Nouvelles stratégies de mise en cache et de mobilité intelligentes pour MEC / architectures basées ICN

Sarra Mehamel

le cnam

École doctorale Informatique, Télécommunications et Électronique (Paris)

Centre d'Études et de Recherche en Informatique et Communications

Faculté de génie électrique et informatique

Laboratoire de Recherche en Informatique

THÈSE DE DOCTORAT

présentée par : **Sarra MEHAMEL**

soutenue le : **04 Novembre 2020**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers et de l'Université Mouloud MAMMERI de Tizi-Ouzou**

Spécialité : **Informatique**

New intelligent caching and mobility strategies for MEC /ICN based architectures

THÈSE dirigée par

Mme Samia BOUZEFRANE	*Professeure, Cnam, France*
M. Mehammed DAOUI	*Professeur, UMMTO, Algérie*

RAPPORTEURS

M. Bouabdellah KECHAR	*Professeur, Université d'Oran 1, Algerie*
M. Nadjib ACHIR	*Maître de Conférences/HDR, Institut Galilée Université Paris 13, France*

EXAMINATEURS

Mme. Malika BELKADI	*Maître de Conférences/HDR, UMMTO, Algérie*
M. Hassine MOUNGLA	*Maître de Conférences/HDR, Université Paris Descartes, France*

Abstract

Mobile edge computing (MEC) concept proposes to bring the computing and storage resources in close proximity to the end user by placing these resources at the network edge which could be any type of base station in the network. The motivation is to alleviate the mobile core and to reduce latency for mobile users due to its close proximity. MEC servers are candidates to host mobile applications and serve web contents. Edge caching is one of the most emerging technologies recognized as a content retrieval solution in the edge of the network. It has been also considered as enabling technology of mobile edge computing that presents an interesting opportunity to perform caching services. Particularly, the MEC servers are implemented directly at the base stations which enable edge caching and ensure deployment in close-proximity to the mobile users. However, the integration of servers in mobile edge computing environment (base stations) complicates the energy saving issue because the power consumed by mobile edge computing servers is costly especially when the load changes dynamically over time. Furthermore, users with mobile devices arise their demands, introducing the challenge of handling such mobile content requests beside the limited caching size. Thus, it is necessary and crucial for caching mechanisms to consider the mentioned factors, meanwhile most existing studies focus on cache allocation, content popularity and cache design. In this thesis, we present a novel energy-efficient fuzzy caching strategy for edge devices that takes into consideration four influencing features of mobile environment, while introducing a hardware implementation using Field-Programmable Gate Array (FPGA) to cut the overall energy requirements. Performing an adequate caching strategy on MEC servers opens the possibility of employing artificial intelligence (AI) techniques and machine learning at mobile network edges. Exploiting users context information intelligently makes it possible to design an intelligent context-aware mobile

edge caching. Context awareness enables the cache to be aware of its environment, while intelligence enables each cache to make the right decisions of selecting appropriate contents to be cached so that to maximize the caching performance. Inspired by the success of reinforcement learning (RL) that uses agents to deal with decision making problems, we extended our fuzzy-caching system into a modified reinforcement learning model. The proposed framework aims to maximize the cache hit rate and requires a multi awareness. The modified RL differs from other RL algorithms in the learning rate that uses the method of stochastic gradient decent beside taking advantage of learning using the optimal caching decision obtained from fuzzy rules.

Keywords : Information Centric Network, Caching, Mobile Edge Computing, Reinforcement Learning, Fuzzy Logic, FPGA.

Résumé

Le paradigme de MEC (Mobile Edge Computing) repose sur le placement des ressources de calcul et de stockage aux « extrémités » du réseau à proximité des utilisateurs finaux. Le terme « edge » désigne n'importe quel type de station de base du réseau. Les motivations pour l'adoption de ce nouveau concept sont principalement la réduction de la charge au cœur du réseau et la diminution de la latence grâce à la proximité des ressources afin d'améliorer l'expérience utilisateur. Les serveurs MEC sont de bons candidats pour héberger les applications mobiles et diffuser le contenu Web. La mise en cache à l'extrémité du réseau, ou Edge Caching en anglais, est l'une des technologies les plus émergentes connues comme solution de récupération de contenus au bord du réseau. Elle est aussi considérée comme une technologie permettant la mise en place du concept MEC puisqu'elle présente une opportunité intéressante pour implémenter les services de mise en cache. En particulier, les serveurs MEC sont implémentés directement au niveau des stations de base, ce qui permet la mise en cache à l'extrémité du réseau et le déploiement à proximité des utilisateurs finaux. Cependant, l'intégration des serveurs MEC dans les stations de base complexifie le problème de la consommation énergétique parce que l'environnement est dynamique et sujet à des changements au fil du temps. Par ailleurs, la demande des utilisateurs des appareils mobiles est en constante augmentation notamment en termes d'expérience utilisateur. Sachant que la taille des caches est limitée, il devient nécessaire et crucial que les mécanismes de mise en cache soient en mesure de faire face à cette situation et de proposer des solutions efficaces et satisfaisantes à long terme. La plupart des études existantes se sont focalisées sur l'allocation de cache, la popularité du contenu ou encore la manière de concevoir le cache. Dans cette thèse, nous présentons une nouvelle stratégie de mise en cache écoénergétique basée sur la logique floue. Notre proposition

prend en compte les quatre caractéristiques d'un environnement mobile et introduit une implémentation matérielle en utilisant les FPGA (Field-Programmable Gate Array) pour réduire les contraintes énergétiques. L'adoption d'une stratégie de mise en cache adéquate sur les serveurs MEC ouvre la possibilité d'utiliser des techniques d'intelligence artificielle et d'apprentissage automatique aux extrémités des réseaux mobiles. L'exploitation des informations de contexte des utilisateurs permet de concevoir une mise en cache intelligente sensible au contexte. La connaissance du contexte permet au gestionnaire de cache de tenir compte de l'environnement, tandis que l'intelligence lui permet de prendre les bonnes décisions pour la mise en cache afin d'optimiser les performances.

Inspiré par le succès de l'apprentissage par renforcement utilisant des agents pour traiter des problèmes de prise de décision, nous avons étendu notre système de mise en cache basé sur la logique floue à un modèle d'apprentissage par renforcement modifié. La méthode d'apprentissage par renforcement modifiée diffère des autres algorithmes par le taux d'apprentissage qui utilise la méthode du gradient stochastique en plus de tirer parti de l'apprentissage en utilisant la décision de mise en cache optimale obtenue à partir des règles de la logique floue.

Mots clés : Mise en cache, informatique mobile de proximité, Réseaux de contenus, logique floue, FPGA, apprentissage par renforcement.

Page de remerciements :

Remerciements

Acknowledgement

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Samia Bouzerfane for the continuous support, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor. I would like also to express my deepest thanks to my advisors Prof. Mehammed Daoui and Dr Soumya Banerjee for their dedicated support, help and guidance.

Besides, I would like to thank the members of my thesis committee, who took time to report my thesis and to be part of my jury support, namely Dr Malika Belkaid, Dr. Nadjib Achir, Prof. Bouabdellah Kechar and Dr. Hassine Moungla.

My sincere thanks also goes to my father Djamel who taught me once that our greatest weakness lies in giving up and the most certain way to succeed is always to try just one more time. My mom Fahima who have been always supporting me no matter what. My grandmother, sisters khawla, Aya and Ibtihel and my little brothers Adem and Abd elmoudjib who were the happiness in my dark days. My husband Billel,for his encouragement and sacrifice without which I could not have strengths to overcome all the difficulties during my overseas study time. My father in law Laid Lasledj and my mother in law Houria and my sister in law Hadjer for their endless love.

Finally a big thanks to everyone contributed from near or far especially my friends Hasna,Ahlem, Imene and Younes, my collegues Nacer,Lyes and Khaled for their encouragements.

Contents

List of Tables

List of Figures

Résumé de la Thèse

Nouvelles stratégies intelligentes de gestion de cache et de mobilité pour les architectures MEC utilisant des réseaux de contenus

Les appareils mobiles sont aujourd'hui des plates-formes incontournables pour offrir des services et des contenus divers à l'utilisateur. De plus, les clients sont devenus de plus en plus exigeants en termes de consommation de contenus mais aussi en matière de sécurité et de qualité du service. L'informatique distante ou mobile a été initialement bâtie sur le modèle client-serveur impliquant une hiérarchie à deux niveaux. L'évolution de l'informatique a permis l'introduction du concept de "cloud" pour désigner une collection de serveurs disposant de ressources informatiques ou de machines virtuelles, sur lesquelles tourne tout calcul déporté par un client mobile. C'est ce qu'on appelle le cloud mobile. Le cloud mobile prend en compte divers facteurs liés à la mobilité contrairement aux techniques traditionnelles client-serveur, tels que l'énergie des appareils, le coût d'utilisation de la bande passante, la connectivité réseau, la mobilité, la sensibilité au contexte et à la localisation [92][102].

Malgré les avantages du cloud mobile, il reste un certain nombre de défis à relever dus à l'éloignement des serveurs cloud des équipements mobiles. Ce qui est un frein pour avoir de bonnes performances et un temps de réponse acceptable pour l'utilisateur. C'est ainsi que l'idée est venue de déporter le calcul de l'utilisateur sur de petits data centers qui seraient déployés à proximité de l'utilisateur, c'est-à-dire, au sein des stations de base. Ce nouveau paradigme est ce qu'on appelle en anglais MEC (Mobile Edge Computing) et qu'on peut traduire comme l'infrastructure réseau mobile de proximité. La technologie MEC est aujourd'hui une solution prometteuse pour satisfaire différentes exigences telles que : une

faible latence, la proximité de l'utilisateur, une bande passante élevée, des informations fournies en temps réel et la prise en compte de la localisation de l'utilisateur. La technologie MEC est perçue aujourd'hui comme l'une des technologies clés favorisant l'émergence de la nouvelle génération de réseaux mobiles comme la 5G. La demande croissante de services sur le réseau mobile soulève plusieurs problèmes comme la capacité de stockage et la bande passante. Dans l'architecture traditionnelle des réseaux mobiles, les demandes des utilisateurs sont traitées par des fournisseurs de contenus traversant l'ensemble du réseau mobile, ce qui mène à une congestion du réseau et donc à un temps d'accès aux services élevé. L'architecture MEC offre à l'utilisateur des ressources cloud qui seraient hébergées à l'extrémité du réseau afin de rapprocher les contenus de l'utilisateur mobile. Ces extrémités réseau peuvent être des passerelles, des stations de base ou même des équipements mobiles de l'utilisateur. Au niveau de ces extrémités, se trouvent des serveurs MEC qui traitent et stockent des contenus utilisateurs évitant ainsi la congestion et la latence réseau. En exécutant des analyses et des mises en cache sur les serveurs MEC, le volume de données transmis au coeur du réseau pour traitement ainsi que l'impact du trafic de données à travers le réseau sont réduits. Cependant, les caches ont été conçus pour permettre une communication fluide et rapide, ainsi qu'une distribution des contenus entre les serveurs MEC et les équipements des utilisateurs. Cependant, ces caches ne permettent de diffuser que des contenus populaires autour du réseau sans tenir compte d'autres facteurs comme la mobilité des utilisateurs.

Le paradigme traditionnel de l'Internet présente plusieurs limites. Tout d'abord, bien qu'Internet fournisse des contenus multiples et différents, ces contenus ne sont pas liés entre eux. D'autre part, il existe une relation forte entre le contenu et son emplacement, ce qui a une forte incidence sur la qualité de service. En effet, à chaque fois qu'un contenu est demandé, une adresse spécifique liée à l'emplacement du contenu est nécessaire, alors que l'objectif principal est d'obtenir le contenu le plus rapidement possible quelque soit son emplacement. Pour résoudre ce problème de manière explicite, il faut remplacer où (emplacement) par quoi (information). Sur la base de cette prémisse, les réseaux de contenus ICN (pour Information Centric Networking en anglais) proposent un changement radical avec un modèle centré sur l'hôte en se concentrant sur ce qui est livré ou fourni. Aujourd'hui,

les utilisateurs consomment des contenus web en grande quantité, en particulier des vidéos. Les statistiques de *YouTube* dans le monde selon les prévisions de 2016 à 2021 montrent que le nombre de visionneuses de plates-formes vidéo en ligne s'élèvera à 1,86 milliard en 2021, contre 1,47 milliard en 2017 [1]. Sur Netflix, le nombre prévu de souscriptions pour un abonnement à la vidéo à la demande à travers le monde passerait de 250 millions en 2018 à 450 millions en 2022 [2]. Cela ouvre la voie à un problème sérieux où la congestion du réseau devient incontrôlable et où de nouveaux mécanismes sont nécessaires pour répondre aux exigences du réseau et des utilisateurs.

Dans ce contexte, le stockage de contenus temporaires sur plusieurs serveurs du réseau a été utilisé pour répondre aux demandes des utilisateurs. Ce stockage temporaire est exécuté sur un système appelé système de mise en cache. Le concept de mise en cache n'est pas nouveau dans les réseaux cellulaires. Il a été utilisé dans la mise en cache Web mais aussi dans les réseaux de contenus qui visent à améliorer l'évolutivité du réseau, en mettant en cache les contenus dans des serveurs proxy ou dans des noeuds intermédiaires du réseau. Le déploiement de caches à très grande échelle augmente l'espace de stockage disponible. Mais en même temps, cela augmente la complexité de gestion de ces caches qui nécessitent des stratégies capables de déterminer où et quand le contenu doit être mis en cache tout en tenant compte de sa popularité. Cependant, la croissance émergente de la demande des utilisateurs et du paradigme Internet exige une gestion des caches qui soit sensible aux contextes multiples, offrant ainsi la possibilité de définir de meilleures stratégies.

À l'heure où Internet est inondé par les données, le stockage des contenus populaires à l'extrémité du réseau semble être une technique prometteuse pouvant satisfaire les exigences de l'utilisateur tout en atténuant la surpopulation du coeur de réseau. À cet effet, les serveurs MEC équipés de caches locaux peuvent être utilisés intelligemment. Par ailleurs, les progrès techniques récents et le développement d'applications MEC ont favorisé la croissance du trafic de données puisque les services MEC utilisent des données massives issues de domaines variés. En effet, les serveurs MEC sont équipés de capacités de calcul et de stockage, permettant le traitement de quantités importantes de requêtes et de contenus,

[1] https://www.statista.com/statistics/805656/number-youtube-viewers-worldwide/
[2] https://www.statista.com/statistics/322737/svod-households-worldwide/

améliorant ainsi la qualité de service (QoS) du réseau et la qualité d'expérience (QoE) de l'utilisateur. Les caches locaux font partie des équipements MEC qui aident à réduire le trafic et à améliorer le temps de réponse. En particulier, la mise en cache au niveau des extrémités du réseau permet d'accélérer la récupération des données et de réduire le trafic sur le réseau, car les requêtes des utilisateurs peuvent être satisfaites par les noeuds extrémités les plus proches de l'utilisateur. Cependant, la mise en cache dans les extrémités du réseau est considérée comme un goulot d'étranglement pour le développement des systèmes MEC, car les caches de proximité ont des capacités de stockage limitées. Ces caches utilisent en général différentes stratégies de placement et de remplacement, qui sont généralement conçues pour permettre la diffusion de contenus populaires sur le réseau. Répondre aux exigences du réseau et de l'utilisateur est un point clé dans les réseaux MEC car il s'agit d'une part de déléguer la gestion de caches aux extrémités du réseau et d'autre part de prendre une décision qui soit optimale dans tout le réseau. En d'autres termes, comment résoudre le problème du réseau stochastique-dynamique tout en améliorant la décision optimale de mise en cache

Plusieurs travaux pionniers sur la gestion des caches dans les réseaux MEC ont été proposés et donnent de très bons résultats. Cependant, ces travaux présentent les inconvénients suivants :

- Entrées insuffisantes : les travaux existants considèrent uniquement la popularité comme facteur d'information clé, alors qu'en fait il y a plusieurs facteurs qui influent sur la décision de mise en cache.

- Combinaison des stratégies : Ils considèrent soit des stratégies de placement, soit des stratégies de remplacement, mais ne considèrent pas les deux types de stratégies à la fois.

- Conditions dynamiques : la dynamique de l'environnement et le système de mise en cache ne sont pas bien traités.

- Isolement indicatif : La plupart des stratégies de gestion de caches ne tiennent pas compte de l'effet à long terme de la décision actuelle de mise en cache.

D'autres travaux de la littérature proposent des solutions optimales, mais elles ne sont pas suffisamment intelligentes pour adapter la décision en fonction de l'évolution du système. Afin de combler cette lacune, dans cette thèse, nous nous concentrons sur la conception de nouvelles stratégies de mise en cache qui s'appuient sur les méthodes d'apprentissage automatique. En particulier, nous nous intéresserons à l'apprentissage par renforcement qui permet aux agents de prendre des décisions en apprenant à l'aide d'interactions avec l'environnement [104].

La principale contribution de cette thèse est de formuler un système de décision intelligent pour la gestion de caches dans le cas où les serveurs MEC distribués de façon stochastique sont équipés de caches à capacité limitée. Notre travail diffère des travaux existants dans le sens où il propose un système qui intègre à la fois le placement et le remplacement de contenus pour une architecture MEC tout en reposant sur un modèle d'apprentissage par renforcement modifié.

Cette thèse comporte trois parties divisées en 6 chapitres. Chaque chapitre est résumé comme ci-dessous: le premier chapitre décrit le paradigme MEC qui est actuellement en cours de standardisation par l'ETSI (ISG) [3].

Le concept de "edge computing" permet d'optimiser l'utilisation du cloud computing. Plutôt que de transférer les données générées par des appareils connectés vers le Cloud, il s'agit de traiter les données en périphérie du réseau directement là où elles sont générées. Cette méthode réduit les besoins en bande passante des communications entre les capteurs et le centre de traitement des données en entreprenant les analyses et les connaissances au plus près de la source des données. En d'autres termes, les données sont traitées directement par le périphérique qui les génère (objet connecté, smartphone, etc.) ou par un ordinateur/serveur local. Jusqu'à récemment, le concept de "edge computing" a permis principalement d'ingérer, de stocker, de filtrer et d'envoyer des données sur des systèmes dans le cloud. Toutefois, nous sommes aujourd'hui dans la situation où ces systèmes possèdent plus de puissance de calcul, de stockage et d'analyse pour consommer et agir sur les données dans le réseau machine. Un facteur de croissance important de la technologie "edge computing" est le besoin croissant de systèmes de communication fonctionnant en

[3] https://www.etsi.org/technologies/multi-access-edge-computing

temps réel. Mobile edge computing est une application du "edge computing". C'est un concept d'architecture en réseau qui offre les capacités du cloud computing dans un environnement de service informatique en périphérie du réseau cellulaire. L'architecture MEC fournit un environnement informatique et des ressources cloud placées à l'extrémité du réseau mais aussi au sein du réseau d'accès radio. L'objectif étant de réduire la latence, d'assurer un fonctionnement correct et une prestation de services efficace de manière à offrir une meilleure expérience à l'utilisateur. Contrairement aux réseaux mobiles traditionnels, les réseau de sorte que le contenu soit plus proche de l'utilisateur. En effectuant des mises en cache des contenus sur les serveurs MEC, le volume de données transmis au coeur du réseau est réduit. Ce qui empêche la congestion et la latence des applications qui tournent sur les serveurs MEC [61]. En d'autres termes, la conception de l'Internet actuel crée des problèmes que l'utilisateur ne peut pas supporter, comme les problèmes de sécurité ou la sensibilité au contexte. Les différents défis tels que la mobilité, la sécurité, la mise en cache et la QoS ont donné lieu à de nouvelles exigences. De plus, les utilisateurs commencent à se concentrer sur l'information désirée, ce qui augmente la nécessité de recentrer la communication sur l'accès au contenu plutôt que sur l'interaction hôte-hôte. Les chercheurs estiment que les réseaux de contenus et les architectures MEC peuvent jouer un rôle clé dans la croissance universelle de l'Internet en traitant les contenus comme l'entité de première classe dans l'architecture réseau et en étant en mesure de stocker le contenu à différents endroits mais à proximité de l'utilisateur.

Le deuxième chapitre décrit les pré-requis de cette thèse. Le développement des nouvelles technologies de communication a considérablement accéléré la livraison de contenus et a amélioré la qualité de l'expérience. En apportant plus d'intelligence dans les prises de décision lors du stockage de contenus en général et dans les architectures MEC en particulier, on pourrait fournir des services efficaces dans un environnement mobile en pleine mutation. Nous présentons dans ce chapitre deux techniques que nous avons utilisées dans nos contributions : La logique floue et l'apprentissage par renforcement.

L'expression "logique floue" recouvre principalement un ensemble de modèles et de techniques mathématiques, qui sont basées sur la notion de (sous)-ensemble flou. Un sous-ensemble flou est décrit sur un référentiel donné par une fonction d'appartenance à

valeur sur une échelle totalement ordonnée, bornée supérieurement et inférieurement. On emploie le plus souvent en pratique l'intervalle réel [0, 1], mais cette échelle peut aussi ne comporter qu'un nombre fini de niveaux . La construction d'un système de logique floue consiste à sélectionner les entrées et sorties du système avant de les découper au sein de catégories adéquates nommées ensembles flous. La prise de décisions en utilisant une logique floue a été utilisée pour de nombreux objectifs, comme pour réduire les coûts et maximiser la qualité du service.

L'apprentissage par renforcement (Reinforcement Learning en anglais) est une méthode d'apprentissage pour les modèles de Machine Learning. Pour faire simple, cette méthode consiste à laisser l'algorithme apprendre de ses propres erreurs. Afin d'apprendre à prendre les bonnes décisions, l'algorithme se retrouve directement confronté à des choix. S'il se trompe, il est "pénalisé". Au contraire, s'il prend la bonne décision, il est "récompensé". Afin d'obtenir toujours plus de récompenses, l'algorithme va donc faire de son mieux pour optimiser sa prise de décision. L'apprentissage par renforcement est né de la rencontre entre la psychologie expérimentale et les neurosciences computationnelles. Il tient en quelques concepts clés simples basés sur le fait que l'agent intelligent:

- Observe les effets de ses actions.
- Déduit de ses observations la qualité de ses actions.
- Améliore ses actions futures.

L'idée de base de l'apprentissage par renforcement est de changer la base des agents programmés à l'aide d'un système de récompense et de punition dans le but de résoudre le problème auquel sont confrontés les agents, où ils doivent apprendre un comportement et des actions spécifiques dans un environnement dynamique et stochastique.

Nous avons dédié le troisième chapitre aux mécanismes de gestion de caches. Un cache est une zone où sont stockées des copies de données qui seront trouvables plus simplement et rapidement par une application. Le but du cache est donc d'optimiser l'accès aux données afin de gagner en vitesse d'affichage ou encore pour limiter une quantité de données stockées en mémoire.

La mise en cache est l'une des technologies les plus émergentes reconnue comme solution de récupération de contenus aux extrémités du réseau. Elle a également été considérée comme une technologie qui offre une opportunité intéressante d'effectuer des services de mise en cache. En particulier, les serveurs MEC sont implantés directement sur les stations de base qui permettent la mise en cache et assurent le déploiement à proximité des utilisateurs mobiles. Ce chapitre présente un aperçu général de la mise en cache, illustre les principales stratégies de mise en cache. Il contient également une description des mesures de rendement ainsi que les critères de base utilisés couramment pour comparer les mécanismes de mise en cache existants. Enfin, nous présentons les principes de base de la logique floue et de l'apprentissage par renforcement qui seront utilisés dans cette thèse.

Même si le placement des caches sur les serveurs MEC semble intéressant en vue de rapprocher les contenus à proximité de l'utilisateur, cela peut poser le problème de la consommation énergétique que ça soit sur les data-centers que ça soit sur les équipements mobiles de l'utilisateur en raison notamment de la taille limitée des caches. Ainsi, il est crucial que les mécanismes de mise en cache proposés tiennent compte de la dimension énergétique mais aussi d'autres facteurs comme la taille du cache, la popularité du contenu, etc.

Ainsi, dans le quatrième chapitre, nous proposons une stratégie de mise en cache efficace en terme d'énergie pour les extrémités MEC, qui prend en compte quatre caractéristiques influençant l'environnement mobile, tout en introduisant une implémentation matérielle à l'aide de FPGA (Field-Programmable Gate Array) pour réduire la consommation énergétique. Dans ce chapitre, nous avons présenté le modèle de logique floue envisagé. Notre nouveau système de mise en cache de proximité, combine la taille, la mobilité et les coûts en utilisant la logique floue. Les tests menés sur notre solution montrent que l'approche proposée améliore le taux de succès et qu'une implémentation basée sur du matériel (FPGA) réduit considérablement la consommation énergétique. Un FPGA sert à implémenter un système numérique, même si un simple microcontrôleur peut souvent faire l'affaire. Les microcontrôleurs sont peu coûteux et faciles à monter sur une carte à circuit imprimé. Les FPGA sont de puissants outils, mais ne conviennent pas forcément à toutes les applications. Ils présentent davantage d'exigences concernant la puissance, la configuration et le circuit

externe, et ces exigences supplémentaires peuvent entraîner un coût prohibitif. Le fait d'utiliser un FPGA beaucoup plus cher et qui implique tout un tas d'exigences spécifiques peut paraître absurde. Après la migration de l'algorithme principal vers le matériel à l'aide de FPGA, nous avons remarqué que le matériel ne consomme que 12 cycles d'horloge. Ce qui peut être considéré comme un avantage par rapport au rôle critique du cache dans le bord du réseau. Nous avons mesuré la consommation d'énergie, les propriétés thermiques, la tension ainsi que le courant électrique lors de l'exécution du système de décision de mise en cache floue. Nous avons remarqué qu'une puissance de P=0,45W est consommée lors de la prise de décision de mise en cache. Contrairement à la stratégie RLU qui consomme P=0,9W, nous avons donc déduit que la stratégie de mise en cache basée sur la logique floue consomme moins d'énergie.

Avec la déportation du calcul sur les extrémités du réseau, il devient pertinent d'accomplir des tâches spécifiques comme l'analyse de données et l'apprentissage automatique au niveau de ces extrémités qui sont dotées de ressources dont les capacités ne sont pas négligeables. L'exploitation intelligente des informations contextuelles des utilisateurs permet de concevoir une mise en cache intelligente qui prend en compte le contexte. La prise en compte du contexte permet au cache de tenir compte de l'environnement et de son évolution, tandis que l'intelligence permet à chaque cache de prendre les bonnes décisions en sélectionnant le contenu approprié à mettre en cache afin d'optimiser les performances.

Afin d'augmenter le taux de succès de la mémoire cache, les études sont destinées à être appropriées pour différentes topologies de réseau, mais elles ne considèrent ni les propriétés du contenu lui-même comme la taille, ni les facteurs d'influence et les caractéristiques de l'utilisateur final comme le coût et la mobilité. Ils ne font qu'accumuler des contenus avec une grande popularité ou une fréquence élevée passée, ce qui peut ne plus être utile au fil du temps. Pour ajuster de manière adaptative les propriétés de contenu variées et leurs facteurs d'influence, nous présentons un système de cache de contrôle flou pour les serveurs de bord qui peuvent choisir le contenu le plus prioritaire à mettre en cache en fonction de facteurs bien choisis. Pour décider s'il convient de mettre en cache ou d'expulser des contenus, le processus décisionnel repose sur les éléments suivants: mobilité, fréquence, occupation de cache et coût de récupération.

Inspirés par le succès de l'apprentissage par renforcement qui utilise des agents pour gérer les problèmes de prise de décision, nous présentons dans le cinquième chapitre un modèle d'apprentissage par renforcement adapté à la mise en cache de contenus dans une infrastructure MEC. La solution proposée vise à maximiser le taux de succès de mise en cache en s'appuyant sur une connaissance multiple des facteurs influents sur les performances du cache. L'apprentissage par renforcement proposé ici diffère des autres algorithmes d'apprentissage dans la mesure où le taux d'apprentissage utilise la méthode de gradient stochastique [18][59] et profite de l'apprentissage en utilisant la décision optimale de mise en cache obtenue à partir de règles de logique floue. Dans ce contexte, nous définissons un système de mise en cache modifié basé sur l'apprentissage par renforcement (mRL) pour MEC, qui rejette les caractéristiques de contenu, à savoir la fréquence, le coût et la taille, ainsi que la fonctionnalité de l'appareil comme la mobilité et la capacité de stockage. Cela vise à résoudre le problème de la mise en cache de manière réaliste. La principale contribution de cet article est d'apporter une solution aux problèmes de mise en cache dans un scénario où les unités de mise en cache sont distribuées de manière stochastique par l'intermédiaire de serveurs MEC avec une capacité de stockage limitée. En particulier, nous nous basons sur un modèle de système de mise en cache et définissons ses mesures de performance (taux de réussite et stabilité du cache). De plus, nous définissons la taille du stockage du cache, la mobilité de l'utilisateur et la distribution de la popularité du contenu. En associant le problème de la décision de mise en cache avec les niveaux d'apprentissage de renforcement, tout en s'appuyant sur les résultats récents de [61], nous montrons qu'un certain taux de réussite peut être atteint en augmentant la taille totale de stockage pendant que le nombre de MECs est fixé. Enfin, nous présentons un schéma de décision réaliste pour le scénario de mise en cache en utilisant un modèle d'apprentissage par renforcement modifié.

Dans le sixième chapitre, nous avons présenté une simulation pour notre système de mise en cache en utilisant ICARUS, qui est un simulateur de mise en cache basé sur Python pour les réseaux de contenus. L'outil de simulation nous permet d'évaluer notre solution de mise en cache pour toute implémentation de type de réseau ICN (Information Centric Networking en anglais).

