangeable. The same principle extends to access technologies: the diversity of cellular, Wi-Fi, and satellite access is not an obstacle to be hidden but a resource to be intelligently coordinated. Future 6G management frameworks should be designed to recognize and leverage this heterogeneity rather than abstracting it away.

Data quality determines the ceiling of intelligent management. Across the ML-driven and LLM-based contributions of this thesis, the quality of input data had a more decisive impact on system performance than architectural choices alone. As 6G networks increasingly rely on AI-driven control, investing in the quality and accessibility of operational data will be as important as the design of the models themselves.

The 6G vision of a fully autonomous, sustainable, and universally accessible network is ambitious, but the research presented in this thesis demonstrates that meaningful progress toward it is achievable. Each contribution addresses one dimension of this vision through real-world experimentation, collectively building an empirical foundation for the intelligent core network management systems that future networks will require.

References

[1] Sokratis Christakis, Theodoros Tsourdinis, Nikos Makris, Thanasis Korakis, and Serge Fdida. Evaluation of User Plane Function Implementations in Real-World 5G Networks. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2024.

[2] Sokratis Christakis, Nikos Makris, Thanasis Korakis, and Serge Fdida. On the Automated Scaling of User Plane Function for 5G: An Experimental Evaluation. In *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 979–984, 2024.

[3] Sokratis Christakis, Nikos Makris, Serge Fdida, and Thanasis Korakis. Dynamic AI-Based UPF Switching: Adapting for Performance/Energy-Efficiency in 5G Networks. In *GLOBECOM 2025 - 2025 IEEE Global Communications Conference*, pages 5580–5585, 2025.

[4] Dimitris Kefalas, Sokratis Christakis, Serge Fdida, Nikos Makris, Ilias Syrigos, Virgilios Passas, and Thanasis Korakis. slAIces: an LLM Chatbot for Simplifying Experiments with the SLICES-RI. In *2024 IFIP Networking Conference (IFIP Networking)*, pages 660–665, 2024.

[5] ITU-R. Framework and Overall Objectives of the Future Development of IMT for 2030 and Beyond. Recommendation ITU-R M.2160-0, International Telecommunication Union, November 2023.

[6] 3GPP. 3GPP Workshop on 6G (IMT-2030), 2025. Workshop held in Incheon, South Korea, 10–11 March 2025. Available at: `https://www.3gpp.org/component/content/article/6gworkshop-2025`.

[7] Zhengquan Zhang, Yue Xiao, Zheng Ma, Ming Xiao, Zhiguo Ding, Xianfu Lei, George K. Karagiannidis, and Pingzhi Fan. 6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies. *IEEE Vehicular Technology Magazine*, 14(3):28–41, 2019.

[8] Ian F. Akyildiz, Chong Han, and Shuai Nie. Combating the Distance Problem in the Millimeter Wave and Terahertz Frequency Bands. *IEEE Communications Magazine*, 56(6):102–108, 2018.

[9] Wei Jiang, Qiuheng Zhou, Jiguang He, Mohammad Asif Habibi, Sergiy Melnyk, Mohammed El-Absi, Bin Han, Marco Di Renzo, Hans Dieter Schotten, Fa-Long Luo, Tarek S. El-Bawab, Markku Juntti, Mérouane Debbah, and Victor C. M. Leung. Terahertz Communications and Sensing for 6G and Beyond: A Comprehensive Review. *IEEE Communications Surveys & Tutorials*, 26(4):2326–2381, 2024.

[10] Ertugrul Basar, Marco Di Renzo, Julien De Rosny, Merouane Debbah, Mohamed-Slim Alouini, and Rui Zhang. Wireless Communications Through Reconfigurable Intelligent Surfaces. *IEEE Access*, 7:116753–116773, 2019.

[11] Khaled B. Letaief, Wei Chen, Yuanming Shi, Jun Zhang, and Ying-Jun Angela Zhang. The Roadmap to 6G: AI Empowered Wireless Networks. *IEEE Communications Magazine*, 57(8):84–90, 2019.

[12] Fan Liu, Yuanhao Cui, Christos Masouros, Jie Xu, Tony Xiao Han, Yonina C. Eldar, and Stefano Buzzi. Integrated Sensing and Communications: Toward Dual-Functional Wireless Networks for 6G and Beyond. *IEEE Journal on Selected Areas in Communications*, 40(6):1728–1767, 2022.

