Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Πλεονεκτήματα οχηματικών δικτύων 5G στηριζόμενα σε αρχιτεκτονική SDN** **Advantages of 5G VANETs on Software Defined Vehicular Networks**
Ονοματεπώνυμο Φοιτητή	**Ριζόπουλος Παναγιώτης**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΣΠ16027**
Επιβλέπων	**Άγγελος Μιχάλας , Εξωτερικός Συνεργάτης Πανεπιστημίου Πειραιά**

Ημερομηνία Παράδοσης Δεκέμβριος 2019

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Δημήτριος Βέργαδος Αναπληρωτής Καθηγητής	Δουληγέρης Χρήστος Καθηγητής	Άγγελος Μιχάλας Εξωτερικός Συνεργάτης Πανεπιστημίου Πειραιά

Τίτλος

Πλεονεκτήματα οχηματικών δικτύων 5G στηριζόμενα σε αρχιτεκτονική SDN

Περίληψη

Τα ασύρματα και κινητά δίκτυα έχουν περάσει μέσα από πολλές αλλαγές τα τελευταία χρόνια προκειμένου να εξυπηρετήσουν τις όλο και αυξανόμενες ανάγκες των χρηστών τους, γίνονται ολοένα και πιο απαιτητικές σε εύρος ζώνης. Η απαίτηση για μεγάλες ταχύτητες, χαμηλότερες καθυστερήσεις και αξιοπιστία υπηρεσιών τείνει να μεγαλώνει με την πάροδο του χρόνου, κυρίως στις εφαρμογές που χρησιμοποιούνται στα οχηματικά δίκτυα (VANETS). Το δίκτυο κινητών επικοινωνιών πέμπτης γενιάς (5G), η εφαρμογή του οποίου βρίσκεται προ των πυλών, υπόσχεται ότι θα μπορέσει να ανταποκριθεί σε αυτές τις απαιτήσεις και ότι θα αλλάξει ριζικά τον τρόπο δικτύωσης.

Στην παρούσα εργασία προτείνουμε μια αρχιτεκτονική VANET, η οποία υποστηρίζει network slicing σε V2X επικοινωνίες. Η αρχιτεκτονική αυτή βασίζεται σε τεχνολογίες λογισμικοποίησης δικτύων (network softwarization technologies), όπως το SDN και το NFV. Την αρχιτεκτονική αυτή, την υλοποιήσαμε σε περιβάλλον προσομοίωσης με την βοήθεια του προγράμματος Mininet–Wifi και δημιουργήσαμε έναν αλγόριθμο για την υλοποίηση network slicing με σκοπό την αυτοματοποίηση της διαδικασίας της διαπομπής (handover) μεταξύ των FOG από τους SDN controllers, σύμφωνα με τον φόρτο χρήσης του κάθε FOG.

Ελέγξαμε την καθυστέρηση (delay), τη χρήση του εύρους ζώνης (bandwidth) και το ποσοστό απώλειας πακέτων (packet loss ratio) με τη χρήση slicing και χωρίς χρήση slicing και διαπιστώσαμε καθυστέρηση σε t=120, 15ms (χωρίς χρήση slicing) έναντι 5ms (με χρήση slicing), εύρος ζώνης για την υπηρεσία μετάδοσης video από server, 20Mbps (χωρίς χρήση slicing) έναντι 25Mbps (με χρήση slicing) και ποιότητα υπηρεσίας video μειωμένη κατά 50% φτάνοντας στο όχημα εξαιτίας απώλειας πακέτων (χωρίς χρήση slicing) έναντι ποσοστού απώλειας πακέτων 10%-20% (με χρήση slicing).

Λέξεις κλειδιά: V2X, 5G, MEC, SDN, VANET, FOG

Title

Advantages of 5G VANETs on Software Defined Vehicular Networks

Abstract

Wireless and mobile networks have been through major evolutions during last decades to meet with the increasing demands of the end users and the global market of telecommunications, in particular regarding the demands in bandwidth, high-speed connections, less delays and increased reliability of services, mostly in the applications used in VANETs. 5G Mobile Networks, a technology to be implemented widely in the near future, promises to fulfil all these demands and to radically change the mode of networking.

This paper is proposing a VANET architecture that supports network slicing in V2X communications based in network softwarization technologies, such as SDN and NFV. We have implemented this architecture in a simulation using Mininet–Wifi and creating an algorithm for implementing network slicing aiming to the automation of the handover process between FOGs by SDN controllers, depending on the load usage in each FOG.

We tested the delay, the bandwidth and the packet loss ratio with slicing and no slicing process and have come to the conclusion that on t=120, delay was 15ms (without using slicing) versus 5ms (when using slicing), bandwidth for video streaming service from the server was 20Mbps (without using slicing) versus 25Mbps (when using slicing) and Quality of Service for the video regarding the packet loss ratio 50% (without using slicing) versus 10%-20% (when using slicing).

Keywords: V2X; 5G; MEC; SDN; VANET;FOG

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Άγγελο Μιχάλα για την καθοδήγηση, τη συνεργασία και την αμέριστη υποστήριξή του καθ' όλη τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας καθώς και τον β' επιβλέποντα κ. Δημήτρη Βέργαδο για τις πολύτιμες συμβουλές και παρατηρήσεις του.

Ευχαριστώ ιδιαίτερα τον κ. Ramon Fontes, Developer του προγράμματος Mininet-WiFi για την άμεση ανταπόκριση και τις χρήσιμες διευκρινήσεις που μου παραχώρησε κατά την πειραματική διαδικασία.

Περιεχόμενα

ΠΙΝΑΚΑΣ ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

3GPP - 3rd Generation Partnership Project

AMPS - Advanced Mobile Phone System

C2B - Car to Broadband Cloud

C2C - Car to Car

C2I - Car to Infrastructure

CoMP - Coordinated MultiPoint

EDGE - Enhanced Data Rates for Global evolution

EGPRS - Enhanced General Packet Radio Service

FDD - Frequency Division Duplexing

FDMA – Frequency Division Multiplexing Access

GPRS - General Packet Radio Service

GPS - Global Positioning System

GSM – Global System for Mobile Communication

IMT-2000 - International Mobile Telecommunications 2000

ITU-R - International Telecommunications Union, Radio Communications

LTE-A - Long-term evolution advanced

MANET - Mobile Ad-hoc Network

MC-CDMA – Multi-Carrier Code Division Multiple Access

MEC - Mobile Edge Computing

MEH - Mobile Edge Host

MEP – Mobile Edge Platform

mMTC - Massive Machine-Type Communication

MTC - Machine Type Communication

NB-IoT - Narrow-Band IoT

NFV - Network Functions Virtualization

NFVI - Network Functions Virtualization Infrastructure

NLoS – Non-Line of Sight

OFDM – Orthogonal Frequency Division Multiplexing

OFDMA - Orthogonal Frequency Division Multiple Access

ONF - Open Networking Foundation

QoS - Quality of Service

RAN – Radio Access Network

RAT – Radio Access Technology

RNC – Radio Network Controller

RSU – Road-Side Unit

SAE/EPC - System Architecture Evolution/Evolved Packet Core

SDN - Software-Defined Networking

TACS - Total Access Communications System

TD-CDMA - Time Division CDMA

TDD - Time Division Duplexing

TDMA – Time Division Multiplexing Access

uMTC – ultra-reliable Machine-Type Communication

UMTS – Universal Mobile Telecommunications System

V2B - Vehicle to Broadband Cloud

V2I - Vehicle to Infrastructure

V2V - Vehicle to Vehicle

V2X - Vehicle to Everything

VANET - Vehicular Ad-hoc Network

VNF - Virtualized Network Functions

WAVE - Wireless Access in Vehicular Environments

WCDMA - Wideband Code Division Multiple Access

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΓΡΑΦΗΜΑΤΩΝ

Εισαγωγή

Βρισκόμαστε σε μια εποχή όπου η ανάπτυξη καινοτόμων τεχνολογιών και εφαρμογών γίνεται με ταχύτατους ρυθμούς. Κύρια παραδείγματα είναι οι επενδύσεις των μεγάλων αυτοκινητοβιομηχανιών στο να κάνουν όσο πιο αυτόνομα μπορούν τα αυτοκίνητα αναπτύσσοντας εφαρμογές V2X(Vehicle to Everything), θέτοντας έτσι μια πληθώρα από καινούριες απαιτήσεις στα ήδη υπάρχοντα οχηματικά δίκτυα (VANETs).

Αν και τα σημερινά VANETs, για να καλύψουν αποτελεσματικά τις ανάγκες των χρηστών τους, χρησιμοποιούν ευέλικτες αρχιτεκτονικές, όπως το Software Define networks (SDN) και Network Function Virtualization (NFV), καθώς και τεχνολογίες cloud computing, όπως Fog Computing και Edge Computing, οι απαιτήσεις αυξάνονται με εκθετικό ρυθμό.

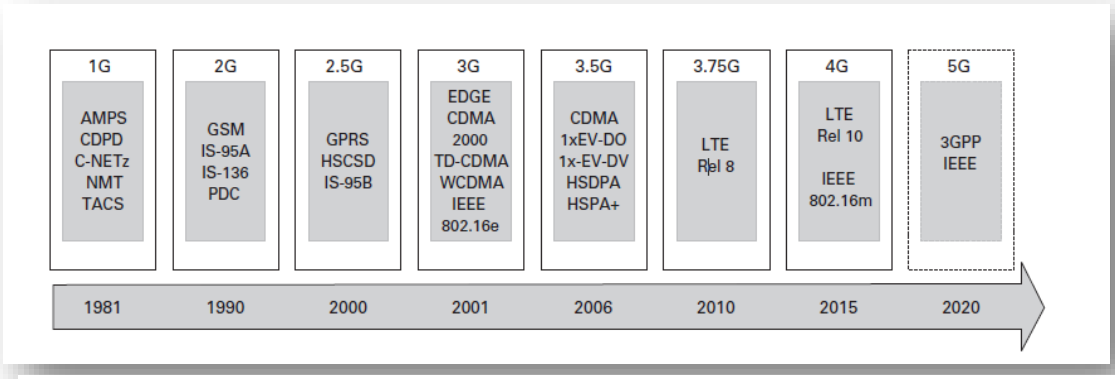
Τις απαιτήσεις αυτές έρχεται να καλύψει η επόμενη γενιά κινητών δικτύων 5G. Η τεχνολογία 5G είναι η πέμπτη γενιά της τεχνολογίας ευρυζωνικών κυψελοειδών δικτύων, η οποία διαδέχθηκε τα 4G (LTE/WiMax), 3G (UMTS) και 2G (GSM). Κύριος στόχος των δικτύων πέμπτης γενιάς είναι ο υψηλός ρυθμός μετάδοσης δεδομένων, η μειωμένη καθυστέρηση, η μεγαλύτερη γεωγραφική κάλυψη καθώς επίσης η αυξημένη ασφάλεια και η αξιοπιστία.

Σκοπός της παρούσας εργασίας είναι να διερευνήσει ποιες είναι οι βελτιώσεις και αναθεωρήσεις που χρειάζεται να αναπτυχθούν σε αρχιτεκτονικές SDN και NFV καθώς και στις τεχνολογίες cloud computing, που ήδη χρησιμοποιούνται στα VANETs, ώστε να ανταποκρίνονται στις σύγχρονες ανάγκες χρήσης των υπηρεσιών των δικτύων επόμενης γενιάς 5G. Ειδικότερα, εστιάζει στην ενσωμάτωση της εφαρμογής *network slicing* και ελέγχεται, σε συνθήκες προσομοίωσης, η ενσωμάτωση της εφαρμογής *τεμάχισης δικτύου* (network slicing) σε πρωτόκολλα επικοινωνίας *Vehicle to Everything (V2X)* στα Vehicular Ad-hoc Network αλλά και να αναδείξει τις νέες δυνατότητες στο παρόν και στο μέλλον τόσο για τις υποδομές όσο και για την εμπειρία του τελικού χρήστη.

Η παρούσα εργασία διαρθρώνεται σε επτά κεφάλαια. Στο πρώτο κεφάλαιο γίνεται αναφορά στην εξέλιξη των τεχνολογιών (γενιές) κινητής κυψελοειδούς επικοινωνίας 1^{ης}, 2^{ης} και 3^{ης} γενιάς (1G, 2G και 3G, αντίστοιχα) και στα πλαίσια που τις υποστηρίζουν και στις ενδιάμεσες βελτιωμένες εκδόσεις τους 2.5G και 3.5G, αντίστοιχα, ως τεχνολογίες βάσης για τα πιο προηγμένα συστήματα που ακολούθησαν, όπως το 4G, που χρησιμοποιείται λειτουργικά έως και σήμερα και το 5G, που ήδη έχει αναπτυχθεί αλλά έχει εφαρμοστεί μόνο σε πειραματικό στάδιο προς το παρόν, ωστόσο αποτελεί την καινοτόμο πρόταση του μέλλοντος σε αυτού του είδους τις τεχνολογίες. Στο δεύτερο κεφάλαιο παρουσιάζεται η τεχνολογία VANETs, μια σχετικά νέα και καινοτόμος τεχνολογία στα κινητά ασύρματα δίκτυα με

καθοριστικό ρόλο για τον διαμοιρασμό δεδομένων μεταξύ οχημάτων που διαθέτουν ασύρματες διεπιφάνειες (ITS), τα οποία βρίσκονται στο επίκεντρο του ενδιαφέροντος της παρούσας εργασίας. Το τρίτο κεφάλαιο αναφέρεται στις τεχνολογίες Software-defined networking (SDN) και Network Functions Virtualization (NFV), που αποτελούν δομικά συστατικά της εύρυθμης λειτουργίας και των προηγμένων δυνατοτήτων των VANETs και επιτρέπουν εν πολλοίς, το μεν SDN τον συντονισμό και αυτοματισμό των διαδικτυακών υπηρεσιών, το δε NFV επιτρέπει στα λογισμικά να παρέχουν καινοτόμες υπηρεσίες σε υψηλές ταχύτητες. Ειδική αναφορά γίνεται στις δυνατότητες που παρέχει το υπολογιστικό νέφος (cloud computing) και στο ρόλο του στα προηγμένα κινητά δίκτυα ασύρματης επικοινωνίας. Ακολουθεί το Τέταρτο κεφάλαιο, όπου παρουσιάζονται καινοτόμες τεχνολογίες που αναμένεται να υλοποιηθούν από τα 5G VANETs στο άμεσο μέλλον. Στο Πέμπτο κεφάλαιο παρουσιάζεται το σενάριο εργασίας και τα εργαλεία που χρησιμοποιούνται για την υλοποίησή του και συζητείται αναλυτικά κάθε στάδιο. Στο Έκτο κεφάλαιο συζητώνται τα αποτελέσματα της πειραματικής διαδικασίας. Η εργασία ολοκληρώνεται με το έβδομο κεφάλαιο που περιλαμβάνει τα συμπεράσματα και τις προτάσεις για μελλοντική έρευνα. Στο Παράρτημα παρατίθενται (α) οι απαιτήσεις για την εγκατάσταση του προγράμματος Mininet-WiFi, (β) οι οδηγίες εγκατάστασης του ίδιου προγράμματος, (γ) ο κώδικας για την υλοποίηση του σεναρίου εργασίας και, (δ) ο κώδικας για την υλοποίηση του network slicing.

1 – Η εξέλιξη των Κινητών Δικτύων



Εικόνα 1. *Χρονολόγιο της ιστορικής εξέλιξης των κυψελοειδών συστημάτων επικοινωνίας από την εμφάνισή τους στη δεκαετία 1970 (1G, τεχνολογία πρώτης γενιάς) ως τη δεκαετία 2020 (5G, τεχνολογία πέμπτης γενιάς).*

Τα κυψελοειδή συστήματα επικοινωνίας από την εμφάνισή τους στη δεκαετία 1970 έως και σήμερα αποτελούν τεχνολογία αιχμής και έχουν συμβάλλει σημαντικά στην ανάπτυξη νέων καινοτόμων προϊόντων υψηλής προστιθέμενης αξίας. Πρόκειται για μια τεχνολογία διαρκώς εξελισσόμενη, η οποία έχει τη δυνατότητα να ενσωματώνει λειτουργίες και χαρακτηριστικά που ανταποκρίνονται άμεσα και αποτελεσματικά στις απαιτήσεις και τις ανάγκες του καταναλωτικού κοινού, καθώς αυτές τροποποιούνται με την πάροδο των ετών, ενώ διατηρεί τις δυνατότητές της για διαρκή βελτίωση, όπως φαίνεται και από τον παραπάνω πίνακα (Εικόνα 1). Στην Εικόνα 1, παρουσιάζονται σε χρονολογική σειρά οι τεχνολογίες ορόσημα στην εξέλιξη των κυψελοειδών συστημάτων επικοινωνίας, για τις οποίες θα μιλήσουμε αναλυτικότερα στις ενότητες που ακολουθούν.

1.1 Κινητά Δίκτυα 1ης, 2ης και 3ης γενιάς

Οι δεκαετίες 1950 και 1960 αποτέλεσαν σταθμό για τα συστήματα κινητής επικοινωνίας, αφού τότε αναπτύχθηκαν για πρώτη φορά και άρχισαν να τίθενται σε εφαρμογή τα πρώτα εμπορικά αναλογικά συστήματα κινητής επικοινωνίας (ICT-317669 METIS project – Deliv. D1.1, 2013), αν και με χαμηλή ακόμα ικανότητα κάλυψης. Το 1981 εμφανίζονται για πρώτη φορά εμπορικές εφαρμογές με βάση πρότυπα κινητής κυψελοειδούς επικοινωνίας, τα λεγόμενα Πρώτης Γενιάς (1G).

Τέτοια ήταν το Nordic Mobile Telephone (NMT) στις σκανδιναβικές χώρες, το Total Access Communications System (TACS) στο Ηνωμένο Βασίλειο και το Advanced Mobile Phone System (AMPS) στην Αμερική. Τα πρότυπα αυτά χρησιμοποιούν ραδιοσήματα διαμορφωμένης συχνότητας σε κανάλι ψηφιακής σηματοδοσίας, πρόκειται με άλλα λόγια για αναλογική τεχνολογία και για αυτό ονομάζονται αναλογικά πρότυπα (Osseiran et al., 2016).

Το έτος 1982 ήταν καθοριστικό για την ραγδαία εξάπλωση της νέας αυτής τεχνολογίας στις επικοινωνίες. Η απόφαση της Ευρωπαϊκής Διάσκεψης των Ταχυδρομικών και Τηλεπικοινωνιακών Οργανισμών (CEPT) για την ανάγκη ανάπτυξης ενός πανευρωπαϊκού συστήματος κινητής επικοινωνίας Δεύτερης γενιάς (2G) αποτέλεσε το εφαλτήριο για την κυριαρχία της τεχνολογίας συστημάτων 2G. Η κυριαρχία των συστημάτων κινητής επικοινωνίας 2G από το 1991 και μετά οδήγησε στη δημιουργία του Παγκόσμιου Συστήματος Κινητών Επικοινωνιών (GSM), πρωταρχικός σκοπός του οποίου ήταν η δημιουργία ενός κοινού τηλεφωνικού δικτύου φωνητικής τηλεφωνίας, η οποία θα επέτρεπε τη διεθνή περιαγωγή (roaming) σε ολόκληρη την Ευρώπη. Το σύστημα GSM χρησιμοποιεί τη μέθοδο Πολλαπλής Πρόσβασης Διαίρεσης Χρόνου (TDMA)/ Πολλαπλής Πρόσβασης με Διαίρεση Συχνότητας (FDMA). Μια σημαντική διαφοροποίηση από τα συστήματα 1G, που χρησιμοποιούσαν μόνο FDMA (Report ITUR M.2134, 2008), κάτι που δικαιολογεί το μεγάλο ενδιαφέρον και τη ραγδαία εξάπλωσή τους και πέρα από τα ευρωπαϊκά σύνορα με τα νέα αυτά επικοινωνιακά συστήματα ψηφιακής μετάδοσης και μεταγωγής να αρχίσουν να χρησιμοποιούνται διεθνώς (Osseiran et al., 2016). Πλέον η ψηφιακή τεχνολογία έδινε νέες δυνατότητες που επέτρεπαν σημαντικές βελτιώσεις σε υπηρεσίες των συστημάτων, όπως στην ποιότητα φωνής και τη χωρητικότητα δικτύου καθώς και στην ανάπτυξη βοηθητικών υπηρεσιών αλλά και προηγμένων εφαρμογών, όπως για παράδειγμα η Υπηρεσία Σύντομων Μηνυμάτων (SMS) για την αποθήκευση και την προώθηση πληροφοριών με τη μορφή κειμένων.

Η επαναστατική εξέλιξη στην τεχνολογία 2G εμφανίζεται με την έκδοση 2.5G που έχει τη δυνατότητα να υποστηρίζει υπηρεσίες δεδομένων πακετομεταγωγής (packet-switched data services), οι οποίες έρχονται να προστεθούν στις υπηρεσίες μετάδοσης φωνής και δεδομένων κυκλωματομεταγωγής (voice and circuit-switched data). Η εξελιγμένη αυτή μορφή της τεχνολογίας 2G, ονομάστηκε 2.5G, καθώς

διατηρεί αρκετά από τα χαρακτηριστικά 2G και εισάγει νέα. Το βασικό πρότυπο στην τεχνολογία 2.5G είναι το General Packet Radio Service (GPRS), το οποίο αποτελεί επέκταση του GSM, στο οποίο αναφερθήκαμε παραπάνω. Οι απαιτήσεις στην αγορά αλλά και οι ολοένα και πιο βελτιωμένες δυνατότητες που μπορούσε να παρέχει κάθε νέα εξέλιξη σε αυτόν τον τομέα, πολύ γρήγορα ώθησε στην ανάπτυξη νέων συστημάτων, που το ένα βασιζόταν στο άλλο, όπως το Enhanced Data Rates for Global evolution (EDGE) και το Enhanced General Packet Radio Service (EGPRS). Ένα από τα κύρια νέα χαρακτηριστικά αυτής της γενιάς ήταν η προσθήκη προγραμμάτων διαμόρφωσης υψηλότερης τάξης και κωδικοποίησης. Η περαιτέρω εξέλιξη της τεχνολογίας GSM/EDGE οδήγησε στην πιο πρόσφατη έκδοση του προτύπου 3GPP, το οποίο υποστηρίζει μεγαλύτερη ευρυζωνικότητα και εντελλόμενη πρόσβαση σήματος για τη διεπαφή ραδιοεπικοινωνίας (Osseiran et al., 2016).

Η μεγάλη απήχηση και η ραγδαία εξάπλωση της τεχνολογίας 2G σε λειτουργικό πλέον επίπεδο, ώθησε τη βιομηχανία των επικοινωνιών πολύ γρήγορα να αναζητήσει λύσεις για ακόμα πιο εξελιγμένα πρότυπα προηγμένης γενιάς. Διεθνείς οργανισμοί όπως η International Telecommunications Union, Radio Communications (ITU-R) στοχεύοντας στην ταξινόμηση των νέων συστημάτων, σύμφωνα με το πρότυπο International Mobile Telecommunications 2000 (IMT-2000), διαμόρφωσαν τις προδιαγραφές που θα έπρεπε να έχουν τα νέα συστήματα για τον σκοπό αυτό. Μάλιστα, τον Ιανουάριο του 1998, το Ευρωπαϊκό Ινστιτούτο Τυποποίησης στον τομέα των Τηλεπικοινωνιών (ETSI) υιοθέτησε δύο παραλλαγές του CDMA: το Wideband Code Division Multiple Access (WCDMA) και το Time Division CDMA (TD-CDMA), ως το Παγκόσμιο Σύστημα Κινητών Επικοινωνιών (UMTS). Το UMTS ήταν το πρώτο σύστημα που έλαβε πιστοποίηση IMT-2000 και θεωρείται το κορυφαίο σύστημα κινητής επικοινωνίας Τρίτης γενιάς (3G). Επιπλέον, με το IMT-2000 πιστοποιήθηκαν έξι ραδιοδιεπαφές, μεταξύ αυτών η UWC-1362, μια έκδοση του GSM/EDGE που βασίζεται στην τεχνολογία CDMA και δύο ακόμα τεχνολογίες που βασίζονται στο OFDMA (Report ITU-R M.2135, 2008).

Στο πλαίσιο του 3rd Generation Partnership Project (3GPP), με σκοπό τη συνεργασία Εθνικών Οργανισμών Ανάπτυξης (SDOs) από όλο τον κόσμο σε επίπεδο κοινοπραξίας, αναπτύχθηκαν κοινές τεχνικές προδιαγραφές για τα συστήματα

κινητής, κυψελοειδούς επικοινωνίας (UMTS) 3ης γενιάς, γνωστές ως 3G Evolution καθώς και για την επέκταση αυτής της τεχνολογίας, γνωστή και ως 3.5G (βλ. Εικόνα 1) (Osseiran et al., 2016; ETSI).

