Ανοικτό Πανεπιστήμιο Κύπρου
Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή

Μελέτη ρυθμαπόδοσης MEC

Νικόλαος Ζουγλής

Επιβλέπων Καθηγητής
Αναστάσιος Νταγιούκλας

Νεόμβριος 2019

Ανοικτό Πανεπιστήμιο Κύπρου
Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Πληροφοριακά και Επικοινωνιακά Συστήματα*

Μεταπτυχιακή Διατριβή

Μελέτη ρυθμαπόδοσης MEC

Νικόλαος Ζουγλής

Επιβλέπων Καθηγητής
Αναστάσιος Νταγιούκλας

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών στα Πληροφοριακά και Επικοινωνιακά Συστήματα από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών του Ανοικτού Πανεπιστημίου Κύπρου.

Νοέμβριος 2019

Περίληψη

Η παρούσα διατριβή αφορά την σχεδίαση και την μελέτη μιας εφαρμογής επιτήρησης (surveillance) για την αρχιτεκτονική **Multi-Access Edge Computing (MEC)**. Οι τεχνικές για το **caching** στη **MEC** είναι οι κάτωθι:

- **Predictive Caching (Προβλεπτικό Caching)**
- **Co-operative Caching (Συνεργάσιμο Caching)**

Ο αντικειμενικός σκοπός της παρούσας μεταπτυχιακής διατριβής είναι η δημιουργία και χρήση ενός σεναρίου Co-operative Caching σε μια εφαρμογή surveillence.

Σύντομη περιγραφή

Η **MEC** αρχιτεκτονική είναι μια αναδυόμενη τεχνολογία που αποσκοπεί στην παροχή αποκεντρωμένων υπηρεσιών δικτύωσης, εφαρμογών και αποθήκευσης είτε σε τελικούς χρήστες είτε σε μεγάλες επιχειρήσεις. Η αρχιτεκτονική αυτή αποσκοπεί στην μείωση της καθυστέρησης και στην παροχή επιπλέον υπολογιστικών πόρων στις συσκευές που βρίσκονται στην εγγύτητα της. Η καθυστέρηση θεωρείται ανασταλτικός παράγοντας στην παροχή υπηρεσιών από το Νέφος γιατί εξαιτίας αυτής οι τελικές υπηρεσίες έχουν μειωμένο **QoE**. Ωστόσο, η αρχιτεκτονική **MEC** υφίσταται περιορισμούς σε θέματα υλικού και συγκεκριμένα σε θέματα αποθήκευσης και υπολογιστικών πόρων.

Στην παρούσα διατριβή θα εξομοιωθεί η αρχιτεκτονική **MEC** για εφαρμογή επιτήρησης και εν συνεχεία θα μελετηθεί η ρυθμαπόδοση της σε θέματα υπολογιστικής ισχύος, αποθήκευσης και προσωρινής μνήμης **(Caching)**.

Summary

The present project is about the design and design of a surveillance application for the **Multi-Access Edge Computing** Architecture. The caching techniques being deployed are the below:

- **Predictive Caching**
- **Co-operative Caching**

This purpose of this project will describe a creation and deploying scenario of Co-operative caching technique for a face recognition application.

Abstract

MEC architecture is an emerged technology with the aim of provisioning decentralized services in terms of networking, application and storage either to the end users or to big enterprises. The aim of this architecture is to reduce the latency and the provision extra computing resources to the end devices at its edge. Latency is considered to be an inhibitor for the Cloud services because of the reduced **QoE**. However, **MEC** architecture underwent to restrictions in terms of storage and computer resources.

In this paper a **MEC** architecture is about to be simulated and will be studied in terms of computing power, storage and caching.

Λέξεις κλειδιά: Multi-Access Edge Computing, MEC, Cloud, Cloud Computing, Cache, Fog Nodes, Internet of Things, Cloudlets, Virtual Machines, Containers, Face Recognition, Histogram of Oriented Gradients, Αποκεντρωμένη Υπολογιστική, Νέφος, Υπολογιστική Νέφους, Εικονικές Μηχανές, Αναγνώριση Προσώπου, Ιστόγραμμα Κατευθυνόμενων Διανυσμάτων

Ευχαριστίες

Η παρούσα διατριβή έγινε στα πλαίσια των μεταπτυχιακών σπουδών μου στην Σχολή Θετικών και Εφαρμοσμένων Επιστημών, στο πρόγραμμα **"Πληροφοριακά και Επικοινωνιακά Συστήματα"**. Αν και έχω σπουδάσει Μηχανολόγος Μηχανικός, οι σπουδές μου στα Πληροφορικά Συστήματα ήταν επίμονος στόχος για μένα για τον λόγο ότι η επιστήμη της Πληροφορικής έχει ήδη προσφέρει πολλά για το καλό της ανθρωπότητας και στο άμεσο μέλλον αναμένεται να προσφέρει ακόμα περισσότερα.

Αρχικά, θα ήθελα να ευχαριστήσω το Ανοιχτό Πανεπιστήμιο Κύπρου που μου έδωσε την δυνατότητα να ικανοποιήσω τον στόχο μου. Θα ήθελα να ευχαριστήσω τους καθηγητές που γνώρισα αυτά τα χρόνια και μου έδωσαν τα απαραίτητα ερεθίσματα για να αποκτήσω τις κατάλληλες γνώσεις και συγκεκριμένα τους εξής:

- Γιώργο Χατζημιλιούδη
- Τάσο Νταγιούκλα
- Μιχάλη Γεωργιάδη
- Χρήστο Γκουμόπουλο
- Αικατερίνη Ιωάννου
- Ιωάννη Κατάκη
- Δημοσθένη Βουγιούκα
- Νίκο Νομικό

Θα ήθελα να ευχαριστήσω την οικογένεια μου για την υπομονή και την συμπαράσταση που μου παρείχε μέχρι την ολοκλήρωση του μεταπτυχιακού προγράμματος. Ευχαριστώ πολύ τον Δρ.Βασίλη Τσακανίκα και του εύχομαι καλή σταδιοδρομία.

Τέλος, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου, **Δρ.Τάσο Νταγιούκλα** για την αμέριστη συμπαράσταση και ηθική υποστήριξη που μου παρείχε μέχρι την ολοκλήρωση της παρούσας διατριβής. Θα ήθελα να του ευχηθώ καλή σταδιοδρομία στην ακαδημαϊκή του καριέρα.

Ιόλη μου….

Από τις 25 Ιουλίου 2012 που ήρθες στον κόσμο,

μου δίδαξες να έχω νόημα στην ζωή μου!!

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

1.1 Πρόλογος

Την τελευταία δεκαετία η ζωή των ανθρώπων έχει προσαρμοστεί στην εξέλιξη της σύγχρονής τεχνολογίας. Η δημιουργία και η παραγωγή των έξυπνων κινητών (Smartphones) και εν συνεχεία η εύκολη χρήση αυτών υπήρξε ορόσημο στην καθημερινή ζωή τους. Πλέον, το κινητό τηλέφωνο δεν θεωρείται μόνο ένα μέσον για φωνητική επικοινωνία και ανταλλαγή γραπτών μηνυμάτων. Η δυνατότητα εγκατάστασης διαφόρων εφαρμογών σε αυτές τις συσκευές, από εφαρμογές παραγωγικότητας μέχρι και παιχνίδια, καθιστά παρωχημένη την χρήση του κινητού τηλεφώνου μόνο για επικοινωνία μέσω φωνής. Τα κινητά τηλέφωνα και κατ' επέκταση κάθε κινητή συσκευή έχουν διευρυμένη χρήση και αποτελούν ισχυρά και "έξυπνα" εργαλεία.

Το γεγονός ότι την προηγούμενη δεκαετία μόνο το 20% του παγκόσμιου πληθυσμού κατείχε κινητό τηλέφωνο ενώ, εν έτη 2018, υπάρχουν 5,2 δισεκατομμύρια ενεργοί χρήστες κινητής τηλεφωνίας, οδήγησε στους κατασκευαστές των "έξυπνων" κινητών συσκευών να επενδύσουν σε αυτή την τεχνολογία και να θεσπίσουν νέα πρότυπα. Μεγάλη πληθώρα εφαρμογών κατακλύζουν τις δημοφιλείς πλατφόρμες όπως εργαλεία επικοινωνίας, πλοήγησης, εφαρμογές τραπεζικών συναλλαγών, παιχνίδια κλπ.(Taleb:1, The Mobile Economy)

Η τεχνολογική εξέλιξη των κεντρικών επεξεργαστών, η αύξηση της μνήμης, η βελτίωση των ολοκληρωμένων κυκλωμάτων και η αύξηση της χωρητικότητας των δικτύων δημιουργούν ένα κύμα ανάπτυξης εφαρμογών με δυνατότητες που πριν από μια δεκαετία άνηκαν στην επιστημονική φαντασία. Η επαυξημένη πραγματικότητα, η προβολή video 4K σε πραγματικό χρόνο, ο χειρισμός, με αξιοπιστία, μη επανδρωμένων οχημάτων ή αεροσκαφών (drones, UAVs) αποτελούσαν και αποτελούν προκλήσεις που καλούνταν να αντιμετωπίσουν οι κατασκευαστές και οι προγραμματιστές.

Αν και η φαντασία έχει γίνει πραγματικότητα με την προβολή video 4K σε πραγματικό χρόνο και χειρισμού UAVs από απόσταση, ωστόσο, η εξέλιξη των δικτύων, όσον αφορά το θέμα της ταχύτητας και της χωρητικότητας, δημιουργεί ένα ανάχωμα στην μαζική χρήση όλων των παραπάνω τεχνολογιών σε καθημερινή βάση. Για να κατανοήσουμε το πρόβλημα, θα

χρειαστεί να εισάγουμε τον όρο “καθυστέρηση” (**Latency**). Με άλλα λόγια πως θα ήταν δυνατόν να χειριστούμε ένα UAV εάν υπάρχει μεγάλη καθυστέρηση διάδοσης του σήματος μεταξύ του χειριστή και της απόκρισης του UAV.

Το πρόβλημα της καθυστέρησης δεν υφίσταται μόνο στην περίπτωση ενός UAV. Το πρόβλημα της καθυστέρησης υφίσταται και σε πιο μικρής κλίμακας εφαρμογές και είναι καθολικό. Παράδειγμα εφαρμογής όπου η καθυστέρηση παίζει πρωταγωνιστικό ρόλο είναι η πρόσβαση πολλών χρηστών σε έναν κόμβο του Νέφους. Στην περίπτωση ύπαρξης πολλών χρηστών, όπου οι τελευταίοι κάνουν πολλές αιτήσεις (requests), η καθυστέρηση αυξάνεται με αποτέλεσμα η αξιοπιστία του συστήματος του Νέφους να φθίνει. Η κατάσταση γίνεται ακόμα δραματική εάν η ταχύτητα του δικτύου μειωθεί. Προκειμένου να αντιμετωπιστεί το πρόβλημα της καθυστέρησης, μια νέα τεχνολογία έκανε την εμφάνιση της η οποία ονομάζεται **Multi-Access Edge Computing** ή **MEC.** Η **MEC** τεχνολογία δεν είναι μόνο για να αμβλυνθεί ο πρόβλημα της καθυστέρησης αλλά αποτελεί και βασικό πυλώνα της ανάπτυξης των **5G** δικτύων.

Η λύση που εισάγει η τεχνολογία **MEC** είναι απλή. Υποθέτουμε ότι ο προηγούμενος κόμβος του Νέφους είναι επιφορτισμένος με μια υπηρεσία και δέχεται πολλά requests. Η περίπτωση αυτή ομοιάζει με μια αρχιτεκτονική η οποία αποτελείται από έναν κεντρικό εξυπηρετητή που εξυπηρετεί πολλές αιτήσεις. Υποθέτουμε ότι η υπηρεσία που καλείται να εξυπηρετήσει ο κόμβος του Νέφους είναι απαιτητική σε υπολογιστικούς πόρους. Αμέσως αντιλαμβανόμαστε τις συνέπειες τις οποίες ήδη προαναφέραμε. Η τεχνολογία MEC και ειδικά η τεχνική του **Edge Computing (Αποκεντρωμένη Υπολογιστική)** αναλαμβάνει να αποφορτίσει τις δυσχέρειες που έχει η προηγούμενη αρχιτεκτονική. Σύμφωνα με την τεχνική της Αποκεντρωμένης Υπολογιστικής ένας τοπικός κόμβος αναλαμβάνει να διεκπεραιώσει την υπηρεσία του κεντρικού κόμβου του Νέφους. Κάθε κινητή συσκευή που εισέρχεται εντός της κάλυψης του δικτύου του τοπικού κόμβου αυτή θα δύναται να χρησιμοποιεί τις υπηρεσίες που προσφέρει ο τοπικός κόμβος.

Στην παρούσα διατριβή θα εξομοιώσουμε έναν MEC κόμβο και έναν κόμβο του Νέφους και θα επιφορτιστούν με την υπηρεσία αναγνώρισης προσώπου από μια υποτιθέμενη κάμερα επιτήρησης (camera surveillance) (Shahzadi:1).

1.2 Ορισμοί

Παρακάτω θα αναλυθούν, επιγραμματικά, οι ορισμοί που θα συναντήσουμε στην διατριβή.

1.2.1 Νέφος (Cloud)

Είναι μια αρχιτεκτονική, σύμφωνα με την οποία, σε ένα σύνολο υπολογιστικών μονάδων εγκαθίσταται ένα σύνολο εφαρμογών με σκοπό την παροχή κάποιας υπηρεσίας. Η υπηρεσία μπορεί να είναι η αποθήκευση, η παροχή υπολογιστικών πόρων ή οποιαδήποτε άλλη εφαρμογή με αποδέκτες τους τελικούς χρήστες. Κύριο χαρακτηριστικό της αρχιτεκτονικής του Νέφους είναι ότι η πρόσβαση των χρηστών στις υπηρεσίες του γίνεται αποκλειστικά και μόνο μέσω δικτύου.

1.2.2 Αποκεντρωμένη Υπολογιστική (Edge Computing)

Η Αποκεντρωμένη Υπολογιστική είναι μια αρχιτεκτονική σύμφωνα με την οποία μια υπολογιστική μονάδα, όντας ως τοπικός κόμβος πρόσβασης στους τελικούς χρήστες, αναλαμβάνει να διεκπεραιώσει τις υπηρεσίες του Νέφους. Το μεγάλο πλεονέκτημα αυτής της αρχιτεκτονικής είναι η μικρή καθυστέρηση στην απόκριση των υπηρεσιών. (Shahzadi:6)

1.2.3 Multi-Access Edge Computing

Είναι η επέκταση της αρχιτεκτονικής της Αποκεντρωμένης Υπολογιστικής συμπεριλαμβάνοντας την διασύνδεση συσκευών μέσα από ετερογενή δίκτυα. (Taleb:1)

1.2.4 Cloudlet

Είναι μια αρχιτεκτονική του Νέφους η οποία αποτελείται από τρία επίπεδα. Το πρώτο επίπεδο απαρτίζεται από την κινητή συσκευή, το δεύτερο επίπεδο απαρτίζεται από έναν αποκεντρωμένο κόμβο του Νέφους **(Edge Cloud Node)** και το τρίτο από το Νέφος. (Shahzadi:4, Taleb:3)

1.2.5 Ομιχλώδης Επεξεργασία (Fog Computing)

Είναι μια αρχιτεκτονική αποκεντρωμένης υπολογιστικής του Νέφους η οποία αναπτύχθηκε από την Cisco. Το σκεπτικό της συγκεκριμένης αρχιτεκτονικής είναι η συμπερίληψη των κόμβων που βρίσκονται σε διαφορετικές γεωγραφικές τοποθεσίες. (Taleb:3, Shahzadi:5)

1.2.6 Διαδίκτυο των Πραγμάτων (Internet of Things)

Είναι μια αρχιτεκτονική σύμφωνα με την οποία παρέχεται επικοινωνία μεταξύ των αντικειμένων χωρίς την ανθρώπινη μεσολάβηση. (Shahzadi:5, Taleb:5)

1.2.7 Εικονικές Μηχανές (VMs) και Containers

Είναι αρχιτεκτονικές του Νέφους που χρησιμοποιούνται από τους τελικούς χρήστες για την χρήση της υπηρεσίας του Νέφους. Η αρχιτεκτονική των Εικονικών Μηχανών είναι κατάλληλη για βαριές υπολογιστικές εφαρμογές και όπου τίθεται θέμα ασφαλείας. Η αρχιτεκτονική των **Containers** ενδείκνυται ελαφριές υπολογιστικές εφαρμογές όπου οι τελικοί χρήστες έχουν πρόσβαση με κινητές συσκευές και η κινητικότητα παίζει βασικό ρόλο. (Taleb:6)

Κεφάλαιο 2
Edge Computing

2.1 Εισαγωγή

Οι εξελίξεις στην πληροφορική και στις κινητές τηλεπικοινωνίες αποτελούν κύριο κόμβο στην βελτίωση της ποιότητας ζωής των ανθρώπων. Η σύγκλιση αυτών των δύο συνιστωσών θεωρείται σημείο καμπής. Άνθρωποι σε όλο τον πλανήτη που χρησιμοποιούσαν παρωχημένες συσκευές κινητής τηλεφωνίας, αναλαμβάνοντας τις δυνατότητες χρήσης των έξυπνων κινητών τηλεφώνων, κάνουν στροφή 180^{o} προς την νέα τεχνολογία. Πλέον οι υπηρεσίες επικοινωνίας μέσω video, η ανταλλαγή φωτογραφιών, η χρήση του mobile banking θεωρούνται δεδομένα την σύγχρονη εποχή σε σχέση με παλαιότερες εποχές όπου τα τελευταία θεωρούνταν πολυτέλεια.

Από την μεριά τους οι πάροχοι υπηρεσιών κινητής τηλεφωνίας προχώρησαν σε επενδύσεις οι οποίες απέδωσαν τα μέγιστα. Η ευρεία αύξηση των υπηρεσιών δεδομένων σε συνδυασμό με την μείωση του κόστους απέδωσε τα μέγιστα στις επενδύσεις τους και η οικονομική απόσβεση έγινε σε πολύ σύντομο χρόνο.

Το έτος 2019 θεωρείται η μετάβαση στα δίκτυα υπερύψηλων ταχυτήτων τόσο από άποψη σταθερών συνδέσεων με οπτική ίνα όσο και σε ασύρματα δίκτυα 5ης γενιάς **(5G)**. Ωστόσο, όλες αυτές οι υπηρεσίες αποτελούν πρόκληση στους παρόχους κινητής τηλεφωνίας. Οι τελευταίοι καλούντα πλέον να επενδύσουν σε ένα δίκτυο υψηλών ταχυτήτων με την μέγιστη ασφάλεια, την ελάχιστη καθυστέρηση και το ελάχιστο κόστος. Καθώς το κόστος των υπηρεσιών μειώνεται με την πάροδο του χρόνου έχουμε ώθηση επιπρόσθετων υπηρεσιών, πέρα των επικοινωνιών με φωτογραφίες και video κυρίως μέσω των κινητών συσκευών. (Taleb:1, Shahzadi:2)

Όλη αυτή η προαναφερθείσα συνισταμένη της χρήσης των κινητών, των αναδυόμενων υπηρεσιών και η αύξηση των απαιτήσεων οδηγεί σε ένα αυστηρό πλαίσιο το οποίο αποκαλείται **"Ποιότητας της Εμπειρίας"**, το λεγόμενο **Quality of Experience (QoE)**. Ένα δείγμα των κανονισμών που επιβάλει το **QoE** είναι οι προδιαγραφές των δικτύων 5ης γενιάς **(5G)** προκειμένου, το τελευταίο να ανταποκρίνεται στο **QoE**. Αναλυτικά, η λειτουργία του δικτύου 5ης γενιάς **(5G)** θα πρέπει να υποστηρίζει διακίνηση όγκου δεδομένων 1000 φορές μεγαλύτερο σε σχέση με τον όγκο δεδομένων του δικτύου 4ης γενιάς **(4G)**. Όσον αφορά το θέμα του δικτύου θα πρέπει η ονομαστική ταχύτητα μετάδοσης να ισούνται, τουλάχιστον, με

10 Gb/s και η μέγιστη καθυστέρηση να κυμαίνεται περί τα **5ms**. Όσον αφορά το θέμα των διασυνδέσεων, θα πρέπει κάθε κυψέλη να υποστηρίζει τουλάχιστον 300.000 συνδέσεις για να υποστηρίξει την αναδυόμενη υπηρεσία της κινητής και διάχυτης υπολογιστικής **(Ubiquitous Computing)**. Τέλος, όσον αφορά την λειτουργία του δικτύου 5ης γενιάς θα πρέπει να έχει 99,999% αξιοπιστία και 90% χαμηλότερη κατανάλωση ενέργειας. (Taleb:1)

Εκτός από όλα τα παραπάνω που αφορούν το πλαίσιο ή γενικά τις προδιαγραφές του δικτύου 5ης γενιάς, θα πρέπει να ληφθούν υπόψιν και ο παράγοντας της διαλειτουργικότητας των δικτύων δηλαδή η συνεργασία και με άλλα ετερογενή δίκτυα όπως τα δίκτυα 4η γενιάς και οι ενσύρματες συνδέσεις.

Οι προκλήσεις του δικτύου 5ης γενιάς δεν σταματά εδώ. Είναι γεγονός ότι οι υπερύψηλες ταχύτητες είναι αρεστές, ωστόσο, δεν προσφέρονται μόνο για βιντεοκλήσεις με πολύ καλή ποιότητα εικόνας. Θα έλεγε κανείς ότι τα δίκτυα 5ης γενιάς είναι ιδανικά για υπηρεσίες αποκλειστικού περιεχομένου **(Context-aware Services)** και υπηρεσίες εγγύτητας **(Proximity Services)** δίνοντας έμφαση σε χώρους με υψηλή συγκέντρωση ανθρώπων και εν κινήσει.

Όλα τα παραπάνω, οδήγησαν στην ανάπτυξη καινοτόμων τεχνολογιών όπως της Υπολογιστική Νέφους **(Cloud Computing)** και η Αποκεντρωμένη Υπολογιστική **(Edge Computing)** όπου η χρήση αυτών είναι καίριος παράγοντας για την βελτίωση του **QoE** των δικτύων 5ης γενιάς.

2.2 Ιστορία και τυποποίηση

Τον Δεκέμβριο του 2014 ο **European Telecommunication of Standards Institute (ETSI)** εισήγαγε τον όρο **"Mobile Edge Computing"**. Ουσιαστικά πρόκειται για μια αρχιτεκτονική η οποία προοριζόταν να υιοθετηθεί από διαφόρους παρόχους υπηρεσιών Υπολογιστικής Νέφους **(Cloud Computing)**. Είναι δεδομένο ότι η σύγκλιση προς τα δίκτυα 5ης γενιάς ώθησε παρόχους Υπολογιστικού Νέφους να παρέχουν υπηρεσίες και εφαρμογές στους τελικούς χρήστες. Η αρχιτεκτονική του νέφους, οι διαλείψεις, η συμφόρηση και η αδιάλειπτη χρήση του όλου δικτύου, επιφέρουν καθυστέρηση **(Latency)** στην απόκριση οποιασδήποτε υπηρεσίας. Σύμφωνα με την αρχιτεκτονική **"Mobile Edge Computing"** οι υπηρεσίες και οι εφαρμογές που είναι εγκατεστημένες στο νέφος, αποκεντρώνονται και εγκαθιστώνται σε έναν τοπικό κόμβο. Το κύριο χαρακτηριστικό αυτού του κόμβου είναι ότι οι υπηρεσίες και οι εφαρμογές θα παρέχονται στα όρια κάλυψης του δικτύου του και θα παρέχονται στους τελικούς χρήστες λόγω της εγγύτητας τους. Έτσι εισάγεται και ο όρος Αποκεντρωμένη

Υπολογιστική **(Edge Computing)**. Αποτέλεσμα της Αποκεντρωμένης Υπολογιστικής **(Edge Computing)** είναι η μείωση της κίνησης του δικτύου κορμού και η μείωση της καθυστέρησης **(Latency)** στην απόκριση των υπηρεσιών. Επιπροσθέτως, δύναται στους τοπικούς κόμβους να εκτελούν βαριές υπολογιστικές διεργασίες, για χάρη των κινητών συσκευών, εξαιτίας του γεγονότος ότι οι τελευταίες δεν έχουν τους απαιτούμενους πόρους.

Λαμβάνοντας υπόψιν τα παραπάνω, η αρχιτεκτονική της αποκεντρωμένης υπολογιστικής αποτελεί βασικό πυλώνα του δικτύου 5ης γενιάς. Πλέον οι σταθμοί βάσης θα έχουν τον επιπλέον ρόλο του Υπολογιστικού Νέφους και κάθε τοπικός κόμβος θα έχει τον ρόλο της Αποκεντρωμένης Υπολογιστικής.

Το 2016 ο Ευρωπαϊκός Οργανισμός Τηλεπικοινωνιακών Προτύπων αντικατέστησε τον όρο **"Mobile Edge Computing"** με τον όρο **"Multi-Access Edge Computing"** έτσι ώστε να διευρυνθεί η αρχική αρχιτεκτονική ενσωματώνοντας και ετερογενή δίκτυα όπως το **WiFi** και τα δίκτυα οπτικών ινών.

Συμπερασματικά, η αρχιτεκτονική της Αποκεντρωμένης Υπολογιστικής **(Edge Computing)** δεν αφορά μόνο την μείωση της καθυστέρησης **(Latency)** στην απόκριση των υπηρεσιών και την μείωση της κίνησης στο δίκτυο κορμού. Η αρχιτεκτονική της Αποκεντρωμένης Υπολογιστικής **(Edge Computing)** είναι μια πλήρης δομή με πλατφόρμα και υπηρεσίες προς τους τελικούς χρήστες οι οποίοι έχουν πρόσβαση από οποιαδήποτε συσκευή είτε κινητή είτε σταθερή όπως ο προσωπικός υπολογιστής. (Taleb:1,)

2.3 Σύγκλιση προς την MEC

Η ευρεία χρήση των έξυπνων τηλεφώνων και η εξέλιξη των δικτύων 5ης γενιάς ώθησε τους παρόχους σε δημιουργία αρχιτεκτονικών και υποδομών υπολογιστικού νέφους. Από την πλευρά των κατασκευαστών των έξυπνων τηλεφώνων υφίστανται κατασκευαστικοί περιορισμοί από την άποψη της επεξεργαστικής ισχύος, μνήμης και κατανάλωσης ενέργειας. Επομένως, όλο το βάρος πέφτει στους παρόχους οι οποίοι καλούνται πλέον να βρουν την χρυσή τομή.

Η χρήση του **Mobile Cloud Computing (MCC)** ήταν μια πρωτοποριακή ιδέα για την εποχή, όπου ο τελικός χρήστης θα ήταν κερδισμένος. Αναλυτικά, με την χρήση του **MCC** ο τελικός χρήστης θα απολάμβανε υπηρεσίες πέραν των δυνατοτήτων της κινητής του συσκευής ή του έξυπνου τηλεφώνου του. Χρησιμοποιώντας την αρχιτεκτονική **Infrastructure-as-a-Service (IaaS)**, οι επιπρόσθετες υπηρεσίες που προσφέρει η **MCC** στα έξυπνα τηλέφωνα είναι η

επέκταση του υλικού τους όσον αφορά τον χώρο αποθήκευσης και την υπολογιστική ισχύς. Επίσης, η αρχιτεκτονική **IaaS** δίνει την δυνατότητα στον τελικό χρήστη και κατ' επέκταση στους παρόχους κινητής τηλεφωνίας ή ακόμα στις εταιρείες λογισμικού να προσαρμόσουν την Υπολογιστική Νέφους σύμφωνα με τις ανάγκες τους και τα προϊόντα τους.

Ωστόσο, η όλη αδυναμία που αναδύεται στην Υπολογιστική Νέφους η καθυστέρηση **(Latency)** με αποτέλεσμα την υποβάθμιση των υπηρεσιών και των προϊόντων προς τους τελικούς χρήστες. Έτσι λοιπόν εισήχθηκε ο όρος αποκεντρωμένη υπολογιστική **(Edge Computing)**.

Αρχικά, η εισαγωγή προς την αποκεντρωμένη υπολογιστική έγινε με την αρχιτεκτονική των **Cloudlets** όπου θεωρείται πρόδρομος των επιμέρους αρχιτεκτονικών της **Ομιχλώδους Επεξεργασίας (Fog Computing)** και της **Multi-Access Edge Computing (MEC)**. (Taleb:2, Shahzadi:2)

2.3.1 Cloudlets

Ο όρος **Cloudlet** περιγράφει μια αρχιτεκτονική αποκεντρωμένης υπολογιστικής. Αναλυτικά, το μοντέλο **Cloudlet** αποτελείται από τρία επίπεδα **(Tiers)**. Το πρώτο επίπεδο θεωρείται η κινητή συσκευή ή το έξυπνο τηλέφωνο. Το δεύτερο επίπεδο είναι ο τοπικός **Cloudlet** κόμβος και το τρίτο το απομακρυσμένο Νέφος **(Cloud)**.

Η λειτουργία των **Cloudlets** βασίζεται στις Εικονικές Μηχανές **(Virtual Machines-VMs)**, δηλαδή σε εξομοιωμένα περιβάλλοντα πλήρως απομονωμένα. Το πλεονέκτημα των **VMs** είναι η ασφάλεια λειτουργίας τους, δηλαδή, σε ένα σύστημα με πολλαπλές **VMs** οι λειτουργίες μεταξύ αυτών δεν επηρεάζονται, δεν έχουν κοινούς πόρους και μπορούν να "καταστραφούν" οποιαδήποτε στιγμή όπως ακριβώς δημιουργήθηκαν. Επομένως, με την χρήση των **VMs** δύναται να δημιουργηθούν ποίκιλες εφαρμογές κάτω από ένα λειτουργικό σύστημα **(VMOs)**.

Όταν ένας χρήστης με μια κινητή συσκευή βρεθεί στην εγγύτητα του δικτύου ενός **Cloudlet** τότε δύναται σε αυτόν να χρησιμοποιήσει τις υπηρεσίες αυτού υπό την έννοια ότι ο τοπικός **Cloudlet** θα αναλάβει το υπολογιστικό φόρτο κάτι που δεν μπορεί να αναλάβει η κινητή συσκευή. Υπάρχουν δύο προσεγγίσεις όσον αφορά την χρήση των **Cloudlet** υπηρεσιών.

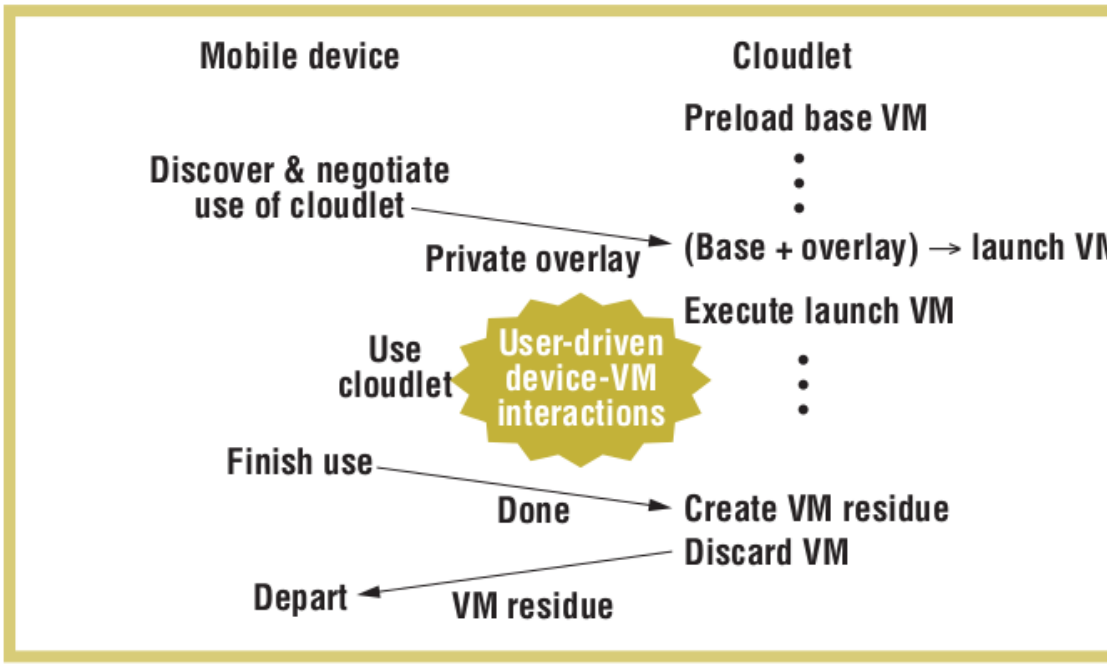
Η πρώτη προσέγγιση, η οποία ονομάζεται **VM Migration**, προϋποθέτει την εγκατάσταση μιας **VM** στην κινητή συσκευή. Ο τελικός χρήστης έχει δυνατότητα να προσαρμόσει την εγκατεστημένη **VM** στις ανάγκες του. Προκειμένου η κινητή συσκευή να απαλλαχθεί από τον υπολογιστικό φόρτο, η **VM** της κινητής συσκευής αναστέλλεται λειτουργικά και όταν ο

τελικός χρήστης βρεθεί εντός της εγγύτητας του δικτύου ενός τοπικού **Cloudlet** κόμβου τότε διαβιβάζονται οι καταστάσεις του επεξεργαστή, του δίσκου και της μνήμης στον τοπικό **Cloudlet** κόμβο. Στην συνέχεια η **VM** του **Cloudlet** θα εκτελεστεί στον τοπικό **Cloudlet** κόμβο από το τελευταίο σημείο που έχει ανασταλεί, σύμφωνα με τις διαβιβαζόμενες καταστάσεις του επεξεργαστή, του δίσκου και της μνήμης της κινητής συσκευής.

