ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ασύρματες τεχνολογίες IEEE 802.11ah σε συστήματα “Διαδικτύου των Αντικειμένων” (Internet of things)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΤΖΗΖΗΣΗ ΝΙΚΟΛΑΟΥ

(ΑΕΜ: 941)

Επιβλέπων : **Δημήτριος Ι. Βέργαδος**

Επίκουρος Καθηγητής

Καστοριά Νοέμβριος - 2022

Ασύρματες τεχνολογίες IEEE 802.11ah σε συστήματα “Διαδικτύου των Αντικειμένων” (Internet of things)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ασύρματες τεχνολογίες IEEE 802.11ah σε συστήματα “Διαδικτύου των Αντικειμένων” (Internet of things)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΤΖΗΖΗΣΗ ΝΙΚΟΛΑΟΥ

(ΑΕΜ: 941)

Επιβλέπων : **Δημήτριος Ι. Βέργαδος**

Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης

.....	.....	.....
Ον/μο Μέλους	Ον/μο Μέλους	Ον/μο Μέλους
Ιδίοτητα Μέλους	Ιδίοτητα Μέλους	Ιδίοτητα Μέλους

Καστοριά Νοέμβριος - 2022

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής διπλωματικής μου εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες σε όλους όσους συνέβαλλαν στην εκπόνησή της.

Ευχαριστώ θερμά τον επιβλέπων καθηγητή μου, κύριο Δημήτριο Βέργαδο, για την εμπιστοσύνη που μου έδειξε εξ’ αρχής, αναθέτοντάς μου το συγκεκριμένο θέμα, την επιστημονική του καθοδήγηση, τις υποδείξεις του, την επιμονή του, το αμείωτο ενδιαφέρον του, τη συμπαράστασή του, τη συνεχή του υποστήριξη και το αμείωτο ενδιαφέρον που έδειξε από την αρχή μέχρι το τέλος.

Τέλος, θα ήθελα εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου για όλη τη στήριξη, τη συμπαράσταση και την κατανόησή τους, καθ’ όλη τη διάρκεια των σπουδών μου.

Περίληψη

Η εποχή του Internet of Things (IoT) χαρακτηρίζεται από εκατομμύρια συσκευές συνδεδεμένες στο δίκτυο. Υπάρχουν δύο κατηγορίες τεχνολογιών επικοινωνίας που χρησιμοποιούνται επί του παρόντος για εφαρμογές M2M. Το πρώτο είναι το ασύρματο δίκτυο αισθητήρων (WSN) που χρησιμοποιείται για τη διασύνδεση πολλών κόμβων αισθητήρων κατανεμημένων σε μια συγκεκριμένη περιοχή. Το δεύτερο είναι ένα κανονικό δίκτυο κινητής τηλεφωνίας που χρησιμοποιείται για απομονωμένους κόμβους για να επιτρέψει στην πύλη του WSN να φτάσει στο διαδίκτυο.

Πολλές τεχνολογίες επικοινωνίας, όπως Zigbee, Bluetooth, LoWPAN και 802.15.4, έχουν χρησιμοποιηθεί για τη μετάδοση δεδομένων σε ένα σύστημα M2M. Η ομάδα εργασίας IEEE 802.11ah (TGah) δημιουργήθηκε το 2010. Κυκλοφόρησε μια νέα τυποποίηση του WiFi, που ονομάζεται Wi-Fi Halow, για να υποστηρίξει την ασύρματη επικοινωνία M2M για να καλύψει το χάσμα μεταξύ του υπάρχοντος δικτύου κινητής τηλεφωνίας 3ης γενιάς (3GPP) και Ασύρματο δίκτυο αισθητήρων. Η τεχνολογία IEEE 802.11ah χρησιμοποιεί μη αδειοδοτημένο πρότυπο παγκόσμιου ασύρματου τοπικού δικτύου (WLAN) κάτω από 1 GHz για μελλοντική επικοινωνία M2M για την υποστήριξη μιας ευρείας σειράς σεναρίων. Υποστηρίζει μεγάλο αριθμό συσκευών (έως 8.191 συσκευές) για σύνδεση σε ένα σημείο πρόσβασης (AP). Άλλα πλεονεκτήματα του IEEE 802.11ah περιλαμβάνουν την αποδοτική κατανάλωση ενέργειας, την κάλυψη μεγάλης εμβέλειας, τους ρυθμούς δεδομένων έως και 100 kbps και την υψηλή απόδοση .

Σκοπός της παρούσης εργασίας είναι να εξεταστούν τα χαρακτηριστικά και ο τρόπος λειτουργίας των πρωτοκόλλων που χρησιμοποιούνται στο δίκτυο των πραγμάτων- internet of things. Η εργασία εστιάζει στην λειτουργία του πρωτοκόλλου 802.11ah. Το 802.11ah είναι ένα πρωτόκολλο επιπέδου ασύρματης επικοινωνίας PHY και MAC. Στο επίπεδο MAC, το 802.11ah εισάγει μηχανισμούς όπως η οργάνωση του συστήματος που θα εξηγηθούν περαιτέρω στην επόμενη

ενότητα. Σε σχέση με την ανάλυση προσομοίωσης του Σχεδίου Διαμόρφωσης και Κωδικοποίησης (MCS) σε διάφορα περιβάλλοντα, μπορεί να υποτεθεί ότι το MCS 08 με εύρος ζώνης 2 MHz είναι κατάλληλο για τοποθεσία υψηλής πυκνότητας σε αστικές περιοχές. Εν τω μεταξύ, MCS 0 με εύρος ζώνης 1 MHz κατάλληλο για τοποθεσία χαμηλής πυκνότητας (αγροτική περιοχή).

Αρχικά στο θεωρητικό μέρος της εργασίας θα γίνει εκτενής ανασκόπηση της βιβλιογραφίας σχετικά με το διαδίκτυο των πραγμάτων και τα βασικά πρωτόκολλα επικοινωνίας που επιτρέπουν την επικοινωνία των συσκευών IOT. Στην συνέχεια στο εμπειρικό μέρος της εργασίας παρουσιάζονται τα αποτελέσματα προσομοίωσης του πρωτοκόλλου 802.11ah με σκοπό να γίνει κατανοητός ο τρόπος λειτουργίας του.

Λέξεις Κλειδιά: *πρωτόκολλο* 802.11ah*, Internet of Things, ασύρματα δίκτυα*

Abstract

The era of the Internet of Things (IoT) is characterized by millions of devices connected to the network. There are two classes of communication technologies currently used for M2M applications. The first is wireless sensor network (WSN) which is used to interconnect many sensor nodes distributed in a certain area. The second is a regular cellular network used for isolated nodes to allow the gateway of the WSN to reach the internet. Many communication technologies, such as Zigbee, Bluetooth, LoWPAN, and 802.15.4, have been used to transmit data in an M2M system. The IEEE 802.11ah (TGah) working group was formed in 2010. A new WiFi standard, called Wi-Fi Halow, was launched to support M2M wireless communication to bridge the gap between the existing 3rd generation mobile network (3GPP) and Wireless Sensor Network. IEEE 802.11ah technology uses an unlicensed worldwide wireless local area network (WLAN) standard below 1 GHz for future M2M communication to support a wide range of scenarios. It supports a large number of devices (up to 8,191 devices) to connect to one access point (AP).

Other advantages of IEEE 802.11ah include energy efficiency, long-range coverage, data rates up to 100 kbps, and high throughput. The purpose of this paper is to examine the characteristics and the mode of operation of the protocols used in the internet of things network.

This Thesis focuses on the operation of the 802.11ah protocol. 802.11ah is a wireless PHY and MAC layer protocol. At the MAC layer, 802.11ah introduces mechanisms such as system organization that will be further explained in the next section. Regarding the simulation analysis of Modulation and Coding Scheme (MCS) in various environments, it can be assumed that MCS 08 with 2 MHz bandwidth is suitable for high density location in urban areas. Meanwhile, MCS 0 with 1 MHz bandwidth suitable for low density location (rural area).

Initially, in the theoretical part of the paper, an extensive literature review will be made regarding the Internet of Things and the basic communication protocols

that allow the communication of IOT devices. Then, in the empirical part of the work, the simulation results of the 802.11ah protocol are presented in order to understand its mode of operation.

Key Words: *802.11ah protocol, Internet of Things, wireless networks*

Πίνακας Περιεχομένων

Λίστα Εικόνων

Εισαγωγή

Στην παρούσα ενότητα δίνεται η εισαγωγή της εργασίας στην οποία γίνεται αναφορά στο θέμα της εργασίας, το σκοπό και τους τρόπους ανάπτυξης του θέματος.

Η ταχεία ανάπτυξη της επικοινωνίας Internet-of-Things (IoT) και Machine-to-Machine (M2M) καθιστά απαραίτητο τον σχεδιασμό συστημάτων επικοινωνίας που λειτουργούν σε διαφορετικό ασύρματο φάσμα ως εναλλακτική λύση σε συστήματα ασύρματης πρόσβασης με υψηλή συμφόρηση. Επιπλέον, η ανάπτυξη ασύρματων συσκευών έξυπνων μετρητών αυξάνεται και αναμένεται ότι τέτοιες συσκευές θα κατακλύσουν την αγορά στο εγγύς μέλλον ανταγωνιζόμενοι για το ίδιο ασύρματο φάσμα.

Η ομάδα εργασίας τυποποίησης IEEE 802.11ah στοχεύει σε ένα παγκόσμιο πρότυπο ασύρματου LAN (WLAN) που λειτουργεί με συχνότητες φορέα κάτω από 1 GHz στη ζώνη ISM (Industrial, Scientific, and Medical) και θα βοηθήσει στην παροχή εγγυημένων συσκευών με δυνατότητα Wi-Fi πρόσβαση για βραχυπρόθεσμες εκπομπές σε λιγότερο συμφορημένες ζώνες συχνοτήτων. Εκτός από την εκμετάλλευση του υποχρησιμοποιημένου φάσματος Sub 1 GHz, το βελτιωμένο εύρος κάλυψης επιτρέπει την εμφάνιση νέων εφαρμογών όπως δίκτυα αισθητήρων ευρείας περιοχής, συστήματα backhaul αισθητήρων και πιθανές λειτουργίες εκφόρτωσης Wi-Fi.

Είναι ευρέως αποδεκτό ότι το κυρίαρχο εσωτερικό (ασύρματο) δίκτυο βασίζεται στο πρότυπο IEEE 802.11, συμπεριλαμβανομένων σημείων πρόσβασης εσωτερικών χώρων (APs) και σταθμών (STAs) που βασίζονται στο IEEE 802.11, όπως φορητούς υπολογιστές, εκτυπωτές, PDA που χρησιμοποιούν διεπαφές ασύρματου LAN (WLAN). Λειτουργώντας στα 2,4 GHz/5 GHz, αυτό το πρότυπο και οι τροποποιήσεις του, όπως το 802.11a/b/g/n, επιτρέπουν την ασύρματη πρόσβαση χωρίς άδεια στη ζώνη ISM (Βιομηχανική, Επιστημονική και Ιατρική) και έτσι διευκολύνουν τη δημιουργία του χρήστη του Διαδικτύου το δικό τους ασύρματο δίκτυο πρόσβασης. Αυτή η εκτεταμένη ανάπτυξη ασύρματων δικτύων που βασίζονται στο IEEE 802.11 σε εσωτερικά περιβάλλοντα και επίσης για εγκαταστάσεις αστικών πόλεων έχει οδηγήσει σε δραματικά προβλήματα παρεμβολών και μείωση της απόδοσης του δικτύου όπου οι

χρήστες ασύρματου Διαδικτύου υποφέρουν από χαμηλή ταχύτητα ή ακόμα και διακοπή δικτύου. Εκτός από τα εσωτερικά περιβάλλοντα, οι λεγόμενες διεπαφές Wi-Fi μπορούν να βρεθούν σε διάφορες φορητές συσκευές, όπως tablet PC και έξυπνα τηλέφωνα, αυξάνοντας έτσι τον ανταγωνισμό για ασύρματη πρόσβαση χωρίς άδεια στη ζώνη ISM. Η εφαρμογή έξυπνου δικτύου, το Διαδίκτυο των πραγμάτων (IoT) και η επικοινωνία από μηχανή με μηχανή (M2M) θα οδηγήσουν περαιτέρω σε κορεσμένο φάσμα όταν χρησιμοποιούνται οι ίδιες συχνότητες, συμπεριλαμβανομένων των συστημάτων επικοινωνίας που βασίζονται σε Wi-Fi και IEEE 802.15.4.

Εκτός από την επιτυχή ανάπτυξη συσκευών IEEE 802.11 στη ζώνη συχνοτήτων 2,4 GHz/5 GHz, ο σχεδιασμός δικτύων RFID και αισθητήρων που λειτουργούν σε χαμηλότερες συχνότητες όπως το ISM 900 MHz κατέλαβε θέση στην ανάπτυξη ασύρματων συστημάτων. Με την ταχεία ζήτηση για πρόσβαση χωρίς άδεια, πανταχού παρούσα σε ζώνες συχνοτήτων με λιγότερες παρεμβολές, οι ζώνες ISM των 900 MHz έχουν προκαλέσει νέα έλξη όχι μόνο στον τομέα της έρευνας αλλά και στην τυποποίηση.

Ο στόχος είναι να οριστεί ένα παγκόσμιο πρότυπο WLAN που να λειτουργεί σε συχνότητες ISM κάτω από 1 GHz. Για παράδειγμα, στις Ηνωμένες Πολιτείες, την Ευρώπη και την Ιαπωνία, τέτοιες συχνότητες είναι διαθέσιμες, ωστόσο δεν υπάρχει ακόμη ένα πρότυπο που να χρησιμοποιεί τέτοιες συχνότητες. Η νέα ομάδα εργασιών (TG) στον οργανισμό τυποποίησης IEEE 802 στοχεύει στη δημιουργία ενός προτύπου WLAN για PHY και MAC που λειτουργεί σε συχνότητες κάτω του 1 GHz. Το λεγόμενο σύστημα WLAN Sub 1GHz βρίσκεται υπό τρέχουσα τυποποίηση στην ομάδα IEEE 802.11ah.

1. Διαδίκτυο των Πραγμάτων και Επικοινωνία

1.1 Internet of Things

Η ιδέα του IOT(Internet of Things) επινοήθηκε από ένα μέλος της κοινότητας ανάπτυξης Ραδιοφωνικής Αναγνώρισης (RFID) το 1999 και πρόσφατα έγινε πιο σχετική με τον πρακτικό κόσμο, κυρίως λόγω της ανάπτυξης των κινητών συσκευών, της ενσωματωμένης και πανταχού παρούσας επικοινωνίας, του υπολογιστικού νέφους και των δεδομένων. αναλυτικά.[12] Ο κοινός ορισμός του Διαδικτύου των πραγμάτων ορίζεται ως: Το Διαδίκτυο των πραγμάτων (IOT) είναι ένα δίκτυο φυσικών αντικειμένων. Το διαδίκτυο δεν είναι μόνο ένα δίκτυο υπολογιστών, αλλά έχει εξελιχθεί σε ένα δίκτυο συσκευών όλων των τύπων και μεγεθών, οχημάτων, έξυπνων τηλεφώνων, οικιακών συσκευών, παιχνιδιών, φωτογραφικών μηχανών, ιατρικών οργάνων και βιομηχανικών συστημάτων, ζώων, ανθρώπων, κτιρίων, όλα συνδεδεμένο, όλες οι πληροφορίες επικοινωνίας και κοινής χρήσης βασίζονται σε καθορισμένα πρωτόκολλα προκειμένου να επιτευχθούν έξυπνες αναδιοργανώσεις, εντοπισμός θέσης, εντοπισμός, ασφαλής και έλεγχος και ακόμη και προσωπική online παρακολούθηση σε πραγματικό χρόνο, διαδικτυακή αναβάθμιση, έλεγχος και διαχείριση διαδικασιών.

Ορίζουμε το IOT σε τρεις κατηγορίες ως εξής: Το Διαδίκτυο των πραγμάτων είναι ένα Διαδίκτυο τριών πραγμάτων:

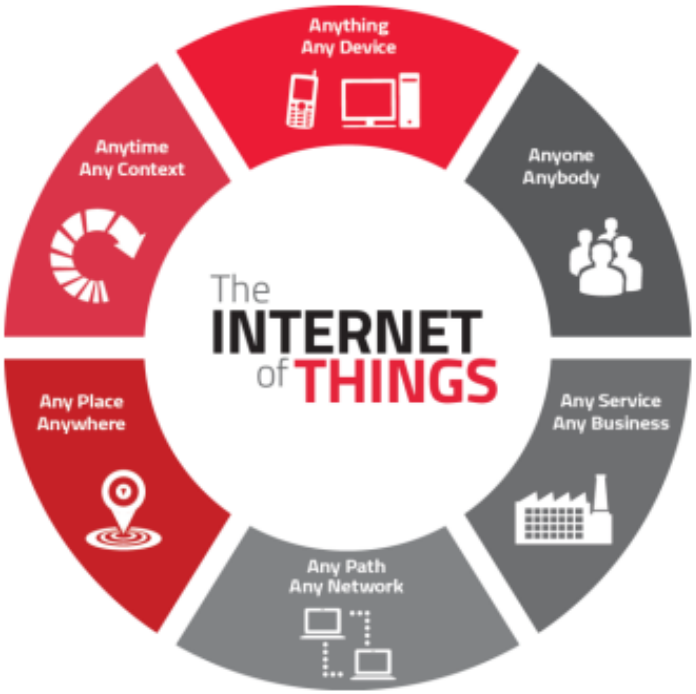
(1). Άνθρωποι σε ανθρώπους

(2) Άνθρωποι με μηχανή /πράγματα

(3) Πράγματα /μηχανή προς πράγματα /μηχανή, Αλληλεπίδραση μέσω Διαδικτύου

Το Διαδίκτυο των Πραγμάτων (IoT) είναι μια έννοια και ένα παράδειγμα που εξετάζει τη διάχυτη παρουσία στο περιβάλλον μιας ποικιλίας πραγμάτων/αντικειμένων που μέσω ασύρματων και ενσύρματων συνδέσεων και μοναδικών σχημάτων διευθύνσεων μπορούν να αλληλεπιδρούν μεταξύ τους και να συνεργάζονται με άλλα πράγματα/αντικείμενα για τη δημιουργία νέων εφαρμογών/υπηρεσιών και την επίτευξη κοινών στόχων. Σε αυτό το πλαίσιο, οι προκλήσεις έρευνας και ανάπτυξης για

τη δημιουργία ενός έξυπνου κόσμου είναι τεράστιες. Ένας κόσμος όπου το πραγματικό, το ψηφιακό και το εικονικό συγκλίνουν για να δημιουργήσουν έξυπνα περιβάλλοντα που κάνουν την ενέργεια, τις μεταφορές, τις πόλεις και πολλούς άλλους τομείς πιο έξυπνους.



Εικόνα 1- Το διαδίκτυο των Πραγμάτων

Το Διαδίκτυο των Πραγμάτων αναφέρεται στη γενική ιδέα των πραγμάτων, ιδιαίτερα των καθημερινών αντικειμένων, που είναι αναγνώσιμα, αναγνωρίσιμα, εντοπίσιμα, διευθυνσιοδοτούμενα μέσω συσκευής ανίχνευσης πληροφοριών ή/και ελεγχόμενα μέσω του Διαδικτύου, ανεξάρτητα από τα μέσα επικοινωνίας (είτε μέσω RFID, ασύρματο LAN, δίκτυα ευρείας περιοχής ή άλλα μέσα). Τα καθημερινά αντικείμενα περιλαμβάνουν όχι μόνο τις ηλεκτρονικές συσκευές που συναντάμε ή τα προϊόντα ανώτερης τεχνολογικής ανάπτυξης, όπως οχήματα και εξοπλισμό, αλλά πράγματα που συνήθως δεν θεωρούμε καθόλου ηλεκτρονικά - όπως τρόφιμα, ρούχα, καρέκλες, ζώα, δέντρα, νερό κ.λπ. [1,2] Το Διαδίκτυο των Πραγμάτων είναι μια νέα επανάσταση του Διαδικτύου.

Τα αντικείμενα γίνονται αναγνωρίσιμα και αποκτούν νοημοσύνη παίρνοντας ή επιτρέποντας αποφάσεις που σχετίζονται με το πλαίσιο χάρη στο γεγονός ότι μπορούν

να επικοινωνήσουν πληροφορίες για τον εαυτό τους. Μπορούν να έχουν πρόσβαση σε πληροφορίες που έχουν συγκεντρωθεί από άλλα πράγματα ή μπορεί να αποτελούν στοιχεία σύνθετων υπηρεσιών. Αυτός ο μετασχηματισμός είναι ταυτόχρονος με την εμφάνιση δυνατοτήτων υπολογιστικού νέφους και τη μετάβαση του Διαδικτύου προς το IPv6 με σχεδόν απεριόριστη χωρητικότητα διευθύνσεων. Ο στόχος του Διαδικτύου των Πραγμάτων είναι να επιτρέψει τη σύνδεση των πραγμάτων ανά πάσα στιγμή, οπουδήποτε, με οτιδήποτε και οποιονδήποτε, ιδανικά χρησιμοποιώντας οποιαδήποτε διαδρομή/δίκτυο και οποιαδήποτε υπηρεσία.

1.2 Η Σπουδαιότητα των πρωτοκόλλων στο Διαδίκτυο των Πραγμάτων

Η τεχνολογία είναι θεμελιώδες μέρος της καθημερινότητάς μας και μας εμφανίζεται με άυλο τρόπο. Στην περίπτωση των ασύρματων επικοινωνιών, τα ασύρματα δίκτυα χρησιμοποιούνται 24 ώρες την ημέρα και 7 ημέρες την εβδομάδα γύρω μας χωρίς καν να γίνονται αντιληπτοί. Οι τεχνολογίες πληροφοριών και επικοινωνιών (ΤΠΕ) εξελίσσονται με γρήγορο τρόπο. Ως μέρος αυτής της εξέλιξης, τις τελευταίες 3 δεκαετίες το Διαδίκτυο αναπτύχθηκε από ένα απλό δίκτυο που σχηματίστηκε από εκατοντάδες χρήστες μέχρι το σημερινό μεγα-δίκτυο που είναι σε θέση να συνδέσει εκατοντάδες εκατομμύρια χρήστες ανάμεσά τους. Η ανθρώπινη περιέργεια εξελίσσεται κάθε μέρα και μας οδηγεί κάθε φορά σε νέα και πιο δύσκολα όρια, όπως είναι η ιδέα ότι το Διαδίκτυο δεν δημιουργήθηκε μόνο για να διασυνδέει τους ανθρώπους.

Η ιδέα του Διαδικτύου έχει βελτιωθεί με την προσθήκη της ιδέας ότι πρέπει να συνδέει πράγματα (αισθητήρες, μηχανές, έξυπνες συσκευές, αυτοκίνητα, πόλεις) με ανθρώπους ταυτόχρονα, οραματιζόμενος και φτιάχνοντας έτσι ένα νέο σενάριο που χρειάζεται να αναπτύξει νέα δικτυακή επικοινωνία πρότυπα ή τεχνολογίες για την εκπλήρωση των νέων απαιτήσεων. Επιπλέον, αυτές οι νέες τεχνολογίες πρέπει να είναι ικανές να αναπτύσσουν και να ικανοποιούν τις νέες ανάγκες του χρήστη, όπως η κάλυψη της κινητικότητας των χρηστών στις πόλεις, οι καλύτεροι ρυθμοί μεταφοράς δεδομένων, τα μεγαλύτερα εύρη κάλυψης και τα συστήματα χρόνου απόκρισης χαμηλού λανθάνοντος χρόνου.

Ως απάντηση στη βελτίωση του δικτύου Διαδικτύου που απαιτείται από τα νέα σενάρια, η έννοια της πανταχού παρουσίας έρχεται στο προσκήνιο και μπορεί να εξηγηθεί ως ένα διάχυτο χαρακτηριστικό δικτύωσης που συνίσταται στην ύπαρξη συνδεσιμότητας δικτύου παντού όπου κινούνται οι χρήστες. Τα πανταχού παρόντα δεδομένα δικτύωσης και οι πανταχού παρόντες υπολογιστές προσφέρουν τις πιο πρόσφατες εμπειρίες αιχμής, όπως παιχνίδια εικονικής πραγματικότητας, εργασία εξ αποστάσεως, επιχειρηματικές εφαρμογές, υποδομές, χειρουργικές επεμβάσεις εξ αποστάσεως και εφαρμογές τεχνητής νοημοσύνης που οραματίζονται μεταξύ άλλων άνθρωποι και εταιρείες. Επίσης, οι ευφυείς εφαρμογές cloud και edge περιλαμβάνονται επίσης ως μέρος των υπηρεσιών αυτής της βελτίωσης που προσφέρονται στους ανθρώπους για να βελτιώσουν τον τρόπο λειτουργίας τους και να ανταγωνίζονται για να προσφέρουν λύσεις απόκρισης σε πραγματικό χρόνο σε έναν ταχέως μεταβαλλόμενο κόσμο, που εξελίσσεται και αναδιαμορφώνει τις ανάγκες των ανθρώπων συνεχώς. αυτό, τόσο τα νέα σενάρια όσο και οι απαιτήσεις των χρηστών δημιουργούν την ανάγκη ανάπτυξης νέων προτύπων, τα οποία πρέπει να μπορούν να διασυνδέουν ανθρώπους και έξυπνα αντικείμενα (έξυπνα πράγματα) όπως υπολογιστές, συσκευές ηλεκτρονικής υγείας, αισθητήρες για περιβαλλοντικές μετρήσεις, οικιακό αυτοματισμό, αυτόνομο και ημιαυτόνομα οχήματα, μεταξύ άλλων.