Les réseaux ICN définissent un paradigme dans lequel les contenus échangés sont au cœur du fonctionnement du réseau, depuis le nommage des paquets jusqu'au routage. Apparues en 2006, les solutions actuelles sont suffisamment matures pour envisager leur déploiement et c'est notamment le cas de NDN (Named Data Networking), architecture largement étudiée dans la littérature scientifique qui possède une implémentation fonctionnelle et maintenue. Toutefois, pour un opérateur de l'Internet, envisager de déployer une telle solution protocolaire n'est pas envisageable pour deux raisons. La première est le coût associé à son déploiement qui nécessite un changement d'infrastructure; la seconde est le manque de solutions pour sa surveillance et sa sécurité.

Dans ce chapitre, nous avons comparé les stratégies de mise en cache les plus répandues (LRU, FIFO) avec notre propre solution. Nous avons présenté un scénario commun pour évaluer différentes stratégies dans le même environnement de simulation. Les résultats obtenus montrent que notre solution de cache offre de meilleures performances comparées à celles des autres stratégies. Nous avons implémenté notre système de cache flou avec le simulateur ICARUS qui a été conçu pour répondre à deux exigences principales non fonctionnelles: extensibilité et évolutivité. L'architecture de haut niveau du simulateur contient les phases suivantes :

- *Configuration et génération de scénarios :* Cette phase inclut toutes les étapes nécessaires à la configuration d'une topologie réseau entièrement configurée et d'un générateur d'événements aléatoires pour la simulation. La génération de scénarios est basée sur la chaîne d'outils Fast Network Simulation Setup (FNSS) .

- *Orchestration:* Dans cette phase, le simulateur utilise le fichier de configuration afin d'extraire la plage de paramètres (par exemple, la taille du cache, la stratégie utilisée par le cache, la distribution de popularité du contenu) et entame principalement des expériences avec tous les paramètres choisis.

- *Exécution :* Après des expériences, une exécution réelle de la simulation commence par une mesure d'instance des différentes mesures.

- *Collecte et analyse des résultats :* À la fin de l'expérience et de l'exécution, les résultats sont recueillis et agrégés avec la possibilité de calculer les intervalles de

confiance et les résultats de la courbe de confiance.

Des expériences ont été réalisées avec LRU, FIFO et notre système de mise en cache flou. Nous avons évalué les performances dans une gamme de tailles de catalogue de contenu de 10^3 à 10^7. Il convient de mentionner que la taille du cache est calculée comme le rapport entre la taille cumulée du cache et le catalogue de contenu. Par conséquent, un scénario avec un catalogue de contenu volumineux a aussi une grande taille de cache. Afin d'évaluer l'évolution du système de mise en cache flou et de valider sa capacité à fonctionner dans un environnement ICN à grande échelle, nous avons évalué ses performances en termes de taux de réussite et de latence du cache par des conditions variables. En particulier, notre analyse se concentre sur la mesure du taux de succès du cache et de la latence de: FCS, LRU et FIFO avec différentes tailles de diffusion et de cache. Le taux de succès du cache et la modification de la latence de la topologie GEANT par rapport à la taille du cache avec un paramètre Zipf et un ratio de population de contenu variable (taille totale du cache réseau en tant que fraction de la population de contenu). Nous avons remarqué que FCS fonctionne mieux que FIFO et LRU et que la latence diminue avec la croissance de la taille du cache. Nous devions tenir compte du fait que plus la taille du cache est grande, plus il est facile d'obtenir de bons résultats de mise en cache. Par conséquent, notre solution reste un bon candidat pour la mise en cache dans des réseaux de contenus. La thèse se termine par une conclusion qui rappelle le contexte et les principales contributions, avant de citer quelques pistes de recherche en termes de perspectives à ce travail. Dans cette thèse, nous avons proposé un nouveau système intelligent de mise en cache pour la mise en oeuvre de mobile edge computing utilisant la logique floue et un modèle d'apprentissage renforcé. Ce système de mise en cache a été construit en deux étapes : La première étape consiste à fournir des services de mise en cache efficaces dans un environnement mobile dynamique et fortement limité en ressources. Nous avons proposé une technique de mise en cache floue économe en énergie pour les périphériques de bord qui améliore le taux de succès du cache et qui consomme moins d'énergie. Dans la deuxième étape, nous avons étendu l'algorithme du système de mise en cache à l'aide d'un apprentissage par renforcement modifié afin d'avoir une décision optimale de mise en cache. En termes de perspectives, à court terme, nous souhaitons améliorer la phase d'apprentissage de la politique de mise en cache afin de

maximiser le taux de succès. Une autre perspective à ce travail serait d'étudier la flexibilité du système de mise en cache et son adaptation à différents environnements ICN. Il serait, en outre, intéressant d'intégrer notre solution au sein d'un serveur MEC réel.

Introduction

The present architecture of mobile cloud computing is the result of the tremendous demand from users to have different types of services on their mobile devices. By the end of the decade, global mobile data traffic will have reached 30.6 exabytes – approximately 30 quintillion bytes or 30 billion gigabytes – per month, up from 3.7 exabytes in 2015 [4]. In addition, over 75% of global mobile data traffic will be video content. Given the current trend, more than 100 videos will be added each minute in *YouTube*. Counting the trend to the volume of data representing the video that has been produced so far, this volume will increase to 62% instead of 57%. As result, mobile devices became a dominant platform in the worldwide. In time with this, clients enhanced their expectations facing substantial number of issues related to the performance, environment, security and quality of service. The preliminary mobile computing scheme is originally client-server model which adopted 2-level hierarchy. Later on, the terminology "cloud" was used to represent a collection of servers with computational and information resources, which leads to the research on mobile cloud computing [92]. Mobile cloud computing considers various mobile-related factors compared to the traditional computation offloading techniques, such as device energy, bandwidth utilization cost, network connectivity, mobility, context awareness and location awareness [45], [102].

Despite the merits of mobile cloud computing, it experiences unavoidable issues due to the distance between mobile devices and the cloud. Realizing the benefits of providing services and cloud resources more closely, Mobile edge computing (MEC)**, which deploys cloud servers in base stations, has become promising solution towards factors like: low latency, proximity, high bandwidth, real-time radio network information and location

[4] https://www.statista.com/statistics/271405/global-mobile-data-traffic-forecast/

awareness. MEC is recognized as one of the key technologies for the next generation 5G networks by the European 5G PPP (5G Infrastructure Public Private Partnership) [37].

The increasing demand for mobile services over network poses several challenges concerning storage capacity and bandwidth. In traditional centralized mobile network architecture, the requests of mobile users are served by content providers crossing the whole mobile network leading to service congestion. MEC is providing cloud computing capabilities to bring popular contents at the network edge (e.g., gateways, base stations and end-user devices), which helps the content to be closer to the end user. Applications and analytic at the MEC servers are not impacted by congestion and latency. In fact, by performing analytic or caching contents at the MEC servers, the volume of data transmitted to the core for processing is reduced, and the impact of the data traffic through the network is minimized. However, the caches originally designed to enable a smooth and fast communication as well as a content distribution between edge servers and end users, allow to spread only popular contents around the network without taking into consideration other factors such as user mobility.

The traditional Internet paradigm presents several limitations. Firstly, although the Internet provides multiple and different contents, these contents are not linked together. Hence, TCP/IP, CDN (Content Delivery Network), and P2P are used in order to achieve a fast and reliable content transfer. Secondly, the fact of the strong relation between retrieving content and location is an important factor that impacts the QoS. Each time the content is requested, a specific address linked to a particular location is needed, while the most important need is to get the content as fast as possible. Resolving this problem in an explicit way requires replacing where (location) by what (information). Based on this premise, ICN (Information Centric Network) is a shift from a host-centric pattern by focusing on what is being delivered and by introducing the named information as a focal point regardless to the location. Today, users consume web contents in a huge quantity especially videos. The statistics of *YouTube* viewers worldwide according to the forecast from 2016 to 2021 shows that the number of online video platform viewers will amount to 1.86 billion in 2021, up from 1.47 billion in 2017 [5]. On netflix, the projected

[5] https://www.statista.com/statistics/805656/number-youtube-viewers-worldwide/

number of subscription video on demand (SVoD) households worldwide would grow from 250 million in 2018 to 450 million in 2022 [6]. This opens up a serious issue where network congestion becomes out of control and new mechanisms are needed to fulfill network and user requirements. In this context, storing content temporary on several servers over the network has been used in order to supply user's requests. This temporary storage is running over a system called caching system. The concept of caching is not new in cellular networks. It has been used in web caching but also in ICN that aims to improve the scalability of the network, by caching contents in the proxy servers and/or intermediate nodes of the network. Deploying caches in a very large scale, increases the available storing space but in the same time it increases the complexity of management. Managing the caches requires strategies that decide where and when the content should be cached depending on the content popularity. However, the emerging growth of user demand and Internet paradigm requires multi awareness caching management posing the possibility of defining better strategies. In the era of data flood, storing popular contents at the edge is a promising technique to satisfy the user's demands while alleviating the overcrowding on the back-haul. To this purpose, mobile edge computing servers equipped with local caches must be intelligently used. In the other hand, recent technical advances and the development of MEC applications have led to unprecedented growth of data traffic as MEC services are relying on big data with different types and of significant amount. Specifically, mobile edge computing servers (MECs) are equipped with computation, analytic and storage capability, which deal with the significant amount of content requests leading to improve the quality-of-service (QoS) of the network and the quality-of-experience (QoE) to the end user. Local caches are one of the MEC equipments that helps to reduce the request traffic and improves the response time. In particular, edge caching can speed up the data retrieval and reduce the traffic in the network, since data requests could be satisfied by edge nodes closer to the end user. This also implies that edge caching may relax the need of continuous connectivity by decoupling the producer and the receiver. However, edge caching has been widely investigated due to the requirements in term of information freshness. Caching is considered as a critical bottleneck for the development of MEC systems, as edge caches

[6] https://www.statista.com/statistics/322737/svod-households-worldwide/

are located at the edge of the network and physically closer to the end user. With limited storage, edge caches use different strategies of placement and replacement, which are in general designed to allow spreading only popular contents around the network. In addition, MEC servers present an exclusive opportunity to implement edge caching and to perform caching placement and replacement strategy design. Therefore, considering the benefits of avoiding potential network congestion and alleviating the backhaul links burden, caching popular content at MEC servers for backhaul capacity-limited mobile network has emerged as a cost effective solution [55]. Fulfilling the requirement of both network and the end user is a key point of MEC that consists of a dynamic computational offloading that might cause congestion in the network hence decreasing QoS and raising the decision making and the optimization problem on the whole system. That is, how to jointly encompass the problem of stochastic dynamics of the network and to enhance the decision of caching? Several pioneer works about caching in mobile edge computing have been proposed and realize quite good results but these works suffer from the following issues:

- Not enough inputs: They consider only popularity as a key information factor while actually there are several factors that effect on the caching decision.
- Strategy combination: They consider either placement or replacement strategies and not the whole caching system.
- Dynamic conditions: the dynamic of the environment and the caching system are not well addressed.
- Tentative isolation: Most of caching strategies do not consider the long term effect of the current caching decision.

The other proposed solutions are optimal but they suffer from a lack of intelligence that can serve to adapt the decision depending on the evolution of the system. In order to fill this gap, in this thesis, we focus on the design of novel intelligent caching strategies that tailor Mobile edge computing servers among learning methods. As one of the learning methods, reinforcement learning (RL) is a method that enables agents to deal with decision making problems by learning through interactions with the environment [104].

The main contribution of this thesis is to formulate an intelligent caching decision problem in a scenario where stochastically distributed MECs are equipped with caching units characterized by a limited backhaul and a storage capacity. Our work differs from the previous works in terms of studying a new system model with both placement and replacement aspects of edge caching using a modified reinforcement learning model.

This thesis contains three main parts:

- In Part I, we focus on the context and the state of the art. In particular:
  - *Chapter 1* recalls our context by presenting ICN environment and MEC characteristics.
  - *Chapter 2* is the background. It is related to two intelligent mechanisms that have been used in the contributions : fuzzy logic and reinforcement learning.
  - *Chapter 3* is the problem statement. It takes a close view on caching mechanisms in mobile edge computing using the intelligent mechanisms.
- In the second part of the thesis, we take a more practical approach to investigate the gains of caching throw our contributions. In particular:
  - *Chapter 4* contains the first contribution where we proposed a novel energy-aware caching strategy using fuzzy logic by relying on a hardware implementation using Field-Programmable Gate Array (FPGA) as an alternative computational architecture that cuts overall energy requirements.
  - *Chapter 5* contains the second contribution. We introduced a novel reinforcement learning-based caching system. This is done by modifying the traditional reinforcement learning algorithm and use the results extracted from fuzzy caching system interactions, referred to as source entries. This prior information with reinforcement learning techniques are incorporated where the goal is to optimally cache the contents.
  - *Chapter 6* that contains caching strategies simulation and clarifies the gain of our proposed strategy against the existing ones in ICN/MEC based environment.
- Finally, Part III includes our conclusions and future works for this thesis.

Part I

Context and State of the Art

Chapter 1

Context: Mobile Edge Computing and Information Centric Networking

1.1 Introduction

Mobile Edge Computing (MEC) is a new technology which is currently being standardized in an ETSI Industry Specification Group (ISG) [1] of the same name. Mobile Edge Computing provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers. The aim is to reduce latency, ensure highly efficient network operation and service delivery, and to offer an improved user experience [37]. MEC increases response time from the edge and allows contents, services and applications to be accelerated in order to enhance mobile user's experience over an efficient network and services. Unlike traditional centralized mobile networks, MEC is providing cloud-computing capabilities to bring popular contents at the network edge (e.g., gateways, base stations and end-user devices) so that the content is closer to the end user. Applications and analytics at the MEC servers are not impacted by congestion and latency [61]. In fact, by performing analytics or caching contents at the MEC servers, the volume of data transmitted to the core for processing is reduced, and the impact of the data traffic through the network is minimized.

[1] https://www.etsi.org/technologies/multi-access-edge-computing

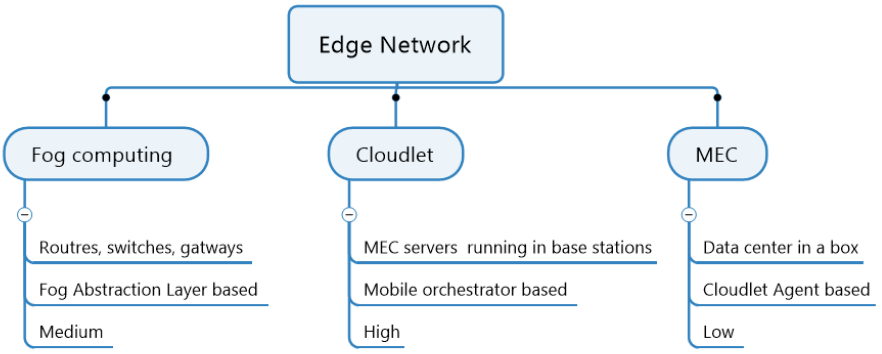

Figure 1.1: Edge network schemes

1.2 Mobile edge network

The main idea of mobile edge networks is to shift the network components closer to end users, using virtualization technologies like NVF and programmable networks like SDN. The network resources mainly include computing, storage or caching, and communication resources. While, in some literature, caching is included in computing resources [15], in this thesis we discuss the caching from the two aspects while the cache is a computing storage and also a computing resource. Three different schemes of edge computing have been proposed in the literature : mobile edge computing, fog computing and cloudlet. A demonstration of the three schemes regarding to the architecture, node device and context awareness is shown in figure 1.1

1.3 Mobile edge computing

The basic idea of mobile edge computing is to reduce latency by providing the ability of computing and storing in close proximity and allowing the services to be hosted on the edge. The prime objective of mobile edge computing, particularly, raises in indoor scenarios like improving quality of services and users' Quality of Experience (QoE), by providing customized services and reducing latency, beside optimizing network efficiency. It also provides services relevant to Machine-to-Machine scenarios, Big data analytics

and offloading. With the tight integration with environment, making easy to understand network and user characteristics. Figure **??** illustrates the outdoor scenarios of MEC including technology integration and the supported use cases that can be enabled by MEC such as e-Health, connected vehicles, augmented reality, gaming, IoT services, distributed contents and DNS caching. Figure **??** represents the MEC architecture where the basic elements are:

- Mobile devices, which are connected to the internet via a backhaul;
- MEC servers where there are two variations depending on whether the server is incorporated directly with the base station (BS) (interacting with BS in very sensitive way) or not incorporated directly with the BS (aggregation site);
- Edge cloud has the responsibility of network traffic controlling and hosting various mobile edge applications;
- Public cloud is the cloud infrastructure hosted in the Internet.

1.3.1 Mobile Edge Computing servers

The services that have provided by MEC servers are the following:

- Offloading: to increase the computation limit storage, bandwidth and device battery capacity, while reducing the energy consumption.
- Edge Content Delivery: MEC servers provide resources for deploying additional delivery content services to edge networks. MEC servers function as distribution nodes of local contents and use cached contents.
- Aggregation: Instead of routing all the data separately to the base routers, MEC servers are able to aggregate similar or related traffic and therefore to reduce network traffic.
- Local Connectivity: When the traffic is routed through MEC servers, the servers are able to separate the traffic flow and redirect it to other destinations.

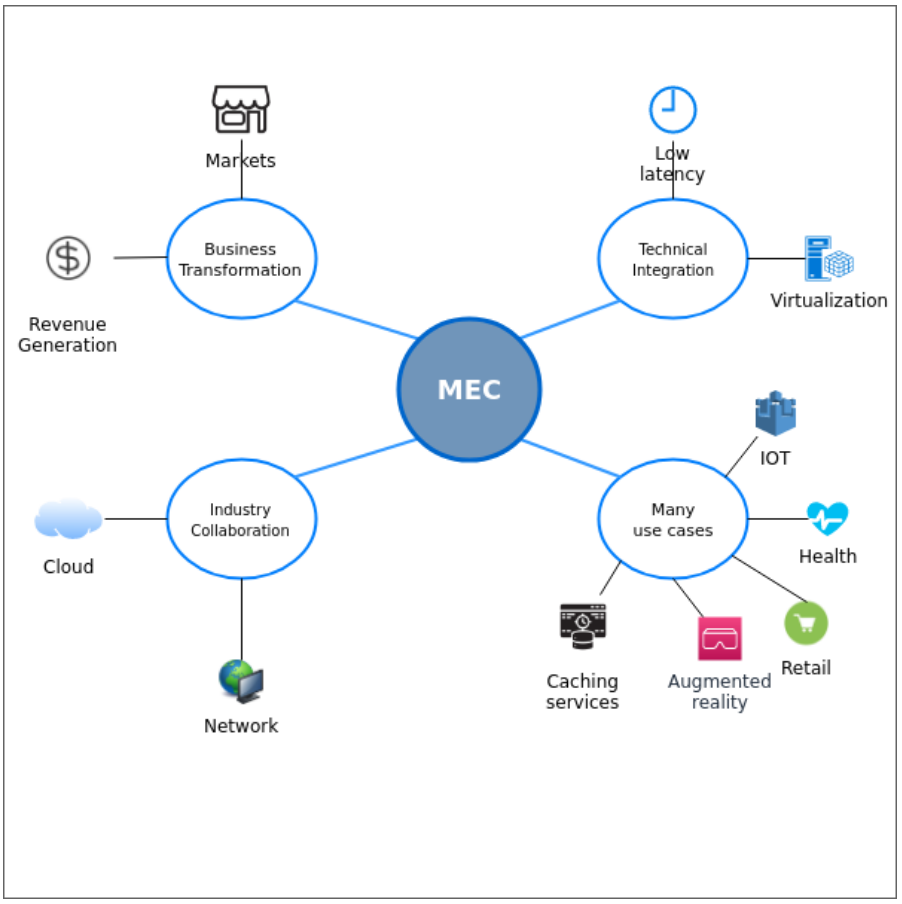

Figure 1.2: The outdoor scenarios of MEC

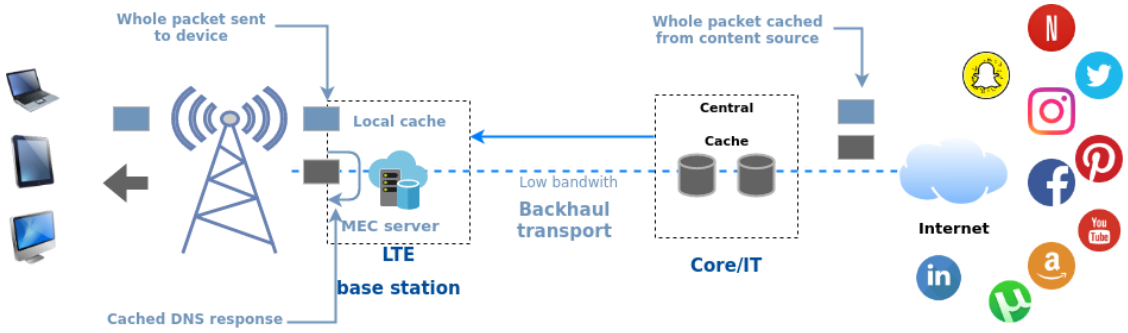

Figure 1.3: MEC architecture

- Augmentation: when additional data are available at the base station site, these data can be shared to improve the quality of the experience.

Typically, mobile edge computing is characterized by its locality and isolation, because it can have direct access to local resources that run isolated from the network. In addition, MEC nodes are so close from the end users and the source of information, which can easily extract information for analytics in order to analyze user's behaviors. In addition to providing context-aware services. As edge services run close to end devices, this reduces latency and increases the bandwidth, due to the close proximity of the edge and being a part from the wireless network. Mobile edge computing platform has characteristics which enable MEC servers to host and preform many kinds of applications which are not suitable to run in user's devices regarding to the different requirements, especially storage capacity. 2.1 presents these characteristics in presence and absence of MEC servers. The realization of MEC technology needs support of various key technologies like virtualization and servers including components and functional elements. Ejaz et Arif [6] presented a taxonomy of mobile edge computing. As in figure 1.4, the taxonomy shows that MEC is characterized by many properties and also, comprises many individuals and organizations representing different actors with different roles. Regarding to the services, MEC offers a huge range of applications in all domains to enhance quality of experience. Devices could access to mobile edge computing environment, by using any of the available access technologies. MEC is also characterized by its objectives. Its principal goal is to minimize latency and energy consumption. Finally, realizing a MEC environment necessitates the support of various key enablers represented by different technologies which contribute to provide better services to the mobile user.

1.4 Information centric networking

Resource sharing was a primary goal in the 1960s and the design of the Internet appears in order to reach this goal. High-speed devices and computers were hosted shared on sites. Currently, Internet Protocol (IP), which names the attachment point, was designed to support this sharing and to deliver data. Internet designers had not visualize the

	Without MEC servers	With MEC servers
Content Optimization	traditional content optimization is performed to satisfy user expectation. It uses user's web history stored in the database.	content optimizer can be hosted at MEC servers in order to enhance network performance, Quality of Experience and new services can be added.
Offloading and Aggregation	traditional high tensive computational like offloading and aggregation can not be performed in the device. In order to solve this issue, the applications are split into small tasks and perform at the core network	Offloading the tasks at the edge server without transferring the tasks to the core network will certainly reduce the latency.
Big Data Analytic	The process of data collection from the edge devices and transfer it to the core network takes high bandwidth and latency.	Mobile Edge Computing server can perform the big data analytic process and after the analysis, the results can be sent to the core network. Hence, reducing bandwidth consumption and improving the latency.

Table 1.1: The potential need of Mobile Edge Computing server

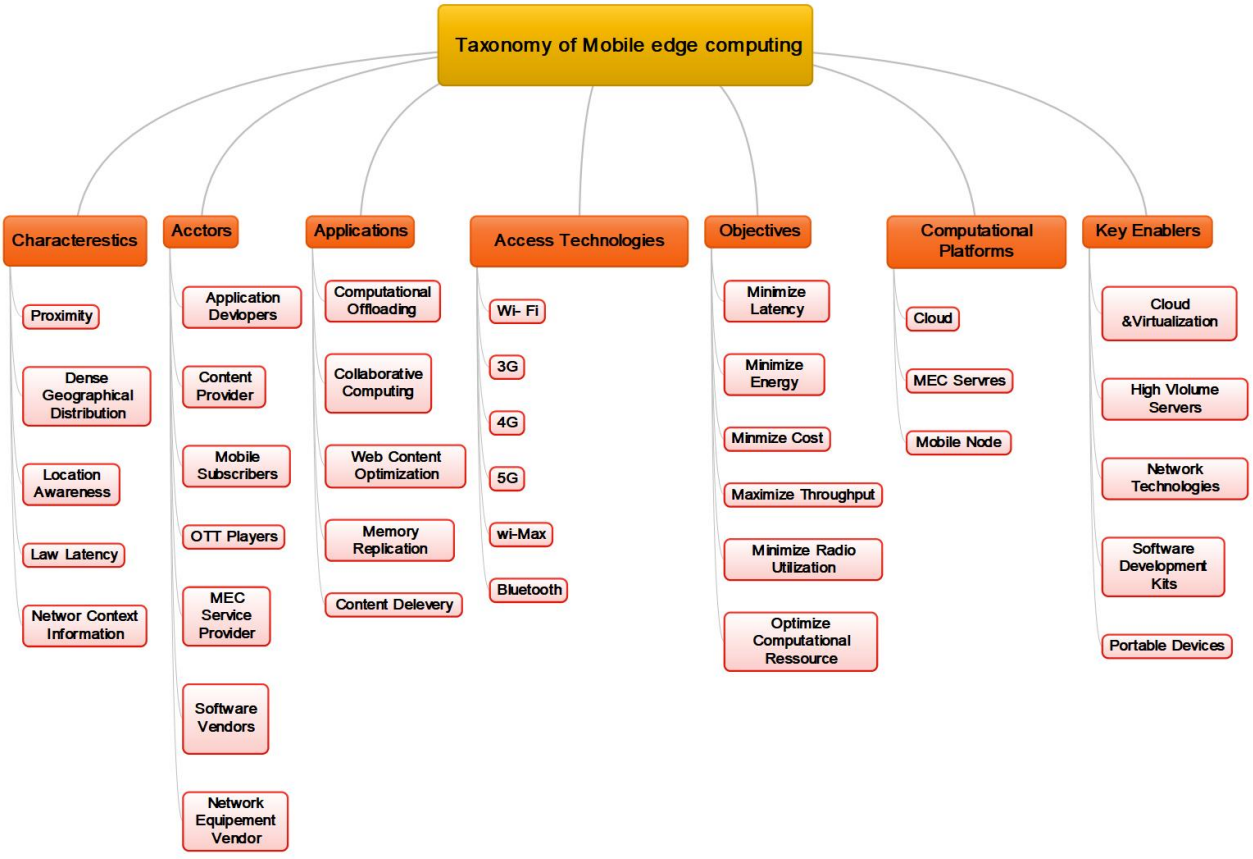

Figure 1.4: Taxonomy of the Mobile Edge Computing [6]

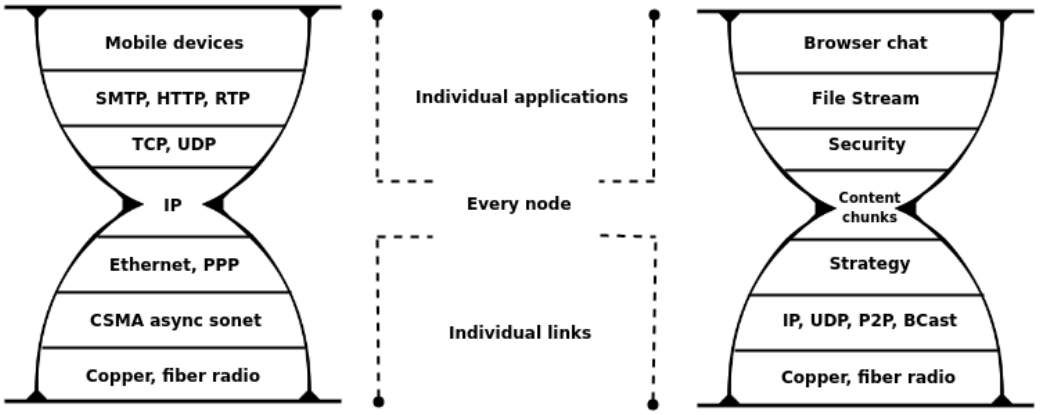

Figure 1.5: The hourglass approach of ICN (right) compared to the current Internet (left).

tremendous growth and the plenty ways to use the Internet, especially distributing data for the large scale using cloud and edge computing. On the other side, resources are distributed everywhere, while the user essentially cares about the data itself not the location. While IP, focusing on the location (where), is applied to deliver data (what), most of the recent applications such as gaming, big data, IoT, etc. do not give importance to the location, but to the data. A clean-slate method to address the above issues is to abandon the IP paradigm which was designed for resource sharing on the limited number of sites 50+ years ago, replacing with what to ship bits. Hence, Information-centric Networking (ICN) is proposed by naming the data directly, which breaks the limitation of point-to-point IP semantic [94].

The term Information-Centric Networking (ICN) is generally used to refer to the entire class of Internet architectures that focus on contents as the focal entity as opposed to a host centric networking architecture. ICN defined as a shift in networking paradigm that allows applications, services and networks to interact using network primitives centered on the information to be transferred. The key design of ICN is attributed to the hourglass approach, as shown in Figure 1.5, followed by the Internet's protocol architecture: the network layer forming the waist of the hourglass is transparent enough, so that almost any application can run on top of it and simple enough, so that it can run over almost any link-layer technology.

key concept	Description
Naming data	ICN primitives are based on naming the data. The names of each data must be unique and persistent. The name identity also must be independent from storage and the transmission path either in hierarchical or self-certifying scheme.
Retrieving data	Retrieving a requested content is divided into two tasks : The first is the step of discovering the content i.e forwarding requests towards data location. The second step is content delivery, which is the operation of transmitting contents to the requester. To satisfy future requests, caching data along the path is used.
Securing data	The security in ICN is decoupled from content containers and it is based on itself, allowing contents to be verified wherever it is stored and whoever retrieves it by including a signature directly.