1.2 Κινητά Δίκτυα 4ης γενιάς

Τα κινητά δίκτυα 4ης γενιάς (4G) αναπτύχθηκαν κατά τα έτη 2010–2012 (Ashiho, 2003; Moses, 2014; Kanani et al., 2014; Ohmori et al., 2000; Li et al., 2009; Miki et al., 2005) και χαρακτηρίζονται από τη δυνατότητά τους να υποστηρίζουν υψηλής ταχύτητας μεταφορά πολυμεσικών δεδομένων, περιεχόμενο τηλεόρασης υψηλής πιστότητας (HDTV) και υψηλές ταχύτητες πρόσβασης στο διαδίκτυο παρέχουν απρόσκοπτη πρόσβαση στο διαδίκτυο οποτεδήποτε και από οπουδήποτε, αξιόπιστα δίκτυα, βελτιωμένες υπηρεσίες ποιότητας και ενισχυμένη κινητικότητα και ασφάλεια. (Miki et al., 2005).

Οι τεχνολογίες 4G είναι πλήρως ολοκληρωμένα συστήματα βασισμένα στο Πρωτόκολλο Ίντερνετ (IP) και κάθε εργασία στο διαδίκτυο διενεργείται λόγω της συνεργασίας ενσύρματων και ασύρματων δικτύων όπως οι υπολογιστές, οι ηλεκτρονικές συσκευές ευρείας κατανάλωσης, η τεχνολογία επικοινωνίας και η ικανότητα να παρέχει ταχύτητες 100 Mbps and 1 Gbps, αντίστοιχα, τόσο σε εσωτερικό όσο και σε εξωτερικό χώρο με καλύτερη ποιότητα παρεχόμενης υπηρεσίας (QoS) και βελτιωμένη ασφάλεια, παρέχοντας τη δυνατότητα στον χρήστη να απολαμβάνει κάθε είδους διαδικτυακές υπηρεσίες κάθε στιγμή, από οπουδήποτε, με χαμηλό κόστος και με ενιαία χρέωση. (Miki et al, 2005; Debashis, 2016).

Μια από τις κυρίαρχες τεχνολογίες 4G, σημαντική για την εξέλιξη των κινητών δικτύων, με ειδικό ενδιαφέρον για την παρούσα εργασία, είναι η Long-term evolution advanced (LTE-A) (Debashis, 2016). Πρόκειται για μια συλλογή προτύπων 3GPP για υψηλής ταχύτητας ασύρματη επικοινωνία. Απαρτίζεται από μια νέα διεπαφή ραδιοεπικοινωνίας, η οποία βασίζεται στο Orthogonal Frequency Division Multiple Access (OFDMA) και επιτρέπει την πρόσβαση μέσω ραδιοφώνου στο εξελιγμένο Παγκόσμιο Δίκτυο Επίγειας Ραδιοεπικοινωνίας (EUTRAN) αλλά και μέσω μη ραδιοφωνικής επικοινωνίας. Διαθέτει νέα αρχιτεκτονική και Κεντρικό Δίκτυο

(CN), γνωστό και ως System Architecture Evolution/Evolved Packet Core (SAE/EPC) (Chaves et al., 2014).

Το LTE δεν είναι αντιστρόφως συμβατό με το UMTS. Στην πραγματικότητα, αναπτύχθηκε με σκοπό την εκχώρηση μπλοκ υψηλότερου φάσματος από ότι μπορούσε να παρέχει UMTS κατά την Παγκόσμια Διάσκεψη Ραδιοτηλεπικοινωνιών (WRC) του 2007. Το νέο αυτό πρότυπο σχεδιάστηκε, επίσης, για να λειτουργεί με συνιστώσες φέρουσας συχνότητας, που διαθέτουν μεγάλη ευελιξία ως προς τη διάταξη, ενώ μπορεί να υποστηρίζει φέρουσες συχνότητες 1.4 MHz έως 20 MHz. Πράγματι, το πρότυπο LTE έφερε σημαντικές βελτιώσεις στη χωρητικότητα και στη μείωση του κόστους σε σχέση με προηγούμενες γενιές ανάλογης τεχνολογίας. Οι πρώτες τεχνικές προδιαγραφές για το LTE εγκρίθηκαν για 3GPP με την ονομασία LTE Release 8. Το σύστημα LTE Release 8 διαθέτει ρυθμό μεταφοράς δεδομένων περίπου 326 Mbps, αυξημένη φασματική απόδοση και σημαντικά συντομότερο χρόνο αναμονής από ότι τα προηγούμενα συστήματα. Και παρότι το WiMAX ήταν αναγνωρισμένο ως τεχνολογία 4G, δεν έτυχε ευρείας αποδοχής και γρήγορα υποσκελίστηκε από το LTE. Μάλιστα, στο LTE Release 10 προστέθηκαν αρκετά νέα τεχνικά χαρακτηριστικά, όπως MIMO υψηλότερης τάξης και την εντελλόμενη πρόσβαση, γεγονός που βελτίωσε την χωρητικότητα και τη δυναμικότητα της προηγούμενης έκδοσης, Release 8. Η εντελλόμενη πρόσβαση έως τα 100 MHz συνολικού εύρους ζώνης επιτρέπει την αύξηση στο ρυθμό μεταφοράς στο μέγιστο των 3 Gbps κατά την καθοδική ζεύξη και στα 1.5 Gbps κατά την ανοδική ζεύξη. Οι διατάξεις MIMO υψηλότερης τάξης έως και 8×8 κατά την καθοδική ζεύξη και 4×4 κατά την ανοδική σχετίζονται επίσης με τη βελτίωση της επίδοσης (Report ITU-R P.1411, 1999; CELTIC / CP5-026 WINNER+ D5.3, 2010).

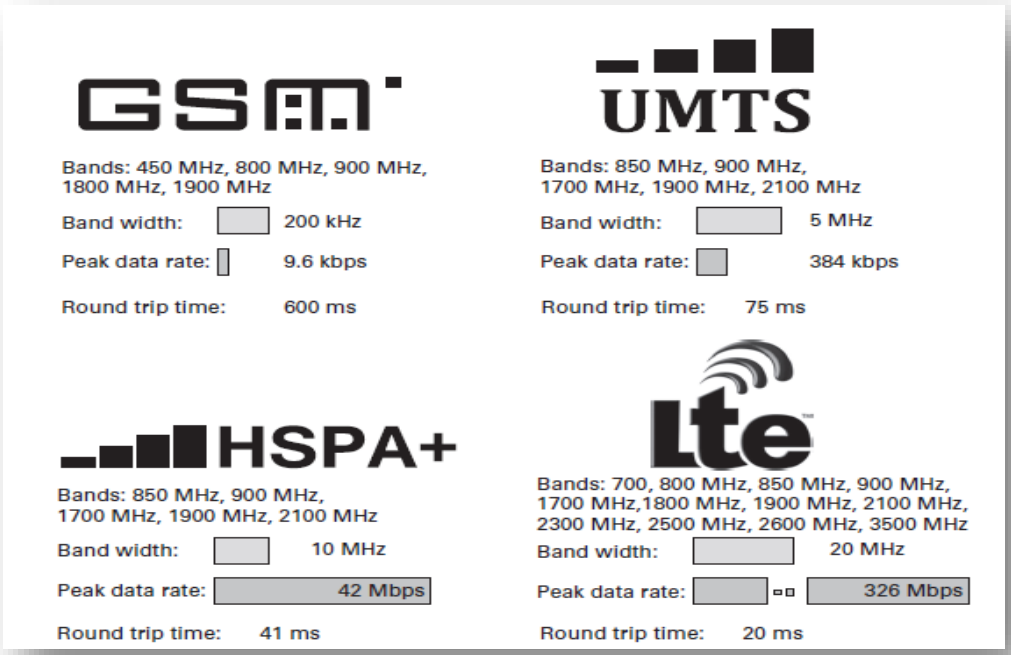
Η τυποποίηση 3GPP του of LTE (λ.χ. από Release 11 ως Release 13) είναι μια διαρκής διαδικασία και αναμένεται να συνεχιστεί και πέραν της Release 13. Το LTE Release 11 διατίθεται με βελτιωμένα τα χαρακτηριστικά του LTE Release 10, συγκεκριμένα έχει ενισχυθεί η εντελλόμενη πρόσβαση, η αναμετάδοση και η ακύρωση των παρεμβολών (relaying & interference cancellation). Προστέθηκαν νέες ζώνες συχνότητας και προσδιορίστηκε η χρήση του coordinated multipoint transmission and reception (CoMP). Στην LTE Release 12, που ολοκληρώθηκε τον Μάρτιο 2015, προστέθηκαν αρκετά χαρακτηριστικά που βελτιώνουν την

υποστήριξη ετερογενών δικτύων, ακόμα υψηλότερης τάξης από MIMO και συγκέντρωση μεταξύ μεταφορείς Frequency Division Duplexing (FDD) και Time Division Duplexing (TDD). Προσδιορίσηκαν επίσης αρκετά χαρακτηριστικά για την εκφόρτωση οπισθοζευκτικού και κεντρικού δικτύου.

Επιπλέον, στις LTE Releases 12 and 13, παρουσιάστηκαν νέες λύσεις, γνωστές ως LTE-M και Narrow-Band IoT (NB-IoT) για να υποστηρίξουν τις συσκευές μαζικής επικοινωνίας Machine Type Communication (MTC), όπως είναι οι αισθητήρες και οι ενεργοποιητές (3GPP TSG RAN Meeting #65, 2014; 3GPP TSG RAN Meeting #69, 2015). Οι λύσεις αυτές παρέχουν βελτιώσεις σε ό,τι αφορά τη διευρυμένη κάλυψη, μακρύτερο κύκλο ζωής της μπαταρίας και μειωμένο κόστος λειτουργίας. Η Release 13, επίσης, στοχεύει σε εξαιρετικά υψηλές ταχύτητες δεδομένων εύρους ζώνης χρησιμοποιώντας συνάθροιση φερόντων (carrier aggregation) έως 32 φορείς (carriers).

Στα μέσα 2015, η παγκόσμια αγορά κυψελοειδούς επικοινωνίας αριθμούς περί τα 7,5 δισεκατομμύρια συνδρομητές (gsmaintelligence.com/), με την οικογένεια του GSM/EDGE να χρησιμοποιεί ως κυρίαρχο το Access Network (RAN), που περιλαμβάνει EGPRS για συνδεσιμότητα δεδομένων. Το GSM με μερίδιο μεγαλύτερο του 57% στην παγκόσμια αγορά (που αντιστοιχεί σε 4.26 δισεκατομμύρια συνδρομητές), έχει ξεπεράσει την εποχή της ακμής του και σήμερα βρίσκεται σε ύφεση. Από την άλλη, ο αριθμός των συνδρομητών 3G subscribers, περιλαμβανομένων των HSPA, ανερχόταν σε 1.94 δισεκατομμύρια συνδρομητές έως το 2010, αριθμός που αντιπροσωπεύει ένα μερίδιο 26% στην αγορά. Η Ericsson Mobility Report προβλέπει ότι οι συνδρομές WCDMA/ HSPA θα αυξηθούν θεαματικά έως το 2020 φτάνοντας στην τιμή κορυφής και θα αρχίσουν να μειώνονται από το σημείο αυτό και μετά (Ericson Mobility Report, 2015).

Στο κυρίαρχο πρότυπο 4G, το LTE, συγκράτησε περίπου 910 εκατομμύρια συνδρομητές (ή 12% της συνολικής αγοράς) έως το τέλος 2015 και αναμένεται να φτάσει τις 4.1 δισεκατομμύρια συνδρομές έως το 2021 (Ericson Mobility Report, 2015), καθιστώντας την τη μεγαλύτερη τεχνολογία κινητής επικοινωνίας.



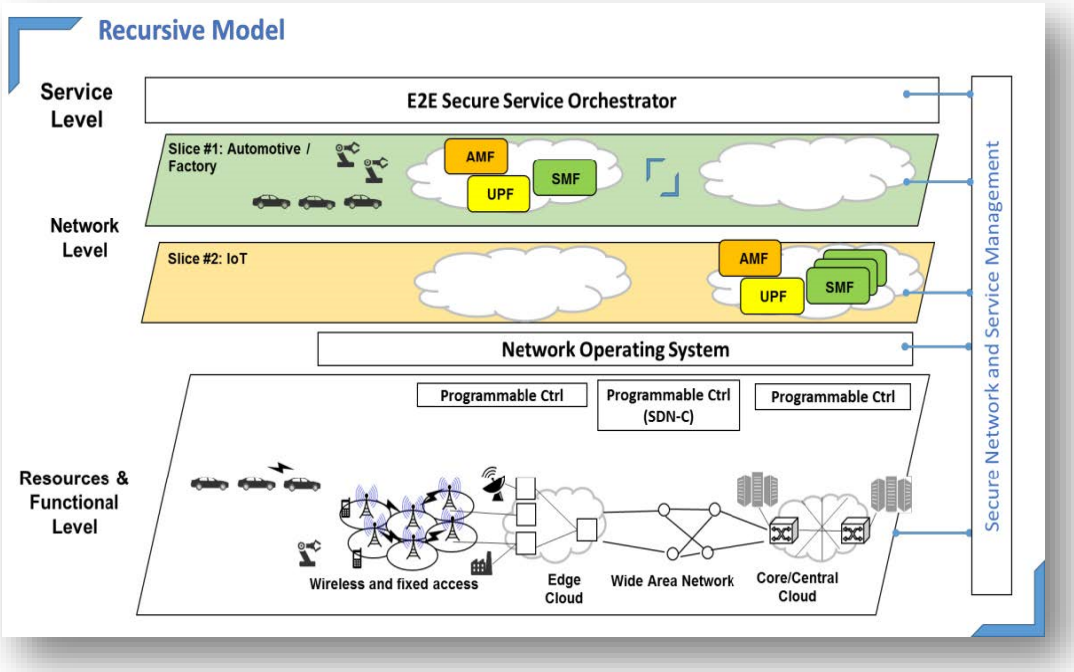
Εικόνα 2. Τα βασικά χαρακτηριστικά των συστημάτων 3GPP, που διατίθενται σήμερα στην αγορά. Επισημαίνεται η τάση προς μια ευρέως διαδεδομένη χρήση φάσματος, υψηλότερου εύρους ζώνης, υψηλότερης φασματικής απόδοσης και χαμηλότερης καθυστέρησης.
Πηγή: 5G Mobile and Wireless Communications Technology. (2016), (eds.) A. Osseiran, J. F. Monserrat, and P. Marsch. Cambridge University Press.

1.3 Κινητά Δίκτυα 5ης γενιάς

Η τεχνολογία κινητών δικτύων επικοινωνίας 5ης γενιάς αποτελεί μια από τις πλέον υποσχόμενες περιοχές έρευνας στον τομέα των κινητών επικοινωνιών του μέλλοντος. Πρόκειται για τεχνολογία που προτείνεται από τους ειδικούς ως απάντηση στις αυξανόμενες απαιτήσεις των χρηστών αυτών των υπηρεσιών για ολοένα και μεγαλύτερης ταχύτητας κινητές επικοινωνίες. Βεβαίως, η ανάπτυξη μιας τέτοιας τεχνολογίας αναμένεται να μπορεί να βρίσκει εφαρμογή σε κάθε είδους δίκτυα ραδιοπρόσβασης. Η μετάβαση από την κυρίαρχη τεχνολογία 4G στην 5G προβλέπεται να γίνει ομαλά, ώστε η νέα τεχνολογία 5G να αποδώσει τα μέγιστα οφέλη της στους παγκόσμιους φορείς εκμετάλλευσης. Επιπλέον, θα καταστεί περισσότερο εφικτή η διαλειτουργικότητα, εφόσον κατασκευαστεί μια ενιαία πλατφόρμα διαχείρισης για όλες τις τεχνολογίες 5G, που πρακτικά θα υποστηρίζει κάθε είδους τεχνολογία ραδιοπρόσβασης (RAT). Τα κινητά δίκτυα επικοινωνίας πέμπτης γενιάς (5G) είναι μια ασύρματη ποικιλία, η οποία θα υποστηρίζεται από

συγχρονισμένη πολλαπλή πρόσβαση διαίρεσης κώδικα ευρείας ζώνης (LAS-CDMA), από ορθογώνιο πολλαπλό σύστημα με διαίρεση συχνότητας (OFDM), πολλαπλή πρόσβαση διαίρεσης κώδικα με πολλές φέρουσες (MC-CDMA), από υπερευρεία ζώνη (UWB), από τοπικό πολυσημειακό σύστημα διανομής (network-LMDS) και από Πρωτόκολλο Ίντερνετ έκδοσης 6 (IPv6) (Sharma, 2013; Sharma et al., 2013). Το IPv6 αποτελεί το στοιχειώδες πρωτόκολλο για τη λειτουργία τόσο του 4G όσο και του 5G (Sharma et al., 2013). Το σύστημα κινητής επικοινωνίας 5G είναι ένα σύστημα που βασίζεται εξ ολοκλήρου σε IP. Στο 5G, οι εφαρμογές και οι υπηρεσίες που βασίζονται σε κινητά δίκτυα στη βάση IP, όπως οι πύλες (portals) κινητής επικοινωνίας, το εμπόριο μέσω κινητής επικοινωνίας, η φροντίδα υγείας μέσω κινητής επικοινωνίας, η διακυβέρνηση μέσω κινητής επικοινωνίας κ.λπ. επιτυγχάνονται μέσω πόρων υπολογιστικών νεφών (CCRs) (Sharma, 2013; Debashis, 2016)

Θα αναφερθούμε στα υπολογιστικά νέφη σε ειδική ενότητα παρακάτω.



Εικόνα 3. Γενική Αρχιτεκτονική 5G

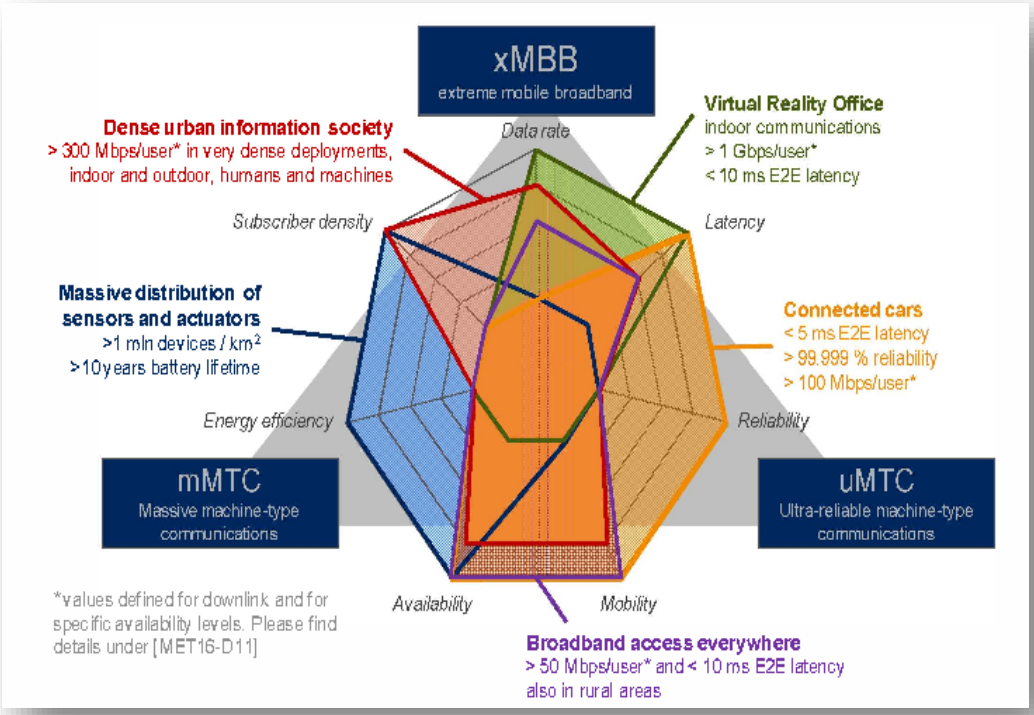
1.4 5GPPP

Για την χρηματοδότηση, την ανάπτυξη, την εξέλιξη και την τυποποίηση της νέας τεχνολογίας 5G, η Ευρωπαϊκή Ένωση υλοποίησε τα τελευταία χρόνια την πρωτοβουλία 5G Public Private Partnership (5GPPP), στο πλαίσιο της οποίας συνεργάζονται δημόσιοι και ιδιωτικοί φορείς που δραστηριοποιούνται στον χώρο των τηλεπικοινωνιών. Η 5GPPP έχει χρηματοδοτήσει και υποστηρίζει μια σειρά από ερευνητικά έργα που καλύπτουν περιοχές από το επίπεδο φυσικής σύνδεσης έως την συνολική αρχιτεκτονική του 5G, τη διαχείριση δικτύων καθώς και τα λογισμικά δικτύωσης. Πρόκειται για μια σημαντική πρωτοβουλία, δεδομένου ότι το 5G δεν αποτελεί μόνο την τεχνολογία του μέλλοντος στον τομέα των ραδιοεπικοινωνιών αλλά και το πλαίσιο στο οποίο νέες καινοτόμες τεχνολογίες συνεργάζονται με υφιστάμενες τεχνολογίες με σκοπό να ανταποκριθούν στις απαιτήσεις των εφαρμογών 5G και βεβαίως, στις απαιτήσεις χρήσης των δικτύων 5G από ανθρώπους και πράγματα. Αυτές συνοψίζονται στις ακόλουθες τρεις κατηγορίες χρήσης (Εικόνα 4):

- *Massive broadband (xMBB),* που παρέχει εύρος ζώνης σε Gbit/sec κατ' απαίτηση

- *Massive machine-type communication (mMTC),* που συνδέει δισεκατομμύρια αισθητήρων και μηχανημάτων

- *Critical machine-type communication (uMTC),* που επιτρέπει την άμεση ανατροφοδότηση υψηλής αξιοπιστίας, τον απομακρυσμένο έλεγχο από ρομπότ καθώς και αυτόνομη οδήγηση (5G PPP Architecture Working Group, 2016).



Εικόνα 4. Υπηρεσίες 5G

Συμπερασματικά, μπορούμε να πούμε ότι η σημαντικότητα της τεχνολογίας 5G μπορεί να κατανοηθεί καλύτερα, εάν αναλογιστεί κανείς τους περιορισμούς της τεχνολογίας 4G σε σχέση και με τις σύγχρονες απαιτήσεις καταναλωτών και συσκευών. Η τεχνολογία 5G αναμένεται να είναι διαθέσιμη στην αγορά για ευρεία χρήση στα μέσα του 2020 περίπου (Debashis, 2016).

2 – Δίκτυα VANETs

2.1 Εισαγωγή στα VANETs

Η τεχνολογία Vehicular Ad-hoc Network (VANET) έχει αναπτυχθεί σχετικά πρόσφατα στον χώρο των ασύρματων κινητών δικτύων, ωστόσο παίζει σημαντικό ρόλο για το ITS. Το ITS είναι μια υπηρεσία διαμοιρασμού δεδομένων μεταξύ των οχημάτων που διαθέτουν ασύρματες διεπαφές (Martinez, et al. 2011). Τα μετακινούμενα οχήματα και συγκοινωνιακές κυκλοφοριακές υποδομές χρησιμοποιούν την τεχνολογία ασύρματης επικοινωνίας δημιουργώντας κινητά δίκτυα. Μέσα σε αυτά τα δίκτυα, τα οχήματα παίζουν το ρόλο του ασύρματου κόμβου (node) ή του ασύρματου δρομολογητή. Έτσι, τα οχήματα που κινούνται σε μικρή ή μεσαία απόσταση δημιουργούν συνδέσμους μεταξύ τους δημιουργώντας ένα δίκτυο μεγαλύτερης κλίμακας. Αν ένα όχημα βρεθεί εκτός σήματος, τα υπόλοιπα οχήματα του δικτύου συνεχίζουν να μπορούν να συνδέονται στο δίκτυο μεταξύ τους δημιουργώντας ένα νέο κινητό δίκτυο.