Η δεύτερη προσέγγιση προϋποθέτει την εγκατάσταση μιας **VM Overlay** στην κινητή συσκευή στην οποία **VM** έχει ανατεθεί μια υπηρεσία που θα χρησιμοποιήσει ο τελικός χρήστης. Κάθε **VM Overlay** προσαρμόζεται για κάθε συσκευή τελικού χρήστη. Στην προκειμένη περίπτωση, όταν ο τελικός χρήστης βρεθεί εντός της εγγύτητας του δικτύου ενός **Cloudlet** τότε η **VM** αναστέλλει την λειτουργία της και θα αποστείλει στο **Cloudlet** ένα **VM Overlay Image** δηλαδή ένα πιστό αντίγραφο της **VM Overlay**. Το **VM Overlay Image** περιέχει τα δεδομένα που έχουν συλλεχθεί από την εκτέλεση της στην κινητή συσκευή. Στην συνέχεια το **VM Overlay Image** συνεχίζει να εκτελείται από το σημείο που αναστάλθηκε την τελευταία φορά η **VM Overlay**. Αφού εκτελεστεί, επιστρέφει τα αποτελέσματα στον τελικό χρήστη.

Εκτός από την προαναφερθείσα ασφάλεια, επιπρόσθετα χαρακτηριστικά των **Cloudlets** και του Νέφους, γενικά, είναι οι μηχανισμοί πρόσβασης και αυθεντικοποίησης για την αποτροπή μη εξουσιοδοτημένης πρόσβασης. Επίσης, θεωρούνται μείζον ζητήματα η πολλαπλή υποστήριξη χρηστών, η κατανομή πόρων για την εκτέλεση εφαρμογών μεγάλου υπολογιστικού φόρτου προκειμένου όλο το σύστημα **Cloud- Cloudlet** να λειτουργεί εύρυθμα και το **handover** δηλαδή την διαδικασία κατά την οποία όταν ο τελικός χρήστης είναι συνδεδεμένος σε έναν τοπικό **Cloudlet** κόμβο και καθώς κινείται βρίσκεται έξω από τα όρια αυτού και στην συνέχεια διεισδύει στα όρια κάποιου άλλου **Cloudlet** κόμβου, τότε ο αρχικός κόμβος θα πρέπει να μεταφέρει όλα τα δεδομένα που αφορούν τον χρήστη στον τελικό **Cloudlet** κόμβο. Αποτέλεσμα του **handover** είναι η επιπρόσθετη επιβάρυνση του δικτύου κορμού. (Taleb:3)

Ένα παράδειγμα **Cloudlet** απεικονίζεται στην παρακάτω εικόνα.



Εικόνα 1: Μετάφραση κειμένου μέσω Cloudlet (Πηγή: Satyanarayanan 2009:7)

Στην περίπτωση που α
εγγύτητα του **Cloudlet**
συσκευή του τελικού χ
δεδομένα φωνής. Στην
αυθεντικοποίηση και η
φωνής στο **Cloudlet**.
τελικό χρήστη ενώ η **VM**

2.3.2 Ομιχλώδης Επεξεργασίας (Fog Computing)

Ως εξέλιξη της αρχιτεκτονικής των **Cloudlets** θεωρείται η **Fog Computing** ή **Ομιχλώδης Επεξεργασία**. Ο όρος **Fog Computing** περιγράφει μια αρχιτεκτονική αποκεντρωμένης υπολογιστικής προκειμένου να μειωθεί η καθυστέρηση **(Latency)** σε συστήματα Υπολογιστικού Νέφους. Η αρχιτεκτονική της **Fog Computing** αναπτύχθηκε από την **Cisco** και χαρακτηριστικό του είναι ότι οι **Fog Computing** κόμβοι δεν είναι κατανεμημένοι μόνο σε έναν περιορισμένο χώρο αλλά δύναται να είναι κατανεμημένοι σε διαφορετικές γεωγραφικές τοποθεσίες παρέχοντας ένα ολοκληρωμένο πακέτο υπηρεσιών. Κάθε κόμβος δύναται να φιλοξενεί είτε υπηρεσίες αποθήκευσης είτε διάφορες εφαρμογές στους τελικούς χρήστες οι οποίοι βρίσκονται εντός της εγγύτητας του δικτύου της.

Πλεονέκτημα που χαρακτηρίζει την **Fog Computing** αρχιτεκτονική είναι η ευελιξία σε θέματα υλικού **(Hardware)** και λογισμικού **(Software)**. Αναλυτικά, όσον αφορά το θέμα του υλικού, κάθε συσκευή δικτύωσης, όπως οι μεταγωγείς **(Switch)**, οι δρομολογητές **(Routers)** ή ακόμα και οι σταθμοί βάσης μπορούν να μετατραπούν σε κόμβους μιας μεγάλης **Fog Computing** δικτύωσης υιοθετώντας την αρχιτεκτονική της.

Όσον αφορά τα θέματα λογισμικού, η ευελιξία που παρέχει έχει να κάνει με την εγκατάσταση εφαρμογών σε κάθε **Fog Computing** κόμβο παρέχοντας μεγάλη πληθώρα υπηρεσιών. Το γεγονός αυτό την καθιστά κατάλληλη για εγκατάσταση υπηρεσιών που βασίζονται στο

Internet-of-Things. Το **Internet-of-Things** ή το **Διαδίκτυο των Πραγμάτων** είναι μια υπηρεσία σύμφωνα με την οποία κάθε αντικείμενο συνδέεται στο Διαδίκτυο και επικοινωνεί με άλλο αντικείμενο χωρίς την ανθρώπινη παρέμβαση. (Taleb:3, Shahzadi:5)

2.3.3 Multi-Access Edge Computing

Η αρχιτεκτονική **Multi-Access Edge Computing (MEC)** θεωρείται η καταληκτική εξέλιξη της **Cloudlets** αρχιτεκτονικής. Όπως προαναφέρθηκε το **European Telecommunication of Standards Institute** μετονόμασε τον αρχικό όρο από **Mobile Edge Computing (MEC)** σε **Multi-Access Edge Computing (MEC)** προκειμένου να ενσωματωθούν τα επιμέρους ετερογενή δίκτυα **(WiFi, Οπτική ίνα)** στην αρχιτεκτονική της.

Αν και ο τελικός χρήστης έχει πρόσβαση μέσω ασυρμάτου πρωτοκόλλου, η αρχιτεκτονική **MEC** του προσφέρει μια πληθώρα εφαρμογών, υπηρεσιών και καινοτομιών μέσα από ένα περιβάλλον υψηλών ταχυτήτων και χαμηλής καθυστέρησης **(Latency)**.

Η αρχιτεκτονική **MEC** προσφέρει ένα ασύρματο περιβάλλον με δυνατότητες αποθήκευση και υπολογιστικούς πόρους. Είναι με πλατφόρμα ανοιχτής πρόσβασης γεγονός που της δίνει πολλά πλεονεκτήματα για επενδύσεις. Προγραμματιστές και εταιρείες την αξιοποιούν κατασκευάζοντας διάφορες εφαρμογές ή ακόμα και **APIs** προς τους τελικούς χρήστες. Ειδικά, η χρήση της **MEC** από τους τηλεπικοινωνιακούς παρόχους της δίνει το πλεονέκτημα για την χρήση της προς την ανάπτυξη του **Διαδικτύου των Πραγμάτων- IoT**, της **Διάχυτης Υπολογιστικής (Ubiqutious Computing)** και της επικοινωνίας μεταξύ των μηχανών χωρίς την ανθρώπινη παρέμβαση **(Machine-To-Machine – M2M)**. (Taleb:3)

Εικόνα 2: Προϊόν δικτύωσης για εφαρμογές **M2M** με μέγιστη καθυστέρηση **1msec**

2.4 Σύγκριση Υπολογιστικής Νέφους και Αποκεντρωμένης Υπολογιστικής

Εν κατακλείδι, παρακάτω συνοψίζονται οι ουσιαστικές διαφορές μεταξύ των δύο υπολογιστικών αρχιτεκτονικών.

2.4.1 Υπολογιστική Νέφους (Cloud Computing)

Η αρχιτεκτονική Υπολογιστικής Νέφους απευθύνεται σε τελικούς χρήστες οι οποίοι είναι συνδεδεμένοι είτε μέσω ασύρματου ή ενσύρματου δικτύου. Το μέγεθος της είναι κλιμακωτό δηλαδή προσαρμόζεται ανάλογα το είδος της υπηρεσίας που παρέχει αφού διαθέτει μεγάλο αποθηκευτικό χώρο και υπολογιστικούς πόρους . Το γεγονός αυτό της δίνει περιθώρια να διαθέσει τον χώρο αποθήκευσης στους τελικούς χρήστες ή ακόμα να διαθέσει και τους

υπολογιστικούς πόρους στους προγραμματιστές. Αποτελείται από λίγους κόμβους οι οποίοι επικοινωνούν μεταξύ τους. Επειδή η πρόσβαση στον κόμβο είναι κυρίως μέσω του διαδικτύου, μεταξύ του χρήστη και του κόμβου μεσολαβούν αρκετοί ενδιάμεσοι κόμβοι ή **hops**.

Οι υπηρεσίες που παρέχονται στους τελικούς χρήστες εξέρχονται, αποκλειστικά, από τον κεντρικό κόμβο με αποτέλεσμα να υπάρχουν δυσχέρειες προς τους τελικούς χρήστες όπως η καθυστέρηση **(Latency)**. Εξαιτίας της καθυστέρησης δεν παρέχονται διαδραστικές υπηρεσίες από την αρχιτεκτονική Υπολογιστικής Νέφους. Τέλος, τα θέματα ασφαλείας της Υπολογιστικής Νέφους ρυθμίζονται από τον κεντρικό κόμβο.

2.4.2 Αποκεντρωμένη Υπολογιστική (Edge Computing)

Η αρχιτεκτονική Αποκεντρωμένης Υπολογιστικής απευθύνεται σε τελικούς χρήστες οι οποίοι είναι συνδεδεμένοι κυρίως ασύρματα και εφόσον βρίσκονται εντός της εγγύτητας του δικτύου του τοπικού κόμβου. Σε αντίθεση με την Υπολογιστική Νέφους, το υλικό της Αποκεντρωμένης Υπολογιστικής έχει περιορισμένες δυνατότητες σε χώρο αποθήκευσης και σε υπολογιστικούς πόρους με αποτέλεσμα κάθε κόμβος να προορίζεται για πολύ περιορισμένο αριθμό υπηρεσιών. Ωστόσο, το γεγονός της αποθήκευσης αντισταθμίζεται από το πλήθος των κόμβων, για παράδειγμα, εάν σε κάποιο κλειστό χώρο που συγκεντρώνονται άνθρωποι, υπάρχει μεγάλος αριθμός από τοπικούς κόμβους τότε μπορούμε να διαμοιράσουμε μια βάση δεδομένων σε κάθε έναν κόμβο.

Η σύνδεση του Νέφος με κάθε κόμβο της Αποκεντρωμένης Υπολογιστικής είναι άμεση, ενός **hop,** δηλαδή δεν μεσολαβούν επιμέρους κόμβοι.

Το γεγονός ότι:

1. κάθε κόμβος είναι προορισμένος για πολύ μικρό αριθμό υπηρεσιών καθώς,
2. η σύνδεση με τους τελικούς χρήστες είναι άμεση και ασύρματη,
3. υπάρχει περιορισμένος αριθμός χρηστών ο οποίος είναι συνδεδεμένος σε κάθε κόμβο εξαιτίας του γεγονότος ότι υπάρχουν πολλοί κόμβοι

δίνει, στην Αποκεντρωμένη Υπολογιστική, το πλεονέκτημα της μικρής καθυστέρησης **(Latency)**. Εξού και η Αποκεντρωμένη Υπολογιστική είναι κατάλληλη για διαδραστικές υπηρεσίες. Τέλος, τα θέματα ασφαλείας της Αποκεντρωμένης Υπολογιστικής ρυθμίζονται τοπικά σε κάθε κόμβο.

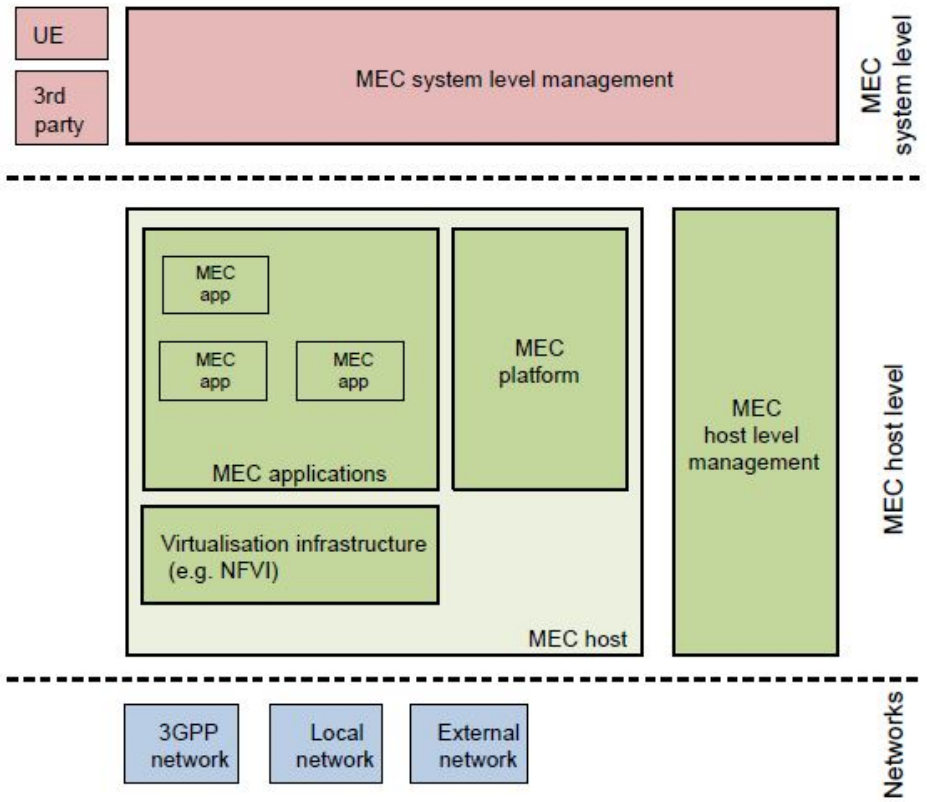
Κεφάλαιο 3
Αρχιτεκτονική , Περιπτώσεις Χρήσης και Τεχνολογίες

3.1 Θέσπιση της Αρχιτεκτονικής Αναφοράς

Το 2014 το **European Telecommunication of Standards Institute** σε συνεργασία με την επιτροπή **Industry Specification Group (ISG)** εκδώσαν την αρχιτεκτονική **ETSI MEC ISG** με σκοπό την παροχή Αποκεντρωμένης Υπολογιστικής από ασύρματα δίκτυα σε τρίτους, χωρίς τις δυσχέρειες της Υπολογιστικής Νέφους από την άποψη της καθυστέρησης **(Latency)** και υπολογιστικών πόρων. Η αρχιτεκτονική αποτελούταν από ένα σύνολο αλγορίθμων και τεχνικών κανονισμών και ανάλογα την περίπτωση χρήσης γινόντουσαν οι απαραίτητες προσαρμογές και τροποποιήσεις αυτών.

Μετά από εκτεταμένη έρευνα εκδόθηκε η τελική μορφή της αρχιτεκτονικής και των κανονισμών. Η τελική έκδοση της αρχιτεκτονικής ονομάζεται και **Αρχιτεκτονική Αναφοράς (Reference Architecture)**. Με την έκδοση της **Αρχιτεκτονική Αναφοράς** θεσπίστηκε η πλατφόρμα δημιουργίας και διαχείρισης εφαρμογών, οι αποκεντρωμένες υπηρεσίες εντός της εγγύτητας του δικτύου **(Edge Services)** και τα **APIs** έτσι ώστε οι κινητές συσκευές να έχουν πρόσβαση στις υπηρεσίες και στις εφαρμογές με την μικρότερη καθυστέρηση **(Latency)**. (Shahzadi:4, Taleb:2)

3.2 Δομή της MEC (MEC Framework)



Εικόνα 3: Δομή της MEC (Πηγή: ETSI GS MEC 003)

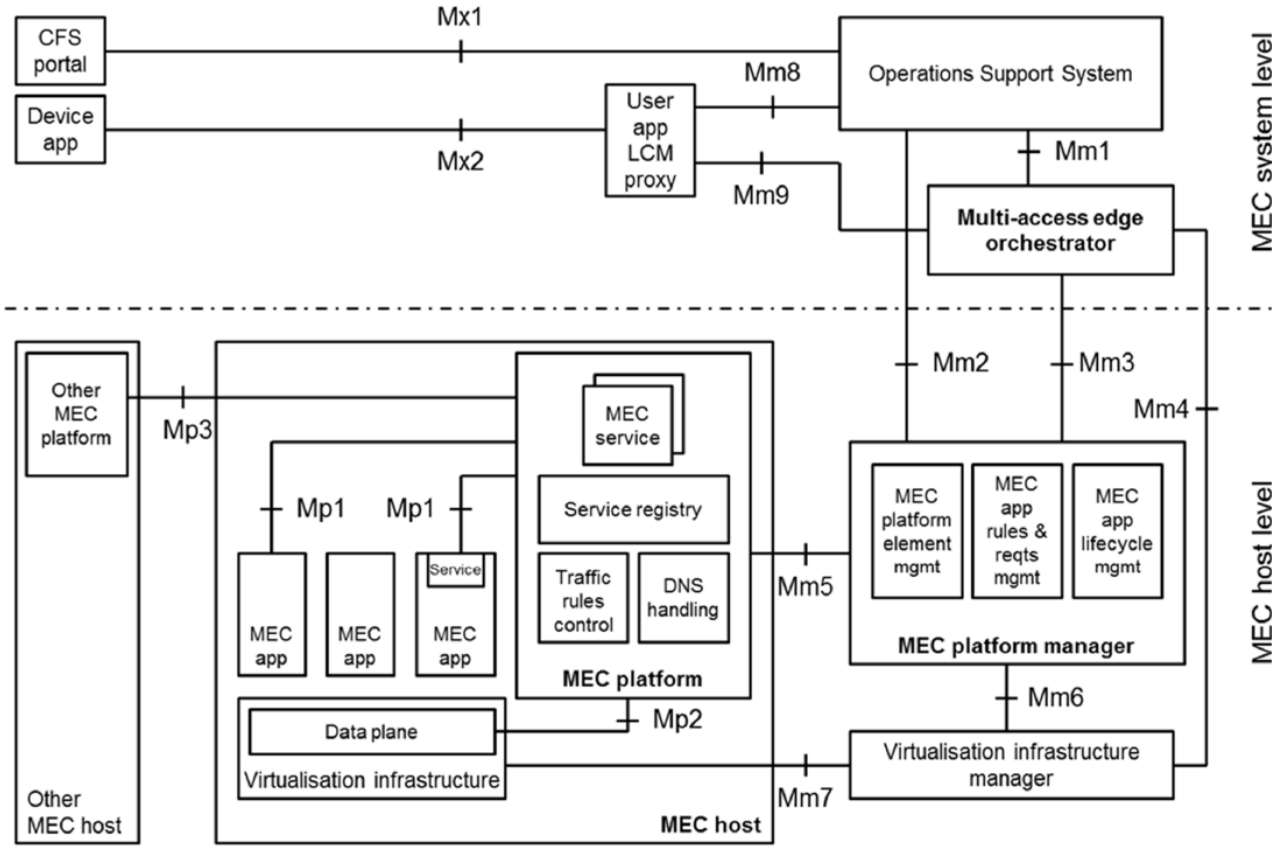
Η Εικόνα 3 απεικονίζει την δομή της **MEC**. Η δομή αποτελείται από τρία επίπεδα, το **Networks Level**, το **MEC Host Level** και το **MEC System Level**.

Το επίπεδο **Networks Level** παρέχει τις λειτουργίες που έχουν να κάνουν με την διασύνδεση του **MEC** είτε προς τις κινητές συσκευές, μέσω πρωτοκόλλου **3GPP** είτε μέσω του επιμέρους δικτύου με την χρήση των πρωτοκόλλων **WiFi (802.11x)**, **LANs (802.3x)** και οπτικής ίνας.

Το επίπεδο **MEC Host Level** αποτελείται από τις οντότητες **MEC Host Level Managment** και **MEC Host**. Η **MEC Host** οντότητα αποτελείται από την πλατφόρμα **(MEC Platform)**, τις εφαρμογές που προσφέρει στους τελικούς χρήστες **(MEC Applications)** και την **Virtualization Infrastructure**.

Το επίπεδο **MEC System Level** βρίσκεται στην κορυφή των επιπέδων και αποτελείται από το **MEC System Level Management** το οποίο διαχειρίζεται τις εφαρμογές, το **Operations Support System** και το **User Application Lifecycle Management Proxy.**

3.2.1 Αρχιτεκτονική Αναφοράς (Reference Architecture)



Εικόνα 4: Αρχιτεκτονική Αναφοράς (Reference Architecture) (Πηγή: ETSI GS MEC 003)

Η Εικόνα 4 απεικονίζει την Αρχιτεκτονική Αναφοράς της **MEC** όπου διακρίνονται τα επίπεδα **MEC Host Level** και **MEC System Level**. Κάθε επίπεδο απαρτίζεται από τις οντότητες του και κάθε οντότητα εκτελεί ένα σύνολο λειτουργιών. Μεταξύ των οντοτήτων υπάρχει επικοινωνία υπό την έννοια της ανταλλαγής δεδομένων μεταξύ τους με σκοπό την εύρυθμη λειτουργία της **MEC**. Η επικοινωνία μεταξύ των οντοτήτων ονομάζεται **"Σημείο αναφοράς"**.

Το επίπεδο **MEC System Level** αποτελεί την "πύλη εισόδου" των εφαρμογών του τελικού χρήστη προς την **MEC**, ενώ το επίπεδο **MEC Host Level** αποτελεί τον θεμέλιο λίθο της **MEC.** Το τελευταίο παρέχει την υποδομή εκτέλεσης, τις πολιτικές πόρων και αποθήκευσης των εφαρμογών και υπηρεσιών που φιλοξενεί η **MEC**.(ETSI GS MEC 003:9)

3.3 Οντότητες και σημεία αναφοράς

3.3.1 MEC Host

Ο **MEC Host** αποτελείται από την **MEC Platform**, την **Virtualization Infrastructure** και τις επιμέρους εφαρμογές **(MEC Applications) (**ETSI GS MEC 003:11).

MEC Platform

Η **MEC Platform** εκτελεί τις εξής λειτουργίες:

- Παρέχει χώρο αποθήκευσης στις εφαρμογές που φιλοξενεί.
- Παρέχει περιβάλλον εκτέλεσης κάθε εφαρμογής και ανακοινώνει τις υπηρεσίες της προς τους τελικούς χρήστες, με άλλα λόγια, όταν ένας χρήστης βρεθεί εντός της εγγύτητας και της εμβέλειας του **MEC** κόμβου, τότε δύναται να ανακαλύψει και να εξερευνήσει τις υπηρεσίες που προσφέρει η πλατφόρμα μέσω του κόμβου. Επίσης, δύναται η **MEC Platform** να ανακοινώνει επιμέρους υπηρεσίες οι οποίες παρέχονται από γειτονικούς ή απομακρυσμένους **MEC** κόμβους.
- Επικοινωνεί με τον **MEC Platform Manager** μέσω του σημείου αναφοράς **Mm5**. Συγκεκριμένα, από τον **MEC Platform Manager** λαμβάνει τις ρυθμίσεις που αφορούν την ίδια την **MEC Platform**. Επίσης, λαμβάνει κανόνες και απαιτήσεις που αφορούν τον κύκλο ζωής των εφαρμογών και της ανακατανομής τους που φιλοξενούνται στον **MEC Host**.
- Λαμβάνει τις εγγραφές της **DNS** υπηρεσίας από τον **MEC Platform Manager**, μέσω του σημείου αναφοράς **Mm5** και ενημερώνει τις αντίστοιχες **DNS** εγγραφές που βρίσκονται τοπικά σε αυτήν.
- Επικοινωνεί με την **Virtualization Infrastructure**, μέσω του σημείου αναφοράς **Mp2** διαβιβάζοντας εντολές λειτουργίας προς το πλάνο δεδομένων **(data plane)**. Οι εντολές λειτουργίας αφορούν κανονισμούς διακίνησης των δεδομένων προς τις εφαρμογές, τις υπηρεσίες και της διακίνησης του δικτύου.
- Επικοινωνεί με τις επιμέρους **MEC** πλατφόρμες, μέσω του σημείου αναφοράς **Mp3** ανταλλάσσοντας δεδομένα μεταξύ τους.
- Η **MEC Platform**, μέσω του σημείου αναφοράς **Mp1**, παρέχει από/σε κάθε εφαρμογή πληροφορίες που αφορούν την υπηρεσία της, την διαθεσιμότητα των υπηρεσιών προς τους τελικούς χρήστες, και επικοινωνιακή υποστήριξη μεταξύ επιμέρους εφαρμογών και υπηρεσιών. Κάθε εφαρμογή εκπροσωπείται από ένα αναγνωριστικό **(Token)**. Η **MEC Platform** μεταφράζει αυτό το αναγνωριστικό και προορίζει τα δεδομένα στην αντίστοιχη εφαρμογή.
- Επιπροσθέτως, παρέχει προς τους τελικούς χρήστες πληροφορίες που αφορούν την τρέχουσα κατάσταση συνεδρίας κάθε εφαρμογής, την δυνατότητα μετακίνησης κάθε εφαρμογής σε άλλον κόμβο, τους κανόνες διακίνησης των δεδομένων της εντός του δικτύου, διαμέσου του **DNS**, την πρόσβαση στον χώρο αποθήκευσης και την ώρα και

ημερομηνία.

Virtualization Infrastructure

Το πλάνο δεδομένων **(data plane)**, της **Virtualization Infrastructure,** επικοινωνεί με την **MEC Platform**, μέσω του σημείου αναφοράς **Mp2** λαμβάνοντας κανονισμούς λειτουργίας. Το πλάνο δεδομένων **(data plane)** παρέχει κανόνες διακίνησης των δεδομένων που προορίζονται στις εφαρμογές, κανόνες για την διακίνηση των υπηρεσιών και των δεδομένων του δικτύου.

Έχοντας η **Virtualization Infrastructure** λάβει τις εντολές από την **MEC Platform**, μέσω του σημείου αναφοράς **Mp2,** θα προβεί στην εφαρμογή και εκτέλεση των εντολών διακίνησης και δρομολόγησης δεδομένων είτε μεταξύ εφαρμογών και υπηρεσιών είτε προς τα τοπικά ή εξωτερικά δίκτυα.

Επιπροσθέτως, επικοινωνεί με τον **Virtualization Infrastructure Manager**, μέσω του σημείου αναφοράς **Mm7**, λαμβάνοντας κανονισμούς κατανομής, διαχείρισης και διαθεσιμότητας των διαθέσιμων πόρων.

MEC Applications

Οι εφαρμογές που φιλοξενεί η **MEC Host** εκτελούνται σαν εικονικές μηχανές **(Virtual Machines-VMs)** ή **Containers**. Η υποδομή εκτέλεσης των εφαρμογών παρέχεται από την **Virtualization Infrastructure**. Κάθε εφαρμογή επικοινωνεί με την **MEC Platform**, μέσω του σημείου αναφοράς **Mp1**. Σε κάθε εφαρμογή εφαρμόζονται, από την **MEC Platform**, κανονισμοί και απαιτήσεις που αφορούν τους πόρους που θα χρησιμοποιήσει, την μέγιστη καθυστέρηση **(Latency)** και ότι άλλο απαιτηθεί. Οι συγκεκριμένοι κανονισμοί ανακοινώνονται από το επίπεδο **MEC System Level**.

3.3.2 MEC Host Level Management

Ο **MEC Host Level Management** αποτελείται από την **MEC Platform Manager** και τον **Virtualization Infrastructure Manager (**ETSI GS MEC 003:13).

MEC Platform Manager

Ο **MEC Platform Manager** επικοινωνεί με την **MEC Platform**, μέσω του σημείου αναφοράς **Mm5**, για να διαβιβάσει τους κανόνες διακίνησης και διαχείρισης των εφαρμογών και

δεδομένων. Οι επιπρόσθετες λειτουργίες οι οποίες εκτελεί ο **MEC Platform Manager** είναι οι κάτωθι:

- Διαχειρίζεται τον κύκλο ζωής και τους κανονισμούς των εφαρμογών που είναι εγκατεστημένες στον **MEC Host**.
- Διαχειρίζεται τις εξουσιοδοτήσεις χρήσης των υπηρεσιών είτε για τους τελικούς χρήστες είτε για τρίτες εφαρμογές.
- Διευθύνει τους κανόνες διακίνησης δικτύου και τον **DNS** και επιλύει τυχόν συγκρούσεις **(Conflicts)** μεταξύ εφαρμογών.
- Λαμβάνει αναφορές σφαλμάτων, μετρήσεις απόδοσης και λειτουργίας από τον **Virtualization Infrastructure** για περαιτέρω επεξεργασία με σκοπό την κατάληξη σε κανονισμό.
- Επικοινωνεί με τον **Multi-Access Edge Orchestrator**, του **MEC System Level,** μέσω του σημείου αναφοράς **Mm3**, για να τον ενημερώσει για τους κύκλους ζωής των εφαρμογών και τους κανονισμούς λειτουργίας αυτών και την διαθεσιμότητα κάθε υπηρεσίας.
- Επικοινωνεί με τον **Operations Support System**, του **MEC System Level,** μέσω του σημείου αναφοράς **Mm2**, για να διαμορφώσει την λειτουργία της και να διαχειριστεί τα σφάλματα λειτουργίας.
- Επικοινωνεί με τον **Virtualization Infrastructure Manager**, μέσω του σημείου αναφοράς **Mm6**, για να διαχειριστεί την λειτουργία του σε θέματα εφαρμογών.

Virtualization Infrastructure Manager

Ο **Virtualization Infrastructure Manager** εκτελεί τις εξής λειτουργίες:

- Κατανέμει, διαχειρίζεται και ανακοινώνει τις διαθέσιμες υπηρεσίες της **Virtualization Infrastructure**. Οι υπηρεσίες αφορούν τους υπολογιστικούς πόρους, την αποθήκευση και την δικτύωση.
- Επικοινωνεί με την **Virtualization Infrastructure**, μέσω του σημείου αναφοράς **Mm7**, για να την διαμορφώσει και να την προετοιμάζει έτσι ώστε να εκτελεστεί ένα αντίγραφο μιας εφαρμογής **(Software Image)**. Επίσης, η προετοιμασία περιλαμβάνει και την αποθήκευση του παραπάνω αντιγράφου της εφαρμογής.
- Μετακίνηση μιας εφαρμογής στο Νέφος ή από ο Νέφος.
- Συλλέγει και αναφέρει πληροφορίες και σφάλματα της **Virtualization Infrastructure** και τα διαβιβάζει στον **MEC Platform Manager**.

3.3.3 MEC System Level Management

Ο **MEC System Level Management** αποτελείται από τον **Multi-Access Edge Orchestrator**, το **Operations Support System**, το **User Application Lifecycle Management Proxy**, το **Customer Facing Service Portal** και την **Device Application (**ETSI GS MEC 003:12).

Multi-Access Edge Orchestrator

Ο **Multi-Access Edge Orchestrator** αποτελεί της καρδιά του **MEC System Level Management** και εκτελεί τις εξής λειτουργίες:

- Διατηρεί την τρέχουσα κατάσταση της **MEC** αρχιτεκτονικής από άποψη εφαρμογών, υπηρεσιών, διαθέσιμων πόρων και τοπολογίας δικτύου.
- Λαμβάνει και καταχωρεί τις εισερχόμενες εφαρμογές και υπηρεσίες από τους τελικούς χρήστες προβαίνοντας στην αυθεντικοποίηση τους και καθιερώνει τις απαιτήσεις και τους κανόνες λειτουργίας των εφαρμογών. Οι απαιτήσεις και οι κανονισμοί προωθούνται στον **Mobile Edge Platform Manager**, μέσω του σημείου αναφοράς **Mm3**.
- Επιλέγει τον κατάλληλο **MEC** κόμβο για κάθε εφαρμογή ανάλογα με τις απαιτήσεις, όπως η καθυστέρηση **(Latency)**, οι διαθέσιμοι πόροι και οι υπηρεσίες.
- Μετακινεί κάθε εφαρμογή από κάποιον **MEC** κόμβο σε κάποιον άλλον, εάν και εφόσον απαιτηθεί.
- Επικοινωνεί με τον **Virtualization Infrastructure Manager**, μέσω του σημείου αναφοράς **Mm4**, για να προωθήσει τους κανόνες λειτουργίας της **Virtualization Infrastructure**.
- Επικοινωνεί με το **Operations Support System**, μέσω του σημείου αναφοράς **Mm1**, για να εκκινήσει και να τερματίσει κάθε εφαρμογή.
- Επικοινωνεί με τον **User Application Lifecycle Management Proxy**, μέσω του σημείου αναφοράς **Mm9**, για να διαχειριστεί κάθε εφαρμογή της κινητής συσκευής του τελικού χρήστη, η οποία αιτεί μια **MEC** εφαρμογή που είναι εγκατεστημένη στον **MEC Host**.