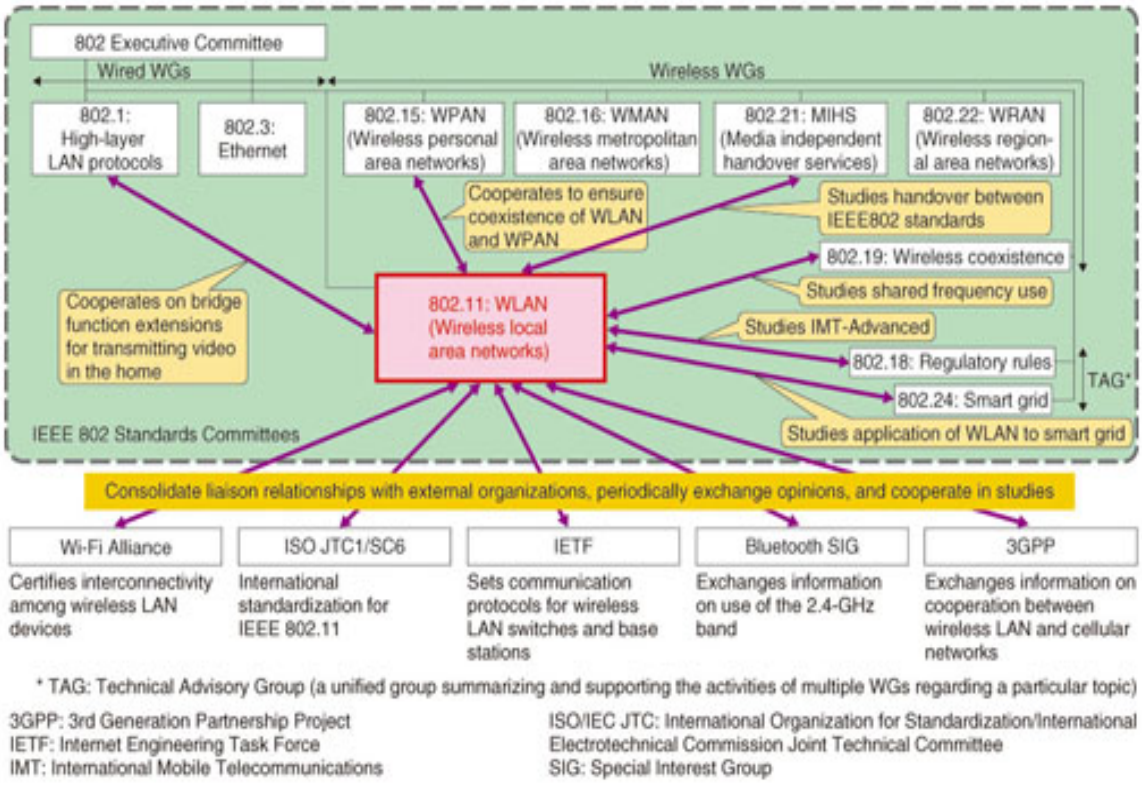
Δίνοντας νόημα στο Διαδίκτυο των Πραγμάτων (IoT), του οποίου το μοναδικό σύνθημα είναι «να υπάρχει σύνδεση δικτύου μεταξύ των χρηστών ανά πάσα στιγμή και σε οποιοδήποτε μέρος». Η έννοια της τηλεμετρίας χρειάζεται για να διευθετηθεί ως μια ροή δεδομένων που δημιουργούν οι συσκευές IoT και να ανοίξει τα νέα σενάρια για τιμές τηλεμετρίας σε διαφορετικές περιπτώσεις χρήσης όπου εμπλέκονται η επιτάχυνση, η υγρασία, η τοποθεσία, η πίεση, η θερμοκρασία και η ταχύτητα. Επιπλέον, το λεγόμενο «Industry 4.0» είναι ένα νέο πλεονέκτημα στη βιομηχανική διαδικασία που χρησιμοποιεί τεχνολογίες IoT και περιπτώσεις χρήσης που επωφελούνται από το IoT στις γραμμές παραγωγής του, στα ράφια λιανικής, στα ταμεία λιανικής, στον εξοπλισμό οδήγησης κ.λπ.

Αναμφίβολα, το Ινστιτούτο Ηλεκτρολόγων and Electronics Engineers (IEEE) 802.11 (εικόνα 2). η τεχνολογία ασύρματων δικτύων ή το λεγόμενο «Wi-Fi», που είναι παγκοσμίως διαδεδομένο και χρησιμοποιείται ευρέως στις μέρες μας, είναι μια

τεχνολογία που είναι ένας από τους εξέχοντες παράγοντες που θα μπορούσαν να χρησιμοποιηθούν για την αντιμετώπιση των νέων προκλήσεων και απαιτήσεων για τις έξυπνες εφαρμογές. Ως αποτέλεσμα των εκατομμυρίων χρηστών τεχνολογιών ασύρματης επικοινωνίας σε όλο τον κόσμο και του τεράστιου αριθμού συσκευών που είναι συνδεδεμένες στο IoT, υπολογίζεται ότι περίπου 50 δισεκατομμύρια συσκευές συνδέονται το 2020 [1].

Η Επιτροπή 802 του Ινστιτούτου Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (IEEE) (από εδώ και στο εξής "802 Επιτροπή") είναι ο οργανισμός που αναπτύσσει και διατηρεί πρότυπα για τα φυσικά και τα επίπεδα σύνδεσης δεδομένων των μέσων μετάδοσης για ενσύρματα και ασύρματα δίκτυα. Η Ομάδα Εργασίας 802.11 (WG) ιδρύθηκε σε αυτόν τον οργανισμό το 1990 για τη δημιουργία προτύπων για την εφαρμογή ασύρματου Ethernet. Το 802.11 WG πραγματοποιεί ανάπτυξη συνεργαζόμενος ευρέως με οργανώσεις τυποποίησης και βιομηχανίας εντός και εκτός των 802 υποεπιτροπών (Εικ. 1). Συγκεκριμένα, η Wi-Fi Alliance, η οποία πιστοποιεί τη διαλειτουργικότητα μεταξύ συσκευών ασύρματου τοπικού δικτύου (LAN) IEEE 802.11, έχει παίξει μεγάλο ρόλο στην εκρηκτική εξάπλωση των προϊόντων ασύρματου LAN [1].

Όπως φαίνεται στην εικόνα 2 από τότε που δημοσιεύτηκε το αρχικό πρότυπο 802.11, έχει επεκταθεί με πολλές τροποποιήσεις που αντιμετωπίζουν την αυξανόμενη ταχύτητα και τη λειτουργικότητα, οι οποίες περιλαμβάνουν μεγαλύτερες ταχύτητες μετάδοσης (802.11a/b/g/n/ac/ad), έλεγχος προτεραιότητας με βάση τον τύπο κυκλοφορίας (802.11e/aa/ae), τις βελτιώσεις ασφαλείας (802.11i/w), την εκτεταμένη λειτουργία ασύρματου LAN (802.11F/h/k/r/s/u/v/z) και την υποστήριξη για συχνότητες που χρησιμοποιούνται σε διαφορετικές χώρες (802.11d/j/y).



Εικόνα 2- ***Τοποθέτηση του 802.11 WG εντός των Επιτροπών Προτύπων IEEE 802.***

Πηγή: [28]

Επιπλέον, η επιτυχία των ασύρματων δικτύων δημιούργησε την ανάγκη για τη δημιουργία της Συμμαχίας Wi-Fi [2], η οποία είναι ένας οργανισμός που πιστοποιεί προϊόντα που υποδεικνύουν ότι πληρούν τα βιομηχανικά πρότυπα για ασφάλεια, διαλειτουργικότητα και εφαρμογές για συγκεκριμένα πρωτόκολλα. Το πρότυπο IEEE 802.11 εξελίσσεται αναπτύσσοντας την επόμενη γενιά ασύρματων δικτύων IEEE 802.11. Προφανώς, η δημοτικότητα των ασύρματων τοπικών δικτύων (WLAN) το εντοπίζει ως μία από τις πιο αξιόπιστες και παγκοσμίως διαδεδομένες τεχνολογίες με παρουσία σχεδόν παντού. Επιπλέον, το IEEE 802.11 εντόπισε την ανάγκη ανάπτυξης ενός νέου προτύπου επικεντρωμένου στις απαιτήσεις IoT, το οποίο είναι ικανό να υποστηρίζει χιλιάδες σταθμούς (STAs) εντός του ίδιου δικτύου, επιτρέποντας μεγάλες αποστάσεις εμβέλειας κάλυψης και παρέχοντας μηχανισμούς εξοικονόμησης ενέργειας. Ως το λεγόμενο Wi-Fi HaLow [3] από την πιστοποίηση Wi-Fi Alliance, αυτή η νέα τεχνολογία κατασκευάστηκε ειδικά για να καλύπτει αυτά τα χαρακτηριστικά IoT και το πρότυπο IEEE 802.11ah [4] κυκλοφόρησε στα τέλη του 2016.

2. Πρωτόκολλα IEEE 802.11

Στην παρούσα ενότητα θα μελετηθούν τα πρωτόκολλα IEEE 802.11

Στον παρακάτω πίνακα παρουσιάζονται συνοπτικά τα πρωτόκολλα της οικογένειας 802.11

Session		MQTT, SMQTT, CoRE, DDS, AMQP , XMPP, CoAP, …	Security	Management
Network	Encapsulation	6LowPAN, 6TiSCH, 6Lo, Thread, …	TCG, Oath 2.0, SMACK, SASL, ISASecure, ace, DTLS, Dice, …	IEEE 1905, IEEE 1451, …
	Routing	RPL, CORPL, CARP, …		
Datalink		WiFi, Bluetooth Low Energy, Z-Wave, ZigBee Smart, DECT/ULE, 3G/LTE, NFC, Weightless, HomePlug GP, 802.11ah, 802.15.4e, G.9959, WirelessHART, DASH7, ANT+, LTE-A, LoRaWAN, …		

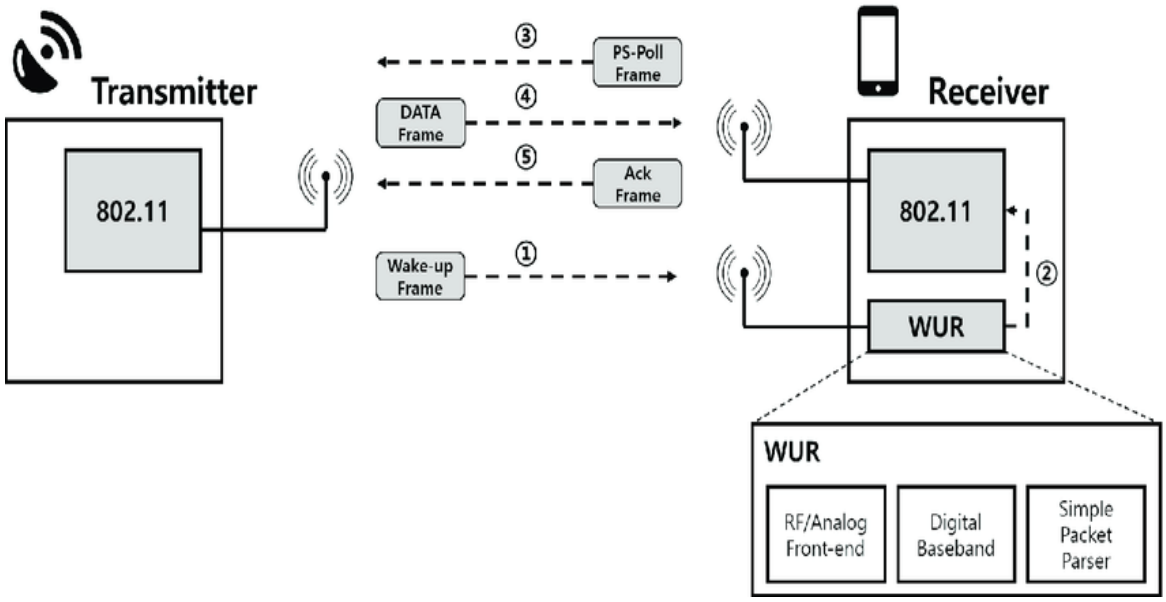
Εικόνα 3- Τα πρωτόκολλα δικτύων που χρησιμοποιούνται στο Internet of Things

IEEE 802.11 ασύρματα δίκτυα επόμενης γενιάς για επικοινωνίες IoT

Λαμβάνοντας υπόψη ότι τα πρότυπα επικοινωνίας ασύρματων δικτύων εξελίσσονται ανάλογα με τις ανάγκες του χρήστη και, λόγω της συνεχούς έρευνας και ανάπτυξης, έχουμε γίνει μάρτυρες τα τελευταία χρόνια πώς τα ασύρματα δίκτυα πληρούν το Internet for Humans (IoH) απαιτήσεις. Προκειμένου να υποστηριχθεί η προσθήκη του IoT στο IoH, θα απαιτηθεί η βελτίωση σε διακριτούς μηχανισμούς και βελτίωση των χαρακτηριστικών που θα χρειαστούν για την αντιμετώπιση των συγκεκριμένων περιπτώσεων χρήσης που δημιουργούνται από αυτά τα νέα σενάρια. Ακολουθώντας τα χαρακτηριστικά IoH και IoT και ενώνοντας την έννοια του Διαδικτύου των Πάντων (IoE) (βλ. Εικόνα 3), αναπτύχθηκε μια επόμενη γενιά ασύρματων δικτύων από το IEEE 802.11. Το κύριο πρότυπο που επικεντρώνεται στις απαιτήσεις εφαρμογών IoT είναι το IEEE 802.11ah.

Από την άλλη πλευρά, η επικείμενη τροποποίηση του IEEE 802.11ba περιλαμβάνει το IoT ως πεδίο εφαρμογής. Επίσης, προσέξτε ότι τα χαρακτηριστικά Φυσικού επιπέδου

(PHY) και Μεσαίου επιπέδου ελέγχου πρόσβασης (MAC) είναι αυτά που κάνουν διάκριση μεταξύ των διαφορετικών προτύπων προδιαγραφών IEEE 802.11.



Εικόνα 4- ***Αρχιτεκτονική του πρωτοκόλλου 802.11ab***

Πηγή: [29]

2.1 IEEE 802.11ah

Οι τροποποιήσεις IEEE 802.11 a/g, n ή ac δεν επικεντρώθηκαν στην ανάπτυξη οποιασδήποτε προδιαγραφής IoT. Επιπλέον, το νέο πρότυπο IEEE 802.11ah κυκλοφόρησε το 2016, υπογραμμίζοντας τις προσεγγίσεις IEEE 802.11 στο IoT. Αυτό το πρότυπο προορίζεται να παρέχει έναν τρόπο λειτουργίας χαμηλού κόστους, με μεγαλύτερη περιοχή κάλυψης και χιλιάδες σχετικούς σταθμούς ανά κυψέλη. Για να αξιολογήσουμε πώς αυτό το νέο πρότυπο IEEE 802.11ah προσθέτει αξία στις οικογένειες IEEE 802.11 όσον αφορά το εύρος και την απόδοση για νέες περιπτώσεις χρήσης, συγκρίνουμε την απόδοσή του με τις τρέχουσες τροποποιήσεις του IEEE 802.11.

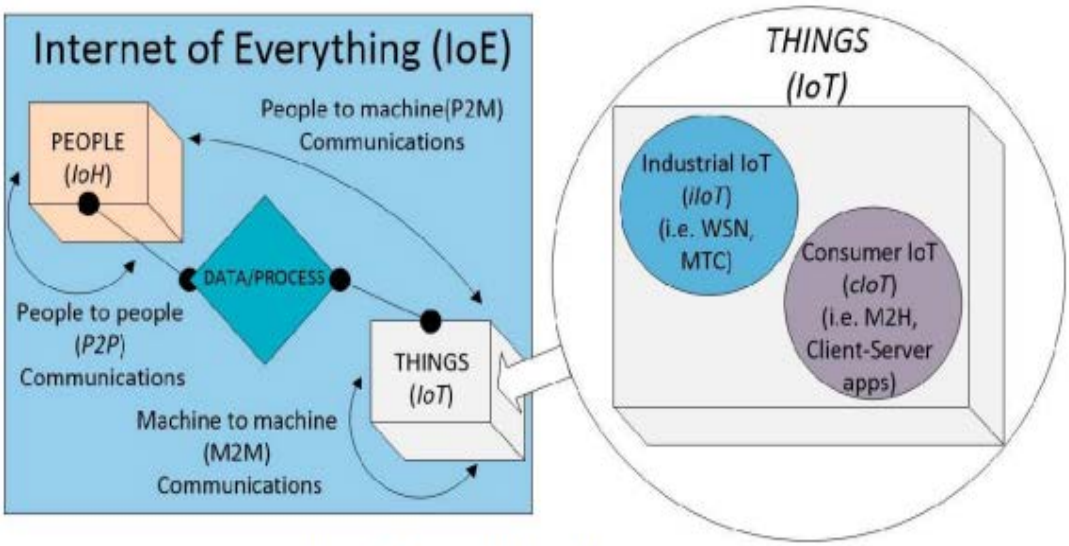


Figure 1. Internet of Everything concept.

Εικόνα 5- Διαδίκτυο των Πραγμάτων (τρόπος επικοινωνίας [12]

Το IEEE 802.11ah είναι μια ελαφριά (χαμηλής ενέργειας) έκδοση του αρχικού προτύπου ασύρματης πρόσβασης μέσου IEEE 802.11. Έχει σχεδιαστεί με λιγότερα έξοδα για να ανταποκρίνεται στις απαιτήσεις του IoT. Τα πρότυπα IEEE 802.11 (γνωστά και ως Wi-Fi) είναι τα πιο συχνά χρησιμοποιούμενα ασύρματα πρότυπα. Έχουν χρησιμοποιηθεί και υιοθετηθεί ευρέως για όλες τις ψηφιακές συσκευές, συμπεριλαμβανομένων φορητών υπολογιστών, κινητών, tablet και ψηφιακών τηλεοράσεων. Ωστόσο, τα αρχικά πρότυπα WiFi δεν είναι κατάλληλα για εφαρμογές IoT λόγω της επιβάρυνσης πλαισίου και της κατανάλωσης ενέργειας. Ως εκ τούτου, η ομάδα εργασίας του IEEE 802.11 ξεκίνησε την ομάδα εργασιών 802.11ah για την ανάπτυξη ενός προτύπου που υποστηρίζει επικοινωνία χαμηλής κατανάλωσης, φιλική προς την ενέργεια κατάλληλη για αισθητήρες και κινήσεις [Park15].

Τα βασικά χαρακτηριστικά του πρωτοκόλλου 802.11ah

Τα βασικά χαρακτηριστικά του επιπέδου MAC 802.11ah περιλαμβάνουν:

• Πλαίσιο συγχρονισμού: Ένας σταθμός δεν επιτρέπεται να εκπέμπει εκτός εάν έχει έγκυρες πληροφορίες μέσου που του επιτρέπουν να καταγράφει το μέσο και να διακόπτει την ανταλλαγή πακέτων από άλλους. Μπορεί να γνωρίζει τέτοιες πληροφορίες εάν λάβει σωστά το πακέτο πεδίου διάρκειας. Εάν δεν το λάβει σωστά,

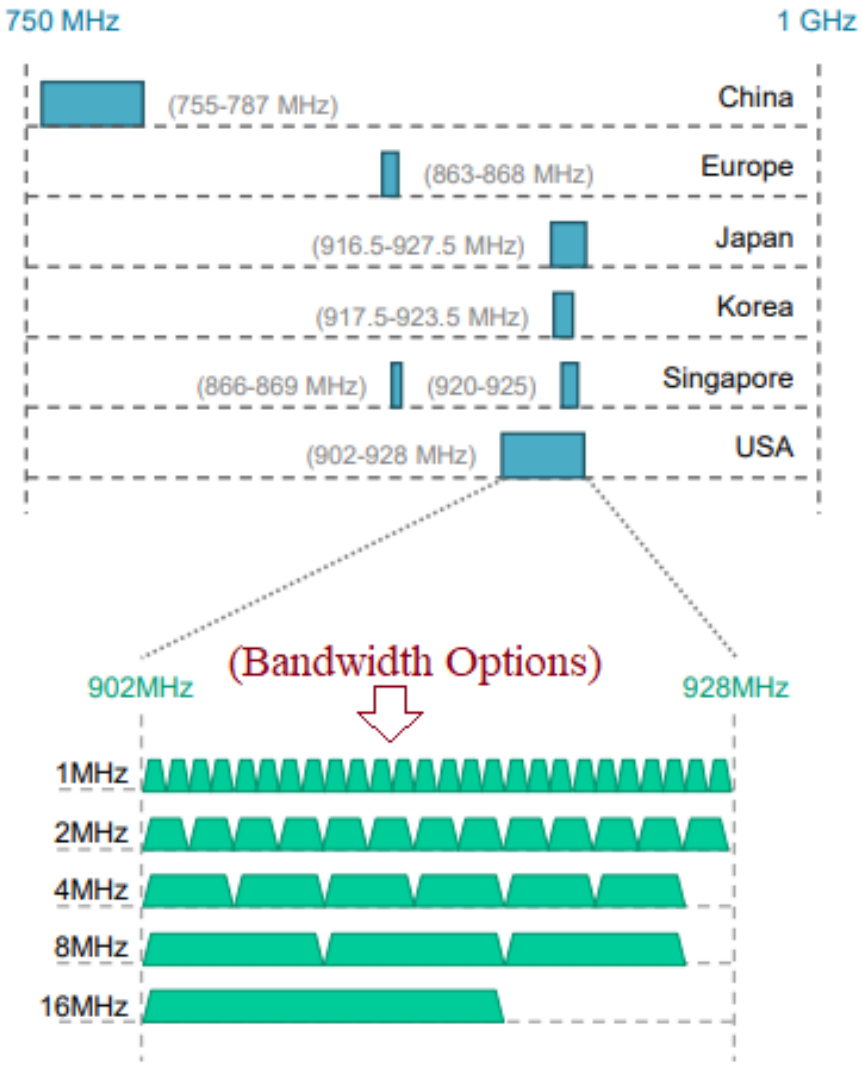
τότε θα πρέπει να περιμένει για μια διάρκεια που ονομάζεται Καθυστέρηση ανίχνευσης. Η Καθυστέρηση ανίχνευσης μπορεί να διαμορφωθεί από τα σημεία πρόσβασης στο 802.11ah και να ανακοινωθεί μεταδίδοντας ένα πλαίσιο συγχρονισμού στην αρχή της χρονοθυρίδας.

• Αποτελεσματική αμφίδρομη ανταλλαγή πακέτων: Αυτή η δυνατότητα επιτρέπει στη συσκευή αισθητήρα να εξοικονομεί περισσότερη ενέργεια επιτρέποντας την επικοινωνία ανερχόμενης και κατερχόμενης ζεύξης μεταξύ του σημείου πρόσβασης και του αισθητήρα και επιτρέποντάς της να τεθεί σε αδράνεια μόλις ολοκληρώσει την επικοινωνία.

• Σύντομο πλαίσιο Mac: Το κανονικό πλαίσιο IEEE 802.11 είναι περίπου 30 byte, το οποίο είναι πολύ μεγάλο για εφαρμογές IoT. Το IEEE 802.11ah μετριάζει αυτό το πρόβλημα ορίζοντας ένα σύντομο πλαίσιο MAC με περίπου 12 byte.

• Null Data Packet: Στο IEEE 802.11 τα πλαίσια ελέγχου, όπως τα πλαίσια Acknowledgment (ACK), είναι περίπου 14 byte και δεν έχουν δεδομένα, γεγονός που προσθέτει μεγάλο κόστος. Το IEEE 802.11ah μετριάζει αυτό το πρόβλημα αντικαθιστώντας το πλαίσιο ACK με ένα προοίμιο, ένα μικροσκοπικό σήμα.

• Αύξηση του χρόνου ύπνου: Το 802.11ah έχει σχεδιαστεί για αισθητήρες χαμηλής ισχύος και, ως εκ τούτου, επιτρέπει μεγάλη χρονική περίοδο ύπνου και σπάνια αφύπνιση για ανταλλαγή δεδομένων μόνο.



Εικόνα 6- ***Εύρος ζώνης και συχνότητα του πρωτοκόλλλου 802.11ah***

Πηγή: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-WiFi-HaLow.html

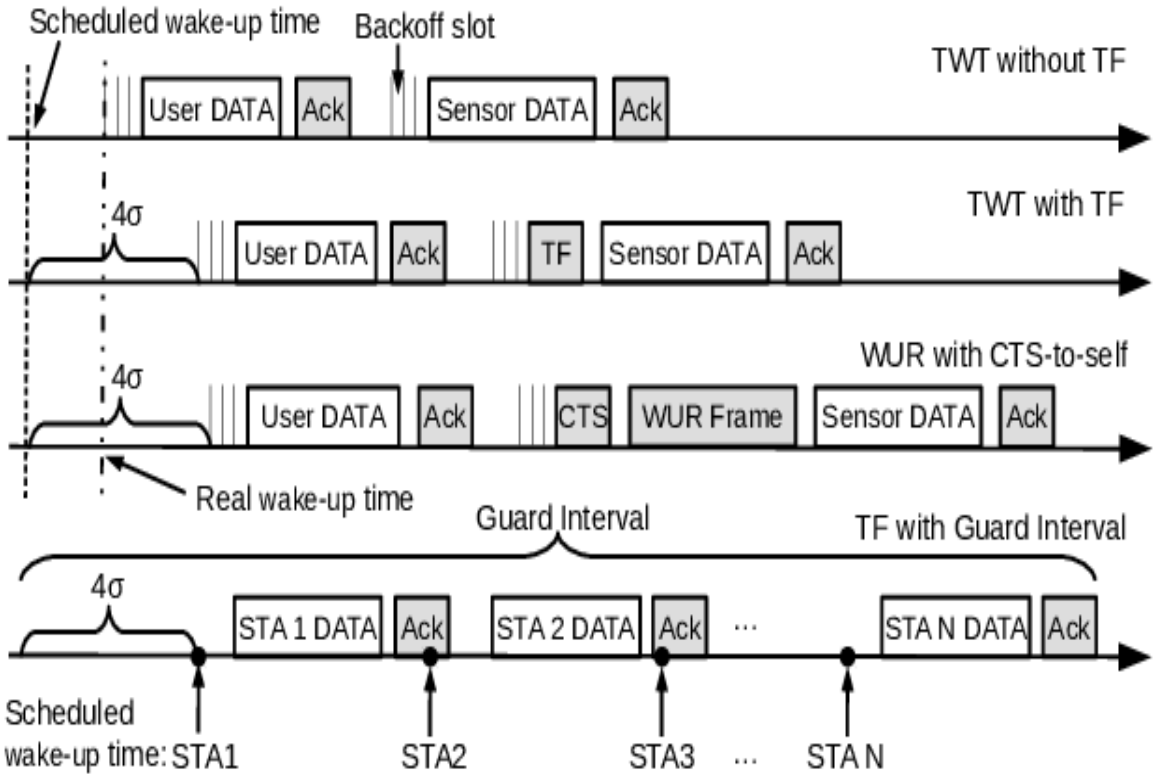
IEEE 802.11ba

Πιο πρόσφατα, η Ομάδα Εργασίας IEEE 802.11 (WG) ενεργοποίησε άλλες μελλοντικές προδιαγραφές για να συμπεριλάβει την περίπτωση χρήσης IoT. Από τον Ιούλιο του 2015, ξεκίνησε η δημιουργία μιας νέας Ομάδας Ενδιαφέροντος Θέματος (TIG) για τη λειτουργία Long-Range Low-Power (LRLP) για το IoT [5], η οποία είχε ως στόχο να φέρει ορισμένες από τις νέες δυνατότητες IEEE 802.11ah στο 2.4 Ζώνη GHz, ενώ διατηρείται η συμβατότητα με τις κύριες συσκευές IEEE 802.11 σε αυτήν τη ζώνη.