Table 1.2: Key concepts and principles of ICN

1.4.1 Information centric networking vs current Internet

ICN has proposed solutions to several issues of the current Internet architectures, such as resource occupancy and utilization and security, as well as mobility and scalability. Although the ICN has reached much popularity, it has also many challenges such as caching, naming, routing and security [35].Among all these challenges, caching is considered the most crucial component since it needs a flexible strategy to decide how to cache via the network. The need of shifting towards a new network paradigm became a necessity in order to address the current host centric communication model issues and limitations. In ICN, the data becomes independent from location, application and storage enabling caching and replication. We present the expected benefits focusing on four main concepts and principles: information naming, information delivery, mobility and security. Table 1.2 summarizes the key concepts and principles of ICN, and shows how each one of them aims to treat some of the current Internet problems.

1.5 Conclusion

The design of the current Internet creates some issues that the user cannot support like denial of service attacks, spams, and the lack of information awareness inside the network. Besides, the different challenges like mobility, security, caching and QoS have given rise to new requirements. As well, the users start focusing on the desired information, which increases the need of refocus communication centering on content access rather than on host-to-host interaction. Researchers believe that Information Centric Networking (ICN) and Mobile Edge Computing (MEC) could play a key role towards universal Internet growth by treating the data content as the first class entity in network architecture and being able to store/cache server contents from various locations in close proximity to the end user. The next chapter introduces fuzzy logic and reinforcement learning, which are elementary concepts to build our caching system.

Chapter 2

Intelligent mechanisms: fuzzy logic and reinforcement learning

2.1 Introduction

Along the development of the network communication, mobile edge computing has significantly accelerated the content delivery and improved the quality of experience. Bringing more intelligence to the network on general and to the edge computing in particular is a necessity to provide effective services in the dynamically changing mobile computing environment. We present in this chapter two intelligent techniques that we have used in our contributions. In this section, we review some elementary concepts whose understanding is necessary fully benefit from this thesis.

2.2 Fuzzy logic

2.2.1 Preliminaries

Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth which denotes the extent to which a proposition is true. While classical logic holds that everything can be expressed in binary terms (0 or 1, black or white, yes or no), fuzzy logic replaces Boolean truth values with a degree of truth. The degree of truth is often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision [26].

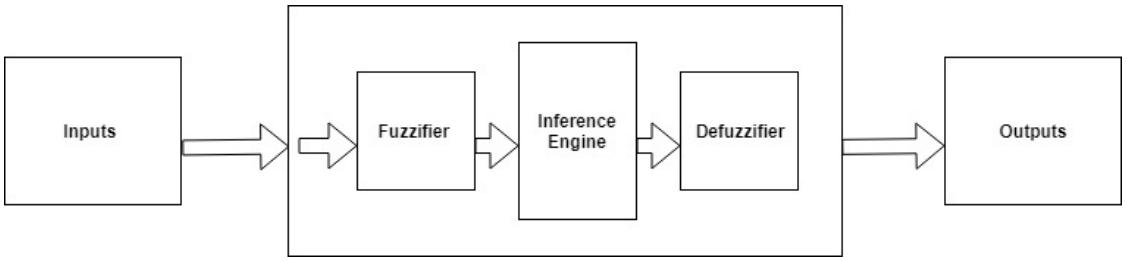

Figure 2.1: Block diagram of Intuitionistic fuzzy system

A Fuzzy Inference System (FIS) consists of an input phase, a processing phase and an output phase. In the input phase, the inputs are mapped to an appropriate membership function with specific values. The processing stage consists in performing each appropriate rule and in generating a corresponding result. It then combines the results. Finally, the output phase converts the combined result back into a specific output value. The membership function of a fuzzy set represented by divided ranges defines how each value in the input space is mapped to a membership degree. The inference system is based on a set of IF-THEN statements representing logic rules, where the IF part is called the "antecedent" and the THEN part is called the "consequent".

2.2.1.1 Definitions

Definition 1: An Intuitionistic Fuzzy Set A in X is defined as an object of the form A = $\{< \text{x}, \mu A(x), \vartheta A(x) > : \text{x} \in \text{X}\}$ where the functions $\mu A : \text{X} \rightarrow [0, 1]$ and $\vartheta A : \text{X} \rightarrow [0, 1]$ define the degree of membership and the degree of non-membership of the element $\text{x} \in \text{X}$, respectively, and for every $\text{x} \in \text{X}$ in A, $0 \leq \mu\text{A(x)} + \vartheta\text{A(x)} \leq 1$ holds.

Definition 2: For every common fuzzy subset A on X, intuitionistic fuzzy index of x in A is defined as $\pi A(x) = 1 - \mu A(x) - \vartheta A(x)$. It is also known as the degree of hesitancy or the degree of uncertainty of the element x in A. Obviously, for every $\text{x} \in \text{X}$, $0 \leq \pi A(x) \leq 1$.

Figure 2.1 shows a block diagram of an intuitionistic fuzzy decision maker.

2.2.1.2 Membership function

The membership functions are a very delicate point in the design of the fuzzy system. The main restriction that a membership function should satisfy is that its values must

Name	Description
Rectangular	$\mu = \left\{ \begin{array}{ll} 1, & left \leq x \leq right \\ 0, & Otherwise \end{array} \right\}$
Triangular	$\mu(x,a,b,c) = \left\{ \begin{array}{l} 1 - \frac{x-c}{b-c}, c < x < b \\ 1 - \frac{c-x}{b-c}, b < x < c \\ 1, c = x \\ 0, otherwise \end{array} \right\}$
Gaussian	$\mu(x,c) = \epsilon(\frac{x-c}{2\theta^2})$

Table 2.1: The membership function several shapes

be in the [0,1] range. Therefore, A fuzzy set can be represented by an infinite number of membership functions, unlike a crisp one. The membership function could be defined in several shapes: triangular, trapezoidal or Gaussian ones and so forth (see Table 2.1). The fact that a fuzzy set can be described by an infinite number of membership functions is advantageous in term of making the adjustment of a fuzzy model possible.

In general, we can sort out three methods to determine a membership function following three ways :

1. Automatic method : This method is used when no experts are available or in the case when there are so many data about the case study. Determine such a membership function consists of two main phases: the first one is creating a primary membership function which is aleatory and incorrectly adjusted. The second one is the adjustment and the optimization of such primary function.

2. Statistical method: In the statistical methods, data is represented in the form of frequency histograms or other probability curves. This data is used as a base to construct a membership function. There is a variety of possible conversion methods, each with its own mathematical and methodological representation.

3. Psychological method: This method is a natural extraction of membership functions as a general rule. The experts in the field specify the membership curve that is appropriate to the given problem, or choose a curve from an assigned set of possible membership functions. In this case, it will appear as an infinite number of possible membership functions. Hence, the choice is restricted to predefined membership functions.

We note that all these methods share the same idea of a fuzzy set made of two parts: an exact part and the gradual one. The exact part is defined as : $X_0 \cup X_1$, where X_0 = X - $\sup(A) and \mathrm{X}_1$ = Ker(A). While the gradual one is defined as : $X_f = \sup(A) - \ker(A)$.

2.2.1.3 Mamdani Fuzzy Control

An outer-loop for the fuzzy system, Mamdani's fuzzy inference method, is the most commonly fuzzy methodology. It was proposed in 1975 by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination. Mamdani-type inference expects the output membership functions to be fuzzy sets. After the fuzzification process, there is a fuzzy set for each output variable that needs defuzzification.

Mamdani method is widely accepted for capturing expert knowledge. It allows to describe the expertise in more intuitive, more human-like manner. Particularly for dynamic non linear systems, it can be used to customize the membership functions so that the fuzzy system behaves in better way. The most fundamental difference between Mamdani-type FIS and Sugeno-type FIS is the way the crisp output is generated from the fuzzy inputs. While Mamdani-type FIS uses the technique of defuzzification of a fuzzy output, Sugeno-type FIS uses weighted average to compute the crisp output. The expressive power and interpretability of Mamdani output is lost in the Sugeno FIS since the consequents of the rules are not fuzzy [34].

2.2.2 General Procedure for Intuitionistic Fuzzy decision system

Building a fuzzy decision system consists in selecting the inputs and the outputs of the system before dividing them into adequate categories named fuzzy set. This division is used to create a set of rules which determines the conduct of the defined fuzzy set. Each input

variable changes according to the corresponding membership function. Inputs, outputs, membership function and processing build together a Fuzzy Inference System (FIS) which consists of the following stages:

- Fuzzification : to establish the linguistic input variables.
- Inference: which is the processing stage that applies appropriate rules and generates the corresponding results.
- Defuzzification: is the process of producing a quantifiable result, given fuzzy sets and the corresponding membership degrees.

In the input phase, the inputs are mapped to an appropriate membership function with specific values. The processing stage consists in performing each appropriate rule and generating a corresponding result. It then combines the results. Finally, the output phase converts the combined result back into a specific output value. The membership function of a fuzzy set represented by divided ranges defines how each value in the input space is mapped to a membership degree.

2.2.3 Fuzzy logic : decision making

Decision-making is a logical human judgment process for identifying and choosing options or actions based on the values and preferences of the decision maker. Recently, decision-making has gained immense popularity in industries because of their global competitiveness. Therefore, decision-making plays a vital role especially in minimizing costs and time as well as improving the quality of service. Hence, network's problems in decision-making generally faces many confusions due to the uncertainty and subjectivity. To deal with this kind of vagueness, fuzzy logic has been widely used in the field of networks in order to make the best decision in the environment from a given information.

2.3 Reinforcement learning

Machine Learning is a form of artificial intelligence (AI) that allows computers to acquire the ability to improve their performance on a specific task. Reinforcement learning

(RL) appearance backs to the days of statistics in computer science and neuroscience. RL has attracted increasing interest in the filed of machine learning and artificial intelligence communities. In fact, it is possible to combine these different techniques. This is why it is difficult to clearly distinguish reinforcement learning from other machine learning methods. Reinforcement learning can be defined as a specialized application of Machine Learning and Deep Learning techniques, designed to solve problems in a specific way. Particularly, RL is based on a system of rewards and penalties to allow the computer to learn and solve a problem autonomously. The human programmer just changes the learning environment and makes changes to the rewards system. This method is particularly relevant when there is no single way to perform the requested task, but rules must be followed. What distinguishes reinforcement learning from other machine learning techniques is the way the AI agent is trained. Rather than inspecting the data provided, the model interacts with the environment and seeks solutions to maximize its rewards. Figure 5.6 summaries the explanation above.

2.3.1 Reinforcement Learning Model

The reinforcement learning enables agents to deal with decision making problems by learning through interactions with the environment [104] [80]. The basic components of reinforcement learning enrolment, as shown in figure 5.5, are:

- A reinforcement learning agent that learns through the interaction with environment over time. At each time step t, the agent observes a state S_t in a state space S about its environment, and chooses an action a_t from an action space A, following a behaviour policy $\pi = \mathrm{P}(a_t|\mathrm{s}_t)$ which is a mapping from state s_t to a probability of choosing action a_t.

- Then, the agent obtains a reward r_t and transitions to a new state S_{t+1}, according to the environment dynamics or model, for reward function $R(s, a)$ and a state transition probability $\mathrm{P}(S_{t+1} \mid S_t, a_t)$ respectively. The accumulated reward is defined as return $R_t = \infty\ k = 0\ \gamma^k\ r_{t+k}$ with a discount factor $\gamma \in (0, 1]$.

- The goal of the agent is to find an optimal policy, π^*, which achieves the maximum

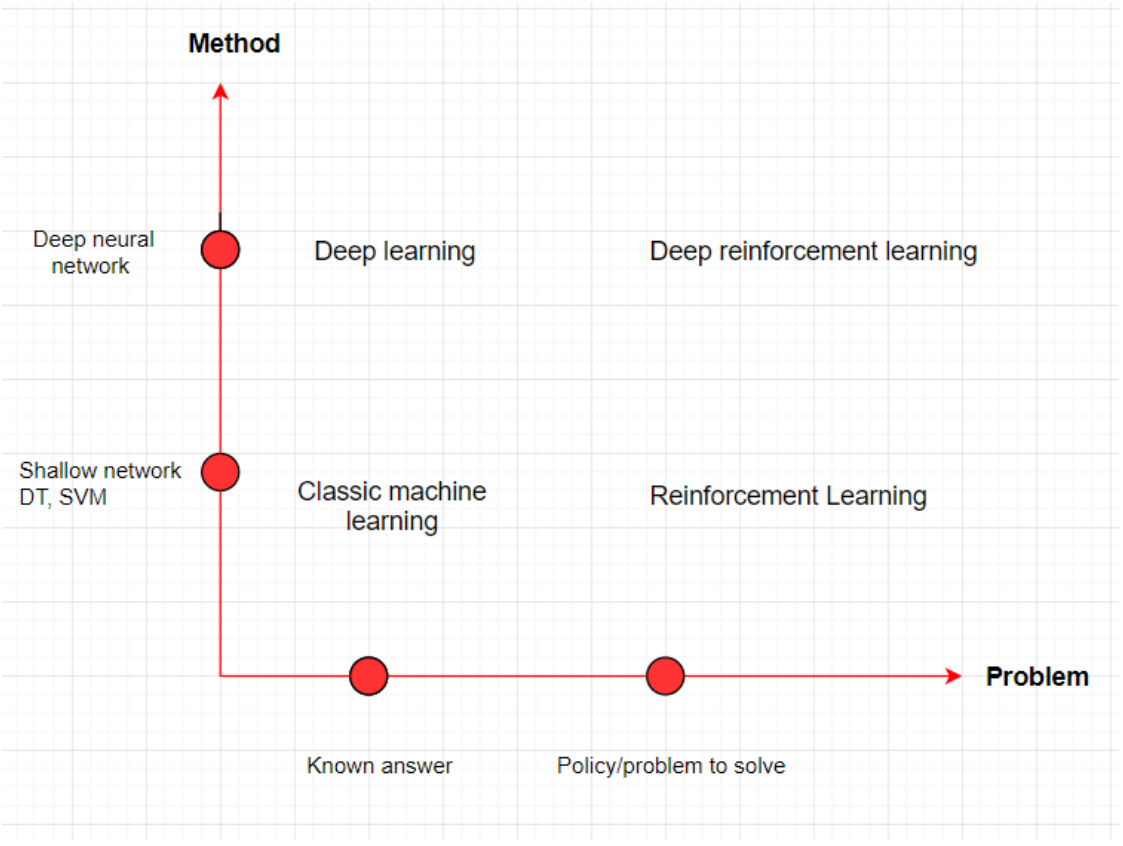

Figure 2.2: Reinforcement Learning, Machine Learning and Deep Learning relation

expected return from all states. The state-value function V π(s) = E[R_t | s_t = s] and the action-value function Q π (s, a) = E[R_t | s_t = s, a_t = a] can measure how good π is. V π (s) represents the expected return for the following policy π from state s, and Q π(s, a) represents the expected return for selecting initial action a in state s and then following π.

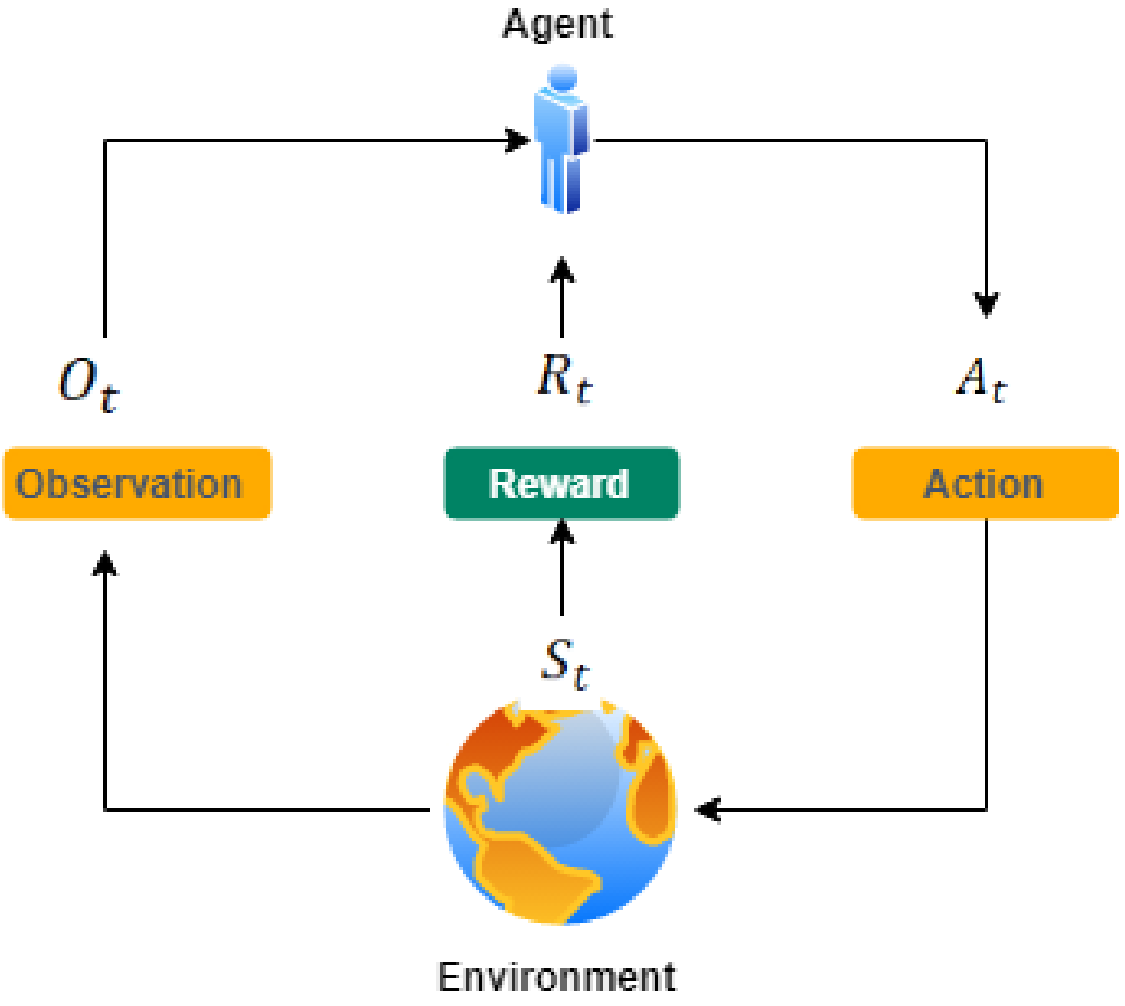

Figure 2.3: Reinforcement learning model

In all RL methods, it is critical for RL agents to trade off exploitation and exploration. Exploitation means taking the perceived best action to maximize rewards greedily. Exploration means taking perceived non-optimal actions with the aim of finding better actions when the policy is not optimal yet or the environment is not entirely explored. An example

of this is the e-greedy exploration policy, which selects a random action with probability $e \in [0, 1]$, and the optimal action otherwise.

2.4 Conclusion

Building a fuzzy decision system consists in selecting the inputs and the outputs of the system before dividing them into adequate categories named fuzzy set. Decision making using fuzzy logic has been used many objectives such as minimizing cost and maximizing quality of service. The basic idea of RL is to change the base of programming agents by reward and punishment for the aim of solving the problem that face the agents, where they must learn a specific behavior and actions through a dynamic and stochastic environment. The next chapter will introduce caching over these two intelligent mechanisms.

Chapter 3

Caching mechanisms in Mobile Edge Computing

3.1 Introduction

Edge caching is one of the most emerging technologies recognized as a content retrieval solution in the edge of the network. It has been also considered as enabling technology of mobile edge computing hat presents an interesting opportunity to perform caching services. Particularly, the MEC servers are implemented directly at the base stations (BSs) which enable edge caching and ensure deployment in close-proximity to the mobile users. This chapter presents a general overview on caching, illustrates the main caching strategies and their principles. Moreover, it contains a description of the performance measures as well as the basic criteria used commonly to compare between the existed caching mechanisms. Finally, we introduce how fuzzy logic and reinforcement learning have been used to enhance caching operation.

3.2 Caching: the Beginning

Caching is the process of storing data in a cache. A cache is a temporary storage area, from where we can get the files rather than the original server, hence saving the time retrieval and saving the network from additional traffic. Caching has been constantly growing with the load on the Internet and Web servers. Web caching has proven to be a valuable tool. Three features of Web caching make it attractive to all Web participants,

including users, network managers and content creators [24] namely:

- Caching reduces network bandwidth usage.
- Caching reduces user-perceived delays.
- Caching reduces loads on the origin server.

3.2.1 Caching Places

Generally, caching operation can be performed in three main locations: the core network, the radio access network (RAN) and the end user devices. For edge network, the caching places are the following:

- Macro base stations (MBSs) Caching: The macro BS manages the requests that can not be handled locally. It has more coverage areas, hence, it can serve more users. Beside, a better cache hit can be obtained.
- small-cell base stations (SBSs) caching : small-cell base stations can be equipped with cache memories to store popular contents. Multiple SBSs can serve a mobile user and create a flexibility association, which, in a broad sense, aim to maximize the number of served contents and reduce the load on the macro-cell base stations (MBSs).
- Device (D2D) caching: D2D communication is one of the key technologies in next generation 5G networks. It takes advantage of the storage capabilities of other user's devices in the system i.e a request can be served by another user through a device-to-device (D2D) communication, without requiring a connection to the small or macro base stations. It allows a fast serving due to the short distance communication.

3.2.2 Caching Strategies

A caching strategy is a management policy that handles temporary storage resources. A caching strategy decides what, where and when information is stored. In this section, we present the principles of caching strategies and then the caching strategies proposed in the literature.

3.2.2.1 Caching Strategies Principles

For better understanding of caching strategies, some concepts must be understood. Every edge node holds a cache that stores content temporarily. The cache is managed with placement and replacement policies. Cache placement policies face two challenging issues: which content should be stored in nodes (i.e. what to cache, since content is the pivotal factor in connections) and which cache nodes of the network should be selected as caching points (i.e. where to cache, since most nodes inherently hold caching capacity)[46] The placement policy defines the admission phase when the cache requires to store contents. The replacement policy defines the structure of the cache to evict content when space is required. Most of the caching systems have been studied only from the replacement aspect. The most common replacement policies are Least Recently Used (LRU), First-In First-Out (FIFO), Last Frequently Used (LFU) and Random that will be explained in the subsection below. Different caching strategies exhibit the same performance in the long term. However, caching strategies target the optimization of a metric. The metric could be any specific objective in order to enhance the network performance including traffic reduction, decreasing latency and achieving best QoE. Counting the number of hits and measuring the delay are one of the most common metrics used. In a caching operation, after requesting a content, a hit occurs when the content is found in the cache whereas a miss occurs when the content is not found. Cache hit-ratio metric is the number of cache hits divided by the sum of cache misses and hits (total number of requests) in a given time. A high hit ratio means that a cache performs well. A low hit ratio means that the data in cache should not be cached or that the cache size is too small to stand temporal locality of all the contents. The delay metric describes the time incurred to retrieve a piece of content. Caching can be evaluated also based on other metrics such as stretch and diversity which are very common metrics to evaluate caching strategies. Stretch is the ratio of the complete path from client to server. Diversity expresses the ratio of a unique content stored across all the caches.

3.2.2.2 Performance Measures

The performance measures for caching-mechanisms evaluation are the following :

- Cache Hit Ratio: A cache hit means that a request for a content can be satisfied from the cache directly. Otherwise, if a request cannot be satisfied by the cache, a cache miss occurs. Therefore, the cache hit ratio is defined as the ratio of the number of cache hits to the sum of the number of cache hits and cache misses. A high cache hit ratio can reduce the access latency and the server load.

- Cache Hit Distance: let $D(r)$ the distance from the requester to the MEC server containing the cache that will serve the request. The hit distance is then defined as the average of distance $D(r)$ over all requests. A shorter cache hit distance is preferred for a better user experience.

- Cost: the operational cost is the time incurred to retrieve a content. Therefore, it is desirable that the operational cost is as low as possible, so that a caching mechanism can be applied to large scale networks.

In general, edge caching means that popular contents can be cached in edge nodes such as macro base stations (MBSs), small base stations (SBSs) or even user equipment (UE) as shown in figure 3.1. Caching content when close to the end users can effectively reduce the redundant data traffic and greatly improve the QoE of users[12].

Concerning what to cache, the range of contents is widely increasing including videos, audio files and Internet-of-Things data, what makes the number of the reachable contents over the Internet extremely huge. Not all of the available contents should be cached, regarding to the fact of limited storage space of edge nodes. In the literature, to decide what to cache, content's popularity is one of the prime concern. It represents the probability of requesting contents by users or the frequency of demanding the contents over specific time period. This property should be taken into consideration as a main factor. Most of current works on mobile edge caching assume that content popularity follows a static Zipf distribution [52].

However, since the user groups associated with individual edge nodes are different and user preferences may change with time, considering only content popularity is not enough parameter to infer the caching decision. Edge caches select appropriate contents to be cached using caching policies. The caching policies decide to obtain different objectives

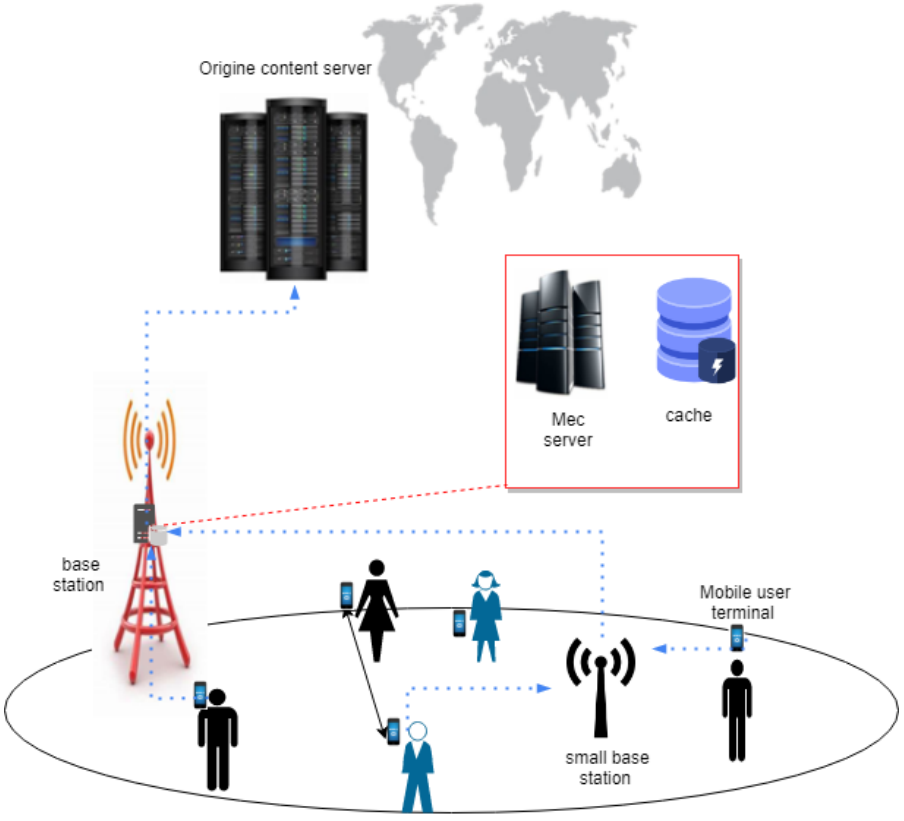

Figure 3.1: Edge caching architecture

such as traffic offloading, quality of experience, energy consumption and so on, in order to maximize the cache hit ratio. Caching policies can be divided into categories depending on their characteristics:

- Depending on cache coordination:

Coordinated: caches are required to exchange information in order to have a good estimation of where to cache the content and avoid storing too many contents of the same content. Coordinating cache is represented by a distributed session feature that allows multiple instances of a session to broadcast content changes among each other so that each cache is updated.

Uncoordinated: in uncoordinated scheme, each cache is working in an individual way.

- Depending on cache size:

Homogeneous: all caches in BS have the same size.
Heterogeneous: each cache has a different size.

- Depending on the cooperation between caches

Cooperative: caches cooperate with each other by establishing a cache state that allows other caches to know the different states like in [88].

Non-cooperative: caches make caching decisions independently and do not advertise the information of cache state.

- Depending on where the content is cached:

On path: caching only the contents caught a long the downloading path.
Off path: caching the content caught outside the downloading path.
In general, conventional caching policies are divided into two main phases: placement phase

and replacement phase. The placement phase which is the process that decides whether we should cache and how and when we can cache the contents. The replacement phase which is the process that decides which data to drop if there is no free storage space.