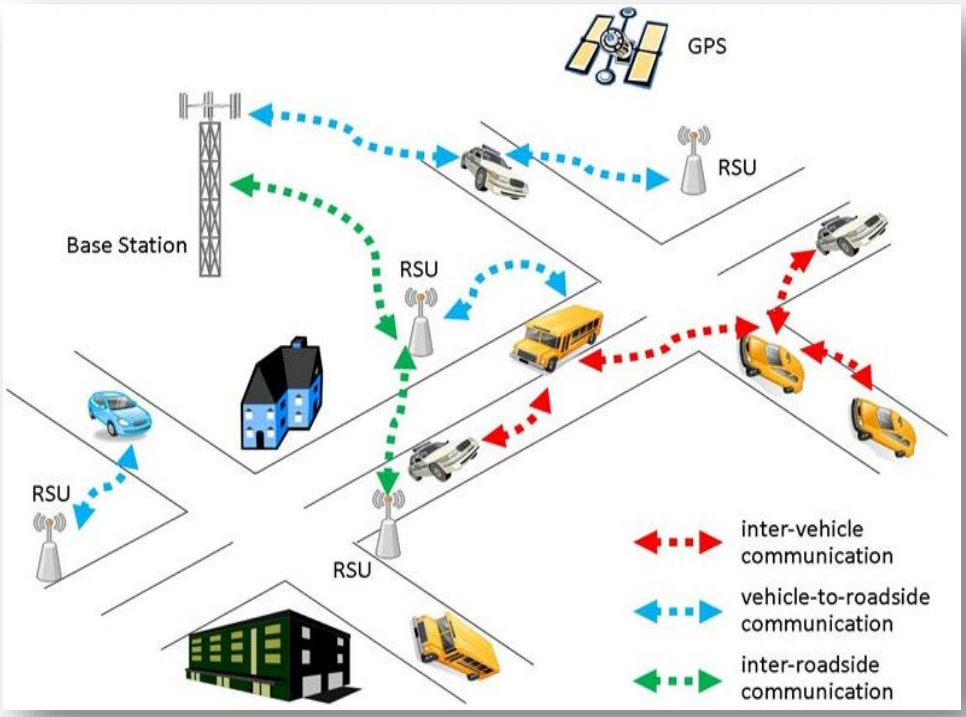
Το Vehicular Ad-hoc Network (VANET) αποτελεί μια ειδική περίπτωση του Mobile Ad-hoc Network (MANET) και διαθέτει τα παρακάτω διακριτά χαρακτηριστικά:

- ✓ Καθώς οι κόμβοι (nodes) κινούνται γρήγορα, η τοπολογία του δικτύου αλλάζει με γρήγορους ρυθμούς με αποτέλεσμα μικρής διάρκειας συνδέσεις.
- ✓ Ευαισθησία λόγω ασταθούς ποιότητας στον ραδιοδίαυλο εξαιτίας παρόδιων εγκαταστάσεων, συνθηκών δρόμου, τύπου οχήματος και σχετικής ταχύτητας οχήματος.
- ✓ Η ταχύτητα φορτίου αλλάζει ανάλογα με την πυκνότητα της κίνησης των οχημάτων και άρα, ο κόμβος θα πρέπει να είναι σε θέση να προσαρμόζεται σε αυτή την αλλαγή.
- ✓ Οι κόμβοι σε ένα VANET είναι οχήματα ή παρόδιες μονάδες με μεγάλες δυνατότητες παροχής ενέργειας, καλές ασύρματες επικοινωνίες, καλές δυνατότητες αποθήκευσης και υπολογισμού.
- ✓ Ο κόμβος διαθέτει ένα εύρος εξωτερικής πληροφορίας, όπως είναι το Global Positioning System (GPS) και το Geographic Information System (GIS).

- ✓ Ο κόμβος μετακινείται σε τακτική βάση, έτσι ώστε η διεύθυνση της κίνησης, η ταχύτητα και ο οδικός σύνδεσμος μπορούν να προβλεφθούν. (Zhang, 2017).

2.2 Αρχιτεκτονική των VANETs και η εξέλιξή τους

Η Αρχιτεκτονική των VANETs επιτρέπει στα οχήματα όχι μόνο να επικοινωνούν μεταξύ τους (V2V) αλλά και να συνδέονται με Road-Side-Units (RSU) και με σταθμούς βάσεις (V2I) (Truong et al., 2015).



Εικόνα 5. Αρχιτεκτονική VANET

Με την αύξηση των κινητών συσκευών και ως αποτέλεσμα την αύξηση του mobile traffic, οι επικοινωνίες οχήματος-οχήματος (V2V) και οχήματος-υποδομής (V2I) έχουν εκθετική αύξηση με αποτέλεσμα να δημιουργείται ένα μεγάλο κενό καθώς τα VANETs χρησιμοποιούνται για ένα ευρύ φάσμα υπηρεσιών, όπως υπηρεσίες ρύθμισης της οδικής κυκλοφορίας για την ασφάλεια των οχημάτων, υπηρεσίες επίβλεψης και υπηρεσίες νέφους.

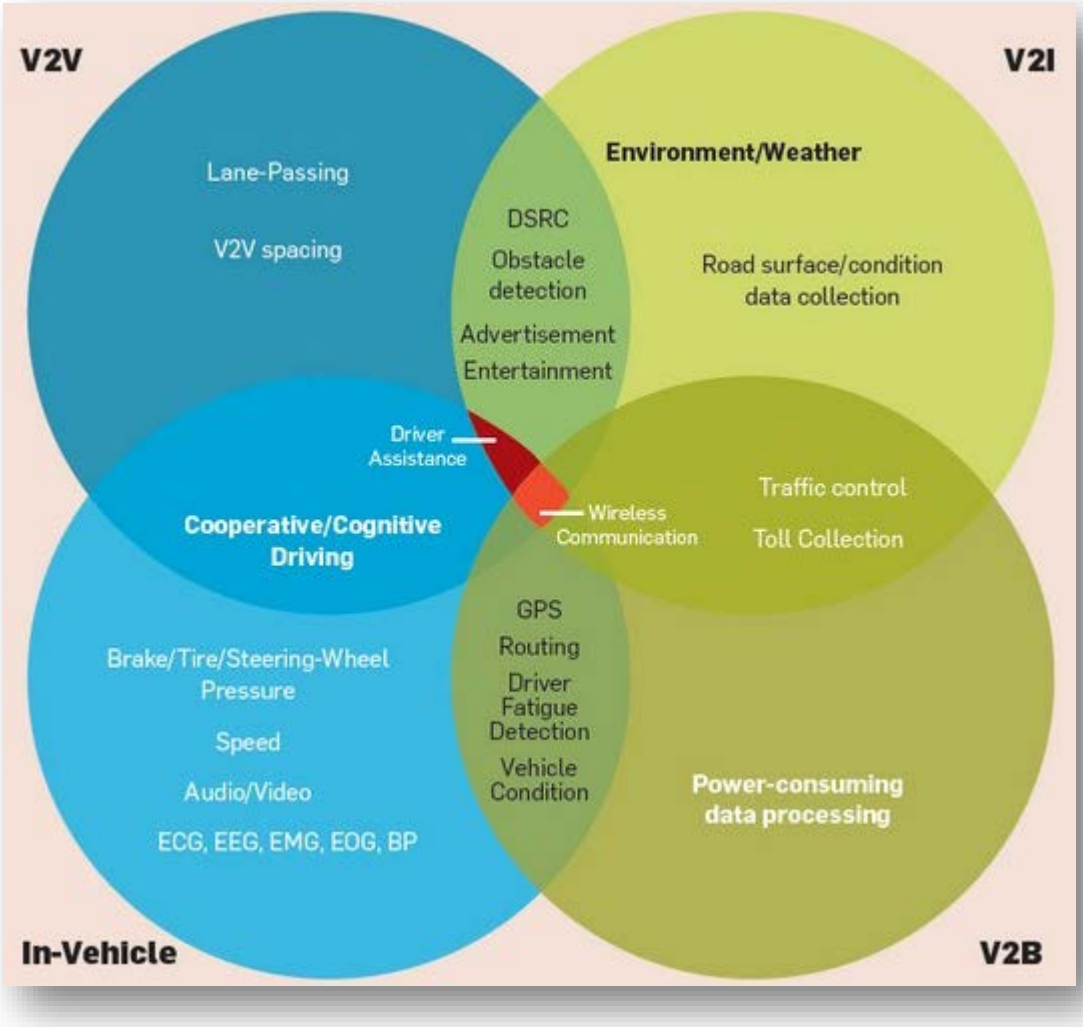
Τα VANET διακρίνονται σε τέσσερις τύπους επικοινωνίας (Εικόνα 7):

- *Vehicle to Vehicle (V2V) ή Car to Car (C2C) επικοινωνία*, που παρέχει μια βοηθητική πλατφόρμα ανταλλαγής δεδομένων για τη ραδιοφωνική μετάδοση και τον διαμοιρασμό πληροφοριών ή προειδοποιητικών μηνυμάτων για τους οδηγούς. Αναφέρεται στο Ad-hoc Domain.

- *Vehicle to Infrastructure (V2I) ή Car to Infrastructure (C2I) επικοινωνία,* παρέχει στους οδηγούς ενημερωμένη πληροφόρηση για την κίνηση στους δρόμους και για τον καιρό σε πραγματικό χρόνο.

- *In-Vehicle επικοινωνία,* αναφέρεται στην περιοχή υψηλού πεδίου εντός του οχήματος (In-vehicle domain). Μπορεί να ανιχνεύσει την απόδοση του οχήματος, να επεξεργαστεί τα δεδομένα από τον διαμοιρασμό της πληροφορίας και να παρέχει πληροφορίες παρέχοντας τις απαραίτητες λειτουργίες ασφάλειας για τους οδηγούς και το δημόσιο συμφέρον. Αυτού του είδους η επικοινωνία αποκτά ολοένα και μεγαλύτερη σημασία για την έρευνα στα VANET.

- *Vehicle to Broadband Cloud (V2B)* ή *Car to Broadband Could (C2B)* επικοινωνία, απαντά ως V2I. Πληροφορίες και δεδομένα αποθηκεύονται στο νέφος ευρείας ζώνης, αντί σε παρόδιες υποδομές, μέσω ασύρματων μηχανισμών ευρείας ζώνης, όπως 3G ή 4G σε οχήματα (Khan et al., 2018).



Εικόνα 6. Κύρια στοιχεία και λειτουργίες των τεσσάρων τύπων επικοινωνιών στο VANET

2.3 WAVE Protocol (IEEE 802.11p).

Το IEEE 802.11p, είναι μια τροποποιημένη έκδοση του IEEE 802.11, έτσι ώστε να ενσωματωθεί η ασύρματη επικοινωνία στη επικοινωνία οχημάτων. Είναι γνωστό και ως WAVE (Wireless Access in Vehicular Environments). Ορίζει τα πρότυπα για τις V2V και V2I επικοινωνίες καθώς και για το σχεδιασμό του (PHY) φυσικού επιπέδου.

Το 802.11p λειτουργεί στη ζώνη των 5,9 GHz (5,85GHz-5,925GHz). Η ζώνη αυτή είναι χωρισμένη σε 7 κανάλια των 10 MHz και υποστηρίζει υψηλές πραγματικές ταχύτητες που κυμαίνονται από 3 έως 27 Mbps για κάθε κανάλι. Στο φυσικό

επίπεδο βασίζεται στη μέθοδο μετάδοσης Ορθογωνική Πολύπλεξη με Διαίρεση Συχνότητας –OFDM.

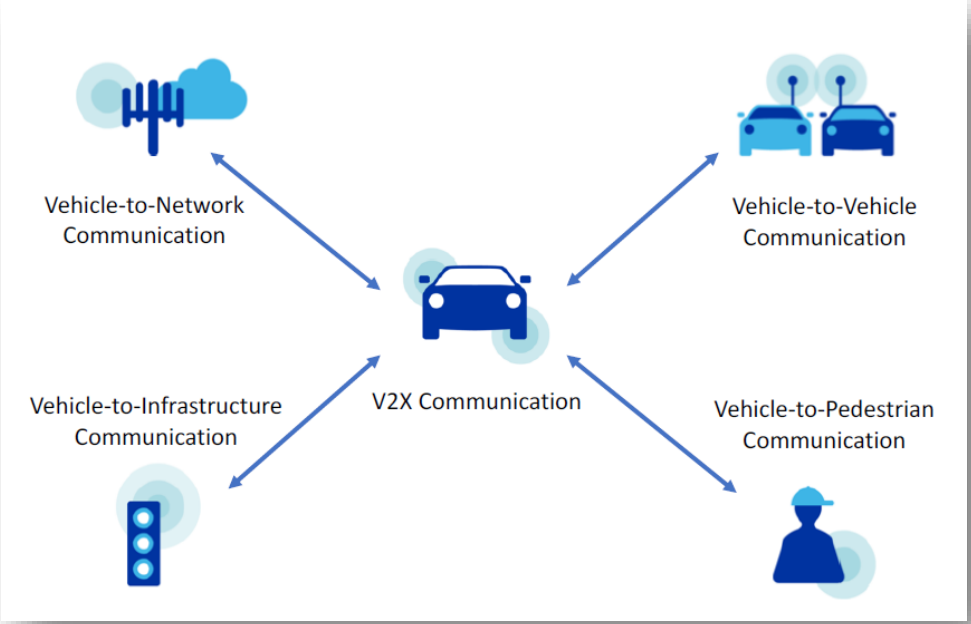
Στις ΗΠΑ το 2010 και πρόσφατα στην Ευρώπη δεσμεύτηκε η συχνότητα 5,9Hz για την ανάπτυξή του IEEE 802.11p με σκοπό να παρέχει ένα ενιαίο πλαίσιο επικοινωνίας σε όλους του εθνικούς οδικούς δρόμους. Το IEEE 802.11p στηρίζεται στο πρότυπο ASTM E2213-03 (Faezipour et al., 2012).

Το πρότυπο 802.11p για να ικανοποιήσει τις απαιτήσεις για επικοινωνία V2V σε δίκτυα VANET, τροποποιεί τα αρχικά πρότυπα της οικογενείας 802.11 για τους παρακάτω βασικούς λόγους :

- Υποστήριξη μεγαλύτερης εμβέλειας
- Υψηλή ταχύτητα των οχημάτων
- Κινητικότητα των οχημάτων
- Αντιμετώπιση καθυστερήσεων λόγω της αντανάκλασης των σημάτων στα μεταλλικά μέρη των αυτοκινήτων
- Ανάγκη για αξιόπιστη μετάδοση
- Υποστήριξη εφαρμογών ITS
- Ανάγκη για πολλαπλά επικαλυπτόμενα ad-hoc ασύρματα δίκτυα με εξαιρετικά υψηλή ποιότητα των παρεχόμενων υπηρεσιών (QoS)
- Έλεγχος προτεραιοτήτων και έλεγχος της ισχύς των σημάτων

2.4 V2X Communications

Οι επικοινωνίες V2X επιτρέπουν στα οχήματα να ανταλλάσσουν δεδομένα μεταξύ τους και με τις υποδομές, με στόχο τη βελτίωση της οδικής ασφάλειας, την αποτελεσματικότητα της κυκλοφορίας και τη διαθεσιμότητα των υπηρεσιών ψυχαγωγίας. Οι επικοινωνίες V2X, όπως ορίζονται στο 3GPP, αποτελούνται από τέσσερις τύπους επικοινωνιών, που αναλύσαμε πιο πάνω, δηλαδή οχήματος-οχήματος (V2V), οχήματος-υποδομής (V2I), οχήματος-δικτύου (V2N) και οχήματος-πεζών (V2P), όπως φαίνεται γραφικά στο παρακάτω σχήμα (Sassan, 2019).



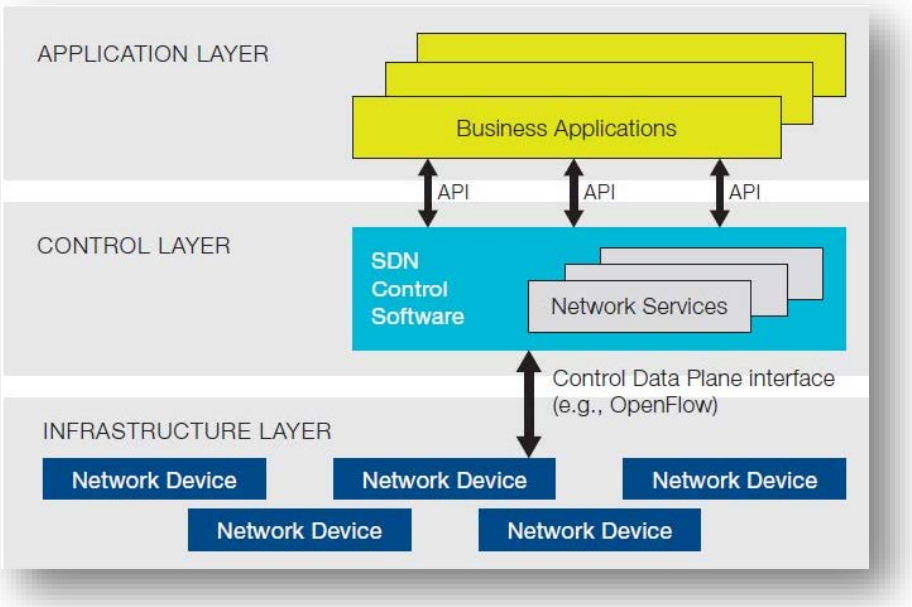
Εικόνα 7. Επικοινωνίες V2X

Η ανάπτυξη των V2X επικοινωνιών θα παίξει καταλυτικό ρόλο στο μέλλον των μεταφορικών τεχνολογιών καθώς όλο και περισσότερο πληθαίνουν τα ευφυή οχήματα. Πλέον, τα σημερινά αυτοκίνητα είναι εξοπλισμένα με διάφορους ενεργούς αισθητήρες, όπως κάμερες και ραντάρ, που αναγκάζουν τους ασύρματους αισθητήρες V2X να παρέχουν μεγαλύτερη εμβέλεια και αξιοπιστία, ειδικά σε σενάρια μη ορατότητας (NLoS), όπου υπάρχουν άλλα οχήματα και κτίρια παρεμποδίζουν τα συστήματα όρασης του οχήματος.

Η 3GPPP ως μέρος της εξέλιξης του LTE, αναπτύσσει το C-V2X με σκοπό την επιτάχυνση ανάπτυξης συστημάτων 5G, τα οποία θα υλοποιούν V2X επικοινωνίες και θα παρέχουν μεγαλύτερη κάλυψη και μεγαλύτερο εύρος ζώνης.

3 – Εισαγωγή στις τεχνολογίες SDN & NFV

Software-defined networking (SDN) είναι μια αρχιτεκτονική, η οποία έχει σχεδιαστεί για να παρέχει μεγαλύτερη ευελιξία και βελτιστοποίηση στο δίκτυο. Η αρχιτεκτονική SDN δίνει τη δυνατότητα για τον σχεδιασμό, την κατασκευή και τη διαχείριση δικτύων, διαχωρίζοντας τα επίπεδα ελέγχου και προώθησης δεδομένων. Σαν αποτέλεσμα, είναι εφικτός ο απευθείας προγραμματισμός του επιπέδου ελέγχου και ο διαχωρισμός των υφιστάμενων υποδομών για εφαρμογές και υπηρεσίες δικτύου (www.cisco.com).

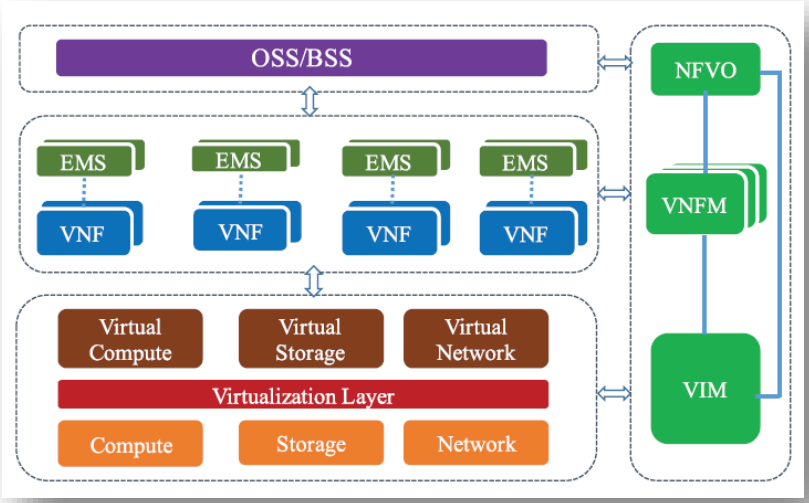


Εικόνα 8. Αρχιτεκτονική SDN

Network Functions Virtualization (NFV) είναι η τεχνολογία δικτύου, η οποία διευκολύνει τον διαχωρισμό των λειτουργιών του δικτύου, επιτρέποντάς τους να εγκαθίστανται, να ελέγχονται και να είναι δυνατός ο χειρισμός τους μέσω λογισμικών που εκτελούνται σε υπολογιστικούς κόμβους (www.juniper.net).

Το NFV βασίζεται στον παραδοσιακό εικονικό εξυπηρετητή αλλά διαφέρει από αυτόν ως προς το ότι οι Virtualized Network Functions (VNF) μπορεί να απαρτίζονται από ένα ή περισσότερα εικονικά μηχανήματα που εκτελούν διαφορετικά λογισμικά

και διεργασίες για να αντικαταστήσουν τις προκαθορισμένες συσκευές υλικού. Το NFV απαιτεί ένα πλαίσιο ενορχήστρωσης, θα λέγαμε, το οποίο επιτρέπει την κατάλληλη ενημέρωση των μεταβλητών (instantiation), την παρακολούθηση (monitoring) και τη λειτουργία (operation) των VNFs και των Network Functions (Osseiran et al., 2016)



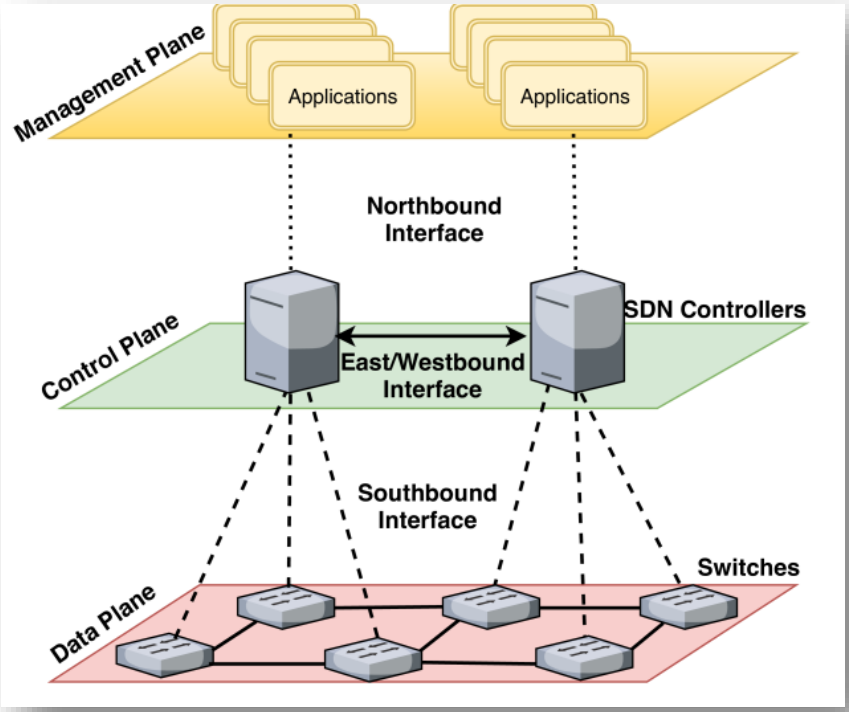
Εικόνα 9. Το αρχιτεκτονικό σύστημα NFV

3.1 Αρχιτεκτονική SDN και NFV

Η βασική ιδέα του SDN είναι ο διαχωρισμός μεταξύ της διαχείρισης του δικτύου (control plane) και της μετάδοσης των δεδομένων (data plane). Στο SDN οι συσκευές δικτύου, όπως γραμμές μεταγωγής (switched lines) και δρομολογητές (routers), απλώς προωθούν τα πακέτα σύμφωνα με τις πολιτικές (κανόνες) που βρίσκονται σε κάθε συσκευή. Αυτοί οι κανόνες δημιουργούνται ή τροποποιούνται από έναν ελεγκτή (controller), πριν από την αποστολή στις συσκευές όλου δικτύου. Ο ελεγκτής είναι η λογική ελέγχου που υπαγορεύει τη συνολική συμπεριφορά του δικτύου (Chaves et al., 2014).

Οι συσκευές δικτύου απλά δέχονται κανονισμούς από τον ελεγκτή χωρίς να κατανοούν και να εφαρμόζουν τα διάφορα πρωτόκολλα δικτύων, ελέγχοντας,

προγραμματίζοντας, οργανώνοντας και συντονίζοντας απευθείας τους πόρους του δικτύου στον SDN. Το χαρακτηριστικό αυτό εξοικονομεί εργατοώρες και πόρους.
Για να αναπτυχθεί η αρχιτεκτονική SDN, βασική προϋπόθεση είναι ένα πρωτόκολλο επικοινωνίας μεταξύ Data Plane και Control Plane (South-bound Interface SBI). Αυτό το πρωτόκολλο είναι απαραίτητο να είναι τυποποιημένο και χωρίς ταυτοποίηση παρόχου υπηρεσίας (vender-agnostic). Για τον σκοπό αυτό οι εταιρείες δικτύωσης είναι απαραίτητο να αναπτύξουν προϊόντα με αυτή τη λογική. Το πρωτόκολλο SBI διευκολύνει, κατά τον ίδιο τρόπο, ετερογενείς switches και routes δικτύωσης, απλοποιώντας τις λειτουργίες του συστήματος δικτύωσης. Το OpenFlow αποτελεί χαρακτηριστικό παράδειγμα αυτού του είδους πρωτοκόλλου (Truong et al., 2015; ONF, 2013; Pepelnjak, 2018).