Operations Support System (OSS)

Το **Operations Support System** επικοινωνεί με το **Customer Facing Service Portal**, μέσω του σημείου αναφοράς **Mm8**, για να λάβει αιτήσεις από τις κινητές συσκευές του τελικού χρήστη οι οποίες αφορούν αιτήσεις προς τις εφαρμογές που βρίσκονται στον **MEC Host**. Έχοντας λάβει κάποια αίτηση από το **Customer Facing Service Portal**, το **Operations Support System** αποφασίζει την έγκριση ή την απόρριψη της αίτησης δηλαδή επιτρέπει ή όχι

την χορήγηση της αιτούσας υπηρεσίας. Η έγκριση ή η απόρριψη εξαρτάται από τις εκάστοτε συνθήκες. Κάθε εγκεκριμένη αίτηση προωθείται προς τον **Multi-Access Edge Orchestrator** για περαιτέρω επεξεργασία. Εάν υποστηρίζεται, το **Operations Support System** δύναται να λάβει αιτήσεις από κινητές συσκευές του τελικού χρήστη για να μετατοπίσει κάποια εφαρμογή από το Νέφος στον **MEC** κόμβο.

Επιπροσθέτως,

- Επικοινωνεί με τον **Multi-Access Edge Orchestrator**, μέσω του σημείου αναφοράς **Mm1**, για να εκκινήσει και να τερματίσει κάθε εφαρμογή.
- Επικοινωνεί με τον **Mobile Edge Platform Manager**, του **MEC System Level**, μέσω του σημείου αναφοράς **Mm2**, για να διαμορφώσει την λειτουργία της και να διαχειριστεί τα σφάλματα και την λειτουργικότητα της.
- Επικοινωνεί με τον **User Application Lifecycle Management Proxy**, μέσω του σημείου αναφοράς **Mm8**, για να διαχειριστεί τις αιτήσεις που προέρχονται από τις εφαρμογές της κινητής συσκευής του τελικού χρήστη, οι οποίες προορίζονται σε εφαρμογές του **MEC** συστήματος.
- Επικοινωνεί με τον **Customer Facing Service Portal**, μέσω του σημείου αναφοράς **Mx1**, όταν κάποιος κατασκευαστής λογισμικού αιτηθεί να εκτελεστεί μια εφαρμογή της εντός του **MEC** συστήματος.

User Application Lifecycle Management Proxy

Μια εφαρμογή ενός χρήστη **(User Application)** ορίζεται η εφαρμογή που βρίσκεται στον **MEC Host** η οποία αποκρίνεται μετά από έμμεση αίτηση του χρήστη. Η αίτηση προέρχεται από εφαρμογή η οποία εκτελείται στην κινητή του συσκευή.

Ο **User Application Lifecycle Management Proxy** επιτρέπει στις εφαρμογές της κινητής συσκευής να έχουν πρόσβαση στο **MEC** και να αιτηθούν την χρήση ή τον τερματισμό μιας **User Application**. Επίσης, εάν και εφόσον υποστηρίζεται, ο **User Application Lifecycle Management Proxy** επιτρέπει την μετατόπιση μιας **User Application** μέσα ή έξω από το **MEC** σύστημα. Επίσης, επιτρέπει την πληροφόρηση κάποιας εφαρμογής, την κινητής συσκευής του χρήστη, για την κατάσταση της αντίστοιχης **User Application**.

Επιπροσθέτως, επικοινωνεί με την κινητή συσκευή του τελικού χρήστη, μέσω του σημείου αναφοράς **Mx2**, όταν μια μια εφαρμογή **(Device Application)**, η οποία είναι εγκατεστημένη στην κινητή συσκευή του τελικού χρήστη, αιτηθεί να εκτελεστεί μια εφαρμογή η οποία είναι εγκατεστημένη στο **MEC** σύστημα. Επίσης, μέσω του σημείου αναφοράς **Mx2**, δύναται μια εφαρμογή να μετακινηθεί εντός ή εκτός του **MEC** συστήματος.

Customer Facing Service Portal

Το **Customer Facing Service Portal** επιτρέπει σε κατασκευαστές λογισμικού να επιλέξουν ή να παραγγείλουν ένα σύνολο από προεγκατεστημένες εφαρμογές του **MEC** συστήματος, για τις δικές τους ανάγκες με αποδέκτη τον τελικό χρήστη.

Device Application

Η **Device Application** είναι μια εφαρμογή η οποία είναι εγκατεστημένη στην κινητή συσκευή του τελικού χρήστη, η οποία εφαρμογή αλληλεπιδρά με μια **MEC** εφαρμογή μέσω κατάλληλης διεπαφής **(Interface)**.

3.4 Βοηθητικές Τεχνολογίες

Όπως έχει αναφερθεί σε προηγούμενο Κεφάλαιο, η αρχιτεκτονική **MEC** είναι επέκταση της αρχιτεκτονικής της Υπολογιστικής Νέφους, βελτιώνει και επεκτείνει τις δυνατότητες των κινητών συσκευών και βασίζεται στην τεχνολογία των Εικονικών Μηχανών ή των **Containers**. Ωστόσο, το οικοδόμημα της αρχιτεκτονική **MEC** στηρίζεται και σε επιμέρους βοηθητικές τεχνολογίες όπως η **Network Function Virtualization (NFV)**, τα **Software Defined Networks (SDN)** και το **Network Slicing**. Η χρήση των βοηθητικών τεχνολογιών δίνει στην αρχιτεκτονική **MEC** την απαραίτητη ευελιξία και πολυχρηστική υποστήριξη (Taleb:6).

3.4.1 Μοντέλα Υπολογιστικού Νέφους

Η αρχιτεκτονική Υπολογιστικής Νέφους προσφέρει υπηρεσίες στους τελικούς χρήστες, μέσα από ένα περιβάλλον με υψηλούς υπολογιστικούς πόρους και μεγάλο χώρο αποθήκευσης. Οι παραπάνω δυνατότητες του υλικού **(Hardware)** καθιστά την Υπολογιστική Νέφους μια πλατφόρμα παροχής πολλών υπηρεσιών. Οι τελικοί χρήστες δεν εστιάζουν στο γεγονός ότι για την χρήση ενός συνόλου υπηρεσιών είναι αναγκασμένοι να κατέχουν συσκευές υψηλών επιδόσεων αφού η Υπολογιστική Νέφους ανταποκρίνεται στις προσδοκίες τους.

Πλέον στην αγορά υπάρχουν οργανισμοί οι οποίοι προσφέρουν υπηρεσίες Υπολογιστικής Νέφους, όπως η **Google Drive** η οποία προσφέρει χώρο αποθήκευσης με την κατάλληλη διεπαφή **(Interface)**. Γενικά, τα μοντέλα της Υπολογιστικής Νέφους χωρίζονται σε δυο κατηγορίες. Αυτές είναι τα τεχνολογικά μοντέλα και τα μοντέλα υπηρεσιών.

Υπάρχουν τα εξής τεχνολογικά μοντέλα Υπολογιστικής Νέφους στην αγορά:

(a) **Private Cloud**. Το μοντέλο **Private Cloud** απευθύνεται κυρίως σε εταιρείες ή οργανισμούς. Κάθε εταιρεία ή οργανισμός κατέχει τις δικές της υπηρεσίες Υπολογιστικής Νέφους, αποκλειστικά, προς δική της χρήση. Οι υπηρεσίες προς τους χρήστες των εταιρειών ή των οργανισμών παρέχονται με εξουσιοδότηση διαφορετικά κάθε μη εξουσιοδοτημένος χρήστης βρίσκεται αντιμέτωπος με τοίχος προστασίας **(Firewall)**.

(b) **Public Cloud**. Το μοντέλο **Public Cloud** παρέχει υπηρεσίες οι οποίες είναι δημόσιες, σε αντίθεση με το μοντέλο **Private Cloud**. Η πρόσβαση των υπηρεσιών του **Public Cloud** παρέχεται σε εξουσιοδοτημένους χρήστες έναντι χρηματικού αντιτίμου.

(c) **Hybrid Cloud**. Το μοντέλο **Hybrid Cloud** παρέχει έναν συνδυασμό υπηρεσιών του **Public Cloud** και **Private Cloud.**

(d) **Community Cloud**. Το μοντέλο **Community Cloud** είναι μια δομή παροχής υπηρεσιών προς τελικούς χρήστες. Κύριο χαρακτηριστικό της δομής είναι ότι δημιουργήθηκε από διαφορετικούς οργανισμούς οι οποίοι διαμοιράζονται κάτι κοινό μεταξύ τους, όπως για παράδειγμα, θέματα ασφαλείας δικτύου.

Ομοίως, τα μοντέλα υπηρεσιών χωρίζονται στις εξής κατηγορίες:

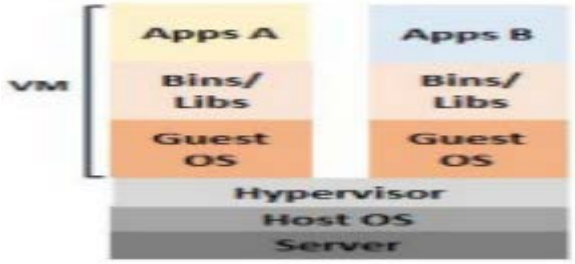
(a) **Infrastructure-as-a-service (Iaas)**. Το μοντέλο **Infrastructure-as-a-service** προσφέρει μια πλήρη εικονική υπολογιστική δομή. Με άλλο λόγια προσφέρει έναν αποθηκευτικό χώρο, υπολογιστικούς πόρους και υποδομή για δικτύωση. Με αυτά τα μέσα, ο τελικός χρήστης ή ένας οργανισμός μπορεί να αναπτύξει την υπηρεσία της αρεσκείας του.

(b) **Platform-as-a-service (Paas)**. Το μοντέλο **Platform-as-a-service** προσφέρει μια πλατφόρμα με τα εργαλεία δημιουργίας και ανάπτυξης εφαρμογών και υπηρεσιών.

(c) **Software-as-a-service (Saas)**. Το μοντέλο **Software-as-a-service** προσφέρει έτοιμο λογισμικό πρόσβασης το οποίο λειτουργεί σαν διεπαφή μεταξύ της υπηρεσίας και του τελικού χρήστη.

3.4.2 Εικονικές Μηχανές (Virtual Machines - VMs) και Containers

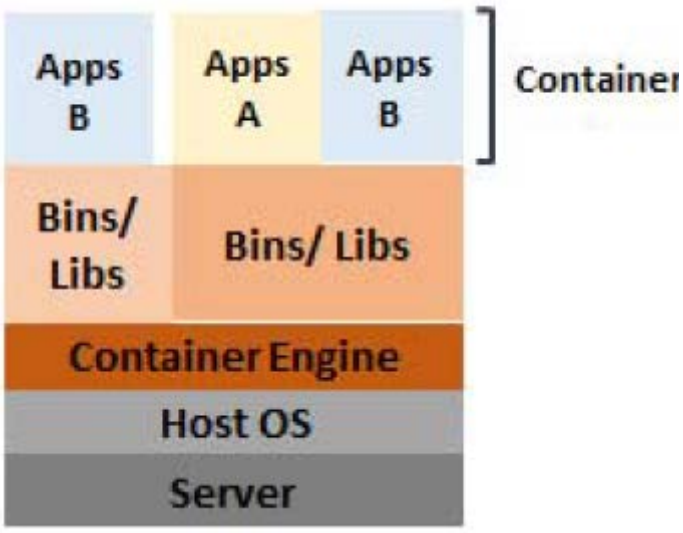
Μια πλατφόρμα Υπολογιστικής Νέφους αποτελείται από ένα σύνολο υπολογιστικών μονάδων **(Hardware)** όπου η σύνθεση αυτών δημιουργεί την οντότητα του Υπολογιστικού Νέφους. Η απαίτηση είναι ότι αυτή η οντότητα θα πρέπει να διαμοιράζεται στους τελικούς χρήστες, στους οργανισμούς ή οπουδήποτε αλλού ζητείται η υπηρεσία που παρέχει. Για να επιτευχθεί ο διαμοιρασμός χρησιμοποιούνται διάφορες αρχιτεκτονικές όπου μια από αυτές ονομάζεται **Hypervisor**. Το επίπεδο της **Hypervisor** βρίσκεται πάνω από το επίπεδο του λειτουργικού συστήματος που έχει εγκατασταθεί στις υπολογιστικές μονάδες.



Εικόνα 5: Αρχιτεκτονική Hypervisor

Πάνω στην **Hypervisor** εκτελούνται οι Εικονικές Μηχανές **(Virtual Machines – VMs)**. Κάθε Εικονική Μηχανή εκτελεί μια συγκεκριμένη λειτουργία και είναι απομονωμένη από τις επιμέρους Εικονικές Μηχανές που μπορεί να εκτελούνται, ταυτόχρονα, πάνω στην **Hypervisor**. Το γεγονός ότι οι Εικονικές Μηχανές εκτελούνται πλήρως απομονωμένες τις δίνει το πλεονέκτημα της ασφάλειας και της ευστάθειας αφού δεν έχει κοινούς πόρους με άλλες. Επίσης, οι Εικονικές Μηχανές δύναται να εκκινηθούν και να τερματιστούν ανά πάσα στιγμή χωρίς να επηρεάζεται η λειτουργία των υπολογιστικών μονάδων. Ωστόσο, οι Εικονικές Μηχανές χρησιμοποιούν πλήρως τους πόρους του λειτουργικού συστήματος που φιλοξενούνται, όπως τους υπολογιστικούς πόρους και μέρος του χώρου αποθήκευσης με αποτέλεσμα η διαδικασία εκκίνησης αυτών να είναι σχετικά αργή.

Μια άλλη αρχιτεκτονική που εφαρμόζεται είναι η αρχιτεκτονική της **Container Engine** η οποία βρίσκεται ένα επίπεδο πάνω από το λειτουργικό σύστημα των υπολογιστικών μονάδων.



Εικόνα 6: Αρχιτεκτονική Container Engine

Κάθε **Container Engine** παράγει ένα **Container** το οποίο είναι ένα στιγμιότυπο μιας υπηρεσίας που παρέχει το Υπολογιστικό Νέφος. Αντίθετα με τις Εικονικές Μηχανές, τα **Containers** χρησιμοποιούν ένα μέρος των πόρων του λειτουργικού συστήματος. Με άλλα λόγια τα **Containers** χρησιμοποιούν μόνο τις βιβλιοθήκες που είναι απαραίτητες για την εκτέλεση κάποιας υπηρεσίας και όχι όλες τις βιβλιοθήκες όπως γινόταν με τις Εικονικές Μηχανές. Η εκτέλεση και ο τερματισμός κάθε **Container** γίνεται εύκολα στις υπολογιστικές μονάδες και ο χρόνος εκκίνησης τους κυμαίνεται μεταξύ μερικών χιλιοστών του δευτερολέπτου σε αντίθεση με τον χρόνο εκκίνησης των Εικονικών Μηχανών ο οποίος κυμαίνεται μεταξύ μερικών δευτερολέπτων. Ωστόσο, μειονέκτημα της χρήσης των **Containers** είναι το θέμα ασφαλείας και ευστάθειας καθώς δύναται να υπάρξει σύγκρουση **(Conflict)** μεταξύ εφαρμογών για τον ίδιο πόρο.

Συνοψίζοντας, τα **Containers** αποτελούν οντότητες οι οποίες δεν καταναλώνουν σημαντικούς πόρους από τις υπολογιστικές μονάδες γιατί χρησιμοποιούν τις απαραίτητες βιβλιοθήκες για κάθε υπηρεσία, εκκινούνται γρήγορα δηλαδή σε πολύ μικρό χρονικό διάστημα και μοιράζονται τον ίδιο πυρήνα **(Kernel)** του λειτουργικού συστήματος. Ωστόσο, δεν παρέχουν ασφάλεια και ευστάθεια και δεν θεωρούνται επαρκείς σε βαριές εφαρμογές. Οι Εικονικές Μηχανές λειτουργούν απομονωμένα το οποίο σημαίνει ότι χρησιμοποιούν όλες τις παρεχόμενες βιβλιοθήκες, έχουν μεγάλο χρόνο εκκίνησης, δεν μοιράζονται τον πυρήνα **(Kernel)** αλλά κάθε Εικονική Μηχανή χρησιμοποιεί εξ ολοκλήρου ένα **Image** αυτού, παρέχουν την μέγιστη ασφάλεια και ευστάθεια και θεωρούνται κατάλληλες για βαριές εφαρμογές.

Λαμβάνοντας υπόψιν τα παραπάνω, τα **Containers** αποτελούν μια απλή και χωρίς απαιτήσεις λύση για την **MEC** αρχιτεκτονική η οποία απευθύνεται σε τελικούς χρήστες με κινητές συσκευές. Το γεγονός της γρήγορης εκκίνησης τους τα καθιστά ιδανικά για περιπτώσεις διασύνδεσης πολλαπλών χρηστών σε έναν **MEC** κόμβο αλλά και στην περίπτωση διασύνδεσης πολλαπλών **MEC** κόμβων μεταξύ τους. Επιπροσθέτως, κάθε **Container** στιγμιότυπο δύναται να αποθηκευτεί. Η **Container Engine** μπορεί να το ανακαλεί όταν και όποτε χρειαστεί και παρέχεται η δυνατότητα η ρύθμιση του κύκλου ζωής κάθε **Conatier** μέσω της κατάλληλης **API**.

Από την άλλη, οι Εικονικές Μηχανές αποτελούν με την σειρά τους λύση σε **MEC** αρχιτεκτονική σε περιπτώσεις όπου δεν υπάρχει η κινητικότητα υπηρεσιών γιατί θα

απαιτούσε πολύ χρόνο η όλη διαδικασία της κινητικότητας λόγω του χρόνου εκκίνησης των Εικονικών Μηχανών. Επομένως, οι Εικονικές Μηχανές είναι καταλληλότερες σε περιβάλλονται που εκτελούν βαριές υπολογιστές εργασίες όπως σε εφαρμογές του Διαδικτύου των Πραγμάτων (Taleb:6).

3.4.3 Network Function Virtualization

Η τεχνολογία ή αρχιτεκτονική **Network Function Virtualization (NFV)** εξομοιώνει τις λειτουργίες δικτύωσης, οι οποίες εκτελούνται στο επίπεδο υλικού **(Hardware)**, να εκτελούνται στο επίπεδο λογισμικού μέσω των τεχνολογιών εικονοποίησης **(Virtualization)**. Με άλλα λόγια οι λειτουργίες δικτύωσης, όπως **Packet Data Network Serving- Gateways, Firewall,** κ.α. θα εκτελούνται μέσω λογισμικού. (Taleb:8).

Η **Network Function Virtualization (NFV)** είναι μια ευέλικτη αρχιτεκτονική ή οποία προσαρμόζεται ανάλογα με τις απαιτήσεις των υπηρεσιών. Επίσης, δύναται να δημιουργηθούν πολλαπλά στιγμιότυπα αυτής για κάθε Εικονική Μηχανή.

Η δομή της **NFV** αποτελείται από τα εξής:

(a) **Virtual Network Functions (VNFs)**. Η **Virtual Network Function** είναι μια λειτουργία δικτύου η οποία εκτελείται εικονικά δηλαδή μέσω λογισμικού εικονοποίησης.

(b) **Network Function Virtualization Infrastructure (NFVI)**. Είναι η δομή πάνω στην οποία εκτελείται η **VNF**. Η δομή αποτελείται από ένα σύνολο υλικού (όπως **CPU**) και λογισμικού.

(c) **NVF Management and Orchestration (NFV MANO)**. Το **NVF Management and Orchestration** είναι ένα πλαίσιο το οποίο διαχειρίζεται και οργανώνει τους πόρους της υποδομής **NFVI**, τους κύκλους ζωής και την αστοχία των **VNFs**.

Η χρήση της αρχιτεκτονικής **NFV** επιφέρει πολλά πλεονέκτημα στην αρχιτεκτονική **MEC**. Αναλυτικά, η αρχιτεκτονική **NFV** ελέγχει τα στιγμιότυπα των Εικονικών Μηχανών που σχετίζονται με τις εφαρμογές ή τις υπηρεσίες στην **MEC** και προσφέρει ευελιξία ή μετακίνηση ανάλογα με τις ανάγκες κάθε εφαρμογής. Παράδειγμα, εάν μια δημοφιλή εφαρμογή χρειαστεί επιπλέον πόρους τότε η επίτευξη μπορεί να γίνει είτε με ένα εικονικό λογισμικό, το οποίο θα παρέχεται από την **NFV** είτε μέσω υλικού δηλαδή με παροχή πραγματικής μνήμης ή επεξεργαστικής ισχύος.

Το πλαίσιο **NFV MANO** διαχειρίζεται τους κύκλους ζωής και την αστοχία των **VNFs**. Ειδικά, σε περίπτωση αστοχίας μιας **VNF** τότε το **NFV MANO** θα προβεί στην επανεκκίνηση ή στην αντικατάσταση της. Επίσης, το **NFV MANO** δύναται να προβεί σε διασύνδεση επιμέρους **MEC** κόμβων που βρίσκονται σε διαφορετικές γεωγραφικές τοποθεσίες. Παράδειγμα, εάν σε

κάποιον **MEC** κόμβο υπάρχει συμφόρηση λόγω πολλαπλών συνδέσεων με αποτέλεσμα την πτώση της απόδοσης του συγκεκριμένου **MEC** κόμβου, τότε το **NFV MANO** θα αναλάβει να μεταφέρει πόρους από άλλους **MEC** κόμβους έτσι ώστε να αυξηθεί η απόδοση του.

Εν κατακλείδι, η αρχιτεκτονική **NFV** παρέχει τις εξής διευκολύνσεις στην αρχιτεκτονική **MEC**:

(a) Κινητικότητα υπηρεσιών μεταξύ **MEC** κόμβων ή μεταξύ **MEC** κόμβων και του Νέφους ακόμα και εάν βρίσκονται σε διαφορετικά δίκτυα.

(b) Καθολική και πλήρης υποστήριξη υπηρεσιών σε όλη την έκταση του δικτύου ακόμα και στους κόμβους οι οποίοι είναι εγκατεστημένοι σε διαφορετικές γεωγραφικές τοποθεσίες.

(c) Διαμοιρασμός των πόρων εικονικού δικτύου ανάλογα με τις ανάγκες κάθε εφαρμογής.

(d) Διαθεσιμότητα πόρων για παροχή όταν αυτοί ζητηθούν.

3.4.4 Software Define Networks (SDNs)

Η τεχνολογία **Software Define Networks** προσφέρει μια δυναμική διαχείριση του δικτύου υπό την έννοια την προσαρμογής αυτού σε συγκεκριμένες συνθήκες. Η προσαρμογή επιτυγχάνεται με χρήση του κατάλληλου λογισμικού ή με άλλα λόγια με της ικανότητα προγραμματισμού αυτού **(Programmability)**.

Συγκεκριμένα, κάθε **MEC** κόμβος προσφέρει εφαρμογές, υπηρεσίες και δυνατότητα αποθήκευσης στους τελικούς χρήστες και ακόμα επιπλέον πόρους στις κινητές συσκευές υπό την προϋπόθεση όλοι οι παραπάνω να βρίσκονται εντός της εγγύτητας του δικτύου του **MEC** κόμβου. Η τεχνολογία **Software Define Networks** δίνει στην αρχιτεκτονική **MEC** την δυνατότητα διαχείρισης του δικτύου κατά το δοκούν ανεξάρτητα την ήδη υπάρχουσα υποδομή αυτού. Η διαχείριση γίνεται διαμέσου ενός κεντρικού ελεγκτή ή λογισμικού. Παρέχεται η δυνατότητα άμεσης δημιουργίας και άμεσου τερματισμού ενός εικονικού δικτύου προσαρμοσμένο σε κάποια συγκεκριμένη υπηρεσία ανεξάρτητα των εικονικών δικτύων που εκτελούνται εκείνη την χρονική στιγμή. Επίσης, η τεχνολογία **Software Define Networks** υποστηρίζει την κινητικότητα δηλαδή την περίπτωση όπου μια κινητή συσκευή μετακινείται μεταξύ διαφόρων κόμβων και η εφαρμογή που τρέχει πάνω σε αυτή την συσκευή απαιτεί ένα συγκεκριμένο εικονικό δίκτυο.

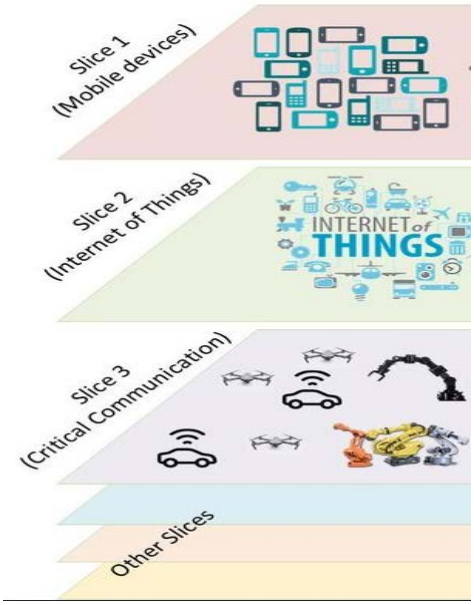
Η τεχνολογία **Software Define Networks** υπερνικά τις δυσχέρειες που δημιουργούνται με την διασύνδεση ετερογενών συσκευών διαμορφώνοντας κατάλληλα το σήμα του δικτύου. Η διασύνδεση ετερογενών συσκευών δημιουργεί προβλήματα στον κορμό του δικτύου καθώς ανάλογα με την υπηρεσία ή το υλικό απαιτείται και η ανάλογή διαμόρφωση είτε του σήματος

είτε της **IP** διεύθυνσης. Παράδειγμα, μια **IP** διεύθυνση που αφορά ένα σήμα **alarm** διαφέρει από μια **tunneling IP**.

Τέλος, η τεχνολογία **Software Define Networks** αναλαμβάνει τον ρόλο του μεταγωγέα **(Switch)** ή του δρομολογητή **(Router)** με την χρήση δεδομένων η οποία δύναται να ελέγχεται είτε κατά την μετάδοση προς τα έξω είτε κατά την μετάδοση προς τα μέσα του κόμβου (Taleb:8).

3.4.5 Network Slicing

Η τεχνολογία **Network Slicing** είναι μια αρχιτεκτονική διαχείρισης δικτύου. Αναλυτικά, πρόκειται για μια πολυπλεξία ή συνισταμένη εικονικών δικτύων δηλαδή ο διαχωρισμός του αρχικού δικτύου σε στιγμιότυπα από εικονικά δίκτυα. Όλες τα εικονικά δίκτυα έχουν ως κοινή υποδομή το αρχικό φυσικό δίκτυο.



Εικόνα 7: Network Slicing

Το αποτέλεσμα της **Network Slicing** είναι η πλήρης απομόνωση των εικονικών δικτύων δηλαδή η πλήρης απομόνωση των προσφερόμενων υπηρεσιών. Με άλλα λόγια η αρχιτεκτονική **Network Slicing** εισάγει ένα πολυχρηστικό και ευέλικτο περιβάλλον ανεξαρτήτων δικτύων, εύκολα ρυθμιζόμενων και προορισμένων για συγκεκριμένες υπηρεσίες.

Η Εικόνα 7 απεικονίζει ένα μοντέλο της αρχιτεκτονικής **Network Slicing**. Ο ρόλος της αρχιτεκτονικής **MEC** είναι καταλυτικός στις περιπτώσεις που απεικονίζονται.

Το εικονικό δίκτυο **Slice 1** χρησιμοποιείται από χρήστες έξυπνων τηλεφώνων, οι οποίοι χρησιμοποιούν εφαρμογές με περιεχόμενο όπως οι φωτογραφίες ή videos. Στην προκειμένη περίπτωση οι απαιτήσεις είναι η υψηλή ταχύτητα, επομένως, ο τοπικός **MEC** κόμβος δρα ως προσωρινή αποθήκευση **(Caching)** περιεχομένων με αποτέλεσμα οι τελικοί χρήστες να βρίσκουν στον κόμβο το περιεχόμενο που αναζητούν εύκολα και γρήγορα. Επίσης, ο τοπικός **MEC** κόμβος μπορεί να αναλάβει τον φωτορεαλισμό σε κάποια φωτογραφία απολαμβάνοντας, ο τελικός χρήστης, το βέλτιστο αποτέλεσμα.

Το εικονικό δίκτυο **Slice 2** θα χρησιμοποιηθεί για το Διαδίκτυο των Πραγμάτων και συγκεκριμένα για την διαχείριση Μεγάλων Δεδομένων. Στην προκειμένη περίπτωση, οι απαιτήσεις είναι η χαμηλή κατανάλωση ενέργειας, επομένως, ο **MEC** κόμβος θα αναλάβει την επεξεργασία των αλγορίθμων **Big Data Analytics** και ανάλογα την κρισιμότητα των δεδομένων θα προβεί σε έγκαιρη ειδοποίηση **(Alarm)** κάποιου επερχόμενου κινδύνου.

Το εικονικό δίκτυο **Slice 3** θα χρησιμοποιηθεί κυρίως σε αυτοκίνηση. Στην προκειμένη περίπτωση, οι απαιτήσεις η χαμηλή καθυστέρηση **(Latency)**, επομένως, στον **MEC** κόμβο θα λειτουργούν αυξημένες λειτουργίες δικτύου για να επιτευχθεί η χαμηλή καθυστέρηση (Taleb:9).

3.5 Περιπτώσεις Χρήσης

Η χρήση της Αποκεντρωμένης Υπολογιστικής και ιδίως της **MEC** βελτιώνει την ποιότητα παροχής υπηρεσιών και κυρίως την Ποιότητα Εμπειρίας **(Quality of Experience-QoE)**. Η χρήση της **MEC** έχει ως αποτέλεσμα την μείωση της καθυστέρησης **(Latency)** σε εφαρμογές που απαιτούν ταχύτατη απόκριση. Επιπροσθέτως, αποτελεί βασικό πυλώνα των δικτύων 5ης γενιάς **(5G)** και θεωρείται η εξέλιξη των σταθμών βάσης. Το γεγονός αυτό παρουσιάζεται στους παρόχους κινητής τηλεφωνίας και σε άλλες εταιρείες συνάμα, ως επιχειρηματική ευκαιρία με πολλαπλά οφέλη και έσοδα για αυτούς και ποιοτικές υπηρεσίες για τους τελικούς χρήστες. Σε αυτή την ενότητα θα παρουσιαστούν οι βασικές περιπτώσεις χρήσης όλης αυτής της αναδυόμενης τεχνολογίας.

3.5.1 Διαδίκτυο των Πραγμάτων (IoT)

Το Διαδίκτυο των Πραγμάτων είναι μια τεχνολογία όπου επιτρέπει την άμεση επικοινωνία διαφόρων "Πραγμάτων" μεταξύ τους, μέσω δικτύου, χωρίς ανθρώπινη παρέμβαση ή κάποιου ενδιάμεσου υπολογιστή. Η έννοια "Πραγμάτων" περικλείει συσκευές ή έμψυχα αντικείμενα. Παράδειγμα, υποθέτουμε ότι ένας ασθενής φοράει έναν βηματοδότη όπου στον οποίον, βηματοδότη, έχει ανατεθεί μια IP διεύθυνση. Ο βηματοδότης, μέσω IP δικτύου, επικοινωνεί με μια πλατφόρμα και στέλνει στην τελευταία δεδομένα που αφορούν την καρδιά του ασθενούς. Η πλατφόρμα είναι εγκατεστημένη στο Νέφος. Στην πλατφόρμα είναι εγκατεστημένη μια εφαρμογή, όπου μέσω της μηχανικής μάθησης, ειδοποιεί τον ιατρό εάν τα δεδομένα της καρδιάς είναι κρίσιμα.

Εγκαθιστώντας την παραπάνω περίπτωση χρήσης σε ένα σύστημα με βάση την αρχιτεκτονική **MEC** παρέχεται στην φορητή συσκευή του ασθενούς οι υπολογιστικοί πόροι και ο χώρος αποθήκευσης έτσι ώστε να η διάγνωση να γίνεται απρόσκοπτα. Στην προκειμένη περίπτωση, η διάγνωση πραγματοποιείται από μια πλατφόρμα που έχει εγκατασταθεί σε έναν τοπικό **MEC** κόμβο δηλαδή είναι αποκεντρωμένη από το Νέφος. Το αποτέλεσμα της παροχή υπολογιστικών και αποθηκευτικών πόρων από την **MEC** είναι η διάγνωση με μικρή καθυστέρηση **(Latency)** και η μείωση της κατανάλωσης ενέργειας της φορητής συσκευής του ασθενούς.

Επίσης, η μείωση της ροής του δικτύου εξαιτίας των **MEC** κόμβων και της χαμηλής κινητικότητας, λόγω της κατάστασης των ασθενών, επιτρέπει την εγκατάσταση ενός δικτύου μικρής χωρητικότητας. Τέλος, η εγκατάσταση φίλτρων σε κάθε **MEC** κόμβο θα έχει ως αποτέλεσμα την περαιτέρω μείωση των δεδομένων που θα διαβιβάζονται προς το Νέφος.