Τον Μάιο του 2016, η TIG συμφώνησε να επικεντρωθεί στο θέμα της χαμηλής κατανάλωσης ενέργειας (αφήνοντας κατά μέρος τη δυνατότητα μεγάλης εμβέλειας), δημιουργώντας μια Ομάδα Μελέτης (SG), τον LP-WUR (Low-Power Wake-Up Receiver) SG.

Επομένως, το LRLP TIG έχει καταργηθεί. Παρόλα αυτά, από τον Δεκέμβριο του 2016 και μετά από σημαντικό ενδιαφέρον για αυτό το θέμα από ερευνητές και επιστήμονες, το IEEE 802.11ba WG εργάζεται σε προσχέδια εγγράφων της τροποποίησης, ορίζοντας χαρακτηριστικά σε επίπεδα PHY και MAC για να έχει μια επιτυχημένη λειτουργία ενός Wakeup Radio (WuR) σύστημα [6].

Τα κύρια χαρακτηριστικά των συσκευών IoT περιλαμβάνουν αποκρίσεις χαμηλής ισχύος και χαμηλής καθυστέρησης. Προκειμένου να υποστηριχθεί το χαρακτηριστικό χαμηλής κατανάλωσης, η Ομάδα Εργασίας IEEE 802.11ba (TG) περιλάμβανε μια πρωταρχική δυνατότητα όπως είναι η δυνατότητα WuR σε αυτό το μελλοντικό πρότυπο [6].



Εικόνα 7- Πρωτόκολλο 802.11 ab- χαρακτηριστικά

Έτσι, προκειμένου να βελτιωθεί η εξοικονόμηση ενέργειας, το WuR λειτουργεί συνήθως ως πρόσθετη διεπαφή ραδιοφώνου, όπου η κατανάλωση ενέργειας μειώνεται ασυνήθιστα, και αυτό επιτρέπει στο AP να μεταδίδει δεδομένα ελέγχου στους STA, εν τω μεταξύ, ο κύριος ασύρματος είναι σε κατάσταση νάρκης. Συνοψίζοντας, όσο λιγότερος χρόνος αφυπνίζονται οι συσκευές για να λάβουν πληροφορίες, τόσο περισσότερη εξοικονόμηση ενέργειας. Σημειώστε ότι η TG στοχεύει να κυκλοφορήσει αυτό το πρότυπο το 2020. Κύρια ασύρματα δίκτυα IEEE 802.11.

Τα κύρια ασύρματα δίκτυα IEEE 802.11 κατέχουν πρωταρχική θέση στην καθημερινή μας ζωή. Με στόχο τη βελτίωση χαρακτηριστικών όπως η απόδοση του φάσματος και η απόδοση της περιοχής, και λαμβάνοντας υπόψη τον συνεχώς αυξανόμενο αριθμό χρηστών για εγκαταστάσεις σε εσωτερικούς και εξωτερικούς χώρους σε μειωμένες περιοχές στις πόλεις, καθώς και λαμβάνοντας υπόψη την παρουσία χιλιάδων σταθμών που λειτουργούν ως πηγή παρεμβολής μεταξύ τους. Ως συνέπεια των προαναφερθέντων στόχων, το IEEE 802.11 εντοπίζει ετερογενείς προκλήσεις δικτύου σε πυκνά σενάρια και τις αντιμετωπίζει με τα πρότυπα IEEE 802.11ac και IEEE 802.11ax.

2.2 IEEE 802.11ac

Η προδιαγραφή IEEE 802.11ac (Wi-Fi 5 από την Wi-Fi Alliance) παρέχει υψηλούς ρυθμούς έως 6,93 Gbps, χρησιμοποιεί πολλαπλή έξοδο πολλαπλής εισόδου (MIMO) ενισχυμένη με έως και 8 ροές, περιλαμβάνει 160 MHz και (80 + 80 MHz) εύρος ζώνης σήματος, διαμόρφωση 256 QAM και εισάγει το MIMO πολλαπλών χρηστών (MU-MIMO) [7]. Το MU-MIMO είναι μια τεχνική που επιτρέπει σε ένα AP να ανταλλάσσει δεδομένα με πολλαπλούς STAs ταυτόχρονα με στόχο να μειώσει τον χρόνο που κάθε STA πρέπει να περιμένει για μετάδοση, βελτιώνοντας έτσι την απόδοση.

Το IEEE 802.11 ac είναι ένα νέο πρότυπο ασύρματου δικτύου στην οικογένεια IEEE 802.11. Αυτό το Πρότυπο ονομάζεται επίσης ασύρματο LAN Gigabyte, το οποίο είναι η πιο πρόσφατη έκδοση των προτύπων Wi-Fi. Το IEEE 802.11ac έχει προταθεί για τη βελτίωση της απόδοσης του προτύπου IEEE 802.11n πέρα από τη δυνατότητα Gigabit

ανά δευτερόλεπτο και πελάτη που απαιτούνται ολοένα και περισσότερο από τους χρήστες κινητών τηλεφώνων. Ακόμη και όταν το δίκτυο δεν έχει φορτωθεί πλήρως, ο χρήστης μπορεί να δει τη λήψη των αρχείων και το συγχρονισμό του gmail με χαμηλή ταχύτητα GB.

Ο σκοπός της τροποποίησης 802.11ac είναι να βελτιώσει την εμπειρία χρήστη Wi-Fi παρέχοντας σημαντικά υψηλότερη απόδοση για υπάρχουσες περιοχές εφαρμογών και να επιτρέψει σε νέα τμήματα της αγοράς να λειτουργούν κάτω από 6 GHz, συμπεριλαμβανομένης της διανομής πολλαπλών ροών δεδομένων.

Ζώνη συχνοτήτων 5 GHz

Το 802.11n λειτουργεί και στις δύο μπάντες RF 2,5 GHz και 5 GHz, ενώ το 802.11ac λειτουργεί μόνο σε ζώνες RF 5 GHz για τον περιορισμό της χρήσης σε αυτή τη ζώνη καθορίζεται κυρίως από τις απαιτήσεις ευρύτερου εύρους ζώνης καναλιών211ac80.

Διαμόρφωση

Το 802.11ac χρησιμοποιεί ορθογώνια πολυπλεξία διαίρεσης συχνότητας (OFDM) για τη διαμόρφωση των bits για μετάδοση. Ενώ η μέθοδος διαμόρφωσης είναι η ίδια με αυτή που χρησιμοποιείται στο 802.11n, το 802.11ac επιτρέπει προαιρετικά τη χρήση 256 QAM. Αυτό αυξάνει τον αριθμό των bit ανά δευτερεύοντα φορέα από 6 σε 8, με αποτέλεσμα έως και 33% αύξηση του ρυθμού δεδομένων PHY.

Συμβατότητα προς τα πίσω

Το 802.11ac παρέχει συμβατότητα προς τα πίσω με συσκευές 802.11a και 802.11n που λειτουργούν στη ζώνη των 5 GHz. Αυτό σημαίνει ότι το 802.11ac συνεργάζεται με συσκευές που υποστηρίζουν τεχνολογίες 802.11a και 802.11n και δομές πλαισίου 802.11ac μπορούν να φιλοξενήσουν μετάδοση με συσκευές 802.11a και 802.11n.

Περισσότερες χωρικές ροές MIMO - Υποστήριξη έως και οκτώ χωρικών ροών (τέσσερις στο 802.11n)

Ταχύτητα

Η ελάχιστη ταχύτητα των 802.11ac 1300 Mbps και η μέγιστη ταχύτητα των 802.11ac πρότυπο είναι περίπου 6,93 Gbps. Η μέγιστη ταχύτητα είναι αυξημένη σε σύγκριση με το πρότυπο 802.11n IEEE.

Οι παραπάνω διαφορές συνοψίζονται στον παρακάτω πίνακα.

	802.11ac	802.11n
Spectrum	5GHz	2.4GHz 5GHz
Bandwidth	20MHz 80MHz 40MHz 160MHz	20MHz 40MHz
Maximum Antennas	8X8 MIMO(MU-MIMO)	4X4 MIMO
Speed	433 Mbps 867 Mbps 1300 Mbps (80MHz)	150 Mbps 300 Mbps 450 Mbps (40MHz)
Max. Speed	6.93Gbps	600 Mbps

Εικόνα 8- Σύγκριση πρωτοκόλλων 802.11ac Και 802.11n

2.3 IEEE 802.11ax

Η προσεχής προδιαγραφή IEEE 802.11ax (Wi-Fi 6 από την Wi-Fi Alliance [8]) επικεντρώνεται στην αντιμετώπιση αυξημένης πυκνότητας συσκευών και απαιτήσεων υψηλής απόδοσης, προκλήσεις που υπάρχουν επί του παρόντος σε περιβάλλοντα Wi-Fi. Θεωρώντας ως σενάρια-στόχους του προαναφερθέντος προτύπου, περιβάλλοντα όπως το σπίτι (πυκνές πολυκατοικίες), οι επιχειρήσεις και τα κτίρια γραφείων, το πρότυπο IEEE 802.11ax εστιάζει στην παροχή υψηλών αποδόσεων σε εξαιρετικά πυκνά σενάρια [9].

Τα κύρια χαρακτηριστικά του Wi-Fi 6 είναι: η χρήση του MU-MIMO (downlink και uplink), τα αυξημένα εύρη ζώνης από 20 έως 160 MHz ικανότητα χρήσης καναλιού, το μέγεθος Fast Fourier Transform (FFT) από 256 έως 2048 για αύξηση της ευρωστίας,

78.125 Διάστημα υποφορέων kHz για αύξηση της εμβέλειας σε σχέση με την κάλυψη συστημάτων Ορθογώνιας Πολυπλεξίας Διαίρεσης Συχνότητας (OFDM), τη χρήση ενός έως οκτώ χωρικών ροών (SS), τη συμπερίληψη της 1024 Quadrature Amplitude Modulation για τη βελτίωση της απόδοσης και τη χρήση του Target Wake Λειτουργία up Time (TWT), η οποία βελτιώνει την εξοικονόμηση ενέργειας [10].

CHARACTERISTIC	802.11ac	802.11ax
BANDS	5 GHz	2.4 and 5 GHz
CHANNEL BANDWIDTH	20, 40, 80, 80+80 & 160 MHz	20, 40, 80, 80+80, & 160 MHz
FFT SIZES	64, 128, 256, 512	256, 512, 1024, 2048
SUBCARRIER SPACING	312.5 kHz	78.125 kHz
OFDM SYMBOL DURATION	3.2 us + 0.8/0.4 us CP	12.8 us + 0.8/1.6/3.2 us CP
HIGHEST MODULATION	256 QAM	1024 QAM
DATA RATES	433 Mbits/s (80 MHz, 1 SS) 6.933 Gbits/s (160 MHz, 8 SS)	600.4 Mbits/s (80 MHz, 1 SS) 9.6078 Gbits/s (160 MHz, 8 SS)

Εικόνα 9- ***Σύγκριση πρωτοκόλλων 802.11ac και 802.11 ax***

Πηγή: https://www.electronicproducts.com/nis-wlan-measurement-suite-tests-to-early-802-11ax-spec/

2.4 Ασύρματα δίκτυα μεγάλης εμβέλειας IEEE 802.11

Ένα βασικό χαρακτηριστικό στις ασύρματες επικοινωνίες IoT είναι το εύρος κάλυψης. Σε αντίθεση με τα παλαιού τύπου χαρακτηριστικά του IEEE 802.11, όπου το εύρος κάλυψης δεν θα μπορούσε να είναι μεγαλύτερο από μερικές εκατοντάδες μέτρα, οι νέες τεχνολογίες, όπως το IEEE 802.11ah, στοχεύουν να αυξήσουν σημαντικά αυτήν την κάλυψη. Επιπλέον, η ανάπτυξη του προτύπου IEEE 802.11af καλύπτει αυτό το χαρακτηριστικό βελτιώνοντας τη δυνατότητα εμβέλειας απόστασης ώστε να φτάνει τα

χιλιάδες μέτρα. Ένα από τα κύρια χαρακτηριστικά του προτύπου IEEE 802.11af είναι η λειτουργία σε TV White Spaces (TWS, 470-790 MHz στην Ευρώπη), το οποίο κυκλοφόρησε το 2014. Η χρήση αυτής της συνθήκης συχνότητας μετάδοσης επιτρέπει την αύξηση του εύρους κάλυψης ενός AP. Επιπλέον, αξίζει να αναφέρουμε ότι το PHY αυτού του προτύπου είναι παρόμοιο με το Wi-Fi 5. Επιπλέον, προστέθηκε μια πρόσθετη ζώνη προστασίας για την αποφυγή παρεμβολών στα γειτονικά κανάλια λόγω τηλεοπτικών μεταδόσεων [11]. Το IEEE 802.11af λειτουργεί σε αδειοδοτημένες ζώνες συχνοτήτων.

Οι κύριες ασύρματες τεχνολογίες για συσκευές IoT

Οι κύριες ασύρματες τεχνολογίες για συσκευές IoT χαμηλής κατανάλωσης διαδραματίζουν κρίσιμο ρόλο στις εφαρμογές IoT, οι οποίες περιλαμβάνουν τις επικοινωνίες Machine to Machine (M2M) ή Machine to Human (M2H). Ομοίως, οι προδιαγραφές IEEE 802.11 που είναι επιρρεπείς στο IoT, εμφανίζονται και άλλες νέες τεχνολογίες στην αρένα του IoT που περιλαμβάνουν συγκεκριμένα χαρακτηριστικά που είναι ελκυστικά για διαφορετικά σενάρια και εφαρμογές, όπως τεχνολογίες Low Power Wide Area Network (LPWAN) όπως LoRaWAN, Sigfox και Internet of Things στενής ζώνης (NB-IoT). Επιπλέον, τεχνολογίες όπως ZigBee, Bluetooth Low Energy (BLE) και 3rd Generation Partnership Project (3GPP) Machine Type Communications (MTC) αποτελούν μέρος των διαθέσιμων τεχνολογιών που είναι σε θέση να εκπληρώσουν τις απαιτήσεις εφαρμογών IoT.

2.5 LoRaWAN

Το στάνταρ LoRaWAN είναι κατασκευασμένο για εφαρμογές μεγάλης εμβέλειας, χαμηλής ισχύος και χαμηλής ταχύτητας δεδομένων [12]. Λειτουργεί σε ζώνη συχνοτήτων χωρίς άδεια από 867 έως 928 MHz, προσφέροντας ταχύτητες δεδομένων έως και 25 kbps. Όσον αφορά το εύρος κάλυψης, αυτή η τεχνολογία φτάνει περίπου τα

20 km, υποστηρίζοντας περισσότερες από 10.000 συσκευές [13]. Σημειώστε ότι το προαναφερθέν LoRaWAN είναι μια ιδιόκτητη τεχνολογία.

Το LoRa πήρε το όνομά του από το «Long Range», είναι μια πατενταρισμένη τεχνολογία ασύρματης επικοινωνίας δεδομένων. Η τεχνολογία αναπτύχθηκε από την Cycleo της Γκρενόμπλ της Γαλλίας και αποκτήθηκε από τη Semtech το 2012. Το βασικό χαρακτηριστικό που διαφοροποιεί το LoRa από άλλες διαθέσιμες τεχνολογίες ασύρματου WAN είναι η μετάδοση μεγάλης εμβέλειας με χαμηλή κατανάλωση ενέργειας. Ωστόσο, υποστηρίζει εφαρμογές χαμηλού ρυθμού bit με λιγότερη ζήτηση κινητικότητας και αξιοπιστίας. Αυτή η τεχνολογία έχει αναπτυχθεί για να υποστηρίξει έναν τύπο ασύρματου δικτύου τηλεπικοινωνιών ευρείας περιοχής, κοινώς γνωστά ως δίκτυα ευρείας περιοχής χαμηλής ισχύος (LPWAN), το οποίο στοχεύει στην επίτευξη κέρδους +20 dB σε σχέση με το παλιό κυψελοειδές σύστημα. Αν και ορισμένες παραδοσιακές λύσεις όπως Bluetooth, WiFi, ZigBee, WLAN, Z wave, δίκτυα κινητής τηλεφωνίας, GSM, LTE μπορούν να παρέχουν ασύρματη σύνδεση των συσκευών IoT σε ένα δίκτυο, αυτές οι λύσεις απαιτούν υψηλό κόστος, υψηλή κατανάλωση ενέργειας και υψηλή πολυπλοκότητα. Το European Telecommunications Standard Institute (ETSI), το Third Generation Partnership Project (3 GPPP), το Institute of Electrical and Electronic Engineers (IEEE) και το Internet Engineering Task Force (IETF) εργάζονται προς την κατεύθυνση των ανοιχτών προτύπων για τις τεχνολογίες LPWAN[30].

Σε αντίθεση με ορισμένες από τις άλλες τεχνολογίες LPWAN που χρησιμοποιούν ζώνη φάσματος χωρίς άδεια, το LoRa χρησιμοποιεί κρυπτογράφηση 128AES σε πολλαπλά επίπεδα για όλα τα δεδομένα από αισθητήρες έως διακομιστή εφαρμογών ως μέσο προστασίας δεδομένων και ιδιωτικότητας. Επιπλέον, δεν υπάρχει κανένα τεχνικό εμπόδιο για τη λειτουργία μιας ιδιωτικής λύσης LoRaWAN σε μια αδειοδοτημένη ζώνη, αν και η τεχνολογία εξελίσσεται με βάση τη ζώνη χωρίς άδεια χρήσης [30].

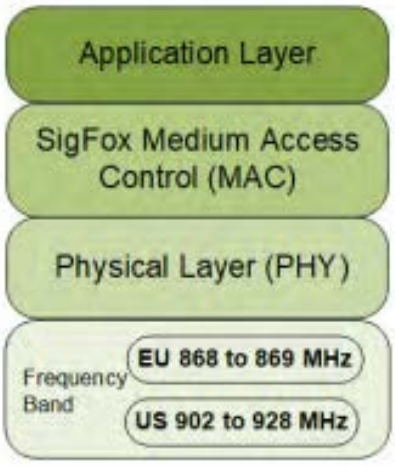
Feature	LoRaWAN	Narrow-Band	LTE Cat-1 2016 (Rel12)	LTE Cat-M 2018 (Rel13)	NB-LTE 2019(Rel13+)
Modulation	SS Chirp	UNB / GFSK/BPSK	OFDMA	OFDMA	OFDMA
Rx bandwidth	500 - 125 KHz	100 Hz	20 MHz	20 - 1.4 MHz	200 KHz
Data Rate	290bps - 50Kbps	100 bit/sec 12 / 8 bytes Max	10 Mbit/sec	200kbps – 1Mbps	~20K bit/sec
Max. # Msgs/day	Unlimited	UL: 140 msgs/day	Unlimited	Unlimited	Unlimited
Max Output Power	20 dBm	20 dBm	23 - 46 dBm	23/30 dBm	20 dBm
Link Budget	154 dB	151 dB	130 dB+	146 dB	150 dB
Batery lifetime - 2000mAh	105 months	90 months		18 months	
Power Efficiency	Very High	Very High	Low	Medium	Med high
Interference immunity	Very high	Low	Medium	Medium	Low
Coexistence	Yes	No	Yes	Yes	No
Security	Yes	No	Yes	Yes	Yes
Mobility / localization	Yes	Limited mobility, No loc	Mobility	Mobility	Limited Mobility No Loc

Εικόνα 10- ***Χαρακτηριστικά του LoRaWAN***

Πηγή: [30]

2.6 Το Sigfox

Το Sigfox μοιράζεται τα χαρακτηριστικά της τεχνολογίας του δικτύου LPWAN και ένα από τα κύρια χαρακτηριστικά της τεχνολογίας Sigfox είναι ότι περιλαμβάνει ρυθμούς δεδομένων στην περιοχή του 1 kbps ή χαμηλότερο, με εύρος κάλυψης έως 40 km και υποστήριξη περισσότερων από 1.000.00 συσκευές [14].



Εικόνα 11- Επίπεδα επικοινωνίας του πρωτοκόλλου Sigfox

Όπως και το LoRaWAN, το Sigfox είναι επίσης μια ιδιόκτητη τεχνολογία. Το Sigfox μαζί με το LoRaWAN έχουν υψηλότερη πολυπλοκότητα διασύνδεσης και περιορισμένο διαθέσιμο εύρος ζώνης σε αντίθεση με το IEEE 802.11ah, το οποίο μπορεί να προσφέρει τα υψηλότερα εύρη ζώνης και καλύτερη διασύνδεση.

Το επίπεδο ραδιοσυχνοτήτων περιλαμβάνει τα κανάλια επικοινωνίας SigFox, το Φυσικό επίπεδο περιλαμβάνει το σχήμα διαμόρφωσης, τον έλεγχο πρόσβασης μέσου, καθώς και την ανίχνευση σφαλμάτων και την πρόσβαση καναλιού και το επίπεδο εφαρμογής που ορίζονται από τις απαιτήσεις και τις προδιαγραφές του χρήστη. Ο μηχανισμός μετάδοσης χρησιμοποιεί πλεονασμό, επομένως, τα δεδομένα μεταδίδονται 3 φορές σε διαφορετικά κανάλια (για να εξασφαλιστεί η διαφοροποίηση συχνότητας) σε διαφορετικά χρονικά διαστήματα (για να εξασφαλιστεί η χρονική διαφοροποίηση)[31].

Αυτός ο πλεονασμός κάνει την επικοινωνία πιο εύρωστη σε παρεμβολές. Τα Gateways ακούν το φάσμα εύρους ζώνης συχνοτήτων (από 868,034 MHz έως 868,226 MHz για την περιοχή της Ευρώπης). Η αρχική διαμόρφωση του SigFox ήταν μια μονοκατευθυντική σύνδεση επικοινωνίας, όπου μόνο ο κόμβος προς την επικοινωνία Gateway επιτρεπόταν (uplink). Ο μηχανισμός επικοινωνίας Gateway to node (downlink) για να διασφαλιστεί ότι το σχήμα αμφίδρομης επικοινωνίας προστέθηκε αργότερα. Ο κόμβος μπορεί να στείλει έναν αριθμό 140 μηνυμάτων την ημέρα με μέγιστο ωφέλιμο

φορτίο 12 byte (περιορίζεται από κανονισμούς και επομένως από την πολιτική του δικτύου) και μπορεί να λαμβάνει από την εφαρμογή μέσω της πύλης 4 μηνύματα την ημέρα με μέγιστο ωφέλιμο φορτίο 8 byte . Όλοι αυτοί οι περιορισμοί έχουν κατά νου την ενεργειακή απόδοση και πρέπει να σέβονται μια «συμφωνία κυρίων», όπως την αναφέρει ο επίσημος ιστότοπος της SigFox [31].

Παρόμοια με μια συσκευή LoRaWAN κατηγορίας A, όταν η εφαρμογή θέλει να στείλει ένα μήνυμα στον κόμβο, θα πρέπει να περιμένει να εκχωρηθεί η χρονική θυρίδα λήψης. Μια τελική συσκευή SigFox μπορεί να εκπέμπει μόνο 36 δευτερόλεπτα ανά ώρα, δηλαδή περίπου. 6 δευτερόλεπτα ως χρόνος on air ανά μετάδοση. Κατά συνέπεια, αυτό θα έχει ως αποτέλεσμα μόνο 6 μηνύματα ανά ώρα, με ωφέλιμο φορτίο 12 byte, με 2,08 δευτερόλεπτα για το πακέτο.

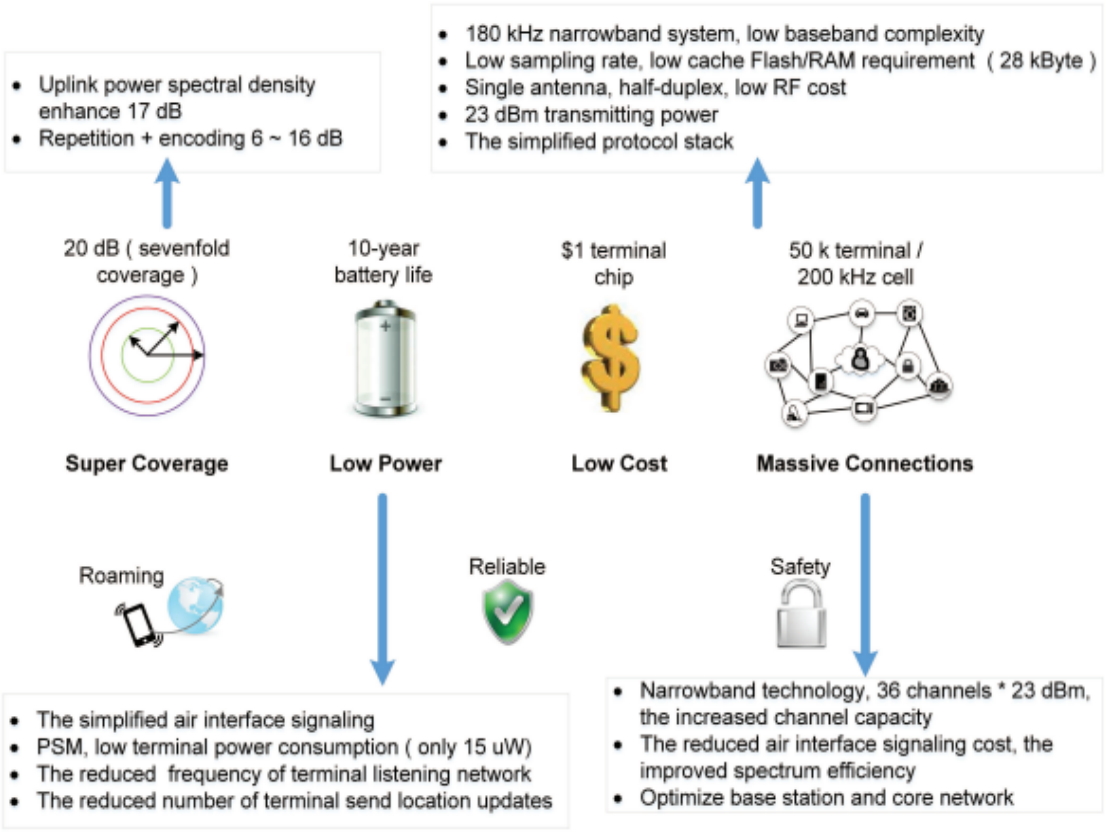
Αυτή η υποδοχή ενεργοποιείται 20 δευτερόλεπτα αφού ο κόμβος στείλει ένα μήνυμα. Το παράθυρο λήψης έχει μήκος μεταξύ 20,1 δευτερολέπτων και 44,5 δευτερολέπτων κατά το οποίο ο κόμβος ακούει ένα μήνυμα από την πύλη [21]. Η εφαρμογή στέλνει ένα μήνυμα στον κόμβο μέσω της πλησιέστερης πύλης. Η τυχαία πρόσβαση είναι ένα βασικό χαρακτηριστικό για την επίτευξη υψηλού QoS (Quality of Service).