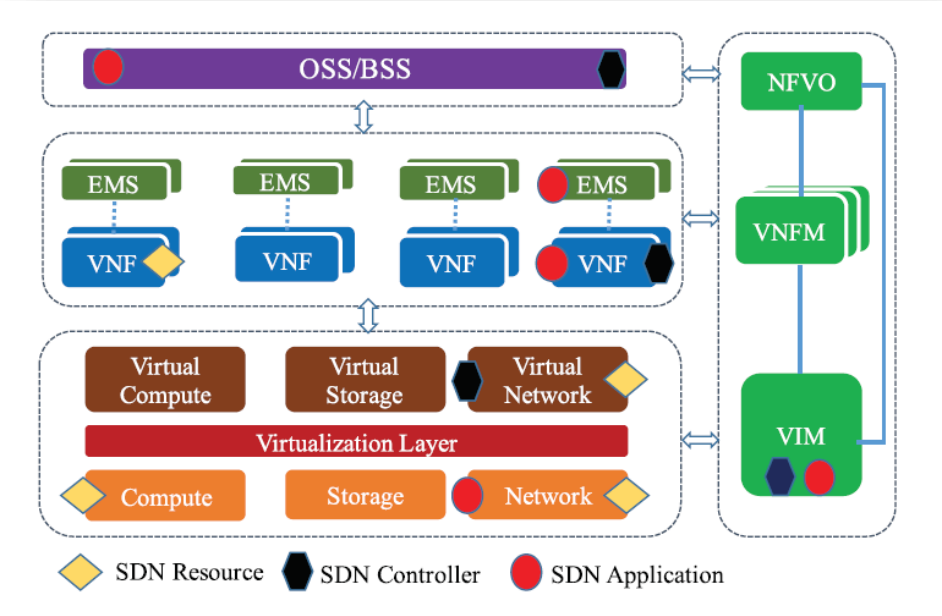


Εικόνα 10. Διεπαφές SDN (Northbound - Southbound Interfaces)

Από την άλλη πλευρά, το NFV είναι η τάση στη δικτύωση με στόχο την μεταφορά λειτουργιών από ιδιόκτητο υλικό σε εφαρμογές που εκτελούν γενικές λειτουργίες υλικού (hardware). Σκοπός της ανάπτυξης του NFV είναι η μείωση του κόστους και η

αύξηση της ελαστικότητας των λειτουργιών δικτύου, δημιουργώντας εικονικές λειτουργίες δικτύου (VNFs) που είναι συνδεδεμένες ή αλυσοδεμένες μεταξύ τους για την κατασκευή υπηρεσιών επικοινωνίας.

Το NFV και το SDN είναι λύσεις που μπορούν να εφαρμοστούν και μόνες τους, αλλά ο συνδυασμός των δύο είναι περισσότερο αποτελεσματικός. Η τεχνολογία NFV είναι ικανή να υποστηρίζει το SDN παρέχοντας την υποδομή πάνω στην οποία θα τρέχει το λογισμικό του SDN. Επιπλέον, η τεχνολογία NFV είναι σύμφωνη με τους αντικειμενικούς σκοπούς του SDN για χρήση εμπορικών server και switches (Nguyen et al., 2017).

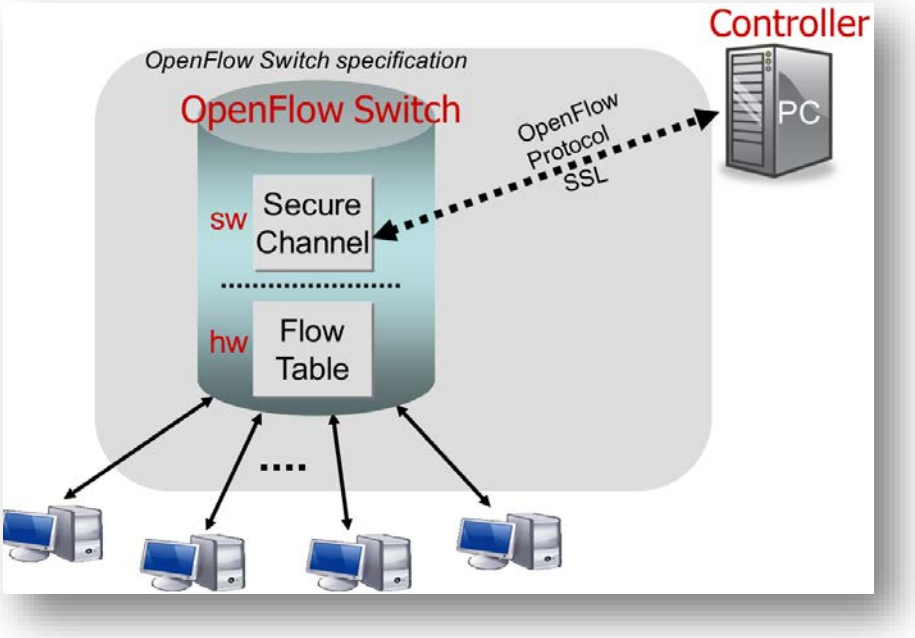


Εικόνα 11. Χαρτογράφηση των στοιχείων SDN στο αρχιτεκτονικό σύστημα NFV

3.2 Πρωτόκολλα Ανοικτής Ροής (OpenFlow)

Το πρωτόκολλο **Ανοικτής Ροής**, γνωστό και ως **OpenFlow**, είναι ένα σύστημα πολλαπλών πωλητών (multivendor standard), όπως έχει οριστεί από το Open Networking Foundation (ONF) για την υλοποίηση του SDN σε εξοπλισμό δικτύωσης. Ορίζει τη διεπαφή μεταξύ ενός ελεγκτή Ανοικτής Ροής (OpenFlow Controller) και ενός μεταγωγέα Ανοικτής Ροής (OpenFlow switch) (Εικόνα 12). Το πρωτόκολλο Ανοικτής Ροής επιτρέπει στον ελεγκτή Ανοικτής Ροής να δίνει εντολές στον

μεταγωγέα Ανοικτής Ροής καθοδηγώντας τον πώς να διαχειρίζεται τα εισερχόμενα πακέτα δεδομένων.



Εικόνα 12. Διεπαφή controller και switch μέσω πρωτοκόλλου OpenFlow

Ο διακόπτης OpenFlow είναι προγραμματισμένος για να:

(1) αναγνωρίζει και κατηγοριοποιεί πακέτα δεδομένων που εισέρχονται από την πύλη εισόδου με βάση τα headers των διαφόρων πακέτων,

(2) επεξεργάζεται τα πακέτα με διάφορους τρόπους και τροποποιεί τα headers,

(3) ορίζει που να προωθούνται τα πακέτα, είτε σε συγκεκριμένες πύλες εξόδου είτε προς τον ελεγκτή OpenFlow.

Οι οδηγίες Ανοικτής Ροής που μεταδίδονται από έναν ελεγκτή Ανοικτής Ροής προς ένα μεταγωγέα Ανοικτής Ροής δομούνται ως «ροές». Κάθε μια ροή περιλαμβάνει πεδία αντιστοίχισης πακέτων, προτεραιότητα ροής, διάφορους μετρητές, οδηγίες επεξεργασίας πακέτων και ροή περιόδων αναμονής. Όλες οι ροές είναι οργανωμένες σε πίνακες. Ένα εισερχόμενο πακέτο υφίσταται επεξεργασία από τις ροές σε πολλαπλούς πίνακες «ταχείας επεξεργασίας» (“pipelined”) πριν αποχωρήσουν από την πύλη εξόδου. Το πρωτόκολλο Ανοικτής Ροής εξελίσσεται με γοργούς ρυθμούς.

3.3 Cloud Computing

Το υπολογιστικό νέφος είναι μια τεχνολογία, που προσφέρει κατά απαίτηση, κλιμακούμενες υπηρεσίες επεξεργασίας και αποθήκευσης για διάφορες εφαρμογές. Αποτελείται από ένα κοινόχρηστο τμήμα εικονικών πόρων (υπολογιστική ισχύ, αποθήκευση, εφαρμογές και υπηρεσίες), το οποίο στεγάζεται στα κεντρικοποιημένα μεγάλης εκτάσεως Data Center. Οι πόροι αυτοί μπορούν να διατεθούν γρήγορα και άμεσα με ελάχιστες διαδικασίες διαχείρισης. Έτσι, λόγω της κεντρικοποιημένης φύσης του, όλες οι υπολογιστικές εργασίες μετακινούνται προς το νέφος και κατ' επέκταση στα data center. (Debashis, 2016).

Ωστόσο, η ταχεία ανάπτυξη του IoT, του Mobile Internet και των πολυμεσικών υπηρεσιών παράγουν πρωτοφανή μεγέθη δεδομένων, όπου είναι σχεδόν αδύνατο να σταλούν στο cloud για επεξεργασία και αποθήκευση καθώς το εύρος ζώνης του δικτύου δεν μπορεί να υποστηρίξει τον όγκο των δεδομένων, με αποτέλεσμα να δημιουργείται bottleneck στο υπολογιστικό νέφος. Άρα λόγω της αύξησης των δεδομένων δημιουργούνται τα παρακάτω προβλήματα στο cloud computing:

1. *Μεγάλη καθυστέρηση*: Η μεταφορά μεγάλου όγκου δεδομένων προς το cloud και πάλι στο τελικό χρήστη, καθιστά μεγάλο πρόβλημα σε εφαρμογές που βασίζονται σε πραγματικό χρόνο, όπως εφαρμογές παρακολούθησης της υγείας.
2. *Κόστος*: Η συνεχής ανάπτυξη των υποδομών επιβαρύνει, τόσο σε υλικό επίπεδο (κατασκευή μεγαλύτερων data center), ώστε να μπορούν να ανταπεξέλθουν στις εκάστοτε ανάγκες του τελικού πελάτη, όσο και σε ενεργειακό επίπεδο, αύξηση των πόρων, που σημαίνει περισσότερη ενέργεια.
3. *Αξιοπιστία*: Λόγω της κεντρικοποιημένης φύσης του cloud computing, σε περίπτωση κάποιου προβλήματος σε ένα data center, όλες οι υπηρεσίες που υποστηρίζονται από το cloud θα είναι μη-προσβάσιμες.

Για την επίλυση των παραπάνω προβλημάτων, έχουν αναπτυχθεί δυο τεχνολογίες computing: το Fog Computing και το Edge Computing στις οποίες θα αναφερθούμε αναλυτικά στη συνέχεια.

3.3.1 Fog Computing

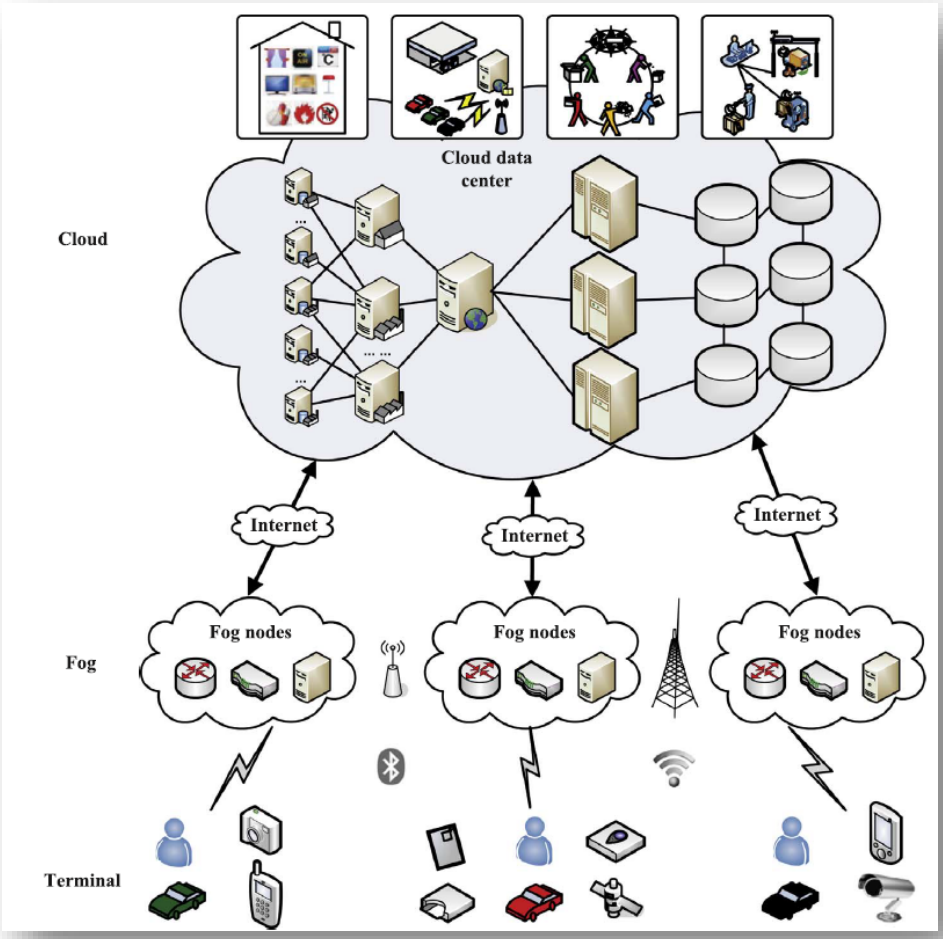
Το Fog computing είναι ένα υπολογιστικό παράδειγμα, το οποίο προεκτείνει το παραδοσιακό υπολογιστικό νέφος και τις υπηρεσίες στο τερματικό του δικτύου. Παρέχει τον υπολογισμό, την επικοινωνία, τον έλεγχο, την αποθήκευση και δυνατότητες εξυπηρέτησης στο τέρμα του δικτύου. Η μη-κεντρικοποιημένη πλατφόρμα είναι διαφορετική ως προς την αρχιτεκτονική σε σχέση με άλλα συμβατικά υπολογιστικά μοντέλα. Συγκρίνοντας την αρχιτεκτονική και τα χαρακτηριστικά του fog computing με εκείνα του cloud computing και του edge computing μπορούμε να πούμε ότι οι διαφορές προκύπτουν κυρίως εξαιτίας της θεμελιώδους δομής των τριών στρωμάτων του. Το Fog computing προεκτείνει τις υπηρεσίες νέφους στο τέρμα του δικτύου εισάγοντας ένα στρώμα fog ανάμεσα στις τερματικές συσκευές και το νέφος.

Η ιεραρχική αρχιτεκτονική του fog computing αποτελείται από τα ακόλουθα τρία στρώματα (Εικόνα 13):

Terminal layer: Αυτό το επίπεδο βρίσκεται πιο κοντά στον τελικό χρήστη και στο φυσικό περιβάλλον. Αποτελείται από διάφορες συσκευές IoT, όπως αισθητήρες, κινητά τηλέφωνα, ευφυή οχήματα, έξυπνες κάρτες, συσκευές ανάγνωσης κ.ο.κ. Ειδικά τα κινητά τηλέφωνα και τα ευφυή οχήματα με υπολογιστική ισχύ, χρησιμοποιούνται σε αυτή την περίπτωση μόνο ως συσκευές ευφυής ανίχνευσης. Τέτοιες συσκευές είναι διάσπαρτα και ευρέως κατανεμημένες γεωγραφικά. Είναι δε υπεύθυνες να ανιχνεύουν και να αισθάνονται τα χαρακτηριστικά δεδομένα φυσικών αντικειμένων ή γεγονότων και να τα μεταδίδουν στο ανώτερο στρώμα.

Fog layer: Αυτό το στρώμα βρίσκεται στο τέρμα του δικτύου. Το στρώμα Fog computing αποτελείται από έναν μεγάλο αριθμό fog κόμβων, οι οποίοι γενικώς περιλαμβάνουν δρομολογητές, μεταγωγείς, σημεία πρόσβασης, σταθμούς βάσης, ειδικούς εξυπηρετητές fog, κ.λπ. Τέτοιοι fog κόμβοι είναι ευρέως κατανεμημένοι μεταξύ τερματικών συσκευών και νέφους, για παράδειγμα, καφετέριες, πολυκαταστήματα, τερματικοί σταθμοί λεωφορείων, δρόμοι, πάρκα κ.λπ. Μπορεί να πρόκειται για σταθερές τοποθεσίες ή για κινητά σημεία που βρίσκονται πάνω σε

μετακινούμενο φορέα. Οι τερματικές συσκευές συνδέονται με fog κόμβους για να συνδεθούν με υπηρεσίες. Έχουν την ικανότητα να υπολογίζουν, να μεταδίδουν και να αποθηκεύουν προσωρινά τα δεδομένα που έλαβαν. Η ανάλυση σε πραγματικό χρόνο και οι εφαρμογές ευαίσθητες σε καθυστέρηση μπορούν να υλοποιηθούν σε ένα fog στρώμα.



Εικόνα 13. Αρχιτεκτονική Fog Computing

Cloud layer: Το στρώμα υπολογιστικού νέφους αποτελείται από πολλαπλούς εξυπηρετητές υψηλής απόδοσης και αποθηκευτικές συσκευές και παρέχει διάφορες υπηρεσίες εφαρμογής, όπως ευφυή σπίτια, ευφυής μετακίνηση, ευφυές εργοστάσια, κ.λπ. Διαθέτει υπολογιστική ισχύς και αποθηκευτικές δυνατότητες για να υποστηρίζει την εκτεταμένη υπολογιστική ανάλυση και τη μόνιμη αποθήκευση τεράστιων όγκων δεδομένων. Ωστόσο, αν και διαφορετική από την παραδοσιακή

αρχιτεκτονική υπολογιστικού νέφους, δεν πραγματοποιούνται όλες οι υπολογιστικές και αποθηκευτικές εργασίες μέσω του νέφους. Ανάλογα με τη ζήτηση φόρτου εργασία, οι καίριες οντότητες του νέφους (cloud core modules), διαχειρίζονται και προγραμματίζονται αποτελεσματικά ,μέσω στρατηγικών ελέγχων που αποσκοπούν στην βελτιστοποίηση της χρήσης των πόρων του νέφους (Sarkar and Misra, 2016).

Τα κύρια πλεονεκτήματα του Fog computing είναι ότι εξασφαλίζει (Pengfei Hua et al., 2017):

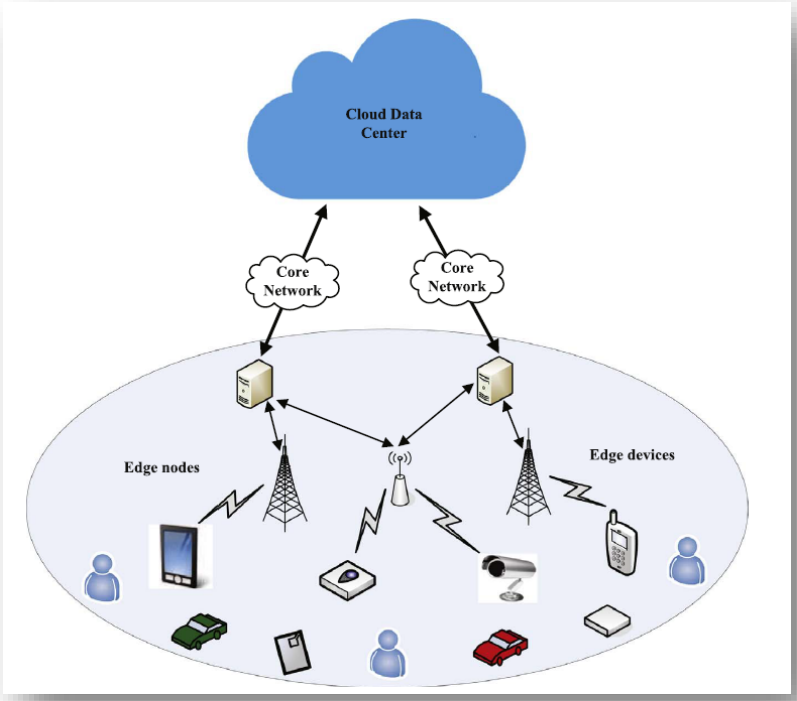
- Μείωση της καθυστέρησης και προσφέρει αλληλεπιδράσεις σε πραγματικό χρόνο
- Βελτιστοποιεί την χρήση του εύρος ζώνης
- Υποστηρίζει την κινητικότητα

3.3.2 Edge Computing

Το Edge computing, είναι ένα επιπλέον υπολογιστικό μοντέλο, το οποίο διέπεται από τις ίδιες αρχές με το Fog Computing. Το Edge computing, όπως και το Fog computing, προεκτείνει τις υπηρεσίες νέφους σε τερματικές συσκευές. Αναφέρεται σε εκείνες τις τεχνολογίες που επιτρέπουν να εκτελείται ο υπολογισμός και η αποθήκευση σε τερματικές συσκευές (Pengfei Hua et al., 2017; Ahmed and Ahmed, 2016; Beck et al., 2016).

Η Εικόνα 14 παρουσιάζει την αρχιτεκτονική του Edge Computing. Οι τερματικοί κόμβοι και συσκευές με υπολογιστική ικανότητα εκτελούν έναν μεγάλο αριθμό εργασιών υπολογισμού, όπως επεξεργασία δεδομένων, προσωρινή αποθήκευση, διαχείριση συσκευών, λήψη αποφάσεων και προστασία ιδιωτικότητας, μειώνοντας την καθυστέρηση στο δίκτυο και την κυκλοφορία ανάμεσα σε τερματικές συσκευές και νέφος (Shi et al., 2016). Αυτοί οι τερματικοί κόμβοι (edges) μπορούν να απαρτίζονται από έξυπνους αισθητήρες, ευφυή τηλέφωνα και ευφυή οχήματα, ακόμα και από ειδικούς τερματικούς εξυπηρετητές. Μπορούν να διασυνδέονται και να διεπικοινωνούν τοπικά σχηματίζοντας ένα τερματικό δίκτυο. Επιπλέον, οι τερματικές συσκευές συνδέονται με ένα κέντρο δεδομένων νέφους μέσω ενός κεντρικού δικτύου. Το Edge computing παρέχει “έξυπνες» υπηρεσίες πληροφοριών

κοντά στις τερματικές συσκευές για να ανταποκριθεί στις υψηλές απαιτήσεις της ψηφιακής βιομηχανίας, στις υπηρεσίες σε πραγματικό χρόνο, στη βελτιστοποίηση δεδομένων, στις ανάγκες των εξελισσόμενων εφαρμογών, στην ασφάλεια και στην προστασία της ιδιωτικότητας (Pengfei Hua et al., 2017).



Εικόνα 14. Αρχιτεκτονική Edge Computing

Η αρχιτεκτονική του Edge computing, όπως και του Fog computing είναι ιεραρχική, μη-κεντρικοποιημένη και κατανεμημένη, η οποία είναι διαφορετική σε σύγκριση με την κεντρικοποιημένη αρχιτεκτονική του cloud computing. Οι υπηρεσίες που προσφέρουν υλοποιούνται κοντά στο χρήστη. Στο Edge computing οι υπηρεσίες τοποθετούνται στις τελικές συσκευές, ενώ στο Fog computing στις τελικές συσκευές του δικτύου, οι οποίες βρίσκονται ένα επίπεδο πιο πάνω από τις τελικές συσκευές. Οι πόροι του Edge computing είναι περιορισμένοι σε σύγκριση με αυτούς του Fog computing και πολύ περισσότερο από αυτές του cloud computing. Αυτό συμβαίνει επειδή οι συσκευές του Fog computing και, κατ' επέκταση του cloud computing, διαθέτουν ισχυρότερες δυνατότητές. Επίσης, το Edge και το Fog, υποστηρίζουν μοντέλα κινητικότητας στους τελικούς χρήστες καθιστώντας τα ιδανικά για την ενσωμάτωσή τους σε VANETs.

4 – Καινοτόμες τεχνολογίες που υλοποιούνται από 5G VANETs

4.1 Τεχνολογίες δικτύων λογισμικού (Network softwarization technologies)

Οι τεχνολογίες δικτύων λογισμικού (network softwarization) εισάγονται ως θεμελιώδεις παράγοντες που επιτρέπουν την υλοποίηση των διάφορων απαιτήσεων, όπως εύκολος προγραμματισμός, ευελιξία της υποδομής και προσαρμοστικότητα ανάλογα με τις εκάστοτε ανάγκες, οι οποίες αναμένονται να παίξουν σημαντικό ρόλο στην εξέλιξη των 5G VANETs.

Τα πλεονεκτήματα των τεχνολογιών αυτών, περιλαμβάνουν τη μείωση των λειτουργικών δαπανών (OPEX) και των κεφαλαιουχικών δαπανών (CAPEX), την ταχεία δημιουργία και ανάπτυξη υπηρεσιών, την αποτελεσματική διαχείριση του κύκλου ζωής των υπηρεσιών, τη μείωση της κατανάλωσης ενέργειας και τη βελτίωση της ποιότητας της εμπειρίας για τους χρήστες.

Το network softwarization αξιοποιεί τεχνολογίες όπως το SDN και το NFV καθώς από την μια πλευρά χρησιμοποιεί τον προγραμματισμό λογισμικού για το σχεδιασμό, την υλοποίηση, την ανάπτυξη, τη διαχείριση και τη συντήρηση του εξοπλισμού, εξαρτημάτων και υπηρεσιών του δικτύου, ενώ από την άλλη πλευρά χρησιμοποιεί τεχνολογία εικονοποίησης (virtualization) για την ευελιξία, επαναχρησιμοποίηση των πόρων και γρήγορο επανασχεδιασμό του δικτύου και των υπηρεσιών.