[13] 3GPP. Study on New Radio (NR) to support non-terrestrial networks. Technical Report TR 38.811 V15.4.0, 3rd Generation Partnership Project, September 2020.

[14] Ericsson. Ericsson Mobility Report. Technical report, Ericsson AB, Stockholm, Sweden, November 2023. [Online]. Available: `https://www.ericsson.com/en/reports-and-papers/mobility-report` [Accessed: Jan. 2025].

[15] 3GPP. Service requirements for the 5G system (Release 17). Technical Specification TS 22.261, 3rd Generation Partnership Project, 2022.

[16] ITU-R. IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. Recommendation M.2083-0, International Telecommunication Union, 2015.

[17] 3GPP. System architecture for the 5G System (5GS) (Release 15). Technical Specification TS 23.501, 3rd Generation Partnership Project, 2018.

[18] 3GPP. Procedures for the 5G System (5GS) (Release 16). Technical Specification TS 23.502, 3rd Generation Partnership Project, 2020.

[19] Docker Inc. Docker Overview, 2024. Available at: `https://docs.docker.com/get-started/docker-overview/`. Accessed: March 2025.

[20] Docker Inc. Docker Compose Overview, 2024. Available at: `https://docs.docker.com/compose/`. Accessed: March 2025.

[21] Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson. *Kubernetes: up and running*. " O'Reilly Media, Inc.", 2022.

[22] Serge Fdida, Nikos Makris, Thanasis Korakis, Raffaele Bruno, Andrea Passarella, Panayiotis Andreou, Bartosz Belter, Cédric Crettaz, Walid Dabbous, Yuri Demchenko, and Raymond Knopp. SLICES, a scientific instrument for the networking community. *Computer Communications*, 193:189–203, 2022.

[23] Nikos Makris, Christos Zarafetas, Spyros Kechagias, Thanasis Korakis, Ivan Seskar, and Leandros Tassiulas. Enabling open access to LTE network components; the NITOS testbed paradigm. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[24] Navid Nikaein et al. OpenAirInterface: A Flexible Platform for 5G Research. *SIGCOMM Comput. Commun. Rev.*, 44(5):33–38, October 2014.

[25] Ali Gungor. UERANSIM: Open source state-of-the-art 5G UE and RAN (gNodeB) simulator. `https://github.com/aligungr/UERANSIM`, 2024. Accessed: 2026-03-17.

[26] free5GC. free5GC: Open Source 5G Core Network, 2024. Available at: `https://free5gc.org`. GitHub: `https://github.com/free5gc/free5gc`. Accessed: March 2025.

[27] Nikolaos Apostolakis, Marco Gramaglia, and Pablo Serrano. Design and Validation of an Open Source Cloud Native Mobile Network. *IEEE Communications Magazine*, 60(11):66–72, 2022.

[28] Hai T. Nguyen, Tien Van Do, and Csaba Rotter. Scaling UPF Instances in 5G/6G Core With Deep Reinforcement Learning. *IEEE Access*, 9:165892–165906, 2021.

[29] Theodoros Tsourdinis, Nikos Makris, Serge Fdida, and Thanasis Korakis. DRL-based Service Migration for MEC Cloud-Native 5G and beyond Networks. In *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, pages 62–70, 2023.

[30] Irian Leyva-Pupo, Alejandro Santoyo-González, and Cristina Cervelló-Pastor. A Framework for the Joint Placement of Edge Service Infrastructure and User Plane Functions for 5G. *Sensors*, 19(18), 2019.

[31] Zhi Wang and Yigang Cai. Management Optimization of Mobile Edge Computing (MEC) in 5G Networks. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2019.

[32] Wenjun Zhu, Peng Li, Baozhou Luo, He Xu, and Yujie Zhang. Research and Implementation of High Performance Traffic Processing Based on Intel DPDK. In *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 62–68, 2018.

[33] Haoran Zhang, Zikang Chen, and Yang Yuan. High-Performance UPF Design Based on DPDK. In *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pages 349–354, 2021.

[34] Thiago A. Navarro do Amaral, Raphael V. Rosa, David F. Cruz Moura, and Christian E. Rothenberg. An In-Kernel Solution Based on XDP for 5G UPF: Design, Prototype and Performance Evaluation. In *2021 17th International Conference on Network and Service Management (CNSM)*, pages 146–152, 2021.