2.7 NB-IoT

Το 3GPP εισήγαγε το NB-IoT, το οποίο επιτρέπει στους χειριστές να χρησιμοποιούν ένα ελάχιστο μέρος του διαθέσιμου φάσματος (Μακροπρόθεσμη εξέλιξη, LTE ή Παγκόσμιο Σύστημα Κινητών Επικοινωνιών, GSM, δίκτυα). Το NB-IoT παρουσιάζει μεγάλες περιοχές κάλυψης και αυξημένο αριθμό υποστηριζόμενων συσκευών [15].

Λειτουργεί σε αδειοδοτημένη ζώνη συχνοτήτων (700-900 MHz) και έχει εμβέλεια κάλυψης έως 15 km και ρυθμούς μετάδοσης δεδομένων έως 50 kbps. Αυτή η τεχνολογία υποστηρίζει επίσης περισσότερες από 100.000 συσκευές. Σημειώστε ότι το IEEE 802.11ah εκπέμπει κάτω από ζώνες συχνοτήτων χωρίς άδεια που μπορεί να θεωρηθεί ως πλεονέκτημα έναντι του NB-IoT. Επιπλέον, το κόστος εγκατάστασης και λειτουργίας θα είναι υψηλότερο από αυτό του IEEE 802.11ah. 1

Τα κύρια χαρακτηριστικά NB-IoT φαίνονται στην εικόνα 12 και παρουσιάζονται συνοπτικά στη συνέχεια.



Εικόνα 12- Βασικά χαρακτηριστικά και οφέλη από την χρήση του πρωτοκόλλου NB-IoT

Χαμηλή κατανάλωση ενέργειας

Χρησιμοποιώντας τη λειτουργία εξοικονόμησης ενέργειας (PSM) και την εκτεταμένη ασυνεχή λήψη (eDRX), μπορεί να επιτευχθεί μεγαλύτερος χρόνος αναμονής στο NB-IoT. Σε αυτό, η τεχνολογία PSM προστέθηκε πρόσφατα στο Rel-12, όπου στη λειτουργία εξοικονόμησης ενέργειας το τερματικό εξακολουθεί να είναι εγγεγραμμένο στο διαδίκτυο, αλλά δεν είναι δυνατή η πρόσβαση μέσω σήματος, προκειμένου να τεθεί το τερματικό σε βαθύ ύπνο για μεγαλύτερο χρονικό διάστημα για να επιτευχθεί η εξοικονόμηση ενέργειας. Από την άλλη πλευρά, το eDRX προστέθηκε πρόσφατα στο Rel-13, το οποίο επεκτείνει περαιτέρω τον κύκλο ύπνου του τερματικού σε κατάσταση αναμονής και μειώνει την περιττή εκκίνηση της κυψέλης λήψης. Σε σύγκριση με το PSM, το eDRX προωθεί σημαντικά την προσβασιμότητα downlink. Ο μηχανισμός εξοικονόμησης ενέργειας του PSM και του eDRX είναι όπως φαίνεται στην Εικ. [30].

Το NB-IoT απαιτεί η διάρκεια ζωής του τερματικού μιας μπαταρίας σταθερού όγκου να είναι 10 χρόνια για τυπική υπηρεσία χαμηλής συχνότητας χαμηλής συχνότητας. Σύμφωνα με προσομοιωμένα δεδομένα του TR45.820, για απώλεια σύζευξης 164 dB και χρήση τόσο PSM όσο και eDRX, η διάρκεια ζωής της μπαταρίας 5 Wh μπορεί να είναι 12,8 χρόνια εάν αποστέλλεται ένα μήνυμα 200 byte μία φορά την ημέρα από το τερματικό, όπως φαίνεται στον Πίνακα 2 [30]. (2) Βελτιωμένη κάλυψη και ευαισθησία χαμηλής καθυστέρησης Σύμφωνα με προσομοιωμένα δεδομένα του TR45.820, μπορεί να επιβεβαιωθεί ότι η ισχύς κάλυψης του NB-IoT μπορεί να φτάσει τα 164 dB σε λειτουργία ανεξάρτητης ανάπτυξης.

Η δοκιμή προσομοίωσης διεξήχθη τόσο για ανάπτυξη εντός ζώνης όσο και για ανάπτυξη ζώνης φρουράς. Προκειμένου να πραγματοποιηθεί η βελτίωση της κάλυψης, μηχανισμοί όπως η αναμετάδοση (200 φορές) και η διαμόρφωση χαμηλής συχνότητας υιοθετούνται από το NB-IoT. Προς το παρόν, η υποστήριξη NB-IoT για το 16QAM είναι ακόμα υπό συζήτηση.

Για απώλεια σύζευξης 164 dB, εάν παρέχεται αξιόπιστη μετάδοση δεδομένων, η καθυστέρηση αυξάνεται λόγω αναμετάδοσης μαζικών δεδομένων. Οι προσομοιώσεις για το TR45.820 δείχνουν την καθυστέρηση για το σενάριο ακανόνιστης υπηρεσίας αναφοράς και τις διαφορετικές απώλειες σύζευξης (συμπίεση κεφαλίδας ή όχι) με αξιοπιστία 99% [30]. Επί του παρόντος, η ανεκτή καθυστέρηση στο 3GPP IoT είναι 10 δευτερόλεπτα. Στην πραγματικότητα, μπορεί επίσης να υποστηριχθεί χαμηλότερη καθυστέρηση περίπου 6 δευτερολέπτων για μέγιστες απώλειες σύζευξης. Για περισσότερες λεπτομέρειες, ανατρέξτε στα αποτελέσματα προσομοίωσης του NB-IoT για το TR45.820.

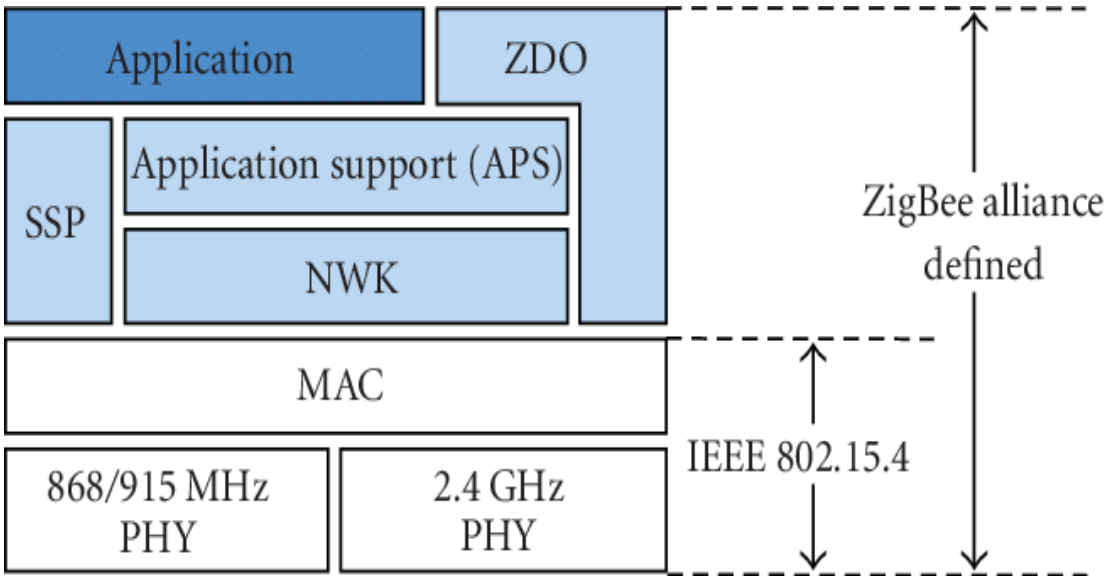
Τρόπος μετάδοσης

Η ανάπτυξη του NB-IoT βασίζεται στο LTE. Η τροποποίηση γίνεται κυρίως σε σχετικές τεχνολογίες LTE σύμφωνα με μοναδικά χαρακτηριστικά NB-IoT. Το εύρος ζώνης RF του φυσικού στρώματος NB-IoT είναι 200 kHz. Στην κατερχόμενη ζεύξη, το NB-IoT υιοθετεί το μόντεμ QPSK και την τεχνολογία OFDMA με απόσταση υποφερόντων 15 KHz [32]. Στην ανερχόμενη ζεύξη, υιοθετείται το μόντεμ BPSK ή QPSK και η τεχνολογία SC-FDMA, συμπεριλαμβανομένου ενός δευτερεύοντος φορέα και πολλαπλών

υποφερόντων. Μια τεχνολογία ενός δευτερεύοντος φορέα με απόσταση υποφερόντων 3,75 kHz και 15 kHz ισχύει για τερματικό IoT με εξαιρετικά χαμηλό ρυθμό και εξαιρετικά χαμηλή κατανάλωση ενέργειας.

2.8 ZigBee/IEEE 802.14.4e

Το πιο ελκυστικό χαρακτηριστικό του προτύπου ZigBee/IEEE 802.14.4e είναι το χαμηλό κόστος υλοποίησης, καθώς αυτή η τεχνολογία χρησιμοποιείται στα περισσότερα από τα ασύρματα δίκτυα αισθητήρων (WSN). Επιτρέπει μεγάλο αριθμό υποστηριζόμενων συσκευών (έως 65000 συσκευές) και προσφέρει ταχύτητες δεδομένων από 20 έως 250 Kbps [16], φτάνοντας το εύρος κάλυψης έως και 100 μέτρα και λειτουργώντας στη ζώνη συχνοτήτων 2,4 GHz. Σημειώστε ότι σε σύγκριση με το IEEE 802.11ah, οι χαμηλότεροι ρυθμοί δεδομένων υποστηρίζονται από την τεχνολογία ZigBee/802.15.4e.



Εικόνα 13- zigbee

2.9 Bluetooth χαμηλής ενέργειας

Το Bluetooth χαμηλής ενέργειας BLE αποτελείται από μια τροποποίηση του παλαιού τύπου Bluetooth με βελτιωμένα χαρακτηριστικά ώστε να αποτελεί μέρος του περιβάλλοντος επικοινωνιών IoT και να επικεντρώνεται στην κατανάλωση ενέργειας και

στην επικοινωνία χαμηλής ταχύτητας μικρής εμβέλειας [17]. Η επιτυχία αυτής της τεχνολογίας βασίζεται στο γεγονός ότι λειτουργεί στη ζώνη συχνοτήτων των 2,4 GHz, φτάνοντας ταχύτητες δεδομένων έως και 1 Mbps Επιπλέον, καλύπτει περιοχές μέγιστου μήκους 50 μέτρων και είναι σε θέση να υποστηρίξει απεριόριστο αριθμό συσκευών (ανάλογα του διαμορφωμένου χώρου διευθύνσεων). Παρατηρήστε ότι το BLE επιτυγχάνει χαμηλότερες ταχύτητες από τους ρυθμούς δεδομένων IEEE 802.11ah. 1.4.6 3GPP MTC Το φάσμα GSM έχει αναδιοργανωθεί επιτρέποντας συχνότητες κάτω του 1 GHz, αυτές είναι οι συχνότητες που χρησιμοποιούνται από το 3GPP MTC και περιλαμβάνονται στις εκδόσεις 12 και 13. Το 3GPP MTC περιλαμβάνει τη δυνατότητα μεγαλύτερης περιοχής κάλυψης και τον υψηλότερο αριθμό υποστηριζόμενων συσκευών (περισσότερες από 100.000 συσκευές) [18]. Η προδιαγραφή 3GPP MTC λειτουργεί σε αδειοδοτημένο φάσμα σε ζώνες συχνοτήτων κάτω των 5 GHz, με ταχύτητα μετάδοσης δεδομένων έως και 1 Mbps, φτάνοντας το εύρος κάλυψης 100 km. Με τον ίδιο τρόπο όπως και με την τεχνολογία NB-IoT, το IEEE 802.11ah εκπέμπει υπό ζώνες συχνοτήτων χωρίς άδεια και το κόστος ανάπτυξης και λειτουργίας του 3GPP MTC μπορεί να είναι υψηλότερο από αυτό του IEEE 802.11ah.

2.10 IEEE 802.11ah (Wi-Fi HaLow)

Σήμερα, δισεκατομμύρια συσκευές IoT έχουν ήδη αναπτυχθεί, οι επικοινωνίες δικτύου IoT πρέπει να υποστηρίζουν μεταδόσεις των ήδη συνδεδεμένων συσκευών και να προετοιμάζονται για τα επερχόμενα 75 δισεκατομμύρια συσκευές που αναμένεται να συνδεθούν έως το 2025 [22] σενάρια. Αυτές οι τεράστιες εφαρμογές του παραδείγματος IoT θα φέρουν αλλαγές σε πολλές πτυχές της ζωής μας. Η συζήτηση σχετικά με το ποια τεχνολογία θα πρέπει να ηγηθεί αυτής της επανάστασης δεν έχει ακόμη διευθετηθεί. Με τα χρόνια, υπήρχαν πολλοί διεκδικητές, ενώ το Wi-Fi φαινόταν να παρατηρεί από τον πάγκο.

Λαμβάνοντας υπόψη το νέο παγκόσμιο σενάριο για τις επικοινωνίες IoT, το IEEE 802.11 WG κυκλοφόρησε το πρότυπο IEEE 802.11ah ή HaLow, όπως φέρει την επωνυμία της Wi-Fi Alliance. Το IEEE 802.11ah έχει αναπτυχθεί για την υποστήριξη εφαρμογών IoT και τις προκλήσεις που απαιτούνται για αυτά τα δίκτυα IoT, όπως:

- μεγάλος αριθμός αυτόνομων συσκευών που στέλνουν κίνηση ταυτόχρονα
- χαμηλή κατανάλωση ενέργειας
- και μεγάλο εύρος κάλυψης.

Επομένως, το IEEE 802.11ah είναι η πρώτη προσέγγιση του IEEE 802.11 WG που μπορεί να ενεργοποιήσει συγκεκριμένες λειτουργίες IoT σε χιλιάδες σταθμούς που λειτουργούν στη ζώνη συχνοτήτων κάτω του 1 GHz Βιομηχανική, Επιστημονική και Ιατρική (ISM). [23], [24], [25]. Στο κεφάλαιο αυτό, θα παρουσιαστούν οι βασικές αρχές για το πρότυπο επικοινωνιών IoT IEEE 802.11ah.

Προκειμένου να αντιμετωπιστούν οι απαιτήσεις των σεναρίων IoT, εξηγείται λεπτομερώς η βελτίωση στο επίπεδο PHY και MAC. Επιπλέον, γίνεται σύγκριση των κύριων χαρακτηριστικών του IEEE 802.11ah και των προηγούμενων τροποποιήσεων του IEEE 802.11. Μέρος αυτού του κεφαλαίου έχει δημοσιευθεί στο [19] Επισκόπηση της τροποποίησης IEEE 802.11ah Το πρότυπο IEEE 802.11ah στοχεύει στην οργάνωση των επικοινωνιών μεταξύ διαφόρων συσκευών που χρησιμοποιούνται σε εφαρμογές IoT όπως έξυπνα δίκτυα, έξυπνοι μετρητές, έξυπνα σπίτια, συστήματα υγειονομικής περίθαλψης και έξυπνη βιομηχανία . Προκειμένου να εκτεθούν οι βασικοί μηχανισμοί της επερχόμενης τροποποίησης IEEE 802.11ah, οι συγγραφείς στο [1] παρέχουν μια ολοκληρωμένη επισκόπηση.

Ομοίως, στο [26] οι συγγραφείς περιγράφουν λεπτομερώς τα διακριτά χαρακτηριστικά του IEEE 802.11ah. Στο [27], οι συγγραφείς υπογραμμίζουν τη σημασία του προτύπου IEEE 802.11ah ως μία από τις βασικές τεχνολογίες για χαμηλό κόστος, ενεργειακά αποδοτική και μαζική ανάπτυξη συσκευών IoT στο μέλλον. Επιπλέον, οι συγγραφείς αξιολογούν τη μέγιστη επιτευχθείσα απόδοση σε τρία διαφορετικά σχήματα διαμόρφωσης και κωδικοποίησης (MCS) του IEEE 802.11ah χρησιμοποιώντας σημαντικές υποθέσεις. Επίσης, στο [26], οι συγγραφείς δείχνουν αποτελέσματα απόδοσης για δεδομένα μέτρησης του IEEE 802.11ah ως προς τον ρυθμό και το εύρος. Συγκρίνουν το IEEE 802.11ah με το 802.11b και το 802.11n για τρεις εσωτερικές θήκες χωρίς να λαμβάνουν υπόψη το σενάριο εξωτερικού χώρου που είναι η πιο χρησιμοποιήσιμη περίπτωση για το IEEE 802.11ah. Η εργασία στο [24] παρέχει μια

ολοκληρωμένη επισκόπηση του IEEE 802.11ah. Επιπλέον, οι συγγραφείς συνοψίζουν τις διαδικασίες τυποποίησης καθώς και τις τεχνικές προκλήσεις που αναμένονται στην προσαρμογή του προτύπου IEEE 802.11ah. Στο [28] , οι συγγραφείς ορίζουν διαφορετικές περιπτώσεις καινοτόμου χρήσης για το πρότυπο IEEE 802.11ah.

Μεταξύ των προτεινόμενων περιπτώσεων χρήσης, οι συγγραφείς επισημαίνουν μια ενδιαφέρουσα περίπτωση όπου το πρότυπο IEEE 802.11ah θα είναι σε θέση να παρέχει το κατάλληλο χαρακτηριστικό ως σύνδεσμο backhaul για να εξυπηρετεί την ανταλλαγή κίνησης σε μεγάλες αποστάσεις (δηλ. αισθητήρες φύλλων και ροή εικόνων κάμερας ή βίντεο παρακολούθησης)

2.10.1 ΕΠΙΣΚΟΠΗΣΗ ΤΟΥ IEEE 802.11AH

Ο κύριος στόχος σχεδιασμού για το IEEE 802.11ah είναι να καλύψει τις απαιτήσεις πολλών εφαρμογών IoT και M2M. Υπονοεί ότι το πρότυπο πρέπει να σχεδιαστεί έτσι ώστε να υποστηρίζει τη λειτουργία ενός τεράστιου αριθμού συσκευών σε μέσα που βασίζονται σε διαμάχη υπό αυστηρές συνθήκες περιορισμένης ενέργειας. Το IEEE 802.11ah υπόσχεται να λύσει αυτά τα προβλήματα εισάγοντας διάφορα χαρακτηριστικά και μηχανισμούς που το καθιστούν κατάλληλο για ασύρματα δίκτυα εκτεταμένης εμβέλειας υψηλής πυκνότητας για τερματικούς σταθμούς με μπαταρία [11]. Πριν συνεχίσουμε να εξηγούμε τις λεπτομέρειες του IEEE 802.11ah, είναι σημαντικό να εστιάσουμε στα προβλήματα που προσπαθεί να λύσει και τις πιθανές και προβλεπόμενες περιπτώσεις χρήσης του. Στην επόμενη ενότητα περιγράφουμε συνοπτικά αυτά τα προβλήματα και ως εκ τούτου το κίνητρο πίσω από την ανάπτυξη αυτού του νέου προτύπου.

2.10.2 Σύγκριση του 802.11ah σε σχέση με τα άλλα πρωτόκολλα IoT

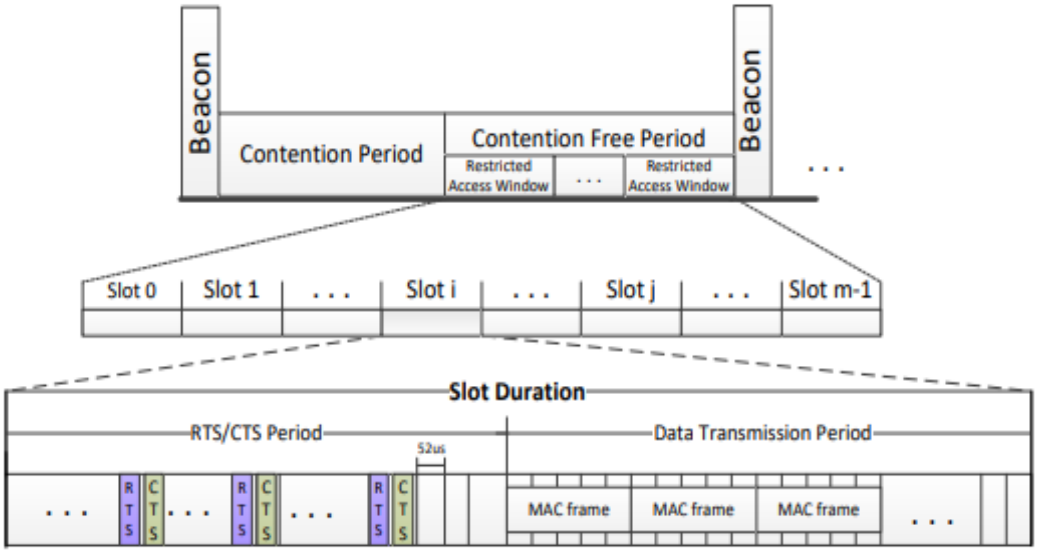
Σε σύγκριση με το 802.11, το 80.2.11ah προσθέτει τρία πεδία (σε κίτρινο χρώμα) σε ένα beacon, συγκεκριμένα τη σημαία Change Sequence (CS), Time of Next Beacon και Time of Next Beacon[33].

(1) Αλλαγή ακολουθίας (CS): Το μήκος είναι ένα byte. Κάθε φορά που το AP στέλνει ένα νέο μήνυμα beacon, η τιμή του CS αυξάνεται κατά ένα.

(2) Time of Next Beacon: Το μήκος είναι τρία byte για την καταγραφή του χρόνου αποστολής του επόμενου beacon.

(3) Time of Next Beacon: Το μήκος είναι μόνο ένα bit, που χρησιμοποιείται για να υποδείξει εάν το πεδίο Time of Next Beacon προστίθεται στην κεφαλίδα Beacon.

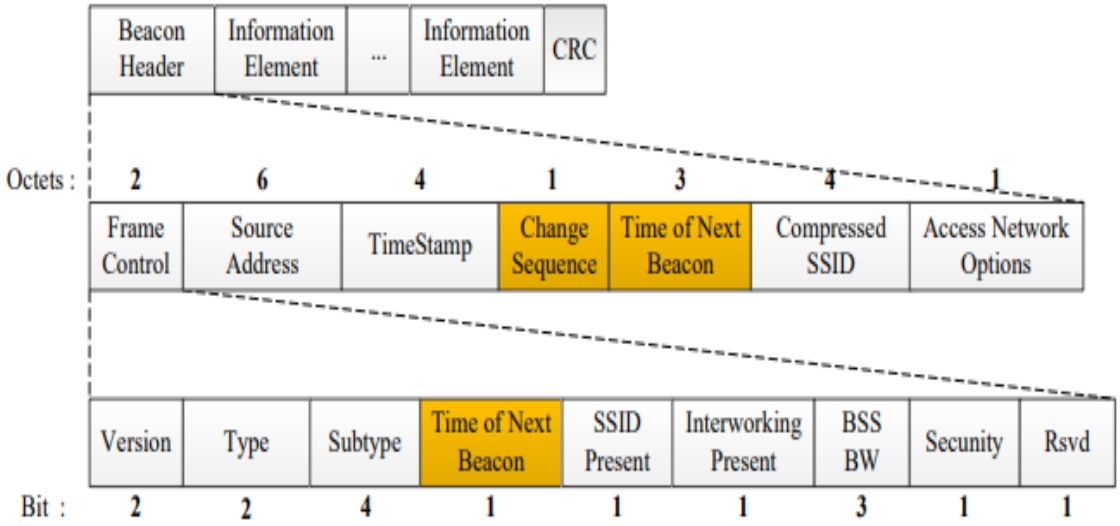
Το πρωτόκολλο 80.2.11ah χρησιμοποιεί ζώνες εξαιρούμενων αδειών 900 MHz για να παρέχει εκτεταμένη εμβέλεια δικτύων Wi-Fi, σε σύγκριση με τα συμβατικά δίκτυα Wi-Fi που λειτουργούν στις ζώνες 2,4 GHz και 5 GHz. Επιπλέον, επωφελείται από τη χαμηλότερη κατανάλωση ενέργειας, επιτρέπει τη δημιουργία μεγάλων ομάδων σταθμών ή αισθητήρων που συνεργάζονται για να μοιράζονται ένα ασύρματο μέσο και υποστηρίζει την έννοια του IoT. Για να μετριάσει τη σοβαρή διαμάχη που προκαλείται από μεγάλο αριθμό συσκευών αισθητήρων σε περιβάλλοντα M2M, το 802.11ah εισάγει ένα παράθυρο περιορισμένης πρόσβασης (RAW) σε κάθε διάστημα beacon (BI). Όπως φαίνεται στην εικόνα 14, το 802.11ah διαιρεί ένα BI σε μια περίοδο διενέξεων (CP) και μια περίοδο χωρίς αμφισβητήσεις (CFP).