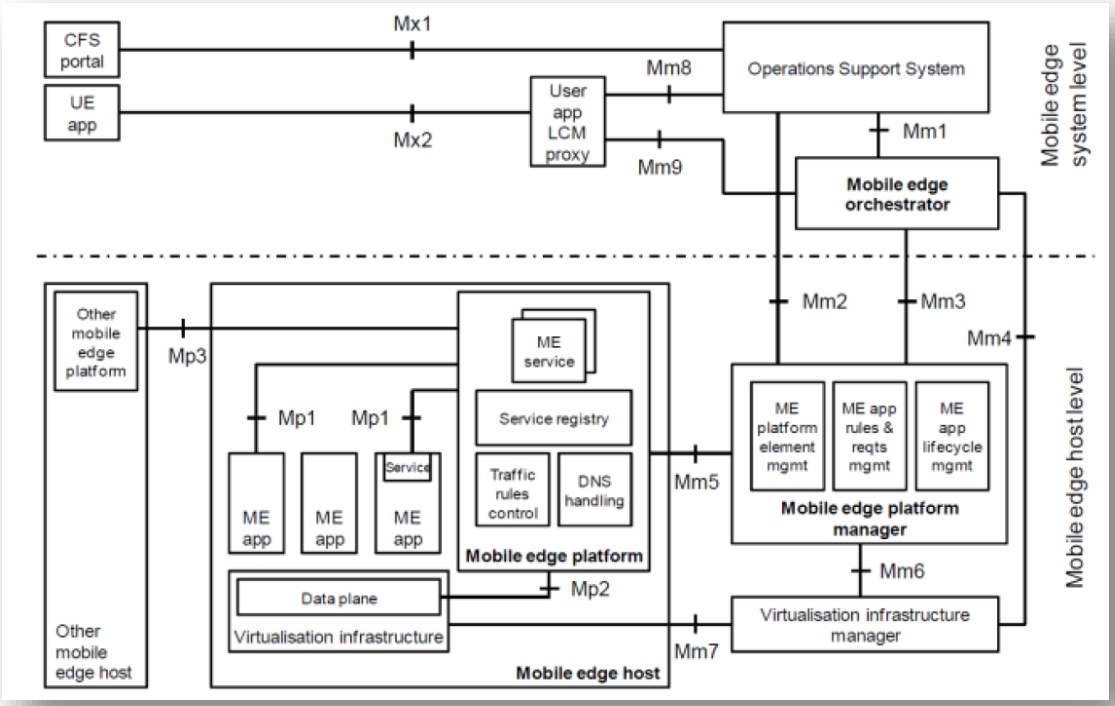
Οι κύριες τεχνολογίες δικτύων λογισμικού που μελλοντικά θα υλοποιηθούν από τα 5G VANETs είναι το Mobile Edge Computing και το Network Slicing, τα οποίες θα έχουν άμεσο αντίκτυπο και στην εξέλιξη των V2X επικοινωνιών (5G PPP Architecture, White Paper, EuCNC 2016).

4.2 Mobile Edge Computing (MEC)

Η κύρια και πιο βραχυπρόθεσμη επίδραση του softwarization στις V2X επικοινωνίες θα συμβαίνει στην άκρη του δικτύου, όπου είναι προσβάσιμοι οι πόροι, όπως η υπολογιστικότητα και η αποθήκευση, με την χαμηλότερη καθυστέρηση.

Την παραπάνω απαίτηση έρχεται να καλύψει το Mobile Edge Computing, τεχνολογία που έχει αναπτυχθεί από τον Ευρωπαϊκό Οργανισμό Τηλεπικοινωνιακών Προτύπων (ETSI) και σκοπός του είναι να προσφέρει μείωση της καθυστέρησης, υψηλό εύρος ζώνης και μείωση του ρυθμού μεταφοράς των δεδομένων. Το ETSI δημιούργησε το 2014 τη βιομηχανία MEC Industry Specification Group που παρείχε τις πρώτες προδιαγραφές. Το 2017, το όνομα MEC επεκτάθηκε σε Πολλαπλής πρόσβασης Edge Computing (Multi-access Edge Computing) για να συμπεριλάβει τις μη-κυψελοειδείς και σταθερής πρόσβασης περιπτώσεις. Το MEC υποστηρίζει υπηρεσίες πολλαπλών υπηρεσιών, υπηρεσίες πολυεθνικής μίσθωσης καθώς και τρίτα μέρη, τα οποία μπορούν να κάνουν χρήση υπηρεσιών του MEC, όπως αποθήκευση και επεξεργασία.

Η αρχιτεκτονική του MEC, σύμφωνα με τον ETSI, παρουσιάζεται στην Εικόνα 15 και αποτελείται από δύο επίπεδα: το Mobile Edge Host Level και το Mobile Edge System Level. Το Mobile Edge Host Level αποτελεί το κύριο υποσύστημα του MEC, το οποίο αποτελείται από το Mobile Edge Host (MEH), που περιλαμβάνει μια υποδομή εικονικοποίησης, βασισμένη στη λειτουργία δικτύου Υποδομή εικονικοποίησης–NFVI, η οποία προέρχεται από το Πλαίσιο NFV και την πλατφόρμα διαχείρισης (MEP), τα οποία υποστηρίζουν την εκτέλεση κινητών εφαρμογών στο άκρο του δικτύου (Borcoci et al., 2018).



Εικόνα 15. Αρχιτεκτονική MEC σύμφωνα με τον Ευρωπαϊκό Οργανισμό Τηλεπικοινωνιακών Προτύπων (ETSI)

Ένας MEC server μπορεί να εγκατασταθεί σε διάφορα σημεία στο άκρο δικτύου, όπως στον σταθμό βάσης 5G/LTE-A, σε RSU, στον ελεγκτή ραδιοελέγχου του δικτύου (RNC) κ.α. Το MEC μελλοντικά φαίνεται να είναι αποδοτική τεχνολογία για την υποστήριξη των 5G VANET, λόγω του ότι τα οχήματα βρίσκονται στο άκρο του δικτύου και οι πληροφορίες που ανταλλάζουν μεταξύ τους αλλά και με το δίκτυο είναι σε πραγματικό χρόνο (Borcoci et al., 2018).

4.3 Network Slicing

Network Slicing στην απλούστερη περιγραφή του, είναι η αξιοποίηση μιας τεχνολογίας εικονικοποίησης, δηλ. NFV ή SDN, για τον σχεδιασμό, την οργάνωση και τη βελτιστοποίηση των επικοινωνιών και τους υπολογιστικούς πόρους μιας φυσικής υποδομής σε πολλαπλά λογικά δίκτυα, με σκοπό την παροχή και την υποστήριξη διάφορών υπηρεσιών.

Με την ανάπτυξη του Network Slicing, μια ενιαία φυσική υποδομή δικτύου τεμαχίζεται σε πολλά εικονικά δίκτυα, καθένα από τα οποία ονομάζεται Network Slice. Κάθε Slice μπορεί να έχει τη δική της αρχιτεκτονική, εφαρμογές, επεξεργασία πακέτων και σημάτων και είναι υπεύθυνη για την παροχή ειδικών εφαρμογών και υπηρεσιών σε συγκεκριμένους τελικούς χρήστες. Ένα παράδειγμα ενός τέτοιου network slice μπορεί να είναι το εξής: ένα slice να εξυπηρετεί τη λειτουργία τηλεχειρισμού ενός εργοστασίου, ένα άλλο slice να παρέχει συγκεκριμένες υπηρεσίες για τις ανάγκες μιας εταιρίας και ένα άλλο να είναι αφιερωμένο στην παροχή υπηρεσιών έκτακτης ανάγκης (emergency health services) κ.ο.κ.

Ένα slice αποτελείται από εικονικές λειτουργίες δικτύου (VNFs), που είναι κατάλληλες για την υποστήριξη και την οικοδόμηση υπηρεσιών, οι οποίες θα παραδοθούν στους τελικούς χρήστες.

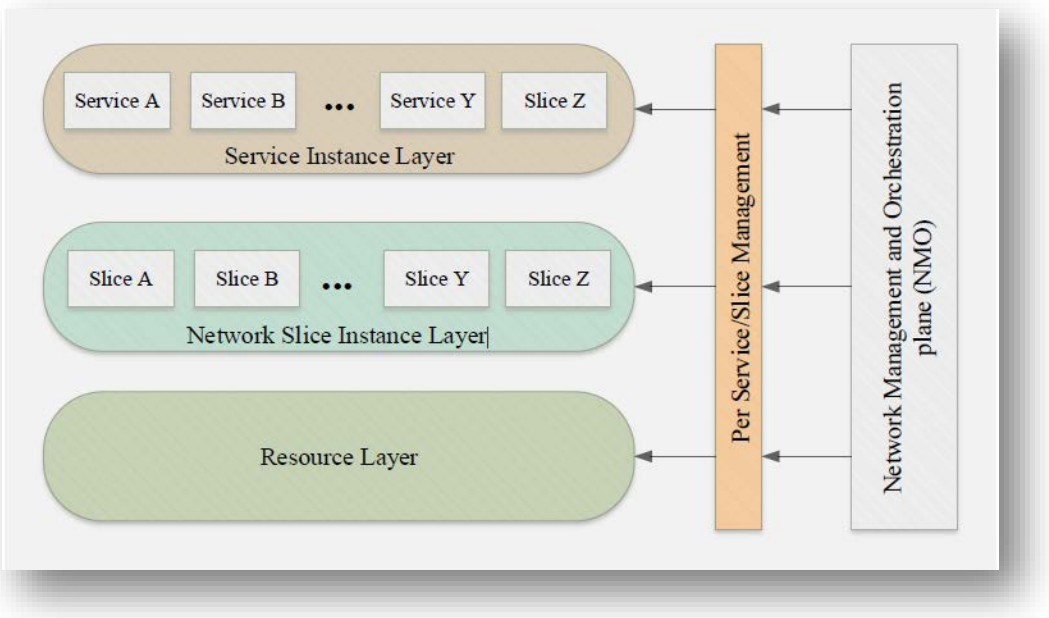
Η ανάπτυξη του Network Slicing περιλαμβάνει δύο κύριες φάσεις: αυτή της δημιουργίας και αυτή του χρόνου εκτέλεσης. Στη φάση δημιουργίας του slice, ο τελικός χρήστης αιτείται ένα slice από τον κατάλογο των προσφερόμενων slices και ο διαχειριστής του δικτύου προσφέρει αμέσως το συγκεκριμένο slice. Στη φάση εκτέλεσης χρόνου, διαφορετικά λειτουργικά μπλοκ ανασυγκροτούν κάθε slice σύμφωνα τις ανάγκες του τελικού χρήστη. Κάθε slice έχει τους δικούς του μηχανισμούς ασφαλείας και πρέπει να εξασφαλίζει τη λειτουργία τους προκειμένου να αποφευχθεί η πρόσβαση σε μη εξουσιοδοτημένες οντότητες. Με αυτό τον τρόπο διασφαλίζεται ο περιορισμός επιθέσεων σε ένα slice και όχι σε όλα τα slice του δικτύου. Σκοπός του slicing είναι να βοηθάει τον διαχειριστή του δικτύου να παρέχει υπηρεσίες και εφαρμογές, με τη δημιουργία slice και όχι τη δημιουργία ενός καινούργιου δικτύου, γεγονός που οδηγεί στη μείωση του CAPEX και στην εξοικονόμηση χρόνου.

Τα Network Slice λειτουργούν σε μια εν μέρει κοινόχρηστη υποδομή. Η υποδομή αυτή αποτελείται από dedicated υλικό, όπως στοιχεία του Ραδιο-δικτύου (RAN) και κοινόχρηστο υλικό, όπως πόροι του Network Functions Virtualization Infrastructure (NFVI).

Υπάρχουν δύο διαφορετικές έννοιες για την χρήση του network slicing στα δίκτυα επικοινωνιών. Η πρώτη είναι για σκοπούς Quality of Service (QoS) και η δεύτερη για σκοπούς χρήσης κοινής χρήσης υποδομής.

- *Slicing για QoS*: Η βασική ιδέα είναι η δημιουργία διάφορων slices για να προσφέρουν διαφορετικούς τύπους υπηρεσιών στους τελικούς χρήστες και να εξασφαλίσουν συγκεκριμένους τύπους απαιτήσεων QoS μέσα στο συγκεκριμένο slice. Ένα παράδειγμα αυτού του τύπου slicing, μπορεί να είναι ένα slice, το οποίο έχει δημιουργηθεί με σκοπό να προσφέρει υπηρεσίες σε μια συγκεκριμένη ομάδα συσκευών που έχουν συγκεκριμένες απαιτήσεις QoS, όπως χρήση ευρυζωνικής σύνδεσης για live video streaming για την έκτακτη ιατρική αντιμετώπιση.
- *Slicing για κοινή χρήση υποδομής*: Η βασική ιδέα του είναι η εικονοποίηση συγκεκριμένης υποδομής του RAN για την κοινή χρήση του από διάφορους διαχειριστές δικτύων. Αναλυτικότερα, υπάρχει ένας ιδιοκτήτης του slice και ένας ενοικιαστής. Ο ιδιοκτήτης δίνει ένα slice σε έναν ενοικιαστή, σύμφωνα με τις προδιαγραφές του ενοικιαστή. Ο ενοικιαστής έχει το γενικό έλεγχο τόσο στις λειτουργίες όσο και στην υποδομή του slice. Με αυτό τον τρόπο, το network slicing οδηγεί στη βελτιστοποίηση του δικτύου καθώς και στη μείωση κόστους εξόδων και παράλληλα στην δυνατότητα επεκτασιμότητας του δικτύου.

Σκοπός του network slicing στα δίκτυα 5G είναι να επιτρέψει στους φορείς εκμετάλλευσης να μοιράζονται τις υποδομές μεταξύ τους με ευέλικτο και δυναμικό τρόπο και να διαχειρίζονται αποτελεσματικά τους πόρους ανάλογα με τις ανάγκες των χρηστών τους.



Εικόνα 16. Αρχιτεκτονική διαχείρισης του Network Slicing

Το slicing μπορεί να αναπτυχθεί στα δίκτυα 5G σε δυο διαστάσεις, σε αυτή του Vertical Slicing και σε αυτή του Horizontal Slicing:

- *Vertical Slicing*: Η ανάπτυξη και η ανάπτυξη του κάθετου τεμαχισμού (Vertical Slicing) έχει ήδη ξεκινήσει στα δίκτυα 4G και επικεντρώνεται κυρίως στον βασικό τομέα των δικτύων κινητής τηλεφωνίας (Core Network). Τα κινητά ευρυζωνικά δίκτυα είναι κάθετα κομμένα (Vertical sliced) για να εξυπηρετήσουν υπηρεσίες με τον οικονομικά αποδοτικότερο τρόπο. Διαχωρίζει την κυκλοφορία των συγκεκριμένων υπηρεσιών από τις υπόλοιπες γενικές ευρυζωνικές υπηρεσίες του δικτύου κινητής τηλεφωνίας γεγονός που οδηγεί στην απλοποίηση των παραδοσιακών προβλημάτων του QoS.
- *Horizontal Slicing*: Λόγω της αύξησης των συσκευών του τελικού χρήστη και της κίνησης των δεδομένων που παράγεται στο άκρο του δικτύου, η επέκταση του network slicing από το βασικό τομέα (CN) μέχρι το RAN και τη διεπαφή του αέρα (air interface), ονομάζεται Horizontal Slicing. Έχει σχεδιαστεί για να φιλοξενήσει νέες τάσεις για την κλιμάκωση της χωρητικότητας του συστήματος, επιτρέποντας το cloud computing και τις υπηρεσίες του να διεξάγονται στις δικτυακές συσκευές στο άκρο του

δικτύου. Με αυτό τον τρόπο, το Horizontal Slicing επιτρέπει την ανταλλαγή πόρων μεταξύ κόμβων και συσκευών σε ένα δικτύου. Για παράδειγμα, ένα δίκτυο υψηλής δυνατότητας σε κόμβους, σε συσκευές και σε επικοινωνίες, μοιράζεται τους πόρους με άλλες συσκευές χαμηλότερων δυνατοτήτων, γεγονός που οδηγεί στη συνολική βελτίωση στις επιδόσεις του δικτύου (Habibi et al., 2017).

5 – Μεθοδολογία

Σκοπός της παρούσας εργασίας είναι να διερευνηθούν ποιες βελτιώσεις και αναθεωρήσεις χρειάζεται να αναπτυχθούν σε αρχιτεκτονικές SDN και NFV καθώς και στις τεχνολογίες cloud computing, που ήδη χρησιμοποιούνται στα VANETs, ώστε να ανταποκρίνονται στις σύγχρονες ανάγκες χρήσης των υπηρεσιών των δικτύων επόμενης γενιάς 5G.

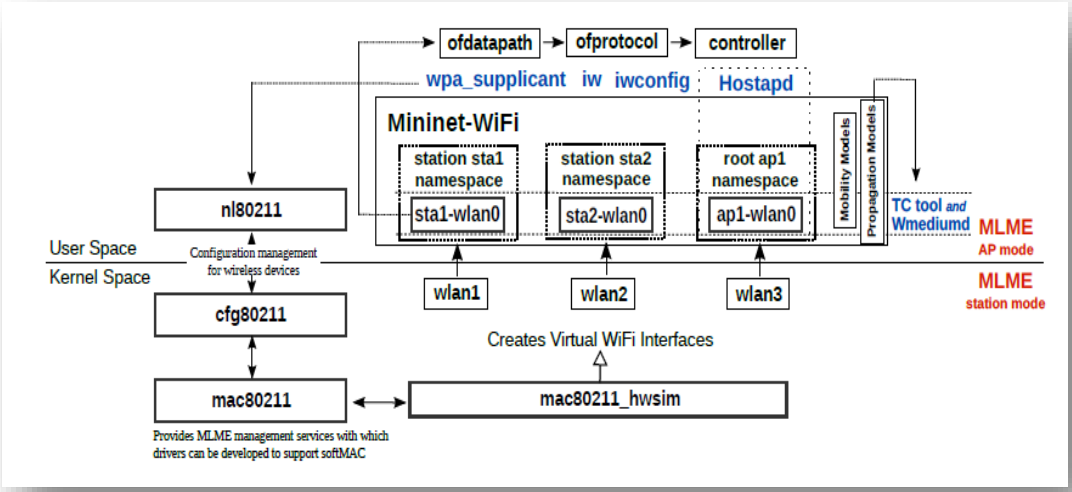
Μεθοδολογικά, συντάξαμε ένα σενάριο εργασίας στο οποίο ελέγχεται, σε συνθήκες προσομοίωσης, η ενσωμάτωση της εφαρμογής *τεμάχισης δικτύου* (network slicing) σε πρωτόκολλα επικοινωνίας *Vehicle to Everything (V2X)* στα Vehicular Ad-hoc Network. Συγκεκριμένα, ελέγχονται η καθυστέρηση (delay), η χρήση του εύρους ζώνης (bandwidth), η απώλεια πακέτων και το αντίκτυπο που έχουν στο QoS. Αξιολογούμε πώς με την εφαρμογή τεμάχισης δικτύου (network slice), επιτυγχάνουμε αυτόματες αποφάσεις διαπομπής από ένα ραδιο-δίκτυο σε άλλο, ανάλογα με τον φόρτο χρήσης τους (load-aware handover decision) με σκοπό να βελτιωθεί η σταθερότητα της απόδοσης του slice.

Για την υλοποίηση του σεναρίου προσομοίωσης θα χρησιμοποιήσουμε (α) το πρόγραμμα Mininet-WiFi και (β) τον Ελεγκτή ανοικτής ροής του SDN Ryu (Open Source SDN Controller Ryu).

(α) Το πρόγραμμα Mininet-Wifi αναπτύχθηκε από τους Ramon Fontes και Christian Esteve Rothenberg, το 2015 και αποτελεί επέκταση του προσομοιωτή SDN δικτύων Mininet (Fontes et al., 2015). Το συγκεκριμένος πρόγραμμα επιλέχθηκε ως το πιο κατάλληλο για την υλοποίηση του σεναρίου προσομοίωσής μας, καθώς παρέχει τη δυνατότητα δημιουργίας εικονικών Wifi σταθμών και σημείων πρόσβασης (access points), κάνοντας χρήση standard Linux wireless drivers και του 80211_hwsim wireless simulation driver και ανταποκρίνεται στις απαιτήσεις της προτεινόμενης στο σενάριό μας αρχιτεκτονικής. Επίσης, δίνει την δυνατότητα προσομοίωσης εικονικών κινητών σταθμών με πραγματικά χαρακτηριστικά, όπως η θέση και η κίνηση σε σχέση με

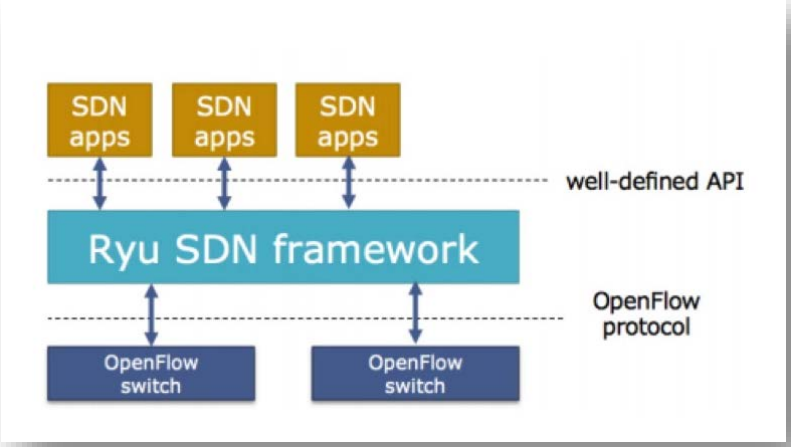
τα σημεία πρόσβασης. Ο κώδικας είναι ανοικτά προσβάσιμος στο GitHub, στο URL https://github.com/intrig-unicamp/mininet-Wifi.

Στην Εικόνα που ακολουθεί παρουσιάζεται σχηματικά η αρχιτεκτονική του Mininet-Wife (Components), όπως αναπτύχθηκε από τους Fontes & Rothenberg (2015).



Εικόνα 17. Αρχιτεκτονική Mininet-Wifi (Components).

(b) Το Ryu Controller είναι ένας ανοικτός ελεγκτής δικτύου (SDN) που έχει σχεδιαστεί για να αυξήσει την ευελιξία του δικτύου, διευκολύνοντας τη διαχείριση και προσαρμογή του χειρισμού της κυκλοφορίας. Επιλέχθηκε ειδικά για τις ανάγκες του σεναρίου προσομοίωσής μας επειδή έχει αναπτυχθεί εξ ολοκλήρου σε κώδικα python και υποστηρίζει πρωτόκολλα επικοινωνίας, όπως το OpenFlow 1.3, που θα χρησιμοποιήσουμε στο σενάριό μας. Εκτός από τις εκδόσεις του OpenFlow (1.1, 1.2, 1.3, 1.4, 1.5), υποστηρίζει επίσης το Netconf, το OF-config κ.α. (https://osrg.github.io/ryu/)

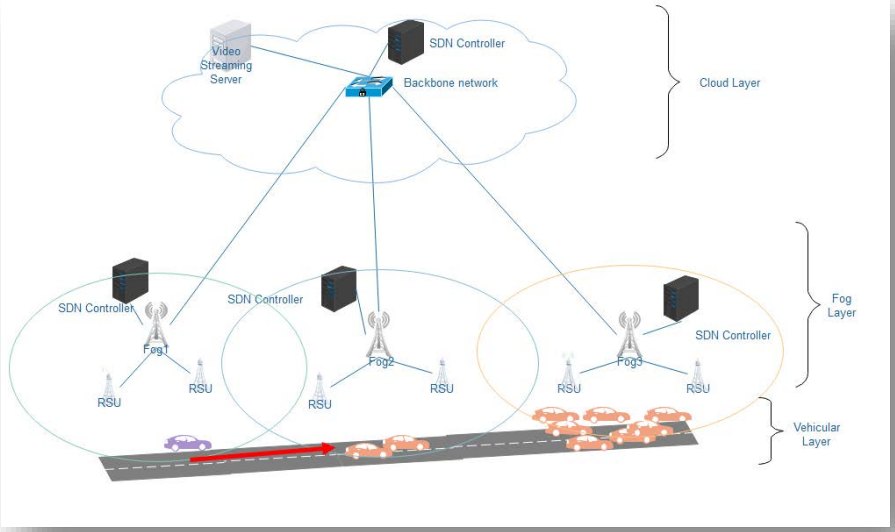


Εικόνα 18. Ο Ryu Controller σε περιβάλλοντα SDN

Στο σενάριο μας, η αρχιτεκτονική που προτείνουμε και αναλύουμε αποτελείται από 3 επίπεδα με τα παρακάτω components:

- *Cloud Layer*: Περιλαμβάνει έναν *εικονικό εξυπηρετητή* (virtual server), ο οποίος μεταδίδει video-υπηρεσία στα οχήματα, έναν *εικονικό διακόπτη ανοικτής ροής* (virtual open-flow switch), ο οποίος στο σενάριο μας λειτουργεί ως κεντρικό δίκτυο (backbone network) και από έναν *ελεγκτή SDN* (SDN controller), που διαχειρίζεται τον διακόπτη ανοικτής ροής (open-flow switch).