3.5.2 Μεγάλα Δεδομένα (Big Data)

Μεγάλα Δεδομένα **(Big Data)** είναι ένα πεδίο ανάλυσης ενός μεγάλου συνόλου δεδομένων όπου η εξαγωγή συμπερασμάτων απαιτεί την χρήση πολύπλοκων αλγορίθμων. Η διαδικασία εξαγωγής συμπερασμάτων από ένα μεγάλο σύνολο δεδομένων ονομάζεται **Big Data Analytics**.

Μια περίπτωση χρήσης είναι ο συνδυασμός των τεχνολογιών του Διαδικτύου των Πραγμάτων **(IoT)** με τα Μεγάλα Δεδομένα **(Big Data)**. Ο συνδυασμός αυτός δύναται να χρησιμοποιηθεί στις "έξυπνες" πόλεις. Σε μια "έξυπνη" πόλη εγκαθιστώνται ένα σύνολο από αισθητήρες ή κάμερες. Κάθε αισθητήρας μπορεί να καταγράφει οτιδήποτε, όπως για παράδειγμα μια θερμοκρασία ενώ με την χρήση των καμερών καταγράφεται η κυκλοφορία.

Αντιλαμβάνεται κανείς ότι ο όγκος των δεδομένων που συσσωρεύεται είναι αρκετά μεγάλος σε χωρητικότητα εάν και εφόσον η χρήση της τεχνολογίας είναι σε καθημερινή βάση. Η κατάσταση δυσχεραίνεται ακόμα περισσότερο καθώς το Διαδίκτυο των Πραγμάτων επιτρέπει την διασύνδεση πολλών συσκευών με αποτέλεσμα την διόγκωση των δεδομένων.
Η χρήση της **MEC** αρχιτεκτονικής αμβλύνει το παραπάνω πρόβλημα της επεξεργασίας και αποθήκευσης των δεδομένων. Κάθε **MEC** κόμβος επεξεργάζεται και αποθηκεύει τοπικά τα δεδομένα που συλλέγει από τους αισθητήρες. Με την χρήση των αλγορίθμων, **Big Data Analytics**, εξάγει συμπεράσματα ανάλογα την υπηρεσία που υποστηρίζει και απορρίπτει τα επιμέρους. Τα εξαγόμενα συμπεράσματα τα προωθεί στο Νέφος για αποθήκευση. Αποτέλεσμα της όλης διαδικασίας είναι η μείωση του φορτίου του δικτύου από και προς το Νέφος. Οφέλη της όλης διαδικασίας είναι η βελτίωση της **QoE** του τελικού χρήστη, έγκαιρη ειδοποίηση για κάποιο επικίνδυνο συμβάν κλπ.

3.5.3 Υπολογιστική Αποφόρτιση

Με τον όρο Υπολογιστική Αποφόρτιση **(Computation Offloading)** εννοούμε την διαδικασία όπου μια εφαρμογή, με απαιτήσεις σε υπολογιστικούς πόρους, “ανεβαίνει” από την κινητή συσκευή σε έναν **MEC** κόμβο προκειμένου να εκτελέσει κάποια υπηρεσία. Μετά το πέρας της εκτέλεσης της, η εφαρμογή επιστρέφει τα αποτελέσματα στην κινητή συσκευή. Το αποτέλεσμα είναι η μείωση της κατανάλωσης ενέργειας της κινητής συσκευής δηλαδή την αύξηση της διάρκειας της μπαταρίας.
Μια περίπτωση χρήσης είναι η βιντεοκλήση. Όταν πραγματοποιείται μια βιντεοκλήση η εφαρμογή επιλέγει τον κατάλληλο αλγόριθμο κωδικοποίησης της εικόνας, ειδικότερα όσον αφορά τον φωτορεαλισμό **(Rendering)**. Ο φωτορεαλισμός είναι μια διαδικασία σύνθεσης φωτισμού και χρωμάτων και είναι μια βαριά υπολογιστική διαδικασία. Ανεβάζοντας την εικόνα, από κάθε frame, στον τοπικό **MEC** κόμβο, ο τελευταίος, με την κατάλληλη εφαρμογή, αναλαμβάνει την διαδικασία του φωτορεαλισμού και αποστέλλει το frame στον παραλήπτη. Επομένως, με την χρήση της Υπολογιστικής Αποφόρτισης η κινητή συσκευή του χρήστη αποκτάει, έμμεσα, επιπλέον δυνατότητες χωρίς να καταναλώσει περισσότερη ενέργεια η κινητή του συσκευή.

3.5.4 Διαμοιρασμένο περιεχόμενο – Προσωρινή αποθήκευση

Μια βάση δεδομένων, εγκατεστημένη σε ένα Νέφος, είναι μια ευρύς υπηρεσία η οποία έχει πρόσβαση από πολλούς τελικούς χρήστες. Μια περίπτωση χρήσης είναι το περιεχόμενο της

βάσης να αποτελείται από βίντεο. Στην προκειμένη περίπτωση η πρόσβαση από πολλούς χρήστες, ταυτόχρονα, καθίσταται δυσχερείς λαμβάνοντας υπόψιν τις απαιτήσεις του βίντεο σε επεξεργασία και σε πόρους. Οι δυσχέρειες εντοπίζονται στην καθυστέρηση **(Latency)** αλλά και στο **Jitter.** Με τον όρο **Jitter** εννοούμε τις στιγμιαίες αποκλίσεις καθοριστικών τμημάτων ενός ψηφιακού σήματος σε σχέση με τις ιδανικές θέσεις τους στον χρόνο. Ο συνδυασμός των δυο παραπάνω μειώνει την **QoE** του τελικού χρήστη. Εγκαθιστώντας την βάση δεδομένων των σε έναν **MEC** κόμβο, ο τελικός χρήστης απολαμβάνει το περιεχόμενο των video έχοντας άριστη ποιότητα.

Κεφάλαιο 4
Face Recognition over Edge-Computing

4.1 Εισαγωγή

Η αναγνώριση αντικειμένων από τον άνθρωπο είναι μια συνήθης διαδικασία, σε καθημερινή βάση, για αυτόν. Εστιάζοντας ένα αντικείμενο είτε με την όραση του είτε με τις παλάμες των χεριών του, άμεσα, το αναγνωρίζει. Ακόμα και εάν το αντικείμενο είναι άγνωστο για αυτόν, μπορεί να το περιεργαστεί με τις αισθήσεις του και να καταλήξει στην πιο πιθανή ερμηνεία του. Η ικανότητα του ανθρώπου να αναγνωρίζει αντικείμενα ή πρόσωπα οφείλεται στην εκπαίδευση του εγκεφάλου του από την στιγμή της γέννησης του. Η εκπαίδευση επιτυγχάνεται από την καθημερινή εμπειρία, που αποκτάει, με τον έξω κόσμο.

Η διαδικασία αναγνώρισης ενός αντικειμένου από μια μηχανή είναι μια δύσκολη διαδικασία. Φανταστείτε ότι έχουμε έναν ρομποτικό βραχίονα του οποίου η εργασία είναι να σφηνώσει έναν άξονα μέσα σε έναν ένσφαιρο τριβέα. Για να επιτύχει η παραπάνω εργασία του ρομποτικού βραχίονα, θα πρέπει να εκτελέσει τα εξής βήματα:

(a) Θα πρέπει να αναγνωρίσει τον τύπο του τριβέα ανεξάρτητα του χωρικού του προσανατολισμού.
(b) Θα πρέπει να τον ανακαλύψει τον τριβέα ανάμεσα από πολλούς άλλους τριβείς ή επιμέρους εξαρτήματα.
(c) Θα πρέπει να υπολογίσει τις χωρικές συντεταγμένες του τριβέα έτσι ώστε να τον πιάσει.
(d) Θα πρέπει να αναγνωρίσει το εσωτερικό δαχτυλίδι του τριβέα για να τον σφηνώσει στον άξονα.
(e) Θα πρέπει να αναγνωρίσει τις συντεταγμένες του άκρου του άξονα και στην συνέχεια να σφηνώσει τον τριβέα.

Προκειμένου να εκτελέσει τα παραπάνω βήματα ο ρομποτικός βραχίονας θα πρέπει να του δοθεί ένα σύνολο δεδομένων έτσι ώστε να αναγνωρίζει τα επιμέρους αντικείμενα. Με την έννοια “αναγνώριση” εννοούμε την δημιουργία μοτίβων τα οποία, κατά κάποιον τρόπο, θα περιγράφουν το αντικείμενο. Επομένως, όντας γνωστά τα μοτίβα των επιμέρους

αντικειμένων στον ρομποτικό βραχίονα, ο τελευταίος τα αναγνωρίζει με επιτυχία και επιτυγχάνει την εργασία που του έχει ανατεθεί, δηλαδή το σφήνωμα του άξονα στον ένσφαιρου τριβέα.

Η διαδικασία δημιουργίας μοτίβων δεν περιορίζεται μόνο στα άψυχα αντικείμενα. Μοτίβα δύναται να δημιουργηθούν για την αναγνώριση της ανθρώπινης ομιλίας, γεωφυσικών χαρακτηριστικών και φυσικά για την αναγνώριση ανθρώπινων προσώπων.

4.2 Θεωρία

Η θεωρία αναγνώρισης μοτίβων χρονολογείται από το 1986 όταν ο εφευρέτης **Robert K. McConnell** εφηύρε μια ευρεσιτεχνία για την κάνει πράξη. Η αναγνώριση μοτίβων έχει την βάση της στην θεωρία της πληροφορίας. Αναλυτικά, θεωρούμε ότι έχουμε μια πηγή η οποία εκπέμπει ένα μήνυμα, μήκους **N** συμβόλων με **Q** διαφορετικά σύμβολα. Τότε η ολική πληροφορία που εκπέμπεται από την πηγή δίνεται από την μαθηματική σχέση:

$$I = NH = -N\sum_{i=1}^{Q} p_i \log_b(p_i)$$

όπου $\mathbf{p_i}$ είναι η πιθανότητα εμφάνισης του **ith** συμβόλου στο μήνυμα και **b** είναι η βάση των λογαρίθμων. Εάν **b=2** τότε η πληροφορία είναι κωδικοποιημένη με τα δυαδικά ψηφία 0 και 1. Η ποσότητα **H** ονομάζεται εντροπία ή μέση πληροφορία και ισούνται με:

$$H = -\sum_{i=1}^{Q} p_i \log_b(p_i)$$

και αποτελεί το βασικό μέγεθος αναγνώρισης μοτίβων.

Θεωρούμε ότι έχουμε δυο διαφορετικές πηγές μηνυμάτων οι οποίες παράγουν μηνύματα με πιθανότητα εμφάνισης $\mathbf{p_1}$ και $\mathbf{p_2}$ αντίστοιχα. Προκειμένου να αναγνωριστεί η πηγή κάποιου μηνύματος, αρχικά, παράγεται ένα μήνυμα ελέγχου το οποίο μήνυμα περιέχει σύμβολα με πιθανότητα **p** εμφάνισης συμβόλων. Μετά την παραγωγή του μηνύματος ελέγχου, γίνεται σύγκριση των πιθανοτήτων εμφάνισης των συμβόλων μεταξύ του μηνύματος ελέγχου και του μηνύματος κάθε πηγής. Η σύγκριση των μηνυμάτων θέτει τις εξής προϋποθέσεις:

(a) Τα σύμβολα με μεγαλύτερη συχνότητα εμφάνισης έχουν μεγαλύτερο βάρος. Η συνέπεια αυτής της προϋπόθεσης είναι ότι σε περίπτωση όπου σύμβολα που εμφανίζονται στο μήνυμα ελέγχου δεν εμφανίζονται στο μήνυμα της πηγής τότε δεν θα γίνεται καλό ταίριασμα μεταξύ των μηνυμάτων.

(b) Το μήνυμα ελέγχου θα πρέπει να έχει σχετικά μεγάλο μήκος, διαφορετικά, εάν το μήνυμα ελέγχου έχει σχετικά μικρό μήκος τότε δεν θα είναι ορθό το ταίριασμα των μηνυμάτων μεταξύ αυτού και της πηγής.

Οι παραπάνω προϋποθέσεις λαμβάνονται υπόψιν εάν τροποποιηθεί κατάλληλα η μαθηματική έκφραση της ολικής πληροφορίας. Έστω μαθηματική έκφραση της ολικής πληροφορίας:

$$I = -N \sum_{i=1}^{Q} p_i \log_b(p_i)$$

Θέτουμε $\mathbf{m_i = Np_i}$ και η έκφραση του αθροίσματος γίνεται ως εξής:

$$I(T:R) = -\sum_{i=1}^{Q} m_i \log_b(p_i)$$

όπου

$\mathbf{m_i}$ είναι το πλήθος εμφάνισης του **ith** συμβόλου στο μήνυμα ελέγχου,

$\mathbf{p_i}$ είναι η πιθανότητα εμφάνισης του **ith** συμβόλου στο μήνυμα πηγής.

Ομοίως, η μαθηματική σχέση μέσης πληροφορίας γίνεται ως εξής:

$$H(T:R) = \frac{-1}{M} \sum_{i=1}^{Q} m_i \log_b(p_i)$$

όπου

M είναι το πλήθος των συμβόλων στο μήνυμα ελέγχου και ισούνται με

$$M = \sum_{i=1}^{Q} m_i$$

Αξιοσημείωτο είναι το γεγονός ότι και στις δύο παραπάνω εξισώσεις υπάρχει ο όρος $\mathbf{m_i \log_b(p_i)}$ δηλαδή τα σύμβολα με τις περισσότερες εμφανίσεις θα έχουν μεγαλύτερο βάρος, επομένως, ικανοποιούν και τις δύο προϋποθέσεις που αναφέρθηκαν προηγουμένως.

Εύκολα γίνεται αντιληπτό το γεγονός η ταύτιση μεταξύ του μηνύματος ελέγχου και πηγής να μην είναι απόλυτη δηλαδή τα μηνύματα να μην ταιριάζουν απόλυτα. Για να γίνει κατανοητό, θεωρούνται οι εξισώσεις:

$$I(T:T) = -\sum_{i=1}^{Q} m_i \log_b(p_i)$$

και

$$H(T:T) = \frac{-1}{M} \sum_{i=1}^{Q} m_i \log_b(p_i)$$

όπου

m_i είναι το πλήθος εμφάνισης του **ith** συμβόλου στο μήνυμα ελέγχου,

p_i είναι η πιθανότητα εμφάνισης του **ith** συμβόλου στο μήνυμα ελέγχου.

Για να υπάρχει ταύτιση μεταξύ του μηνύματος ελέγχου και της πηγής θα πρέπει ισότητα:

$$\mathbf{D_1 = H(T:R) - H(T:T) \quad (1)}$$

να ισούνται με μηδέν. Το γεγονός η ισότητα **(1)** να ισούνται με μηδέν είναι σχετικά απίθανο, θεωρείται ότι όσο η ισότητα, $\mathbf{D_1}$, τείνει προς το μηδέν τόσο η ταύτιση των μηνυμάτων μεταξύ τους είναι μεγαλύτερη.

Ακόμα μια ισότητα που εισάγεται είναι η:

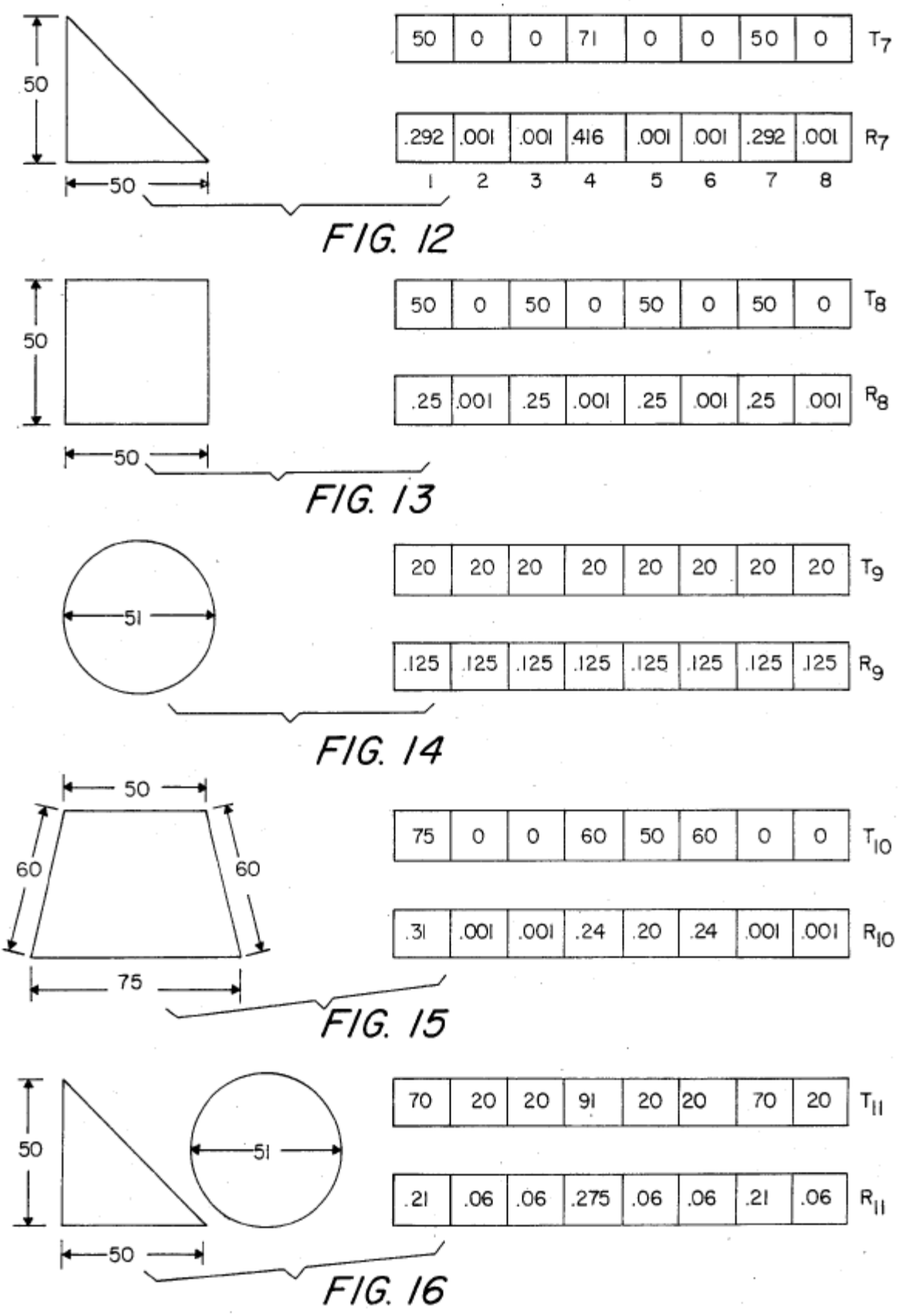
$$\mathbf{D_2 = I(T:R) - I(T:T) \quad (2)}$$

όπου και αυτή με την σειρά της αποτελεί μέτρο σύγκρισης των μηνυμάτων καθώς εκφράζει την σύγκριση της ολικής πληροφορίας και όχι της μέσης πληροφορίας όπως γινόταν με την ισότητα **(1)**.

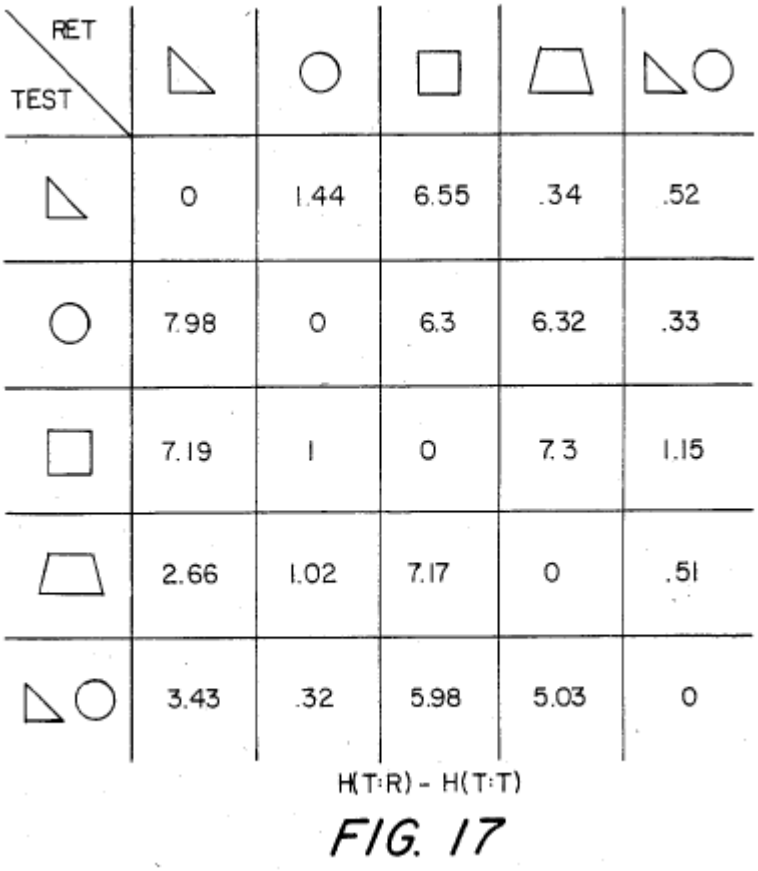
Οι παραπάνω εξισώσεις και ισότητες αποτελούν τον προπομπό της δημιουργίας μοτίβων και εν τέλη την αναγνώριση αντικειμένων και προσώπων. Ο τρόπος που γίνεται η αναγνώριση, σύμφωνα με τις παραπάνω μεθόδους, είναι η δημιουργία ενός μοτίβου αναφοράς το οποίο θα περιγράφει το αντικείμενο, όπως ακριβώς η πηγή παράγει μηνύματα πιθανότητας **p**. Εν συνεχεία, κάθε νέο αντικείμενο, το οποίο δεν έχει αναγνωριστεί, θεωρείται ως μήνυμα ελέγχου δηλαδή θα παράγεται ένα μοτίβο το οποίο θα το αντιπροσωπεύει. Εάν οι ισότητες **(1)** και **(2)** τείνουν προς το μηδέν σημαίνει ότι το μοτίβο ελέγχου ταυτίζεται με το μοτίβο αναφοράς δηλαδή το αντικείμενο αναγνωρίστηκε με επιτυχία.

Αποδεικνύεται ότι τα μοτίβα ελέγχου και αναφοράς μπορούν να εκφραστούν με ιστογράμματα. Σε κάθε κελί του ιστογράμματος αποθηκεύεται το σύμβολο του μηνύματος

και η συχνότητα που αναπαριστά το ιστόγραμμα ισούνται με το βάρος του αντίστοιχου συμβόλου (Robert K. McConnell:14).



***Εικόνα 8**: Ιστογράμματα διαφόρων σχημάτων (Πηγή: Robert K. McConnell:7)*



***Εικόνα 9**: Συγκρίσεις ισοτήτων (Πηγή: Robert K. McConnell:8)*

4.3 Ιστόγραμμα κατευθυνόμενων διανυσμάτων (Histogram Oriented Gradients - HOG)

Η ευρεσιτεχνία του **Robert K. McConnell** ήταν πρωτοποριακή και στα επόμενα έτη αναπτύχθηκαν πολλοί αλγόριθμοι αναγνώρισης μοτίβων. Μια εφαρμογή της ευρεσιτεχνίας του **Robert K. McConnell** αποτελεί η εργασία των **William T. Freeman** και **Michal Roth** όπου γίνεται χρήση των ιστογραμμάτων κατευθυνόμενων διανυσμάτων για την αναγνώριση μοτίβων χεριών.

Το 2005 οι ερευνητές **Navneet Dalal** και **Bill Triggs** ανέπτυξαν έναν αλγόριθμο ο οποίος αναγνώριζε τους πεζούς μέσα από στατικές εικόνες. Ο ίδιος αλγόριθμος αναπτύχθηκε και εφαρμόζεται σε κινούμενες εικόνες και κυρίως στην αναγνώριση προσώπων. Η τεχνική αναγνώρισης βασίζεται στον αριθμό των εμφανίσεων των κατευθυνόμενων διανυσμάτων μέσα σε ένα τμήμα μιας εικόνας. Επιπροσθέτως, η τεχνική αναγνώρισης εφαρμόζεται σε πυκνό πλέγμα εικονοστοιχείων, τα οποία ισαπέχουν μεταξύ τους και για μεγαλύτερη

ακρίβεια χρησιμοποιεί κανονικοποίηση επικάλυψης τοπικών αντιθέσεων **(Overlapping local contract normalization)** (Navneet:1).

4.3.1 Feature Descriptor

Ως **Feature Descriptor** είναι ένα σύνολο αλγορίθμων οι οποίοι παράγουν σε μια εικόνα περιγραφές αντικειμένων. Υπάρχουν δυο κατηγορίες των **Feature Descriptors**.

(a) **General Information Descriptors**. Εξάγουν χαμηλού επιπέδου περιγραφή αντικειμένου, όπως το σχήμα, η υφή, το χρώμα και η τοποθεσία. Ειδικά, στην περίπτωση που εξάγεται ένα σχήμα ενός αντικειμένου δύναται να ακολουθεί και η περιγραφή του όπως ακριβώς συμβαίνει με την ανθρώπινη αντίληψη, δηλαδή, ένας άνθρωπος αναγνωρίζει ένα αντικείμενο μόνο και μόνο από το σχήμα του.

(b) **Specific domain information descriptors**. Αποτελούν ένα σύνολο αλγορίθμων οι οποίοι εξάγουν αντικείμενα τα οποία δεν είναι ευκόλως εντοπίσιμα. Σε αυτή την κατηγορία ανήκουν οι αλγόριθμοι αναγνώρισης προσώπου.

Εφαρμόζοντας τους κατάλληλους αλγορίθμους πάνω σε μια εικόνα εξάγουμε το επιθυμητό **Feature Descriptor** δηλαδή προβαίνουμε στην αναγνώριση του επιθυμητού αντικειμένου. Σε αυτό το στάδιο εμφανίζονται στην εικόνα οι επιθυμητές πληροφορίες που αναφέρονται μόνο στο επιθυμητό αντικείμενο. Ο αλγόριθμος που θα εφαρμοστεί είναι ο υπολογισμός των **Histogram Oriented Gradients**, των ερευνητών **Navneet Dalal** και **Bill Triggs** (Navneet:1).

4.3.2 Υπολογισμός των Histogram Oriented Gradients

Ο υπολογισμός των **Histogram Oriented Gradients** παρουσιάζει πλεονεκτήματα σε σχέση με άλλους και έτσι καθίσταται κατάλληλος για την ανίχνευση προσώπων. Συγκεκριμένα, ο αλγόριθμος επεξεργάζεται μια ομάδα από εικονοστοιχεία, είναι ανεξάρτητος από την γεωμετρία αλλά εξαρτάται από τον προσανατολισμό του κάθε αντικειμένου. Ωστόσο, ο προσανατολισμός δύναται να διορθωθεί με κατάλληλους αλγορίθμους.

***Εικόνα 10**: Επιλογή περιοχής και προσαρμογή (Πηγή: OpenCV)*

(a) **Υπολογισμός Διανυσμάτων.** Το στάδιο αυτό περιλαμβάνει τον υπολογισμό του μεγέθους και της κλίσης των διανυσμάτων. Αρχικά, επιλέγουμε την περιοχή της εικόνας που μας ενδιαφέρει και αλλάζουμε το μέγεθος της περιοχής αυτής σε 64x128 εικονοστοιχεία.

Εφαρμόζουμε στην περιοχή το φίλτρο βάθμωσης **(Gradient Filter) [-1,0,1]** και **$[-1,0,1]^T$**. Έτσι προκύπτει η εικόνα με τα διανύσματα με κατεύθυνση στους άξονες **xx'** και **yy'**. Στην συνέχεια υπολογίζουμε το μέτρο και την κλίση τους. Το μέτρο κάθε διανύσματος και η γωνία δίνεται από την σχέση:

$$G=(Gx^2+Gy^2)^{1/2}$$
$$\theta=\arctan(Gy/Gx)$$

Κατόπιν εφαρμογής του φίλτρου βάθμωσης, η εικόνα έχει την παρακάτω μορφή:

***Εικόνα 11**: Κατευθυνόμενα διανύσματα στου άξονες xx, yy και συνισταμένη (Πηγή: OpenCV)*

Η Εικόνα 11 απεικονίζει την αρχική εικόνα κατόπιν εφαρμογής των φίλτρων βάθμωσης **(Gradient Filter)**. Αξιοσημείωτο είναι το γεγονός ότι στις γωνίες και στις ακμές το μέτρο των διανυσμάτων έχει μεγαλύτερη τιμή σε σχέση με το μέτρο των διανυσμάτων στις επίπεδες επιφάνειες. Εξ ου και η επιτυχής εφαρμογή του αλγορίθμου στην ανίχνευση προσώπου (Navneet:3).

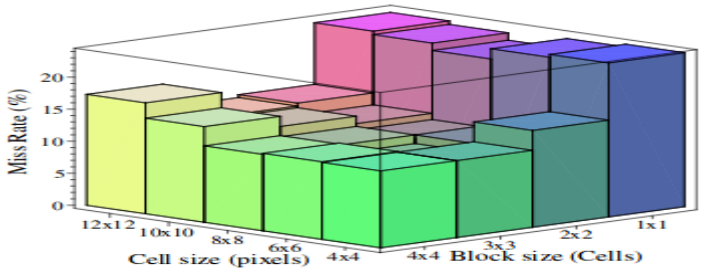
Να σημειωθεί ότι οι **Navneet Dalal** και **Bill Triggs** εφάρμοσαν διάφορα επιμέρους φίλτρα αλλά τα αποτελέσματα δεν ήταν τα αναμενόμενα .

(b) Χωρίζουμε την αρχική εικόνα σε πλέγμα. Κάθε κελί του πλέγματος είναι διαστάσεων **8x8** εικονοστοιχείων. Επομένως θα προκύψουν 8x8x3=192 τιμές. Λαμβάνοντας υπόψιν το μέτρο και την κλίση των διανυσμάτων, τα οποία ήδη έχουμε υπολογίσει προηγουμένως, θα προκύψει ένας πίνακας-διάνυσμα όπου σε 64 θέσεις θα αποθηκευτεί το μέτρο των διανυσμάτων και σε επιπλέον 64 θέσεις θα αποθηκευτεί η κλήση κάθε διανύσματος. Άρα, θα προκύψει ένα διάνυσμα 128 θέσεων. Σύμφωνα με τους **Navneet Dalal** και **Bill Triggs** το κελί διαστάσεων 8x8 εικονοστοιχεία θεωρείται η βέλτιστη επιλογή γιατί δεν επηρεάζεται από τυχόν παρουσία θορύβου.

Έχοντας δημιουργήσει το διάνυσμα 128 θέσεων δημιουργούνται τα ιστογράμματα όπου κάθε ιστόγραμμα αντιπροσωπεύει ένα εύρος γωνιών ή κλίσεων. Το πρώτο ιστόγραμμα αντιπροσωπεύει το εύρος γωνιών 0^o-20^o, το δεύτερο ιστόγραμμα αντιπροσωπεύει το εύρος γωνιών 20^o-40^o κοκ. Άρα, δημιουργούνται συνολικά 9 ιστογράμματα. Το βάρος κάθε ιστογράμματος θα ισούνται με το άθροισμα των μέτρων των διανυσμάτων (Navneet:6).

(c) Υπάρχουν περιπτώσεις όπου έχουμε εσφαλμένο αποτέλεσμα λόγω έντονου τοπικού φωτισμού σε μια εικόνα ή λόγω αντίθεσης. Προκειμένου να ξεπεράσουμε αυτή την δυσχέρεια, θεωρείται αναγκαίο να γίνει κανονικοποίηση **(Normalization)**.

Οι Navneet Dalal και Bill Triggs υπολόγισαν ότι ένα κελί με διαστάσεις 6x6 εικονοστοιχεία και ομαδοποίηση των 3x3 κελιών δηλαδή 18x18 εικονοστοιχεία, έχει τις ελάχιστες απώλειες, δηλαδή ο ρυθμός άστοχης αναγνώρισης είναι ο ελάχιστος σε σχέση με επιμέρους ομαδοποιήσεις.