Εικόνα 14- ***Ανάλυση Bacon Internal – πρωτόκολλο 802.11AH***

Πηγή: [33]

Το CFP είναι απλώς το RAW. Το RAW χωρίζεται περαιτέρω σε πολλές χρονοθυρίδες, καθεμία από τις οποίες περιλαμβάνει μια περίοδο RTS/CTS (RCP) και μια περίοδο μετάδοσης δεδομένων (DTP). Κατά τη διάρκεια του RCP, οι συσκευές αισθητήρων επιτρέπεται να διεκδικούν ευκαιρίες μετάδοσης στο DTP ακολουθώντας τη λειτουργία κατανεμημένου συντονισμού (DCF). Για το άλλο τμήμα σε ένα BI, δηλαδή το CP, επιτρέπεται σε οποιεσδήποτε συσκευές να διεκδικούν πρόσβαση στο μέσο. Πιο αναλυτικά, η εικόνα 15 δείχνει τη μορφή ενός τέτοιου πλαισίου[33].



Εικόνα 15- Μορφή Beacon στα δίκτυα με πρωτόκολλο 802.11ah

Πηγη: [33]

2.10.3 ORIGINAL 802.11AH MODULE

Ο αρχικός προσομοιωτής IEEE 802.11ah ns-3 αναπτύχθηκε και δημοσιεύτηκε το 2016 και βασίζεται στο υπάρχον μοντέλο IEEE 802.11n στο ns-3, έκδοση 3.23. Όπως φαίνεται στην Εικόνα 2, η μονάδα 802.11ah ns-3 αποτελείται από 4 κύρια στοιχεία:

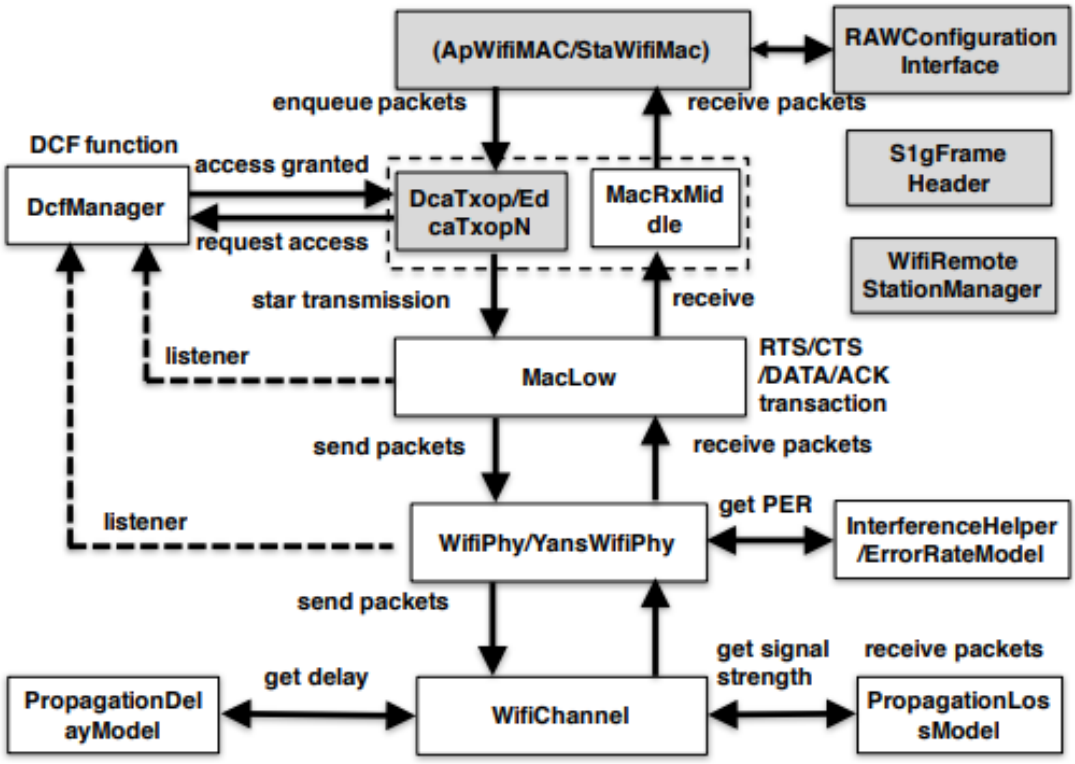
• Κανάλι Wi-Fi: Μια προσέγγιση του φυσικού μέσου μέσω του οποίου μεταδίδονται τα δεδομένα, συμπεριλαμβανομένου του μοντέλου απώλειας διάδοσης και του μοντέλου καθυστέρησης. Το μοντέλο απώλειας διάδοσης χαρακτηρίζει την απώλεια ισχύος σήματος κατά τη μετάδοση μέσω του αέρα, το μοντέλο καθυστέρησης διάδοσης περιγράφει την καθυστέρηση μετάδοσης μεταξύ δύο κόμβων.

• Επίπεδο Wi-Fi Phy: Το τμήμα PHY της μονάδας IEEE 802.11ah, όπου ορίζεται η μορφή του πλαισίου Μονάδας Δεδομένων Πρωτοκόλλου PLCP (PPDU) και τα καρέ

αποστέλλονται και λαμβάνονται μέσω του καναλιού Wi-Fi. Αποτελείται από τις κλάσεις WifiPhy/YansWifiPhy, InterferenceHelper και ErrorRateModel.

• Χαμηλό επίπεδο Mac: Περιλαμβάνει τις τάξεις MacLow, DcaTxop/EdcaTxopN, MacRxMiddel και WifiStationManager, υλοποιεί συναλλαγές RTS/CTS/DATA/ACK, Distributed Coordination Function (DCF), Enhanced Distributed Channel Access (EDCA), μέρος των RAW, πακέτα , κατακερματισμός, αναμετάδοση και έλεγχος ρυθμού.

• Υψηλό επίπεδο Mac: Εφαρμόζει λειτουργίες διαχείρισης όπως δημιουργία beacon, ανίχνευση, συσχέτιση, γρήγορη σύνδεση και μέρος του RAW. Αποτελείται από τις κλάσεις ApWifiMac (Σημείο Πρόσβασης (AP)) και StaWifiMac (μη σταθμός AP), οι οποίες μοιράζονται μια κοινή γονική κλάση RegularWifiMac

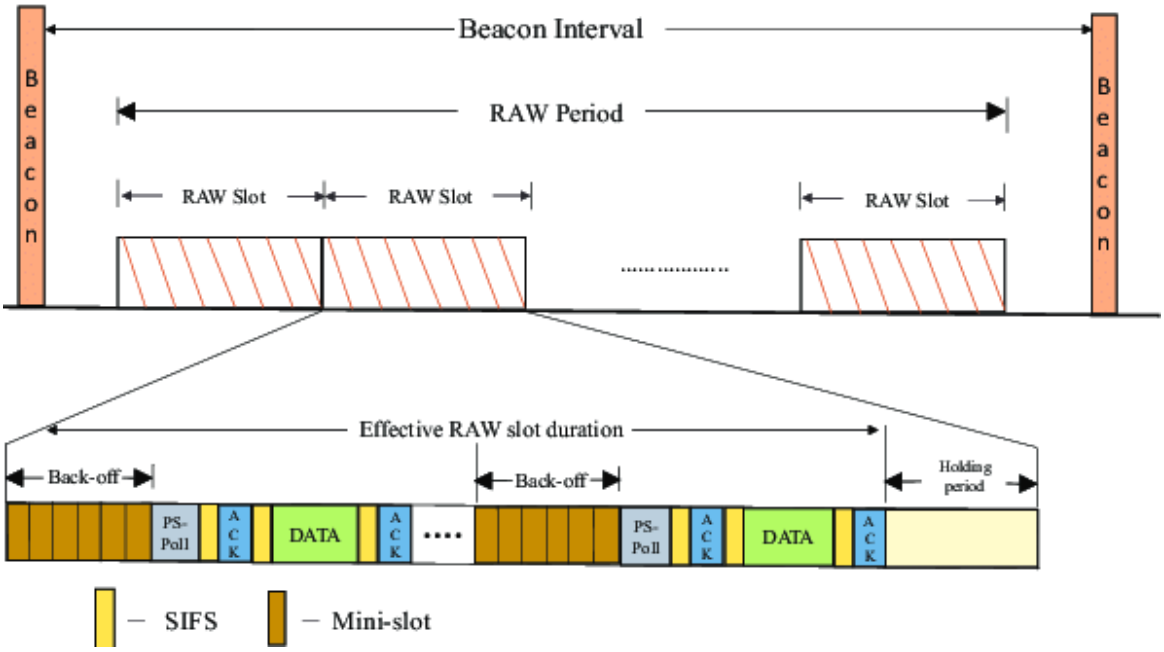


Εικόνα 16- ns-3 802.11ah μοντέλο Wi-Fi.

2.10.4 Χαρακτηριστικά επιπέδου πρόσβασης μέσο του πρωτοκόλλου IEEE 802.11ah

Το IEEE 802.11ah εισάγει αρκετές νέες δυνατότητες στο επίπεδο MAC, όπως γρήγορη συσχέτιση, RAW, τμηματοποίηση TIM και TWT, με στόχο να ανταποκριθεί στις

απαιτήσεις των πυκνών δικτύων IoT. Μεταξύ αυτών των δυνατοτήτων, οι μηχανισμοί γρήγορης συσχέτισης και RAW υλοποιούνται τροποποιώντας τα επίπεδα Mac high (δηλαδή StaWifiMac, ApWifiMac) και χαμηλά επίπεδα Mac (δηλαδή DcaManager, DcaTxop, EdcaTxopN). Η δυνατότητα γρήγορης συσχέτισης μειώνει τον χρόνο συσχέτισης όταν ένας μεγάλος αριθμός σταθμών προσπαθούν ταυτόχρονα να συσχετιστούν με το AP.



Εικόνα 17- ***Raw μηχανισμός στο πρωτόκολλο***

Πηγή: [32]

Ένα στοιχείο ελέγχου ελέγχου ταυτότητας περιέχει ένα όριο που μεταδίδεται σε όλους τους σταθμούς στο πλαίσιο του φάρου. Όταν αρχικοποιείται, ένας σταθμός δημιουργεί μια τυχαία τιμή από το διάστημα [0, 1022] και στέλνει ένα αίτημα συσχέτισης μόνο εάν η τυχαία τιμή είναι μικρότερη από το όριο που λαμβάνεται από τον λαμβανόμενο φάρο. Διαφορετικά αναβάλλει τη συσχέτιση μέχρι τον επόμενο φάρο. Το AP προσαρμόζει δυναμικά το όριο για να επιτρέψει σε όλους τους σταθμούς να συσχετιστούν σε εύθετο χρόνο.

Ο στόχος του μηχανισμού RAW είναι να μειώσει τις συγκρούσεις και να βελτιώσει την απόδοση σε πυκνά δίκτυα IoT όπου εκατοντάδες ή χιλιάδες συνδεδεμένοι σταθμοί χρειάζονται πρόσβαση στα κανάλια. Όπως δείχνει το σχήμα 3, το RAW χωρίζει τους σταθμούς σε ομάδες. Κάθε ομάδα έχει μία ή περισσότερες κουλοχέρηδες στις οποίες οι

σταθμοί που ανήκουν σε αυτήν την ομάδα χωρίζονται ομοιόμορφα. Κατά τη διάρκεια μιας υποδοχής RAW, μόνο οι σταθμοί που ανήκουν σε αυτήν την υποδοχή επιτρέπεται να έχουν πρόσβαση στο κανάλι.

Το IEEE 802.11ah χρησιμοποιεί το στοιχείο πληροφοριών RAW Parameter Set (RPS) για τη μετάδοση πληροφοριών σχετικά με ομάδες RAW στην αρχή κάθε διαστήματος beacon. Καθορίζει τους σταθμούς που ανήκουν σε κάθε ομάδα RAW, τον αριθμό των υποδοχών, τη μορφή υποδοχής και τον αριθμό της διάρκειας υποδοχής. Τα δύο τελευταία καθορίζουν από κοινού τη διάρκεια της υποδοχής RAW. Οι σταθμοί που ανήκουν σε μια ομάδα RAW έχουν διαδοχικό αναγνωριστικό συσχέτισης (AID) και έχουν εκχωρηθεί σε υποδοχές RAW με τρόπο στρογγυλής ροής.

Οι παράμετροι που σχετίζονται με το RAW μπορούν να προρυθμιστούν μέσω χαρακτηριστικών της κλάσης ApWifiMac από πειραματιστές. Πριν αρχίσει να εκτελείται η προσομοίωση, αυτή η ρύθμιση παραμέτρων RAW που ορίζει ο χρήστης χρησιμοποιείται για τη διαμόρφωση των ιδιοτήτων που σχετίζονται με RAW του ApWifiMac. Κατά τη διάρκεια μιας προσομοίωσης, το AP δημιουργεί ένα στοιχείο RPS με την προκαθορισμένη διαμόρφωση και το προσαρτά σε κάθε πλαίσιο beacon. Το RPS μπορεί να διαμορφωθεί μόνο στατικά και δεν μπορεί να αλλάξει μόλις αρχίσει να εκτελείται η προσομοίωση, ακόμα κι αν αλλάξουν οι συνθήκες δικτύου και οι προρυθμισμένες παράμετροι RAW καταστούν μη βέλτιστες.

Τα συστήματα M2M και IoT αναμένεται να επιτρέψουν ένα ευρύ φάσμα σημαντικών υπηρεσιών και εφαρμογών, όπως έξυπνη μέτρηση, παρακολούθηση υγειονομικής περίθαλψης, διαχείριση και παρακολούθηση eet, τηλεπισκόπηση, βιομηχανικός αυτοματισμός, αγροτική παρακολούθηση και συναλλαγές κατά παραγγελία επιχειρηματικής χρέωσης μεταξύ πολλών άλλων [14, 18].

Ένας τεράστιος αριθμός συσκευών είναι πιθανό να συνδεθεί για να ενεργοποιήσει αυτές τις υπηρεσίες και εφαρμογές, κάτι που είναι η κύρια πρόκληση για τις υπάρχουσες τεχνολογίες. Επιπλέον, σε πολλά σενάρια των περιπτώσεων χρήσης IoT, τα τερματικά πρέπει να λειτουργούν χωρίς αντικατάσταση ή επαναφόρτιση μπαταρίας για έως και πολλά χρόνια. Η ενεργειακή απόδοση γίνεται επομένως πρωταρχικής σημασίας κατά το σχεδιασμό ενός δικτύου M2M [18, 19].

Επιπλέον, κρίνεται επίσης απαραίτητο για τους φορείς εκμετάλλευσης δικτύων να μπορούν να προσφέρουν υπηρεσίες και συσκευές M2M σε χαμηλότερα επίπεδα κόστους ενώ εξυπηρετούν σχετικά μεγαλύτερες περιοχές. Λόγω του αυξανόμενου ενδιαφέροντος για εφαρμογές IoT, M2M και αισθητήρων, η βιομηχανία WLAN έχει επίσης λάβει σημαντικά βήματα για την αντιμετώπιση αυτού του επιχειρηματικού τμήματος εισάγοντας τη νέα τροποποίηση IEEE 802.11ah [11, 20] στο βασικό πρότυπο IEEE 802.11 λαμβάνοντας υπόψη κυρίως τα προαναφερθέντα εφαρμογές

3. Προκλήσεις και πλεονεκτήματα για το IEEE 802.11ah

Στην παρούσα ενότητα θα γίνει αναφορά στις προκλήσεις και τα πλεονεκτήματα για το πρότυπο IEEE 802..11 ah.

Η κύρια πρόκληση για το πρότυπο IEEE 802.11ah είναι οι αυστηρές απαιτήσεις χωρητικότητας και ενεργειακής απόδοσης πολλών από τις περιπτώσεις χρήσης. Στη συνέχεια θα συζητήσουμε αυτές τις προκλήσεις και τα προτεινόμενα χαρακτηριστικά στο πρότυπο για την αντιμετώπισή τους.

3.1 Λειτουργία πυκνού δικτύου

Το IEEE 802.11ah ενδέχεται να απαιτεί πολύ πυκνή λειτουργία του δικτύου. Για παράδειγμα, έως και 6000 σταθμοί μπορούν να συσχετιστούν σε ένα μόνο σημείο πρόσβασης (AP) σε μία από τις περιπτώσεις χρήσης. Δεν είναι δυνατό να παρασχεθούν αποδεκτοί ρυθμοί δεδομένων σε τόσο υψηλά σενάρια αντιπαράθεσης χρησιμοποιώντας τις τεχνικές MAC παλαιού τύπου. Το IEEE 802.11ah, επομένως, περιγράφει πολλούς μηχανισμούς για την αντιμετώπιση αυτού του προβλήματος. Το Παράθυρο περιορισμένης πρόσβασης (RAW) είναι ένας από τους μηχανισμούς που επιτρέπει τη λειτουργία μεγάλου αριθμού συσκευών σε ένα μόνο BSS χωρίς να υποβαθμίζει την απόδοση της απόδοσης σε ανεπαρκή επίπεδα. Αυτό το κάνει περιορίζοντας την πρόσβαση καναλιού στο BSS σε μια υποομάδα σταθμών στο καθορισμένο χρονικό διάστημα. Ένας άλλος μηχανισμός μετριασμού που περιγράφεται στο πρότυπο είναι ο μηχανισμός τομεοποίησης που αντιμετωπίζει ζητήματα όπως παρεμβολές και πρόβλημα κρυφού κόμβου σε δίκτυα υψηλής πυκνότητας. Αυτοί οι μηχανισμοί θα συζητηθούν αργότερα λεπτομερώς.

Πρόβλημα OBSS

Το πρόβλημα OBSS αναφέρεται στην περίπτωση που δύο ή περισσότερα BSS, που δεν είναι συγχρονισμένα και λειτουργούν στο ίδιο κανάλι, είναι αρκετά κοντά ώστε να ακούγονται μεταξύ τους. Σε ένα τέτοιο σενάριο, οι εκπομπές από ορισμένα STA στο ένα BSS θα ακούγονται από τα STA στο άλλο και τελικά θα υποβαθμίσουν τη συνολική απόδοση. Μια τέτοια μελέτη έγινε στο [5] όπου οι συγγραφείς κατέληξαν στο συμπέρασμα ότι το πρόβλημα OBSS υποβαθμίζει την απόδοση ουσιαστικά καθώς αυξάνεται η επικάλυψη μεταξύ των δύο σημείων πρόσβασης. Για να το αντιμετωπίσετε αυτό, οι προδιαγραφές IEEE 802.11ah παρέχουν τον μηχανισμό τομεοποίησης. Ο μηχανισμός τομεοποίησης όχι μόνο βοηθά στη μείωση της παρεμβολής από γειτονικά BSS, αλλά και απαλλαγεί από το πρόβλημα του κρυφού κόμβου, το οποίο είναι επίσης μια βαθιά πηγή βλάβης στα ασύρματα δίκτυα χρησιμοποιώντας βασικό μηχανισμό πρόσβασης. Υπάρχουν δύο είδη μηχανισμών τομεοποίησης που περιγράφονται στο νέο πρότυπο, τους οποίους θα καλύψουμε αργότερα λεπτομερώς.

Η επιλεκτική μετάδοση υποκαναλιών είναι μια άλλη τεχνική που μπορεί να χρησιμοποιηθεί για τη μείωση του αντίκτυπου των παρεμβολών από γειτονικό BSS. Σε αυτή τη εργασία αξιολογούμε διεξοδικά την απόδοση αυτών των νέων μηχανισμών μέσω εκτενών προσομοιώσεων που δίνουν μια ρεαλιστική εκτίμηση των βελτιώσεων απόδοσης που μπορούν να επιτευχθούν με τη χρήση αυτών των χαρακτηριστικών. Τώρα που περιγράψαμε τα κίνητρα πίσω από αυτό το νέο πρότυπο και τις ισχύουσες περιπτώσεις χρήσης, μπορούμε να προχωρήσουμε στην παροχή μιας επισκόπησης του ίδιου του προτύπου.

Προκειμένου να οπτικοποιηθούν οι προκλήσεις στις επικοινωνίες IoT, μπορούμε να διακρίνουμε τις τυπικές απαιτήσεις όπως μεγάλος αριθμός αυτόνομων συσκευών που στέλνουν κίνηση (ταυτόχρονα ή σε μεταγενέστερους χρόνους), χαμηλή κατανάλωση ενέργειας και μεγάλο χρόνο ύπνου. Σε αυτήν την ενότητα παρέχουμε μια επισκόπηση του μηχανισμού που χρησιμοποιείται από το IEEE 802.11ah για την αντιμετώπιση αυτών των προκλήσεων.

3.2 Εύρος κάλυψης

Ορισμένες από τις εφαρμογές IoT απαιτούν κάλυψη άνω του 1 km για την επιθυμητή λειτουργία τους. Εκτός από το γεγονός ότι χρησιμοποιεί μια ζώνη

χαμηλότερης συχνότητας (υπό-1 GHz) με καλύτερα χαρακτηριστικά διάδοσης, στο IEEE 802.11ah, η απαίτηση εκτεταμένης εμβέλειας ικανοποιείται με την εισαγωγή ευρείας μετάδοσης 1 MHz και με τη χρήση ενός νέου σχήματος διαμόρφωσης και κωδικοποίησης (MCS).) ευρετήριο (MCS10). Αυτό το σχήμα είναι ουσιαστικά MCS0 (BPSK 1/2) με προσθήκη 2× επανάληψης. Μαζί με το εύρος ζώνης καναλιού 1 MHz (CBW), το IEEE 802.11ah υποστηρίζει επίσης 2, 4, 8 και 16 MHz (αναμένεται ότι οι πρώτες εμπορικές συσκευές υποστηρίζουν έως και 4 MHz).

Αυτά τα στενότερα εύρη ζώνης συνεπάγονται μεγαλύτερη διάρκεια συμβόλου από το παλαιού τύπου IEEE 802.11. Με μεγαλύτερα σύμβολα (και διαστήματα προστασίας), οι εκπομπές IEEE 802.11ah είναι πιο ανθεκτικές σε παρεμβολές μεταξύ συμβόλων που συναντώνται σε μεγαλύτερες συνδέσεις και σενάρια εξωτερικού χώρου (μεγάλη εξάπλωση καθυστέρησης). Με την υποστήριξη πολλαπλής εισόδου πολλαπλής εξόδου (MIMO), το IEEE 802.11ah επωφελείται από τη χωρική ποικιλομορφία, η οποία βελτιώνει την ποιότητα του λαμβανόμενου σήματος και, ως εκ τούτου, καθιστά δυνατές μεγαλύτερες συνδέσεις. Η προδιαγραφή εξετάζει επίσης τη λειτουργία multi-hop με ρελέ ή δικτύωση πλέγματος για επέκταση της κάλυψης.

3.3 Πόροι χρόνου και συχνότητας

Πολλές τεχνολογίες λειτουργούν ταυτόχρονα στην υπερπλήρη ζώνη συχνοτήτων των 2,4 GHz (IEEE 802.15.4e, BLE, IEEE 802.11, κ.λπ.), όπου προκαλούνται πολλές παρεμβολές, οι οποίες υποβαθμίζουν σοβαρά την απόδοση του δικτύου. Με την έλευση του IoT και την αύξηση του αριθμού των συσκευών που εφαρμόζουν αυτές τις τεχνολογίες, η μοίρα αυτής της ζώνης δεν φαίνεται πολλά υποσχόμενη. Αντίθετα, τα προβλήματα επικοινωνίας, όπως η παρεμβολή μεταξύ καναλιών, η οποία είναι ιδιαίτερα επιβλαβής σε συστήματα πρόσβασης που μοιάζουν με Carrier Sense Multiple Access (CSMA), θα επιδεινωθούν.

Ωστόσο, η τροποποίηση IEEE 802.11ah προορίζεται να λειτουργεί κάτω από 1 GHz, η οποία, εκτός από τη βελτιωμένη κάλυψη, αντιμετωπίζει λιγότερες παρεμβολές. Αυτό

το χαρακτηριστικό του IEEE 802.11ah φαίνεται ιδιαίτερα ελκυστικό για εφαρμογές IoT, όπου αναμένεται να συνυπάρχουν εκατοντάδες ή χιλιάδες συσκευές.

3.4 Υποστήριξη μεγάλου αριθμού συσκευών IoT

Τα δίκτυα IoT έχουν το κύριο χαρακτηριστικό ότι σχηματίζονται από μεγάλο αριθμό αυτόνομων συσκευών (που συνήθως κυμαίνονται από εκατοντάδες έως μερικές χιλιάδες). Αυτό οφείλεται στο γεγονός ότι πολλές από τις εφαρμογές αναμένεται να λειτουργούν σε μεγάλη περιοχή. Ωστόσο, οι συγκρούσεις συμβαίνουν συχνά όταν ένας μεγάλος αριθμός συσκευών προσπαθεί να επικοινωνήσει ταυτόχρονα.

Οι υπερβολικές συγκρούσεις έχουν ως αποτέλεσμα μειωμένη συνολική απόδοση στο δίκτυο και, επομένως, η εύρεση κατάλληλων μεθόδων για τη μείωση των συγκρούσεων αποτελεί πρόκληση για το IoT. Το IEEE 802.11ah ορίζει έναν προαιρετικό νέο μηχανισμό πρόσβασης καναλιού διενέξεων που ονομάζεται Παράθυρο περιορισμένης πρόσβασης (RAW).

Αυτή η μέθοδος πρόσβασης έχει σχεδιαστεί για να μειώνει τις συγκρούσεις βελτιώνοντας την απόδοση του καναλιού διαιρώντας τους σταθμούς σε διαφορετικές ομάδες και περιορίζοντας την πρόσβαση καναλιών μόνο σε μια ομάδα σε μια συγκεκριμένη χρονική περίοδο. Το παλαιού τύπου IEEE 802.11 υποστηρίζει έως και 2007 συσχετισμένους σταθμούς ανά σημείο πρόσβασης (AP), λόγω του περιορισμένου αριθμού διαθέσιμων αναγνωριστικών συσχέτισης (AID) που μπορούν να εκχωρηθούν σε κάθε συσχετισμένο σταθμό.