[35] David Soldani, Petrit Nahi, Hami Bour, Saber Jafarizadeh, Mohammed F. Soliman, Leonardo Di Giovanna, Francesco Monaco, Giuseppe Ognibene, and Fulvio Risso. eBPF: A New Approach to Cloud-Native Observability, Networking and Security for

Current (5G) and Future Mobile Networks (6G and Beyond). *IEEE Access*, 11:57174–57202, 2023.

[36] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. The eXpress data path: fast programmable packet processing in the operating system kernel. CoNEXT '18, page 54–66, New York, NY, USA, 2018. Association for Computing Machinery.

[37] Jianer Zhou, Zengxie Ma, Weijian Tu, Xinyi Qiu, Jingpu Duan, Zhenyu Li, Qing Li, Xinyi Zhang, and Weichao Li. Cable: A framework for accelerating 5G UPF based on eBPF. *Computer Networks*, 222:109535, 2023.

[38] Robert MacDavid, Carmelo Cascone, Pingping Lin, Badhrinath Padmanabhan, Ajay ThakuR, Larry Peterson, Jennifer Rexford, and Oguz Sunay. A P4-based 5G User Plane Function. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, SOSR '21, page 162–168, New York, NY, USA, 2021. Association for Computing Machinery.

[39] Justus Rischke, Christian Vielhaus, Peter Sossalla, Junyue Wang, and Frank H.P. Fitzek. Comparison of UPF acceleration technologies and their tail-latency for URLLC. In *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 19–25, 2022.

[40] TravelPing. UPG-VPP: VPP-based User Plane Gateway. `https://github.com/travelping/upg-vpp`, 2024. Accessed: 2026-03-17.

[41] FD.io. An Extensible Framework That Provides Out-of-the-Box Production Quality Switch/Router Functionality. White paper, FD.io, July 2017.

[42] Tiago Rosado and Jorge Bernardino. An overview of Openstack architecture. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, 2014.

[43] Leonardo Bonati, Michele Polese, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead. *Computer Networks*, 182, 2020.

[44] Manuela-Ioana Csoma, Benjamin Koné, Robert Botez, Iustin-Alexandru Ivanciu, Ahmed Kora, and Virgil Dobrota. Management and orchestration for network function virtualization: An open source MANO approach. In *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2020.

[45] Pierluigi Riti and David Flynn. The HashiCorp Ecosystem. *Beginning HCL Programming: Using Hashicorp Language for Automation and Configuration*, 2021.

[46] PC Amogh, Goutham Veeramachaneni, Anil Kumar Rangisetti, Bheemarjuna Reddy Tamma, and A Antony Franklin. A cloud native solution for dynamic auto scaling of MME in LTE. In *2017 IEEE PIMRC*. IEEE.

[47] Olesia Pozdniakova, Dalius Mažeika, and Aurimas Cholomskis. Adaptive Resource Provisioning and Auto-scaling for Cloud Native Software. In Robertas Damaševičius and Giedrė Vasiljevienė, editors, *Information and Software Technologies*, pages 113–129, Cham, 2018. Springer International Publishing.

[48] Hanqing Zhao, Hyunwoo Lim, Muhammad Hanif, and Choonhwa Lee. Predictive container auto-scaling for cloud-native applications. In *2019 ICTC*. IEEE.

[49] Nicolas Marie-Magdelaine and Toufik Ahmed. Proactive Autoscaling for Cloud-Native Applications using Machine Learning. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020.

[50] David Buchaca, Josep LLuis Berral, Chen Wang, and Alaa Youssef. Proactive Container Auto-scaling for Cloud Native Machine Learning Services. In *IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020.

[51] Sunny Dutta, Tarik Taleb, and Adlen Ksentini. QoE-aware elasticity support in cloud-native 5G systems. In *2016 IEEE International Conference on Communications (ICC)*, 2016.

[52] Virgilios Passas, Nikos Makris, Yue Wang, Apostolos Apostolaras, Asterios Mpatziakas, Anastasios Drosou, Thanasis Korakis, and Dimitrios Tzovaras. Artificial Intelligence for network function autoscaling in a cloud-native 5G network. *Computers and Electrical Engineering*, 103, 2022.

[53] Akrit Mudvari, Nikos Makris, and Leandros Tassiulas. ML-driven scaling of 5G Cloud-Native RANs. In *IEEE GLOBECOM*, 2021.