3.2.2.3 State of the Art on Caching Strategies

To make best usage of edge caching, researchers have prosperously used classical web caching algorithms either for placement or replacement phases that rely on popularity as a main factor. In the literature, we identify distinct placement strategies such as: *leave copy everywhere* (LCE) which is the default caching mechanism in most cache designs. It aims to minimize the upstream bandwidth demand and downstream latency [100]. In LCE, the popular contents are cached at every cache along the path and it is considered as homogeneous-non cooperative caching mechanism. *leave copy down* (LCD) and *move copy down* (MCD) are methods to realize the heterogeneous caching in a cooperative way in order to reduce redundancy [100] [26]. Probabilistic mechanisms have been widely used, like the policy which is referred to *Prob(p)*. It was used as a benchmark scheme in the literature [69][72][22]. This mechanism sets to every content a probability p in advance and does not cache the content for a probability equal to *(1-p)*. This is a heterogeneous and non-cooperative strategy. We also mention *Prob-cache* which is on path, heterogeneous and cooperative probabilistic caching mechanism. It adds additional fields to each request and content Time Since Inception (TSI) and Time Since Birth (TSB) [70] in order to allow the cached content to be visible to neighboring caches, i.e, enhance cooperation. Jason Min et al [90] proposed an explicit cache cooperation named *intra-domain cache cooperation*. Specifically it maintains two cache summary tables that record the information of content currently cached by the content router and exchange it with other caches. For edge caching, classical web caching was used widely. However, several works have proposed new placement strategies for the edge, such as [99]. They present edge buffer as a caching and a pre-fetching strategy, pointing out the insufficiency of strategies that rely on the past history of the device. Instead, they propose a prediction model based on the aggregated network-level statistics. Unlike blind popularity decisions, authors in [57] proposed a mobility-aware probabilistic (MAP) placing scheme, which caches content at edge servers

where the vehicles are connected considering the vehicular trajectory predictions and the time required to serve a content. S. Zhang et al. [101] explore a delay-optimal cooperative edge caching in large-scale, where the content placement and cluster size are optimized based on the stochastic information of network topology, traffic distribution, channel quality and file popularity. The CCPNC caching decision strategy considers the popularities of content objects and the distribution rules of routing nodes. It caches the popular contents to the core routing node with the highest node centrality and its next hop node in the content return path, and the non-popular contents are cached to the non-core routing nodes by the coefficient cache probability related to the request distance [63]

Table 4.6 summaries some of classical placement strategies.

The replacement phase is the phase that decides which content to evict from cache. It is categorized into: recency, frequency based such as the *least frequently used* (LFU) [4] and *least recently used* (LRU) [68], and semantic like *First In First Out* (FIFO) that always replaces the oldest contents in the cache [56]. Recently, an age-based caching replacement was developed in [52]. To minimize the cache replacement, a penalty caused by removing the old contents is proposed in *MAx-Gain In-Network Caching* (MAGIC) [71]. It allows also maximizing the local cache gain of the new cached content. An other strategy named *Cache Less For More* [19] aims to cache at the best location in the downloading path using the concept of "betweenness centrality" [11].

Some works obtain fundamental insights into caching operation like Harald Beck et al. [14]that provide an architecture for the dynamic selection of caching strategies. Their idea is to switch between the existing replacement strategies according to the changes in user's behavior using an intelligent agent (ICA). Matha Deghel et al. [25] investigated the benefits of edge caching in multiple-input multiple-output (MIMO) interference channel. Qinghua Ding et al. [27] have discussed a collaborative edge caching and designed a sequential auction mechanism (SAM) to allocate the cache spaces in a sequential way. For Named Data Networking caching, Noor Abani et al. [3] have considered in their research an NDN-based vehicular networks where they have presented a proactive caching based on mobility prediction using Markov model eliminating the cache redundancy introduced by edge caching. The work in [23] concentrates on comparing the performance of caching at

the edge versus the standard on-path caching on a random geometric topology and chooses then the most adequate strategy. Some works gathered processing and caching capabilities of the cloud such as [82], [84], [85] [83]. In [82], they proposed a new framework to provide a cloud caching/processing of data using Information-theoretic model for F-RAN (Fog Radio Access Network (F-RAN) which is an emerging wireless network architecture that leverages caching capabilities at the wireless edge nodes. The authors in [84], [85] and [83] have proposed a novel cooperative hierarchical caching framework in a Cloud Radio Access Network (C-RAN), in which a new cloud-cache at Cloud Processing Unit (CPU) is envisioned to bridge the storage-capacity/delay-performance gap between the traditional edge-based and core-based caching paradigms. The previous caching mechanisms have been successfully adopted in classical web caching, but they could not have been satisfied performance in the mobile edge caching due the ignorance of other properties and the use only of popularity, thus being unable to take advantage of specific characteristics of both the environments and the end user like mobile network topology uncertainty, user mobility, limited storage and cost.

****

****

3.2.3 Comparison of Caching Strategies

Four parameters have been extracted to point out and compare between caching strategies in diverse simulation environments of the most known caching strategies : topology, popularity model, catalog and cache size.

- Topology : there are several topologies used to evaluate caching strategies. These typologies vary from k-ary trees to complex ISP (Internet Service Provider) level topologies or a combination of the all.

- Popularity model : The popularity indicates the demands of users and the priority. The priority changes as time passes by[50]. The content popularity model is a function that establishes the popularity of every content, i.e., how often the content is going to be requested. The content popularity is usually modeled with a probability

distribution function such as a Zipf or MZipf [5] [72]. Other works have resorted to real traces.

- Catalog : The catalog represents the entire collection of contents in the network. The number of requests for a certain content depends therefore directly on the popularity model. A wide range of values for the catalog size have been used, most commonly, *Youtube* used as a catalog of 10^4 to 10^8 contents.

- Cache size : The cache size determines the space available in every MEC server to store temporally web contents. This size is usually expressed with an absolute value or a ratio with regards to the catalog size. the work in [33] have studied the impact of different object sizes on the performance and the overhead of web caching and show how the awareness of the size and could be crucial for the cache efficiency.

3.2.4 Differentiated caching service

The significance of providing differentiated caching services is emphasized in the study by Lu [54], which can be summarized as follows:

- From the user side: cache is important and useful for end users from the aspect of the fast access and enhance QoE. If the cache is located in the core of the network, then it can reduce the transmission delay. Otherwise, if the cache is in the edge network then the transmission delay will be greatly reduced and the cache utility will be enhanced. Hence, users will be served with faster access delay.

- From the content side: caching the most frequent contents more often improves client grasp performance. Since different web contents contribute differently to users' perception, the caching systems should treat the content as a focal point and requires a distinguish between different contents to improve the quality of experience of users.

3.3 Caching over fuzzy logic

The last few years have witnessed an emerging growth in the number of novel caching strategies. Fuzzy logic has been widely used in many applications because of its rapid

preliminary study on different inputs, its easy adaptation and interpretation of the rules and its ability of using heterogeneous inputs which facilitates the combination between several factors in the most desirable way without using mathematical relations. The work in [44] has attempted to combine recency and frequency. They described the use of fuzzy logic to improve cache replacement decisions by using fuzzy rules that can combine these parameters. Fuzzy logic has been used in [78] to combine removal policies to perform the cleanup task in an efficient way and to optimize the performance of the proxy cache.

An other neuro-fuzzy caching strategy has been introduced in [49]. The authors proposed an adaptive neuro-fuzzy inference system-based caching (ANFIS-BC) scheme to improve the cache performance. The proposed ANFIS-BC scheme used the feature of centrality-measures for selection of a router for caching in a network. Their results demonstrated that the ANFIS-BC scheme consistently achieves better caching gain across the multiple network topologies.

In the field of caching prediction, the authors present a model to predict the data in cache operation on the mobile side by applying fuzzy logic method in order to prevent the transaction failures during execution in the mobile system that communicate across the wireless networks. Despite the success of using fuzzy logic in caching systems, it is a little bit power and time consuming because of the nature of fuzzy inference process. In this thesis, we take into consideration this consumption and we will explain it in the next part.

3.4 Caching over Reinforcement learning

As one of learning methods, reinforcement learning (RL) that enables agents to deal with decision making problems by learning through interactions with the environment have been used in caching systems and addressed to solve several issues such as caching in Next-G cellular networks [75], in which local and global content popularity profiles exhibit in a dynamic environment. Hence, a proactive caching is accommodated by casting the problem in a reinforcement learning framework. A caching control unit is used and it continuously learns, tracks, and possibly adapts to the underlying dynamics of user demands. The work in [95] explores cache design problem in small cell networks (SCNs) with multiple

small base stations (SBSs) when user preferences are unknown. This multi-agent decision making problem is formulated as one of reinforcement learning algorithms. In the same context, authors in [41] proposed a multi-agent reinforcement learning (MARL)-based cooperative content caching policy for MEC architecture where only the historical content demands can be observed. They propose a MARL-based algorithm to solve the caching problem formulated as a multi-agent multi-armed bandit problem. Continuously, authors in [20] proposed a dynamic edge caching policy for heterogeneous vehicular network via Reinforcement Learning on adaptive traffic intensity and hot content popularity. The aim of this work is to enhance the download rate of vehicles by caching the popular contents on heterogeneous network edge stations. For taking the optimal caching decision, the work in [42] has proposed a novel caching strategy using multi-agent reinforcement learning. Specifically, they modeled the D2D (Device to Device) caching problem as a multi-agent multi-armed bandit problem and used Q-learning to learn how to coordinate the caching decisions. Any reinforcement learning based strategy interacts in an unknown environment and tries always to maximize its cumulative reward. It is important for the agent to explore optimal or even near to optimal actions as well as to choose the actions with highest known rewards. Yet, in real-time domains, collecting data and exploring it is not always a possible option, but it is still important to find a suitable policy with a certain performance guaranty.

3.5 Conclusion

In this chapter, we have presented the state of the art of caching mechanisms presenting the main principles. We first discussed the caching from a general aspect and the most known strategies in particular placement and replacement schemes, before ending with the description of caching over fuzzy logic and reinforcement learning, thereby finalizing this pre-apprehension part. In the following part, we will present our contributions that contain the caching system model.

Table 3.1: Summary of some existing classical placement mechanisms

Placement strategy	Type	Description
Leave Copy Everywhere (LCE) [28; 56]	Homogeneous, non-cooperative, on-path	caching the content at each node that it traverses along the downloading path what causes caching redundancy.
Leave Copy Down (LCD) [51; 74]	Heterogeneous, cooperative on-path	caching the content one hop closer to the user or to the next hop down, it caches only popular content and prevent the unpopular
Move Copy Down(MCD) [98]	Heterogeneous, cooperative, on-path	caching all popular contents, the policy moves the copy of the requested content with a hit to its underlying cache in the path and deletes it, it may suffer from caching pollution on the local replica in case of multiple requests
Probabilistic Cache (ProbCache) [21; 87]	Heterogeneous, cooperative, on-path	Contents should be cached closer to their destination with higher probability in order to leave caching space
Intra-AS Cooperative Caching [91; 103]	Homogeneous, cooperative, off-path	allows different caches to serve each other's request leading to solve the limited storage space problem and eliminate redundancy.
Least Unified Value (LUV) [2]	Homogeneous, cooperative, on-path,	Each cached content is assigned a Least Unified Value (LUV) with cache distance from the content source and caches the content later according to this value
WAVE [22]	Homogeneous, non cooperative, on-path	The contents are cached based on their popularity and counts the access frequency of requests.
Opportunistic Caching (OC) [13; 23]	Homogeneous, non cooperative, off-path	Probability based caching policy that uses the distance factor beside the popularity factors to cache a content.
Congestion-Aware Caching (CAC) [9; 29]	Homogeneous, non cooperative, on-path	it aims to decrease the download time by exploiting the available cache capacity, it is based on two factors: the download time and the content popularity
Progressive caching policy (PCP) [64; 89; 96]	Heterogeneous, cooperative, on-path,	it is a combination of some existing caching policies that are based on the position of a cache in the network : intermediate or edge , PCP avoids caching unpopular contents and shares the characteristics of both LCD and probabilistic caching

Table 3.2: Summary of some existing classical replacement mechanisms

Placement strategy	Description
Modified-LRU [48]	LRU modification is made to combine frequency and recently files in the decision stage to replace files so that the Modified-LRU can improve performance more optimally
SF-LRU [7]	Second chance-frequency - least recently used that combines the LRU (least recently used) and the LFU (least frequently used) using the second chance concept. the idea is that the replacement algorithm provides a second chance to the block that may be deleted according to LRU. Thiswas done by comparing the frequency of the block with the block next to it in the set.
NC-SDN [43]	Named Data Networking Cache replacement approach based on SDN, relies on data popularity calculation performed by the switches to define a cache replacement strategy.
Cache Replacement Strategy With Limited Service Capacity [40]	a replacement strategy for heterogeneous networks that consider the limitation of service capacity and user mobility. The authors used the user characteristics, such as user mobility and file popularity, to estimate the user demands, and then define the system cost. The cache strategy design was formulated as a mixed integer linear programming problem to minimize the system cost.
Cache Replacement Strategy Based on Hierarchical Popularity [53]	a cache replacement strategy based on hierarchical popularity in NDN according to the popularity of data packets to be replaced. The simulation results show that the cache replacement strategy based on hierarchical can improve cache hit ratio.
Multi-metric Cache Replacement [67]	considers the content popularity, relevance, freshness, and distance of a node to devise a set of algorithms for selection of the content to be replaced.

Part II

Contributions

Chapter 4

Energy-efficient caching decision using Fuzzy Logic

4.1 Introduction

As stated in the preceding chapter, edge caching is viewed as an enabling technology of mobile edge computing where MEC servers are implemented directly at the base stations to ensure deployment in close-proximity to the mobile users. However, the integration of servers in mobile edge computing environment complicates the energy saving issue because the power consumed by MEC servers is costly especially when the load changes dynamically over time. Furthermore, users with mobile devices arise their demands, introducing the challenge of handling such mobile content requests beside the limited caching size. Thus, it is necessary and crucial for caching mechanisms to consider the mentioned factors, meanwhile most existing studies focus on cache allocation, content popularity and cache design. In this chapter, we propose an energy-efficient fuzzy caching strategy for edge devices, called Fuzzy Caching system for Edge Computing (FCEC), that takes into consideration four influencing features of mobile environment, while introducing a hardware implementation using Field-Programmable Gate Array (FPGA) to cut the overall energy requirements.

4.2 The System model

In order to increase the cache hit ratio, the above studies are directed to be appropriate for different network topologies, but they consider neither the properties of the content

itself like size nor the influencing factors and the end user characteristics such as cost and mobility. They only accumulate contents with past high popularity or high frequency which may no longer be useful over time. To adjust adaptively the varied content properties and its influencing factors, we present a fuzzy control caching system for edge servers which can select the more priority content to be cached based on well chosen factors. To decide whether to cache or evict contents, we propose to base the decision making process on the following: mobility, frequency, cache occupancy and cost of retrieval.

Mobility: since mobility is a fundamental feature of wireless systems, several analytical models are available [10]. Mobility-based caching policies have been investigated, such as in [57] and [99] where they used the mobility of users to predict the next base station in addition to the next requested content to be cached on it. The mobility can be modeled by several ways and the most used in literature is Markov chain. When the user mobility is modelled by a Markov chain random walk, the optimal storage space is approximately solved using the same model in [3] and [97]. The use of mobility in caching leads to insure its influencing role. Authors in [10] and [39] developed a mathematical model to explore the impact of user mobility on the performance of caching and it has been proved that user mobility can be turned into an advantage to enhance caching decision and to minimize the incurred service cost.

Frequency: in the present context, the frequency is defined as how often the contents are requested or intended to be requested. It is extracted from a function that establishes the popularity of every content. Content popularity is commonly modeled with a probability distribution function such as a Zipf or MZipf [31]. Authors in [8; 62; 69] used a Zipf popularity model with α parameter $\in$ [0.6-2.5]. Other authors have extracted real traces from Content delivery network (CDN) [17]. Frequency based caching schemes store only popular contents and keep count of the frequency of contents to determine the ones that are expected to become popular. The above studies have demonstrated how the frequency parameter can be fine tuned to obtain the best caching performance.

Cache Occupancy: The cache occupancy determines the size available in the cache. This size is usually expressed with an absolute value or a ratio. We express it by the following: CO = 1 means that the content occupies 100% of cache size. There are many

related studies in the literature that propose and make use of the distribution of the content sizes [39]. In the literature, mostly, the size of cache is fixed and contents are used with homogeneous or heterogeneous sizes and gathered in a catalogue to represent the entire collection of contents in the network. Hence clarifying the strong correlation of size on cache performance can be determined.

Cost of Retrieval: In our work, we take into account the cost of content retrieval, i.e., the cost associated to the external bandwidth needed by a MEC to retrieve the requested content to the end users. Classical caching strategies aim to maximize cache efficiency and propose solutions with high overall cost. To solve this mismatch, authors in [8] formulate a polynomial-time greedy algorithm to solve the optimization models that either minimize the cost or maximize the hit-ratio. Results show that significant cost savings are attainable. It is confirmed that the consideration of cost can impact the performance of caching systems.

4.2.1 Fuzzy decision system

Building a fuzzy decision system consists in selecting the inputs and the outputs of the system before dividing them into adequate categories named fuzzy set. This division is used to create a set of rules which determines the conduct of the defined fuzzy set. Each input variable changes according to the corresponding membership function. Inputs, outputs, membership function and processing build together a Fuzzy Inference System (FIS) which consists of the following stages:

1. Fuzzification : to establish the linguistic input variables.

2. Inference: which is the processing stage that applies appropriate rules and generates the corresponding results.

3. Defuzzification: is the process of producing a quantifiable result, given fuzzy sets and the corresponding membership degrees.

Our FIS proposed model, illustrated in Figure 4.1, is made of the main FIS components. In our context, the proposed fuzzy algorithm integrates a fuzzification stage that consists

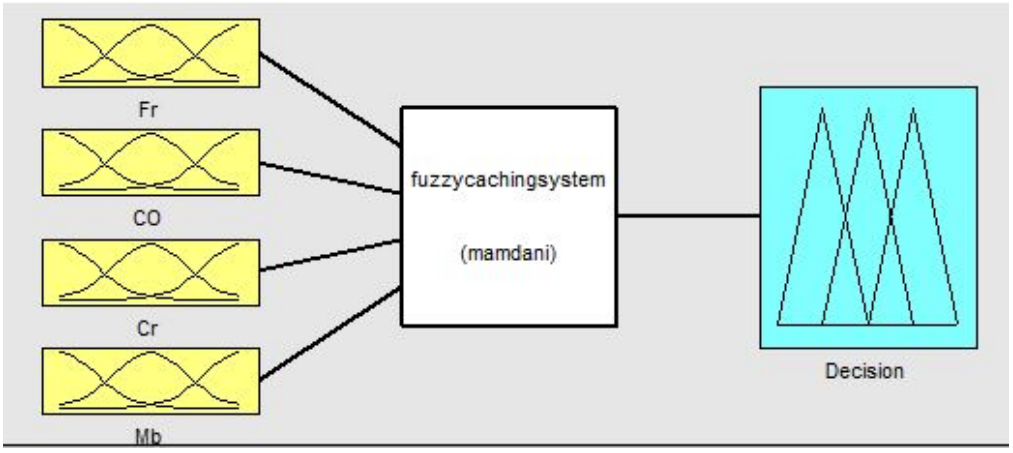

Figure 4.1: Fuzzy inference system of caching decision

of four linguistic variables as shown in table 4.1. The output variable is the decision of caching, as depicted in table 4.2.

Table 4.1: Input variable description

Input	Description
Fr	Frequency of demand of each content in time period
Cr	The time incurred to retrieve the content
CO	Occupancy of content in the cache
Mb	The users proximity from the base station

Table 4.2: Output variable description

OUTPUT	Description
DECISION	The decision taken to cache contents

The first input parameter in FIS is the frequency which is the value assigned to the data content after each request is issued by the user. The second input variable is the cache occupancy which is the free space in the cache. The third input parameter is the user mobility while the fourth one is the cost of retrieval which represents the cost incurred to retrieve the data content. The output is the caching decision. Concerning the processing stage, we use Mamdani's inference method proposed by Mamdani and Assilian [58]. Its principle is to synthesize a set of linguistic control rules obtained from experienced human

operators [38]. Fuzzy rules combine these four parameters, as they are connected, by experts reasoning in the real world.

4.2.1.1 Design of the fuzzy decision algorithm

Our proposed algorithm is designed based on different parts in respect to the fuzzy-logic system, as in the following.

- Input and output variables

It is important to well define the relevant input and output parameters. Frequency (Fr), mobility (Mb), cache occupancy (CO) and cost of retrieval (Cr) are the four factors used as the inputs of the fuzzy decision algorithm. Each factor has three categorical dimensions and must be partitioned into appropriate ranges. The fuzzification process is described in tables 4.3 and 4.4.

Table 4.3: Fuzzification of input variables

	Description	Ranges
Fr	Low, Medium, High	[0-0.2], [0.3-0.5], [0.6-1.0]
Cr	Short, Medium, Long	[0-0.3], [0.4-0.6], [0.7-1.0]
CO	Small, Medium, Large	[0-0.3], [0.4-0.6], [0.7-1.0]
Mb	So-close, Close, Far	[0-0.3], [0.4-0.7], [0.8-1.0]

Table 4.4: Fuzzification of output

	Description	Ranges
Caching decision	Low, Medium, High	[0-0.2], [0.3-0.5], [0.6-1.0]

- Membership functions (inference)

There are three membership functions for each of the input and the output fuzzy variables, all the inputs ranges of membership are [0-1] and each range is divided into three membership functions as shown in table 4.3.

The membership functions of the Frequency parameter (Fr) are {Low, Medium, High} and this is set according to the frequency of demand of each content in the time period. Different contents have each a distinct frequency value, and the most requested content has the highest value. The frequency feature is valuable in static environments where the popularity of objects does not change very much over a specific time period (day, week) [68].

The membership functions of the Cache Occupancy (CO) parameter are: {Small, Medium, Large}. Referring to this parameter, the larger is the cache size the bigger is the possibility of caching more contents. Using the size as input parameter is due either to the variation of the contents sizes or to its combination with other parameters to use its space in a wise way.

The membership functions of the Mobility parameter (Mb) are {So-close, Close, Far} in reference to the user's proximity from the base station where the cache is located. The network is divided into location areas where each area is covered by a base station that is equipped with a MEC server that hosts a cache. Each user moves from a location area to another from time to time. According to GPS traces, the user's location is set and its proximity to a base station is determined. We use this distance between the user and the base station as one of the input parameters to decide whether the content should be in cache or not because the closer is the cache, the less time is spent for retrieval.

Finally, the membership functions of the cost of retrieval parameter (Cr) are {Short, Medium, Long} as a reference to the time incurred to retrieve the content.

The mentioned membership functions are shown in Figure 4.2, where y-axis represents the degree of the membership of each input variable and the x-axis represents the quantized sensed values of the inputs.

The output variable is the decision of caching represented by flowing membership functions :{Low, Medium, High} as in Figure 4.3. Assuming we have N mobile users in the network, everyone is sending one request at the same time. The output values will be used to determine the priority for each request. Based on the first results that show the caching decision order, we observe that each request has a priority order. According to

this order, we put the contents in the cache in respect to the placement phase. In case of a full cache, we remove the one with the less priority in respect to the replacement phase as shown in Figure 4.4. We notice that each parameter has a value that influences on the decision strength and gives as finally a priority order for each content.

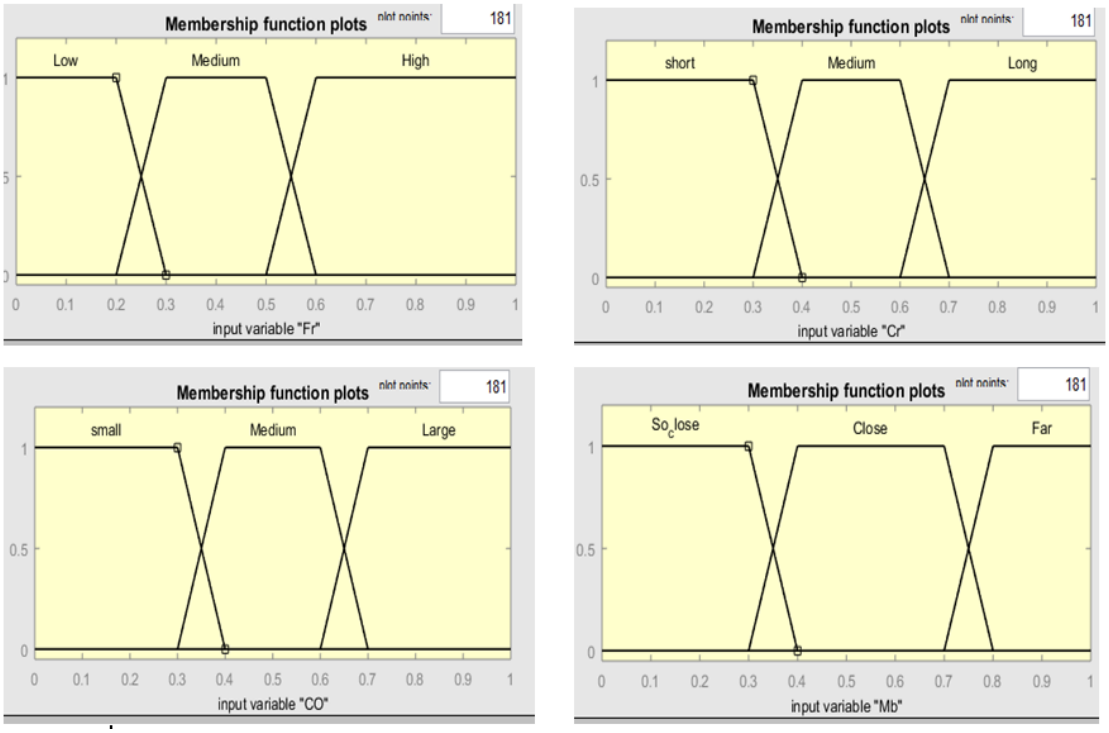

Figure 4.2: Membership function for the input variables

- Rule base

After defining the membership function, we build the rule base. These rules are represented by IF-Then clauses in which the antecedents are the conditions of the mobile user and the properties of the content while the consequence is the caching decision. There is no general procedure to decide the number of fuzzy rules and the role of each involved factor in the decision making. In our case, the set of rules are based on the understanding of cache behaviour under different scenarios. Examples of some fuzzy logic rules are the following: (see Appendix)

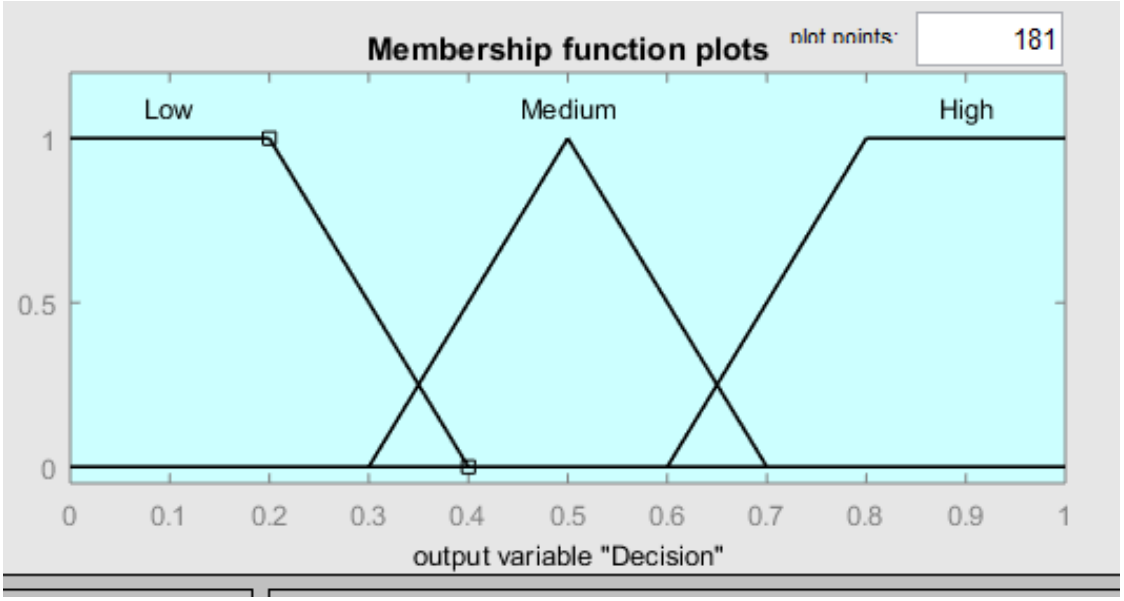

Figure 4.3: Membership function for the output

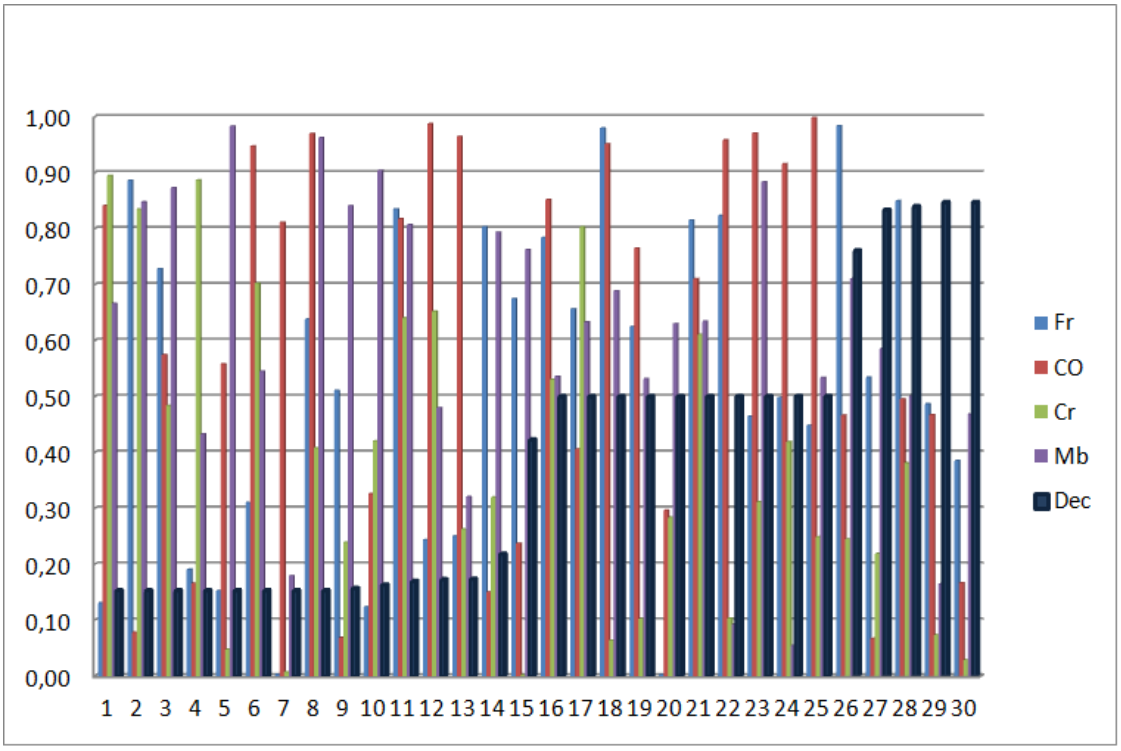

Figure 4.4: Cachig priority according to fuzzy system

IF Fr is LOW and CO is SMALL and Cr is SHORT and Mb is SO_ CLOSE THEN DECISION is MEDIUM
IF Fr is LOW and CO is SMALL and Cr is SHORT and Mb is CLOSE THEN DECISION is MEDIUM
IF Fr is LOW and CO is SMALL and Cr is SHORT and Mb is FAR THEN DECISION is LOW
IF Fr is LOW and CO is SMALL and Cr is MEDIUM and Mb is SO_CLOSE THEN DECISION is MEDIUM

4.3 Implementation and evaluation

This section presents the implementation and the evaluation of our proposed fuzzy caching system for edge computing (FCEC). We divide the evaluation into two parts: first, we compare the software implementation of FCEC with the Least Recently Used (LRU) [16] and First In First Out (FIFO) strategies. We used typescript [1] with RxJS [2] which is a reactive programming library for the software implementation. Then, we moved the algorithm into hardware using FPGA Xilinx virtex-5 LX50T-1156 board from DIGILENT coded with VHDL (VHSIC Hardware Description Language). The basic performance characteristic of a cache is a hit ratio. The hit ratio is computed according to the following equation:

$$cacheHit = \sum_{N}^{i=1} hit_i / (\sum_{N}^{i=1} hitsi + \sum_{N}^{i=1} miss_i) \tag{4.1}$$

4.3.1 Software solution

In our software solution, we designed a client-server model with a cache as a proxy between the client and the server. We generated a set of resources (contents with heterogeneous sizes) and a set of requests, then we investigated how the cache hit ratio changes when the average cache size varies. In the admission phase (placement), we note that the

[1] https://www.typescriptlang.org/
[2] https://rxjs.dev/

fuzzy caching system gives each content in the web a priority and places it in the cache according to this priority, unlike LRU and FIFO.