Προκειμένου να αυξηθεί ο αριθμός των υποστηριζόμενων σταθμών από το AP, το IEEE 802.11ah χρησιμοποιεί μια νέα ιεραρχική δομή AID. Το νέο AID αποτελείται από 13 bit και έτσι ο αριθμός των υποστηριζόμενων σταθμών αυξάνεται σε 213 − 1 (8191). Η δομή του AID αποτελείται από τέσσερα ιεραρχικά επίπεδα (δηλαδή, σελίδα, μπλοκ, υπομπλοκ και ευρετήριο σταθμού σε υπομπλοκ). Το IEEE 802.11ah χρησιμοποιεί την προαναφερθείσα δομή για να ομαδοποιήσει σταθμούς με βάση παρόμοια χαρακτηριστικά (π.χ. μοτίβο κυκλοφορίας, τοποθεσία, επίπεδο μπαταρίας κ.λπ.).

3.5 Χαμηλή κατανάλωση ενέργειας

Λαμβάνοντας υπόψη το γεγονός ότι πολλές συσκευές IoT λειτουργούν με μπαταρία και προορίζονται να λειτουργούν για ημέρες, εβδομάδες, μήνες ή χρόνια (ανάλογα με την εφαρμογή), η χαμηλή κατανάλωση ενέργειας γίνεται μια κρίσιμη πτυχή για την αύξηση της διάρκειας ζωής της μπαταρίας. Οι συσκευές IoT είναι εξοπλισμένες με ενσωματωμένη κάρτα διεπαφής δικτύου (NIC) και έτσι έχουν τη δυνατότητα να επικοινωνούν αυτόνομα μέσα στο δίκτυο στο οποίο ανήκουν. Το ασύρματο NIC αντιπροσωπεύει ένα μεγάλο μέρος της ενέργειας που καταναλώνεται από τη συσκευή και επομένως, ο ορισμός μιας αποτελεσματικής διαχείρισης ενέργειας για το NIC είναι υψίστης σημασίας.

Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας διαφορετικούς χρονοδιακόπτες αφύπνισης και ύπνου. Στο παλαιού τύπου IEEE 802.11, η καθορισμένη μέγιστη περίοδος αδράνειας επιτρέπει σε οποιονδήποτε σταθμό να διατηρεί την κατάσταση συσχέτισης για έως και 18,64 ώρες αδράνειας, ενώ το IEEE 802.11ah στοχεύει στη χρήση διαφορετικών περιόδων για διαφορετικές εφαρμογές, έως και σε κλίμακα έτους. Πολλά νέα χαρακτηριστικά που εισάγονται από το IEEE 802.11ah αποσκοπούν στην επίτευξη πιο αποτελεσματικών μεταδόσεων, επιτρέπει την εξοικονόμηση ενέργειας.

Για παράδειγμα, η μειωμένη επιβάρυνση λόγω μικρότερων κεφαλίδων και μηχανισμών όπως η σιωπηρή επιβεβαίωση (πλαίσια ελέγχου ACK δεν απαιτούνται σε ορισμένες περιπτώσεις), η ανταλλαγή πλαισίου ταχύτητας (μέθοδος που επιτρέπει την ανταλλαγή μιας αμφίδρομης ακολουθίας πλαισίων κατά τη διάρκεια μιας δεσμευμένης ευκαιρίας μετάδοσης (TXOP)), επεκτείνετε τη διάρκεια ζωής της μπαταρίας των σταθμών μειώνοντας το χρόνο μετάδοσης, διατηρώντας τους ξύπνιους για μικρότερα χρονικά διαστήματα.

Κόστος εγκατάστασης/λειτουργίας Wi-Fi

Το HaLow υποστηρίζει εγγενή κίνηση IP, όπως και το παραδοσιακό Wi-Fi. Η απλή εγκατάσταση απαιτεί μόνο AP ή δρομολογητή με δυνατότητα Wi-Fi HaLow. Δεν χρειάζονται ιδιόκτητοι κόμβοι ή πύλες. Η αρχιτεκτονική προσανατολισμένη στα αστέρια δεν απαιτεί πολύπλοκα πλέγματα επαναλήπτες που βαλτώνουν τα μηνύματα και προκαλούν προβλήματα διαχείρισης για άλλες ασύρματες τεχνολογίες. Χρησιμοποιεί

φάσμα κάτω του 1 GHz χωρίς άδεια χρήσης χωρίς επαναλαμβανόμενες χρεώσεις συνδρομής ή προγράμματα δεδομένων που απαιτούνται με παρόχους δικτύων κινητής τηλεφωνίας.

Φάσμα χωρίς άδεια Δ

εν απαιτείται σύμβαση παρόχου υπηρεσιών, καθώς το Wi-Fi HaLow δεν βασίζεται σε παρόχους υπηρεσιών κινητής τηλεφωνίας. Όπως το παραδοσιακό Wi-Fi, οι πελάτες διαθέτουν τον εξοπλισμό τους και χρησιμοποιούν ραδιοφάσμα χωρίς άδεια. Η ασύρματη τεχνολογία λειτουργεί στις ζώνες συχνοτήτων που εξαιρούνται από άδεια σε όλο τον κόσμο, που κυμαίνονται από 750 MHz έως 950 MHz. Δεν χρησιμοποιεί τις συχνότητες των φορέων κινητής τηλεφωνίας που συνήθως απορροφούν το κόστος των τελών άδειας χρήσης και των συνδρομών. Ενδέχεται να ισχύουν ορισμένοι περιορισμοί χρήσης και μπορεί να διαφέρουν ανά χώρα.

Διαλειτουργικότητα και εμπειρία πελάτη

Το Wi-Fi HaLow είναι ένα παγκοσμίως αναγνωρισμένο πρότυπο που καθορίζει τον τρόπο επικοινωνίας των συσκευών. Όπως και οι προηγούμενες γενιές Wi-Fi, οι προμηθευτές εξοπλισμού Wi-Fi HaLow διασφαλίζουν τη διαλειτουργικότητα ακολουθώντας τις οδηγίες της Wi-Fi Alliance, η οποία παρέχει προδιαγραφές και υπηρεσίες πιστοποίησης στα μέλη. Άλλες ασύρματες τεχνολογίες IoT είτε είναι αποκλειστικές είτε δεν έχουν τόσο καλά οργανωμένη καθοδήγηση που μπορεί να επηρεάσει το χρόνο στην αγορά, το κόστος του προϊόντος και την εμπειρία του πελάτη.

Οι χρήστες θα έχουν την ίδια ευκολία χρήσης με τα παραδοσιακά δίκτυα Wi-Fi. Προσθέστε σε αυτό την απλότητα των μεμονωμένων λύσεων AP του Wi-Fi HaLow, προσφέροντας υψηλότερη επιτυχία εγκατάστασης εκτός συσκευασίας και εξαλείφοντας την πολυπλοκότητα και τις προκλήσεις της ανάπτυξης εναλλακτικών δικτύων πλέγματος.

Υποστήριξη εγγενούς IP

Η εγγενής υποστήριξη για επισκεψιμότητα IP ορίζεται στο Wi-Fi HaLow όπως το παραδοσιακό Wi-Fi. Χρησιμοποιώντας έναν δρομολογητή με δυνατότητα Wi-Fi HaLow, όλες οι συσκευές-πελάτες μπορούν να χρησιμοποιήσουν πρωτόκολλα μεταφοράς IPv4/IPv6 για άμεση πρόσβαση στο διαδίκτυο για υπηρεσίες που βασίζονται σε cloud και διαχείριση δεδομένων IoT. Άλλες ασύρματες τεχνολογίες IoT όπως Bluetooth, Zigbee, Z-Wave, LoRa και Sigfox απαιτούν μια αποκλειστική πύλη για τη μετατροπή όλης της τοπικής κίνησης πελατών σε κίνηση IP για πρόσβαση στο Διαδίκτυο. Αυτά τα πρόσθετα στάδια επεξεργασίας πακέτων απαιτούνται για την περιτύλιξη επιπλέον δεδομένων γύρω από τα πακέτα, προσθέτοντας καθυστερήσεις και μειώνοντας την αποτελεσματικότητα των δικτύων τους.

Στο αρχικό στάδιο ανάπτυξης, το Wi-Fi HaLow αναμένεται να χρησιμοποιηθεί τόσο σε εσωτερικές όσο και σε εξωτερικές εφαρμογές όπου το τυπικό Wi-Fi δεν μπορεί να φτάσει, όπως στην περίπτωση συστημάτων επιτήρησης που λειτουργούν με μπαταρία, ασύρματων καμερών και κουδουνιών πόρτας. Μια άλλη τυπική περίπτωση χρήσης θα ήταν οι μεγάλοι χώροι, όπου ένα μόνο σημείο πρόσβασης HaLow μπορεί να υποκαταστήσει μεγάλο αριθμό AP, αποφεύγοντας αναποτελεσματικές, πολύπλοκες αρχιτεκτονικές πλέγματος, απλοποιώντας την εγκατάσταση και μειώνοντας το συνολικό κόστος ιδιοκτησίας.

Ο βιομηχανικός αυτοματισμός, οι αισθητήρες ελέγχου διεργασιών, οι αυτοματισμοί κτιρίων, οι αποθήκες και τα καταστήματα λιανικής, μεταξύ πολλών άλλων, θα χρειαστούν αυτήν την τεχνολογία, επιτρέποντας τα πάντα να παραμένουν συνδεδεμένα σε έναν όλο και πιο αυτοματοποιημένο κόσμο. Πράγματι, το Wi-Fi HaLow ξεχωρίζει για την ευελιξία του.

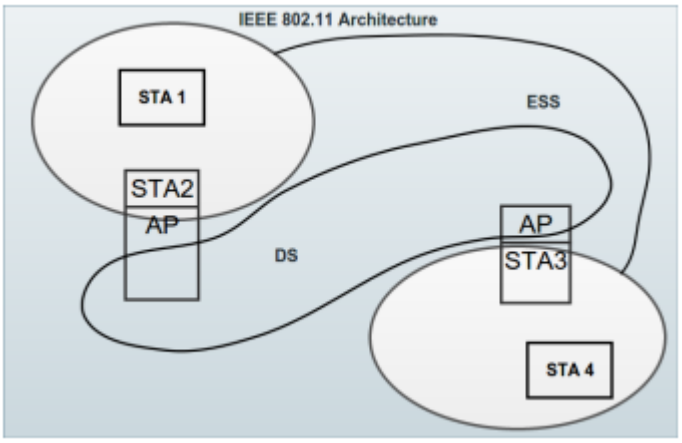
3.6 Η Αρχιτεκτονική του IEEE 802.11ah

Σε ένα δίκτυο 802.11, ένας σταθμός (STA) είναι μια ενιαία διευθυνσιοδοτούμενη μονάδα που είναι η πηγή ή ο προορισμός ενός μηνύματος. Ένα STA μπορεί να είναι xed,

φορητό ή κινητό. Η βασική αρχιτεκτονική μονάδα ενός δικτύου IEEE 802.11 είναι ένα Basic Service Set (BSS) το οποίο μπορεί να θεωρηθεί ως η περιοχή κάλυψης στην οποία τα STA παραμένουν συνδεδεμένα μεταξύ τους. Η περιοχή που καλύπτεται από το BSS ορίζεται ως Basic Service Area (BSA). Ένας STA εκτός του BSS δεν μπορεί να επικοινωνήσει απευθείας με αυτούς εντός του BSS. Ένα ανεξάρτητο βασικό σύνολο υπηρεσιών (IBSS) σχηματίζεται όταν δύο ή περισσότερα STA ικανά να επικοινωνούν απευθείας μεταξύ τους λειτουργούν σε ένα BSS σχηματίζοντας έτσι ένα λεγόμενο δίκτυο ad-hoc.

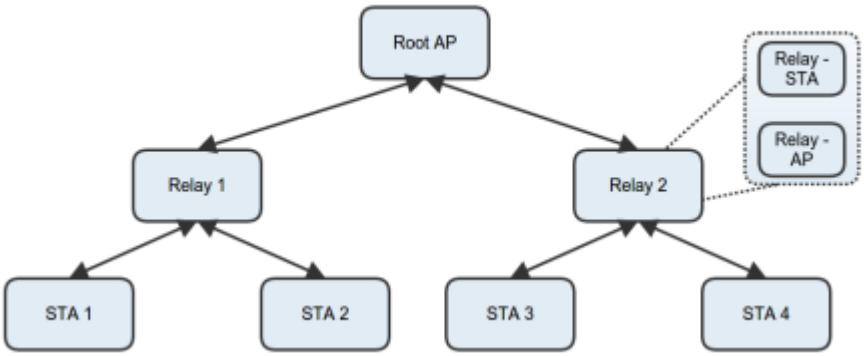
Σε αντίθεση με αυτό, σε μια υποδομή BSS τα STA συνδέονται με ένα Xed STA που ονομάζεται Access Point (AP) το οποίο εκπέμπει ειδικά πλαίσια διαχείρισης όπως πλαίσια beacon για να διατηρεί το δίκτυο συγχρονισμένο. Δύο ή περισσότερα BSS μπορούν να συνδεθούν μέσω ενός συστήματος διανομής (DS) για να σχηματίσουν ένα εκτεταμένο σύνολο υπηρεσιών (ESS). Ένα AP λειτουργεί ως πύλη για τη σύνδεση πολλαπλών BSS για να σχηματιστεί ένα αυθαίρετα μεγάλο Ασύρματο Τοπικό Δίκτυο (WLAN). Ένα ESS που σχηματίζεται με αυτόν τον τρόπο επιτρέπει στα συσχετισμένα STA να επικοινωνούν μεταξύ τους μέσω των AP και DS ακόμα κι αν δεν ανήκουν στο ίδιο BSS.

Η εικόνα 2 δείχνει τη βασική αρχιτεκτονική ενός δικτύου IEEE 802.11. Επιπλέον, τα δίκτυα IEEE 802.11 παρέχουν επίσης τη δυνατότητα σχηματισμού δικτύων πλέγματος σε ένα σύνολο βασικών υπηρεσιών Mesh (MBSS). Σε ένα MBSS κάθε STA είναι συνδεδεμένο με τα γειτονικά του STA και η δρομολόγηση πολλαπλών βημάτων χρησιμοποιείται για την παράδοση πακέτων ενώ δεν υπάρχει κεντρική οντότητα [22].



Εικόνα 18- Αρχιτεκτονική του IEEE 802.11

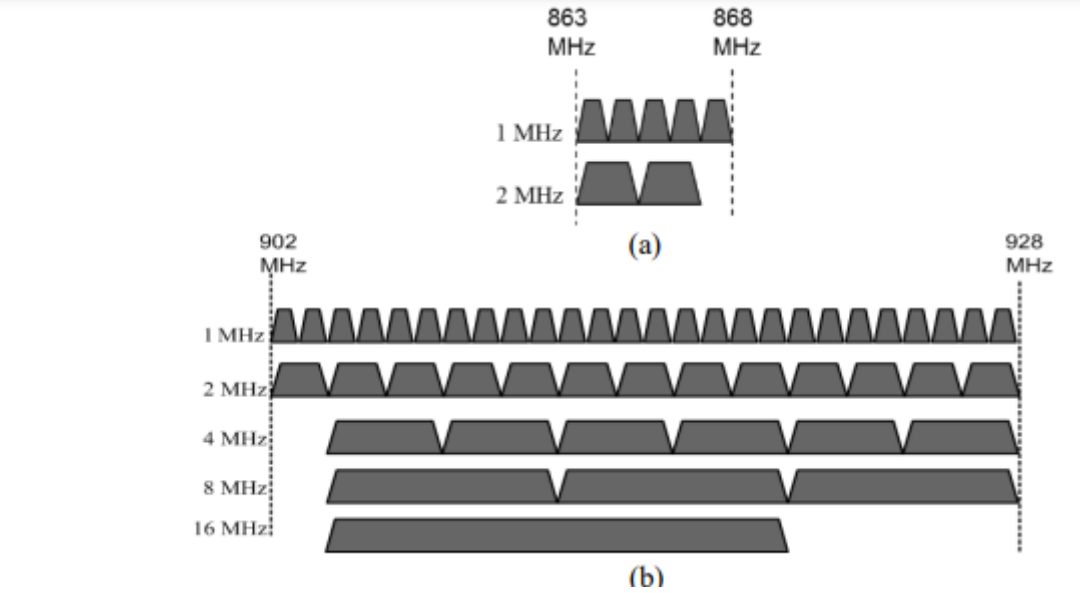
Ένα δίκτυο IEEE 802.11ah διατηρεί την αρχιτεκτονική δικτύου των παλαιών συστημάτων IEEE 802.11, επομένως οι προαναφερθείσες αρχιτεκτονικές δικτύου ενσωματώνονται επίσης στο νέο πρότυπο. Υποστηρίζει εφαρμογές Xed, εξωτερικού χώρου και από σημείο σε πολλαπλά σημεία ενώ είναι συμβατό με το επίπεδο διαχείρισης IEEE 802.11 [13]. Επιπλέον, υποστηρίζει επίσης μια αρχιτεκτονική ρελέ στην οποία η περιοχή κάλυψης ενός AP αυξάνεται με τη χρήση ενός μηχανισμού ρελέ δύο άλματος. Μια υψηλού επιπέδου αφαίρεση της λειτουργίας του ρελέ φαίνεται στην εικόνα 19.



Εικόνα 19- Relay Αρχιτεκτονική στο πρωτόκολλο IEEE 802.11

Μπορεί να φανεί στην εικόνα ότι ο ηλεκτρονόμος αποτελείται από ένα ρελέ-STA και ένα ρελέ-AP. Το ρελέ-STA είναι συνδεδεμένο στο Root AP και το ρελέ-AP συνδέεται με το τερματικό STA. Τα καρέ μπορούν να μεταδοθούν μεταξύ του Root AP και των

τερματικών STA χρησιμοποιώντας τα ρελέ και στις δύο κατευθύνσεις. Τα ρελέ όχι μόνο αυξάνουν την περιοχή κάλυψης του BSS αλλά μπορούν επίσης να μειώσουν το χρόνο μετάδοσης (συνήθως) και την κατανάλωση ενέργειας για επιτυχή παράδοση πακέτων. Στο IEEE 802.11ah, διαφορετικά BSS μπορούν να εγκατασταθούν γύρω από τις ίδιες ή διαφορετικές φέρουσες συχνότητες που καθορίζονται από τους κανονισμούς στην περιοχή δράσης. Επιπλέον, αυτά τα BSS θα λειτουργούν σε εύρη ζώνης καναλιών που κυμαίνονται από 1 MHz έως 16 MHz, ανάλογα με την πολιτική καναλοποίησης της αντίστοιχης χώρας. Η διοχέτευση για το IEEE 802.11ah στην Ευρώπη και τις ΗΠΑ φαίνεται στην εικόνα 20 [23].



Εικόνα 20- Συχνότητα καναλιών του πρωτοκόλλου IEEE 802.11ah α) Ευρώπη και β)Αμερική

Θα εξετάσουμε μόνο την υποδομή BSS στην ανάλυσή μας, καθώς είναι η πιο συχνά αναπτυσσόμενη αρχιτεκτονική δικτύου για WLAN. Σε ένα IBSS τα AP μεταδίδουν κυρίως πλαίσια διαχείρισης (π.χ. πλαίσια beacon) τα οποία βοηθούν τα STA να λειτουργούν και να παραμένουν συγχρονισμένα μέσα στο BSS. Οι AP είναι επίσης υπεύθυνοι για τη δημιουργία συσχετίσεων με τους STA που εισέρχονται στο BSS. Επιπλέον, εξυπηρετούν δεδομένα trac σε STA στην κατερχόμενη ζεύξη και ανταποκρίνονται με ACK για την εισερχόμενη διαδρομή ανερχόμενης ζεύξης. Κατά συνέπεια, λόγω του βασικού τους ρόλου στο BSS, τα AP γενικά χρειάζεται να εκπέμπουν συχνότερα από ένα μεμονωμένο

STA. Στη συνέχεια περιγράφουμε τα επίπεδα ελέγχου φυσικής και μεσαίας πρόσβασης του IEEE 802.11ah και μερικές από τις νέες δυνατότητες που έχει εισαγάγει το πρότυπο.

4. Προσομοίωση πρωτοκόλλου IEEE 802.11ah

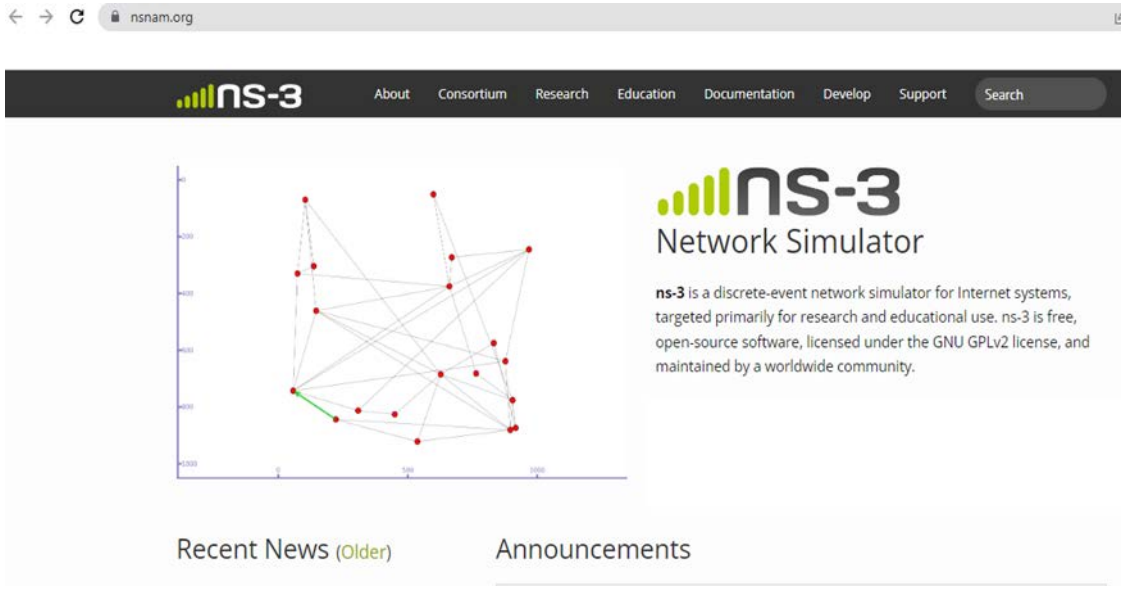
Στο κεφάλαιο αυτό θα παρουσιαστούν τα βήματα και τα αποτελέσματα προσομοίωσης του πρωτοκόλλου IEEE 802.11ah Η προσομοίωση πραγματοποιήθηκε με το λογισμικό Ns-3. Το ns-3 είναι ένας προσομοιωτής δικτύου διακριτών συμβάντων για συστήματα Διαδικτύου, που προορίζεται κυρίως για ερευνητική και εκπαιδευτική χρήση. Το ns-3 είναι δωρεάν λογισμικό ανοιχτού κώδικα, με άδεια χρήσης υπό την άδεια GNU GPLv2 και διατηρείται από μια παγκόσμια κοινότητα.

Σκοπός της προσομοίωσης είναι να δοκιμάσει σε εικονική λειτουργία τα παρακάτω μοντέλα:

- Restricted Access Window (RAW) with interface for dynamic configuration
- Traffic Indication Map (TIM) segmentation
- Energy consumption model
- Adaptive Modulation and Coding Scheme (MCS)

4.1 Λογισμικό ns-3

Αρχικά έγινε λήψη του λογισμικού από την επίσημη ιστοσελίδα: https://www.nsnam.org/



Εικόνα 21- Επίσημος ιστότοπος για την λήψη του λογισμικού δικτύων ns-3

4.2 Κύρια ιδέα της προσομοίωσης

Η προσομοίωση που πραγματοποιήθηκε βασίστηκε στην εργασία των TIAN ET AL. (2018) με τίτλο “Extension of the IEEE 802.11ah ns-3 Simulation.” Οι ερευνητές της συγκεκριμένης εργασίας προτείνουν μια βελτιωμένη έκδοση του πρωτοκόλλου IEEE 802.11ah.

Σκοπός της προσομοίωσης είναι να αξιολογήσουμε έπειτα από δοκιμές προσομοίωσης του πρωτοκόλλου IEEE 802.11ah. Για τον λόγο αυτό θα γίνει δοκιμή με διάφορα σενάρια ώστε να δούμε αν ο αριθμός των αισθητήρων στο δίκτυο επηρεάζει την απόδοση του πρωτοκόλλου. Έτσι η δοκιμή θα γίνει για διαφορετικό αριθμό σταθμών.

Η αξιολόγηση του πρωτοκόλλου IEEE 802.11ah σε τέσσερα σενάρια:

(α) ένα σενάριο παρακολούθησης IoT όπου ένας μεγάλος αριθμός αισθητήρων που τροφοδοτούνται από μπαταρία με ετερογενείς απαιτήσεις κυκλοφορίας στέλνουν μετρήσεις σε έναν διακομιστή υποστήριξης

(β) ένας μεμονωμένος κινητός σταθμός αισθητήρων που στέλνει συνεχώς πακέτα UDP στο AP

(γ) ένα σενάριο ροής βίντεο που χρειάζεται σταθερή υψηλή απόδοση και

(δ) ένα σενάριο ελέγχου κλειστού βρόχου IoT όπου ένας μεγάλος αριθμός αισθητήρων και ενεργοποιητών που τροφοδοτούνται από μπαταρία χρησιμοποιούν αμφίδρομη επικοινωνία με τους αντίστοιχους ελεγκτές τους.

Όλα τα σενάρια εξετάζουν ένα μόνο AP. Οι προεπιλεγμένες παράμετροι του επιπέδου PHY και MAC παρουσιάζονται στον Πίνακα 3. Να σημειωθεί ότι ορισμένες παράμετροι MAC λαμβάνουν διαφορετικές τιμές σε διαφορετικά σενάρια, οπότε προσδιορίζονται επιπλέον για κάθε μεμονωμένο σενάριο. Αξιολογούμε την απόδοση του E-TAROA [11] στο σενάριο (α).

Προσομοιώνονται τρία διαφορετικά συνολικά φορτία κυκλοφορίας T = {0,095, 0,11, 0,15} Mbps και 5 διαφορετικούς αριθμούς σταθμών

128 σταθμούς

512 σταθμούς

1024 σταθμούς

2048 σταθμούς

4096 σταθμούς

Όλοι οι σταθμοί χρησιμοποιούν MCS1 με πλάτος καναλιού 1 MHz και έχουν διαφορετικά διαστήματα μετάδοσης, ακολουθώντας ομοιόμορφη κατανομή και η αναλογία μεταξύ οποιουδήποτε διαστήματος μετάδοσης δύο αισθητήρων σε οποιοδήποτε πείραμα είναι μικρότερη από 20.