[54] Tejas Subramanya, Davit Harutyunyan, and Roberto Riggio. Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks. *Computer Networks*, 166, 2020.

[55] Van-Giang Nguyen, Karl-Johan Grinnemo, Javid Taheri, Johan Forsman, Thang Le Duc, and Anna Brunstrom. On Auto-scaling and Load Balancing for User-plane Gateways in a Softwarized 5G Network. In *2021 17th International Conference on Network and Service Management (CNSM)*, 2021.

[56] Irian Leyva-Pupo, Cristina Cervelló-Pastor, Christos Anagnostopoulos, and Dimitrios P. Pezaros. Dynamic UPF placement and chaining reconfiguration in 5G networks. *Computer Networks*, 215:109200, 2022.

[57] Csaba Rotter and Tien Van Do. A Queueing Model for Threshold-Based Scaling of UPF Instances in 5G Core. *IEEE Access*, 9, 2021.

[58] Tulja Vamshi Kiran Buyakar, Anil Kumar Rangisetti, A Antony Franklin, and Bheemarjuna Reddy Tamma. Auto scaling of data plane VNFs in 5G networks. In *2017 13th International Conference on Network and Service Management (CNSM)*, 2017.

[59] James Turnbull. *Monitoring with Prometheus*. Turnbull Press, 2018.

[60] Mainak Chakraborty and Ajit Pratap Kundan. Grafana. In *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*. Springer, 2021.

[61] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Scientific data*, 2(1), 2015.

[62] Ahmed Al-Quzweeni, Ahmed Lawey, Taisir Elgorashi, and Jaafar Elmirghani. Optimized Energy Aware 5G Network Function Virtualization. *IEEE Access*, PP:1–1, 03 2019.

[63] Reza Moosavi, Saeedeh Parsaeefard, Mohammad Ali Maddah-Ali, Vahid Shah-Mansouri, Babak Hossein Khalaj, and Mehdi Bennis. Energy efficiency through joint routing and function placement in different modes of SDN/NFV networks. *Computer Networks*, 200:108492, 2021.

[64] Adriana Fernandez-Fernandez, Cristina Cervello-Pastor, and Leonardo Ochoa-Aday. Achieving Energy Efficiency: An Energy-Aware Approach in SDN. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2016.

[65] Arturo Bellin, Fabrizio Granelli, and Daniele Munaretto. A measurement-based approach to analyze the power consumption of the softwarized 5G core. *Computer Networks*, 244:110312, 2024.

[66] Christian Tipantuña and Xavier Hesselbach. NFV/SDN Enabled Architecture for Efficient Adaptive Management of Renewable and Non-Renewable Energy. *IEEE Open Journal of the Communications Society*, 1:357–380, 2020.

[67] Tarik Taleb, Adlen Ksentini, and Pantelis A. Frangoudis. Follow-Me Cloud: When Cloud Services Follow Mobile Users. *IEEE Transactions on Cloud Computing*, 7(2):369–382, 2019.

[68] Bangbang Ren, Siyuan Gu, Deke Guo, Guoming Tang, and Xu Lin. Joint Optimization of VNF Placement and Flow Scheduling in Mobile Core Network. *IEEE Transactions on Cloud Computing*, 10(3):1900–1912, 2022.

[69] Olesia Pozdniakova, Dalius Mažeika, and Aurimas Cholomskis. SLA-Adaptive Threshold Adjustment for a Kubernetes Horizontal Pod Autoscaler. *Electronics*, 13(7), 2024.

[70] 3rd Generation Partnership Project (3GPP). System architecture for the 5G System (5GS); Stage 2. Technical Report TS 23.501, 3GPP, 2020. Version 16.4.0.

[71] Johanna Andrea Hurtado Sánchez, Katherine Casilimas, and Oscar Mauricio Caicedo Rendon. Deep Reinforcement Learning for Resource Management on Network Slicing: A Survey. *Sensors*, 22(8), 2022.

[72] Imad Alawe, Adlen Ksentini, Yassine Hadjadj-Aoul, and Philippe Bertin. Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach. *IEEE Network*, 32(6):42–49, 2018.

[73] Hatim Chergui and Christos Verikoukis. Big Data for 5G Intelligent Network Slicing Management. *IEEE Network*, 34(4):56–61, 2020.

[74] Alireza Farshin, Tom Barbette, Amir Roozbeh, Gerald Q. Maguire Jr., and Dejan Kostić. PacketMill: toward per-Core 100-Gbps networking. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, page 1–17, New York, NY, USA, 2021. Association for Computing Machinery.