Table 4.5: Notation

Notation	Description
S	cache size
Id_i	Id or Index of content
N	Number of contents
R	Number of Requests
P_i	Priority of caching decision for each Id_i

Algorithm 1 Fuzzy Caching strategy

Initialization: Initialize N, R with Zipf distribution function
Client sends Request R
Base station receives R
Apply fuzzy rules to set P_i
while $S > 0$ **do**
  if *content in cache* **then**
    $hit \leftarrow hit + 1$
    return Id_i
  else
    $miss \leftarrow miss + 1$
    get $\mathrm{Id}_i fromserver$
    while $S = 0$ **do**
      check $\mathrm{P}(\mathrm{Id}_i)$
      Evict :Id with min{P(Id)}
    end
    Push in cache : Id_i
    return Id_i
  end
end

In Figure 4.5, we have a catalogue of 10^3 different contents with heterogeneous sizes. The requested contents were classified into Low, Medium and High priority in order to take the caching decision.

In the eviction phase (replacement), we used the content priority and the content size to choose the object that should be replaced. The efficiency of our proposed technique

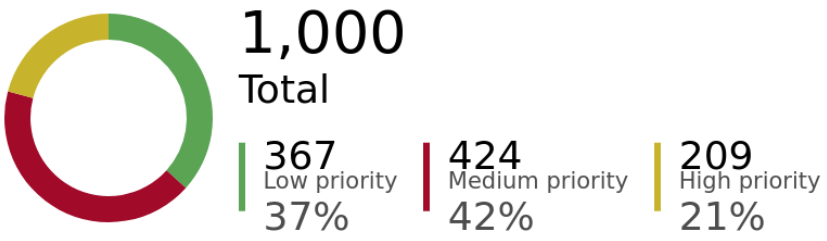

Figure 4.5: Distinction between web object by giving each a priority

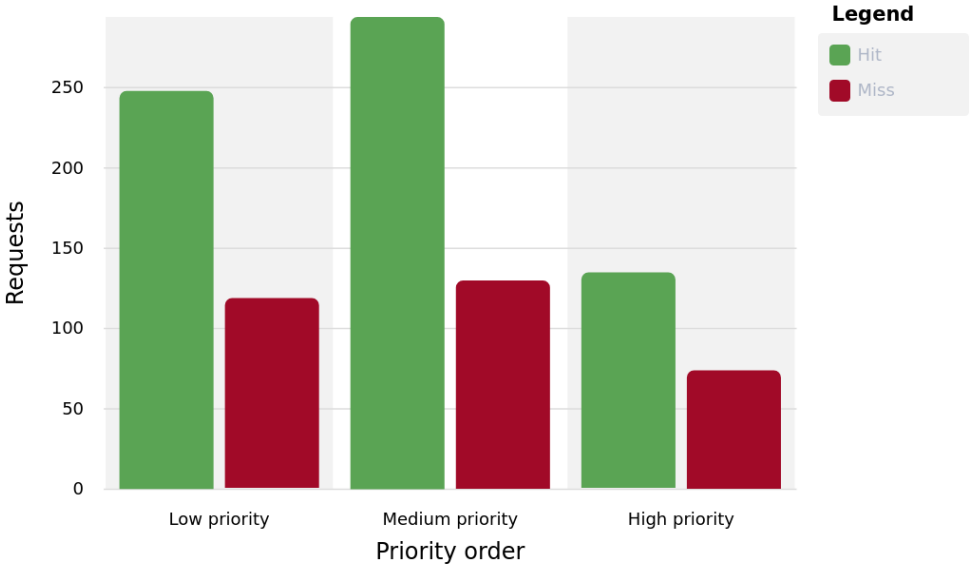

Figure 4.6: Cache hit ratio in high priority requests

appears when we launch random requests over time. We should mention that it became significant that high-priority contents in the web are most likely to be requested followed by Medium and Low ones. In Figure 4.6, a hit rate of 65% appears in high priority content.

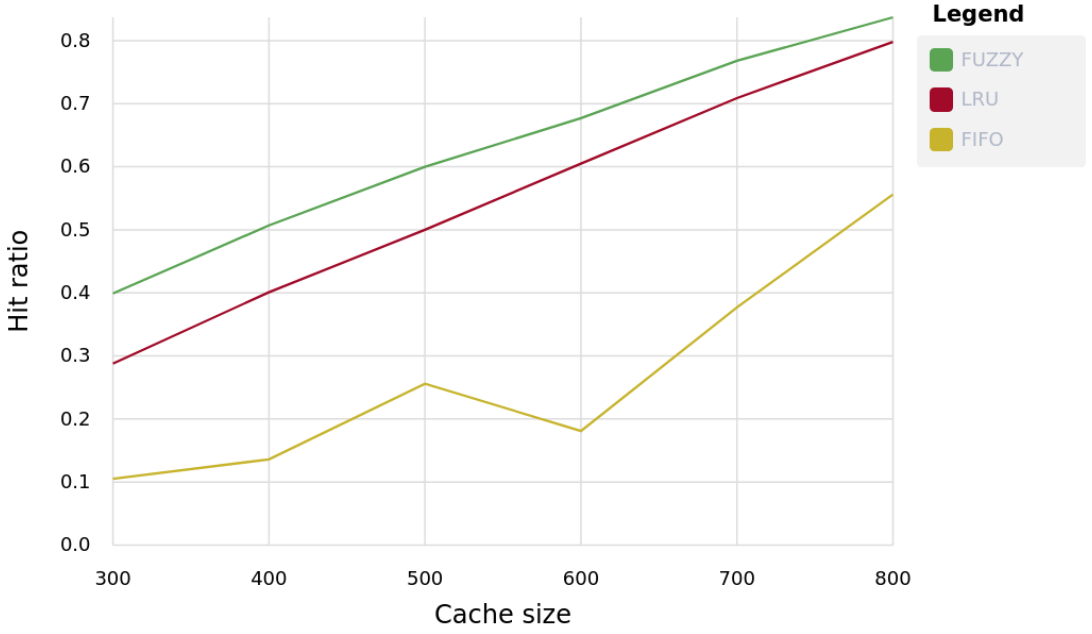

Figure 4.7: Cache hit ratio of fuzzy caching system, LRU and FIFO

4.3.1.1 Comparison and evaluation

To evaluate our proposal, we compared hit ratios with two caching strategies *Least Recently Used* (LRU) and *First In First Out* (FIFO). For the distribution of requests and contents, we used Zipf distribution function. Zipf distributions of finite support for a content catalogue of N contents with request probabilities $Z(r)$ have been determined for the objects popularity ranks $r \in \{1,2,3...,N\}$:

$$Z(r) = \alpha r^{\beta} with : r^{\beta} < 0; \beta = Z(1) = 1/\sum_{r=1}^{N} r^{\beta} > 0 \tag{4.2}$$

where β is an adaptive shape parameter and α a normalization constant. R represents the users requests with $R = 10^3$ that is generated according to the Zipf distribution function with α set as : $\alpha = 0.75$. A catalogue of $N = 10^3$ contents is then generated, where each

content is accompanied with its size, frequency, cost and mobility. These factors are first initialized randomly and then change over time according to α.

4.3.1.2 Long term performance

We first study the stability of cache hit ratio over time to characterize how the hit ratio changes with the changing of contents distribution in order to study the long term cache performance. Then, we focus on the short-term performance by studying the relationship between cache size and hit ratio.

Regarding to the long term study, we express it by the variation of α Zipf parameter to have different request distributions in order to evaluate the capability of our proposed system against the above strategies in term of maintaining good performance over time. We distribute a data set based on the random changing of α Zipf parameter value. From Figure 4.8, we can observe that the proposed fuzzy caching strategy and LRU are stable over time unlike FIFO. We also notice that our proposed algorithm shows more advantage with cache size = 600 and a hit ratio approximating 0.8 %.

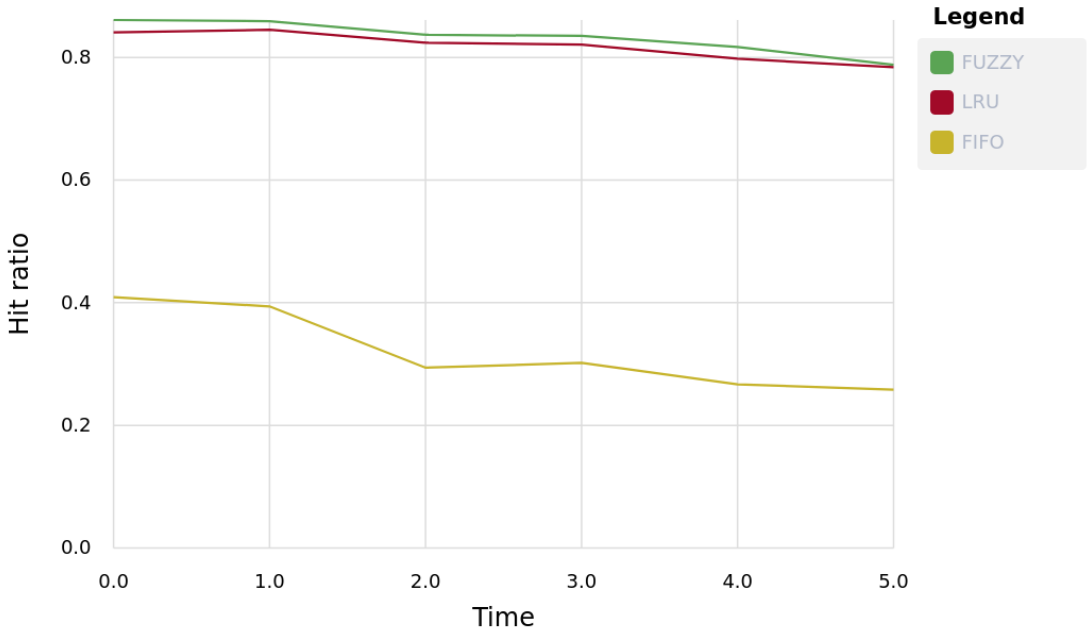

Figure 4.8: Cache hit ratio stability over time

4.3.1.3 Short term performance

For the short term performance, Figure 4.7 shows a hit ratio achieved by our proposed strategy and the other caching strategies introduced above. We can see how the hit ratio varies with the cache size measured with UNIT. We noticed also that each time the cache size increases, the hit ratio augments and the proposed FCEC system provides a higher cache hit rate for all cache capacity values. When the cache size is small, the performance of LRU is close to our proposed strategy. As the cache size increases, the gap between FCEC strategy and other two caching algorithms increases at first, and then gradually becomes almost similar with LRU and wobbling for FIFO. At cache capacity C = 700, the cache hit ratio of the three algorithms is increased at around 0.8%. At this point, the cache hit ratio achieved by the mentioned strategies tends to converge because the cache size is high enough to store all the contents.

4.3.2 Hardware solution

4.3.2.1 FPGA: Field programmable get array utilization

FPGAs are made up of an interconnection of logic blocks in the form of a bi-dimensional array. The logical blocks consist of look-up tables (LUTs) constructed over simple memories that store boolean functions. The FPGA architecture makes it possible to implement any combinational and sequential circuit, which can range from a simple logic function to a high-end soft-processor [47]. We propose an FPGA implementation for fuzzy-caching decision algorithm in the aim of reducing energy consumption since it has been proved in [30] that the energy consumption is lower with an FPGA-based solution than with a software implementation. We implemented our fuzzy system as presented in [66] on the FPGA Xilinx virtex-5 LX50T-1156 board from DIGILENT. The system was coded in VHDL and table 4.6 shows the resources used by our hardware design.

4.3.2.2 Results discussion

We migrated the main algorithm into hardware using FPGA. We noticed that the hardware consumes only 12 clock cycles, which can be considered as an advantage regarding

Table 4.6: the resources used by our hardware design

Logic Utilization	Used	Available	Utilization
Nbr of slices registers	286	28800	0%
Nbr of Slice LUTs	1403	28800	4%
Nbr of fully used LUT-FF pairs	148	1541	9%
Nbr of bonded IOBsMb	52	480	10%
Nbr of BUFG/BUFGCTRLs	1	32	3%
Nbr of DSP48Es	17	48	35%

to the critical role of the cache in the edge of the network. Figure 4.9 shows the measurement of energy consumption, the thermal properties, the voltage and the electric current during the run of the fuzzy caching decision system. We notice that a power of P =0.45 W is consumed when making the caching decision.

When comparing our FCEC strategy with LRU that consumes P =0.9 W [105], it is notable that FCEC consumes less power.

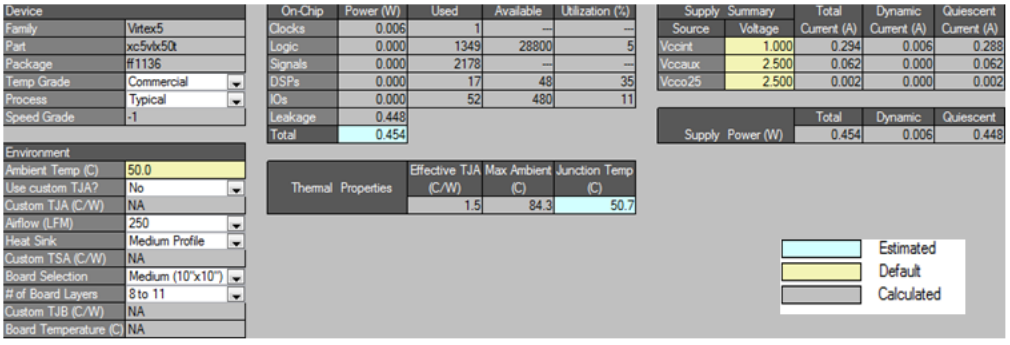

Figure 4.9: The values of energy consumption, thermal properties, voltage and current.

4.4 Conclusion

In this chapter, we present the envisioned Fuzzy decision system model. We have presented a novel caching system for edge devices, that combines size, mobility and cost awareness and continuously optimized using fuzzy logic decision maker model of edge caching. The software implementation shows that the proposed approach improves the hit ratio and the hardware implementation reduces the power utilization. As more different

applications with various contents migrate into the Internet, future mobile edge computing system will experience more variability in requests, content sizes and costs. We believe that the proposed caching strategy can address these challenges.

In the next chapter, we investigate the reinforcement learning as an additional model to improve our current proposal. The reinforcement learning learns with its agents the dynamics of the environment, which will encompass the problem of stochastic dynamics and enhance the quality of caching decision.

Chapter 5

Caching strategy with mRL (modified Reinforcement Learning)

5.1 Introduction

The tremendous growth of Mobile Edge Computing empowers the edge network nodes with more computation capabilities and storage; thus, achieving specific tasks like data analysis and deep learning in edge networks. This allows the possibility of employing artificial intelligence (AI) techniques and machine learning at mobile network edges. Exploiting users context information intelligently makes it possible to design an intelligent context-aware mobile edge caching. Context awareness enables the cache to be aware of its environment, while intelligence enables each cache to make the right decisions of selecting appropriate contents to be cached so that to maximize the caching performance. Inspired by the success of reinforcement learning (RL) that uses agents to deal with decision making problems, in this chapter, we present a modified reinforcement learning (mRL) for content caching in the edge of the network. The proposed solution aims to maximize the cache hit rate and requires a multi awareness of the influencing factors on cache performance. The modified RL differs from other RL algorithms regarding the learning rate that uses the method of stochastic gradient decent(SGD) [18] [59] beside taking advantage of learning using the optimal caching decision obtained from fuzzy rules.

5.2 Presentation of the solution

Despite the efficacy of RL model in many areas, it still has many limitations. The basic RL model does not cope well with domains involving a large space of possible environment states or a large set of possible actions. Therefore, most of the RL models have been applied to highly simplified learning situations [79]. In this context, we define a modified reinforcement learning mRL based caching system for MEC, which considers the content features namely frequency, cost and size and also the device feature like mobility and storage capability. It aims to solve the problem of caching decision making in a realistic way. Modified reinforcement learning is combining RL and Fuzzy logic to embody the idea of exploiting previously acquired knowledge and capabilities from others in order to speed up learning in RL. For example, knowledge about the content distribution, mobile user properties and network conditions can be used by the caching agent. The combination of knowledge methods and RL has significant advantages like constructing more efficient caching policies. On one hand, transfer prior knowledge can be used to help training RL agents, thus making convergence to the optimal caching decision easier. On the other hand, it allows using policies learned by other relative networks with RL agents, thus improving the efficiency and robustness of the current RL algorithm. The modification of the initial RL algorithm consists in using the fuzzy rules as a policy, and in modifying the learning rate. However, the modification of learning rate may alter the conventional method of stochastic gradient descent (SGD) considerably. Typically, we can say that we defined a control of caching decision as a set of equations that maximize the value of the reward function.

5.3 Modified reinforcement learning (mRL) over the edge caching system

5.3.1 The proposed scenario

In this section, we focus on the scenario of caching contents in edge nodes, as depicted in Figure 5.1.

There are C contents, denoted as $c_i = \{1, ...C\} : c \in C$, we note that all mobile users

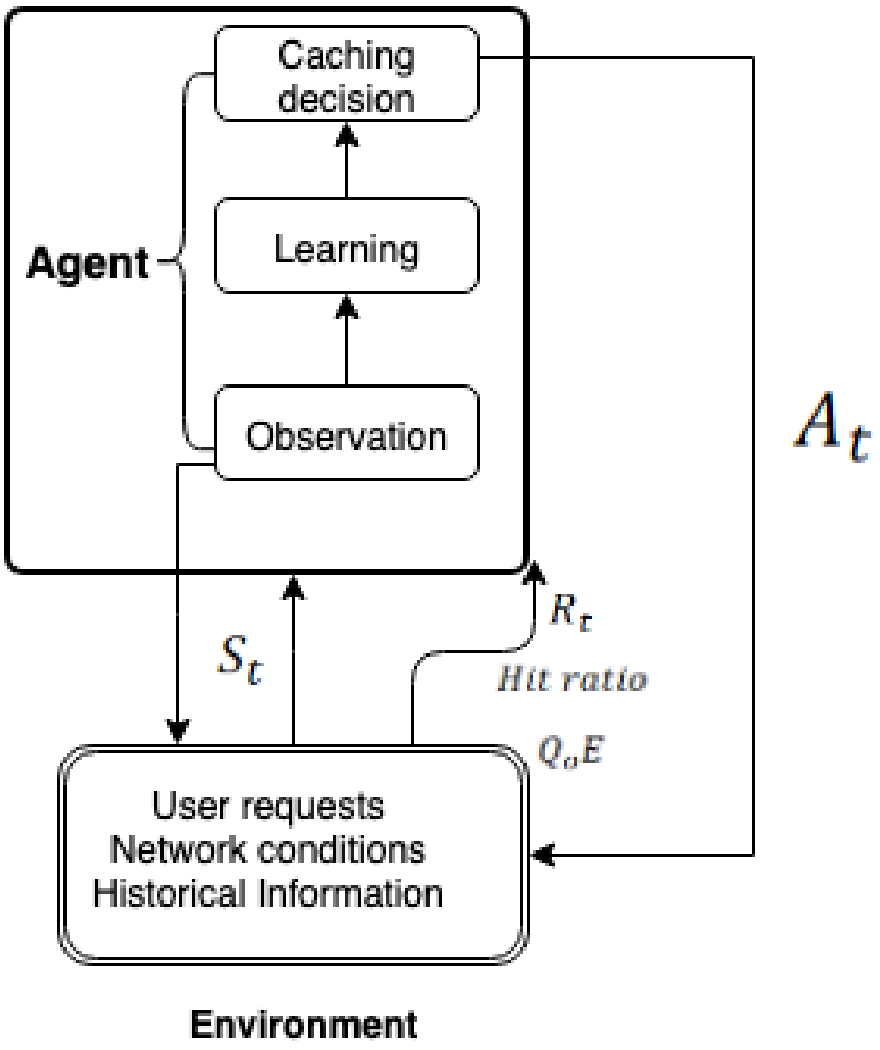

Figure 5.1: scenario of caching using reinforcement learning

in the system may request R requests. The content popularity is defined as *fr*, which is the frequency distribution of content requests from all users. S is the content size denoted as: $s_i = \{1, ...S\}: s \in S$, the user mobility is modeled by the distance between the user and the nearest base station and finally the cost of retrieving the requested content. The contents distribution is described by MZipf distribution function. For each request, the RL agent in the edge node can make a decision to cache or not to cache according to the decision of fuzzy rules, and if yes, the agent determines which local content shall be replaced if there is no storage space. We assume that content distribution, user mobility, cost and average arrival time of the requests are assigned to each content during a t time period. The most important part is how to define a reward function because it directly affects the algorithm performance [65]. In order to design an appropriate reward function to achieve our goal, the key idea is to assign high reward values to content caching actions beside enhancing the cache hit ratios. The gain is calculated by the amount of cache hits to be enhanced by the newly cached content while the loss is calculated by the amount of misses. We illustrate an example of applying mRL to the mobile edge caching, where one MEC server is considered and the storage space of the edge cache is initialized to be enough for caching half of the available contents in the network. The MEC server can serve all the requests directly, according to the arrival time. Initially, the caching policy caches locally the contents according to the priority accomplished by the fuzzy system and to the cache size availability. Otherwise, the cache replaces the content less prior by the highest one. Our aim is to find the optimal caching decision by maximizing the cache hit ratio, that is, the number of contents answered by the edge cache. This problem can be solved based on mRL which requires training an agent for representing the policy, and an appropriate reward function that describes how the agent ought to behave. In other words, reward functions have normative contents that provide what the agents should accomplish.

The detailed features are shown in figures 5.2, 5.3 and explained below :
The environment is modeled as a stochastic finite state with inputs (actions sent from the agent : fuzzy rules and user requests) and outputs (observations and rewards sent to the agent):

- State transition function P($S_{t+1} \mid S_t, a_t$)

- Observation (output) function P(O_t | s_t, a_t)
- Reward function $R_t = \infty$ k=0 γ^k r_{t+k}.

It is noted that the agent observations depend on his actions, which reflects the fact that perception is an active process. The agent is also modelled as stochastic finite state machine with inputs (observations and rewards sent from the environment) and outputs (actions sent to the environment).

- State transition function: S_t = f(S_{t-1} O_t, R_t , a_t)
- Policy/output function: π=P(a_t|s_t)

The agent's goal is to find the optimal policy and the state function so that to maximize the expected sum of discounted rewards.

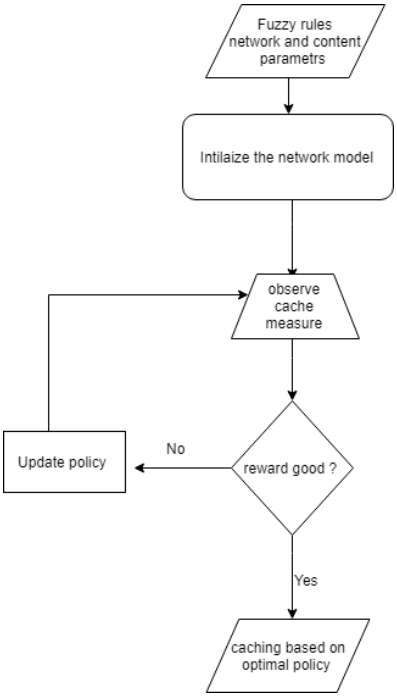

Figure 5.2: block diagram of caching over mRL

5.3.2 Mathematical model

Reinforcement learning is usually preformed using parameterized function approximators to store value functions. RL algorithms are developed and then applied to function

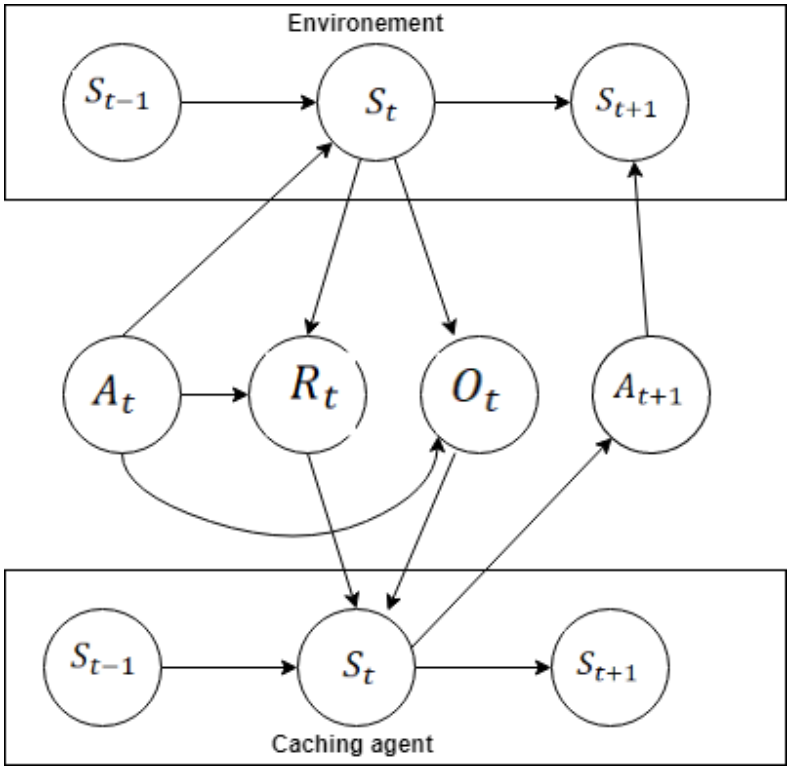

Figure 5.3: caching Agnet process

Table 5.1: List of mathematical symbols

Symbol	description
S_t	State
A_t	Action
π	Policy
R_t	Reward function
V	state-value function
Q	action-value function
θ	weight
α	learning rate
σ	Gradient descent
μ	Mean value
ρ	Variance function
F_t	Fuzzy policy
K	Kernal space

approximators. These algorithms are very complicated and difficult to analyze and to combine with other algorithms. New types of algorithms are derived based on stochastic gradient descent. In order to improve existing types of algorithms from both sides theory and practice, the gradient-descent approach makes it possible to create entirely new classes of reinforcement-learning algorithms. Hence, it became possible to take any of standard algorithms in the field and rederive a new form of it with provable convergence.

It is shown in [98] how gradient descent allows an inelegant, inconvenient algorithm like Advantage updating to be converted into a much simpler and more easily analyzed algorithm.

In case of caching decision with fuzzy policy dynamically, the learning rate should be modified. However, the modification of learning rate may alter the conventional method of *stochastic gradient descent* (SGD) considerably.