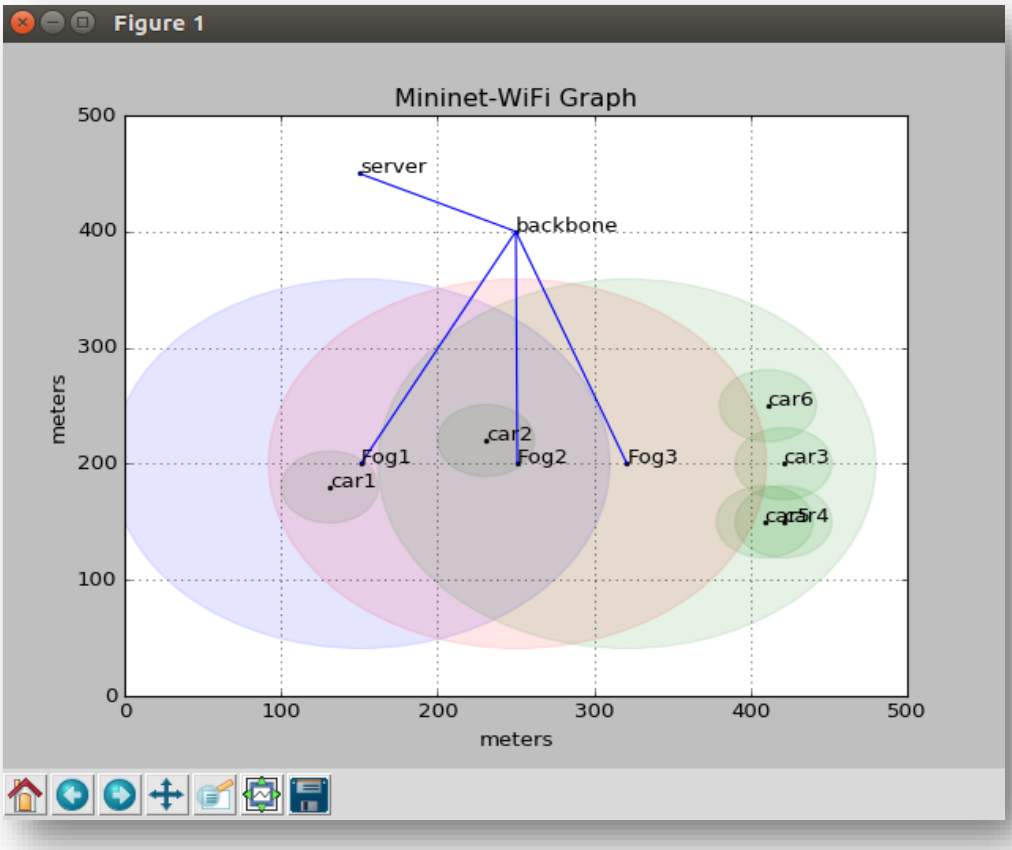
  Fog Layer: Αποτελείται από τρία διαφορετικά Fog networks, καθένα από τα οποία περιλαμβάνει έναν *σταθμό βάσης* (Base Station), δύο *RSU* και, έναν *SDN controller*, που είναι υπεύθυνος για την διαχείριση του εκάστοτε Fog. Η επικοινωνία μεταξύ των Fog και του backbone network, πραγματοποιείται μέσω οπισθοζεύξης (backhaul).
- *Vehicular Layer*: αποτελείται από τα οχήματα τα οποία επικοινωνούν μεταξύ τους και τους σταθμούς βάσης κάνοντας χρήση πρωτόκολλων επικοινωνίας V2V, V2I, V2P, V2X.



Εικόνα 19. Προτεινόμενη αρχιτεκτονική για την υλοποίηση του σεναρίου εργασίας

Το σενάριο εξελίσσεται ως εξής: το αυτοκίνητο car1, όπως φαίνεται στην Εικόνα 18, κινείται από το από το Fog 1 προς το Fog 3. Κατά την κίνηση αυτή, παρατηρούμε ότι η διαδικασία του handover θα πραγματοποιηθεί δυο φορές, την μια μεταξύ του Fog 1 και Fog 2 και μια μεταξύ του Fog 2 και του Fog 3. Αναλύουμε αυτές τις δυο χρονικές στιγμές, που ενεργοποιείται η διαδικασία slicing που έχουμε υλοποιήσει.

Για την εξαγωγή των αποτελεσμάτων συγκρίνουμε δυο συνθήκες: (i) όταν δεν κάνουμε χρήση της διαδικασίας του slicing και (ii) όταν κάνουμε χρήση της διαδικασίας του slicing. Στην πρώτη περίπτωση, το handover γίνεται με τον κλασικό τρόπο, σύμφωνα με το καλύτερο σήμα του σταθμού βάσης, ενώ στη δεύτερη περίπτωση, πως μπορούν οι SDN controllers να παίρνουν αποφάσεις handover αξιοποιώντας το network slicing, σύμφωνα με την κατάσταση του Fog που ελέγχουν.



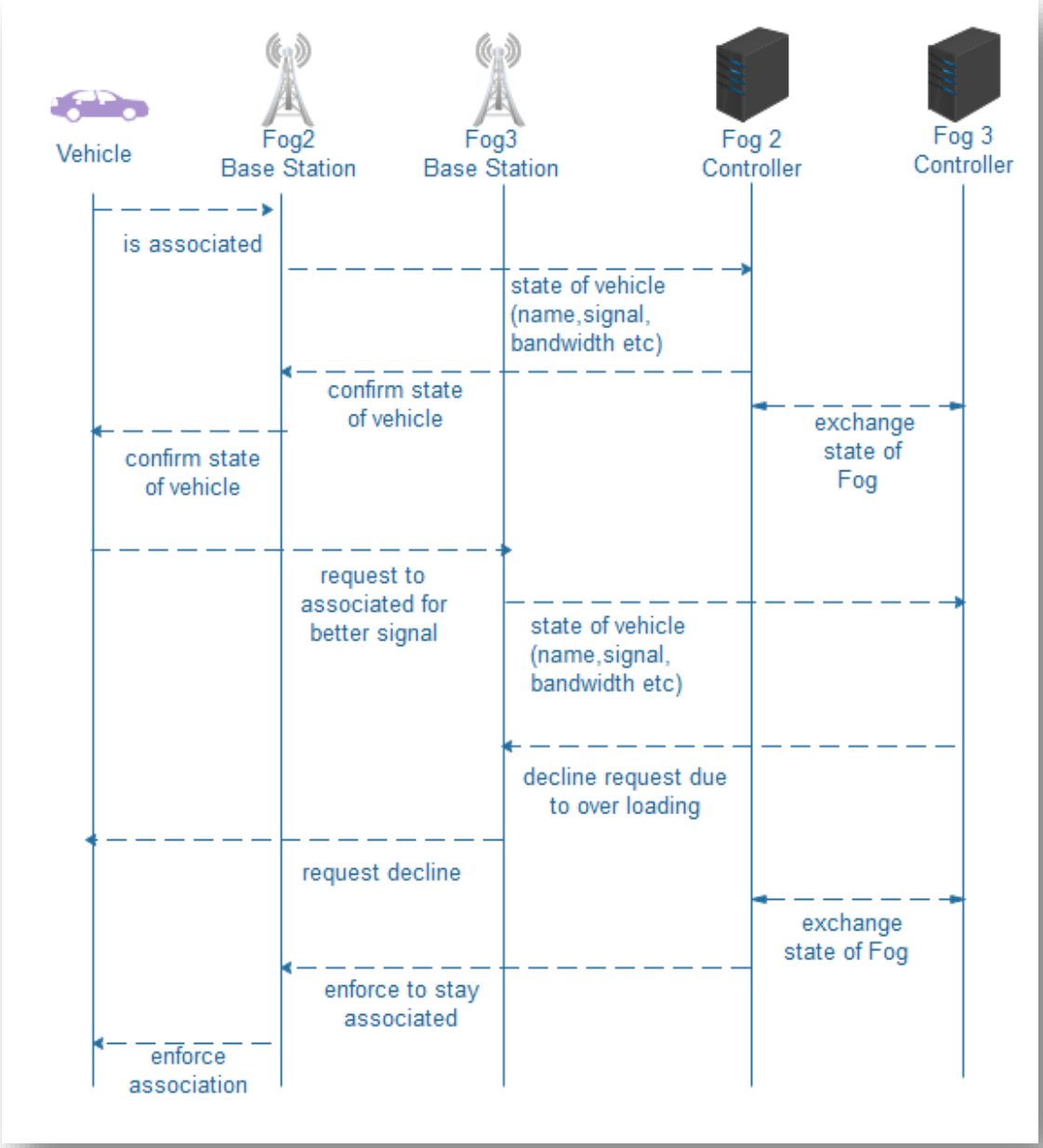
Εικόνα 20. Υλοποίηση Αρχιτεκτονικής στο Mininet-WiFi

Οι κύριες προτεινόμενες αλληλεπιδράσεις κατά την διαδικασία της διαπομπής (handover) από το ένα ραδιο-δίκτυο (FOG) στο άλλο απεικονίζεται στην Εικόνα 19.

Ο SDN Controller παρακολουθεί συνεχώς την κατάσταση (θέση, επιλογές ραδιο-κάλυψης) των εξυπηρετούντων οχημάτων από τον εκάστοτε σταθμό βάσης.

Κάθε controller διαμοιράζεται αυτές τις πληροφορίες με τους γειτονικούς του controllers, με αποτέλεσμα όταν ένα από τα οχήματα ζητάει να εξυπηρετηθεί από τον σταθμό βάσης του Fog 3 λόγω καλύτερου σήματος.

Το αίτημα λαμβάνεται από τον SDN controller του FOG 3, ο οποίος λόγω μεγάλου φόρτου εξυπηρέτησης οχημάτων δεν μπορεί να εξυπηρετήσει επιπλέον οχήματα εξαιτίας της μείωσης της ποιότητας υπηρεσίας, οπότε αποφασίζει να απορρίψει το αίτημα του οχήματος και με τη σειρά του ενημερώνει τον SDN controller του Fog 2 από όπου εξυπηρετούνταν αρχικά. Τέλος, ο controller του Fog 2 εξαναγκάζει το όχημα να παραμείνει συνδεδεμένο με τον σταθμό βάσης του Fog 2.



Εικόνα 21. Διάγραμμα ροής επικοινωνίας των SDN Controllers

Για την υλοποίηση του σεναρίου προσομοίωσής μας, θεωρούμε ότι υπάρχει στο δίκτυο ένας Server, ο οποίος ορίζουμε ότι παρέχει υπηρεσίες streaming video σε όλα τα οχήματα, με σκοπό να μπορέσουμε να μετρήσουμε την καθυστέρηση (delay), τη χρήση εύρους ζώνης (bandwidth), την απώλεια πακέτων και το αντίκτυπο που έχουν στο QoS και στις δύο συνθήκες που ορίσαμε παραπάνω (i και ii), όπου με διάφορα network εργαλεία θα εξάγουμε τα αποτελέσματα.

Η χρονική διάρκεια της προσομοίωσης είναι 3 λεπτά και 20 δευτερόλεπτα.

Για την υλοποίηση του παραπάνω σεναρίου στο Mininet-Wifi, δημιουργήσαμε ένα python script. Η δομή του script είναι ως εξής:

Αρχικά δηλώνουμε τις *βιβλιοθήκες* που θα χρησιμοποιήσουμε από το Mininet και Mininet-Wifi.

```
#!/usr/bin/python

import time
import os

from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mn_wifi.cli import CLI_wifi
from mn_wifi.net import Mininet_wifi
from mn_wifi.link import wmediumd
from mininet.term import makeTerm, cleanUpScreens
from mn_wifi.wmediumdConnector import interference
```

Στη συνέχεια δημιουργούμε τα *οχήματα* και τους *σταθμούς βάσης* καθώς *δηλώνουμε τις IP* που απαντούν οι Ryu controllers, ο server και ο openflow switch που παίζει το ρόλο του backbone network.

```
    info("*** Creating nodes\n")
    cars = []
    car1 = net.addStation('car1', mac='02:00:00:00:00:01',
                          position='130,180,0') #, active_scan=1)

    car2 = net.addStation('car2', mac='02:00:00:00:00:02',
                          position='230,220,0') #, active_scan=1)

    car3 = net.addStation('car3', mac='02:00:00:00:00:03',
                          position='420,200,0') #, active_scan=1)

    car4 = net.addStation('car4', mac='02:00:00:00:00:04',
                          position='420,150,0') #, active_scan=1)

    car5 = net.addStation('car5', mac='02:00:00:00:00:05',
                          position='408,150,0') #, active_scan=1)

    car6 = net.addStation('car6', mac='02:00:00:00:00:05',
                          position='410,250,0') #, active_scan=1)
    cars.append(car1)
    cars.append(car2)
    cars.append(car3)
    cars.append(car4)
    cars.append(car5)
    cars.append(car6)
```

```
Fog1 = net.addAccessPoint('Fog1', mac='00:00:00:00:00:01', ssid="handover",
                          mode="g", channel="1", datapath='user',
                          passwd='123456789a', encrypt='wpa2', ieee80211r='yes',
                          mobility_domain='a1b2', dpid='1',
                          position='150,200,0', inband=True)

Fog2 = net.addAccessPoint('Fog2', mac='00:00:00:00:00:02', ssid="handover",
                          mode="g", channel="1", datapath='user',
                          passwd='123456789a', encrypt='wpa2', ieee80211r='yes',
                          mobility_domain='a1b2', dpid='2',
                          position='250,200,0', color='r', inband=True)

Fog3 = net.addAccessPoint('Fog3', mac='00:00:00:00:00:03', ssid="handover",
                          mode="g", channel="1", datapath='user',
                          passwd='123456789a', encrypt='wpa2', ieee80211r='yes',
                          mobility_domain='a1b2', dpid='3',
                          position='320,200,0',color='g', inband=True)
```

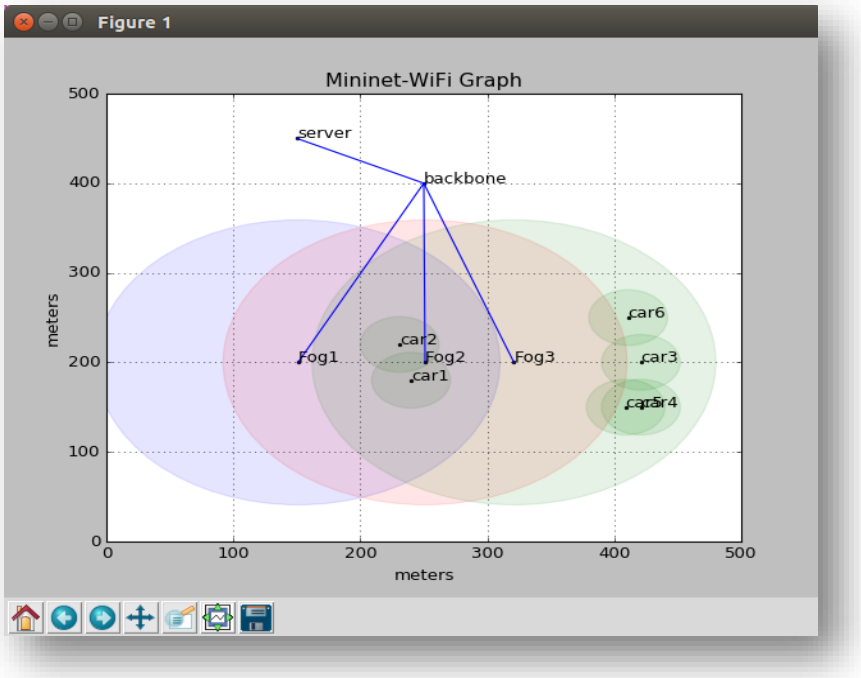
```
backbone = net.addSwitch('backbone', mac='00:00:00:00:00:04', dpid='4',
                         datapath='user', inband=True)
server = net.addHost('server', ip='10.0.0.100/8')
```

```
ip_c0 = '10.0.0.101'
ip_c1 = '10.0.0.102'
ip_c2 = '10.0.0.103'
ip_c3 = '10.0.0.104'
```

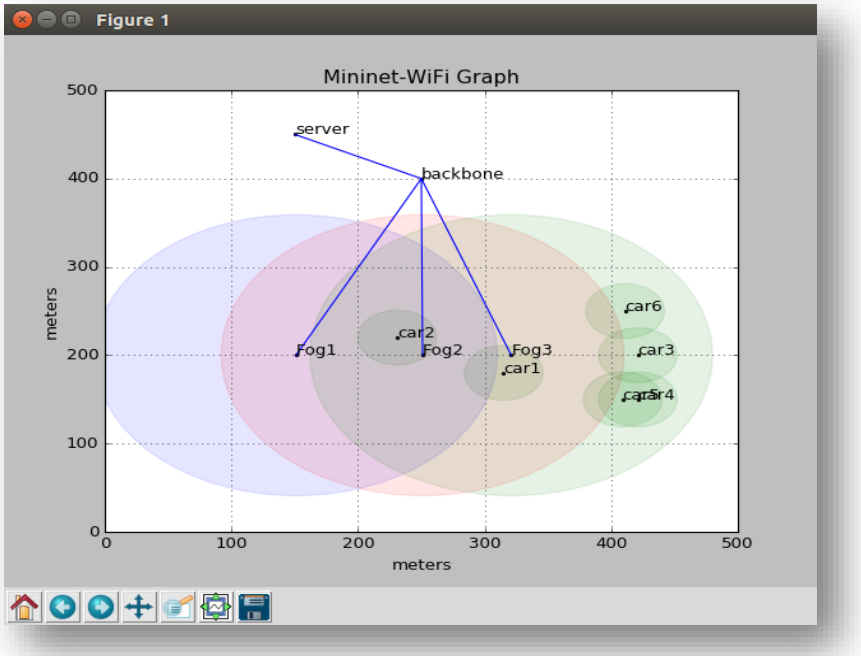
Τέλος, η μετακίνηση του car1 πραγματοποιείται ως εξής:

```
currentTime = time.time()
i = 60
j = 0
while j<3:
    if (time.time() - currentTime) >= i:
        currentTime = time.time()
        if j == 0:
            cars[0].setPosition('240,180,0')
        else:
            cars[0].setPosition('315,180,0')
        j+=1
```

Οπότε, καθορίζουμε τις χρονικές στιγμές που μετακινούμε το car1. Συγκεκριμένα ορίζουμε, χρονική στιγμή t=60 second, όπου το car1 μετακινείται κοντά στο Fog 2 (Εικόνα 20) και χρονική στιγμή t=120 second, όπου το car1 μετακινείται κοντά στο Fog 3 (Εικόνα 21).



Εικόνα 22. Μετακίνηση car1 (t=60 sec)



Εικόνα 23. Μετακίνηση car1 (t=120 sec)

Αφού έχουμε υλοποιήσει το infrastructure, συνεχίζουμε με την υλοποίηση της διαδικασίας του slicing. Το Ryu, επειδή είναι open source πρόγραμμα, μας δίνει τη δυνατότητα να αναπτύξουμε custom scripts σύμφωνα με τις ανάγκες μας. Συγκεκριμένα, αναπτύξαμε έναν κώδικα στον οποίο, ανάλογα με το αριθμό των οχημάτων που συσχετίζονται με ένα σταθμό βάσης, κάθε controller αποφασίζει αν μπορεί να εξυπηρετήσει επιπλέον οχήματα, που στέλνουν αίτημα να εξυπηρετηθούν από αυτόν, ώστε η ποιότητα της υπηρεσίας να μην υποβαθμιστεί.
Στο custom script ο *εκάστοτε controller συλλέγει πληροφορίες* από όλα τα οχήματα που εξυπηρετεί ο σταθμός βάσης.

```
if _udp.src_port == 8000: #Client to Controller
    _wifi = pkt.get_protocol(wifi.WiFiMsg)
    target_rssi = int(_wifi.target_rssi)
    rssi = int(_wifi.rssi)
    client_id = "%01d" % (int(_wifi.client[-2:]),)
    slicing = True
    self.logger.info("wifi msg:: client car%s, rssi %s, bssid %s, ssid %s,"
                     "target_bssid %s, target_rssi %s, load %s, target_load %s",
                     client_id, rssi, _wifi.bssid, _wifi.ssid,
                      _wifi.target_bssid, target_rssi, _wifi.load, _wifi.target_load)
```

Στη συνέχεια, θέτουμε την εξής συνθήκη: ο controller θα αποφασίζει ότι το Fog δεν μπορεί να εξυπηρετήσει αριθμό οχημάτων (αιτημάτων) μεγαλύτερο από αυτόν που ορίζεται στη μεταβλητή n_apps (στο σενάριο μας n_apps=2). Στην περίπτωση που δέχεται μεγαλύτερο αριθμό αιτημάτων από αυτόν της μεταβλητής ορίζουμε ότι ο controller του Fog στο οποίο στέλνει αίτημα σύνδεσης το όχημα θα απορρίψει το αίτημα και θα ενημερώσει τον controller της υφιστάμενης σύνδεσης ότι δεν μπορεί να εξυπηρετήσει το συγκεκριμένο όχημα στη δεδομένη χρονική στιγμή, ώστε να μην διακόψει την επικοινωνία. Έτσι, αποφεύγεται η υποβάθμιση της υπηρεσίας και τότε μιλάμε για *δημιουργία συνθηκών load–awareness handover*.

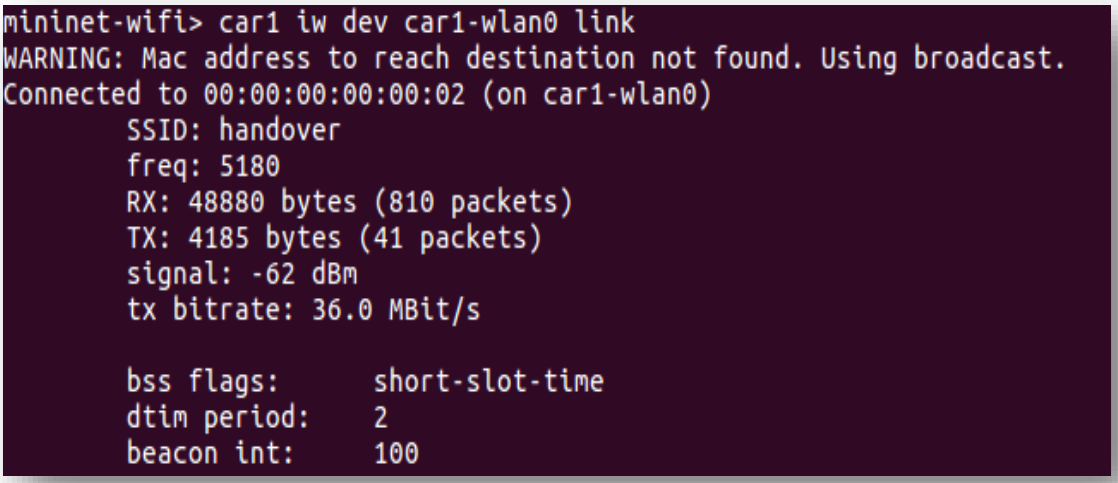
```
    n_aps=0
    if rssi < target_rssi and target_rssi > -70 and wifi.WiFiMsg.association:
        if wifi.WiFiMsg.association['car%s' % client_id] != _wifi.target_bssid:
            self.logger.info('%s car%s wpa_cli -i car%s-wlan0 scan '
                     '>/dev/null 2>&1' % (mn_wifi_dir, client_id, client_id))
            os.system('%s car%s wpa_cli -i car%s-wlan0 scan '
                     '>/dev/null 2>&1' % (mn_wifi_dir, client_id, client_id))
            os.system('%s car%s wpa_cli -i car%s-wlan0 scan_results '
                     '>/dev/null 2>&1' % (mn_wifi_dir, client_id, client_id))
            wifi.WiFiMsg.association['car%s' % client_id] = _wifi.target_bssid

    n_aps = int(subprocess.check_output('%s car%s wpa_cli -i car%s-wlan0 '
                                        'scan_results | wc -l'
                                        % (mn_wifi_dir, client_id, client_id),
                                        shell=True)) - 1

    if n_aps>=2 and 'car%s' % client_id in wifi.WiFiMsg.association and int(_wifi.target_load)<int(_wifi.load):
        if wifi.WiFiMsg.association['car%s' % client_id] == _wifi.target_bssid or slicing and int(_wifi.target_load)+1<int(_wifi.load):
            self.logger.info('%s car%s wpa_cli -i car%s-wlan0 roam %s >/dev/null 2>&1'
                     % (mn_wifi_dir, client_id, client_id, _wifi.target_bssid))
            os.system('%s car%s wpa_cli -i car%s-wlan0 roam %s >/dev/null 2>&1'
                     % (mn_wifi_dir, client_id, client_id, _wifi.target_bssid))
            wifi.WiFiMsg.association['car%s' % client_id] = ''
            ap_id = "%01d" % (int(_wifi.bssid[-2:]),)
            os.system('%s enb%s hostapd_cli -i enb%s-wlan1 deauthenticate '
                    '%s >/dev/null 2>&1' % (mn_wifi_dir, ap_id, ap_id, _wifi.client))
elif _udp.src_port == 8001: #Controller to Controller
    _wifi = pkt.get_protocol(wifi.WiFiCtoCMsg)
    self.logger.info("wifiCtoC msg:: client %s, bssid %s",
                     _wifi.client, _wifi.bssid)
```

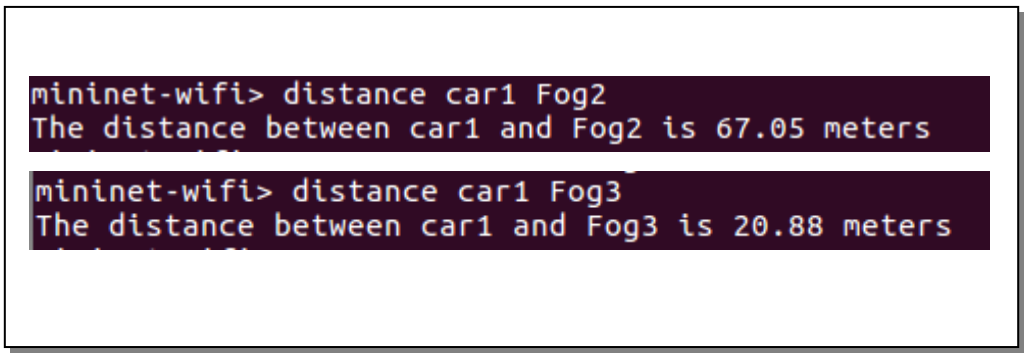
Στη συνέχεια ελέγχουμε την εγκυρότητα της συνθήκης με n_apps=2. Σύμφωνα με το σενάριο, τη χρονική στιγμή t=60 sec, το car 1 βρίσκεται κοντά στο Fog 2 και εξυπηρετείται από αυτό, το οποίο την ίδια στιγμή εξυπηρετεί και το car 2 (ίδιος σταθμός βάσης).