Εικόνα 12: Ομαδοποίηση κελιών. Η 3x3 ομαδοποίηση έχει τις ελάχιστες απώλειες. (Πηγή: Navneet : 5)

Οι **Navneet Dalal** και **Bill Triggs** αξιολόγησαν τέσσερις μεθόδους κανονικοποίησης. Έστω μη κανονικοποιημένο διάνυσμα, **v**, **||v||** η νόρμα του διανύσματος με **k=1,2** και επιπλέον μια μικρή σταθερά, **e**. Τα σχήματα κανονικοποίησης είναι τα εξής:

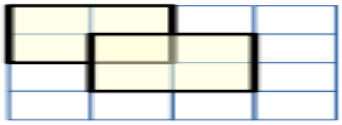
$$L_{2norm} = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L_{2hys}: Όπως ακριβώς η **L_{2norm}** αλλά με ψαλιδισμό του **v**, δηλαδή περιορίζοντας τις μέγιστες τιμές του **v** στο 0,2 και εκτελώντας ξανά την κανονικοποίηση.

$$L_{1\,norm} = \frac{v}{\|v\|_1 + e}$$

$$L_{1\,sqr} = \sqrt{\frac{v}{\|v\|_1 + e}}$$

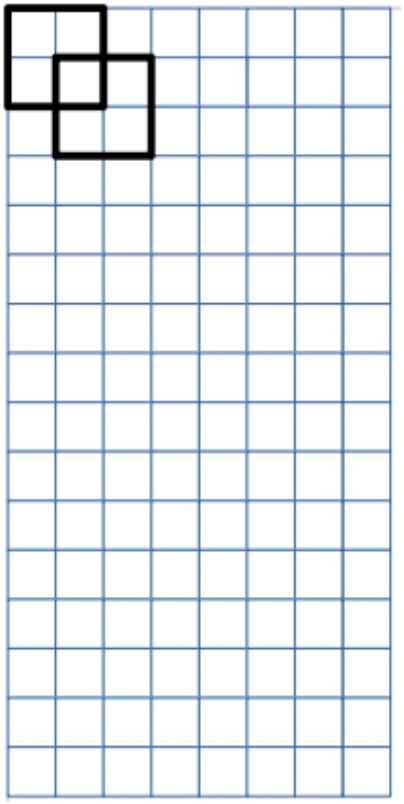
Οι **Navneet Dalal** και **Bill Triggs** απέδειξαν ότι οι μέθοδοι **L_{2norm}**, **L_{2hys}**, και **L_{1sqrt}** αποδίδουν εξίσου καλά αποτελέσματα, η μέθοδος **L_{1norm}** μειώνει την απόδοση κατά 5% περίπου ενώ εάν δεν γίνει κανονικοποίηση τότε η απόδοση μειώνεται κατά 27%. Έχοντας ομαδοποιήσει τα κελιά, στην συνέχεια πραγματοποιείται η κανονικοποίηση σε όλα τα εικονοστοιχεία της εικόνας. Θα πρέπει να επισημανθεί ότι κατά την κανονικοποίηση γίνεται επικάλυψη των ομαδοποιημένων εικονοστοιχείων. Το γεγονός αυτό αποδεικνύεται ιδιαίτερα χρήσιμο γιατί με αυτή την μέθοδο συνεισφέρουν όλα τα εικονοστοιχεία στην κανονικοποίηση.



Εικόνα 13: Κανονικοποίηση εικονοστοιχείων. (Πηγή: Navneet : 6)

Η Εικόνα 13 απεικονίζει την κανονικοποίηση. Σε αυτή την περίπτωση για την κανονικοποίηση ομαδοποιήθηκαν 2x2 κελιά όπου κάθε κελί έχει διαστάσεις 8x8 εικονοστοιχεία. Η επικάλυψη των κελιών είναι εμφανής (Πηγή: Navneet : 6).

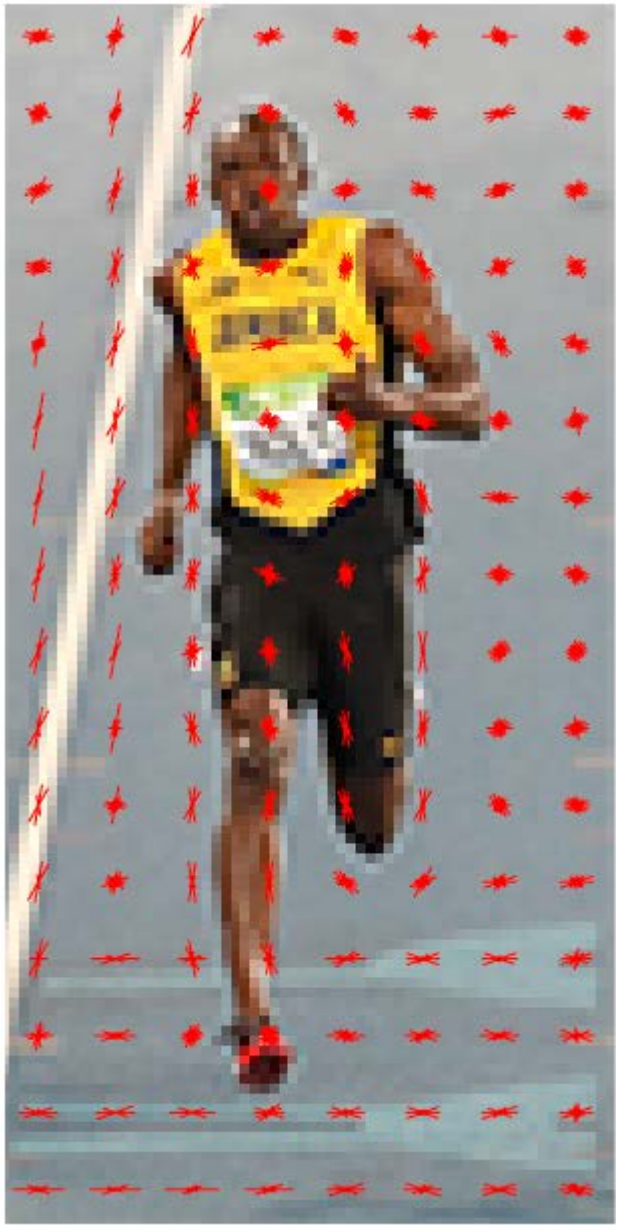
(d) Σε αυτό το στάδιο θα απεικονιστούν στην εικόνα τα κατευθυνόμενα διανύσματα. Έχοντας χωρίσει την αρχική εικόνα, διαστάσεων 64x128 εικονοστοιχείων, σε τμήματα των 8x8 εικονοστοιχείων, ομαδοποιούμε τα τμήματα σε block των 2x2 κελιών. Κάθε κελί θα έχει διάσταση 8x8 εικονοστοιχεία, επομένως, το block των 2x2 κελιών θα έχει διάσταση 16x16 εικονοστοιχεία. Κατά το πέρασμα του block σε όλη την εικόνα θα διανύσει 7 οριζόντιες και 15 κάθετες θέσεις επειδή θα κάνει επικάλυψη.



***Εικόνα 14**: Πλέγμα εικόνας 64x128 εικονοστοιχείων. Block 2x2 κελιών δηλαδή 16x16 εικονοστοιχείων*

Επομένως, το block θα διανύσει συνολικά 7x15=105 θέσεις. Κάθε κελί εξάγει 9 ιστογράμματα και κάθε block 9x4=36 ιστογράμματα. Άρα, όταν το block έχει διανύσει όλη την εικόνα θα έχει παράξει 36x105=3780 ιστογράμματα.

Μετά το πέρας της διαδικασίας υπολογισμού των ιστογραμμάτων, απεικονίζουμε στην εικόνα τα κατευθυνόμενα διανύσματα για κάθε κελί (8x8 εικονοστοιχεία). Τα διανύσματα με το μεγαλύτερο μέτρο ακολουθούν τις γραμμές του σώματος και του προσώπου.



***Εικόνα 15**: Απεικόνιση των HOGs στην αρχική εικόνα. Αξιοσημείωτο είναι ότι τα διανύσματα με το μεγαλύτερο μέτρο ακολουθούν τις γραμμές του σώματος και του προσώπου (Πηγή: OpenCV)*

4.4 Εκπαίδευση-Αναγνώριση

Η αναγνώριση προσώπου από μια εικόνα είναι μια διαδικασία η οποία αποτελείται από δύο στάδια, δηλαδή, το στάδιο του εντοπισμού του προσώπου και το στάδιο της αναγνώρισης. Τόσο στο στάδιο του εντοπισμού του προσώπου όσο και στο στάδιο της αναγνώρισης εφαρμόζεται ο αλγόριθμος των **Histogram Oriented Gradients**.

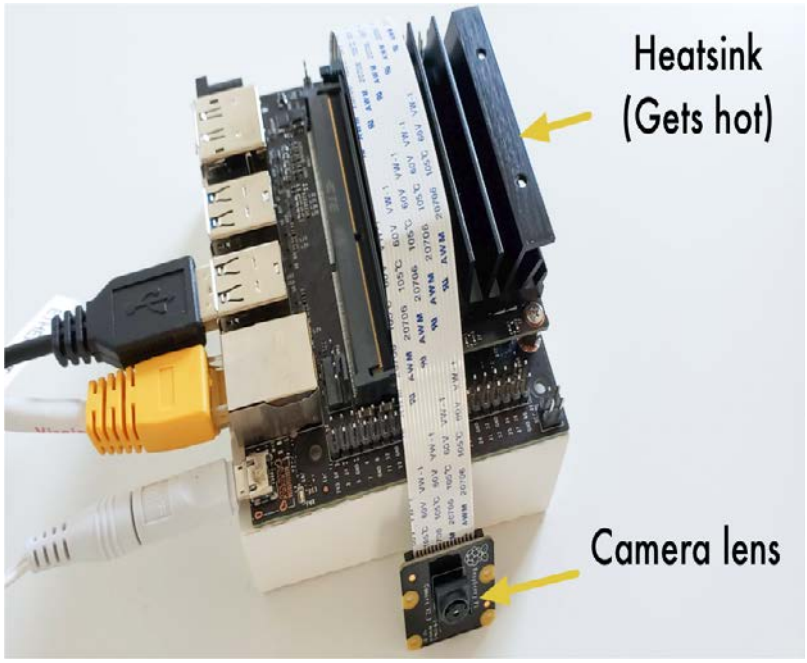
Ωστόσο, προτού χρησιμοποιηθεί η διαδικασία του εντοπισμού και αναγνώρισης θα πρέπει να έχουμε ήδη μια βάση δεδομένων στην οποία θα έχουμε ήδη αποθηκεύσει χαρακτηριστικά προσώπων. Με άλλα λόγια, εάν έχουμε ένα σύστημα που θα αναλάβει να αναγνωρίζει πρόσωπα, απαραίτητη προϋπόθεση είναι η αρχική εκπαίδευση **(Training)** του όλου συστήματος.

Η διαδικασία εκπαίδευσης είναι να θέσουμε το σύστημα να αναγνωρίσει πρόσωπα από εικόνες γνωστών προσώπων. Επομένως, θα δημιουργηθεί μια βάση δεδομένων στην οποία θα αποθηκεύονται τα ονόματα των προσώπων με τα αντίστοιχα διανύσματα των 128 θέσεων που δημιουργήθηκαν κατά την διαδικασία του υπολογισμού των **Histogram Oriented Gradients**. Η διαδικασία εκπαίδευσης μελετάται αναλυτικά στο επόμενο κεφάλαιο.

4.5 A Use Case of Face Recognition over Edge-Computing

Η παρούσα διατριβή είναι μια περίπτωση χρήσης αναγνώρισης προσώπου από έναν **MEC** κόμβο. Η πλατφόρμα του κόμβου φιλοξενεί μια εφαρμογή η οποία με την χρήση του αλγορίθμου των **Navneet Dalal** και **Bill Triggs** εντοπίζει και αναγνωρίζει πρόσωπα από μια εικόνα. Μια κάμερα η οποία είναι συνδεδεμένη στον υλικό της πλατφόρμας, μέσω κατάλληλης θύρας, επιτηρεί έναν χώρο. Η κάμερα διαβιβάζει στην πλατφόρμα κάθε πλαίσιο που καταγράφει και εν συνεχεία, η εφαρμογή προβαίνει στις δέουσες ενέργειες αναγνώρισης. Επίσης, η πλατφόρμα είναι συνδεδεμένη μέσω δικτύου με Νέφος για περαιτέρω ανάκτηση δεδομένων. Η παρούσα διαδικασία περιγράφεται αναλυτικά στο επόμενο Κεφάλαιο.

Μια εναλλακτική αρχιτεκτονική παρουσιάστηκε από την εταιρεία **NVIDIA** όπου πρόκειται για την πλατφόρμα **Jetson Nano**



Εικόνα 16: NVIDIA Jetson Nano

Η **NVIDIA Jetson Nano** είναι μια πλατφόρμα ανάπτυξης εφαρμογών. Στην Εικόνα 16 απεικονίζεται το υλικό πλήρως συνδεδεμένο με την τροφοδοσία εναλλασσόμενου ρεύματος, το τοπικό δίκτυο **802.3x,** το USB πληκτρολόγιο και την κάμερα. Η αποθήκευση υποστηρίζεται από μια **microSD** κάρτα. Αν και θυμίζει **Raspberry Pi** όσον αφορά το θέμα της αρχιτεκτονικής, ωστόσο, η συγκεκριμένη συσκευή έχει επεξεργαστή κάρτας γραφικών **(GPU)** ειδικά σχεδιασμένη να εκτελεί αλγορίθμους Deep Learning. Επίσης, υποστηρίζει βιβλιοθήκες CUDA και η πλατφόρμα της είναι βασισμένη στην γλώσσα **Python**. Ο συνδυασμός των παραπάνω χαρακτηριστικών την καθιστά ιδανική για εφαρμογές αναγνώρισης προσώπου με χρήση αλγορίθμων Νευρωνικών Δικτύων.

Η παραπάνω πλατφόρμα που απεικονίζεται στην Εικόνα 16 θεωρείται μια περίπτωση χρήσης αναγνώρισης προσώπου. Η όλη χρήση αλγορίθμων γίνεται τοπικά δηλαδή αποκλειστικά στην **NVIDIA Jetson Nano**. Επομένως μπορούμε να αποθηκεύσουμε μια τοπική βάση δεδομένων στην **microSD** κάρτα της πλατφόρμας και στην περίπτωση όπου η αναγνώριση αποτύχει θα στείλει ερώτημα σε κάποιον κόμβο του Νέφους μέσω του καλωδίου δικτύωσης. Το αποτέλεσμα της αναγνώρισης θα προβληθεί σε οθόνη που θα είναι συνδεδεμένη στην παρούσα πλατφόρμα μέσω καλωδίου micro HDMI.

Κεφάλαιο 5
Πειραματική διαδικασία

5.1 Εισαγωγή

Το ερευνητικό ερώτημα που καλούμαστε να απαντήσουμε είναι η μελέτη της ρυθμαπόδοσης μιας εφαρμογής επιτήρησης η οποία είναι εγκατεστημένη σε ήδη υπάρχουσα **MEC** αρχιτεκτονική. Με την έννοια ρυθμαπόδοση εννοούμε

(a) τον υπολογισμό του χρόνου απόκρισης της εφαρμογής από τον **MEC** κόμβο,

(b) τον υπολογισμό του χρόνου απόκρισης της εφαρμογής από το Νέφος,

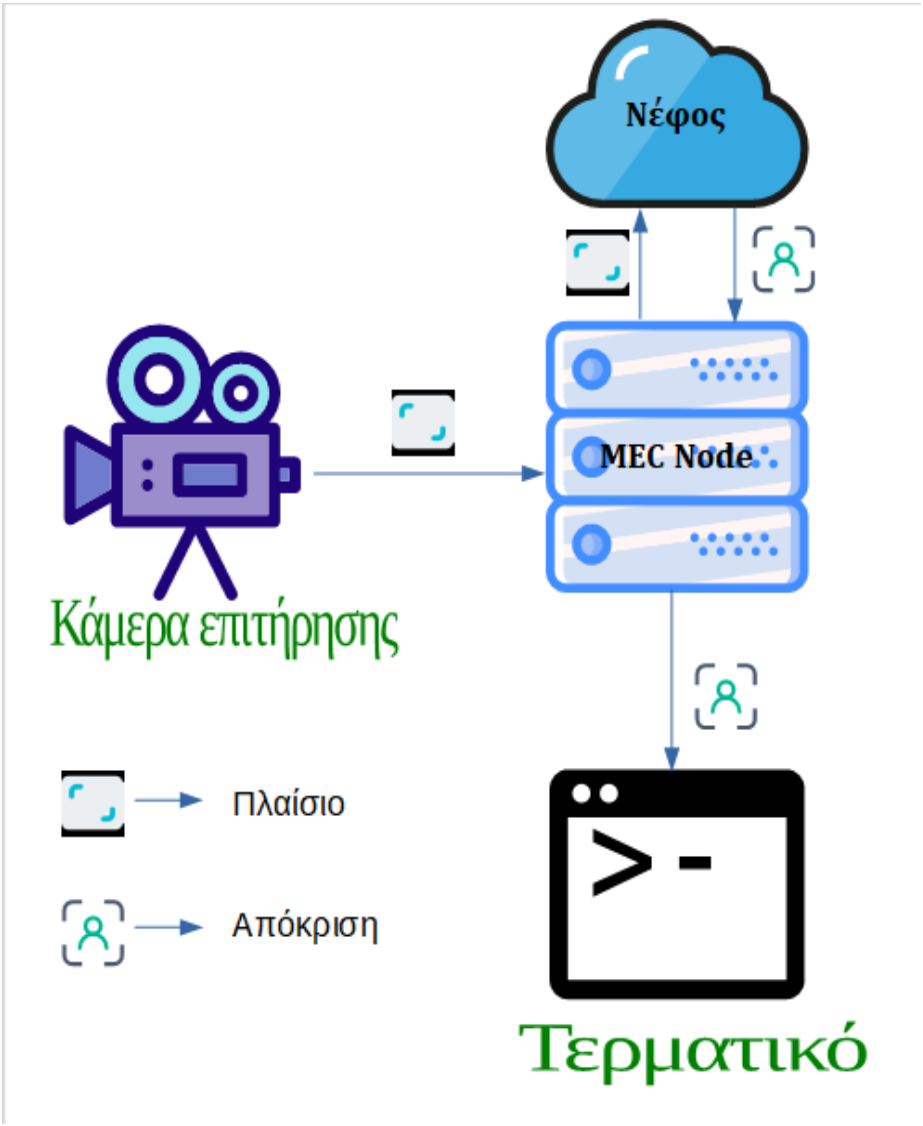
(c) το μέγεθος της τοπικής μνήμης cache του **MEC** κόμβου έτσι ώστε να μειώσουμε την κίνηση του δικτύου από τον τοπικό **MEC** κόμβο προς το Νέφος.

Τα παραπάνω ερευνητικά ερωτήματα θα μελετηθούν και θα απαντηθούν εφαρμόζοντας δύο (2) διαφορετικούς αλγορίθμους ενημέρωσης της τοπικής cache μνήμης του **MEC** κόμβου. Στο τέλος θα γίνει σύγκριση των αποτελεσμάτων ως προς της ρυθμαπόδοση για κάθε αλγόριθμο.

5.2 Περιγραφή πειράματος

Θεωρούμε ότι έχουμε εγκαταστήσει μια κάμερα επιτήρησης σε έναν χώρο η οποία είναι συνδεδεμένη και προσαρμοσμένη στον τοπικό **MEC** κόμβο. Ο σκοπός της κάμερας επιτήρησης είναι η αναγνώριση ανθρώπων. Η κάμερα καταγράφει έναν αριθμό από πλαίσια **(Frames)** ανά δευτερόλεπτο. Υποθέτουμε, την χρονική στιγμή t=0, η κάμερα καταγράφει κάποιον άνθρωπο ο οποίος τυγχάνει να διασχίζει τον χώρο που επιτηρεί. Τότε, η κάμερα θα διαβιβάσει το πλαίσιο στον τοπικό **MEC** κόμβο. Εν συνεχεία η εφαρμογή που είναι εγκατεστημένη στον τοπικό MEC κόμβο θα προβεί στην εξαγωγή των χαρακτηριστικών του προσώπου του ανθρώπου που απεικονίζεται στο πλαίσιο και με βάση αυτά τα χαρακτηριστικά, η εφαρμογή θα προβεί στην αναγνώριση του δηλαδή θα αναζητήσει το όνομα του. Η εξαγωγή του προσώπου και η ταυτοποίηση του γίνεται με τον αλγόριθμο των **Histogram Oriented Gradients**.

Εάν ο άνθρωπος δεν αναγνωρισθεί επιτυχώς από τον **MEC** κόμβο τότε η εφαρμογή θα αναζητήσει το όνομα του ανθρώπου από το Νέφος. Εάν ο άνθρωπος δεν αναγνωρισθεί, επιτυχώς, από το Νέφος τότε το Νέφος θα επιστρέψει την λέξη "**Unknown**".



***Εικόνα 17**: Πειραματική διαδικασία*

Η απόκριση του συστήματος θα είναι το πλαίσιο που διαβίβασε η κάμερα επιτήρησης και θα απεικονίζει τον άνθρωπο με την ετικέτα του ονόματος του.

5.3 Αναλυτική Περιγραφή – Κώδικας

Θα κατασκευαστούν τέσσερις (4) εφαρμογές σε γλώσσα **Python** με ονόματα **Training.py**, **Cloud-OUC.py**, **MEC-CacheLFU-OUC.py** και **MEC-CacheTimer-OUC.py**.

Η πρώτη εφαρμογή ονομάζεται **Training.py**. Κατασκευάστηκε με σκοπό την εκπαίδευση του Νέφους δηλαδή την δημιουργία των διανυσμάτων 128 θέσεων από ένα σύνολο φωτογραφιών όπου σε κάθε φωτογραφία απεικονίζει ένα πρόσωπο και όχι άλλο αντικείμενο όπως ένα τοπίο. Για λόγους νομιμότητας δεν χρησιμοποιηθήκαν φωτογραφίες με άγνωστα πρόσωπα. Για την εκπαίδευση του Νέφους χρησιμοποιήθηκαν φωτογραφίες από επώνυμους

καλλιτέχνες οι οποίοι ηθελημένα δημοσιεύουν φωτογραφίες σε γνωστά κοινωνικά μέσα δικτύωσης ή σε μέσα μαζικής ενημέρωσης όπως διάφορα ηλεκτρονικά περιοδικά. Τα αποτελέσματα της κωδικοποίησης εμφανίζονται σε ένα τερματικό. Ο κώδικας της εφαρμογής **Training.py** βρίσκεται στο **Παράρτημα Α**.

Η δεύτερη εφαρμογή θα εξομοιώνει το Νέφος και το όνομα αυτής είναι **Cloud-OUC.py**. Επικοινωνεί με τις εφαρμογές **MEC-CacheLFU-OUC.py** και **MEC-CacheTimer-OUC.py** όπου οι τελευταίες εξομοιώνουν τους **MEC** κόμβους. Αναλυτικά, η εφαρμογή **Cloud-OUC.py** θα λάβει το ερώτημα **(Query)** από τον **MEC** κόμβο σε μορφή συνάρτησης. Το όρισμα της συνάρτησης θα είμαι το διάνυσμα των 128 θέσεων το οποίο παράχθηκε από τον αλγόριθμο των **Histogram Oriented Gradients**. Εάν η αναγνώριση δεν καταστεί επιτυχής τότε η εφαρμογή, **Cloud-OUC.py**, επιστρέφει **"Unknown"**. Εάν η αναγνώριση καταστεί επιτυχής τότε η εφαρμογή **Cloud-OUC.py** θα επιστρέψει ένα λεξικό με το κλειδί **"names"** που θα περιέχει την λίστα με το όνομα του προσώπου και το κλειδί **"encodings"** με την αντίστοιχη λίστα των διανυσμάτων των 128 θέσεων. Η διασύνδεση μεταξύ της εφαρμογής **Cloud-OUC.py** και των **MEC-CacheLFU-OUC.py**, **MEC-CacheTimer-OUC.py** γίνεται με χρήση **SOCKETS** χρησιμοποιώντας ένα **RTT (Round-Trip-Time)** περί τα 300ms.

Ο κώδικας της εφαρμογής **Cloud-OUC.py** βρίσκεται στο **Παράρτημα Β**.

Η τρίτη και η τέταρτη εφαρμογή είναι συνυφασμένες οι οποίες έχουν σκοπό να εξομοιώσουν τον **MEC** κόμβο. Τα ονόματα αυτών είναι **MEC-CacheLFU-OUC.py** και **MEC-CacheTimer-OUC.py**. Υποτίθεται ότι στον **MEC** κόμβο διαβιβάζεται ένα πλαίσιο **(Frame)** από την κάμερα επιτήρησης. Ως γνωστών η κάμερα επιτήρησης παράγει μια ροή εικόνων ή ένα **stream**. Επειδή για λόγους νομιμότητας δεν δύναται να εγκαταστήσουμε μια κάμερα επιτήρησης σε δημόσιο χώρο και να καταγράφουμε πρόσωπα χωρίς την απαιτούμενη άδεια, δημιουργήσαμε ένα video clip ή **stream**, διάρκειας 40 λεπτών περίπου στο οποίο περιέχονται τα πρόσωπα τα οποία έχουμε ήδη κωδικοποίησει και αποθηκεύσει στο Νέφος. Επομένως, ο **MEC** κόμβος θα προβεί στον εντοπισμό και αναγνώριση του προσώπου από κάθε πλαίσιο τους video clip. Εάν το πρόσωπο που απεικονίζεται στο πλαίσιο του video clip δεν αναγνωριστεί με επιτυχία τότε γίνεται ερώτημα **(Query)** για αναγνώριση προς το Νέφος. Εάν το πρόσωπο αναγνωριστεί με επιτυχία τότε θα εμφανίσει το αποτέλεσμα σε ένα τερματικό, ο **MEC** κόμβος θα ενημερώσει την βάση δεδομένων και την cache μνήμη του. Τα αποτελέσματα της αναγνώρισης είτε από το Νέφος είτε από τον ίδιο **MEC** κόμβο εμφανίζονται σε ένα τερματικό. Στο ίδιο τερματικό εμφανίζονται κάθε φορά τα περιεχόμενα της μνήμης cache και η ενημέρωση αυτής. Σε περίπτωση όπου κάποιο στοιχείο, της μνήμης cache, αποβληθεί σύμφωνα με τον εκάστοτε

αλγόριθμο, τότε θα εμφανίζεται και το στοιχείο που αποβλήθηκε. Επιπροσθέτως, στην οθόνη του τερματικού θα εμφανίζεται ο χρόνος αναγνώρισης του προσώπου εάν η αναγνώριση ήταν επιτυχής στον **MEC** κόμβο ή ο χρόνος αναγνώρισης του προσώπου εάν η αναγνώριση ήταν επιτυχής στο Νέφος. Αν και οι εφαρμογές είναι συνυφασμένες, ωστόσο, διαφέρουν στην εφαρμογή του αλγορίθμου ενημέρωσης της μνήμης. Συγκεκριμένα, η εφαρμογή **MEC-CacheLFU-OUC.py** αναβαθμίζει την μνήμη cache σύμφωνα με τον αλγόριθμο **LFU** ενώ η εφαρμογή **MEC-CacheTimer-OUC.py** αναβαθμίζει την μνήμη cache σύμφωνα με τον αλγόριθμο Χρονοκαθυστέρησης**.** Θα πρέπει να επισημανθεί ότι το αρχικό μέγεθος της μνήμης και της βάσης δεδομένων είναι μηδενικά ανεξάρτητα την μέγιστη χωρητικότητα της μνήμης cache. Ο κώδικας της εφαρμογής **MEC-CacheLFU-OUC.py** βρίσκεται στο **Παράρτημα Γ** ενώ ο αντίστοιχος κώδικας της εφαρμογής **MEC-CacheTimer-OUC.py** βρίσκεται στο **Παράρτημα Δ**.

Η εξομοίωση του παραπάνω συστήματος έγινε σε ηλεκτρονικό υπολογιστή με **CPU i7**, **16GB RAM, NVIDIA Graphics Card.**

5.3.1 Βιβλιοθήκες, Συναρτήσεις

Το λειτουργικό σύστημα που έγινε η εξομοίωση ήταν το Linux Ubuntu 16.04. Η γλώσσα προγραμματισμού που εγκαταστάθηκε και κατασκευάστηκαν οι εφαρμογές ήταν η **Python 3.5.2.**

```
(cv) zouglisnik@zouglisnik-Lenovo-B70-80:~$ python
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Εικόνα 18: *Python 3.5.2*

Στο τερματικό εγκαταστάθηκε η βιβλιοθήκη **OpenCV** η οποία παρέχει ένα σύνολο συναρτήσεων που αφορούν την εικόνα και όραση μέσω ηλεκτρονικού υπολογιστή.

Επιπρόσθετα, εγκαταστάθηκε η βιβλιοθήκη **face_recognition** η οποία παρέχει ένα σύνολο συναρτήσεων που αφορούν την αναγνώριση και διαχείριση προσώπου σε ένα πλαίσιο εικόνας.

Τέλος, εγκαταστάθηκε η βιβλιοθήκη **dlib** η οποία παρέχει ένα σύνολο συναρτήσεων που αφορούν το **Deep Learning** και την διαχείριση των κατευθυνόμενων διανυσμάτων.

5.3.1.1 OpenCV

Από την βιβλιοθήκη **OpenCV** χρησιμοποιήθηκαν οι συναρτήσεις:

cv2.VideoCapture(query)

Η συνάρτηση **cv2.VideoCapture(query)** διαβάζει κάθε πλαίσιο **(Frame)** από το το αρχείο που δίνεται από την όρισμα **query.**

cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

Η συνάρτηση **cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)** εναλλάσσει σε κάθε πλαίσιο **(Frame)** τον χώρο χρωμάτων από **BGR (Blue, Green, Red)** σε **RGB (Red, Green, Blue).**

cv2.rectangle

Η συνάρτηση **cv2.rectangle** σχεδιάζει ένα ορθογώνιο παραλληλόγραμμο σε ένα πλαίσιο εικόνας. Στην περίπτωση μας, το ορθογώνιο παραλληλόγραμμο θα περικλείει το πρόσωπο.

Η σύνταξη της συνάρτησης **cv2.rectangle** είναι η κάτωθι:

cv2.rectangle(image, start_point, end_point, color, thickness)

όπου:

image – Το πλαίσιο εικόνας

start_point – Το αρχικό σημείο του ορθογωνίου παραλληλογράμμου. Το όρισμα **start_point** είναι σε μορφή πλειάδας η οποία περιέχει τις συντεταγμένες του αρχικού σημείου.

end_point – Το τελικό σημείο του ορθογωνίου παραλληλογράμμου. Το όρισμα **end_point** είναι σε μορφή πλειάδας η οποία περιέχει τις συντεταγμένες του τελικού σημείου.

color – Το χρώμα των γραμμών του ορθογωνίου παραλληλογράμμου. Το όρισμα **color** είναι σε μορφή πλειάδας η οποία περιέχει τις συνιστώσες του χρώματος.

thickness – Το πάχος των γραμμών του ορθογωνίου παραλληλογράμμου.

cv2.putText

Η συνάρτηση **cv2.putText** σχεδιάζει ένα κείμενο σε ένα πλαίσιο εικόνας. Στην περίπτωση μας, το κείμενο θα είναι η ετικέτα του ονόματος του προσώπου ή η λέξη "**Unknown**".

Η σύνταξη της συνάρτησης **cv2.putText** είναι η κάτωθι:

cv2.putText(image, text, org, font, fontScale, color[, thickness])

όπου:

image –	Το πλαίσιο εικόνας
text–	Το κείμενο που θα σχεδιαστεί.
org –	Οι συντεταγμένες του άνω αριστερού σημείου του κειμένου. Το όρισμα **org** είναι σε μορφή πλειάδας η οποία περιέχει τις συντεταγμένες του άνω αριστερού σημείου.
font –	Η γραμματοσειρά του κειμένου.
fontScale –	Το μέγεθος της γραμματοσειράς του κειμένου.
color –	Το χρώμα του κειμένου. Το όρισμα **color** είναι σε μορφή πλειάδας η οποία περιέχει τις συνιστώσες του χρώματος.
thickness –	Το πάχος των γραμμών του κειμένου.

cv2.imshow

Η συνάρτηση **cv2.imshow** εμφανίζει το πλαίσιο εικόνας. Στην περίπτωση μας, το πλαίσιο εικόνας θα απεικονίζει το πρόσωπο με την ετικέτα του ονόματος του ή η λέξη "**Unknown"**.

Η σύνταξη της συνάρτησης **cv2.imshow** είναι η κάτωθι:

cv2.imshow(window_name, image)

όπου:

window_name–	Το όνομα του παραθύρου.
image –	Το πλαίσιο εικόνας

5.3.1.2 face_recognition

Από την βιβλιοθήκη **face_recognition** χρησιμοποιήθηκαν οι συναρτήσεις:

face_recognition.compare_faces

Η συνάρτηση **face_recognition.compare_faces** συγκρίνει ένα πρόσωπο με μια λίστα γνωστών προσώπων. Τα πρόσωπα είναι κωδικοποιημένα σε διανύσματα 128 θέσεων.

Η σύνταξη της συνάρτησης **face_recognition.compare_faces** είναι η κάτωθι:

face_recognition.compare_faces(known_face_encodings,face_encoding_to_check)

όπου:

known_face_encodings – Η λίστα με τα κωδικοποιημένα πρόσωπα που βρίσκονται είτε στην βάση δεδομένων του Νέφους είτε στην βάση δεδομένων του τοπικού **MEC** κόμβου.

face_encoding_to_check – Το κωδικοποιημένο πρόσωπο σε μορφή διανύσματος 128 θέσεων.

Η συνάρτηση επιστρέφει μια λίστα, τύπου **Boolean,** με τιμές **True/False** υποδεικνύοντας ποια από τα πρόσωπα της λίστας **known_face_encodings** ταιριάζουν με το πρόσωπο **face_encoding_to_check.**

face_recognition.face_locations

Η συνάρτηση **face_recognition.face_locations** επιστρέφει τις συντεταγμένες των προσώπων που εντοπίστηκαν στο πλαίσιο εικόνας. Η σύνταξη της συνάρτησης **face_recognition.face_locations** είναι η κάτωθι:

face_recognition.face_locations(img, model='hog')

όπου:

img – Το πλαίσιο εικόνας σε μορφή αριθμητικού πίνακα

model='hog' – Εντοπισμός του προσώπου σύμφωνα με τα **Histogram Oriented Gradients**

Η συνάρτηση επιστρέφει μια λίστα με πλειάδες **(Tuples)** όπου κάθε πλειάδα περιέχει τις συντεταγμένες κάθε προσώπου που εντοπίστηκε.

face_recognition.face_encodings

Η συνάρτηση επιστρέφει μια λίστα όπου κάθε πρόσωπο που εντοπίστηκε κωδικοποιείται σε διάνυσμα 128 θέσεων.

Η σύνταξη της συνάρτησης **face_recognition.face_encodings** είναι η κάτωθι:

face_recognition.face_encodings(face_image, known_face_locations)

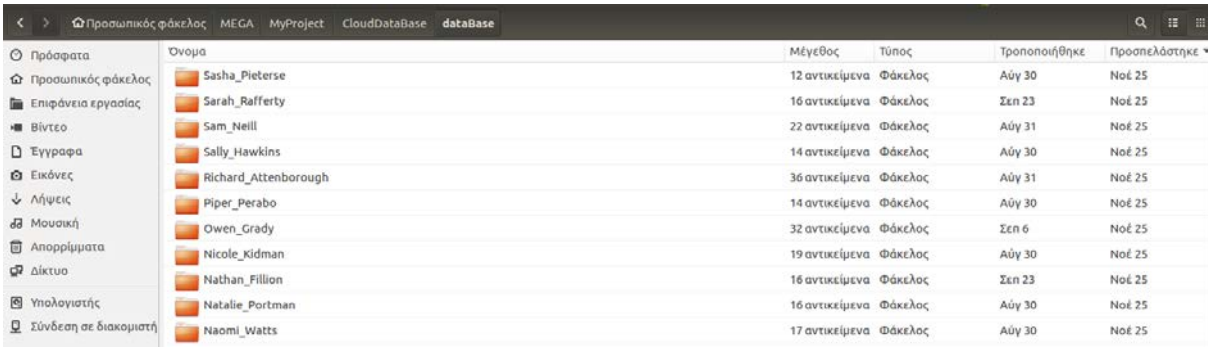
όπου:

face_image – Το πλαίσιο εικόνας το οποίο περιέχει ένα ή περισσότερα πρόσωπα.

known_face_locations – Οι συντεταγμένες του κάθε προσώπου.

5.4 Εκπαίδευση Νέφους

Στον τοπικό σκληρό δίσκο του τερματικού έχουμε δημιουργήσει έναν φάκελο με το όνομα **CloudDatabase**. Εντός του φακέλου **CloudDatabase** υπάρχει ένας υποφάκελος με το όνομα **dataBase** και εντός του υποφακέλου **dataBase** υπάρχει ένα σύνολο φακέλων οι οποίοι έχουν ως όνομα το όνομα κάθε γνωστού καλλιτέχνη.



Εικόνα 19: *Φάκελοι με τα ονόματα των καλλιτεχνών*

Εντός κάθε υποφακέλου με το όνομα του κάθε καλλιτέχνη βρίσκεται ένα σύνολο φωτογραφιών που τον/την απεικονίζουν. Συνολικά υπάρχουν 561 φωτογραφίες από 30 καλλιτέχνες συνολικά.

Εικόνα 20*: Φωτογραφίες γνωστού καλλιτέχνη*

Εκτελούμε την εφαρμογή **Training.py**

```
(cv) zouglisnik@zouglisnik-Lenovo-B70-80:~/MEGA/MyProject$ python Training.py
Επεξεργασία εικόνας 1/561
Επεξεργασία εικόνας 2/561
Επεξεργασία εικόνας 3/561
Επεξεργασία εικόνας 4/561
Επεξεργασία εικόνας 5/561
Επεξεργασία εικόνας 6/561
Επεξεργασία εικόνας 7/561
Επεξεργασία εικόνας 8/561
```

Εικόνα 21: Εκτέλεση της εφαρμογής Training.py

Μετά το πέρας της εκτέλεσης της, λαμβάνουμε στο τερματικό μήνυμα επιτυχούς εκπαίδευσης και το όνομα της βάσης δεδομένων η οποία είναι αποθηκευμένη στο Νέφος.