We represent the weight adjustment of mRL as follows:
new_weight = (existing_weight – learning rate $\times gradient$), $i.e$:

$$\theta = \theta - \alpha \frac{\delta}{\delta_{\theta'}} \zeta \theta' \tag{5.1}$$

with ζ is the tendency of SGD. if α is too small, the gradient descent can be slow. If α is too large, the gradient descent can overshoot the minimum. It may fail to converge or even

to diverge. We define a cost function for a neural network. The goal is to minimize this cost function. Hence, here it becomes an optimization problem using SGD. Mathematically, if the cost function (or loss function) is L, then the goal is to minimize L. For a convex optimization problem like this, we use the derivation of the loss function ΔL. Thus,

$$\theta = \theta - \alpha \Delta L \tag{5.2}$$

while α is the learning rate.

To consider this principle, we initiate certain modifications of α in the reinforcement learning, which will be a considerable modification over SGD. Precisely, we will modify the flowing parameters for mRL with respect to SGD:

Momentum : it will help normal SGD by adding a function of the direction of the previous step to a current step. The function is usually a range of [0.1]. Hence, the proposed model of RL can make use of the following configuration changes of learning:

- The learning rate will select new learning values at the end of each epoch/session combining statistical optimization and rejection strategy.

We provide further the following modification :

1. Input search space S.
2. Find a mean function μ and variance function ρ
3. Optimizing α, as in the following equation:

$$\sigma_m = {}_{\eta \in S} \alpha(\eta; \mu(\eta), \rho(\eta)) \tag{5.3}$$

5.4 Problem formulation and model development

5.4.1 High level description of mRL

The reinforcement learning based algorithms have been successfully applied to various optimization problems in many domains. However, in order to obtain better solutions for

specific optimization problem like our multi awareness caching system. The core of the modified RL algorithm used in this study is to generate a sub-environment based on fuzzy policy as following:

- Fuzzy policy weight : F_t, θ.
- Input search space S, the number of initial search
- Update the value of α toward an optimum value α^*

We minimize the expected value of the objective in next dynamic instances which implies :

$$policy[f(\theta_t) = policy[f(\mu(\theta_{t-1}), \alpha_t)] \tag{5.4}$$

We update the previous learning rate α_{t-1} to the final value α_t. Then, we perform GD:

$$\sigma p = (fo\mu(\theta_t, \alpha_t)/(\eta\alpha_t) \tag{5.5}$$

We also assume that the optimum value of learning rate and at each step descent value will change towards all the sessions. However, we need to introduce a continuous function with the context of noise, hence:

$$\alpha_t = \alpha_{t-1} - \beta\Delta_\theta f(\theta_{t-1}) + \Delta_\alpha \mu(\theta_{t-2}, \alpha_{t-1}) \tag{5.6}$$

The rule can be interpreted as multiplication. This will be an additive adaptation of changing of λ.

5.4.2 Policy improvement

The caching policy can be defined as a set of dynamic artifacts [81]. This phenomenon becomes obvious as the context of caching particularly should be dynamic depending on the condition of network traffic. Hence, the modification of RL also demands an essential formal model to cope up with respect to dynamic policies.

5.4.2.1 Formal model of policy improvement

We define a finite state space S for caching policies including all the possible actions as the policy of caching is stochastic. Therefore, to measure the random changes of the state space of caching denoted by S_c, we introduce a transaction kernel function. For simplicity, we assume a finite set of caching policies and we have as follows:

$$\{K_{sc} : \tau \to [0, \infty]\} < \infty \tag{5.7}$$

where K_{sc} is the kernel state space for caching.

In the dynamic context of caching, we may have more than one context. This additional context either can be combined or mixed to derive the improved policy. Let there are two such transitional finite kernels to make decision for caching from S_c to τ. Therefore, to evaluate these two kernels, we need to multiply them. Hence, the final policy (as improved) can be formed:

$$K^Q_{sc1} \otimes K^R_{sc2}(S_Q, A) = \int_\tau K^Q_{sc1}(S, dt). \int_u K^R_{sc2}(S_R, t)du \times 1A(t, u) \tag{5.8}$$

- If the search space is predefined from a stochastic caching mechanism from τ to u for all t $\in\tau$ u $\in U$.

- If the policy for both kernel K^Q_{sc1} and K^R_{sc2} can be combined by integration, the policy π chosen randomly for caching decision is known as greedy.

Therefore, if there is a caching context with all four parameters, then obviously we will select a policy that gives a maximum reward. Therefore, the estimated policy for caching decision to a particular network content is :

$$P_E = \frac{N_E}{Totalnumberoftrailiyparameters} \tag{5.9}$$

In equation 5.9 , N_E is the number of time this policy has been used and the training parameters are out of our four parameters. Among these parameters, the one that has the highest gain at a particular instance allows us to find the number of iterations that the

same policy can trigger; which will assist us in decision making. Hence, as a high level step, we formulate the following algorithms:

Algorithm 2 Policy iteration scheme

$fr \leftarrow 0$;
$P_E \leftarrow 0.5$;
trails $\leftarrow$P$_E \times$ntrails;
for (int i = 0; $i \leq$ntrails; i++)
$trails \leftarrow$P$_E \times$ntrails;
end for
If ($fr \leq$trails)
detect greedy ();
fr ++;
else
select A();

Algorithm 3 mRL Caching policy for edge node

Input: Parameters of the system including dynamic conditions: Mb, Fr, CO and Cr
initialization: initialize the network model: content and requests, fuzzy rule table
Learning Loop
Choose an action a_t from an action space A following policy based on fuzzy caching system $\pi = \mathrm{P}(a_t|\mathrm{s}_t)$
Measure the delay of downloading and cache hit ratio according to (A1)
Apply greedy policy iteration scheme according to (A2)
Update fuzzy rule table
Generate the new caching policy
EndLoop
Cache content based on the optimal policy

5.4.2.2 Reward function

Recently, there has been novel technologies that replace prediction with a much more efficient way known as reward functions. Reward functions are used for reinforcement learning models and allow to obtain the final results as a conclusion instead of prediction. For decision making problems, prediction is not the only input as the other fundamental input is judgment.

For our caching system, *mRL* treats learning and content placement/replacement as a whole operation. The caching policy adopted by the mRL agent is trained with observations, based on a reward resulting from its actions that relate to the factors that affect caching

performance such as mobility. Usually, this reward should be chosen in a way that covers a wide range of factors that can affect the performance such as offloaded traffic or QoE.

Finding the best reward function to reproduce a set of observations can be implemented by the maximum likelihood estimation, Bayesian, or information theoretic methods. The system reward represents the optimization objective. In our work, the objective is to maximize cache hit ratio. For our caching approach, we have all the variables that can go into known values. The time of retrieving is determined while distributing the contents and the users requests (including the associated parameters of each).

In the scenario of mRL, when the system state s(t) and the system action a(t) are given, the priority can be determined. When the content requests of a typical user arrive, the system can acquire the knowledge from the fuzzy rules, whether the cache should preform the operation of placement or replacement and also satisfies the requests. We defined a network model where several MECs are deployed. At the network edge, the cache with limited resources is connected to the cloud via the backhaul link. Let $C = 1, 2, \ldots, c$. C denotes the content provider which is located in the cloud content center. For simplicity, each cache can store $X (X <= C)$ contents. Meanwhile, a time-slotted system is considered. Let $N = 1, 2, \ldots, n$ the users served by the MECs during time slot t. The distribution of user requests and contents is determined by the ZIPFs distribution function [32]. Let $d_c(t)$ denote the amount of the instantaneous user requests towards C contents during time t. Let $I_t(t)$ denote the cache indicator for the considered MECs during a time slot t. Specifically, I_c (t)=1 (with a corresponding priority) indicates that the c^{th} content is cached during time slot t and $I_c(t) = 0$ otherwise. Correspondingly, the caching decisions are made according to fuzzy caching system where *cache hit rate* is a measure of the caching performance. Hence, it is used as reward function in the mRL – caching framework. In conventional reinforcement learning, an agent learns how to optimize its behaviors in uncertain environment by executing control polices experiencing the decision of rewards and improvising the policy based on the reward. Without satisfyingly strong reward function or decision, learning may look very difficult for the present problem of caching the state space (combination of mobility, cost, frequency and cache size) that could be too extended. Therefore, the time of retrieving information should be maximum and the learning would be

difficult. This type of cases may shape the reward decision of caching as extremely sparse. We can approach three different reward decisions on caching itself as in the following:

1. A reward decision with the sparse value that is multiplied with certain values compared to the goal state. However, this kind of reward decision may also slow down the learning because the agent needs to achieve many actions before getting the decision of good caching or not effective caching

2. Reverse reward in case of collisions where the constraints of caching may suffer from the situation where keeping cost always lower cannot optimize the other values of constraints.

3. Zero reward for any other state (ideally, there may be some situations where no decision can be made satisfying the cost as lower).

The reverse reward can be a primary objective in terms of evaluation of the distance and the time of information to be propagated from an initial base station to a nearest base station. It has been observed that collision may happen not for the initial iteration, but when both the number of iterations are increased and either of the three parameters like MB, Fr and size of the cache are made dynamic. In most of the cases, the designing of the reward function in terms of decision is amalgamated with the respect to the procedure of designing state space. For example the cache is a time depending problem. Therefore, the distance covered to reach the nearest base station can make a good reward decision or not effective reward decision. To simplify, we only mentioned a generic reward function customized with the present problem:

- The state value function $\nu\pi(s)$ gives the long term value of state S when following the policy π

- The action value function $q\pi(s,a)$ is the expected return starting form state s taking an action a as in the following:

$$\nu\pi(s) = X_a \in A\pi(a|s)q\pi(s,a) \tag{5.10}$$

The expected return of decision for caching will be dynamic for caching process. This is because the state of constraints except the cost will change from time to time. This relation is composed with state value function $\nu\pi(s)$ and the action value function $q\pi(s\prime, a)$.

In the given expression below, A belongs to the all number of actions and X denotes the all number f states to achieve the based reward decision of caching.

5.5 Data and result analysis

A catalogue of $N = 10^3$ contents has been generated, where each content is accompanied with its size, frequency and later on the cost and mobility are calculated. The cache occupancy is the percentage of the content size on the free space in the cache. A cache size of 600MB is used and randomly we set a size for each content. Each time the user requests a content, the frequency is calculated depending to the catalogue size and the number of requests. With U number of users $U = \{\text{u},\ldots u_N\}$: $u \in U$, the mobility is formed as the distance $d = XY$ between the users in the points $X = \{x_1, x_2 .... x_U\}$ and the base station in the point Y.

$$Mb = |Y - X_U| \tag{5.11}$$

The cost to retrieve each content is the time incurred to retrieve the content to the end user. This time is calculated according to the maximum bandwidth required to provide peak data rates of at least $B = 0.2$ Mbit/s, and the distance $d = XY$ between the end user and the base station where the cache is located.

$$Cost = \frac{Mb}{B} \tag{5.12}$$

A resource from the catalogue content is an object that contains an ID, a size and a payload that refer to the type of the content we have (video, image, audio). For simplicity, it is set to a string value. For the distribution of requests and contents, studies have confirmed *Zipfs* law as an appropriate distribution model for access pattern to contents on the Internet such as videos hosted on YouTube or peer-to-peer file sharing systems like

Bit-Torrent. According to *Zipfs* law, a small fraction of popular web objects attracts most user requests, which is favourable for the efficiency of small caches. *Zipfs* distributions of finite support for a content catalogue of N contents with request probabilities $Z(r)$ have been determined for the objects popularity ranks :

$$Z(r) = \alpha r^{\beta} with : r^{\beta} < 0; \beta = Z(1) = 1/\sum_{r=1}^{N} r^{\beta} > 0 \tag{5.13}$$

where β is an adaptive shape parameter and α a normalization constant. R represents the users requests with $R = 10^3$, with α set as : $\alpha = 0.75$.

Based on this distribution law, sever characteristics curves can be generated. The characteristics of the curves are defined as following:

- Choosing an optimization function to balance the four optimal parameters like frequency, mobility, size and cost. However, the cost as well as the mobility always have to be kept as minimum as possible.
- Deviation of cache size and time of caching are expected.
- The optimization of objective function is repeated to balance mobility and cost.
- It is also expected that there will be reference mean and there will be actual mean between the max and min values of bandwidth. Hence, therefore setting histogram plots could be worthy to demonstrate these variations.

5.6 Results Discussion

We should mention that unbalanced data have been used for two reasons. First, the scenario is a multi-objective function [60]. Second, using unbalanced data requires methods to solve the problems of optimization under nonlinear constraints of inequalities like Karush-Kuhn-Tucker conditions (KKT) which is an optimization problem with interval-valued objective function [93] [1].

For general characteristics, the deviation of cache size may occur with respect to time of caching (see figure 5.4). In this figure, there are two segments of curves. one in red color

demonstrates that the cache size is 0.01 unit and the time of caching is only variable (in ms), then the slop of the curve becomes linear and deviates more toward a sustain value. After a period of time estimated to 3 ms, there is no significant change in deviation of cache size. However, in the same figure, the upper segment shows cache size between [0.02-0.03] units. If it differs, then the slop of the curve can only be flat after 25 to 30 ms time. This describes that the deviation of cache size while keeping the time of caching fixed can be a significant characteristic to the performance of this model.

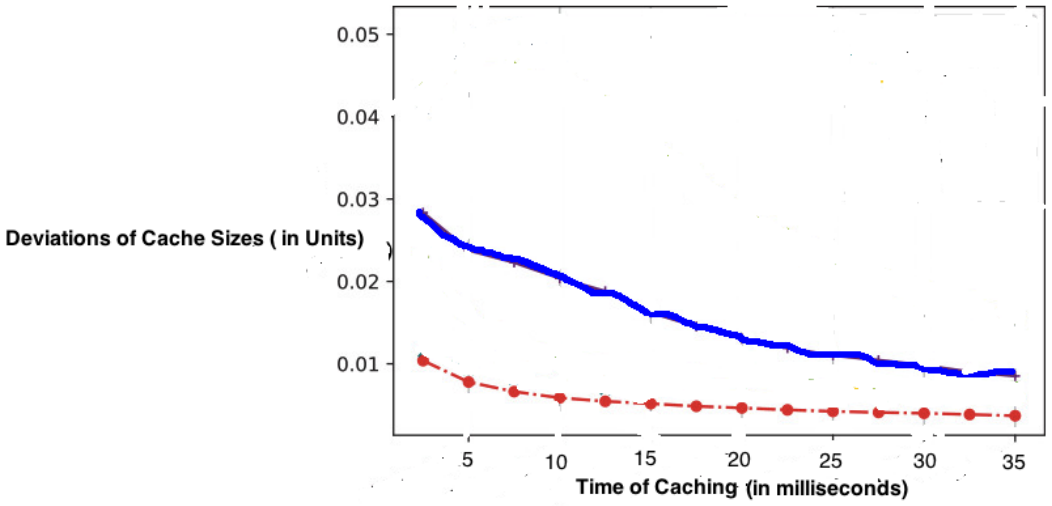

Figure 5.4: Cache size over time

In the histogram plot of figure 5.5, frequency is a range of 10^3 samples. However, these large samples of ranges of frequency may not serve the RL model as part of the given data.

It also shows that the axis x of the plot of histogram can be sustained and becomes non significant after 2 $\times 10^4$ values of frequency. Therefore, there is a requirement to bifurcate the maximum and the minimum range of frequency by using the actual mean value as a reference as shown in the plot. This will help to generate the data proposed for this model accordingly. After formulating the data generation and investigating the tendency of cache size with time, the given performance has been focused to a multi objective optimization problem. Here, the four parameters, used to analyze the trusted caching decision, can vary and significantly impure the results.

For example in figure 5.6, it is shown x axes containing an iteration count and y axes containing an optimization function. In this case, the optimization function can vary with cost and mobility, although two variables are kept as minimum with respect to the remaining two variables. Finally, this will produce at least two optimization functions with respect to the cost and mobility. Therefore, the data part here should be unbalanced to yield an optimal solution of caching. It is shown in the figure that the training data in this model has been occurred from fuzzy model and therefore there may be substantial possibilities to present a data in a mixed cluster of any of these four variables. These features provide that there is a necessity of final clustering on the unbalanced data for optimal caching decision. The figure has demonstrated three dots where clearly the middle part shows dense clusters of any of the four variables line in this region. However, the left and right sides of those dots are clearly isolated from these clusters, which means that the middle dense cluster can be suitable for a good caching decision whereas the other may not. These all dots represent the combination of any of the four parameters without any balance ratio. This necessitates the requirement KKT optimization in the analysis part of these data. First of all, we went through solving convex maximization problems under inequality constraints. Our particular caching problem led to constrained equations with 4 parameters (Mb, Cr, Co, Fr). To determine their optimal values, we used the KKT condition. We assumed that three constraints of our problem were active. We, therefore, eliminated the inactive constraint. Since all these equations happened to be linear, we obtained a solvable algebraic problem. We followed a simple linear regression model according to Figure 5.7.

In this figure, the three dimensional plot designates the relation between cost and

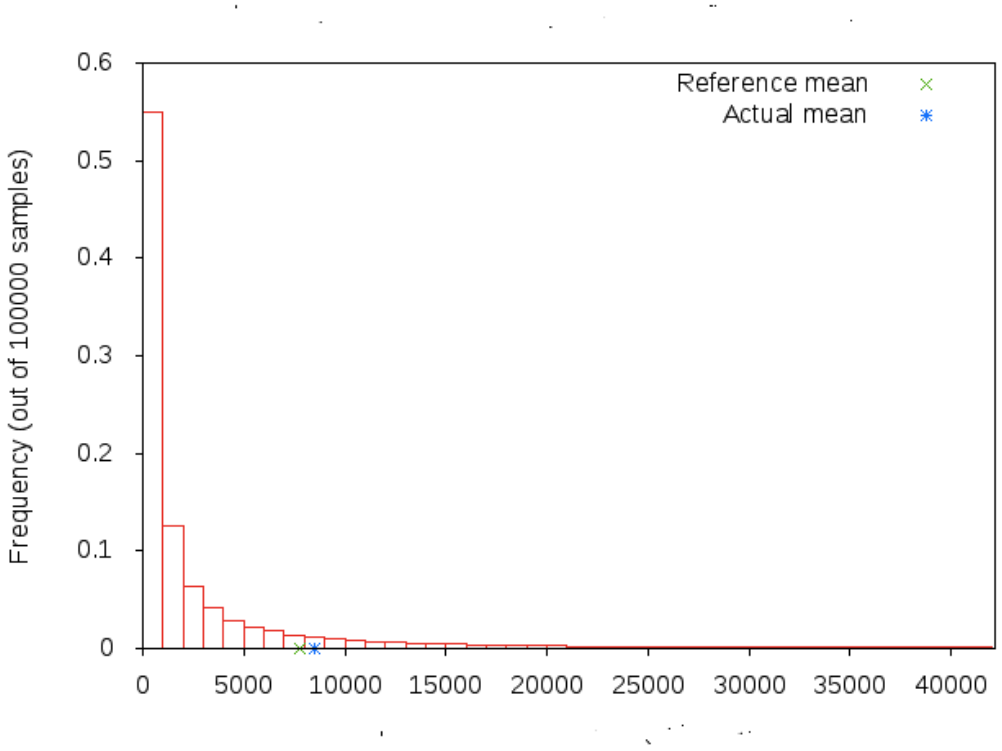

Figure 5.5: Frequency histogram

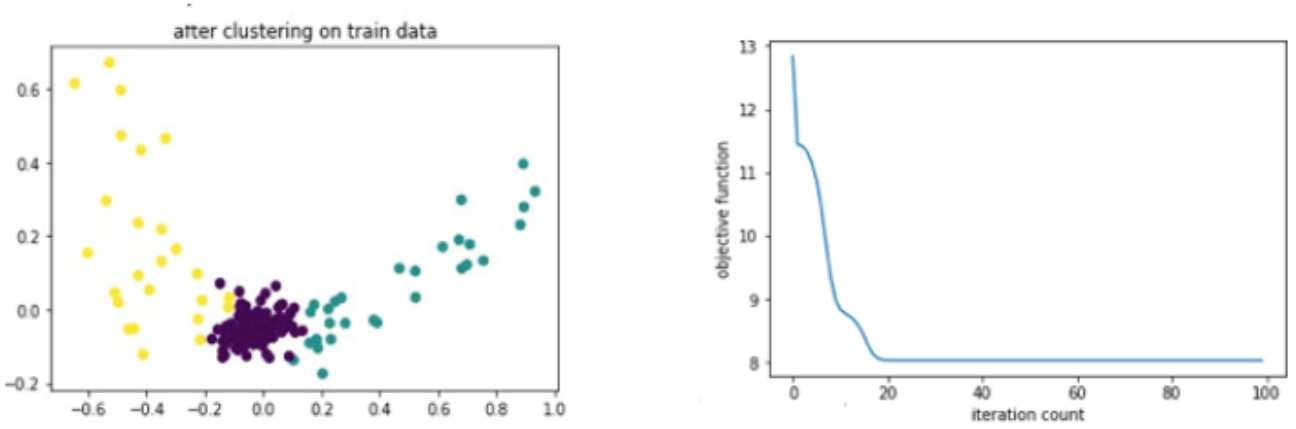

Figure 5.6: Caching decision evaluation and the optimization of the objective function

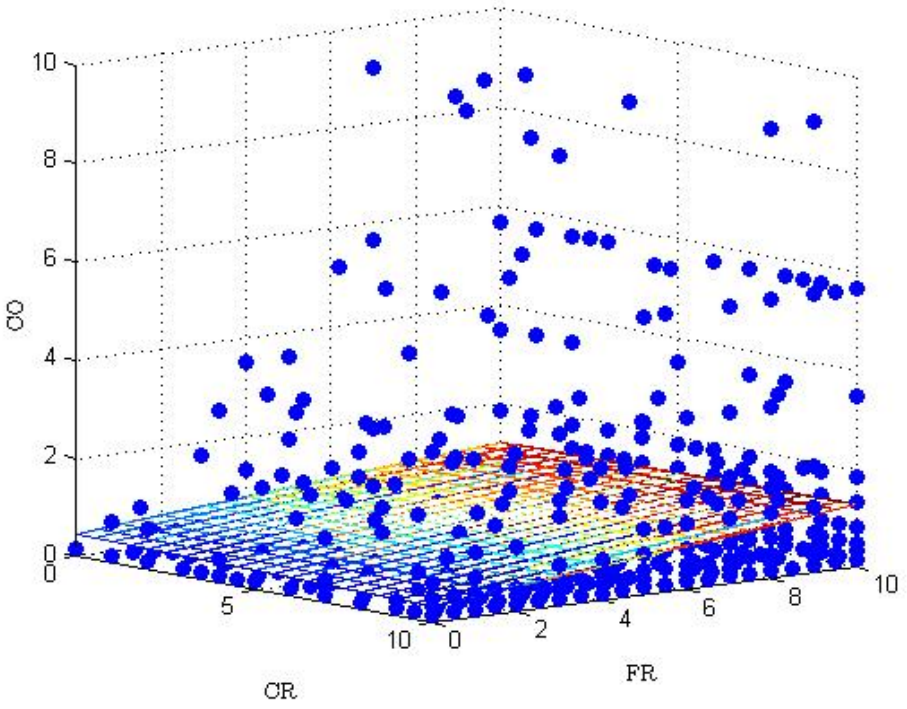

Figure 5.7: Relation between cost, frequency and size

frequency and in the below part we can notice that the regression becomes dense. However, the upper part becomes scattered and we observe that the generated data have not been balanced enough to produce a good outlier decision for caching. Outlier means that, each time, the data and the iterations of caching session should be proportional. Data signifies the combination of four parameters as it is unbalanced. Therefore, the relationship between cost and frequency may not provide any significant decision for caching. Additionally, it can be noted that for bandwidth or frequency, we have referred the histogram plot for reference mean and actual mean of the maximum and minimum frequency. Therefore, we require more optimal clustering for Figure 5.8.

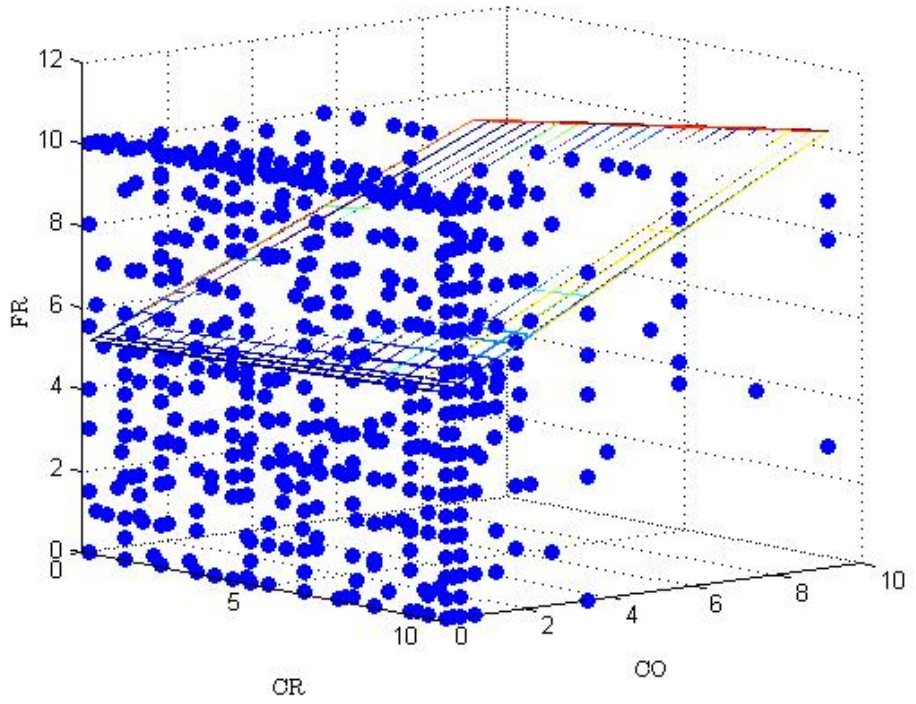

Figure 5.8: Optimal clustering for cost and frequency and size

Here, we observe that there are three axes in the given plot where Fr of the content CR and CO of the content are plotted based on the sample data generated. In the given figure, there are three sections based on the distribution of the optimization function:

- The dense blue dots occupy the Fr and the CR region of the plot. This entropy will distribute certain scattered blue dot from CR toward the mapping region of the cost.

As it is mentioned, the cost should be kept as minimum. Therefore, very few blue dots are available in the cost region satisfying the variable FR and CR according to the generated data.

- This plot is a broader outlier analysis of clustering. It describes the plot regions into different visible sections. However, due to the absence of real effective cost data, it is not possible to find more blue dots which may represent appropriate cost of caching with respect to the other parameters.
- The modified RL thus can distribute a map value for state - action and reward clustered for appropriate decision of caching.

In the both figures:

- 3D distribution of input data has been shown using scatter diagram. Additionally, these following values are analyzed :
  - Means, Median, Std/Variance all attributes, data distribution and histograms.
  - Co-variance of pair of attributes.

Pairwise linear regressions.

- $Fr = a + b \times Cr$.
- $Fr = a + b \times Co$.
- $Fr = a + b \times Mb$.
- $Cr = a + b \times Co$.
- $Cr = a + b \times Mb$ and $Co = a + b \times Mb$
- Multiple linear regressions ($Fr = a + b \times Cr + c \times Co, Fr = a + b \times Cr + c \times MB, Co = a + b \times Fr + c \times Mb, Fr = a + b \times Cr + c \times Co + d \times Mb$)
- Multiple quadratic regressions $Fr = c_0 + c_1 \times Cr + c_2 \times Co + c_3 \times Mb + c_4 \times Cr \times Cr + c_5 \times Co \times Co + c_6 \times Mb \times Mb + c_7 \times Cr \times Co_c 8 \times Cr \times Mb + c_9 \times Co \times Mb$

To identify possible clustering, results are shown using graphical representations. For all regressions, we have used Least Square optimization techniques for the learning of model constants, i.e. regression coefficients.

The scatter diagram is showing the distribution of the data in all directions. There is no such biased or prominent grouping of data. From the graphical and tabular views, it can be easily found that some models are working with certain errors. This is also to emphasize that apparently it seems to formulate a perfect optimization function based on four parameters. However, in real time, it may not be suitable too.

5.7 conclusion

In this chapter, we presented a realistic decision scheme for caching scenario using a modified reinforcement learning model. We also described thef analogy behind the data and demonstrated the four given components as a holistic performance of the model followed by an optimization characteristic curve. The next chapter addresses the simulation part in order to verify that our proposed algorithm works in realistic situations.

Chapter 6

Simulation : caching strategy for MEC/ICN based architecture

6.1 Introduction

Information-Centric Networking (ICN) is a promising network that is based on shifting the main network host identifiers to location content identifiers. Caching capabilities are one of the common key feature used by all ICN proposed architectures. In this chapter, we present a simulation for our proposed caching system in ICARUS, which is a Python-based caching simulator for ICN. The simulation tool allows us to evaluate our caching system for any ICN implementation.

6.2 Architecture and design

In order to compare caching strategies for ICN under simulation environment and common evaluation scenarios, we implemented our fuzzy caching system in ICARUS simulator written in Python as programming language, which provides several advantages such as the high-level syntax [77]. ICARUS has been designed to satisfy two main non-functional requirements: extensibility and scalability. Figure 6.1 shows the High-level architecture of the simulator that contains the following phases :

- *Configuration and scenario generation:* This phase includes all the steps required to set up a fully configured network topology and a random event generator for the simulation. The scenario generation is based on the Fast Network Simulation Setup

(FNSS) toolchain [77].

- *Orchestration:* In this phase, the simulator uses the configuration file in order to extract the range of parameters (e.g. cache sizes, cache policies, content popularity distribution) and starts primarily experiments with all the chosen parameters.
- *Execution:* After experiments, an actual execution of the simulation starts up with an instance measurement of the various metrics.
- *Results collection and analysis:* At the end of the experiment and execution, results are collected and aggregated with the possibility to calculate confidence intervals and plot results.