Άρα, σύμφωνα με τον αλγόριθμό μας δεν “σπάει” η συνθήκη μας, οπότε το handover γίνεται κανονικά (σύμφωνα με το καλύτερο RSSI) και το car 1 συσχετίζεται με το σταθμό βάσης του Fog 2. Αυτό μπορούμε να το διαπιστώσουμε δίνοντας την εντολή ***car1 iw dev car1-wlan0 link*** και να δούμε με ποιο σταθμό βάσης εξυπηρετείται το κάθε όχημα (Εικόνα 22).



Εικόνα 24. Εντολή car1 iw dev car1-wlan0 link

Στην περίπτωση που η συνθήκη μας είναι n_apps >2, τότε διαπιστώνουμε ότι την χρονική στιγμή t=120 sec, *το car 1 βρίσκεται ανάμεσα στο Fog 2 και στο Fog 3* (Εικόνα 23), αλλά πιο κοντά στο Fog 3 και πιο συγκεκριμένα βρίσκεται 67,05 μέτρα μακριά από το Fog 2 και 20,88 μέτρα από το Fog 3.



Εικόνα 25. Εντολή για εμφάνιση απόστασης από του σταθμούς βάσης

Όπως παρατηρούμε στην Εικόνα 21, ο σταθμός βάσης του Fog 3 εξυπηρετεί ακόμη τέσσερα οχήματα, σύμφωνα με τον αλγόριθμό μας, όταν ο αριθμός των εξ υπηρετούντων οχημάτων ξεπεράσει τα δυο οχήματα, ο controller αποφασίζει να μην εξυπηρετήσει επιπλέον οχήματα, αν και το car 1 έχει καλύτερο RSSI στο σταθμό βάσης του Fog 3 και συγκεκριμένα, στο σήμα είναι -60 dbm (Εικόνα 24), ενώ στο Fog 2 είναι -78 dbm (Εικόνα 25).

```
BSS 00:00:00:00:00:03(on car1-wlan0)
        TSF: 1567179912222991 usec (18138d, 15:45:12)
        freq: 5825
        beacon interval: 100 TUs
        capability: ESS Privacy ShortSlotTime (0x0411)
        signal: -60.00 dBm
        last seen: 0 ms ago
        Information elements from Probe Response frame:
        SSID: handover
        Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
        DS Parameter set: channel 1
        ERP: Barker_Preamble_Mode
        Extended supported rates: 24.0 36.0 48.0 54.0
        RSN:     * Version: 1
                 * Group cipher: CCMP
                 * Pairwise ciphers: CCMP
                 * Authentication suites: FT/PSK
                 * Capabilities: 1-PTKSA-RC 1-GTKSA-RC (0x0000)
        Extended capabilities:
                 * Extended Channel Switching
                 * SSID List
                 * Operating Mode Notification
```

Εικόνα 26. Πληροφορίες σήματος του σταθμού βάσης Fog 3

```
BSS 00:00:00:00:00:02(on car1-wlan0)
        TSF: 1567179908935195 usec (18138d, 15:45:08)
        freq: 2412
        beacon interval: 100 TUs
        capability: ESS Privacy (0x0011)
        signal: -78.00 dBm
        last seen: 0 ms ago
        Information elements from Probe Response frame:
        SSID: handover
        Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
        DS Parameter set: channel 1
        ERP: Barker_Preamble_Mode
        Extended supported rates: 24.0 36.0 48.0 54.0
        RSN:     * Version: 1
                 * Group cipher: CCMP
                 * Pairwise ciphers: CCMP
                 * Authentication suites: FT/PSK
                 * Capabilities: 1-PTKSA-RC 1-GTKSA-RC (0x0000)
        Extended capabilities:
                 * Extended Channel Switching
                 * SSID List
                 * Operating Mode Notification
```

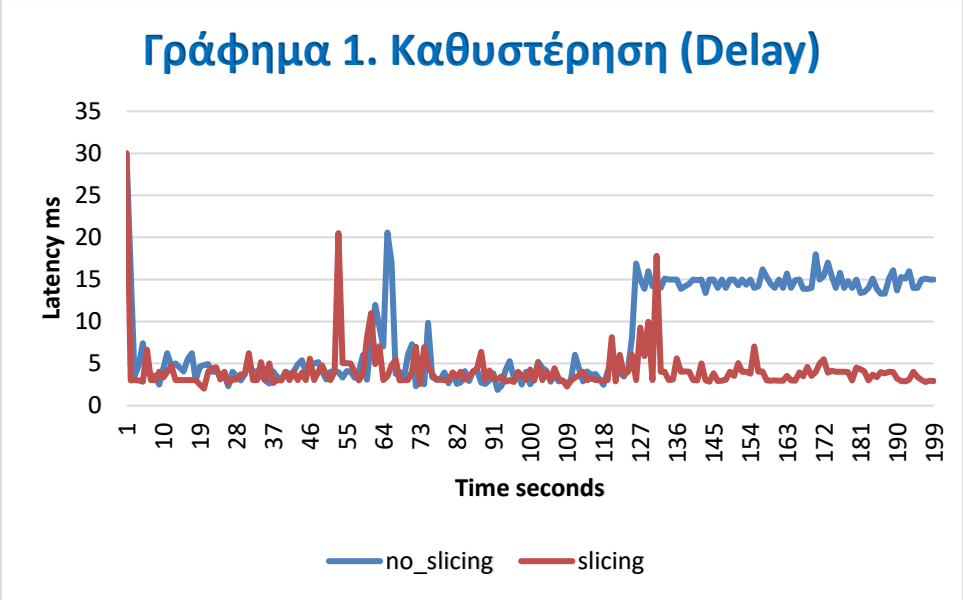
Εικόνα 27. Πληροφορίες σήματος του σταθμού βάσης Fog 2

6 – Αποτελέσματα

Για την εξαγωγή αποτελεσμάτων σε σχέση με την καθυστέρηση (delay) χρησιμοποιούμε το εργαλείο Ping, καθώς μετρά τον χρόνο αντίδρασης (reaction time) μιας σύνδεσης, δηλαδή, πόσο γρήγορη είναι η απόκριση σε ένα αίτημα (request) προς οποιαδήποτε δικτυακή συσκευή. Μονάδα μέτρησης είναι το millisecond (ms).

Έτσι, στο Γράφημα 1, ως προς την καθυστέρηση, παρατηρούμε ότι την χρονική στιγμή t= 120, στην περίπτωση που δεν κάνουμε χρήση του slicing, η καθυστέρηση ανέρχεται στα 15 ms, ενώ στην περίπτωση εφαρμογής του αλγορίθμου μας η καθυστέρηση ανέρχεται στα 5 ms.

Διαπιστώνουμε ότι η υπηρεσία μετάδοσης video από τον server λαμβάνεται από τα οχήματα με μεγάλη καθυστέρηση στην πρώτη περίπτωση, οπότε προκαλεί μεγάλα προβλήματα στην ποιότητα της εμπειρίας (QoE).

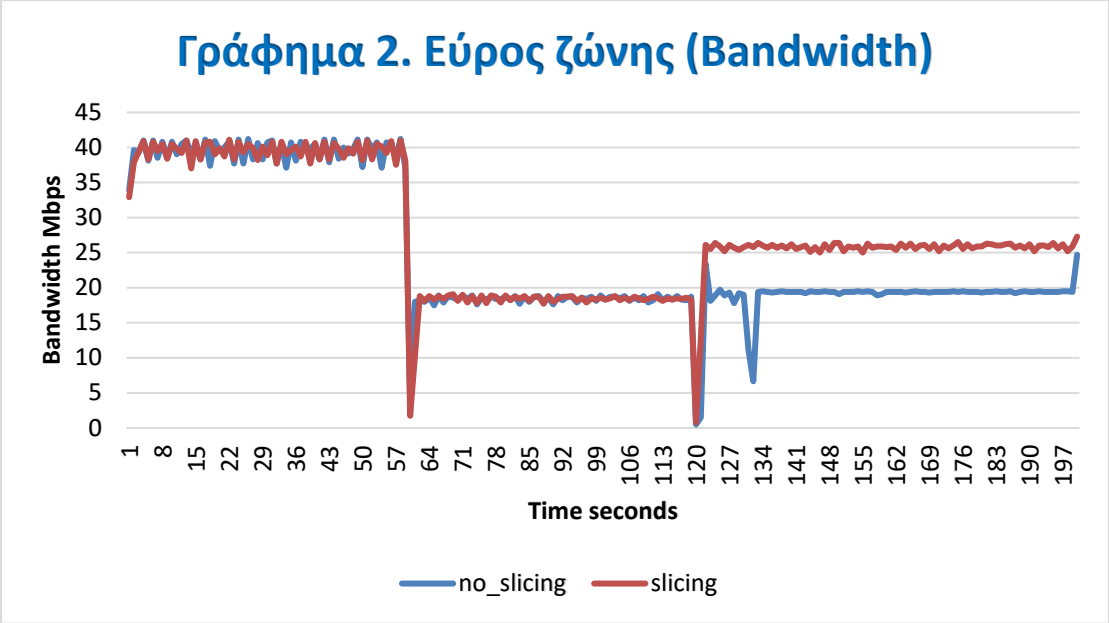


Για την εξαγωγή αποτελεσμάτων σε σχέση με το εύρος ζώνης (bandwidth) και τον ρυθμό απώλειας πακέτων (packet loss ratio) χρησιμοποιούμε το εργαλείο Iperf, ένα πρόγραμμα γραμμής εντολών, χαρακτηριστικό του οποίου είναι ότι τρέχει ταυτόχρονα σε δύο υπολογιστές, όπου ο ένας λειτουργεί σαν server και ο άλλος σαν

client και μπορεί να υποστηρίζει τόσο συνδέσεις TCP όσο και UDP μετρώντας το εύρος ζώνης και τον ρυθμό απώλειας πακέτων.

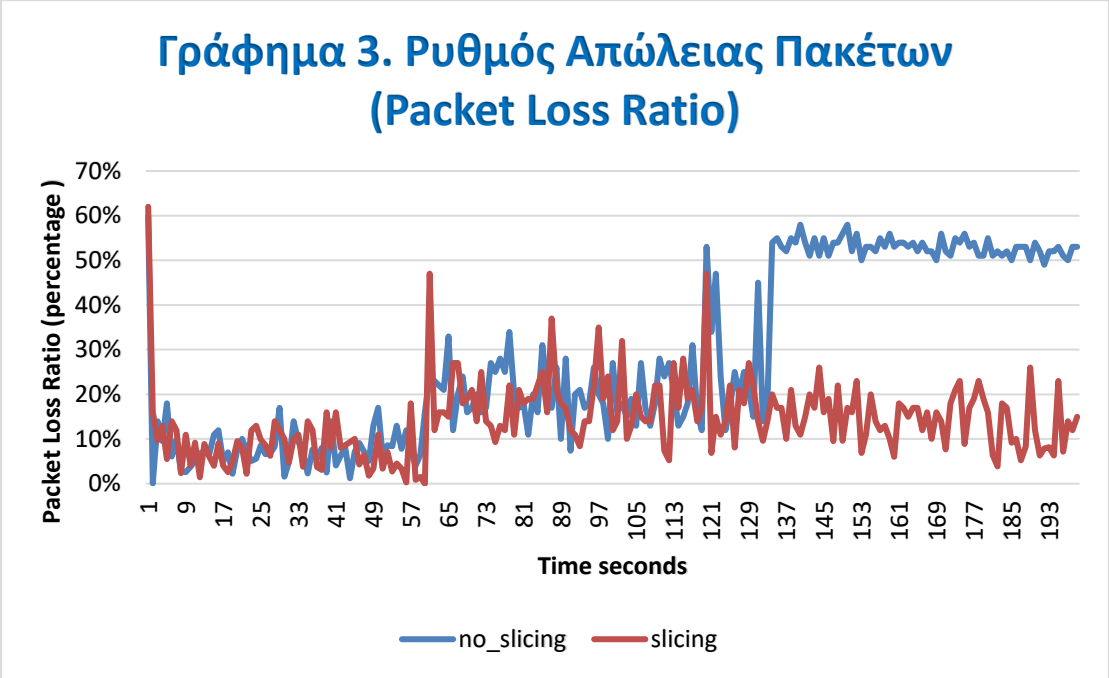
Έτσι στο Γράφημα 2, παρατηρούμε ότι την χρονική στιγμή t=120, στην περίπτωση που κάνουμε χρήση του αλγόριθμου μας, το μέγιστο εύρος ζώνης που μπορεί να κάνει χρήση το car 1 για την υπηρεσία μετάδοσης του video από τον server ανέρχεται στα 25 Mbps, αντίθετα, παρατηρούμε ότι όταν δεν γίνεται η χρήση του network slicing, το διαθέσιμο εύρος ζώνης (bandwidth) δεν υπερβαίνει τα 20 Mbps.

Έτσι, διαπιστώνουμε ότι η ποιότητα της υπηρεσίας είναι σαφώς καλύτερη όταν γίνεται η χρήση του network slicing, αφού ο server μπορεί για την υπηρεσία του να αξιοποιήσει περισσότερο bandwidth, ώστε να βελτιώσει την ποιότητα του video, π.χ από 360p σε 720p.

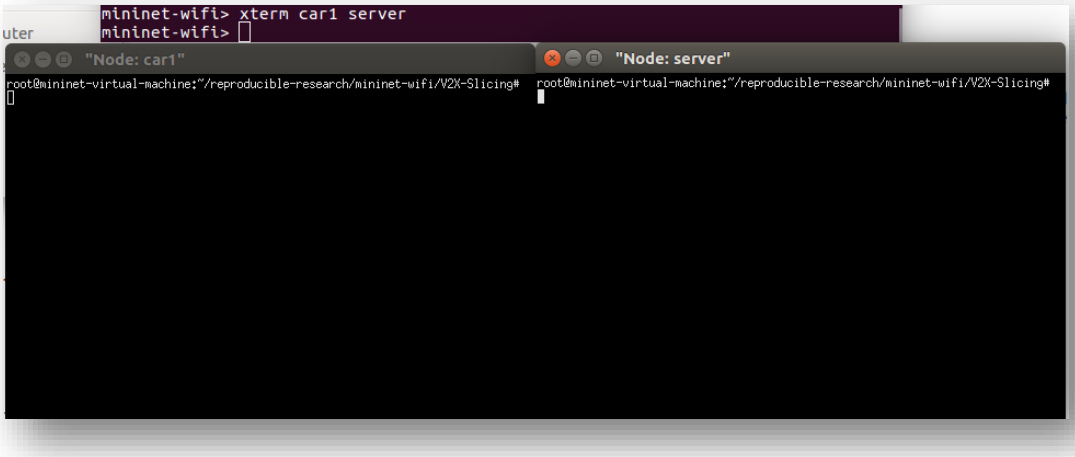


Επίσης, στο Γράφημα 3, παρατηρούμε, ελέγχοντας την ίδια χρονική στιγμή t= 120, όπως και στα προηγούμενα γραφήματα, ότι την χρονική στιγμή t=120, στην περίπτωση που δεν γίνεται χρήση network slicing, ο δείκτης απώλειας πακέτων ανέρχεται σε ποσοστό μεταξύ 50% και 60%, ποσοστό πολύ μεγάλο. Δηλαδή, στα 2 πακέτα που στέλνονται από το server προς car1, το ένα χάνεται. Αντιθέτως, όταν γίνεται η χρήση του network slicing, ο δείκτης απώλειας πακέτων ανέρχεται σε ποσοστό μεταξύ 10% και 20%. Η αιτία που ο δείκτης απώλειας πακέτων είναι τόσο

μεγάλος στην πρώτη περίπτωση, οφείλεται στην καθυστέρηση που αναλύσαμε στο διάγραμμα Delay, όπου πολλά πακέτα γίνονται time out και με αποτέλεσμα να απορρίπτονται (drop).

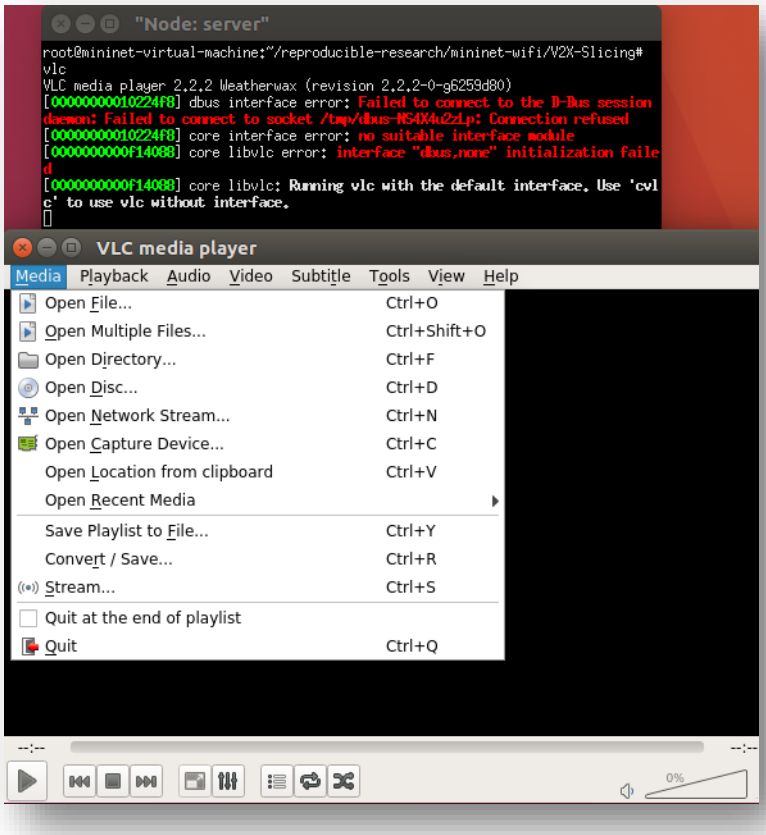


Τέλος, για να αξιολογήσουμε τη βελτιστοποίηση της ποιότητας υπηρεσία (QoS), θα μεταδώσουμε ένα video από τον server στο car1. Το πρόγραμμα που θα χρησιμοποιήσουμε για την πραγματοποίηση της μετάδοσης είναι το VLC. Το VLC είναι ένα πολυμεσικό πρόγραμμα ανοικτού κώδικα, το οποίο υποστηρίζει πρωτόκολλα streaming και μας βοηθά στη μετάδοση video σε ένα τοπικό δίκτυο. Ωστόσο, στην περίπτωση του σεναρίου μας δημιουργείται ο εξής περιορισμός: το VLC δεν είναι δυνατόν να εκτελεστεί απευθείας μέσα από τις γραμμές εντολών του Mininet-WiFi, για τον λόγο αυτόν δίνουμε την εντολή xterm server car1 στο Mininet-Wifi, ώστε να ανοίξει νέα γραμμή εντολών, η οποία θα μας δίνει τη δυνατότητα να τρέξουμε διάφορα προγράμματα, όπως το VLC, χωρίς να βγούμε από το Mininet-Wifi.



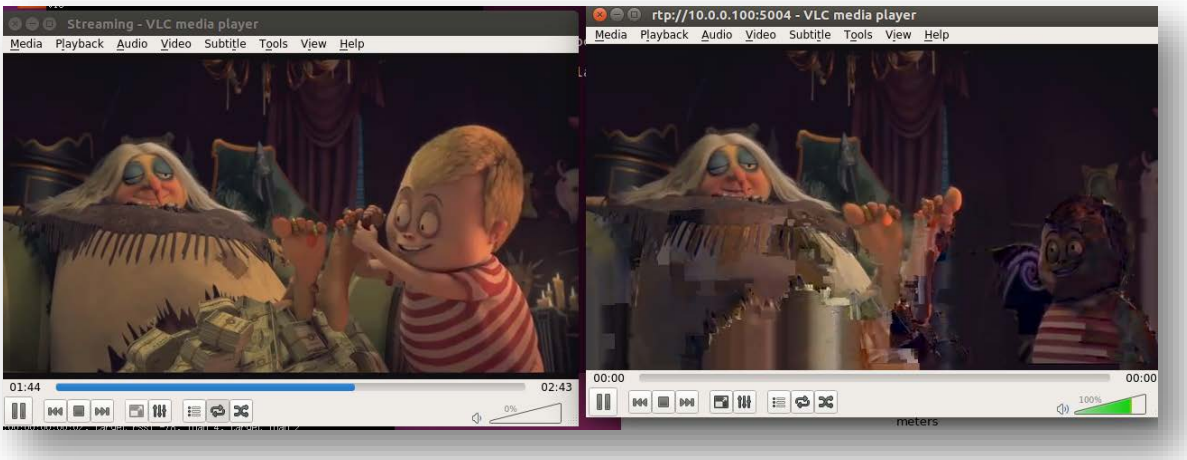
Εικόνα 28. Εντολή xterm

Έτσι, δίνοντας την εντολή VLC, ανοίγει το VLC πρόγραμμα, επιλέγουμε Media και στην συνέχεια Stream. Το video που χρησιμοποιούμε έχει συμπίεση MPEG-4 και ανάλυση ποιότητας 360p.

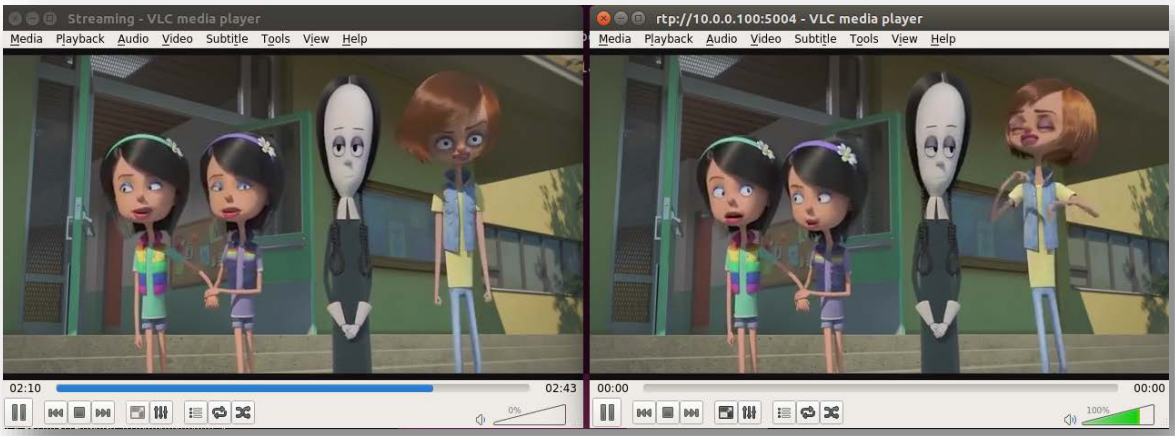


Εικόνα 29. Εντολή εκτέλεσης του προγράμματος VLC

Αρχίζει η αναπαραγωγή και η μετάδοση του video από τον server προς το όχημα (car 1) που έστειλε το αίτημα. Παρατηρούμε ότι η αρχική ποιότητα μετάδοσης video από τον Server (Εικόνα 30, αριστερά), στην περίπτωση που δεν χρησιμοποιείται slicing, μειώνεται δραματικά και φτάνει πιξελοποιημένη κατά 50% στο car 1 (Εικόνα 30, δεξιά). Αυτό οφείλεται στο υψηλό ποσοστό απώλειας πακέτων λόγω της καθυστέρησης και του εύρους ζώνης που αναλύσαμε παραπάνω. Στην περίπτωση που χρησιμοποιηθεί slicing, παρατηρούμε ότι η αρχική ποιότητα μετάδοσης video από τον Server (Εικόνα 31, αριστερά), διατηρεί την αρχική της ποιότητα σε μεγάλο βαθμό, αφού το ποσοστό απώλειας πακέτων είναι μόνον 10%-20% (Εικόνα 31, δεξιά). Συμπεραίνουμε, επομένως, ότι η απώλεια πακέτων που διαπιστώνεται δεν είναι σημαντική για την ποιότητα του video όταν χρησιμοποιείται slicing σε σύγκριση με την περίπτωση που δεν χρησιμοποιείται slicing και επομένως, βελτιστοποιείται η ποιότητα των υπηρεσιών (QoS) στον τελικό χρήστη.