```
Επεξεργασία εικόνας 558/561
Επεξεργασία εικόνας 559/561
Επεξεργασία εικόνας 560/561
Επεξεργασία εικόνας 561/561
Αποθήκευση διανυσμάτων και ονομάτων στο αρχείο CloudDatabaseEncodings.pickle
Επιτυχής κωδικοποίηση...

**************************
*Open University of Cyprus*
* Μελέτη Ρυθμαπόδοσης MEC *
*     Νικόλαος Ζουγλής    *
**************************
(cv) zouglisnik@zouglisnik-Lenovo-B70-80:~/MEGA/MyProject$
```

Εικόνα 22: Πέρας εκτέλεσης και αναφορά επιτυχούς κωδικοποίησης

5.5 Εκκίνηση Συστήματος

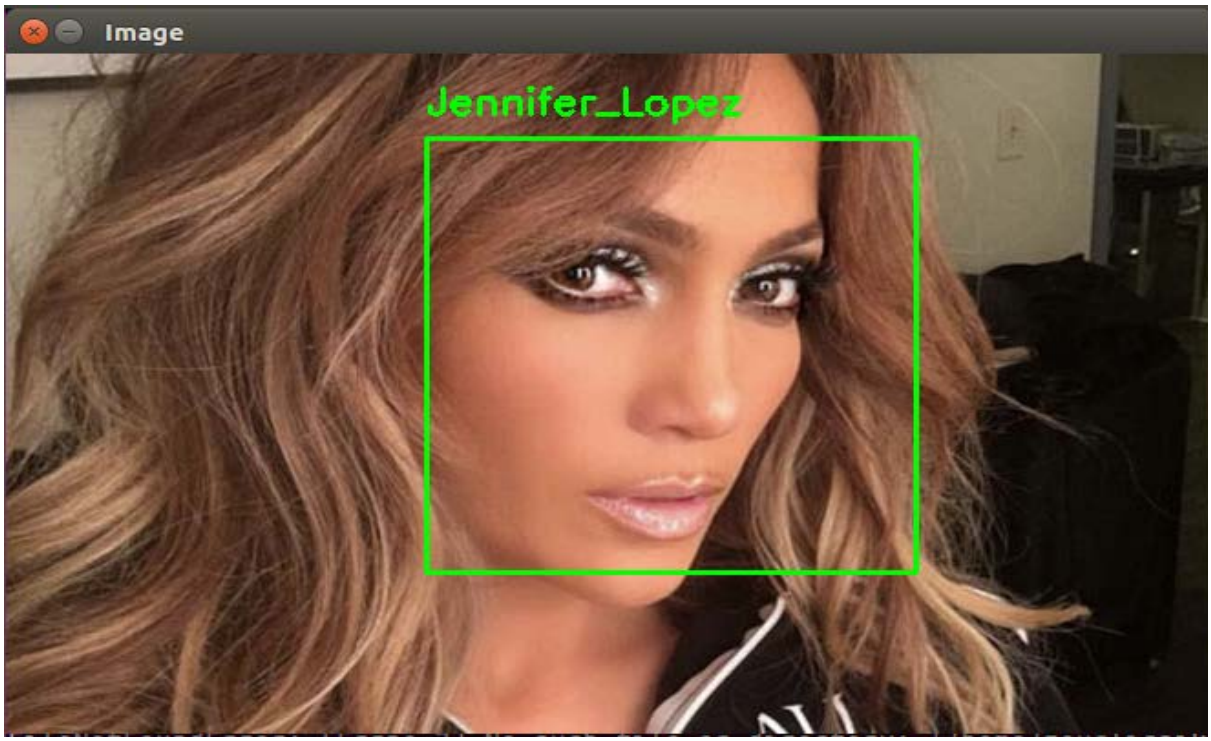
Εκκινούμε τις εφαρμογή **Cloud-OUC.py** και **MEC-CacheLFU-OUC.py**. Η πρώτη εφαρμογή θα έχει τον ρόλο του Νέφους και η δεύτερη θα έχει τον ρόλο του τοπικού **MEC** κόμβου όπου η μνήμη cache θα αναβαθμίζεται σύμφωνα με τον αλγόριθμο **LFU**.