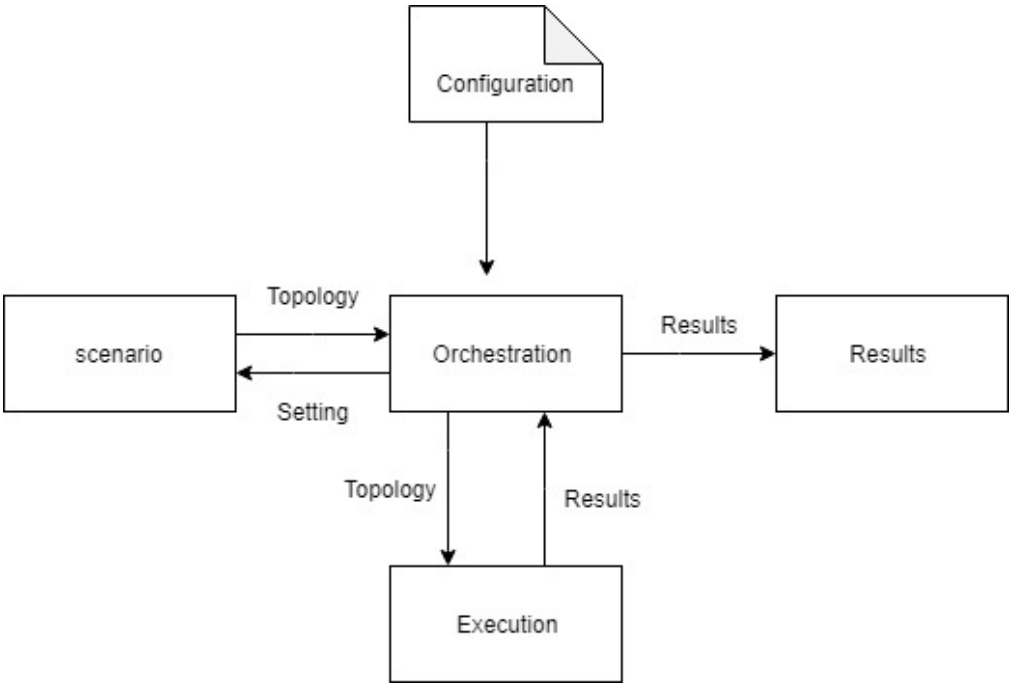

Figure 6.1: High-level architecture and workload of the simulation tool

6.3 Evaluation of Fuzzy Caching System (FCS) based algorithm

This simulation tool needs all the parameters defined in FCS algorithm as described in the preceding chapter. Hence, we perform simulation experiments for our fuzzy caching

system and for the following caching strategies as presented in table 2.1 of Chapter 3:

- Least Recently Used.
- Leave Copy Everywhere.
- First In First Out.
- Random.

We varied the parameters of the common evaluation scenario (Table 6.1). For each simulation experiment, three runs are performed and before running the simulation, the requests are placed randomly in the nodes of the topology. Table 6.1 shows also the list of metrics to be measured in the experiments: Cache Hit which is the ratio of the number of cache hits to the number of lookups, usually expressed as a percentage. Cache size expressed by the total size of network cache as a fraction of content population and latency, which is the delay taken to deliver a content. We varied the range of alpha values of the Zipf distribution function used to generate content requests. We used a catalogue of N = $2 * 10^5$ with a number of requests estimated by $R = 10$ requests per second (over the whole network). The topology used to evaluate the different caching mechanisms is GEANT which is an ISP level topology [36]. The first GÉANT network was launched in 2000 (see Figure 6.2) and has stayed well ahead and used in many works like in [73], [76] and [90].

Table 6.1: Simulation Environment

Simulation	
Runs	2
Evaluation metrics	Cache hit ratio, Cache size, latency
Cache configuration	FCS, LRU, FIFO,
Topology	GEANT topology
Popularity Model	MZipf(α = 0.6, 0.8, 1.0, 1.2, β = 0)
Cache size	0.004, 0.002, 0.01, 0.05

6.3.1 Trace parsers

ICARUS simulator provides a set of functions allowing to parse content traces from the most common datasets. These traces are used to feed a request generator and simulation.

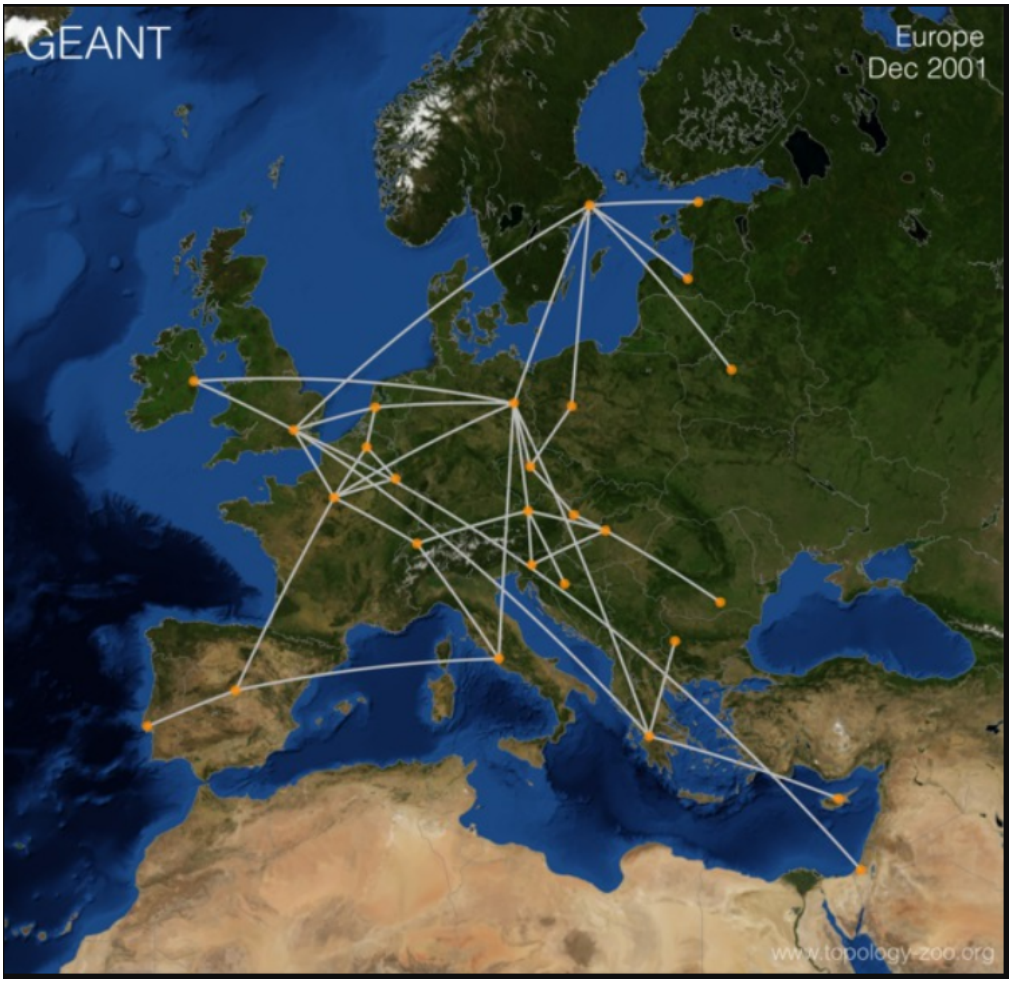

Figure 6.2: GEANT Topology
[1]

ICARUS supports parsing from two data formats. First, it supports the parsing of logs generated by a Squid proxy server. This is one of the most common open-source HTTP proxy available. This is also the format in which the traces provided by the IRCache project are made available. Second, it supports parsing requests from the Wikibench dataset [86]. This is a dataset of requests received by all Wikimedia websites over a period of time between 2007 and 2008 [77].

6.3.2 Performance evaluation

Experiments have been run with LRU, FIFO and Fuzzy caching system. We evaluate the performance in a range of content catalogue sizes from 10^3 to 10^7. It should be mentioned that the cache size is calculated as the ratio between cumulative cache size and content catalogue. Hence, a scenario with a large content catalogue has also a large cache size. In order to assess the scalability of the fuzzy caching system and to validate its fitness for running in large scale ICN environment, we evaluate its performance in terms of both cache hit ratio and latency by varying many conditions. In particular, our analysis focuses on measuring the cache hit ratio and the latency of: FCS, LRU and FIFO with various catalogue and cache sizes. Figure 6.3 shows the cache hit ratio and the latency alteration for GEANT topology over the cache size with a fixed Zipf's parameter $\alpha = 1.2$ and a varied content population ratio (total size of network cache as a fraction of content population). We notice that FCS performs better than FIFO and LRU and the latency decreases with the growth of the cache size. We should pay attention that the bigger the cache size ratio is, the easier to achieve good caching results.

We also study the cache behaviour under various values of α in order to cope different workloads. Regarding the hit ratio achieved by FCS, we can see that it increases with the augmentation of α value and marks better values than FIFO and LRU. In terms of latency, the three caching strategies have approximately the same high value equal to 0.9, then it starts decreasing for the three caching strategies, noticing that the fuzzy caching system shows less latency value. In high popularity scenarios ($\alpha = 1.1 and \alpha = 1.2$), the level of performance of FCS increases.

Finally, in Figure 6.5 we fixed the size of the cache and α value and we measured the

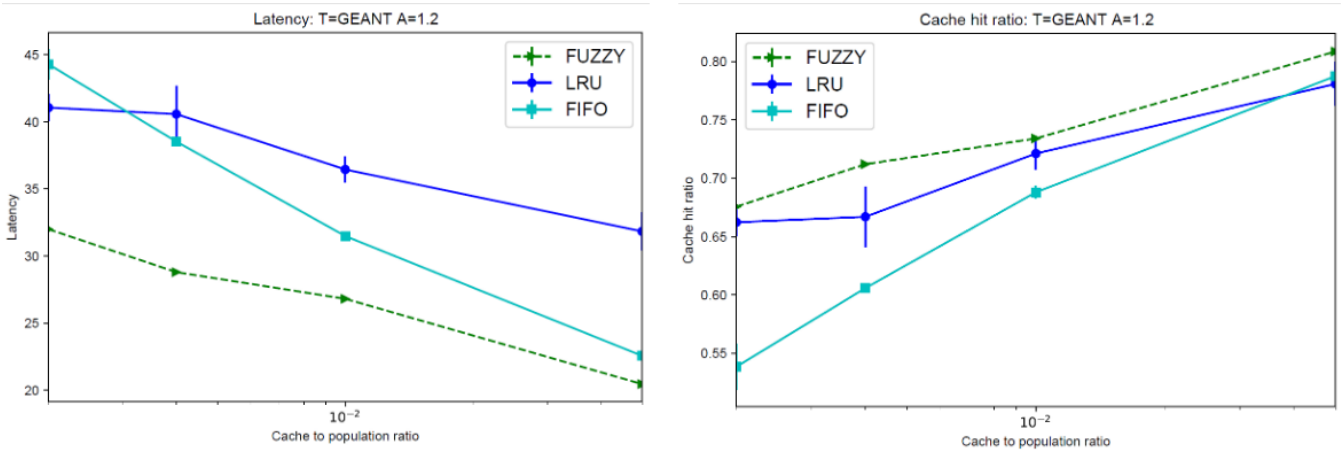

Figure 6.3: cache hit ratio and latency over size with a fixed α

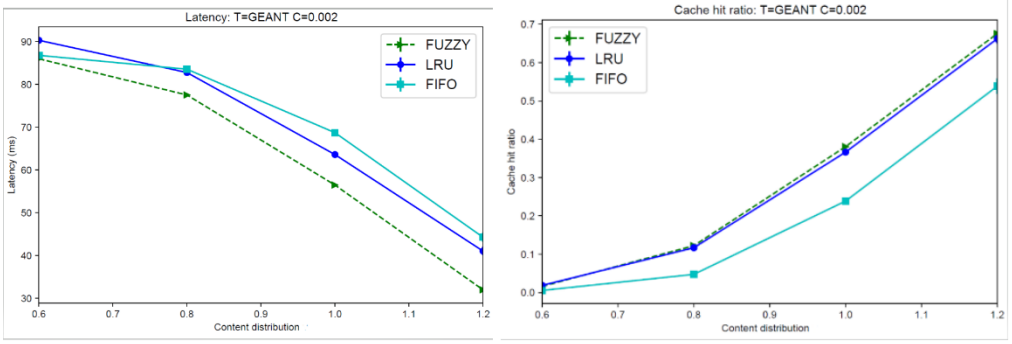

Figure 6.4: Cache hit ratio and latency over content distribution with a fixed cache size

cache hit ratio over time, where it appears that FCS achieves 0.7 which is the highest value compared to LRU and FIFO.

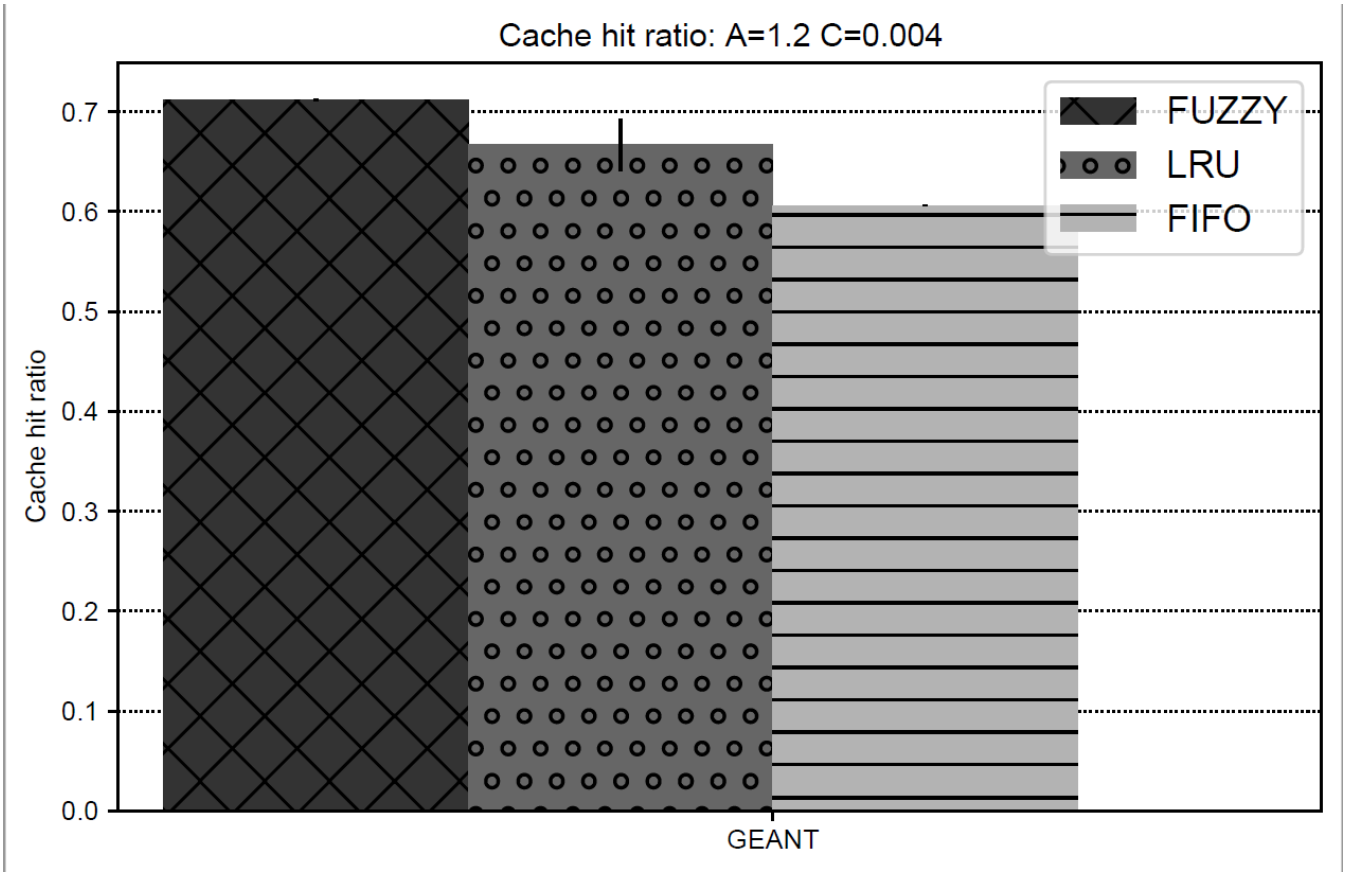

Figure 6.5: Cache hit ratio with fixed size and α

6.4 Conclusion

In this chapter, we compared the most relevant ICN caching strategies (LRU, FIFO) with our solution. We first provide a common evaluation scenario to evaluate the strategies under the same simulation environment. Then, we implemented our fuzzy caching system. We summarized the results that show that our proposed fuzzy caching system outperforms other caching strategies but it still has a high computational cost even if the cost of our caching solution is better than the others. However, it could be a good candidate to be used as caching strategy for ICN. As the evaluation of ICN caching is an important topic,

we aim to complement this work and contribute to the future deployment of the ICN architecture.

Conclusion

In this thesis, we have proposed a new intelligent caching system for mobile edge computing using fuzzy logic and reinforcement learning model. This caching system was built along two steps : The first step is supplying effective caching services in the highly resource constrained and dynamic mobile environment. We proposed an energy-efficient fuzzy caching technique for edge devices that enhances the cache hit ratio and that consumes less power. In the second step, we extended the caching system algorithm with a reinforcement learning model through which we have modified reinforcement learning in order to have an optimal caching decision. We recall, below, the remaining chapters and the contributions of our thesis. We introduced six chapters spread over two main parts, the first part is an introductory part that contains the basic concepts of the thesis and also the state of the arts. This part is organized as follows :

- *Chapter 1*: Presents and describe ICN environment and MEC characteristics.
- *Chapter 2*: Contain the background related to the intelligent mechanism used in the thesis to formulate the intelligent caching mechanism.
- *Chapter 3*: The stat of the art of the existed caching strategies in general , ans in particular using fuzzy logic and reinforcement learning.

the second part contains three chapters each describe and present a contribution. The first contribution presented in chapter 4 is a novel energy efficiency and multi awareness caching strategy using fuzzy logic. We took into consideration four influencing factors related to the network nature and the user behaviour. To cuts overall energy requirements, we relied on a hardware implementation of the caching system using Field-Programmable

Gate Array (FPGA) as an alternative computational architecture. The results showed a better performance through a less power consumption compared with some existing caching strategies. In chapter 5, we presented the second contribution that demonstrated a realistic decision scheme for caching scenarios by using a modified reinforcement learning. Elementary modification in reinforcement learning was done to incorporate RL in the mobile edge caching system. Hence, a formulation of a hybrid model with fuzzy and RL was required. The modification customized the four influencing factors and the dynamic properties of the proposal can support the different caching scenarios. We demonstrated that to keep cost minimum for the whole system and to vary different other parameters, a well balanced machine learning based optimization function was necessary. Finally, Chapter 6 which is the third contribution where we simulated the caching system in an ICN simulator to check the robustness and the ability of applying this system in ICN environment. The results of simulation show that our fuzzy caching system outperforms the other caching strategies used in this simulation. In conclusion, the work in this thesis addressed the challenge of caching for mobile edge computing, in ICN environment. The proposed intelligent caching strategy has showed a good performance and proved that it can be the best candidate to incorporate in the design of MEC/ICN environment. We aim later to use our fuzzy caching algorithm in MEC systems to take advantages from MEC characteristic like processing and data analysis in order to bring intelligence at the edge of the mobile network, hence reducing latency and enhancing the offered capacity. We aim to propose a new architecture that aligns and integrates the caching system with the MEC/ICN based architecture by introducing a system that enriches the network with caching-oriented orchestration capabilities. This opens new opportunities on jointly managing caches by allowing processing and intelligent operations to be triggered by MEC services.

Publications

International Journals

1. S. Mehamel, S. Bouzefrane, S. Banarjee, M. Daoui, V. E. Balas, "Modified Reinforcement Learning based-Caching system for Mobile Edge Computing", International journal of Intelligent Decision Technologies, IOS Press, accepted to appear.

International Conferences

1. S. Mehamel, S.Bouzefrane, K.Slimani, M.Daoui. "Energy-efficient hardware caching decision using Fuzzy Logic in Mobile Edge Computing", The IEEE 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloud'2018), August, 2018, pp. 237-242,Barcelona, Spain, (DOI:10.1109/W-FiCloud.2018.00045)

2. S. Mehamel, S.Bouzefrane, K.Slimani, M.Daoui. "New caching system under uncertainty for Mobile Edge Computing", 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), 10-13 June , 2019, pp. 237-242,Rome, Italy, (DOI:10.1109/FMEC.2019.8795356)

Bibliography

[1] Karush-Kuhn-Tucker (KKT) Conditions. In *Encyclopedia of Operations Research and Management Science*. 2013. doi: 10.1007/978-1-4419-1153-7{_}200359.

[2] WEB CACHING AND RESPONSE TIME OPTIMIZATION BASED ON EVICTION METHOD. *International Journal of Innovative Research in Science, Engineering and Technology*, 2013. ISSN 2319-8753.

[3] Noor Abani, Torsten Braun, and Mario Gerla. Proactive caching with mobility prediction under uncertainty in information-centric networks. In *ICN 2017 - Proceedings of the 4th ACM Conference on Information Centric Networking*, 2017. ISBN 9781450351225. doi: 10.1145/3125719.3125728.

[4] Ibrahim Abdullahi, A Suki, M Arif, and Suki Arif. Cache-skip approach for Information-Centric Network. ISSN 1819-6608. URL `https://www.researchgate.net/publication/284078557`.

[5] Lada A Adamic and Bernardo A Huberman. Zipf's law and the Internet. *Glottometrics 3*, 2002.

[6] Arif Ahmed and Ejaz Ahmed. A survey on mobile edge computing. In *Proceedings of the 10th International Conference on Intelligent Systems and Control, ISCO 2016*. Institute of Electrical and Electronics Engineers Inc., 10 2016. ISBN 9781467378079. doi: 10.1109/ISCO.2016.7727082.

[7] Jaafar Alghazo, Adil Akaaboune, and Nazeih Botros. SF-LRU cache replacement algorithm. In *Records of the IEEE International Workshop on Memory Technology, Design and Testing*, 2004. doi: 10.1109/MTDT.2004.1327979.

[8] Andrea Araldo, Michele Mangili, Fabio Martignon, and Dario Rossi. Cost-aware caching: Optimizing cache provisioning and object placement in ICN. In *2014 IEEE Global Communications Conference, GLOBECOM 2014*, 2014. ISBN 9781479935116. doi: 10.1109/GLOCOM.2014.7036957.

[9] Mikhail Badov, Anand Seetharam, Jim Kurose, Victor Firoiu, and Soumendra Nanda. Congestion-aware caching and search in information-centric networks. In *ICN 2014 - Proceedings of the 1st International Conference on Information-Centric Networking*, 2014. ISBN 9781450332064.

[10] Bitan Banerjee and Chintha Tellambura. Study of mobility in cache-enabled wireless heterogeneous networks. In *IEEE Wireless Communications and Networking Conference, WCNC*, 2017. ISBN 9781509041831. doi: 10.1109/WCNC.2017.7925705.

[11] M. Barthélemy. Betweenness centrality in large complex networks. In *European Physical Journal B*, 2004. doi: 10.1140/epjb/e2004-00111-4.

[12] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Living on the edge: The role of proactive caching in 5G wireless networks. *IEEE Communications Magazine*, 2014. ISSN 01636804. doi: 10.1109/MCOM.2014.6871674.

[13] Arkaprava Basu, Mark D. Hill, and Michael M. Swift. Reducing memory reference energy with opportunistic virtual caching. In *Proceedings - International Symposium on Computer Architecture*, 2012. ISBN 9781467304757. doi: 10.1109/ISCA.2012.6237026.

[14] Harald Beck, Bruno Bierbaumer, Minh Dao-Tran, Thomas Eiter, Hermann Hellwagner, and Konstantin Schekotihin. Stream reasoning-based control of caching strategies in CCN routers. In *IEEE International Conference on Communications*, 2017. ISBN 9781467389990. doi: 10.1109/ICC.2017.7996762.

[15] Michael Till Beck, Martin Werner, Sebastian Feld, and Thomas Schimper. Mobile Edge Computing : A Taxonomy. In *6th International Conference on Advances in Future Internet, (AFIN)*, 2014. ISBN 9781612083773. doi: 10.1.1.670.9418.

[16] Nathan Beckmann and Daniel Sanchez. Modeling cache performance beyond LRU. In *Proceedings - International Symposium on High-Performance Computer Architecture*, 2016. ISBN 9781467392112. doi: 10.1109/HPCA.2016.7446067.

[17] Cesar Bernardini, Thomas Silverston, and Olivier Festor. A comparison of caching strategies for content centric networking. In *2015 IEEE Global Communications Conference, GLOBECOM 2015*, 2015. ISBN 9781479959525. doi: 10.1109/GLOCOM.2014.7417007.

[18] Léon Bottou. Stochastic gradient descent tricks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012. ISSN 03029743. doi: 10.1007/978-3-642-35289-8-25.

[19] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks (extended version). *Computer Communications*, 2013. ISSN 01403664. doi: 10.1016/j.comcom.2013.01.007.

[20] Jiayin Chen, Wenchao Xu, Nan Cheng, Huaqing Wu, Shan Zhang, and Xuemin Sherman Shen. Reinforcement Learning Policy for Adaptive Edge Caching in Heterogeneous Vehicular Network. In *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018. ISBN 9781538647271. doi: 10.1109/GLOCOM.2018.8647483.

[21] Zheng Chen, Nikolaos Pappas, and Marios Kountouris. Probabilistic caching in wireless D2D networks: Cache hit optimal versus throughput optimal. *IEEE Communications Letters*, 2017. ISSN 10897798. doi: 10.1109/LCOMM.2016.2628032.

[22] Kideok Cho, Munyoung Lee, Kunwoo Park, Ted Taekyoung Kwon, Yanghee Choi, and Sangheon Pack. WAVE: Popularity-based and collaborative in-network caching for content-oriented networks. In *Proceedings - IEEE INFOCOM*, pages 316–321, 2012. ISBN 9781467310178. doi: 10.1109/INFCOMW.2012.6193512.

[23] Ali Dabirmoghaddam, Maziar Mirzazad-Barijough, and J. J. Garcia-Luna-Aceves. Understanding optimal caching and opportunistic caching at "the edge" of information-

centric networks. In *ICN 2014 - Proceedings of the 1st International Conference on Information-Centric Networking*, 2014. ISBN 9781450332064.

[24] Brian D Davison. A Web Caching Primer. Technical Report 4. URL `http://www.cs.rutgers.edu/~davison/`.

[25] Matha Deghel, Ejder Bastug, Mohamad Assaad, and Merouane Debbah. On the benefits of edge caching for MIMO interference alignment. In *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, 2015. ISBN 9781479919307. doi: 10.1109/SPAWC.2015.7227119.

[26] Hassan Diab, Ali Kashani, and Ahmad Nasri. Cache replacement engine: A fuzzy logic approach. In *Proceedings of the 2009 International Conference on the Current Trends in Information Technology, CTIT 2009*, 2009. ISBN 9781424457557. doi: 10.1109/CTIT.2009.5423141.

[27] Qinghua Ding, Haitian Pang, and Lifeng Sun. SAM: Cache space allocation in collaborative edge-caching network. In *IEEE International Conference on Communications*, 2017. ISBN 9781467389990. doi: 10.1109/ICC.2017.7996701.

[28] Le Phong Du, Tuan Anh Le, Nguyen Duc Thai, and Phuong Luu Vo. The performance of caching strategies in content centric networking. In *International Conference on Information Networking*, 2017. ISBN 9781509051243. doi: 10.1109/ICOIN.2017.7899573.

[29] Jeffrey Erman, Alexandre Gerber, Mohammad T. Hajiaghayi, Dan Pei, and Oliver Spatscheck. Network-aware forward caching. In *WWW'09 - Proceedings of the 18th International World Wide Web Conference*, 2009. ISBN 9781605584874. doi: 10.1145/1526709.1526749.

[30] Andre Luiz Pereira De Franca, Ricardo Pereira Jasinski, Volnei Antonio Pedroni, and Altair Olivo Santin. Moving network protection from software to hardware: An energy efficiency analysis. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, 2014. ISBN 9781479937639. doi: 10.1109/ISVLSI.2014.89.

[31] Christine Fricker, Philippe Robert, James Roberts, and Nada Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *Proceedings - IEEE INFOCOM*, 2012. ISBN 9781467310178. doi: 10.1109/INFCOMW.2012.6193511.

[32] Xavier Gabaix. Zipf's law for cities: An explanation. *Quarterly Journal of Economics*, 1999. ISSN 00335533. doi: 10.1162/003355399556133.

[33] Frank Hasslinger Oliver Hohlfeld Gerhard Hasslinger, Konstantinos Ntougias. *Fast and Efficient Web Caching Methods Regarding the Size and Performance Measures per Data Object*. 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) : proceedings : 11-13 September 2019, Limassol, Cyprus., Limassol, Cyprus, Cyprus, 2019. ISBN 9781728110165.

[34] Abdelwahab Hamam and Nicolas D. Georganas. A comparison of mamdani and sugeno fuzzy inference systems for evaluating the quality of experience of hapto-audio-visual applications. In *HAVE 2008 - IEEE International Workshop on Haptic Audio Visual Environments and Games Proceedings*, 2008. ISBN 9781424426690. doi: 10.1109/HAVE.2008.4685304.