[75] Darijo Raca, Dylan Leahy, Cormac J. Sreenan, and Jason J. Quinlan. Beyond Throughput, the Next Generation: A 5G Dataset with Channel and Context Metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20)*, pages 303–308. ACM, 2020.

[76] Ronald Schafer. What Is a Savitzky-Golay Filter? [Lecture Notes]. *IEEE Signal Processing Magazine - IEEE SIGNAL PROCESS MAG*, 07 2011.

[77] Harsh Kumar, Hassan M. Al-Jawahry, Murli Manohar, K Karthika, Melvin Victor, and Aishwarya D. Shetty. Real-time Traffic Prediction in 5G Networks Using LSTM Networks. In *2024 IEEE International Conference on Communication, Computing and Signal Processing (IICCCS)*, pages 1–5, 2024.

[78] Edmund Pok Lens Shiang, Wei-Che Chien, Chin-Feng Lai, and Han-Chieh Chao. Gated Recurrent Unit Network-based Cellular Trafile Prediction. In *2020 International Conference on Information Networking (ICOIN)*, pages 471–476, 2020.

[79] N. Deeban and P. Shyamala Bharathi. A Robust and Efficient Traffic Analysis for 5G Network Based on Hybrid LSTM comparing with XGBoost to Improve Accuracy. In *2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF)*, pages 1–8, 2023.

[80] Maryam Mohseni, Soodeh Nikan, and Abdallah Shami. AI-based Traffic Forecasting in 5G network. In *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 188–192, 2022.

[81] Ilias Syrigos, Dimitris Kefalas, Nikos Makris, and Thanasis Korakis. EELAS: Energy Efficient and Latency Aware Scheduling of Cloud-Native ML Workloads. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 819–824, 2023.

[82] Chester Lam. Alder Lake's Power Efficiency – A Complicated Picture. Chips and Cheese, https://chipsandcheese.com/p/alder-lakes-power-efficiency-a-complicated-picture, January 2022. Accessed: Jul. 2026.

[83] Intel Corporation. Intel Core i7-12700K Processor (25M Cache, up to 5.00 GHz) – Product Specifications. https://www.intel.com/content/www/us/en/products/sku/134594/intel-core-i712700k-processor-25m-cache-up-to-5-00-ghz/specifications.html, 2021. Accessed: Jul. 2026.

[84] FD.io VPP Project. Interface Configuration: set interface rx-mode. FD.io VPP Documentation, [Online]. Available: https://docs.fd.io/vpp/17.10/clicmd_src_vnet.html, 2017.

[85] Cisco Systems. Variable Relinquishing of CPU Cycles. UCC 5G UPF Configuration and Administration Guide, Release 2024.04, [Online]. Available: https://www.cisco.com/c/en/us/td/docs/wireless/ucc/upf/2024-04/config-admin/ucc-5g-upf-config-and-admin-guide-2024-04/m-power-savings-using-vrcc.html, 2024.

[86] LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); LTE-WLAN Aggregation Adaptation Protocol (LWAAP) specification. Technical Report TS 36.360, 3rd Generation Partnership Project (3GPP), October 2025.

[87] A. Ford et al. RFC 8684: TCP Extensions for Multipath Operation with Multiple Addresses, 2020.

[88] Y. Liu, Y. Ma, et al. Managing multiple paths for a QUIC connection. Internet-Draft draft-ietf-quic-multipath-21, IETF, March 2026. Work in Progress.

[89] Carlos Barjau et al. Limitations of ATSSS technology in ATSC 3.0 – 5G convergent systems. In *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5, 2021.

[90] Dimitris Kefalas et al. Design, Integration and Slicing of non-3GPP DUs in 5G Networks: a Hybrid 5G-DU Approach. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, Vancouver, Canada, May 2024. IEEE.

[91] Dimitris Kefalas et al. Traffic Steering for O-RAN Multi-RAT 5G Networks Using ML-Based Demand Forecasting. In *2025 IEEE Conference on Network Function Virtualization and Software-Defined Networking (NFV-SDN)*, pages 1–6, 2025.

[92] Xinran Ba et al. Performance Evaluation of Multi-Access Based on ATSSS Rules. In *2022 International Conference on Culture-Oriented Science and Technology (CoST)*, pages 390–394, 2022.