```
(cv) zouglisnik@zouglisnik-Lenovo-B70-80:~/MEGA/MyProject$ python Cloud-OUC.py
Laura_Dern
Μέγεθος δεδομένων (bytes) : 80628
Claire_Dearing
```

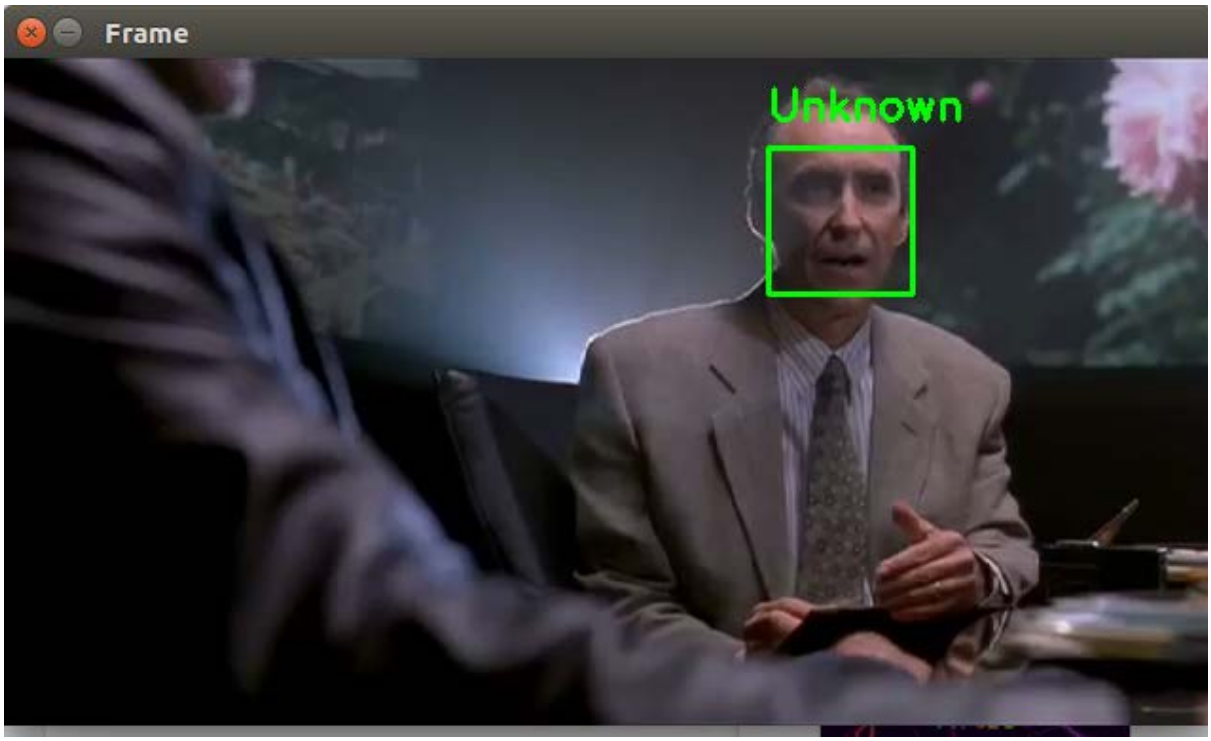
Εικόνα 23: Εκκίνηση εφαρμογής **Cloud-OUC.py**

5.6 Απόκριση Συστήματος

Καθώς ρέει το **Stream** τόσο ο **MEC** κόμβος όσο και το Νέφος προβαίνουν στην επιτήρηση και αναγνώριση των προσώπων. Το αποτέλεσμα εμφανίζεται στην οθόνη του τερματικού, όπου στην προκειμένη περίπτωση πρόκειται για την οθόνη του ηλεκτρονικού υπολογιστή.



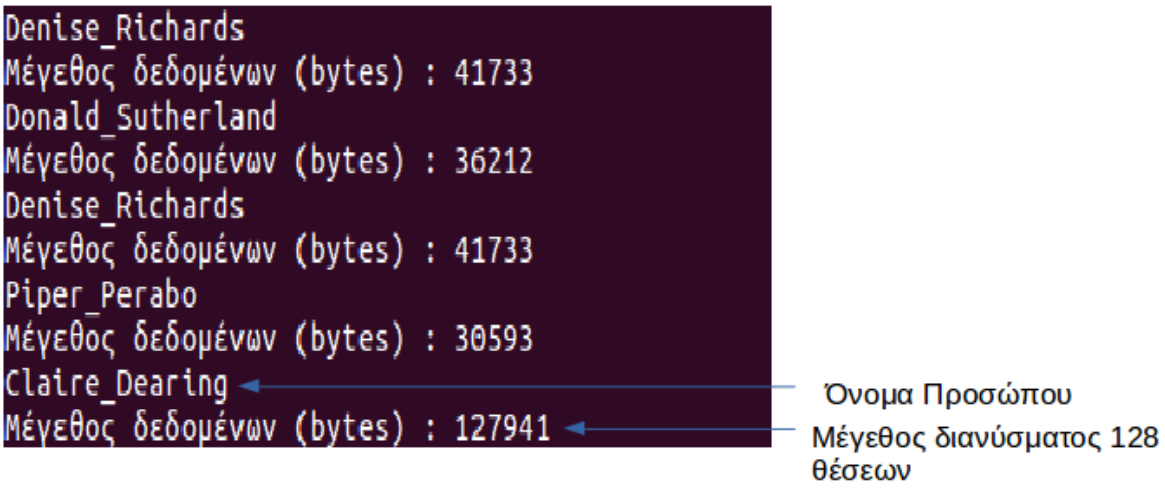
Εικόνα 24: Επιτυχής αναγνώριση



Εικόνα 25: Αποτυχής αναγνώριση

Οι Εικόνες 24 και 25 απεικονίζουν τον επιτυχή εντοπισμό του προσώπου και την επιτυχή ή αποτυχή αναγνώριση αντίστοιχα. Οι Εικόνες 24 και 25 προβάλλονται στην οθόνη του τερματικού.

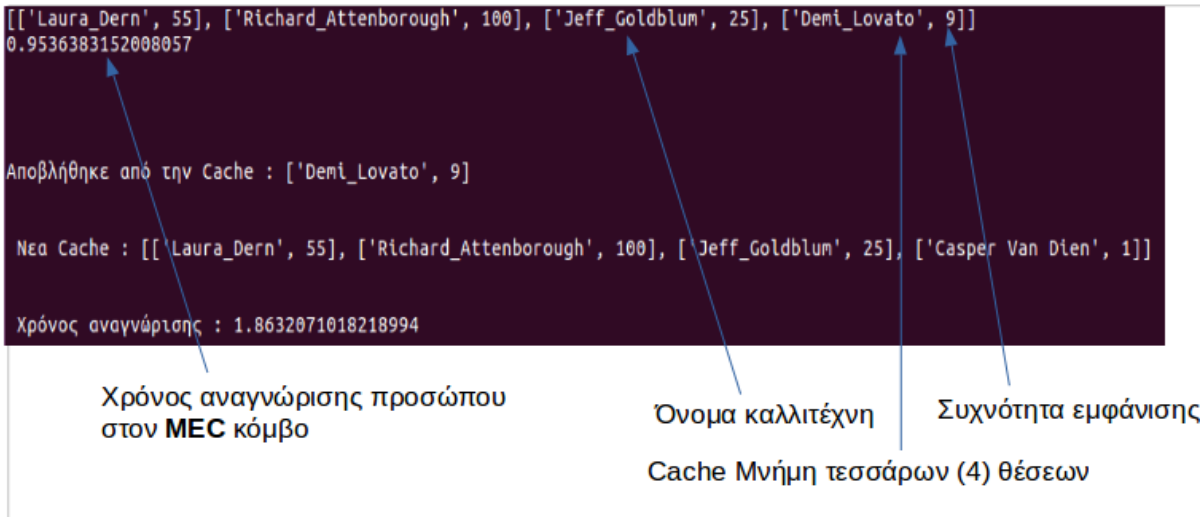
5.7 Απόκριση Νέφους



Εικόνα 26: Απόκριση Νέφους προς τον **MEC** κόμβο

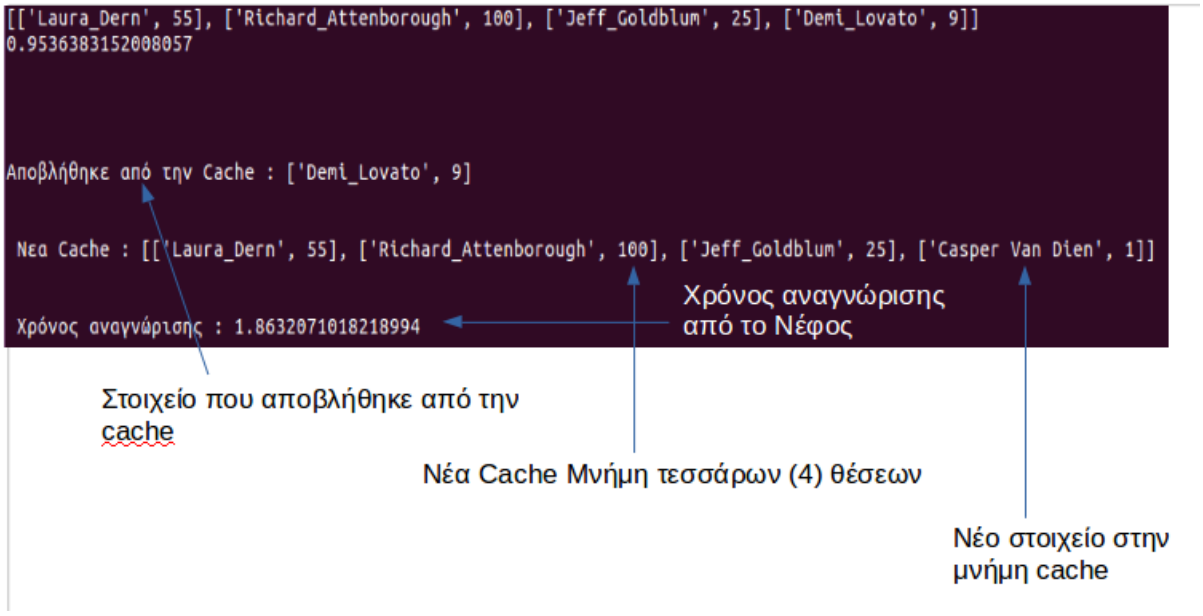
Η Εικόνα 26 απεικονίζει την απόκριση του Νέφους μετά από ερώτηση **(Query)** του **MEC** κόμβου. Διακρίνεται ξεκάθαρα η επιτυχής αναγνώριση του προσώπου αφού εμφανίζεται το όνομα του καθώς και το μέγεθος του πίνακα διανυσμάτων των 128 θέσεων. Επομένως, το Νέφος θα διαβιβάζει, μέσω δικτύου προς τον **MEC** ένα λεξικό με το κλειδί **"names"** που θα περιέχει την λίστα με το όνομα του προσώπου και το κλειδί **"encodings"** με την αντίστοιχη λίστα των διανυσμάτων των 128 θέσεων.

5.8 Απόκριση MEC κόμβου



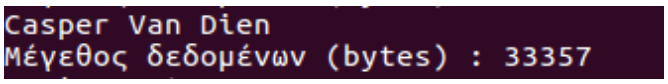
Εικόνα 27: Απόκριση **MEC** κόμβου με αλγόριθμο ενημέρωσης **LFU**

Η Εικόνα 27 απεικονίζει την απόκριση του **MEC** κόμβου με χρήση αλγορίθμου **LFU**. Διακρίνεται η μνήμη cache η οποία έχει μέγεθος τεσσάρων (4) θέσεων δηλαδή η μέγιστη χωρητικότητα της είναι τέσσερις καλλιτέχνες με τα ονόματα τους και τα αντίστοιχα διανύσματα των 128 θέσεων. Επίσης, διακρίνεται η συχνότητα αναγνώρισης των προσώπων τα οποία είναι ήδη αποθηκευμένα στην cache μνήμη. Στην περίπτωση μας η καλλιτέχνης **“Laura Dern”** αναγνωρίσθηκε 55 φορές κατά την διάρκεια του **stream**, ο καλλιτέχνης **“Richard Attenborough”** αναγνωρίσθηκε 100 φορές κατά την διάρκεια του **stream**, ο καλλιτέχνης **“Jeff Goldblum”** αναγνωρίσθηκε 25 φορές κατά την διάρκεια του **stream** και η καλλιτέχνης **“Demi Lovato”** αναγνωρίσθηκε 9 φορές κατά την διάρκεια του **stream**. Επίσης, διακρίνεται ο χρόνος αναγνώρισης του προσώπου όταν τα χαρακτηριστικά του είναι ήδη αποθηκευμένα στον τοπικό **MEC** κόμβο όπου στην περίπτωση μας ο χρόνος αυτός ισούνται με 0,9536 δευτερόλεπτα ή 953,6 msec.



Εικόνα 28: Απόκριση MEC κόμβου με αλγόριθμο ενημέρωσης **LFU**

Συνεχίζοντας με την απόκριση του **MEC** κόμβου, η Εικόνα 28 απεικονίζει επιπλέον στοιχεία. Αναλυτικά, το πρόσωπο που απεικονιζόταν στο τρέχων πλαίσιο **(Frame)** δεν αναγνωρίσθηκε από τον **MEC** κόμβο. Επομένως, ο **MEC** κόμβος θα έστελνε ερώτημα (**Query)** στο Νέφος με την διαδικασία που αναφέραμε σε προηγούμενη παράγραφο. Στην συνέχεια το Νέφος αναγνώρισε το πρόσωπο με επιτυχία και απέστειλε στον **MEC** κόμβο τα στοιχεία του προσώπου δηλαδή τον λεξικό με την λίστα του ονόματος και την λίστα με τα διανύσματα των 128 θέσεων. Επίσης, διακρίνεται ο χρόνος αναγνώρισης του προσώπου όταν τα χαρακτηριστικά του αναζητούνται από το Νέφος όπου στην περίπτωση μας ο χρόνος αυτός ισούνται με 1,8632 δευτερόλεπτα ή 1863,2 msec.

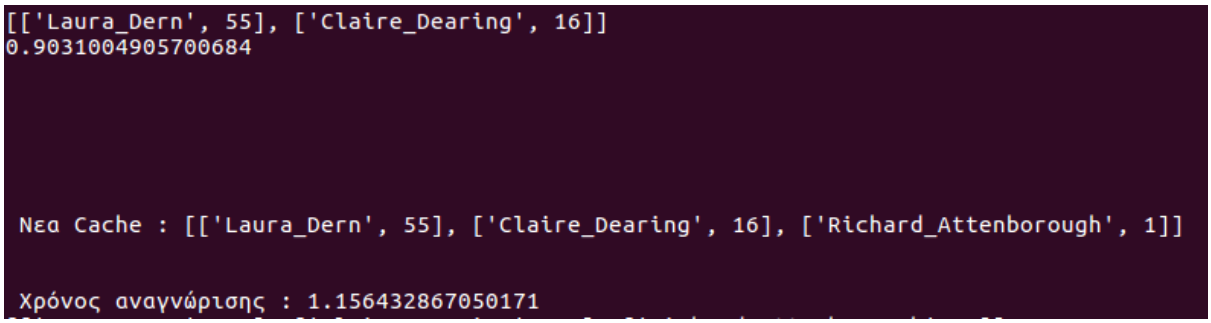


Εικόνα 29: Απόκριση Νέφους

Η Εικόνα 29 απεικονίζει την απόκριση του Νέφους όπου διακρίνεται το όνομα του καλλιτέχνη που αναγνωρίσθηκε με επιτυχία καθώς και το μέγεθος του πίνακα διανυσμάτων των 128 θέσεων. Στην προκειμένη περίπτωση πρόκειται για τον καλλιτέχνη **"Casper Van Dien"** και το μέγεθος του πίνακα διανυσμάτων των 128 θέσεων ισούνται με 33.357 bytes.

Όπως απεικονίζει η Εικόνα 28 η ενημέρωση της μνήμης cache έγινε με την χρήση του αλγορίθμου **LFU (Least Frequency Used)**. Κατά εφαρμογή του αλγορίθμου **LFU** το στοιχείο που αποβάλλεται από την μνήμη cache είναι η καλλιτέχνης **"Demi Lovato"** επειδή έχει την ελάχιστη συχνότητα αναγνώρισης δηλαδή αναγνωρίσθηκε 9 φορές σε σχέση με τα επιμέρους

στοιχεία της μνήμης cache. Επιπροσθέτως, το νέο στοιχείο που εισάγεται στην μνήμη cache, δηλαδή ο καλλιτέχνης **"Casper Van Dien"** θα έχει συχνότητα αναγνώρισης ίση με την μονάδα (1) γιατί πρόκειται για νέο στοιχείο. Θα πρέπει να επισημανθεί ότι εάν το μέγεθος της μνήμης cache ήταν μεγαλύτερο από τέσσερις (4) θέσεις και υπήρχε διαθέσιμη θέση τότε κανένα στοιχείο δεν θα αποβαλλόταν απλά το νέο στοιχείο θα προστίθεται στο τέλος. Η περίπτωση αυτή απεικονίζεται την Εικόνα 30 όπου ο καλλιτέχνης **"Richard Attenborough"** προστέθηκε στο τέλος της μνήμης cache γιατί υπήρχε διαθέσιμη θέση.



Εικόνα 30: Νέο στοιχείο στην μνήμη cache

Όσον αφορά τα μεγέθη των αρχείων βάσης δεδομένων τόσο στο Νέφος όσο και στον **MEC** κόμβο, αυτά απεικονίζονται στις Εικόνες 30 και 31 αντίστοιχα

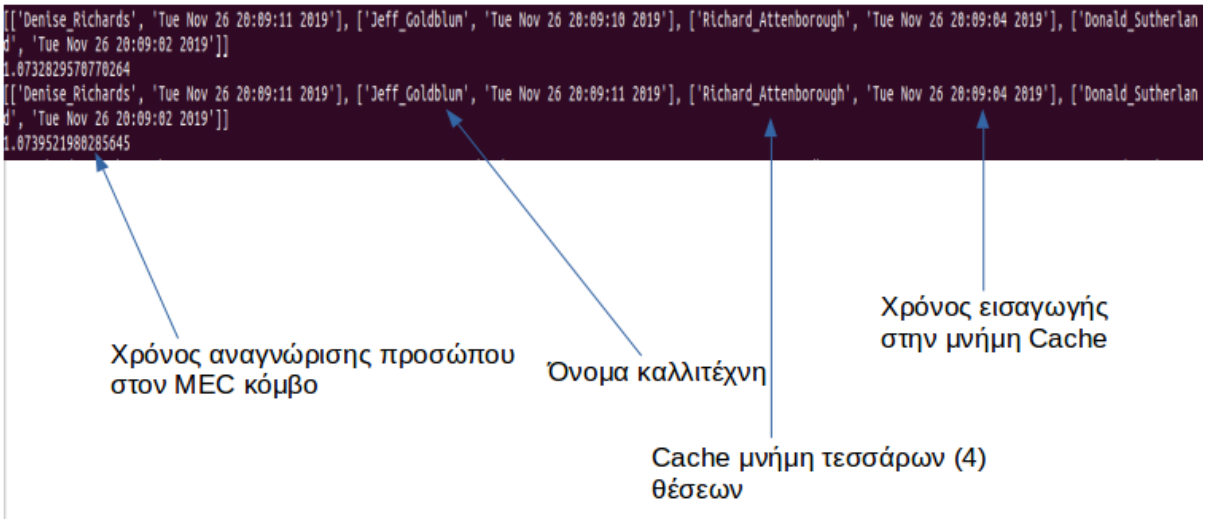
Εικόνα 32: Μέγεθος αρχείο βάσης δεδομένων του Νέφους

Εικόνα 31: Μέγεθος αρχείο βάσης δεδομένων του τοπικού **MEC** κόμβου

Η διαφορά στο μέγεθος είναι προφανής με αναλογία μεγεθών περίπου 6,3:1. Επομένως, στο παραπάνω σύστημα μπορούμε να πετύχουμε ένα καλό αποτέλεσμα με μικρό μέγεθος αρχείου

βάσης δεδομένων κάτι που είναι επιθυμητό λόγω του περιορισμού του υλικού αποθήκευσης του τοπικού **MEC** κόμβου.

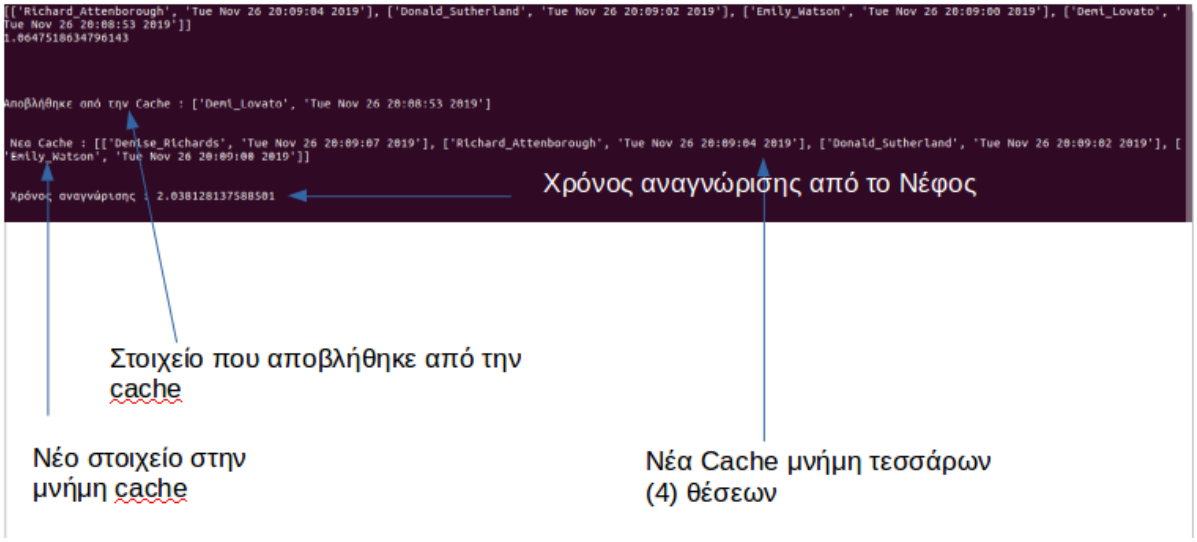
Στην συνέχεια εκκινούμε τις εφαρμογές **Cloud-OUC.py** και **MEC-CacheTimer-OUC.py**. Η διαδικασία που θα ακολουθηθεί θα είναι η ίδια με την προηγούμενη διαδικασία αλλά σε αυτή την περίπτωση η μνήμη cache θα αναβαθμίζεται σύμφωνα με τον αλγόριθμο Χρονοκαθυστέρησης.



Εικόνα 33: Απόκριση **MEC** κόμβου με αλγόριθμο **Χρονοκαθυστέρησης**

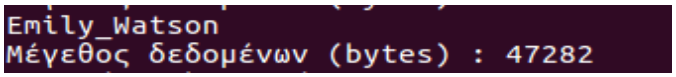
Η Εικόνα 33 απεικονίζει την απόκριση του **MEC** κόμβου με χρήση του αλγορίθμου Χρονοκαθυστέρησης όπου διακρίνεται η μνήμη cache όπως ακριβώς και στην περίπτωση του αλγορίθμου **LFU**. Διακρίνεται ο χρόνος όπου τα στοιχεία εισήλθαν στην cache μνήμη. Στην περίπτωση μας η καλλιτέχνης **"Denise Richards"** εισήχθηκε στην μνήμη cache στις 29 Νοεμβρίου 2019 και ώρα 20:09:11, ο καλλιτέχνης **"Jeff Goldblum"** εισήχθηκε στην μνήμη cache στις 29 Νοεμβρίου 2019 και ώρα 20:09:11, ο καλλιτέχνης **"Richard Attenborough"** εισήχθηκε στην μνήμη cache στις 29 Νοεμβρίου 2019 και ώρα 20:09:04, και ο καλλιτέχνης **"Donald Sutherland"** εισήχθηκε στην μνήμη cache στις 29 Νοεμβρίου 2019 και ώρα 20:09:02. Επίσης, διακρίνεται ο χρόνος αναγνώρισης του προσώπου όταν τα χαρακτηριστικά του είναι ήδη αποθηκευμένα στον τοπικό **MEC** κόμβο όπου στην περίπτωση μας ο χρόνος αυτός ισούνται με 1,07395 δευτερόλεπτα ή 1073,95 msec. Παρατηρώντας τους χρόνους μεταξύ των καλλιτεχνών **"Denise Richards"** και **"Jeff Goldblum"** βλέπει κανείς ότι οι χρόνοι εισαγωγής στην μνήμη cache είναι οι ίδιοι. Αυτό σημαίνει ότι ο καλλιτέχνης **Jeff Goldblum"**

ήταν ήδη αποθηκευμένος στην μνήμη cache από προηγούμενη αναγνώριση κάτι που διακρίνεται στην Εικόνα 34.



Εικόνα 34: Απόκριση MEC κόμβου με αλγόριθμο Χρονοκαθυστέρησης

Συνεχίζοντας με την απόκριση του **MEC** κόμβου, η Εικόνα 34 απεικονίζει επιπλέον στοιχεία. Αναλυτικά, το πρόσωπο που απεικονιζόταν στο τρέχων πλαίσιο **(Frame)** δεν αναγνωρίσθηκε από τον **MEC** κόμβο. Επομένως, ο **MEC** κόμβος θα έστελνε ερώτημα (**Query)** στο Νέφος με την διαδικασία που αναφέραμε σε προηγούμενη παράγραφο. Στην συνέχεια το Νέφος αναγνώρισε το πρόσωπο με επιτυχία και απέστειλε στον **MEC** κόμβο τα στοιχεία του προσώπου δηλαδή τον λεξικό με την λίστα του ονόματος και την λίστα με τα διανύσματα των 128 θέσεων. Επίσης, διακρίνεται ο χρόνος αναγνώρισης του προσώπου όταν τα χαρακτηριστικά του αναζητούνται από το Νέφος όπου στην περίπτωση μας ο χρόνος αυτός ισούνται με 2,0381 δευτερόλεπτα ή 2038,1 msec.



Εικόνα 35: Απόκριση Νέφους

Η Εικόνα 35 απεικονίζει την απόκριση του Νέφους όπου διακρίνεται το όνομα του καλλιτέχνη που αναγνωρίσθηκε με επιτυχία καθώς και το μέγεθος του πίνακα διανυσμάτων των 128 θέσεων. Στην προκειμένη περίπτωση πρόκειται για τον καλλιτέχνη **"Emily_Watson"** και το μέγεθος του πίνακα διανυσμάτων των 128 θέσεων ισούνται με 47.282 bytes.

Όπως απεικονίζει η Εικόνα 34 η ενημέρωση της μνήμης cache έγινε με την χρήση του αλγορίθμου **Χρονοκαθυστέρησης**. Κατά εφαρμογή του αλγορίθμου **Χρονοκαθυστέρησης** το στοιχείο που αποβάλλεται από την μνήμη cache είναι η καλλιτέχνης **"Demi Lovato"** επειδή βρίσκεται στο τέλος της μνήμης cache. Να σημειωθεί ότι ο αλγόριθμος Χρονοκαθυστέρησης ταξινομεί την μνήμη cache κάθε φορά που γίνεται χρήση του κάθε στοιχείου της. Συγκεκριμένα, το στοιχείο που χρησιμοποιήθηκε την τελευταία φορά θα μπει πρώτο στην ουρά ενώ το στοιχείο το οποίο είχε τον μεγαλύτερο χρόνο παραμονής στην ουρά χωρίς να χρησιμοποιηθεί θα βρίσκεται στην τελευταία θέση υπό την προϋπόθεση ότι η μνήμη cache είναι πλήρης.

5.9 Πειραματικά Αποτελέσματα

Η όλη διαδικασία που παρουσιάστηκε στην προηγούμενη ενότητα πραγματοποιήθηκε για διάφορες τιμές την χωρητικότητας της μνήμης cache. Συγκεκριμένα, πραγματοποιήθηκαν μετρήσεις

- του μέσου χρόνου απόκρισης εάν τα στοιχεία του προσώπου είναι αποθηκευμένα στην βάση δεδομένων του **MEC** κόμβου,
- του μέσου χρόνου απόκρισης εάν τα στοιχεία του προσώπου είναι αποθηκευμένα εάν στην βάση δεδομένων του Νέφους,
- του πλήθος των προσώπων που αναγνώρισε του Νέφους συναρτήσει του μεγέθους της μνήμης cache,
- του πλήθος των προσώπων που αναγνώρισε ο **MEC** κόμβος συναρτήσει του μεγέθους της μνήμης cache.

5.9.1 Μετρήσεις χρόνου αναγνώρισης

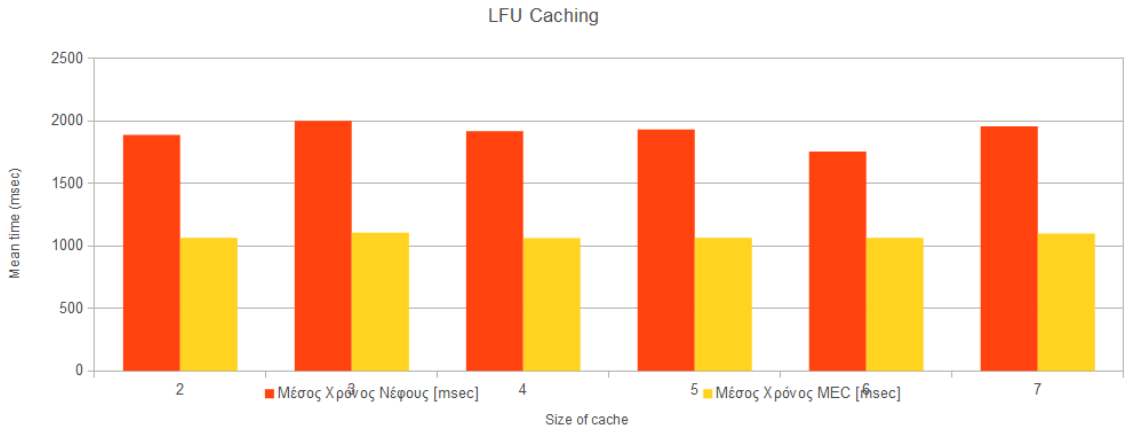
Μέγεθος cache	Μέσος Χρόνος Νέφους [msec]	Μέσος Χρόνος MEC [msec]	%ΔT
2	1889	1065	-43,6
3	2001	1106	-44,7
4	1919	1063	-44,6
5	1932	1064	-44,9
6	1756	1065	-39,4
7	1957	1100	-43,8

Πίνακας 1: Μέσος χρόνος αναγνώρισης με αλγόριθμο **LFU**

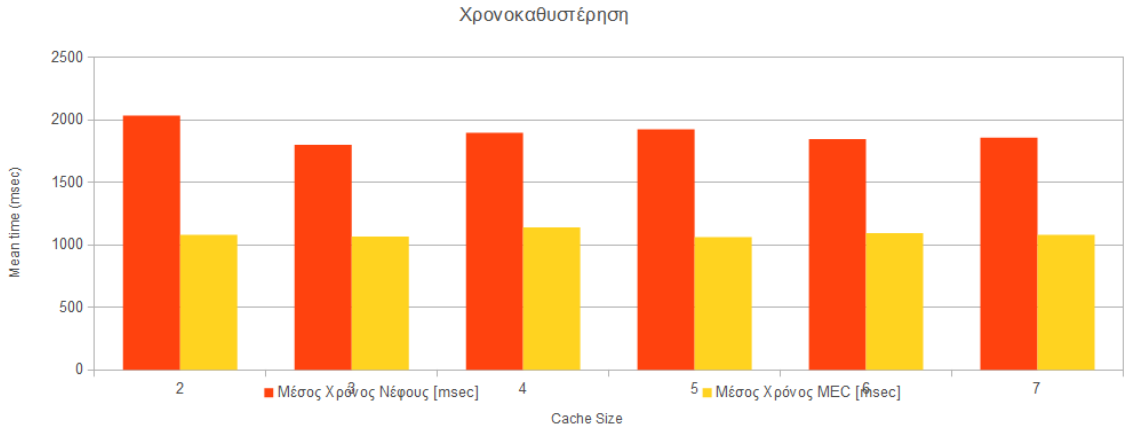
Μέγεθος cache	Μέσος Χρόνος Νέφους [msec]	Μέσος Χρόνος MEC [msec]	%ΔT
2	2037	1080	-47
3	1801	1067	-40,8
4	1898	1140	-39,9
5	1926	1063	-44,8
6	1848	1095	-40,7
7	1859	1081	-41,9

Πίνακας 2: Μέσος χρόνος αναγνώρισης με αλγόριθμο Χρονοκαθυστέρησης

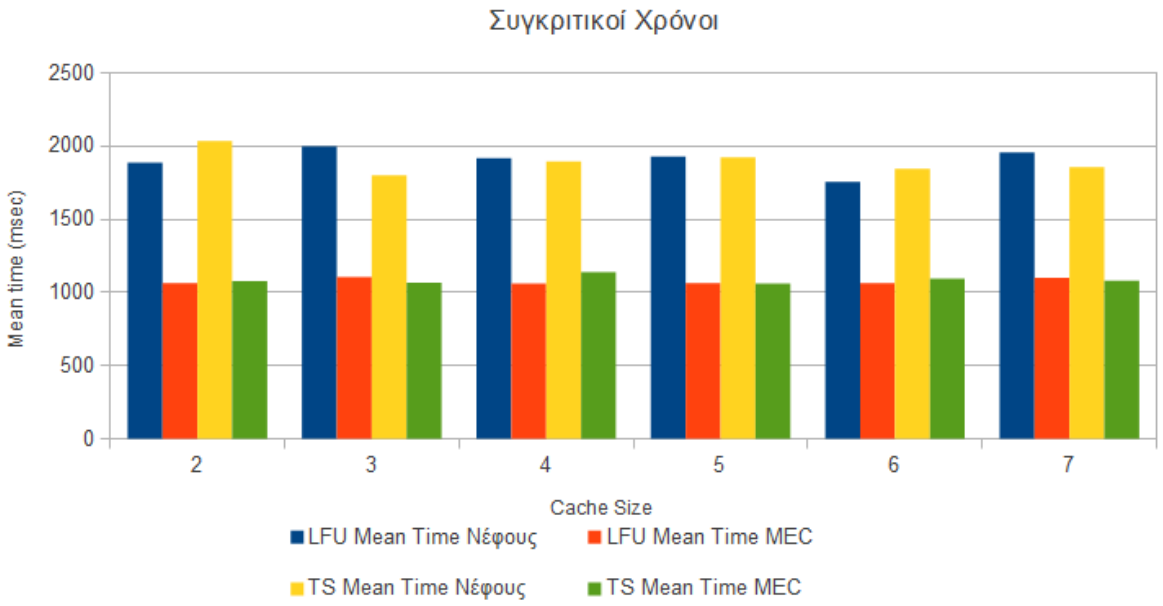
5.9.2 Διαγράμματα χρόνου αναγνώρισης



Διάγραμμα 1: Σύγκριση χρόνων **MEC**/Νέφους vs Μεγέθους Cache **(LFU)**



Διάγραμμα 2: Σύγκριση χρόνων **MEC**/Νέφους vs Μεγέθους Cache **(Χρονοκαθυστέρηση)**



Διάγραμμα 3: Συγκριτικοί χρόνοι αλγορίθμων

5.9.3 Μετρήσεις συχνότητας αναγνωρίσεων

Ως αναγνώριση εννοούμε την διαδικασία αναγνώρισης προσώπου και μόνο αυτή δηλαδή δεν λαμβάνεται υπόψιν η διαδικασία εντοπισμού του προσώπου. Ο λόγος είναι προφανής αφού η διαδικασία εντοπισμού πραγματοποιείται αποκλειστικά από τον **MEC** κόμβο. Επομένως, η εργασία του Νέφος είναι μόνο η διαδικασία αναγνώρισης του προσώπου από το διάνυσμα των 128 θέσεων που έλαβε από τον **MEC** κόμβο.

Κατά την διάρκεια του **streaming** καταγραφόταν σε αρχείο τύπου csv πόσες αναγνωρίσεις έγιναν από τον τοπικό **MEC** κόμβο και πόσες αναγνωρίσεις έγιναν από το Νέφος. Τα παρακάτω διαγράμματα απεικονίζουν τα αποτελέσματα

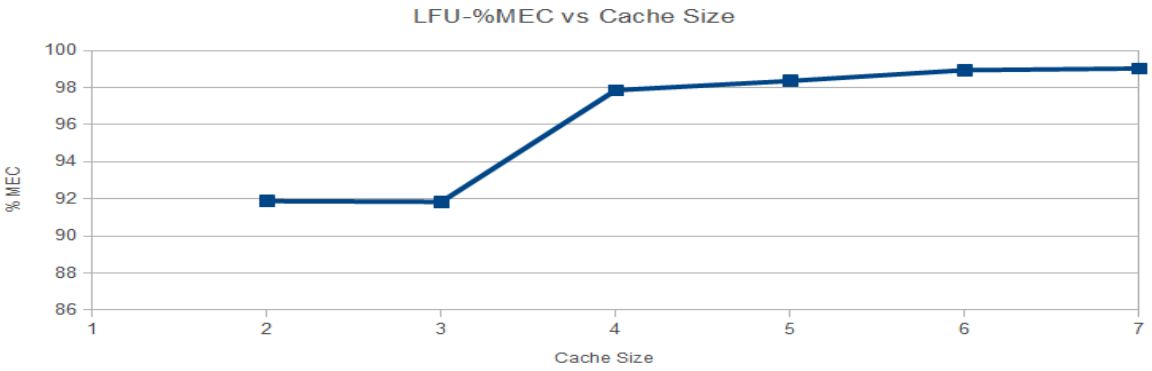
Μέγεθος Cache	MEC	Νέφος	Σύνολο	% MEC
2	1415	125	1540	91,88
3	1339	119	1458	91,84
4	1556	34	1590	97,86
5	1557	26	1583	98,36
6	1589	17	1606	98,94
7	1534	15	1549	99,03

Πίνακας 3: Συχνότητα αναγνωρίσεων με τον αλγόριθμο **LFU**

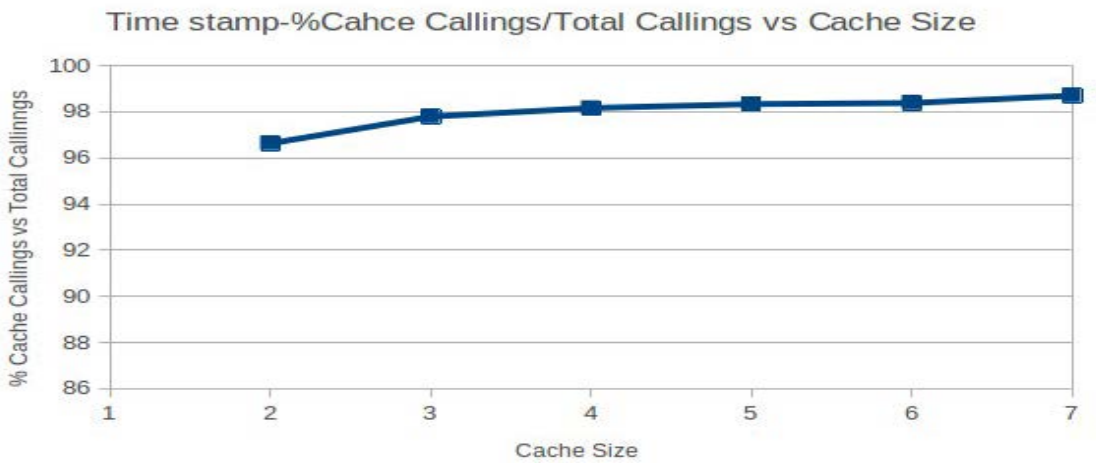
Μέγεθος Cache	MEC	Νέφος	Σύνολο	% MEC
2	1493	52	1545	96,63
3	1546	35	1581	97,79
4	1437	27	1464	98,16
5	1534	26	1560	98,33
6	1522	25	1547	98,38
7	1522	20	1542	98,7

Πίνακας 4: Συχνότητα αναγνωρίσεων με τον αλγόριθμο Χρονοκαθυστέρησης

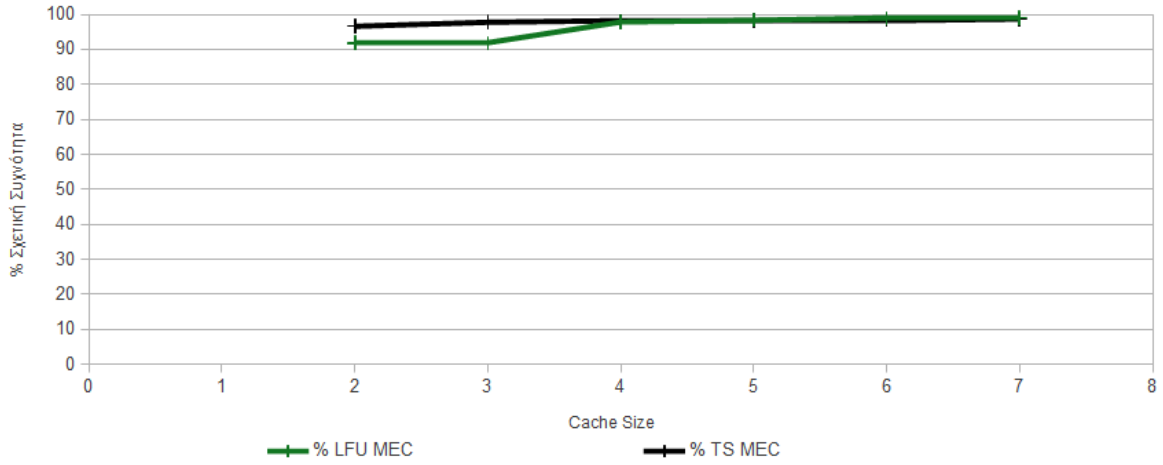
5.9.4 Διαγράμματα συχνότητας αναγνωρίσεων



Διάγραμμα 4: Διάγραμμα %**MEC** σχετικής συχνότητας vs μεγέθους μνήμης Cache με χρήση αλγορίθμου **LFU**



Διάγραμμα 5: Διάγραμμα %**MEC** σχετικής συχνότητας vs μεγέθους μνήμης Cache με χρήση αλγορίθμου Χρονοκαθυστέρησης



Διάγραμμα 6: Συγκριτικό διάγραμμα Σχετικών Συχνοτήτων

5.9.5 Αποτελέσματα

Σύμφωνα με τις μετρήσεις, ο χρόνος απόκρισης είναι σαφώς μικρότερος στην περίπτωση όπου τα στοιχεία του προσώπου είναι αποθηκευμένα στην βάση δεδομένων του τοπικού **MEC** κόμβου από τον χρόνο απόκρισης στην περίπτωση όπου τα στοιχεία του προσώπου είναι αποθηκευμένα στο Νέφος. Ο χρόνος απόκρισης στην περίπτωση του caching στον **MEC** κόμβο είναι σε γενικές γραμμές μικρότερος κατά 40% περίπου σε σχέση με τον χρόνο απόκρισης όταν δεν έχει γίνει caching τοπικά. Θα πρέπει να επισημανθεί ότι αν και η τεχνική caching και εκτενέστερα η **MEC** αρχιτεκτονική μειώνει τον χρόνο απόκρισης ή με άλλα λόγια την καθυστέρηση **(Letancy)**, ωστόσο, στην περίπτωση μας η μείωση του χρόνου δεν εξαρτάται από το μέγεθος της μνήμης cache.

Συγκρίνοντας τους δυο αλγορίθμους ως προς τα μεγέθη **ΔΤ** δηλαδή την διαφορά των χρόνων μεταξύ **MEC** και Νέφους παρατηρείται ότι η χρήση του αλγορίθμου **LFU** παρουσιάζει μεγαλύτερο **ΔΤ**, το οποίο κυμαίνεται μεταξύ 1,27%-4,68%, σε σχέση με τον αλγόριθμο χρονοκαθυστέρησης.

Εν συνεχεία, σύμφωνα με τον Πίνακα 3 και Πίνακα 4 ο αλγόριθμος **LFU** παρουσιάζει μικρότερη σχετική συχνότητα χρήσης του **MEC** κόμβου, για μνήμη cache μικρότερη από τέσσερις (4) θέσεις, σε σχέση με τον αλγόριθμο Χρονοκαθυστέρησης. Για μέγεθος μνήμης cache μεγαλύτερο ίσο με τέσσερις (4) θέσεις παρατηρείται σύγκλιση των αλγορίθμων ως προς την σχετική συχνότητα. Για μέγεθος μνήμης cache μεγαλύτερο από με τέσσερις (4) θέσεις δεν υπάρχει ουσιαστική μεταβολή της σχετικής συχνότητας. Όσο μεγαλύτερη είναι η σχετική συχνότητα αναγνώρισης του προσώπου από τον **MEC** κόμβο τόσο λιγότερη θα είναι

η χρήση του Νέφους, δηλαδή θα μειωθεί η επιβάρυνση του δικτύου διασύνδεσης του **MEC** κόμβου με το Νέφος. Επιπροσθέτως, η αύξηση της μνήμης cache του τοπικού κόμβου όχι μόνο δεν συνεπάγεται την μείωση της κίνησης του δικτύου αλλά αντίθετα αυξάνει και το κόστος του υλικού του. Επομένως, για την περίπτωση μας ο **MEC** κόμβος με μέγεθος μνήμης τεσσάρων (4) θέσεων θεωρείται η ιδανική λύση.

Κεφάλαιο 6
Συμπεράσματα

Ο σκοπός της παρούσας διατριβής ήταν η μελέτη μιας περίπτωσης χρήσης ενός συστήματος επιτήρησης υιοθετώντας την **MEC** αρχιτεκτονική. Ο λόγος υιοθέτησης στης συγκεκριμένης αρχιτεκτονικής ήταν μείωση της καθυστέρησης **(Latency)** της απόκρισης του όλου συστήματος. Συγκεκριμένα, ο σκοπός της κάμερας επιτήρησης είναι να επιβλέπει έναν χώρο και να αναγνωρίζει τα πρόσωπα που διασχίζουν τον χώρο αυτό. Φυσικά πρόκειται για εξειδικευμένη περίπτωση όπου τέτοια συστήματα κατέχουν τα σώματα ασφαλείας. Ποιος ο λόγος εγκατάστασης ενός τέτοιου συστήματος ασφαλείας εάν η απόκριση του συστήματος δεν ήταν άμεση. Με άλλα λόγια ποιος ο λόγος εγκατάστασης μιας κάμερας επιτήρησης η οποία θα αναγνώριζε ένα πρόσωπο μετά από μεγάλο χρονικό διάστημα.

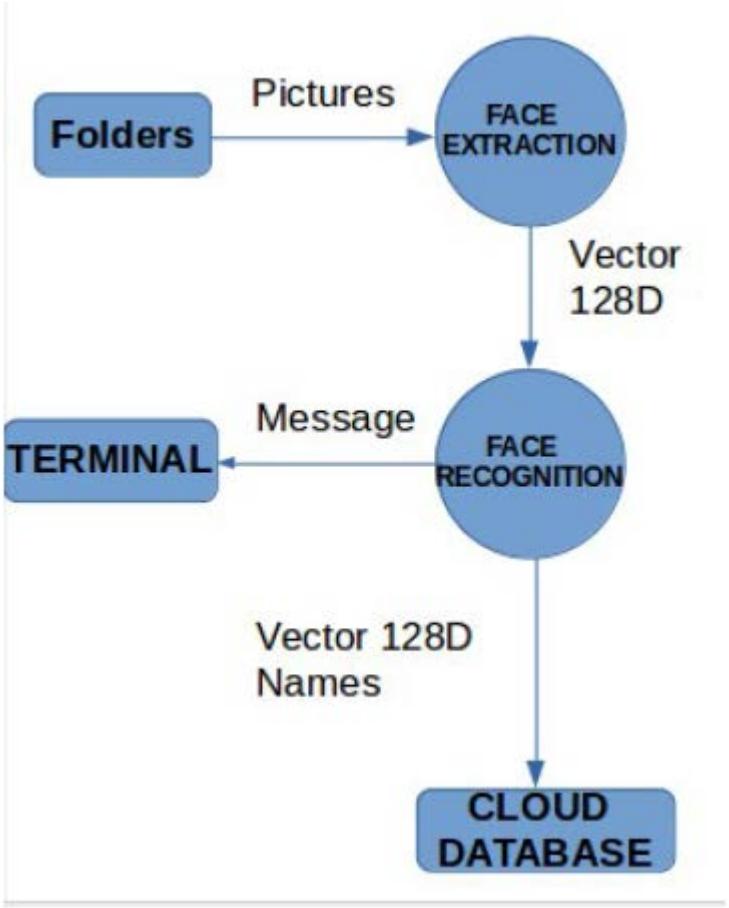
Υιοθετώντας την **MEC** αρχιτεκτονική εξομοιώσαμε ένα τέτοιο σύστημα και προβήκαμε σε επιτυχής αναγνωρίσεις προσώπων με την μέθοδο των **Histogram Oriented Gradients**. Το σύστημα αποτελούταν από το Νέφος και τον τοπικό **MEC** κόμβο. Εφαρμόσαμε δύο (2) αλγορίθμους ανανέωσης της cache μνήμης του τοπικού **MEC** κόμβου και προβήκαμε σε μετρήσεις χρόνου και σχετικής συχνότητας αναγνωρίσεων της μνήμης cache. Αποδείξαμε ότι ο χρόνος αναγνώρισης από τον τοπικό **MEC** κόμβο είναι μικρότερος από τον χρόνο αναγνώρισης από το Νέφος και ότι ο χρόνος απόκρισης δεν εξαρτάται από το μέγεθος της μνήμης cache. Ο ρόλος της μνήμης cache είναι σημαντικός αφού η χρήση της μειώνει την χρήση του Νέφους δηλαδή μειώνει την επιβάρυνση του δικτύου κορμού . Ωστόσο, αποδείξαμε ότι η αύξηση της μνήμης cache δεν επιφέρει ουσιαστική μείωση της επιβάρυνσης του δικτύου κορμού. Επομένως, ο σχεδιαστής ενός συστήματος **MEC** αρχιτεκτονικής είναι υποχρεωμένος να βρει την χρυσή τομή στο υλικό **(Hardware)** που θα εγκαταστήσει και την απόδοση των υπηρεσιών.

Τέλος θα πρέπει να επισημανθεί ότι υπάρχουν περιθώρια βελτίωσης του όλου συστήματος. Παράδειγμα εάν απαιτηθεί η αναγνώριση να είναι πιο ακριβής τότε μπορεί να γίνει χρήση των νευρωνικών δικτύων ή αλγορίθμων μηχανικής μάθησης αντί των **Histogram Oriented Gradients**. Επίσης η εγκατάσταση ισχυρού επεξεργαστή τόσο στο υλικού του **MEC** κόμβου όσο και στο υλικό του Νέφους μειώνει τον χρόνο αναγνώρισης όπως επίσης η εγκατάσταση γρήγορων δικτύων μειώνει τον χρόνο απόκρισης του Νέφους.

Παράρτημα Α
Training.py

```
'''
        ΑΝΟΙΧΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
        ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΝΙΚΟΛΑΟΥ ΖΟΥΓΛΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ'''
'''
Η εφαρμογή αυτή έχει τον ρόλο του Νέφους. Το Νέφος έχει βάση δεδομένων η οποία αποτελείται από 30 πρόσωπα και δέχεται ερωτήματτα από τους MEC κόμβους. Εάν κάποιο πρόσωπο δεν αναγνωριστεί τότε επιστρέφει "Unknown". Εάν το πρόσωπο αναγνωριστεί με επιτυχία τότε αποστέλνονται στους MEC κόμβους τα χαρακτηριστικά του, δηλαδή ένα dictionary με το όνομα του και τα διανύσματα 128 θέσεων.
'''
from imutils import paths
import face_recognition
import pickle
import cv2
import os


# Η λίστα imagePaths περιέχει τους φακέλους με τις φωτογραφίες
imagePaths =
list(paths.list_images('/home/zouglisnik/MEGA/MyProject/CloudDataBase'))

# Αρχικοποιούμε τις λίστες Encodings128D και NamesFromEncodings

# Λστα με τα διανύσματα
Encodings128D = []

#Λίστα με το όνομα που αντιστοιχεί σε κάθε διάνυσμα
NamesFromEncodings = []

for (i, imagePath) in enumerate(imagePaths):
        ''' Εξάγεται το όνομα του προσώπου που θα κωδικοποιηθεί. Το σύνολο των φωτογραφιών κάθε προσώπου αποθηκεύεται σε έναν φάκελο με όνομα το όνομα του προσώπου, ανεξαρτήτως πόσες φωτογραφίες υπάρχουν στον φάκελο '''
        print("Επεξεργασία εικόνας {}/{}".format(i + 1,len(imagePaths)))
```

```
        name = imagePath.split(os.path.sep)[-2]

        # Εναλλάσονται τα κανάλια RGB σε BGR σύμφωνα με τις βιβλιοθήκες OpenCV και dLib
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # Εντοπίζονται οι συντεταγμένες των προσόπων σύμφωνα με τον αλγόριθμο Histogram Oriented Gradients
        boxes = face_recognition.face_locations(rgb,model='hog')

        # Υπολογίεται το διάνυσμα των 128 θέσεων σύμφωνα με τον αλγόριθμο Histogram Oriented Gradients
        encodings = face_recognition.face_encodings(rgb, boxes)

        for encoding in encodings:
                # Κάθε θέση της λίστας Encodings128D, με τα διανύσματα των 128 θέσεων
                Encodings128D.append(encoding)
                # αντιστοιχεί στην ίδια θέση το όνομα του προσώπου
                NamesFromEncodings.append(name)


print("Αποθήκευση διανυσμάτων και ονομάτων στο αρχείο CloudDatabaseEncodings.pickle")
''' Δημιουργούμε το λεξικό με όνομα finalData. Το κλειδί με το όνομα "encodings" θα περιέχει την λίστα με τα διανύσματα των 128 θέσεων ενώ το κλειδί με το όνομα "names" θα περιέχει τη λίστα με τα ονόματα των προσώπων στις ίδιες θέσεις με τις θέσεις των διανυσμάτων των 128 θέσεων'''
finalData = {"encodings": Encodings128D, "names": NamesFromEncodings}
f=open('/home/zouglisnik/MEGA/MyProject/CloudDataBase/CloudDatabaseEncodings.pickle', "wb")
f.write(pickle.dumps(finalData))
f.close()
print('Επιτυχής κωδικοποίηση...\n')
print('*****************************')
print('*Open University of Cyprus*')
print('* Μελέτη Ρυθμαπόδοσης MEC *')
print('*    Νικόλαος Ζουγλής    *')
print('*****************************')
```



Εικόνα 36: Διάγραμμα Ροής Δεδομένων εφαρμογής **Training.py**

Παράρτημα Β
Cloud-OUC.py

```
'''
        ΑΝΟΙΧΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
        ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΝΙΚΟΛΑΟΥ ΖΟΥΓΛΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
'''

'''
Η εφαρμογή αυτή έχει τον ρόλο του server. Κάνοντας χρήση της ενοτλής socket, λαμβάνει ένα διάνυσμα 128 θέσεων από τον MEC κόμβο, το οποίο διάνυσμα κωδικοποιεί ένα πρόσωπο. Στην συνέχεια εκτελείται η διαδικασία αναγνώρισης κάποιου ποσώπου από την βάση δεδομένων BigDatabaseEncodings.pickle. Εάν αναγνωριστεί κάποιο πρόσωπο τότε επιστρέφεται στον MEC κόμβο τα χαρακτηριστικά του δηλαδή το όνομα του προσώπου και τα διανύσματα που το χαρακτηρίζουν αυτό. Εάν δεν αναγνωριστεί κάποιο πρόσωπο τότε επιστρέφεται η τιμμή "Unknown". Το μήνυμα που θα λάβει ο server θα είναι σε μορφή bytes. Το μήνυμα θα μετατραπεί σε μορφή array 128 θέσεων για να εκτελεστεί η διαδικασία αναγνώρισης
'''
import face_recognition
import numpy
import time
import socket
import pickle
import json
import cv2
import sys

# Η σταθερά BIG_DB_FILE περιέχει την πλήρη διαδρομή και το όνομα του αρχείου της βάσης δεδομένων. Το αρχείο μπορεί να βρίσκεται οπουδήποτε στιν δίσκο, αρκεί η σταθερά να έχει την πλήρη διαδρομή του και το όνομα του
BIG_DB_FILE='/home/zouglisnik/MEGA/MyProject/CloudDataBase/CloudDatabaseEncodings.pickle'
# Ο σταθερές HOST και PORT περιέχουν την IP και το PORT του server
HOST = '127.0.0.2'
PORT = 65432
bigDataEncodings={}

# Η συνάρτηση inputData() ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της
```

βάσης δεδομένων

```
def inputData():
        f=open(BIG_DB_FILE,"rb")
        data = pickle.loads(f.read())
        f.close()
        return data

# Η συνάρτηση encodingData(msg) μετατρέπει ένα pickle object σε array
def encodingData(msg):
        y=json.loads(msg)
        y=numpy.asarray(y)
        return y

# Η συνάρτηση sendData στέλνει τα data στον MEC κόμβο
def sendData(data):
        flag=0
        if (data=="Unknown"):
                flag=1
        data=json.dumps(data)
        if (flag==0):
                print("Μέγεθος δεδομένων (bytes) :",len(bytes(data,"utf-8")))
        conn.sendall(bytes(data,"utf-8"))
```

#Η συνάρτηση grabNameAndEncodings μετατρέπει σε dictionary τα στοιχεία του προσώπου που αναγνωρίστηκε, δηλαδή σε λεξικό με κλειδιά το όνομα του προσώπου και τα διανύσματα του. Τα στοιχεία του προσώπου βρίσκονται στην βάση δεδομένων. Ωστόσο, τα διανύσματα του προσώπου θα αποσταλούν σαν απλή λίστα γιατί η βιβλιοθήκη json δεν μπορεί να κάνει 'serialize' έναν array