[35] Suhaidi Hassan, Adib Habbal, Raaid Alubady, and Mays Salman. A Taxonomy of Information-Centric Networking Architectures based on Data Routing and Name Resolution Approaches. ISSN 2289-8131.

[36] Oliver Heckmann, Michael Piringer, Jens Schmitt, and Ralf Steinmetz. Generating realistic ISP-level network topologies. *IEEE Communications Letters*, 2003. ISSN 10897798. doi: 10.1109/LCOMM.2003.814708.

[37] Yun Chao Hu, Milan Patel, Dario Sabella, and Valerie Young. ETSI White Paper No. 11 : Mobile Edge Computing - a key technology towards 5G. Technical report, 2015. URL `www.etsi.org`.

[38] Ion Iancu. A Mamdani Type Fuzzy Logic Controller. In *Fuzzy Logic - Controls, Concepts, Theories and Applications*. 2012. doi: 10.5772/36321.

[39] Chedia Jarray and Anastasios Giovanidis. The effects of mobility on the hit performance of cached D2D networks. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2016*, 2016. ISBN 9781509013111. doi: 10.1109/WIOPT.2016.7492958.

[40] Le Jiang and Xinglin Zhang. Cache Replacement Strategy with Limited Service Capacity in Heterogeneous Networks. *IEEE Access*, 8, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2970783.

[41] Wei Jiang, Gang Feng, Shuang Qin, and Yijing Liu. Multi-Agent Reinforcement Learning Based Cooperative Content Caching for Mobile Edge Networks. *IEEE Access*, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2916314.

[42] Wei Jiang, Gang Feng, Shuang Qin, Tak Shing Peter Yum, and Guohong Cao. Multi-Agent Reinforcement Learning for Efficient Content Caching in Mobile D2D Networks. *IEEE Transactions on Wireless Communications*, 2019. ISSN 15582248. doi: 10.1109/TWC.2019.2894403.

[43] Anwar Kalghoum, Sonia Mettali Gammar, and Leila Azouz Saidane. Towards a novel cache replacement strategy for Named Data Networking based on Software Defined Networking. *Computers and Electrical Engineering*, 66, 2018. ISSN 00457906. doi: 10.1016/j.compeleceng.2017.12.025.

[44] Sedigheh Khajoueinejad, Mojtaba Sabeghi, and Azam Sadeghzadeh. A fuzzy cache replacement policy and its experimental performance assessment. In *2006 Innovations in Information Technology, IIT*, 2006. ISBN 1424406749. doi: 10.1109/INNOVATIONS.2006.301964.

[45] Atta Ur Rehman Khan, Mazliza Othman, Sajjad Ahmad Madani, and Samee Ullah Khan. A survey of mobile cloud computing application models. *IEEE Communications Surveys and Tutorials*, 16(1):393–413, 3 2014. ISSN 1553877X. doi: 10.1109/SURV.2013.062613.00160.

[46] Faria Khandaker, Sharief Oteafy, Hossam S. Hassanein, and Hesham Farahat. A functional taxonomy of caching schemes: Towards guided designs in information-

centric networks. *Computer Networks*, 2019. ISSN 13891286. doi: 10.1016/j.comnet.2019.106937.

[47] Ian Kuon, Russell Tessier, and Jonathan Rose. FPGA architecture: Survey and challenges, 2007. ISSN 15513939.

[48] Fandi Setio Kurniawan, Leanna Vidya Yovita, and Tody Ariefianto Wibowo. Modified-LRU Algorithm for Caching on Named Data Network. In *Proceedings of the International Conference on Electrical Engineering and Informatics*, volume 2019-July, 2019. doi: 10.1109/ICEEI47359.2019.8988836.

[49] Nidhi Lal, Shishupal Kumar, and Vijay Kumar Chaurasiya. An adaptive neuro-fuzzy inference system-based caching scheme for content-centric networking. *Soft Computing*, 23(12):4459–4470, 6 2019. ISSN 14337479. doi: 10.1007/s00500-018-3105-1.

[50] Taofik Lamsub and Pichaya Tandayya. A Dynamic Popularity Caching Policy for Dynamic Adaptive Streaming over HTTP. pages 322–327, Ho Chi Minh City, Vietnam, 2019. IEEE. ISBN 9781728150093. doi: 10.1109/ISCIT.2019.8905163.

[51] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The LCD interconnection of LRU caches and its analysis. *Performance Evaluation*, 2006. ISSN 01665316. doi: 10.1016/j.peva.2005.05.003.

[52] Mathieu Leconte, Georgios Paschos, Lazaros Gkatzikis, Moez Draief, Spyridon Vassilaras, and Symeon Chouvardas. Placing dynamic content in caches with small population. In *Proceedings - IEEE INFOCOM*, 2016. ISBN 9781467399531. doi: 10.1109/INFOCOM.2016.7524380.

[53] Yingqi Li, Meiju Yu, and Ru Li. A Cache Replacement Strategy Based on Hierarchical Popularity in NDN. In *International Conference on Ubiquitous and Future Networks, ICUFN*, volume 2018-July, 2018. doi: 10.1109/ICUFN.2018.8436597.

[54] Ying Lu, Tarek F. Abdelzaher, and Avneesh Saxena. Design, implementation, and evaluation of differentiated caching services. *IEEE Transactions on Parallel and Distributed Systems*, 2004. ISSN 10459219. doi: 10.1109/TPDS.2004.1278101.

[55] Zhaohui Luo, Minghui LiWang, Zhijian Lin, Lianfen Huang, Xiaojiang Du, and Mohsen Guizani. Energy-Efficient Caching for Mobile Edge Computing in 5G Networks. *Applied Sciences*, 2017. doi: 10.3390/app7060557.

[56] Areej M. Osman and Niemah I. Osman. A Comparison of Cache Replacement Algorithms for Video Services. *International Journal of Computer Science and Information Technology*, 10(2):95–111, 5 2018. ISSN 09754660. doi: 10.5121/ijcsit.2018.10208.

[57] A. Mahmood, C. Casetti, C.F. Chiasserini, P. Giaccone, and J. Härri. Mobility-aware edge caching for connected cars. In *2016 12th Annual Conference on Wireless On-Demand Network Systems and Services, WONS 2016 - Conference Proceedings*, 2016. ISBN 9783901882791.

[58] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 1975. ISSN 00207373. doi: 10.1016/S0020-7373(75)80002-2.

[59] Stephan Mandt, Matthew D. Hof Fman, and David M. Blei. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 2017. ISSN 15337928.

[60] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering, 2004. ISSN 1615147X.

[61] Sarra Mehamel, Khaled Slimani, Samia Bouzefrane, and Mehammed Daoui. Energy-efficient hardware caching decision using fuzzy logic in mobile edge computing. In *Proceedings - 2018 IEEE 6th International Conference on Future Internet of Things and Cloud Workshops, W-FiCloud 2018*, pages 237–242. Institute of Electrical and Electronics Engineers Inc., 10 2018. ISBN 9781538678107. doi: 10.1109/W-FiCloud.2018.00045.

[62] John Mitchell, Syed Rizvi, and Jungwoo Ryoo. A Fuzzy-Logic Approach for Evaluating a Cloud Service Provider. In *Proceedings - 2015 1st International Conference on*

Software Security and Assurance, ICSSA 2015, 2017. ISBN 9781509010783. doi: 10.1109/ICSSA.2015.014.

[63] Yunming Mo, Jinxing Bao, Shaobing Wang, Yaxiong Ma, Han Liang, Jiabao Huang, Ping Lu, and Jincai Chen. CCPNC: A Cooperative Caching Strategy Based on Content Popularity and Node Centrality. In *2019 IEEE International Conference on Networking, Architecture and Storage, NAS 2019 - Proceedings*, 2019. ISBN 9781728144092. doi: 10.1109/NAS.2019.8834733.

[64] Quang Ngoc Nguyen, Jiang Liu, Zhenni Pan, Ilias Benkacem, Toshitaka Tsuda, Tarik Taleb, Shigeru Shimamoto, and Takuro Sato. PPCS: A progressive popularity-aware caching scheme for edge-based cache redundancy avoidance in information-centric networks. *Sensors (Switzerland)*, 2019. ISSN 14248220. doi: 10.3390/s19030694.

[65] John P. O'Doherty, Jeffrey Cockburn, and Wolfgang M. Pauli. Learning, Reward, and Decision Making. *Annual Review of Psychology*, 2017. ISSN 0066-4308. doi: 10.1146/annurev-psych-010416-044216.

[66] Davi Nunes Oliveira, Gustavo Alves De Lima Henn, and Otacílio Da Mota Almeida. Design and implementation of a Mamdani fuzzy inference system on an FPGA using VHDL. In *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS*, 2010. ISBN 9781424478576. doi: 10.1109/NAFIPS.2010.5548190.

[67] Svetlana Ostrovskaya, Oleg Surnin, Rasheed Hussain, Safdar Hussain Bouk, Jooyoung Lee, Narges Mehran, Syed Hassan Ahmed, and Abderrahim Benslimane. Towards Multi-metric Cache Replacement Policies in Vehicular Named Data Networks. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, volume 2018-September, 2018. doi: 10.1109/PIMRC.2018.8580741.

[68] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of Web cache replacement strategies. *ACM Computing Surveys*, 35(4):374–398, 2 2004. ISSN 03600300. doi: 10.1145/954339.954341.

[69] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. 2012. doi: 10.1145/2342488.2342501.

[70] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. 2012. doi: 10.1145/2342488.2342501.

[71] Jing Ren, Wen Qi, Cedric Westphal, Jianping Wang, Kejie Lu, Shucheng Liu, and Sheng Wang. MAGIC: A distributed MAx-Gain In-network Caching strategy in information-centric networks. In *Proceedings - IEEE INFOCOM*, 2014. ISBN 9781479930883. doi: 10.1109/INFCOMW.2014.6849277.

[72] Dario Rossi and Giuseppe Rossini. Caching performance of content centric networks under multi-path routing (and more). *Relatório técnico, Telecom ParisTech*, 2011.

[73] Giuseppe Rossini and Dario Rossi. A dive into the caching performance of content centric networking. In *2012 IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD 2012*, 2012. ISBN 9781467331258. doi: 10.1109/CAMAD.2012.6335307.

[74] Giuseppe Rossini and Dario Rossi. Coupling caching and forwarding: Benefits, analysis, and implementation. In *ICN 2014 - Proceedings of the 1st International Conference on Information-Centric Networking*, 2014. ISBN 9781450332064.

[75] Alireza Sadeghi, Fatemeh Sheikholeslami, and Georgios B. Giannakis. Optimal Dynamic Proactive Caching Via Reinforcement Learning. In *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, 2018. ISBN 9781538635124. doi: 10.1109/SPAWC.2018.8445899.

[76] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Hash-routing schemes for information centric networking. In *ICN 2013 - Proceedings of the 3rd, 2013 ACM SIGCOMM Workshop on Information-Centric Networking*, 2013. ISBN 9781450321792. doi: 10.1145/2491224.2491232.

[77] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Icarus: A Caching simulator for Information Centric Networking (ICN). In *SIMUTools 2014 - 7th International Conference on Simulation Tools and Techniques*, 2014. ISBN 9781631900075. doi: 10.4108/icst.simutools.2014.254630.

[78] Hiba Ali Nasir Sirour, Yahia Abdalla M. Hamad, and Amir A.A. Eisa. An agent-based proxy cache cleanup model using fuzzy logic. In *Proceedings - 2013 International Conference on Computer, Electrical and Electronics Engineering: 'Research Makes a Difference', ICCEEE 2013*, 2013. ISBN 9781467362313. doi: 10.1109/ICCEEE.2013.6633987.

[79] Jyrki Suomala and Ville Suomala. Modified Reinforcement Learning Infrastructure. Atlantis Press, 9 2014. doi: 10.2991/icassr-14.2014.27.

[80] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction Second edition, in progress. Technical report, Cambridge, MA, USA: MITPress, 2016.

[81] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000. ISBN 0262194503.

[82] Ravi Tandon and Osvaldo Simeone. Cloud-aided wireless networks with edge caching: Fundamental latency trade-offs in fog Radio Access Networks. In *IEEE International Symposium on Information Theory - Proceedings*, 2016. ISBN 9781509018062. doi: 10.1109/ISIT.2016.7541655.

[83] Tuyen X. Tran and Dario Pompili. Octopus: A Cooperative Hierarchical Caching Strategy for Cloud Radio Access Networks. In *Proceedings - 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2016*, 2017. ISBN 9781509028337. doi: 10.1109/MASS.2016.029.

[84] Tuyen X. Tran, Abolfazl Hajisami, and Dario Pompili. Cooperative Hierarchical Caching in 5G Cloud Radio Access Networks. *IEEE Network*, 2017. ISSN 08908044. doi: 10.1109/MNET.2017.1600307.

[85] Tuyen X. Tran, Duc V. Le, Guosen Yue, and Dario Pompili. Cooperative Hierarchical Caching and Request Scheduling in a Cloud Radio Access Network. *IEEE Transactions on Mobile Computing*, 2018. ISSN 15580660. doi: 10.1109/TMC.2018.2818723.

[86] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 2009. ISSN 13891286. doi: 10.1016/j.comnet.2009.02.019.

[87] Andrea Vattani, Flavio Chierichetti, and Keegan Lowenstein. Optimal probabilistic cache stampede prevention. In *Proceedings of the VLDB Endowment*, 2015. doi: 10.14778/2757807.2757813.

[88] James Z. Wang, Zhidian Du, and Pradip K. Srimani. Cooperative proxy caching for wireless base stations. *Mobile Information Systems*, 2007. ISSN 1875905X. doi: 10.1155/2007/371572.

[89] Jason Min Wang and Brahim Bensaou. Improving content-centric networks performance with progressive, diversity-load driven caching. In *2012 1st IEEE International Conference on Communications in China, ICCC 2012*, 2012. ISBN 9781467328159. doi: 10.1109/ICCChina.2012.6356996.

[90] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-AS cooperative caching for content-centric networks. 2013. doi: 10.1145/2491224.2491234.

[91] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-AS cooperative caching for content-centric networks. In *ICN 2013 - Proceedings of the 3rd, 2013 ACM SIGCOMM Workshop on Information-Centric Networking*, 2013. ISBN 9781450321792. doi: 10.1145/2491224.2491234.

[92] Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, and Wenbo Wang. A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications. *IEEE Access*, 5:6757–6779, 2017. ISSN 21693536. doi: 10.1109/ACCESS.2017.2685434.

[93] Hsien Chung Wu. The Karush-Kuhn-Tucker optimality conditions in an optimization problem with interval-valued objective function. *European Journal of Operational Research*, 2007. ISSN 03772217. doi: 10.1016/j.ejor.2005.09.007.

[94] G. Nan X. Jiang, J. Bi and Z. Li. A survey on Information-centric Networking: Rationales, designs and debates. *China Communications*, 12(7):1–12, 2015.

[95] Xianzhe Xu and Meixia Tao. Collaborative Multi-Agent Reinforcement Learning of Caching Optimization in Small-Cell Networks. In *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018. ISBN 9781538647271. doi: 10.1109/GLOCOM.2018.8647341.

[96] Muhamad Rizky Yanuar and Afwarman Manaf. Performance evaluation of progressive caching policy on NDN. In *Proceedings - 2017 International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2017*, 2017. ISBN 9781538630013. doi: 10.1109/ICAICTA.2017.8090996.

[97] Lin Yao, Ailun Chen, Jing Deng, Jianbang Wang, and Guowei Wu. A Cooperative Caching Scheme Based on Mobility Prediction in Vehicular Content Centric Networks. *IEEE Transactions on Vehicular Technology*, 2018. ISSN 00189545. doi: 10.1109/TVT.2017.2784562.

[98] Ya'nan Ni Jia Shi Yuanjun He, Yi Zhu and Na Zhu. A Cache Strategy in Content-centric Networks Based on Nodes Importance. *Information Technology Journal*, 13: 588–592, 2014. doi: 10.3923/itj.2014.588.592.

[99] Feixiong Zhang, Chenren Xu, Yanyong Zhang, K. K. Ramakrishnan, Shreyasee Mukherjee, Roy Yates, and Nguyen Thu. EdgeBuffer: Caching and prefetching content at the edge in the MobilityFirst future Internet architecture. In *Proceedings of the WoWMoM 2015: A World of Wireless Mobile and Multimedia Networks*, 2015. ISBN 9781479984619. doi: 10.1109/WoWMoM.2015.7158137.

[100] Meng Zhang, Hongbin Luo, and Hongke Zhang. A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys and Tutorials*, 2015. ISSN 1553877X. doi: 10.1109/COMST.2015.2420097.

[101] Shan Zhang, Peter He, Katsuya Suto, Peng Yang, Lian Zhao, and Xuemin Shen. Cooperative Edge Caching in User-Centric Clustered Mobile Networks. *IEEE Transactions on Mobile Computing*, 2018. ISSN 15361233. doi: 10.1109/TMC.2017.2780834.

[102] Weiwen Zhang, Yonggang Wen, Kyle Guan, Dan Kilper, Haiyun Luo, and Dapeng Oliver Wu. Energy-optimal mobile cloud computing under stochastic wireless

channel. *IEEE Transactions on Wireless Communications*, 2013. ISSN 15361276. doi: 10.1109/TWC.2013.072513.121842.

[103] Yan Zhang, Xu Zhou, Yinlong Liu, Bo Wang, and Song Ci. A novel cooperative caching algorithm for massive P2P caches. *Peer-to-Peer Networking and Applications*, 2013. ISSN 19366442. doi: 10.1007/s12083-013-0211-9.

[104] Hao Zhu, Yang Cao, Wei Wang, Tao Jiang, and Shi Jin. Deep Reinforcement Learning for Mobile Edge Caching: Review, New Features, and Open Issues. *IEEE Network*, 2018. doi: 10.1109/MNET.2018.1800109.

[105] Qingbo Zhu and Yuanyuan Zhou. Power-aware storage cache management. *IEEE Transactions on Computers*, 2005. ISSN 00189340. doi: 10.1109/TC.2005.82.

Appendix

Fuzzy rules for caching decision:

```
export const priorities = [
    [ PRIO.MEDIUM, [ 'low', 'short', 'small', 'soClose' ] ],
    [ PRIO.MEDIUM, [ 'low', 'short', 'small', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'short', 'small', 'far' ] ],
    [ PRIO.MEDIUM, [ 'low', 'short', 'medium', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'short', 'medium', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'short', 'medium', 'far' ] ],
    [ PRIO.LOW, [ 'low', 'short', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'short', 'large', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'short', 'large', 'far' ] ],
    [ PRIO.MEDIUM, [ 'low', 'medium', 'small', 'soClose' ] ],
    [ PRIO.MEDIUM, [ 'low', 'medium', 'small', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'medium', 'small', 'far' ] ],
    [ PRIO.MEDIUM, [ 'low', 'medium', 'medium', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'medium', 'medium', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'medium', 'medium', 'far' ] ],
    [ PRIO.LOW, [ 'low', 'medium', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'medium', 'large', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'medium', 'large', 'far' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'small', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'small', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'small', 'far' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'medium', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'medium', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'medium', 'far' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'large', 'close' ] ],
    [ PRIO.LOW, [ 'low', 'long', 'large', 'far' ] ],
    [ PRIO.HIGH, [ 'medium', 'short', 'small', 'soClose' ] ],
    [ PRIO.HIGH, [ 'medium', 'short', 'small', 'close' ] ],
    [ PRIO.LOW, [ 'medium', 'short', 'small', 'far' ] ],
    [ PRIO.HIGH, [ 'medium', 'short', 'medium', 'soClose' ] ],
    [ PRIO.HIGH, [ 'medium', 'short', 'medium', 'close' ] ],
    [ PRIO.MEDIUM, [ 'medium', 'short', 'medium', 'far' ] ],
    [ PRIO.MEDIUM, [ 'medium', 'short', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'medium', 'short', 'large', 'close' ] ],
    [ PRIO.HIGH, [ 'medium', 'short', 'large', 'far' ] ],
    [ PRIO.HIGH, [ 'medium', 'medium', 'small', 'soClose' ] ],
    [ PRIO.LOW, [ 'medium', 'medium', 'small', 'close' ] ],
    [ PRIO.MEDIUM, [ 'medium', 'medium', 'small', 'far' ] ],
    [ PRIO.MEDIUM, [ 'medium', 'medium', 'medium', 'soClose'] ],
    [ PRIO.LOW, [ 'medium', 'medium', 'medium', 'close' ] ],
    [ PRIO.MEDIUM, [ 'medium', 'medium', 'medium', 'far' ] ],
    [ PRIO.MEDIUM, [ 'medium', 'medium', 'large', 'soClose' ] ],
```

```
    [ PRIO.LOW, [ 'medium', 'medium', 'large', 'close' ] ],
    [ PRIO.LOW, [ 'medium', 'medium', 'large', 'far' ] ],
    [ PRIO.LOW, [ 'medium', 'long', 'small', 'soClose' ] ],
    [ PRIO.LOW, [ 'medium', 'long', 'small', 'close' ] ],
    [ PRIO.HIGH, [ 'medium', 'long', 'small', 'far' ] ],
    [ PRIO.HIGH, [ 'medium', 'long', 'medium', 'soClose' ] ],
    [ PRIO.LOW, [ 'medium', 'long', 'medium', 'close' ] ],
    [ PRIO.HIGH, [ 'medium', 'long', 'medium', 'far' ] ],
    [ PRIO.HIGH, [ 'medium', 'long', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'medium', 'long', 'large', 'close' ] ],
    [ PRIO.HIGH, [ 'medium', 'long', 'large', 'far' ] ],
    [ PRIO.MEDIUM, [ 'high', 'short', 'small', 'soClose' ] ],
    [ PRIO.LOW, [ 'high', 'short', 'small', 'close' ] ],
    [ PRIO.HIGH, [ 'high', 'short', 'small', 'far' ] ],
    [ PRIO.HIGH, [ 'high', 'short', 'medium', 'soClose' ] ],
    [ PRIO.LOW, [ 'high', 'short', 'medium', 'close' ] ],
    [ PRIO.HIGH, [ 'high', 'short', 'medium', 'far' ] ],
    [ PRIO.HIGH, [ 'high', 'short', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'high', 'short', 'large', 'close' ] ],
    [ PRIO.HIGH, [ 'high', 'short', 'large', 'far' ] ],
    [ PRIO.MEDIUM, [ 'high', 'medium', 'small', 'soClose' ] ],
    [ PRIO.LOW, [ 'high', 'medium', 'small', 'close' ] ],
    [ PRIO.MEDIUM, [ 'high', 'medium', 'small', 'far' ] ],
    [ PRIO.MEDIUM, [ 'high', 'medium', 'medium', 'soClose' ] ],
    [ PRIO.LOW, [ 'high', 'medium', 'medium', 'close' ] ],
    [ PRIO.MEDIUM, [ 'high', 'medium', 'medium', 'far' ] ],
    [ PRIO.MEDIUM, [ 'high', 'medium', 'large', 'soClose' ] ],
    [ PRIO.LOW, [ 'high', 'medium', 'large', 'close' ] ],
    [ PRIO.MEDIUM, [ 'high', 'medium', 'large', 'far' ] ],
    [ PRIO.MEDIUM, [ 'high', 'long', 'small', 'soClose' ] ],
    [ PRIO.MEDIUM, [ 'high', 'long', 'small', 'close' ] ],
    [ PRIO.LOW, [ 'high', 'long', 'small', 'far' ] ],
    [ PRIO.HIGH, [ 'high', 'long', 'medium', 'soClose' ] ],
    [ PRIO.HIGH, [ 'high', 'long', 'medium', 'close' ] ],
    [ PRIO.LOW, [ 'high', 'long', 'medium', 'far' ] ],
    [ PRIO.HIGH, [ 'high', 'long', 'large', 'soClose' ] ],
    [ PRIO.MEDIUM, [ 'high', 'long', 'large', 'close' ] ],
    [ PRIO.LOW, [ 'high', 'long', 'large', 'far' ] ],
];
```

le cnam

Sarra Mehamel

New Intelligent Caching and mMbility Strategies for MEC /ICN based architectures

le cnam

Abstract :
Mobile edge computing (MEC) concept proposes to bring the computing and storage resources in close proximity to the end user by placing these resources at the network edge. The motivation is to alleviate the mobile core and to reduce latency for mobile users due to their close proximity to the edge. MEC servers are candidates to host mobile applications and serve web contents. Edge caching is one of the most emerging technologies recognized as a content retrieval solution in the edge of the network. It has been also considered as enabling technology of mobile edge computing that presents an interesting opportunity to perform caching services. Particularly, the MEC servers are implemented directly at the base stations which enable edge caching and ensure deployment in close-proximity to the mobile users. However, the integration of servers in mobile edge computing environment (base stations) complicates the energy saving issue because the power consumed by mobile edge computing servers is costly especially when the load changes dynamically over time. Furthermore, users with mobile devices arise their demands, introducing the challenge of handling such mobile content requests beside the limited caching size. Thus, it is necessary and crucial for caching mechanisms to consider context-aware factors, meanwhile most existing studies focus on cache allocation, content popularity and cache design. In this thesis, we present a novel energy-efficient fuzzy caching strategy for edge devices that takes into consideration four influencing features of mobile environment, while introducing a hardware implementation using Field-Programmable Gate Array (FPGA) to cut the overall energy requirements. Performing an adequate caching strategy on MEC servers opens the possibility of employing artificial intelligence (AI) techniques and machine learning at mobile network edges. Exploiting users context information intelligently makes it possible to design an intelligent context-aware mobile edge caching. Context awareness enables the cache to be aware of its environment, while intelligence enables each cache to make the right decisions of selecting appropriate contents to be cached so that to maximize the caching performance. Inspired by the success of reinforcement learning (RL) that uses agents to deal with decision making problems, we extended our fuzzy-caching system into a modified reinforcement learning model. The proposed framework aims to maximize the cache hit rate and requires a multi awareness. The modified RL differs from other RL algorithms in the learning rate that uses the method of stochastic gradient decent beside taking advantage of learning using the optimal caching decision obtained from fuzzy rules.

Keywords :
Information Centric Network, Caching, Mobile Edge Computing, Reinforcement Learning, Fuzzy Logic, FPGA.

Appendix

Résumé :

Le paradigme de MEC (Mobile Edge Computing) consiste à mettre les ressources de calcul et de stockage aux « extrémités » du réseau à proximité des utilisateurs finaux. Le terme « edge » désigne n'importe quel type de station de base de réseau. Les motivations pour l'adoption de ce nouveau concept sont principalement la réduction de la charge au cœur du réseau et la diminution de la latence grâce à la proximité des ressources et ainsi améliorer l'expérience utilisateur. Les serveurs MEC sont de bons candidats pour héberger les applications mobiles et diffuser le contenu Web. La mise en cache à l'extrémité du réseau, ou Edge Caching en anglais, est l'une des technologies les plus émergentes connues comme solution de récupération de contenu au bord du réseau. Elle est aussi considérée comme une technologie permettant la mise en place du concept MEC puisqu'elle présente une opportunité intéressante pour implémenter les services de mise en cache. En particulier, les serveurs MEC sont implémentés directement au niveau des stations de base, ce qui permet la mise en cache à l'extrémité du réseau et assure un déploiement à proximité des utilisateurs finaux. Cependant, l'intégration des serveurs MEC dans les stations de base complexifie le problème de la consommation de l'énergie, particulièrement dans un tel environnement qui est dynamique et sujet à des changements au fil du temps. Par ailleurs, la demande des utilisateurs des appareils mobiles est en constante augmentation ainsi que leur expectation d'une expérience meilleure. Sachant que le cache est d'une taille limitée, il est donc nécessaire et crucial que les mécanismes de mise en cache soient en mesure de faire face à cette situation et de proposer des solutions valables et satisfaisants à long terme. La plupart des études existantes se sont focalisées sur l'allocation de cache, la popularité du contenu ou encore la manière de concevoir le cache. Dans cette thèse, nous présentons une nouvelle stratégie de mise en cache écoénergétique basée sur la logique floue (Fuzzy logic). Notre proposition prend en compte les quatre caractéristiques d'un environnement mobile et introduit une implémentation matérielle en utilisant les FPGA (Field-Programmable Gate Array) pour réduire les besoins globaux en énergie. L'adoption d'une stratégie de mise en cache adéquate sur les serveurs MEC ouvre la possibilité d'utiliser des techniques d'intelligence artificielle (IA) et d'apprentissage automatique (Machine Learning) aux extrémités des réseaux mobiles. L'exploitation des informations de contexte des utilisateurs permet de concevoir une mise en cache intelligente sensible au contexte. La reconnaissance du contexte permet au cache de connaître son environnement, tandis que l'intelligence lui permet de prendre les bonnes décisions en sélectionnant le contenu approprié à mettre en cache afin d'optimiser les performances du caching.

Inspiré par le succès de l'apprentissage par renforcement utilisant des agents pour traiter des problèmes de prise de décision, nous avons étendu notre système de mise en cache basé sur la logique floue à un modèle d'apprentissage par renforcement modifié. Le cadre proposé vise à maximiser le taux de réussite du cache (hit rate) et nécessite une prise de conscience multiple sure les conditions de web et l'utilisateur final. La méthode d'apprentissage par renforcement modifiée diffère des autres algorithmes par le taux d'apprentissage qui utilise la méthode du gradient stochastique décent (stochastic gradient decent) en plus de tirer parti de l'apprentissage en utilisant la décision de mise en cache optimale obtenue à partir des règles de la logique floue.

Mots clés :

Mise en cache, informatique mobile de proximité, réseaux de contenus, logique floue, FPGA, apprentissage par renforcement.