[93] M. Quadrini et al. Implementation and Testing of MP-TCP ATSSS in a 5G Multi-Access Configuration. In *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, 2023.

[94] S Thomas Valerrian Pasca et al. A real-time performance evaluation of tightly coupled LTE Wi-Fi radio access networks. In *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, page 1–6. IEEE Press, 2017.

[95] Khizar Abbas et al. An LTE-WiFi Spectrum Aggregation System for 5G Network: A Testbed. In *2020 International Conference on Information Networking (ICOIN)*, pages 753–755, 2020.

[96] Rui Wang et al. Potentials and Challenges of C-RAN Supporting Multi-RATs Toward 5G Mobile Networks. *IEEE Access*, 2:1187–1195, 2014.

[97] Yoohwa Kang et al. Multipath transmission control protocol–based multi-access traffic steering solution for 5G multimedia-centric network: Design and testbed

system implementation. *International Journal of Distributed Sensor Networks*, 16:155014772090975, 02 2020.

[98] Matan Broner et al. 5G-MANTRA: Multi-Access Network Testbed for Research on ATSSS. In *Proceedings of the 3rd ACM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*, 5G-MeMU '23, page 8–13, New York, NY, USA, 2023. Association for Computing Machinery.

[99] Tim Farnschläder et al. On the Interplay of 5G NR, WiFi and LiFi Towards 6G Multi-Connectivity. In *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 1009–1014, 2024.

[100] Konstantin Schneider, Sebastian Treib, Maximilian Dietrich, Wolfgang Kiess, and Eike Lyczkowski. MultiNet6G: 5G NR, WiFi and Visible Light Communication Working as Single Network. In *IEEE Conference on Standards for Communications and Networking (CSCN)*, Bologna, Italy, 2025.

[101] Quentin De Coninck et al. MultipathTester: Comparing MPTCP and MPQUIC in Mobile Environments. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 221–226, 2019.

[102] Line M. P. Larsen et al. A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks. *IEEE Communications Surveys & Tutorials*, 21(1):146–172, 2019.

[103] Christian Huitema. Picoquic: A Small QUIC Implementation. GitHub repository, Private Octopus Inc., 2017. Accessed: 2025.

[104] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021.

[105] Yuri Demchenko et al. SLICES Data Management Infrastructure for Reproducible Experimental Research on Digital Technologies. In *2023 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2023.

[106] Qingxiu Dong et al. A Survey on In-context Learning, 2023.

[107] Yunfan Gao et al. Retrieval-Augmented Generation for Large Language Models: A Survey, 2024.

[108] Cheonsu Jeong. Fine-tuning and Utilization Methods of Domain-specific LLMs, 2024.

[109] Spyros Kamnis. Generative pre-trained transformers (GPT) for surface engineering. *Surface and Coatings Technology*, 466:129680, 2023.

[110] Bryan Wang et al. LAVE: LLM-Powered Agent Assistance and Language Augmentation for Video Editing, 2024.

[111] Lei Wang et al. A Survey on Large Language Model based Autonomous Agents, 2024.

[112] Xin Luna Dong et al. Towards Next-Generation Intelligent Assistants Leveraging LLM Techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2023.

[113] SLICES-SC. SLICES Scientific Community Documentation. `https://doc.slices-sc.eu/`, 2023. Accessed: 2024-04.

[114] OpenAI. Introducing ChatGPT. `https://openai.com/blog/chatgpt`. Accessed: 2024-04.

[115] OpenAI. GPT-4. `https://openai.com/gpt-4`. Accessed: 2024-04.

[116] Jerry Liu. LlamaIndex, 11 2022.

[117] OpenAI. Preparing Your Dataset for Fine-Tuning. `https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset`. Accessed: 2024-04.

[118] Haihua Chen, Jiangping Chen, and Junhua Ding. Data Evaluation and Enhancement for Quality Improvement of Machine Learning. *IEEE Transactions on Reliability*, 70(2):831–847, 2021.

[119] Explosion AI. en_core_web_lg: English pipeline optimized for CPU. `https://spacy.io/models/en#en_core_web_lg`, 2023. Accessed: 2023-04.

[120] Dimitrios Kefalas et al. slAIces Chatbot Questions Repository, 2024.

[121] SLICES Starting Community. SLICES-RI Blueprint. https://doc.slices-sc.eu/blueprint/, 2023. Accessed: 2024-04.