```
def grabNameAndEncodings(name):
        names=list()
        encodings=list()
        newData=dict()
        for i,j in enumerate(bigDataEncodings['names']):
                if (j==name):
                        names.append(j)
                        encodings.append(bigDataEncodings['encodings'][i].tolist())


        newData['encodings']=encodings
        newData['names']=names
        return newData
```

‘’’ Το dictionary 'newData' περιέχει τα στοιχεία του προσώπου που αναγνωρίστηκε δηλαδή το όνομα του προσώπου και τα διανύσματα του.’’’

Διαβάζεται η βάση δεδομένων

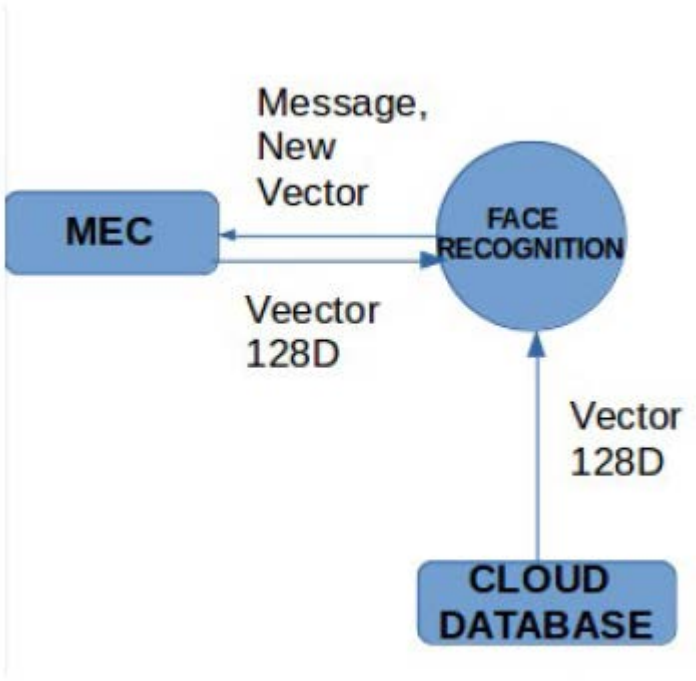
```
bigDataEncodings=inputData()

# Αρχικοποιείται ο server
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen()
conn, addr = s.accept()

# Ο server θα εκτελείται μέχρι να λάβει κάποιο σήμα από τον MEC node
while True:
        msg=''
        while True:
                data = conn.recv(1024)     # Διαβάζεται το μήνυμα που έστειλε ο MEC κόμβος
                time.sleep(0.3)
                msg=msg+data.decode("utf-8")
                if len(msg)==0 and len(data)==0:
                        sys.exit()              # Εάν το μήνυμα είναι κενό τότε ο server τερματίζει την λειτουργία του
                if len(data.decode("utf-8"))<1024:
                        break #Το μήνυμα μεταδίδεται σε πακέτα των 1024 bytes. Μόλις τελειώσει η μετάδοση τερματίζουμε το loop
        encoding=encodingData(msg) #Μετατρέπεται το μήνυμα από string σε array
        # Διαδικασία αναγνώρισης
        matches = face_recognition.compare_faces(bigDataEncodings["encodings"],encoding)
        name = "Unknown"


        if True in matches:

                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}

                for i in matchedIdxs:
                        name = bigDataEncodings["names"][i]
                        counts[name] = counts.get(name, 0) + 1

                name = max(counts, key=counts.get)
        else:
        ‘’’ Εάν δεν αναγνωριστεί το πρόσωπο τότε αποστέλεται το μήνυμα "Unknown", το loop επιστρέφει στην αρχή και ο server αναμένει μέχρι να λάβει το επόμενο
```

μήνυμα'''

```
                sendData(name)
                continue
```

''' Εάν αναγνωριστεί το πρόσωπο τότε δημιουργείται το dictionary με τα χαρακτηριστικά του προσώπου, μέσω της συνάρτησης grabNameAndEncodings'''

```
        newData=grabNameAndEncodings(name)
        print(newData['names'][0]) #Εκτυπώνεται το όνομα του προσώπου που αναγωρήστικε
        sendData(newData)          '''Αποστέλεται στον MEC κόμβο, το loop επιστρέφει στην αρχή και ο server αναμένει μέχρι να λάβει το επόμενο μήνυμα'''
```



Εικόνα 37: Διάγραμμα Ροής Δεδομένων εφαρμογής **Cloud-OUC.py**

Παράρτημα Γ
MEC-CacheLFU-OUC.py

```
'''
        ΑΝΟΙΧΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
        ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΝΙΚΟΛΑΟΥ ΖΟΥΓΛΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
'''
'''
Η εφαρμογή αυτή έχει τον ρόλο του MEC κόμβου. Ο MEC κόμβος δέχεται το πλαίσιο από το stream, εξάγει το πρόσωπο, αναγνωρίζει το πρόσωπο και κάνει caching πιο συχνά αναγνωρίσιμα πρόσωπα σύμφωνα με τον αλγόριθμου LFU. Εάν κάποιο πρόσωπο δεν αναγνωριστεί, τότε τα χαρακτηριστικά του (δηλαδή το διάνυσμα των 128 θέσεων) αποστέλονται στο Νέφος όπου αυτό θα προβεί στην αναγνώριση από μια μεγάλη βάση δεδομένων. Εάν το Νέφος αναγνωρίσει το πρόσωπο τότε ο MEC κόμβος θα λάβει τα χαρακτηριστικά του, δηλαδή ένα dictionary με το όνομα του και τα διανύσματα 128 θέσεων ειδάλλως θα λάβει "Unknown"
'''
import face_recognition
import socket
import numpy
import imutils
import pickle
import time
import cv2
import sys,os
import json
import csv

# Η σταθερά MEC_ENCODING_FILE περιέχει την πλήρη διαδρομή και το όνομα του αρχείου της βάσης δεδομένων. Το αρχείο μπορεί να βρίσκεται οπουδήποτε στιν δίσκο, αρκεί η σταθερά να έχει την πλήρη διαδρομή του και το όνομα του
MEC_ENCODING_FILE='/home/zouglisnik/MEGA/MyProject/MECEncodings.pickle'
MEC_CACHE_FILE='/home/zouglisnik/MEGA/MyProject/CacheLFU.csv'
MEC_SERVER_FILE='/home/zouglisnik/MEGA/MyProject/ServerLFU.csv'
# Cache μνήμη του MEC node
MECCache=list()
#Μέγεθος της cache
sizeOfCache=4
```

```
#Η συνάρτηση findMinimumFreq επιστρέφει το στοιχείο της cache με την ελάχιστη συχνότητα αναζήτησης
def findMinimumFreq():
	minimumFreq=MECCache[0][1]
	for i in MECCache:
		if minimumFreq>i[1]:
			minimumFreq=i[1]
	return minimumFreq

'''Η συνάρτηση findFIFO εξετάζει εάν υπάρχουν δύο ή περισσότερα στοιχεία στην cache με την ίδια ελάχιστη συχνότητα αναζήτησης. Εάν υφίσταται κάτι τέτοιο, επιστρέφει την θέση του στοιχείου που μπήκε πρώτο στην λίστα αλλιώς επιστρέφει την τιμή -1'''
def findFIFO():
	minimumFreq=findMinimumFreq()
	for i in range(0,len(MECCache)-1):
		for j in range(i+1,len(MECCache)):
			if MECCache[i][1]==MECCache[j][1] and
MECCache[i][1]==minimumFreq:
				return i
	return -1

#Προσθέτει στο τέλος της cache το όνομα που αναζήτησε ο χρήστης και αποβάλει το στοιχείο της λίστας σύμφωνα με τον αλφόριθμο LFU. Εάν το όνομα που αναζήτησε ο χρήστης ήδη βρισκεται στην cache, τότε αυξάνει κατα μία μονάδα την συχνότητα αναζήτησης αυτού και απιστρέφει την τιμή -1 αλλιώς επιστρέφει την λίστα που αποβλήθηκε από την cache. Η λίστα περιέχει για στοιχεία το ονόμα του στοιχείου που αποβλήθηκε και την συχνότητα αναζήτησης αυτού
def updateCache(name):
	for i in MECCache:
		if i[0]==name:
			i[1]+=1
			return -1
	if len(MECCache)<sizeOfCache:
		j=[]
		j.append(name)
		j.append(1)
		MECCache.append(j)
		return -1
	isItFIFO=findFIFO()
	if isItFIFO==-1:
		minimumFreq=findMinimumFreq()
```

```
            for i,j in enumerate(MECCache):
                if j[1]==minimumFreq:
                    expelled=MECCache.pop(i)
                    break
        else:
            expelled=MECCache.pop(isItFIFO)
        j=[]
        j.append(name)
        j.append(1)
        MECCache.append(j)

        return expelled

# Η συνάρτηση inputData() ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της βάσης δεδομένων
def inputData():

        f=open(MEC_ENCODING_FILE,"rb")
        data = pickle.loads(f.read())
        f.close()


        return data

# Η συνάρτηση callBigBD αποστέλει στον server το διάνυσμα του προσώπου που δεν αναγνώρισε ο fog node και αναμένει την απάντηση
def callBigBD(encoding):
        encodingToList=encoding.tolist() #Μετατρέπει τον array σε λίστα
        y=json.dumps(encodingToList) #Serializing the list
        s.sendall(bytes(y,"utf-8")) #Αποστέλονται τα δεδομένα στον server
        msg=''
        while True:
            data=s.recv(1024) #Λαμβάνεται η απάντηση από τον server
            msg=msg+data.decode("utf-8")
            if len(data.decode("utf-8"))<1024:
                break
        newData=json.loads(msg)
        if newData!='Unknown': #Εάν το πρόσωπο αναγνωρίστηκε, μετατρέπεται η λίστα που περιέχει τα διανύσματα σε array
            newData['encodings']=numpy.array(newData['encodings'])

        return newData

''' Η συνάρτηση updateEcodingFile δημιουργεί νέα βάση δεδομένων στον fog node με
```

5 πρόσωπα (την MECEncodings.pickle), συμπεριλαμβάνοντας και το πρόσωπο που αναγνωρίστηκε από τον server'''

```
def initializeEcodingFile(newData):
        newDict=dict()
        newEncodings=list()
        newNames=list()
        newData['encodings']=numpy.asarray(newData['encodings'])
        for i,j in enumerate(newData['names']):
                newEncodings.append(newData['encodings'][i])
                newNames.append(newData['names'][i])
        newDict['encodings']=newEncodings
        newDict['names']=newNames
        f=open(MEC_ENCODING_FILE,"wb")
        f.write(pickle.dumps(newDict))
        f.close()

def updateEcodingFile(newData,expelledFromCache):
        data=inputData()
        newDict=dict()
        newEncodings=list()
        newNames=list()
        newData['encodings']=numpy.asarray(newData['encodings'])
        for i,j in enumerate(data['names']):
                if (expelledFromCache!=-1):
                        if (j==expelledFromCache[0]):
                                continue #Στην νέα βάση δεδομένων δεν θα
συμπεριλαμβάνεται το πρόσωπο που έχει απορριφθεί από την cache
                newEncodings.append(data['encodings'][i])
                newNames.append(data['names'][i])
        for i,j in enumerate(newData['names']):
                newEncodings.append(newData['encodings'][i])
                newNames.append(newData['names'][i])
        newDict['encodings']=newEncodings
        newDict['names']=newNames
        f=open(MEC_ENCODING_FILE,"wb")
        f.write(pickle.dumps(newDict))
        f.close()

def cacheWriter(data,timer):
        j=list()
        j.append(data)
        j.append(timer)
        cache_writer.writerow(j)
```

```
def serverWriter(data,timer):
        j=list()
        j.append(data)
        j.append(timer)
        server_writer.writerow(j)

# Οι σταθερές HOST και PORT περιέχουν την IP και το PORT του server
HOST = '127.0.0.2'
PORT = 65432

#Σύνδεση με τον server
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

#Εισάγουμε την διαδρομή και το όνομα ενός stream αρχείου. Εάν το αρχείο δεν
υπάρχει ή πληκτρολογήσουμε 'qq' τοτε η εφαρμογή τερματίζεται
query=input("Please enter an image file (qq to exit):")
if query=='qq':
        s.close()
        sys.exit()
if os.path.exists(query)==False:
        print("No such file")
        sys.exit()

# Διαβάζεται η βάση δεδομένων και τα frames από το stream

stream = cv2.VideoCapture(query)

fc=open(MEC_CACHE_FILE,'w',newline='')
fs=open(MEC_SERVER_FILE,'w',newline='')

cache_writer=csv.writer(fc,delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
server_writer=csv.writer(fs,delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)

while True:
        foundFromServer=0
        # Διαβάζουμε το πλαίσιο εικόνας
        (grabbed, frame) = stream.read()

        # Εάν δεν υπάρχει άλλο πλαίσιο
        # τότε η εφαρμογή σταματάει
        if not grabbed:
```

```
            break

        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) #Αλλάζουμε τον χώρο
χρωμάτων
        rgb = imutils.resize(frame, width=750)
        r = frame.shape[1] / float(rgb.shape[1])

        startTime=time.time() #Έναρξη χρόνου εντοπισμού
        boxes = face_recognition.face_locations(rgb,model='hog') #Εντοπίζουμε τα
πρόσωπα στο πλαίσιο εικόνας
        encodings = face_recognition.face_encodings(rgb, boxes) #Κωδικοποιούμε τα
πρόσωπα με διανύσματα 128 θέσεων
        names = []

        for encoding in encodings:

            name = "Unknown"
            if len(MECCache)==0:
                newData=callBigBD(encoding) #Στέλνουμε αίτημα στον server
                endTime=time.time()
                if newData!="Unknown": #Εάν αναγνωρισθεί πρόσωπο ή πρόσωπα
από την βάση δεδομένων του server
                    name=newData['names'][0]
                    print("\n\n\n")
                    if (sizeOfCache>0):

        expelledFromCache=updateCache(newData['names'][0])
                        initializeEcodingFile(newData)
                        encodingData=inputData() # Η συνάρτηση
inputData() ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της νέας βάσης
δεδομένων
                        print("\n\n Νεα Cache :",MECCache)  #Εκτυπώνουμε
την cache
                    print("\n\n Χρόνος αναγνώρισης :",endTime-startTime)
        #Εκτυπώνουμε τον χρόνο αναζήτησης
                    serverWriter(name,endTime-startTime)
                    foundFromServer=1
            #Εάν η cache μνήμη έχει ήδη αρχικοποιηθεί
            else:
                encodingData=inputData()
                matches =
face_recognition.compare_faces(encodingData["encodings"],encoding)
```

```
                    # Εάν αναγνωρισθεί πρόσωπο ή πρόσωπα από την βάση
δεδομένων του fog node
                    if True in matches:
                        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                        counts = {}
                        endTime=time.time()



                        for i in matchedIdxs:
                            name = encodingData["names"][i]
                            counts[name] = counts.get(name, 0) + 1

                        name = max(counts, key=counts.get)
                        expelledFromCache=updateCache(name) #Ενημερώνουμε
την cache του fog node
                        print(MECCache) #Εκτυπώνουμε την cache
                        print(endTime-startTime) #Εκτυπώνουμε τον χρόνο
αναζήτησης
                        cacheWriter(name,endTime-startTime)
                    else:
                        # Εάν δεν βρεθεί πρόσωπο ή πρόσωπα από την βάση
δεδομένων του MEC κόμβο
                        newData=callBigBD(encoding) #Στέλνουμε αίτημα στον
server
                        endTime=time.time()
                        if newData!="Unknown": #Εάν αναγνωρισθεί πρόσωπο ή
πρόσωπα από την βάση δεδομένων του server
                            name=newData['names'][0]
                            print("\n\n\n")

        expelledFromCache=updateCache(newData['names'][0])
                            if expelledFromCache!=-1:
                                print("Αποβλήθηκε από την Cache
:",expelledFromCache) #Εκτυπώνουμε το πρόσωπο που απορρίφθηκε από την cache
                            updateEcodingFile(newData,expelledFromCache)
                            print("\n\n Νεα Cache :",MECCache)  #Εκτυπώνουμε
την cache
                            print("\n\n Χρόνος αναγνώρισης :",endTime-
startTime)    #Εκτυπώνουμε τον χρόνο αναζήτησης
                            encodingData=inputData() # Η συνάρτηση
inputData() ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της νέας βάσης
δεδομένων
                            serverWriter(newData['names'][0],endTime-
```

```
startTime)
                                        foundFromServer=1
                # Αναβαθμίζουμε την λίστα με τα ονόματα
                names.append(name)

                for ((top, right, bottom, left), name) in zip(boxes, names):

                        top = int(top * r)
                        right = int(right * r)
                        bottom = int(bottom * r)
                        left = int(left * r)

                        # Σχεδιάζουμε το πλαίσιο που περικλείει το πρόσωπο
                        cv2.rectangle(frame, (left, top), (right, bottom),(0, 255, 0), 2)
                        y = top - 15 if top - 15 > 15 else top + 15
                        cv2.putText(frame, name, (left, y),
cv2.FONT_HERSHEY_SIMPLEX,0.75, (0, 255, 0), 2)

        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
                break

stream.release()
s.close()
fc.close()
fs.close()
```

Παράρτημα Δ
MEC-CacheTimer-OUC.py

'''

ΑΝΟΙΧΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΝΙΚΟΛΑΟΥ ΖΟΥΓΛΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

'''

'''

Η εφαρμογή αυτή έχει τον ρόλο του MEC κόμβου. Ο MEC κόμβος δέχεται το πλαίσιο από το stream, εξάγει το πρόσωπο, αναγνωρίζει το πρόσωπο και κάνει caching πιο συχνά αναγνωρίσιμα πρόσωπα σύμφωνα με τον αλγόριθμου Χρονοκαθυστέρησης. Εάν κάποιο πρόσωπο δεν αναγνωριστεί, τότε τα χαρακτηριστικά του (δηλαδή το διάνυσμα των 128 θέσεων) αποστέλονται στο Νέφος όπου αυτό θα προβεί στην αναγνώριση από μια μεγάλη βάση δεδομένων. Εάν το Νέφος αναγνωρίσει το πρόσωπο τότε ο MEC κόμβος θα λάβει τα χαρακτηριστικά του, δηλαδή ένα dictionary με το όνομα του και τα διανύσματα 128 θέσεων ειδάλλως θα λάβει "Unknown"

'''

```
import face_recognition
import socket
import numpy
import imutils
import pickle
import time
import cv2
import sys,os
import json
import csv
```

‘’’ Η σταθερά MEC_ENCODING_FILE περιέχει την πλήρη διαδρομή και το όνομα του αρχείου της βάσης δεδομένων. Το αρχείο μπορεί να βρίσκεται οπουδήποτε στιν δίσκο, αρκεί η σταθερά να έχει την πλήρη διαδρομή του και το όνομα του’’’

```
MEC_ENCODING_FILE='/home/zouglisnik/MEGA/MyProject/MECEncodings.pickle'
MEC_CACHE_FILE='/home/zouglisnik/MEGA/MyProject/CacheTimer.csv'
MEC_SERVER_FILE='/home/zouglisnik/MEGA/MyProject/ServerTimer.csv'
# Cache μνήμη του MEC κόμβου
MECCache=list()
#Μέγεθος της cache
sizeOfCache=4
timer=time.time()

def myF(myCache):

    return myCache[1]
```

‘’’Προσθέτει στο τέλος της cache το όνομα που αναζήτησε ο χρήστης και αποβάλει το στοιχείο της λίστας σύμφωνα με τον αλφόριθμο LFU. Εάν το όνομα που αναζήτησε ο χρήστης ήδη βρισκεται στην cache, τότε αυξάνει κατα μία μονάδα την συχνότητα αναζήτησης αυτού και απιστρέφει την τιμή -1 αλλιώς επιστρέφει την λίστα που αποβλήθηκε από την cache. Η λίστα περιέχει για στοιχεία το ονόμα του στοιχείου που αποβλήθηκε και την συχνότητα αναζήτησης αυτού’’’

```
def updateCache(name):
    for i in MECCache:
        if i[0]==name:
            i[1]=time.ctime()
            MECCache.sort(reverse=True,key = myF)
            return -1
    if len(MECCache)<sizeOfCache:
        j=[]
        j.append(name)
        j.append(time.ctime())
        MECCache.append(j)
        MECCache.sort(reverse=True,key = myF)
        return -1
    j=list()
```

```
        j.append(name)
        j.append(time.ctime())
        expelled=MECCache.pop(sizeOfCache-1)
        MECCache.append(j)
        MECCache.sort(reverse=True,key = myF)
        return expelled

# Η συνάρτηση inputData() ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της βάσης δεδομένων
def inputData():

        f=open(MEC_ENCODING_FILE,"rb")
        data = pickle.loads(f.read())
        f.close()

        return data

# Η συνάρτηση callBigBD αποστέλει στον server το διάνυσμα του προσώπου που δεν αναγνώρισε ο MEC κόμβος και αναμένει την απάντηση
def callBigBD(encoding):
        encodingToList=encoding.tolist() #Μετατρέπει τον array σε λίστα
        y=json.dumps(encodingToList) #Serializing the list
        s.sendall(bytes(y,"utf-8")) #Αποστέλονται τα δεδομένα στον server
        msg=''
        while True:
                data=s.recv(1024) #Λαμβάνεται η απάντηση από τον server
                msg=msg+data.decode("utf-8")
                if len(data.decode("utf-8"))<1024:
                        break
        newData=json.loads(msg)
        if newData!='Unknown': #Εάν το πρόσωπο αναγνωρίστηκε, μετατρέπεται η λίστα που περιέχει τα διανύσματα σε array
                newData['encodings']=numpy.array(newData['encodings'])
```

```
        return newData
```

''' Η συνάρτηση updateEcodingFile δημιουργεί νέα βάση δεδομένων στον MEC κόμβο με 5 πρόσωπα (την MECEncodings.pickle), συμπεριλαμβάνοντας και το πρόσωπο που αναγνωρίστηκε από τον server'''

```
def initializeEcodingFile(newData):
        newDict=dict()
        newEncodings=list()
        newNames=list()
        newData['encodings']=numpy.asarray(newData['encodings'])
        for i,j in enumerate(newData['names']):
                newEncodings.append(newData['encodings'][i])
                newNames.append(newData['names'][i])
        newDict['encodings']=newEncodings
        newDict['names']=newNames
        f=open(MEC_ENCODING_FILE,"wb")
        f.write(pickle.dumps(newDict))
        f.close()

def updateEcodingFile(newData,expelledFromCache):
        data=inputData()
        newDict=dict()
        newEncodings=list()
        newNames=list()
        newData['encodings']=numpy.asarray(newData['encodings'])
        for i,j in enumerate(data['names']):
                if (expelledFromCache!=-1):
                        if (j==expelledFromCache[0]):
                                continue #Στην νέα βάση δεδομένων δεν θα συμπεριλαμβάνεται το πρόσωπο που έχει απορριφθεί από την cache
                newEncodings.append(data['encodings'][i])
                newNames.append(data['names'][i])
```

```
        for i,j in enumerate(newData['names']):
            newEncodings.append(newData['encodings'][i])
            newNames.append(newData['names'][i])
        newDict['encodings']=newEncodings
        newDict['names']=newNames
        f=open(MEC_ENCODING_FILE,"wb")
        f.write(pickle.dumps(newDict))
        f.close()

def cacheWriter(data,timer):
        j=list()
        j.append(data)
        j.append(timer)
        cache_writer.writerow(j)

def serverWriter(data,timer):
        j=list()
        j.append(data)
        j.append(timer)
        server_writer.writerow(j)

# Οι σταθερές HOST και PORT περιέχουν την IP και το PORT του server
HOST = '127.0.0.2'
PORT = 65432

#Σύνδεση με τον server
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

#Εισάγουμε την διαδρομή και το όνομα ενός stream αρχείου. Εάν το αρχείο δεν υπάρχει ή πληκτρολογήσουμε 'qq' τοτε η εφαρμογή τερματίζεται
query=input("Please enter an image file (qq to exit):")
if query=='qq':
```

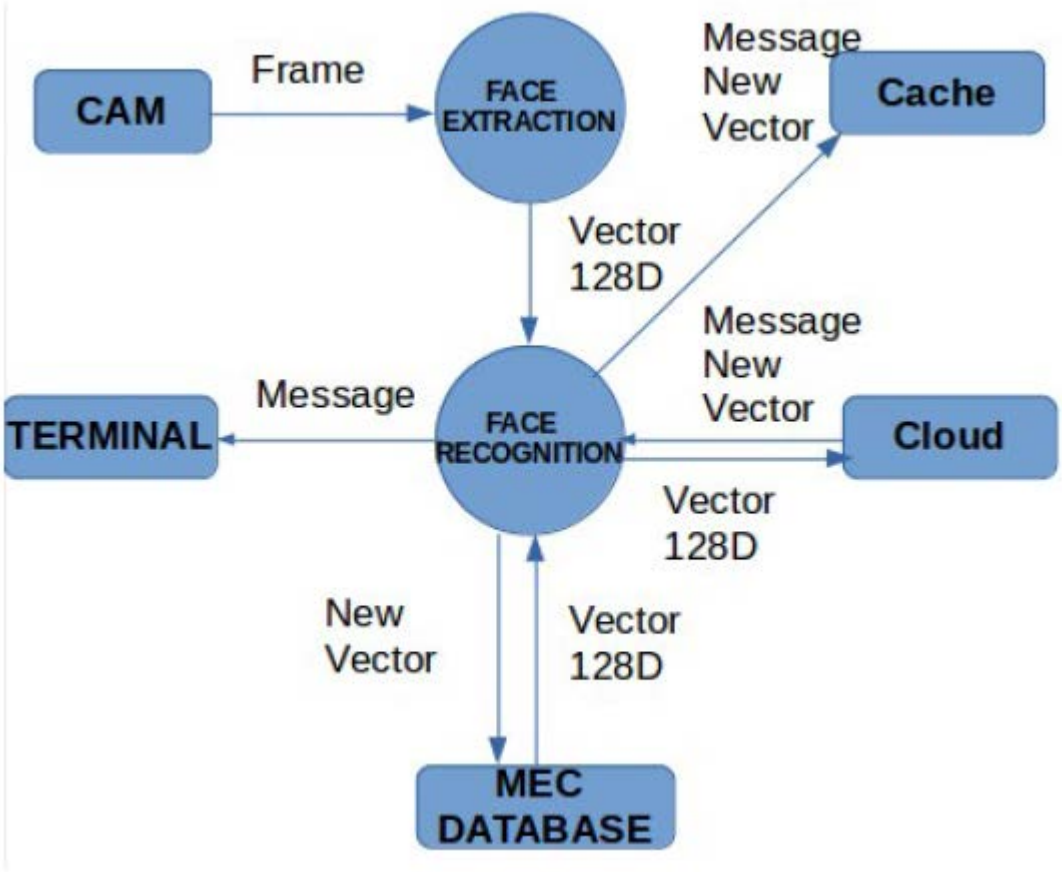
```
        s.close()
        sys.exit()
if os.path.exists(query)==False:
        print("No such file")
        sys.exit()

# Διαβάζεται η βάση δεδομένων και τα frames από το stream
stream = cv2.VideoCapture(query)

fc=open(MEC_CACHE_FILE,'w',newline='')
fs=open(MEC_SERVER_FILE,'w',newline='')

cache_writer=csv.writer(fc,delimiter=',',quotechar='"',quoting=csv.QUOTE_MINIMAL)
server_writer=csv.writer(fs,delimiter=',',quotechar='"',quoting=csv.QUOTE_MINIMAL)

while True:
        foundFromServer=0
        # Διαβάζουμε το πλαίσιο εικόνας
        (grabbed, frame) = stream.read()

        # Εάν δεν υπάρχει άλλο πλαίσιο
        # τότε η εφαρμογή σταματάει
        if not grabbed:
                break

        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) #Αλλάζουμε τον χώρο χρωμάτων
        rgb = imutils.resize(frame, width=750)
        r = frame.shape[1] / float(rgb.shape[1])

        startTime=time.time() #Έναρξη χρόνου εντοπισμού
        boxes = face_recognition.face_locations(rgb,model='hog') #Εντοπίζουμε τα πρόσωπα στο πλαίσιο εικόνας
```

```
        encodings = face_recognition.face_encodings(rgb, boxes) #Κωδικοποιούμε τα πρόσωπα με διανύσματα 128 θέσεων
        names = []

        for encoding in encodings:

                if len(MECCache)==0:
                        newData=callBigBD(encoding) #Στέλνουμε αίτημα στον server
                        endTime=time.time()
                        if newData!="Unknown": #Εάν αναγνωρισθεί πρόσωπο ή πρόσωπα από την βάση δεδομένων του server
                                name=newData['names'][0]
                                print("\n\n\n")
                                if (sizeOfCache>0):

        expelledFromCache=updateCache(newData['names'][0])
                                        initializeEcodingFile(newData)
                                        encodingData=inputData() # Η συνάρτηση inputData() ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της νέας βάσης δεδομένων
                                        print("\n\n Νεα Cache :",MECCache)  #Εκτυπώνουμε την cache
                                print("\n\n Χρόνος αναγνώρισης :",endTime-startTime)
        #Εκτυπώνουμε τον χρόνο αναζήτησης
                                serverWriter(name,endTime-startTime)
                                foundFromServer=1
                #Εάν η cache μνήμη έχει ήδη αρχικοποιηθεί
                else:
                        encodingData=inputData()
                        matches = face_recognition.compare_faces(encodingData["encodings"],encoding)
                        name = "Unknown"
                        # Εάν αναγνωρισθεί πρόσωπο ή πρόσωπα από την βάση δεδομένων του MEC κόμβου
```

```
                if True in matches:
                        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                        counts = {}
                        endTime=time.time()


                        for i in matchedIdxs:
                                name = encodingData["names"][i]
                                counts[name] = counts.get(name, 0) + 1

                        name = max(counts, key=counts.get)
                        expelledFromCache=updateCache(name) #Ενημερώνουμε την cache του MEC κόμβου
                        print(MECCache) #Εκτυπώνουμε την cache
                        print(endTime-startTime) #Εκτυπώνουμε τον χρόνο αναζήτησης
                        cacheWriter(name,endTime-startTime)
                else:
                        # Εάν δεν βρεθεί πρόσωπο ή πρόσωπα από την βάση δεδομένων του MEC κόμβου
                        newData=callBigBD(encoding) #Στέλνουμε αίτημα στον server
                        endTime=time.time()
                        if newData!="Unknown": #Εάν αναγνωρισθεί πρόσωπο ή πρόσωπα από την βάση δεδομένων του server
                                name=newData['names'][0]
                                print("\n\n\n")

        expelledFromCache=updateCache(newData['names'][0])
                                if expelledFromCache!=-1:
                                        print("Αποβλήθηκε από την Cache :",expelledFromCache) #Εκτυπώνουμε το πρόσωπο που απορρίφθηκε από την cache
                                updateEcodingFile(newData,expelledFromCache)
```

```
                                print("\n\n Νεα Cache :",MECCache)  #Εκτυπώνουμε
την cache
                                print("\n\n Χρόνος αναγνώρισης :",endTime-
startTime)  #Εκτυπώνουμε τον χρόνο αναζήτησης
                                encodingData=inputData() # Η συνάρτηση inputData()
ανοίγει και διαβάζει τα περιεχόμενα του αρχείου της νέας βάσης δεδομένων
                                serverWriter(name,endTime-startTime)
                                foundFromServer=1
            # Αναβαθμίζουμε την λίστα με τα ονόματα
            names.append(name)

            for ((top, right, bottom, left), name) in zip(boxes, names):

                top = int(top * r)
                right = int(right * r)
                bottom = int(bottom * r)
                left = int(left * r)

                # Σχεδιάζουμε το πλαίσιο που περικλείει το πρόσωπο
                cv2.rectangle(frame, (left, top), (right, bottom),(0, 255, 0), 2)
                y = top - 15 if top - 15 > 15 else top + 15
                cv2.putText(frame,          name,          (left,          y),
cv2.FONT_HERSHEY_SIMPLEX,0.75, (0, 255, 0), 2)

        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
            break

stream.release()
s.close()
fc.close()
```

fs.close()



Εικόνα 38: Διάγραμμα Ροής Δεδομένων εφαρμογών **MEC-CacheLFU-OUC.py** και **MEC-CacheTimer-OUC.py**

Βιβλιογραφία

[1] Tarik Taleb, Konstantinos Samdanis, Badr Mada,Hannu Flinck, Sunny Dutta, and Dario Sabella, 2017, *On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration*, IEEE Communications Surveys and Tutorials, Vol. 19

[2] Sonia Shahzadi, Muddesar Iqbal, Tasos Dagiuklas and Zia Ul Qayyum, 2017, *Multi-access edge computing: open issues, challenges and future perspectives*, Journal of Cloud Computing: Advances, Systems and Applications

[3] Robert K. McConnell, 1986, *United States Patent 4,567,610,* WAYLAND RES Inc.

[4] Mahadev Satyanarayanan, Victor Bahl, Ramón Cáceres, Nigel Davies, 2009, *The Case for VM-Based Cloudlets in Mobile Computing*, Vol.8, Issue 4, p. 14-23

[5] Navneet Dalal, Bill Triggs, 2005, *Histograms of Oriented Gradients for Human Detection,* IEEE 2005 Computer Society Conference on Computer Vision and Pattern Recognition

[6] *Multi-access Edge Computing (MEC);Framework and Reference Architecture*, 2019, ETSI GS MEC 003 V2.1.1 (2019-01)