Οι σταθμοί τοποθετούνται τυχαία γύρω από το AP σε έναν κύκλο 450 m με την παρουσία κρυφών κόμβων. Η παροχή υπολογίζεται ως ο μέσος αριθμός των byte ωφέλιμου φορτίου που ελήφθησαν με επιτυχία από το AP ανά δευτερόλεπτο. Για να λάβουμε την κατανάλωση ενέργειας, μετρήσαμε τον συνολικό χρόνο εκπομπής, χρόνο λήψης και χρόνο ύπνου για κάθε προσομοιωμένο σταθμό.

Στην κατάσταση SLEEP, το ραδιόφωνο είναι απενεργοποιημένο στο επίπεδο PHY και η ενέργεια που καταναλώνεται είναι αμελητέα (δηλαδή, υποτίθεται ότι είναι 0). Η κατανάλωση ενέργειας για τις άλλες πολιτείες βασίζεται σε μετρήσεις που αναφέρθηκαν από τους Ba et al. [2] για το υλικό πομποδέκτη IEEE 802.11ah. Συγκεκριμένα, χρησιμοποιούμε ισχύ TX 7,2 mW και ισχύ RX/IDLE 4,4 mW.

Το σενάριο (β) χρησιμοποιείται για την αξιολόγηση του προσαρμοστικού MCS όπου ένας κινητός σταθμός μεταδίδει συνεχώς πακέτα UDP 64 byte στο AP, ενώ μετακινείται από -500 m σε 500 m σε σχέση με το κεντρικό AP με ταχύτητα 1 m/s χρησιμοποιώντας το ConstantVelocityMobilityModel. Εκτός από τα υποχρεωτικά πλάτη καναλιών 1 και 2 MHz, τόσο ο σταθμός όσο και το AP υποστηρίζουν επίσης το κανάλι 4 MHz.

Αξιολογήσαμε την αξιοπιστία, το jitter, τον χρόνο μετ' επιστροφής (RTT) και την απόδοση με αμφίδρομη κίνηση στα σενάρια (γ) και (δ).

Στο σενάριο ελέγχου κλειστού βρόχου, εξετάζουμε τυχαία κατανεμημένα ζεύγη κόμβων αισθητήρα/ενεργοποιητή και κόμβους ελεγκτή που έχουν μοντελοποιηθεί ως πελάτες και διακομιστές περιορισμένης εφαρμογής πρωτοκόλλου (CoAP) που βρίσκονται σε απόσταση έως και 500 m από το AP, όπου όλοι οι βρόχοι ελέγχου έχουν ομοιόμορφη δυναμική. Απενεργοποιούμε το όριο μεταξύ θυρίδων (CSB) και αξιολογούμε την ευαισθησία στην τμηματοποίηση TIM, χωρίς να χρησιμοποιούμε RAW. Στο σενάριο ροής βίντεο, οι σταθμοί αντιπροσωπεύουν κάμερες κίνησης IP που μεταδίδουν ροή με ανίχνευση κίνησης. Για να διασφαλίσουμε τη ροή, ορίσαμε την πιθανότητα κίνησης σε 1. Εάν εντοπιστεί κίνηση, οι κάμερες μεταδίδουν ροή για 10 δευτερόλεπτα με καθορισμένο ρυθμό δεδομένων.

Στο επίπεδο MAC, διαμορφώνεται μια μεμονωμένη ομάδα RAW ανά διάστημα φάρου, με τη διάρκεια RAW να μεγιστοποιείται εντός ενός διαστήματος beacon.

Χρησιμοποιούμε μη CSB, διαφοροποιούμε το MCS, τον αριθμό των θυρίδων και τη διάρκεια της υποδοχής. Επίσης, διαφοροποιούμε τις παραμέτρους διαμόρφωσης τμηματοποίησης TIM, δηλαδή το μήκος του τμήματος σελίδας και τον αριθμό των τμημάτων σελίδας που αντιστοιχούν στην περίοδο DTIM. Η μετατόπιση μπλοκ και η μετατόπιση TIM ορίζονται στο μηδέν και ένα διάστημα DTIM αντιστοιχεί σε μία σελίδα.

4.3 Υλοποίηση προσομοίωσης

Parameter	**Value**
Frequency	900 MHz
Transmission power	0 dBm
Transmission gain	0 dB
Reception gain	3 dB
Noise Figure	3 dB
Coding method	BCC
Propagation loss model	Outdoor, macro [4]
Error Rate Model	YansErrorRate
CWmin	15
CWmax	1023
Traffic access categories	AC_BE
Payload size	64 bytes
Cross Slot Boundary (CSB)	enabled
MAC header type	legacy header
Beacon Interval	0.1 s
Station distribution	random
Rate control algorithm	constant

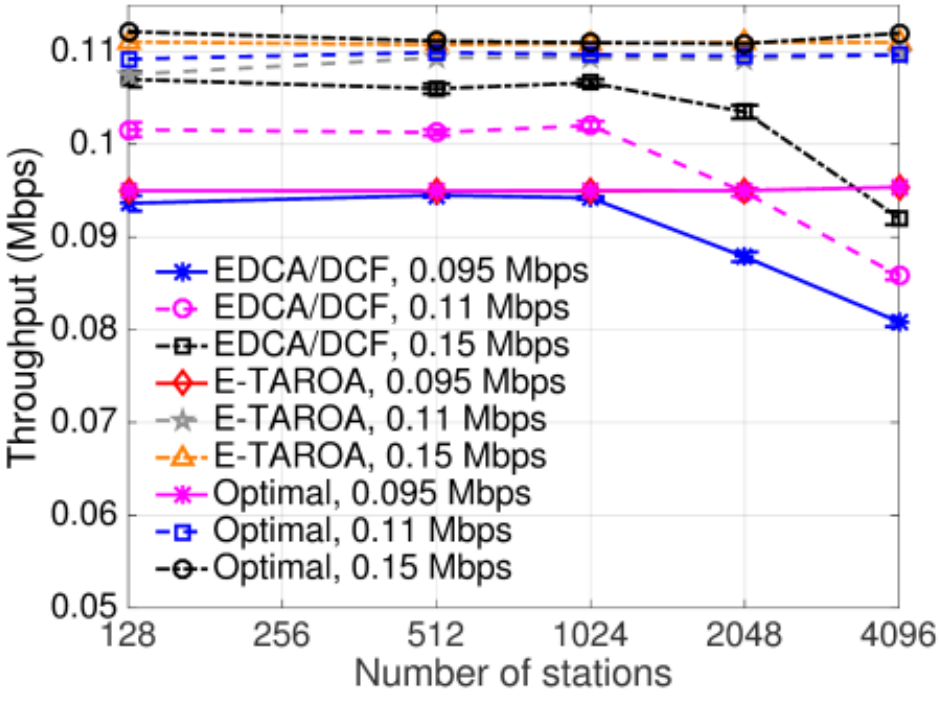
Εικόνα 22- Παράμετροι Προσομοίωσης

Αξιολόγηση Απόδοσης

Αλγόριθμοι βελτιστοποίησης RAW

Σε ρεαλιστικά σενάρια, το AP συνήθως δεν έχει προηγούμενη γνώση για την κίνηση κάθε σταθμού. Προκειμένου να ομαδοποιηθούν αποτελεσματικά οι σταθμοί, το AP πρέπει να εκτιμήσει τα μοτίβα κυκλοφορίας με βάση τις πληροφορίες μετάδοσης πακέτων. Επομένως, προτείναμε το ETAROA, το οποίο αλληλεπιδρά με το AP μέσω της διεπαφής διαμόρφωσης RAW. Εκτιμά τις συνθήκες δικτύου σε κάθε διάστημα beacon με βάση τις πληροφορίες δικτύου που λαμβάνονται από το AP και ενημερώνει τις παραμέτρους RAW (βλ. Πίνακας 1).

Συγκρίνουμε την απόδοση του E-TAROA, της «βέλτιστης» ομαδοποίησης και της μεθόδου πρόσβασης στα κανάλια παλαιού τύπου EDCA/DCF. Η «μέθοδος βέλτιστης ομαδοποίησης υποθέτει ότι το AP γνωρίζει την κίνηση κάθε σταθμού, επιτρέποντας στο AP να ομαδοποιήσει τους σταθμούς με τον βέλτιστο τρόπο. Τα αποτελέσματα φαίνονται στην εικόνα 23.



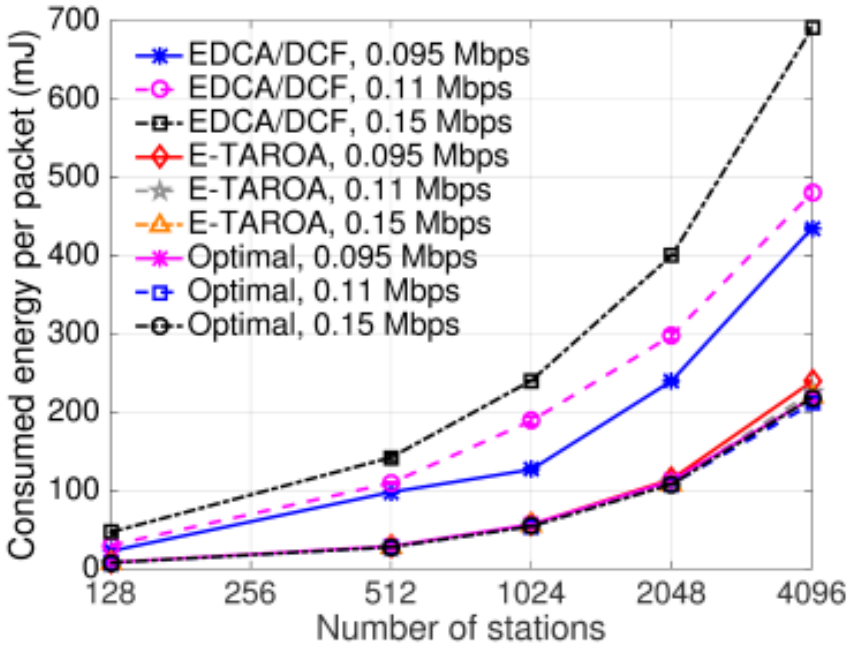
Εικόνα 23- Σύγκριση απόδοσης μεταξύ E-TAROA, βέλτιστη» και EDCA/DCF για διαφορετικά φορτία κυκλοφορίας και αριθμό σταθμών

Καθώς η μέθοδος «βέλτιστη» υποθέτει ότι το AP γνωρίζει εκ των προτέρων το διάστημα μετάδοσης κάθε κόμβου, επιτυγχάνει την υψηλότερη απόδοση. Ενώ το E-TAROA είναι πιο ρεαλιστικό, η επιτυγχανόμενη απόδοση είναι αρκετά κοντά στο «βέλτιστο» και κλιμακώνεται πολύ καλύτερα από το EDCA/DCF σε πυκνά δίκτυα. Για φορτίο κυκλοφορίας 0,095 Mbps, το E-TAROA έχει παρόμοια απόδοση με το EDCA/DCF για 128 έως 1024 σταθμούς, ενώ η διεκπεραίωση του EDCA/DCF πέφτει στα 0,08 Mbps με 4096 σταθμούς. Για φορτίο κυκλοφορίας 0,11 Mbps και 0,015 Mbps, η απόδοση του E-TAROA παραμένει σχεδόν η ίδια για όλους τους διαφορετικούς αριθμούς σταθμών.

Αντίθετα, το EDCA/DCF επιτυγχάνει περίπου 6% και 4% λιγότερη απόδοση για 128 σταθμούς και 22% και 17% λιγότερη για 4096 σταθμούς. Αυτά τα αποτελέσματα δείχνουν τα πλεονεκτήματα της βελτιστοποίησης RAW σε πραγματικό χρόνο, καθώς και την ικανότητα του προσομοιωτή να υποστηρίζει δυναμικές αλλαγές διαμόρφωσης RAW κατά τη διάρκεια της προσομοίωσης.

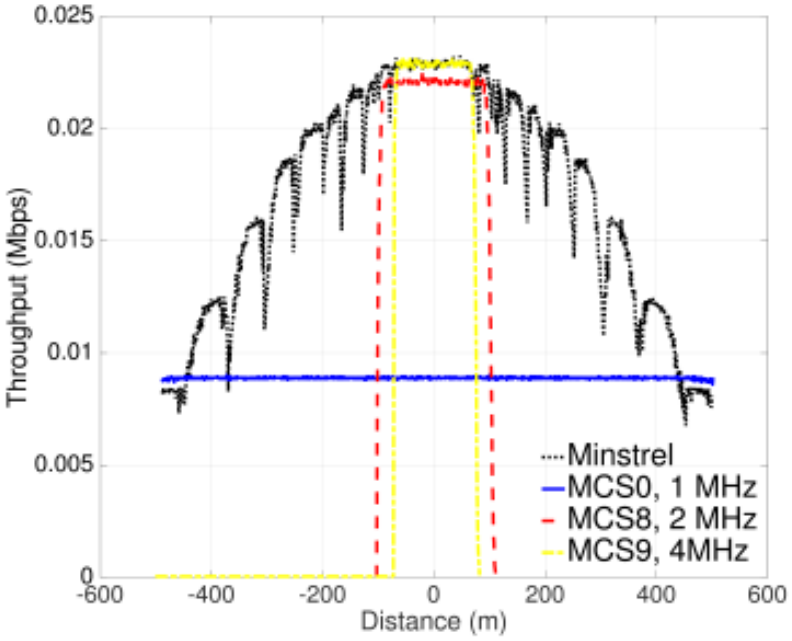
Κατανάλωση Ενέργειας

Σε αυτήν την ενότητα, αξιολογούμε την κατανάλωση ενέργειας του E-TAROA και του EDCA/DCF χρησιμοποιώντας τις εφαρμοσμένες μεταβάσεις ενεργειακής κατάστασης και το μοντέλο κατανάλωσης. Συγκεκριμένα, εξετάζονται τέσσερις καταστάσεις: TX, RX, IDLE και SLEEP. Η ενέργεια που καταναλώνεται από έναν σταθμό IEEE 802.11ah λαμβάνεται πολλαπλασιάζοντας τον χρόνο που ξοδεύει η συσκευή σε κάθε συγκεκριμένη κατάσταση με την αντίστοιχη κατανάλωση ενέργειας αυτής της κατάστασης. Ο άξονας X του εικόνας 24 εμφανίζει τον αριθμό των σταθμών, ενώ ο άξονας Y απεικονίζει την καταναλωμένη ενέργεια (σε mJ).



Εικόνα 24- Σύγκριση κατανάλωσης ενέργειας : Σύγκριση κατανάλωσης ενέργειας μεταξύ E-TAROA, «βέλτιστη» και EDCA/DCF για διαφορετικά φορτία κυκλοφορίας και αριθμό σταθμών

Η κατανάλωση ενέργειας του EDCA/DCF είναι σαφώς υψηλότερη από αυτή του E-TAROA και της βέλτιστης ομαδοποίησης RAW. Αυτό οφείλεται στην ομαδοποίηση των σταθμών, η οποία επιτρέπει την καλύτερη διαχείριση της πρόσβασης στα κανάλια και ο σταθμός μπορεί να εισέλθει σε κατάσταση αναστολής λειτουργίας όταν δεν βρίσκεται στην υποδοχή του ή δεν έχει πακέτα για μετάδοση. Όταν χρησιμοποιείτε το EDCA/DCF, οι σταθμοί περνούν πολύ χρόνο σε κατάσταση αδράνειας κατά τη διάρκεια της επιστροφής, ειδικά σε περιβάλλοντα με μεγαλύτερη πυκνότητα και κίνηση.



Εικόνα 25- ***εύρος για το MinstrelWifiManager και το ConstantRateWifiManager που χρησιμοποιούν διαφορετικά πλάτη MCS και καναλιών, ως συνάρτηση της απόστασης από το AP***

Adaptive MCS

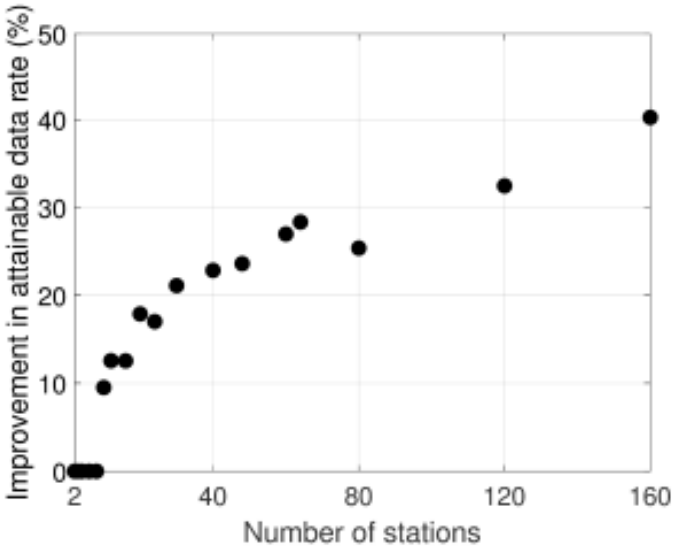
Σε αυτήν την ενότητα, επιλέγουμε το MinstrelWifiManager για να αξιολογήσουμε τις δυνατότητες προσαρμογής MCS. Το συγκρίνουμε με σταθερό MCS χρησιμοποιώντας το ConstantWifiManager. Όπως φαίνεται στην εικόνα 13 , το MinstrelWifiManager μπορεί να προσαρμόσει το MCS και το πλάτος του καναλιού όταν αλλάζει η απόσταση μεταξύ του σταθμού και του AP, προσφέροντας μια δυναμική ανταλλαγή μεταξύ της απόστασης και της απόδοσης.

Αντίθετα, το ConstantWifiManger είναι κατάλληλο μόνο για μια συγκεκριμένη απόσταση. Για MCS9, 4 MHz, μπορεί να επιτύχει σχεδόν την ίδια απόδοση (δηλαδή 0,22 Mbps) με το MinstrelWifiManager για αποστάσεις έως και 70 m. Για μεγαλύτερες αποστάσεις, ο σταθμός δεν μπορεί καν να συσχετιστεί με το AP. Για MCS8 με πλάτος καναλιού 2 MHz, ο σταθμός μπορεί να επικοινωνεί με AP σε απόσταση έως και 100 m, με κόστος ελαφρώς χαμηλότερης απόδοσης. Για MCS0 με πλάτος καναλιού 1 MHz, ο σταθμός έχει την ίδια κάλυψη όπως όταν χρησιμοποιείται το MinstrelWifiManager, αλλά η επιτυγχανόμενη απόδοση είναι κατά μέσο όρο πολύ χαμηλότερη. Αυτά τα

αποτελέσματα δείχνουν ξεκάθαρα την ικανότητα του σταθμού να προσαρμόζει δυναμικά το MCS του με βάση την τρέχουσα ισχύ του σήματος.

Κυκλοφορία αμφίδρομης κατεύθυνσης

Στο σενάριο ροής βίντεο, αποδείξαμε τη σκοπιμότητα αξιόπιστων και υψηλής απόδοσης εφαρμογών, όπως η ροή κάμερας IP και οι ενημερώσεις υλικολογισμικού over-the-air χρησιμοποιώντας κυκλοφορία παλαιού τύπου Διαδικτύου (TCP/IP) για έως και 20 σταθμούς πάνω από 200 m με απόδοση 160 − 256 kbps κατά μέσο όρο ανά σταθμό. Αξιολογούμε την αντιστάθμιση μεταξύ του μέγιστου επιτεύξιμου ρυθμού δεδομένων ανά σταθμό και του συνολικού αριθμού καμερών στο δίκτυο [8].

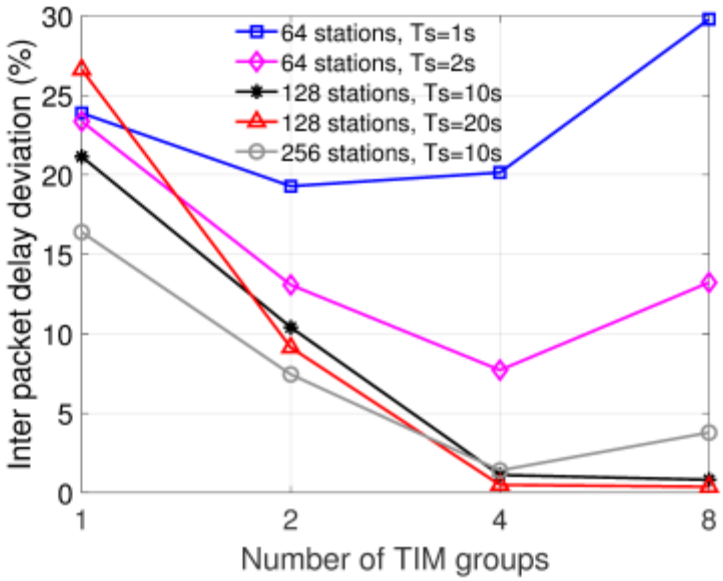


Εικόνα 26- Βελτίωση Μετάδοσης

Όπως δείχνει η εικόνα 27, ήδη για 128 σταθμούς, η κατάλληλη διαμόρφωση TIM και RAW μπορεί να επιτύχει 10 % υψηλότερο ρυθμό δεδομένων από το EDCA/DCF. Για κάμερες 10–80, η βελτίωση στο μέγιστο δυνατό ρυθμό δεδομένων ανέρχεται σε 10–30 % και αυξάνεται περαιτέρω για πυκνότερα δίκτυα. Στο σενάριο ελέγχου κλειστού βρόχου εστιάζουμε στις βέλτιστες διαμορφώσεις RAW και TIM που μπορούν να εγγυηθούν RTT < Ts , όπου το Ts αντιπροσωπεύει την περίοδο δειγματοληψίας μιας απόκρισης συστήματος. Λόγω της διαχείρισης εξοικονόμησης ενέργειας του 802.11ah, το RTT μπορεί να αυξηθεί σημαντικά, δεδομένου ότι ένα ταξίδι μετ' επιστροφής σε

κλειστό βρόχο συνήθως απαιτεί 4 άλματα για ένα μόνο πακέτο, όπου μόνο ένα άλμα ανά περίοδο DTIM μπορεί να πραγματοποιηθεί σύμφωνα με το πρότυπο. Επιπλέον, η αυξανόμενη ευαισθησία στην τμηματοποίηση TIM αυξάνει το ελάχιστο δυνατό RTT και απενεργοποιεί τη χρήση βρόχων ελέγχου με γρήγορη δυναμική.

Σε έναν βρόχο ελέγχου, η καθυστέρηση μεταξύ πακέτων στον αισθητήρα/ενεργοποιητή πρέπει να είναι περίπου σταθερή και μικρότερη από το Ts για να μην αποσταθεροποιηθεί ο βρόχος. Οι προσομοιώσεις μας δείχνουν πώς η καθυστέρηση μεταξύ των πακέτων παραμένει περίπου σταθερή μέχρι το σημείο κορεσμού που εξαρτάται από την πυκνότητα και τη δυναμική του βρόχου, αλλά η αύξηση της ευαισθησίας στην τμηματοποίηση TIM πέρα από αυτό το σημείο οδηγεί σε απόκλιση καθυστέρησης μεταξύ πακέτων όπως φαίνεται στην εικόνα 27.



Εικόνα 27- Η τμηματοποίηση TIM βελτιώνει την απόκλιση καθυστέρησης μεταξύ των πακέτων μέχρι ένα σημείο που εξαρτάται από την πυκνότητα του δικτύου και τη δυναμική των βρόχων ελέγχου

Ο κώδικας στην γλώσσα Python για τα παραπάνω αποτελέσματα είναι ο εξής:

Αρχικά γίνεται εισαγωγή των απαραίτητων βιβλιοθηκών

```
Import
os,
sys,
inspec
t
```

```
VERSION="1.8.19"
REVISION="b1fc8f7baef51bd2db4c2971909a568d"
GIT="22213cd8abbd141bda40667f7ca2a48f2d6ad785"
INSTALL=''
C1='#5'
C2='#/'
C3='#,'
cwd = os.getcwd()
join = os.path.join




WAF='waf'
def b(x):
	return x
if sys.hexversion>0x300000f:
	WAF='waf3'
	def b(x):
		return x.encode()


def err(m):
	print(('\033[91mError: %s\033[0m' % m))
	sys.exit(1)


def unpack_wafdir(dir, src):
	f = open(src,'rb')
	c = 'corrupt archive (%d)'
	while 1:
		line = f.readline()
		if not line: err('run waf-light from a folder containing
waflib')
		if line == b('#==>\n'):
			txt = f.readline()
			if not txt: err(c % 1)
			if f.readline() != b('#<==\n'): err(c % 2)
			break
	if not txt: err(c % 3)
	txt = txt[1:-1].replace(b(C1), b('\n')).replace(b(C2),
b('\r')).replace(b(C3), b('\x00'))
```

```
	import shutil, tarfile
	try: shutil.rmtree(dir)
	except OSError: pass
	try:
		for x in ('Tools', 'extras'):
			os.makedirs(join(dir, 'waflib', x))
	except OSError:
		err("Cannot unpack waf lib into %s\nMove waf in a writable
directory" % dir)


	os.chdir(dir)
	tmp = 't.bz2'
	t = open(tmp,'wb')
	try: t.write(txt)
	finally: t.close()


	try:
		t = tarfile.open(tmp)
	except:
		try:
			os.system('bunzip2 t.bz2')
			t = tarfile.open('t')
			tmp = 't'
		except:
			os.chdir(cwd)
			try: shutil.rmtree(dir)
			except OSError: pass
			err("Waf cannot be unpacked, check that bzip2
support is present")


	try:
		for x in t: t.extract(x)
	finally:
		t.close()


	for x in ('Tools', 'extras'):
		os.chmod(join('waflib',x), 493)


	if sys.hexversion<0x300000f:
		sys.path = [join(dir, 'waflib')] + sys.path
		import fixpy2
```

```
                fixpy2.fixdir(dir)


        os.remove(tmp)
        os.chdir(cwd)


        try: dir = unicode(dir, 'mbcs')
        except: pass
        try:
                from ctypes import windll
                windll.kernel32.SetFileAttributesW(dir, 2)
        except:
                pass


def test(dir):
        try:
                os.stat(join(dir, 'waflib'))
                return os.path.abspath(dir)
        except OSError:
                pass


def find_lib():
        src = os.path.abspath(inspect.getfile(inspect.getmodule(err)))
        base, name = os.path.split(src)


        #devs use $WAFDIR
        w=test(os.environ.get('WAFDIR', ''))
        if w: return w


        #waf-light
        if name.endswith('waf-light'):
                w = test(base)
                if w: return w
                err('waf-light requires waflib -> export WAFDIR=/folder')


        dirname = '%s-%s-%s' % (WAF, VERSION, REVISION)
        for i in (INSTALL,'/usr','/usr/local','/opt'):
                w = test(i + '/lib/' + dirname)
                if w: return w
```

```
        #waf-local
        dir = join(base, (sys.platform != 'win32' and '.' or '') + dirname)
        w = test(dir)
        if w: return w


        #unpack
        unpack_wafdir(dir, src)
        return dir


wafdir = find_lib()
sys.path.insert(0, wafdir)


if __name__ == '__main__':


        from waflib import Scripting
        Scripting.waf_entry_point(cwd, VERSION, wafdir)
```

5. Προσομοίωση Φυσικού Επιπέδου με χρήση του πρωτοκόλλου IEEE 802.11ah (IEEE 802.11ah PHY Simulator)

Σε αυτό το μέρος, περιγράφονται συνοπτικά τα κύρια τμήματα του μπλοκ του προσομοιωτή IEEE 802.11ah PHY. Εάν δεν περιγράφεται μετάλλευμα αναφέρεται, τότε το παρουσιαζόμενο λειτουργικό μπλοκ υλοποιείται στην κύρια συνάρτηση m του προσομοιωτή (main_802_11ah.m). Τα κύρια μπλοκ στην αλυσίδα επεξεργασίας σήματος του τμήματος πομπού (TX) περιγράφονται αναλυτικά στην επόμενη ενότητα. Η προσομοίωση πραγματοποιήθηκε στο λογισμικό του MATLAB.