Εικόνα 30. Ποιότητα αρχικής μετάδοσης video από τον Server (αριστερά). Ποιότητα λήψης video στο όχημα (δεξιά) (χωρίς χρήση slicing).



Εικόνα 31. Ποιότητα αρχικής μετάδοσης video από τον Server (αριστερά). Ποιότητα λήψης video στο όχημα (δεξιά) (με χρήση slicing).

7 – Συμπεράσματα και Προτάσεις για μελλοντική έρευνα

Στην παρούσα εργασία προτείναμε μια αρχιτεκτονική VANET, η οποία υποστηρίζει network slicing σε V2X επικοινωνίες. Η αρχιτεκτονική αυτή βασίζεται σε τεχνολογίες λογισμικοποίησης δικτύων (network softwarization technologies), όπως το SDN και το NFV. Την αρχιτεκτονική αυτή, την υλοποιήσαμε σε περιβάλλον προσομοίωσης με την βοήθεια του προγράμματος Mininet–Wifi και δημιουργήσαμε έναν αλγόριθμο για την υλοποίηση network slicing με σκοπό την αυτοματοποίηση της διαδικασίας της διαπομπής (handover) μεταξύ των FOG από τους SDN controllers, σύμφωνα με τον φόρτο χρήσης του κάθε FOG .

Από την προσομοίωση αυτή, διαπιστώσαμε ότι υλοποίηση του network slicing στα VANETs έχει αρκετά πλεονεκτήματα, όπως μείωση της καθυστέρησης (delay), βελτιστοποίηση της χρήση του εύρους ζώνης (bandwidth) καθώς και καλύτερη διαχείριση του συνολικού δικτύου.

Συμπεραίνουμε ότι το network slicing συμβάλλει στην εξέλιξη των VANETs στα καινούρια δίκτυα 5G, παρέχοντας δυνατότητες για νέες εφαρμογές, όπως αυτή που προτείνει το σενάριο προσομοίωσής μας, load awareness handover, για την βελτιστοποίηση της ποιότητας των παρεχόμενων υπηρεσιών (QoS) και την ποιότητα της εμπειρίας του τελικού χρήστη (QoE).

Δεδομένης της διαρκούς εξέλιξης της τεχνολογίας 5G και Mobile Edge Computing, τεχνολογίες που θα ευρεία εφαρμογή στο άμεσο μέλλον, θα είχε ενδιαφέρον να διερευνηθεί περαιτέρω η προσθήκη του Mobile Edge Computing και τα επιπλέον πλεονεκτήματα που προφέρει το cloud computing στα VANETs.

Αναφορές

5G PPP Architecture Working Group (White Paper). View on 5G Architecture.

EuCNC 2016 conference. Version for public consultation, updated version available on July 1st 2016 (https://5g-ppp.eu/white-papers/).

Ahmed, A., Ahmed, E., 2016. A survey on mobile edge computing. In: Proceedings of the IEEE International Conference on Intelligent Systems and Control, pp. 1–8.

Ashiho, L. S. (2003) Mobile technology: Evolution from 1G to 4G, *Electronics for You*, 94–98.

Beck, M.T., Feld, S., Linnhoff-Popien, C., Ptzschler, U., 2016. Mobile edge computing. Inform.-Spektrum 39 (2), 108–114.

Borcoci E., Vochin M., Obreja S. (2018) Mobile Edge Computing versus Fog Computing in Internet of Vehicles. Conference: The Tenth International Conference on Advances in Future Internet AFIN 2018, Venice, Italy.

ETSI official site, https://www.etsi.org/committee/1418-3gpp. Accessed 24/8/2019.

December 2008, www.itu.int/pub/R-REP-M.2135-2008

CELTIC / CP5-026 WINNER+ project, “Final Channel Models”, Deliverable D5.3, June 2010, http://projects.celtic-initiative.org/ winner+/deliverables_winnerplus.html/

Chaves L.J., Eichemberger V.M., Calciolari Garcia I., and Madeira E.R.M. (2014). Integrating OpenFlow to LTE: some issues toward Software-Defined Mobile Networks(process 118198/2014-9).

Debashis D. (2016). *Mobile Cloud Computing. Architectures, Algorithms and Applications*. CRC PRESS – Taylor & Francis Group.

Ericsson, Nokia Networks, “Further LTE physical layer enhancements for MTC,”Work Item RP-141660, 3GPP TSG RAN Meeting #65, September 2014.

Ericsson, Ericsson Mobility Report, Report No. EAB-15:037849, November 2015, www.ericsson.com/res/docs/2015/mobility-report/ericsson-mobility-report-nov-2015.pdf

Faezipour, M., Nourani, M., Saeed, A., and Addepalli, S. (2012). “Progress and challenges in intelligent vehicle area networks,” *Commun. ACM*, vol. 55, no. 2, p. 90.

Fontes R., Afzal S., Brito, S.H.B., Santos, M.A.S., Rothenberg, C.E. (2015). Mininet-WiFi: Emulating Software-Defined Wireless Networks. 978-3-901882-77-7 c 2015

IFIP. https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/fontes2015mininet.pdf

GSMA, Definitive data and analysis for the mobile industry [Online] https://gsmaintelligence.com/

Habibi M.A., Han B., and Schotten HD (2017). Network Slicing in 5G Mobile Communication: Architecture, Profit Modeling, and Challenges. *ArXiv*:1707.00852.

ICT-317669 METIS project, "Simulation guidelines," Deliverable D6.1, November 2013, www.metis2020.com/documents/deliverables/

ICT-317669 METIS project, "Scenarios, requirements and KPIs for 5G mobile and wireless system," Deliverable D1.1, May 2013, www.metis2020.com/documents/deliverables/

International Telecommunications Union Radio (ITU-R), "Requirements related to technical performance for IMT-Advanced radio interface(s)," Report ITUR M.2134, December 2008, www.itu.int/pub/R-REP-M.2134-2008

International Telecommunications Union Radio (ITU-R), "Guidelines for evaluation of radio interface technologies for IMT-Advanced," Report ITU-R M.2135,

International Telecommunications Union Radio (ITU-R), "Propagation data and prediction methods for the planning of short-range outdoor radiocommunication systems and radio local area networks in the frequency range 300MHz to 100GHz", Report ITU-R P.1411, October 1999, www.itu.int/rec/R-REC-P.1411-0-199910-S

Kanani, P., Shah, K., and Kaul, V. A survey on evolution of mobile networks: 1G to 4G, *International Journal on Science and Research*, 3(2), 802–810, 2014.

Khan A.A., Abolhasan M. (2018). 5G Next generation VANETs using SDN and Fog Computing Framework (2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC).

Li, X., Abudulla, G., Rosli, S., and Omar, Z. (2009) The future of mobile wireless communication networks, in *IEEE International Conference on Communication Software and Networks*, Macau, pp. 554–557.

Miki, T., Ohya, T., Yoshino, H., and Umeda, N. (2005) The overview of the 4th generation mobile communication system, *International Conference on Communications*, 2(3), 1551–1553.

Moses, K. B. (2014) Mobile communication evolution, *International Journal of Modern Education and Computer Science*, 1, 25–33.

Nguyen V-G., Brunstrom A., Grinnemo K-J, Taheri J. (2017). SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey (IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 19, NO. 3, THIRD QUARTER 2017).

Ohmori, S., Yamao, Y., and Nakajima, N. (2000) The future generations of mobile communications based on broadband access technologies, *IEEE Communications Magazine*, 38(12), 134–142.

Open Networking Foundation (ONF), "Software-Defined Networking: The New Norm for Networks", White Paper, Apr., 2013.

Osseiran A., Monserrat J. F., and Marsch P. (eds.) (2016). 5G Mobile and Wireless Communications Technology. Cambridge University Press.

Pengfei Hua, Sahraoui Dhelima, Huansheng Ninga, Tie Qiud (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications 98 (*2017) 27–42. (http://dx.doi.org/10.1016/j.jnca.2017.09.002)

Pepelnjak I., "OpenFlow and SDN: hype, useful tools or panacea?", NIL Data Communications, [Online] http://tv.nil.si/openflow_and_sdn_hype_useful_tools_or_panacea/

Qualcomm Incorporated, "New work item: Narrowband IOT (NB-IOT)," Work Item RP-151621, 3GPP TSG RAN Meeting #69, September 2015.

Sharma, K.K., Kumar, C., and Kumar, D. (2013). An overview on 5G technologies, *International Journal of Information Technology and Knowledge Management*, 6(2), 171–174.

Sassan A. (2019). *5G NR - Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*. Academic Press Elsevier.

Sharma, P (2013). Evolution of mobile wireless communication networks-1G to 5G as well as future prospective of next generation communication network, *International Journal of Computer Science and Mobile Computing*, 2(8), 47–53.

Shi, W., Cao, J., Zhang, Q., Li, Y., 2016. Edge computing: vision and challenges. IEEE Internet Things J. 3 (5), 637–646.

Truong N.B., Lee G.M., and Ghamri-Doudane Y. (2015). Software Defined Networking-based Vehicular Adhoc Network with Fog Computing (978-3-901882-76-0@2015 IFIP).

Zhang, Q. (2017). *A Pervasive Prediction Model Prediction Model Prediction Model Prediction Model for VANET* (thesis), Nottingham Trent University.

CISCO - https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html#~services

JUNIPER - https://www.juniper.net/us/en/products-services/what-is/network-functions-virtualization/

https://noviflow.com/the-basics-of-sdn-and-the-openflow-network-architecture/

https://github.com/ramonfontes/manual-mininet-wifi/raw/master/mininet-wifi-draft-manual.pdf

OSRG GITHUB - https://osrg.github.io/ryu/

Παράρτημα Α – Απαιτήσεις για την εγκατάσταση του Mininet-WiFi

Το Mininet-WiFi λειτουργεί σε οποιαδήποτε έκδοση Ubuntu από 14.04 και πάνω.

Παράρτημα Β – Οδηγίες εγκατάστασης του Mininet-WiFi

Θα πρέπει να ακολουθηθούν οι παρακάτω τέσσερις εντολές στο cli για την εγκατάσταση του Mininet-WiFi.

1. sudo apt-get install git
2. git clone https://github.com/intrig-unicamp/mininet-wifi
3. cd mininet-wifi
4. sudo util/install.sh –Wlnfv

Παράρτημα Γ – Python script για την υλοποίηση του σεναρίου μας

```
#!/usr/bin/python

import time
import os

from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mn_wifi.cli import CLI_wifi
from mn_wifi.net import Mininet_wifi
from mn_wifi.link import wmediumd
from mininet.term import makeTerm, cleanUpScreens
from mn_wifi.wmediumdConnector import interference


class InbandController( RemoteController ):

    def checkListening( self ):
        "Overridden to do nothing."
        return


def topology():

    os.system('service network-manager stop')

    "Create a network."
    net = Mininet_wifi(controller=InbandController,
                       link=wmediumd,
```

```
            wmediumd_mode=interference
            )

ip_c0 = '10.0.0.101'
ip_c1 = '10.0.0.102'
ip_c2 = '10.0.0.103'
ip_c3 = '10.0.0.104'

info("*** Creating nodes\n")
cars = []
car1 = net.addStation('car1', mac='02:00:00:00:00:01',
            position='130,180,0') #, active_scan=1)

car2 = net.addStation('car2', mac='02:00:00:00:00:02',
            position='230,220,0') #, active_scan=1)

car3 = net.addStation('car3', mac='02:00:00:00:00:03',
            position='420,200,0') #, active_scan=1)

car4 = net.addStation('car4', mac='02:00:00:00:00:04',
            position='420,150,0') #, active_scan=1)

car5 = net.addStation('car5', mac='02:00:00:00:00:05',
            position='408,150,0') #, active_scan=1)

car6 = net.addStation('car6', mac='02:00:00:00:00:05',
            position='410,250,0') #, active_scan=1)
cars.append(car1)
cars.append(car2)
cars.append(car3)
cars.append(car4)
cars.append(car5)
```

```
cars.append(car6)


enb1 = net.addAccessPoint('Fog1', mac='00:00:00:00:00:01', ssid="handover",
                    mode="g", channel="1", datapath='user',
                    passwd='123456789a', encrypt='wpa2', ieee80211r='yes',
                    mobility_domain='a1b2', dpid='1',
                    position='150,200,0', inband=True)

enb2 = net.addAccessPoint('Fog2', mac='00:00:00:00:00:02', ssid="handover",
                    mode="g", channel="1", datapath='user',
                    passwd='123456789a', encrypt='wpa2', ieee80211r='yes',
                    mobility_domain='a1b2', dpid='2',
                    position='250,200,0', color='r', inband=True)

enb3 = net.addAccessPoint('Fog3', mac='00:00:00:00:00:03', ssid="handover",
                    mode="g", channel="1", datapath='user',
                    passwd='123456789a', encrypt='wpa2', ieee80211r='yes',
                    mobility_domain='a1b2', dpid='3',
                    position='320,200,0',color='g', inband=True)

backbone = net.addSwitch('backbone', mac='00:00:00:00:00:04', dpid='4',
                     datapath='user', inband=True)
server = net.addHost('server', ip='10.0.0.100/8')

h1 = net.addHost('h1', ip=ip_c0)
h2 = net.addHost('h2', ip=ip_c1)
h3 = net.addHost('h3', ip=ip_c2)
h4 = net.addHost('h4', ip=ip_c3)

c0 = net.addController('c0', controller=InbandController,
                    port=6690, ip=ip_c0)
```

```
c1 = net.addController('c1', controller=InbandController,
                       port=6691, ip=ip_c1)
c2 = net.addController('c2', controller=InbandController,
                       port=6692, ip=ip_c2)
c3 = net.addController('c3', controller=InbandController,
                       port=6693, ip=ip_c3)
net.setPropagationModel(model="logDistance", exp=3.4)

info("*** Configuring wifi nodes\n")
net.configureWifiNodes()



backbone.plot(position='250,400,0')
server.plot(position='150,450,0')

info("*** Associating Stations\n")
net.addLink(backbone, enb1)
net.addLink(backbone, enb2)
net.addLink(backbone, enb3,bw=20, delay='5ms',loss=5)
net.addLink(backbone, server)
net.addLink(h1, enb1)
net.addLink(h2, enb2)
net.addLink(h3, enb3)
net.addLink(h4, backbone)

net.plotGraph(max_x=500, max_y=500)

info("*** Starting network\n")
net.build()
enb1.start([c0])
enb2.start([c1])
enb3.start([c2])
```

```
    backbone.start([c3])

    makeTerm(h1, cmd="bash -c 'cd ryu && ./run.sh h1;'")
    makeTerm(h2, cmd="bash -c 'cd ryu && ./run.sh h2;'")
    makeTerm(h3, cmd="bash -c 'cd ryu && ./run.sh h3;'")
    makeTerm(h4, cmd="bash -c 'cd ryu && ./run.sh h4;'")

    time.sleep(3)

    enb1.cmd('sysctl net.ipv4.ip_forward=1')
    enb2.cmd('sysctl net.ipv4.ip_forward=1')
    enb3.cmd('sysctl net.ipv4.ip_forward=1')
    backbone.cmd('sysctl net.ipv4.ip_forward=1')

    enb1.cmd('ifconfig enb1-eth3 10.0.0.201')
    enb2.cmd('ifconfig enb2-eth3 10.0.0.202')
    enb3.cmd('ifconfig enb3-eth3 10.0.0.203')
    backbone.cmd('ifconfig backbone-eth5 10.0.0.204')

    enb1.cmd('route add 10.0.0.101 dev enb1-eth3')
    enb2.cmd('route add 10.0.0.102 dev enb2-eth3')
    enb3.cmd('route add 10.0.0.103 dev enb3-eth3')
    backbone.cmd('route add 10.0.0.104 dev backbone-eth5')

    cars[0].cmd('iw dev %s-wlan0 interface '
            'add %s-mon0 type monitor'
            % (cars[0].name, cars[0].name))
    cars[0].cmd('ifconfig %s-mon0 up' % cars[0].name)

    enb1.cmd('ovs-ofctl                add-flow                "enb1"
in_port=1,udp,tp_src=8000,actions=controller')
```

```
    enb2.cmd('ovs-ofctl add-flow "enb2" in_port=1,udp,tp_src=8000,actions=controller')
    enb3.cmd('ovs-ofctl add-flow "enb3" in_port=1,udp,tp_src=8000,actions=controller')
    backbone.cmd('ovs-ofctl add-flow "backbone" in_port=1,actions=output:4')
    backbone.cmd('ovs-ofctl add-flow "backbone" in_port=2,actions=output:4')
    backbone.cmd('ovs-ofctl add-flow "backbone" in_port=3,actions=output:4')

    makeTerm(cars[0], cmd="bash -c 'ping 10.0.0.100 -c200 > ping.txt;'")
    makeTerm( server, cmd="bash -c 'iperf -s -i 1 -u > bandwidth_slicing.txt ;'" )
    makeTerm(cars[0], cmd="bash -c 'iperf -c 10.0.0.100 -u -b 50M -t 200 ;'")

    cars[0].cmd('./%s.py &' % cars[0].name)

    currentTime = time.time()
    i = 60
    j = 0
    while j<3:
        if (time.time() - currentTime) >= i:
            currentTime = time.time()
            if j == 0:
                cars[0].setPosition('240,180,0')
            else:
                cars[0].setPosition('315,180,0')
            j+=1
        if j == 2 and (time.time() - currentTime) == 1:
            cars[0].cmd('wpa_cli -i car1-wlan0 scan')
        # force association due to better rssi
        if j == 2 and (time.time() - currentTime) == 5:
            cars[0].cmd('wpa_cli -i car1-wlan0 scan_results')
            cars[0].cmd('wpa_cli -i car1-wlan0 roam 00:00:00:00:00:03')
```

```
    info("*** Running CLI\n")
    CLI_wifi(net)

    os.system('pkill -f \"xterm -title\"')
    os.system('pkill ryu-manager')
    os.system('service network-manager start')

    info("*** Stopping network\n")
    net.stop()


if __name__ == '__main__':
    setLogLevel('info')
    topology()
```

Παράρτημα Δ – Python script για την υλοποίηση του Load-Aware Slice Handover Decision

```
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import subprocess
import os

from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ipv4
from ryu.lib.packet import wifi
from ryu.lib.packet import tcp
from ryu.lib.packet import udp
```

```
from ryu.lib.packet import ether_types
from scapy import all as scapy


class wifiAPP(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(wifiAPP, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

        # install table-miss flow entry
        #
        # We specify NO BUFFER to max_len of the output action due to
        # OVS bug. At this moment, if we specify a lesser number, e.g.,
        # 128, OVS will send Packet-In with invalid buffer_id and
        # truncated packet data. In that case, we cannot output packets
        # correctly.  The bug has been fixed in OVS v2.1.0.
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                          ofproto.OFPCML_NO_BUFFER)]
        self.add_flow(datapath, 0, match, actions)

    def add_flow(self, datapath, priority, match, actions, buffer_id=None):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
```

```
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                priority=priority, match=match,
                                instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst)
    datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    # If you hit this you might want to increase
    # the "miss_send_length" of your switch
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        # ignore lldp packet
        return
    dst = eth.dst
```

```
src = eth.src
dpid = datapath.id

self.mac_to_port.setdefault(dpid, {})
mn_wifi_dir = '~/mininet-wifi/util/m'

_ipv4 = pkt.get_protocol(ipv4.ipv4)
if hasattr(_ipv4, 'proto'):
   if _ipv4.proto == 17:
      _udp = pkt.get_protocol(udp.udp)
      if _udp.src_port == 8000: #Client to Controller
         _wifi = pkt.get_protocol(wifi.WiFiMsg)
         target_rssi = int(_wifi.target_rssi)
         rssi = int(_wifi.rssi)
         client_id = "%01d" % (int(_wifi.client[-2:]),)
         slicing = True
         self.logger.info("wifi msg:: client car%s, rssi %s, bssid %s, ssid %s,"
                  "target_bssid %s, target_rssi %s, load %s, target_load %s",
                  client_id, rssi, _wifi.bssid, _wifi.ssid,
                  _wifi.target_bssid, target_rssi, _wifi.load, _wifi.target_load)
         if rssi > target_rssi:
            wifi.WiFiMsg.association['car%s' % client_id] = _wifi.bssid

         n_aps=0
         if rssi < target_rssi and target_rssi > -70 and wifi.WiFiMsg.association:
            if wifi.WiFiMsg.association['car%s' % client_id] != _wifi.target_bssid:
               self.logger.info('%s car%s wpa_cli -i car%s-wlan0 scan '
                     '>/dev/null 2>&1' % (mn_wifi_dir, client_id, client_id))
               os.system('%s car%s wpa_cli -i car%s-wlan0 scan '
                     '>/dev/null 2>&1' % (mn_wifi_dir, client_id, client_id))
               os.system('%s car%s wpa_cli -i car%s-wlan0 scan_results '
                     '>/dev/null 2>&1' % (mn_wifi_dir, client_id, client_id))
```

```
                wifi.WiFiMsg.association['car%s' % client_id] = _wifi.target_bssid

            n_aps = int(subprocess.check_output('%s car%s wpa_cli -i car%s-wlan0 '
                                    'scan_results | wc -l'
                                    % (mn_wifi_dir, client_id, client_id),
                                    shell=True)) - 1

            if n_aps>=2 and 'car%s' % client_id in wifi.WiFiMsg.association and
int(_wifi.target_load)<int(_wifi.load):
                if wifi.WiFiMsg.association['car%s' % client_id] == _wifi.target_bssid or
slicing and int(_wifi.target_load)+1<int(_wifi.load):
                    self.logger.info('%s car%s wpa_cli -i car%s-wlan0 roam %s >/dev/null
2>&1'
                            % (mn_wifi_dir, client_id, client_id, _wifi.target_bssid))
                    os.system('%s car%s wpa_cli -i car%s-wlan0 roam %s >/dev/null
2>&1'
                            % (mn_wifi_dir, client_id, client_id, _wifi.target_bssid))
                    wifi.WiFiMsg.association['car%s' % client_id] = ''
                    ap_id = "%01d" % (int(_wifi.bssid[-2:]),)
                    os.system('%s enb%s hostapd_cli -i enb%s-wlan1 deauthenticate '
                            '%s >/dev/null 2>&1' % (mn_wifi_dir, ap_id, ap_id, _wifi.client))
            elif _udp.src_port == 8001: #Controller to Controller
                _wifi = pkt.get_protocol(wifi.WiFiCtoCMsg)
                self.logger.info("wifiCtoC msg:: client %s, bssid %s",
                            _wifi.client, _wifi.bssid)


    # learn a mac address to avoid FLOOD next time.
    self.mac_to_port[dpid][src] = in_port

    if dst in self.mac_to_port[dpid]:
        out_port = self.mac_to_port[dpid][dst]
```

```
else:
    out_port = ofproto.OFPP_FLOOD

actions = [parser.OFPActionOutput(out_port)]

# install a flow to avoid packet_in next time
if out_port != ofproto.OFPP_FLOOD:
    match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
    # verify if we have a valid buffer_id, if yes avoid to send both
    # flow_mod & packet_out
    if msg.buffer_id != ofproto.OFP_NO_BUFFER:
        self.add_flow(datapath, 1, match, actions, msg.buffer_id)
        return
    else:
        self.add_flow(datapath, 1, match, actions)
data = None
if msg.buffer_id == ofproto.OFP_NO_BUFFER:
    data = msg.data

out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                          in_port=in_port, actions=actions, data=data)
datapath.send_msg(out)
```

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1988 και τα άρθρα 2,4,6 παρ. 3 του Ν.1256/1982, η παρούας εργασία αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής.

_____