5.1 Εισαγωγή δεδομένων Δημιουργία τυχαίων ροών bit (bit).

Scrambler

Χρησιμοποιείται για τη διάσπαση μεγάλων ακολουθιών μονάδων και μηδενικών. Τα δεδομένα κωδικοποιούνται από ένα σήμα που ορίζεται με το πολυώνυμο της γεννήτριας x7+x4+1. Καλείται από τη συνάρτηση scrambler.m.

Συνελικτικός κωδικοποιητής

Το σχήμα Διόρθωσης Προώθησης Σφάλματος (FEC) για το IEEE 802.11ah, σύμφωνα με τα καθορισμένα σχήματα διαμόρφωσης και κωδικοποίησης (MCS), χρησιμοποιεί 4 ρυθμούς κωδικών: 1/2, 2/3, 3/4 και 5/6. Στον προσομοιωτή IEEE 802.11ah PHY, μπορεί να χρησιμοποιηθεί μόνο συνελικτικός κωδικοποιητής (που δημιουργείται από πολυωνυμικούς συντελεστές G0 = 133 και G1 = 177). Ο κώδικας Low-Density Paity-Check (LDPC) δεν υποστηρίζεται στην τρέχουσα έκδοση του προσομοιωτή.

Interleaver Block Interleaver (στην αλυσίδα επεξεργασίας σήματος TX) χρησιμοποιείται για την παρεμβολή του κωδικοποιημένου μηνύματος (σε επίπεδο bit). Το μισό αυτού του μπλοκ είναι ενσωματωμένο στο κύριο m-αρχείο του προσομοιωτή IEEE 802.11ah PHY (main_802_11ah.m). Το δεύτερο μισό του καλείται από τη συνάρτηση longRunsIntrlv.m.

Χαρτογράφηση αστερισμών

Σύμφωνα με τα καθορισμένα MCS, χρησιμοποιούνται διαμορφώσεις BPSK, QPSK, 16-QAM, 64-QAM και 256-QAM.

Space-Time-Block Coding (STBC)

Χρησιμοποιείται για την πραγματοποίηση της λειτουργίας μετάδοσης MIMO (με τη χρήση της τεχνικής του Alamouti). Η τρέχουσα έκδοση του προσομοιωτή υποστηρίζει μόνο τη χρήση ενός σχήματος MIMO 2 x 2.

Τα μπλοκ OFDM, Pilot Insertion, IFFT και Cyclic Prefix (CP) OFDM, Pilot Insertion, IFFT and CP Insertion διασφαλίζουν τη δημιουργία των συμβόλων OFDM που περιέχουν χαρτογραφημένα δεδομένα και πιλότους, τη μετατροπή του τομέα συχνότητας σε χρόνο και την επέκτασή τους με κυκλικό πρόθεμα (CP), αντίστοιχα. Το CP εφαρμόζεται για την παράκαμψη της παρεμβολής μεταξύ συμβόλων (ISI) που προκαλείται από τη διάδοση πολλαπλών διαδρομών. Είναι δυνατή η επιλογή μεταξύ κανονικού και μικρού μήκους CP. Στο τέλος αυτής της διαδικασίας, δημιουργείται το σήμα ζώνης βάσης IEEE 802.11ah.

Το Channel Block Channel επιτρέπει στον χρήστη του προσομοιωτή να επιλέξει μεταξύ των παρακάτω μοντέλων καναλιών.

Μοντέλο καναλιού AWGN ('AWGN')

Συνιστάται για χρήση για προσομοιώσεις αναφοράς Ρικιανό κανάλι με 20 ανεξάρτητες διαδρομές ('Rician') (συνάρτηση Channel.m) Κανάλι Rayleigh με 20 ανεξάρτητες διαδρομές ('Rayleigh') (συνάρτηση Channel.m) Σημείωση (το τμήμα RX του προσομοιωτή IEEE 802.11ah PHY): Το τμήμα RX του προσομοιωτή IEEE 802.11ah έχει παρόμοια διαμόρφωση με το TX, αλλά η επεξεργασία του σήματος είναι αντίστροφη.

Πρέπει να σημειωθεί ότι η τεχνική εξισορρόπησης Zero-Forcing (ZF) χρησιμοποιείται για την εκτίμηση καναλιών, όταν χρησιμοποιούνται μοντέλα διαλείψεων καναλιών στις προσομοιώσεις. Διαφορετικά (στην περίπτωση του καναλιού AWGN), δεν χρησιμοποιείται η τεχνική εξισορρόπησης ZF.

Για την προσομοίωση δόθηκαν οι κατάλληλες παράμετροι:

Η όλη προσομοίωση, μετά το σύνολο των παραμέτρων εισαγωγής, μπορεί να εκτελεστεί από το αρχείο 'main_802_11ah.m'. Οι παράμετροι που μπορούν να ρυθμιστούν σε αυτή τη συνάρτηση (στην αρχή) είναι οι εξής:

'channelWidth' – εύρος ζώνης σήματος {1, 2, 4, 8, 16}–MHz

«param» – η επιλογή της διαμόρφωσης (εξαρτάται επίσης από το επιλεγμένο σχήμα SISO/MISO – που ορίζεται από την παράμετρο «μοντέλο») – για SISO [0: BPSK, 1,2: QPSK, 3,4: 16QAM, 5,6,7: 64QAM, 8,9: 256QAM, 10: BPSK 2x επανάληψη] και για MIMO [0: BPSK, 1,2: QPSK] – στην περίπτωση του MIMO, η τρέχουσα έκδοση του προσομοιωτή υποστηρίζει μόνο τη χρήση των διαμορφώσεων BPSK και QPSK

«GI» – κυκλικό πρόθεμα (διάστημα προστασίας) – μπορεί να είναι «κανονικό» και «σύντομο»

«model» – το τρόπος μετάδοσης – μπορεί να είναι «SISO» και «MIMO»

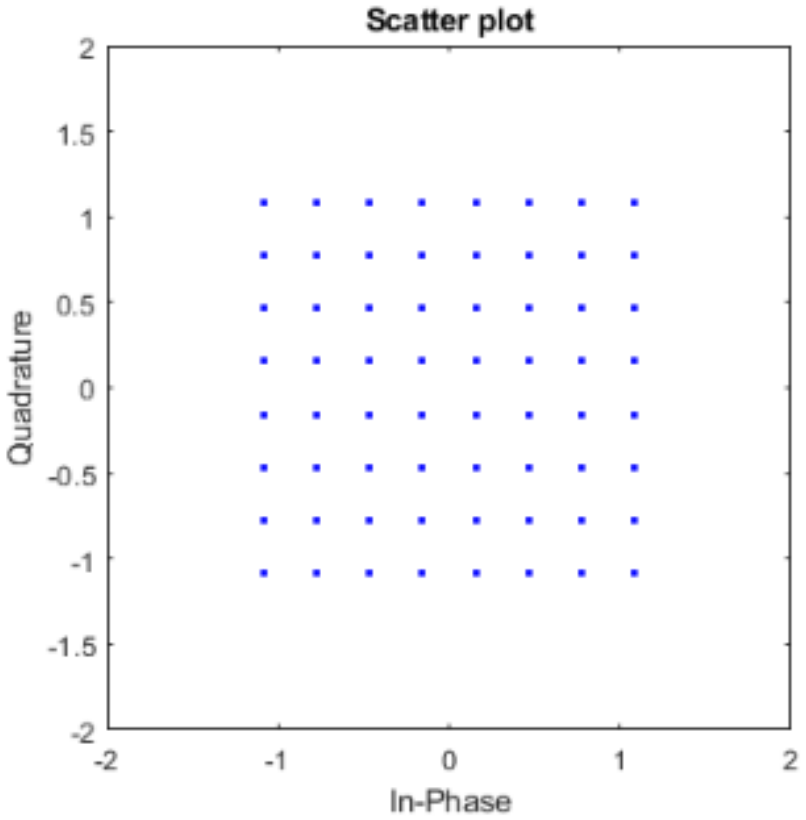
«CNR» – Τιμές CNR/SNR (σε dB) «channelType» – τα χρησιμοποιούμενα μοντέλα καναλιών: «AWGN», «Rician», «Rayleigh»

«useEkvaliz» – το χρήση της εξισορρόπησης: «0» – είναι OFF, «1» – είναι ON «fc» και

«fs» – ο ορισμός του φορέα και της συχνότητας δειγματοληψίας (και τα δύο σε Hz), αντίστοιχα

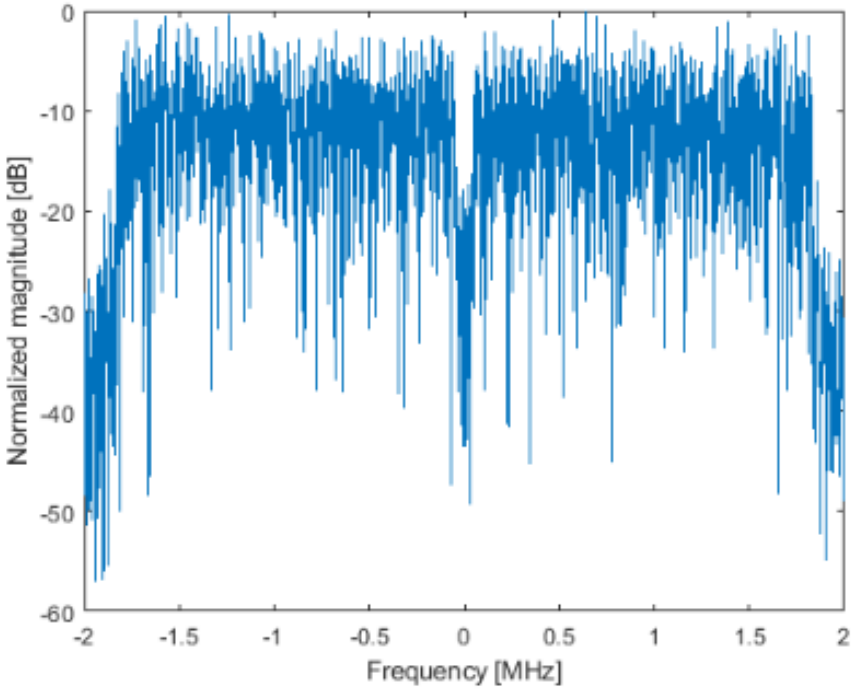
Αξιολόγηση των αποτελεσμάτων

Η έξοδος των προσομοιώσεων (block Evaluation of the results) μπορεί να αξιολογηθεί με βάση το Bit Error Ration - BER (πριν και μετά την αποκωδικοποίηση FEC) και το Modulation Error Ratio (MER) ανάλογα με την τιμή σήματος προς θόρυβο αναλογία (SNR).



Εικόνα 28- Αποτέλεσμα προσομοίωσης φυσικού επιπέδου

Επιπλέον, σχεδιάζονται διαγράμματα αστερισμών των εκπεμπόμενων και λαμβανόμενων (σε περίπτωση εξισορρόπησης – μετά την εξίσωση) σημάτων και κανονικοποιημένου φάσματος του λαμβανόμενου σήματος.



Εικόνα 29- Αποτέλεσμα προσομοίωσης φυσικού επιπέδου

Συμπεράσματα

Με την ανάπτυξη του Διαδικτύου των Πραγμάτων (IoT), η κατανομή πόρων καναλιού για ένα δίκτυο μεγάλης κλίμακας κατέστη απαραίτητη. Οι τεχνολογίες δικτύωσης ευρείας περιοχής χαμηλής ισχύος (LP-WAN) έχουν χρησιμοποιηθεί για τη διαχείριση της επικοινωνίας ενός τεράστιου αριθμού συσκευών IoT με περιορισμένη ενέργεια. Οι τεχνολογίες LP-WAN έχουν επιτρέψει εκτεταμένη εμβέλεια επικοινωνίας έως και αρκετά χιλιόμετρα με χαμηλή κατανάλωση ενέργειας. Οι SigFox και LoRaWAN είναι οι δύο πιο υιοθετημένες λύσεις για το LP-WAN. Με αυστηρούς περιορισμούς στα εξαιρετικά στενής ζώνης σήματα, το μέγιστο ωφέλιμο φορτίο πακέτων, τον αριθμό πακέτων ανά συσκευή ανά ημέρα, το SigFox δεν είναι πολύ ευέλικτο και ανοιχτό.

Επιπλέον, το δίκτυο SigFox είναι ένα ιδιόκτητο δίκτυο που βασίζεται σε χειριστή και δεν είναι δυνατό για ανεξάρτητες αναπτύξεις. Το LoRaWAN, από την άλλη πλευρά, υποστηρίζεται ότι είναι μια από τις πιο επιτυχημένες τεχνολογίες του LP-WAN. Ωστόσο, εξακολουθεί να θέτει περιορισμούς όσον αφορά το μέγεθος και τη χωρητικότητα του δικτύου. Το μέγεθος δικτύου του LoRaWAN περιορίζεται από τους κανονισμούς κύκλου λειτουργίας στις ζώνες ISM. Η χωρητικότητα του δικτύου LoRaWAN αποστραγγίζεται με τη διαδικασία εξασφάλισης αξιοπιστίας και πυκνότητας δικτύου.

Το 2017, κυκλοφόρησε το πρότυπο ασύρματης δικτύωσης IEEE 802.11ah για την αποτελεσματική υποστήριξη των εφαρμογών IoT. Συνδυάζοντας χαρακτηριστικά WiFi και LP-WAN, το πρότυπο IEEE 802.11ah υπόσχεται βελτίωση της επεκτασιμότητας του δικτύου, χαμηλή κατανάλωση ενέργειας, κάλυψη μεγάλης περιοχής και βελτίωση της απόδοσης δικτύου. Ενεργοποιώντας την ασύρματη επικοινωνία στις ζώνες ISM που εξαιρούνται από την άδεια χρήσης κάτω του 1 GHz, το IEEE 802.11ah MAC εφαρμόζει τεχνολογία επικοινωνίας χαμηλής ισχύος ορίζοντας ένα παράθυρο περιορισμένης πρόσβασης (RAW). Το RAW είναι ένα διάστημα που εκχωρείται από το Access Point (AP) για μια ομάδα Σταθμών Χάρτης Ενδείξεων Κυκλοφορίας (TIM STAs) για πρόσβαση στο μέσο εντός ενός σύντομου φάρου.

Υπάρχουν δύο τύποι μεταδόσεων στο RAW: Μετάδοση UpLink (UL) από TIM STA στο AP και DownLink (DL) μετάδοση από AP σε σελιδοποιημένα TIM STA. Αντίστοιχα στους δύο τύπους μετάδοσης, ένα RAW τοποθετείται σε σχισμή και χωρίζεται σε τμήματα UL και DL. Το AP διαχειρίζεται τα STA ομαδοποιώντας τα σε σελίδες, μπλοκ και υπομπλοκ. Σε κάθε STA εκχωρείται ένα μοναδικό αναγνωριστικό συσχέτισης (AID). Στο πρότυπο IEEE 802.11ah , χρησιμοποιούνται μόνο 11 λιγότερο σημαντικά bit ενός AID 16 bit, γεγονός που οδηγεί σε μέγιστο αριθμό 8191 υποστηριζόμενων STA ανά AP.

Οι προσομοιώσεις στα λογισμικά ns3 και Matlab ευελπιστούμε ότι βοήθησαν τον αναγνώστη να κατανοήσει καλύτερα τον τρόπο λειτουργίας του πρωτοκόλλου 802.11 ah. Οι προσομοιώσεις δείχνουν τα οφέλη και τα πλεονεκτήματα του πρωτοκόλλου για την επικοινωνία των συσκευών στο διαδίκτυο των πραγμάτων, εξοικονομώντας ενέργεια.

Συνοψίζοντας θα ήταν χρήσιμο να παρουσιάσουμε εν συντομία τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης του πρωτοκόλλου 802.11ah (WiFi HaLow):

➡ Καταναλώνει λιγότερη ενέργεια από την παραδοσιακή συσκευή Wi-Fi. Ως εκ τούτου, οι συσκευές Wi-Fi HaLow με μπαταρία σε σχήμα νομίσματος λειτουργούν για μήνες ή χρόνια.

➡Προσφέρει μεγαλύτερη εμβέλεια κάλυψης που είναι περίπου 1 Km. Αυτό είναι σχεδόν διπλάσιο από το παραδοσιακό wifi.

➡Το σήμα WiFi HaLow μπορεί εύκολα να διεισδύσει μέσα από τοίχους και άλλα εμπόδια, καθώς είναι χαμηλότερη σε αξία σε σύγκριση με την υψηλή συχνότητα (2,4 GHz/5 GHz) που χρησιμοποιείται από το παραδοσιακό δίκτυο Wi-Fi.

➡Δεν απαιτεί ιδιόκτητους κόμβους ή πύλες.

➡Προσφέρει αξιόπιστη ασύρματη σύνδεση λόγω της χρήσης λιγότερο συμφορημένης ζώνης συχνοτήτων, υψηλότερης ευαισθησίας και περιθωρίου σύνδεσης.

➡Το WiFi HaLow είναι ιδανικό για μικρά και εκρηκτικά δεδομένα που χρησιμοποιούνται για το IoT (Internet of Things) που δεν καταναλώνει περισσότερη ενέργεια και απαιτεί να ταξιδεύετε μεγαλύτερες αποστάσεις. Αυτό είναι χρήσιμο για

εφαρμογές έξυπνων κτιρίων, όπως έξυπνο φωτισμό, έξυπνη ασφάλεια, έξυπνη στάθμευση, έξυπνη πόλη, έξυπνο HVAC κ.λπ.

➡Το στρώμα MAC έχει σχεδιαστεί για να υποστηρίζει χιλιάδες συνδεδεμένες συσκευές.

Το WiFi HaLow μπορεί να υποστηρίξει έως και 8191 συσκευές με ένα μόνο AP (Σημείο Πρόσβασης).

➡Χρησιμοποιείται για εκφόρτωση κίνησης κινητής τηλεφωνίας λόγω της εκτεταμένης εμβέλειάς του σε εξωτερικούς χώρους.

➡Η συχνότητα υπο-GHz που χρησιμοποιείται στις 802,11h υποφέρει από χαμηλότερες παρεμβολές σε σύγκριση με τη ζώνη των 2,4 GHz. Μειονεκτήματα ή μειονεκτήματα του WiFi HaLow

Ακολουθούν τα μειονεκτήματα του WiFi HaLow:

➡Δεν υπάρχει παγκόσμιο πρότυπο για τη ζώνη των 900 MHz σε αντίθεση με τα 2,4 GHz που χρησιμοποιούνται παντού στον κόσμο. Επιπλέον, το WiFi HaLow είναι αμερικανικό. ➡Οι κεραίες στην περιοχή κάτω των GHz είναι μεγαλύτερες σε σύγκριση με το σύστημα WiFi 2,4 GHz.

Προτάσεις μελλοντικής επέκτασης της εργασίας

Συνοψίζοντας, οι εφαρμογές υπολογιστικού νέφους απαιτούν υψηλές αποδόσεις και τακτική καθυστέρηση, μέσω διασυνδεδεμένων δικτύων με απαιτήσεις QoS. Προκειμένου να ικανοποιηθούν οι απαιτήσεις των προαναφερθέντων σεναρίων, το IEEE 802.11ah μπορεί να χρησιμοποιηθεί ως βιώσιμη επιλογή συνδεσιμότητας σε συγκεκριμένες εφαρμογές, όπου τα χαρακτηριστικά του (μεγάλο εύρος κάλυψης, ρυθμός δεδομένων και αριθμός υποστηριζόμενων σταθμών) μπορούν να προσφέρουν καλύτερη προσέγγιση από άλλες τεχνολογίες IoT, και ταυτόχρονα πληρούν το QoS που απαιτείται για αυτού του είδους τις εφαρμογές.

Επιπλέον, τα σενάρια σχετικά με τα δίκτυα υπολογιστών M2M θα μπορούσαν να είναι μια άλλη διαδρομή για τη συνέχιση της μελλοντικής εργασίας αυτής της έρευνας, όπου οι επικοινωνίες δικτύων M2M χρησιμοποιούνται συνήθως ως αισθητήρες, ενεργοποιητές και μηχανές που υπολογίζουν και χρησιμοποιούν πόρους δικτύου στη μέση της διαδρομής μεταξύ πηγών δεδομένων και κέντρα δεδομένων cloud.

Το IEEE 802.11ah θα μπορούσε να είναι η κύρια τεχνολογία για την κάλυψη των απαιτήσεων συνδεσιμότητας υπολογιστών, χρησιμοποιώντας το ως βάση μεταξύ των ασύρματων δικτύων αισθητήρων και των κέντρων δεδομένων cloud, προκειμένου να ικανοποιηθούν τα σενάρια υπολογιστικών αιχμής. Από την άλλη πλευρά, μέσω αυτής της διατριβής, το IEEE 802.11ah έχει αποδείξει τις δυνατότητές του να υποστηρίζει και να εκπληρώνει τις απαιτήσεις εφαρμογής IoT. Δεδομένου ότι και τα δύο σενάρια περιπτώσεων χρήσης IoT έχουν παρόμοιες απαιτήσεις επικοινωνίας, επιδιώκουν να εκπληρώσουν τους ίδιους στόχους, με στόχο τη βελτίωση της ανθρώπινης ευημερίας.

Βιβλιογραφία

[1] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, “A survey on IEEE 802.11ah: An enabling networking technology for smart cities,” Comput. Commun., vol. 58, pp. 53–69, Mar. 2015. [2] “Wi-Fi Alliance.” [Online]. Available: https://www.wi-fi.org/. [Accessed: 21-Jun-2020].

[3] wi-fi.org, “Wi-Fi HaLow | Wi-Fi Alliance,” www.wi-fi.org, 2017. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-halow. [Accessed: 22-Jun-2020].

[4] 802.11ah-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems - Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1GHz License Exempt Operation, 30 April 2017, no. February, p. 594, 2017.

[5] Long Range Low Power (LRLP) Operation in 802.11: Use Cases and Functional Requirements: Guidelines for PAR Development. .

[6] E. Lopez-Aguilera, I. Demirkol, E. Garcia-Villegas, and J. Paradells, “IEEE 802.11-Enabled Wake-Up Radio: Use Cases and Applications,” Sensors, vol. 20, no. 1, p. 66, Dec. 2019. [7] IEEE Standards Association, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specificatio19ns; Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. 2013.

[8] “Wi-Fi CERTIFIED 6 | Wi-Fi Alliance.” [Online]. Available: https://www.wi-fi.org/discover-wifi/wi-fi-certified-6. [Accessed: 22-Jun-2020].

[9] “IEEE P802.11ax - IEEE Draft Standard for Information Technology -- Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks -- Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment Enhancements for High Efficiency WLAN.” [Online]. Available: https://standards-ieee-org.recursos.biblioteca.upc.edu/project/802_11ax.html. [Accessed: 25-Jul2020].

[10] M. S. Afaqui, E. Garcia-villegas, E. Lopez-aguilera, G. Smith, and D. Camps, “Evaluation of Dynamic Sensitivity Control Algorithm for IEEE 802 . 11ax,” no. Wcnc, pp. 1078–1083, 2015. 104

[11] “IEEE Std 802.11af-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, IEEE Std 802.11ad-2012, and IEEE Std 802.11ac2013),” IEEE Std 802.11af-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, IEEE Std 802.11ad-2012, and IEEE Std 802.11ac2013). pp. 1–198, 2014.

[12] LoRa Alliance, “LoRaWAN Backend Interfaces 1.0 Specification,” no. 2010, p. 97331, 2011.

[13] L. Casals, B. Mir, R. Vidal, and C. Gomez, “Modeling the Energy Performance of LoRaWAN,” Sensors, vol. 17, no. 10, p. 2364, Oct. 2017.

[14] C. Gomez, J. C. Veras, R. Vidal, L. Casals, and J. Paradells, “A Sigfox Energy Consumption Model,” Sensors, vol. 19, no. 3, p. 681, Feb. 2019.

[15] O. Raeesi, J. Pirskanen, A. Hazmi, T. Levanen, and M. Valkama, Performance evaluation of IEEE 802.11 ah and its restricted access window mechanism, in Communications Workshops (ICC), 2014 IEEE International Conference on. IEEE, 2014, pp. 460466.

[16] B. B. Olyaei, J. Pirskanen, O. Raeesi, A. Hazmi, and M. Valkama, Performance comparison between slotted IEEE 802.15.4 and IEEE 802.11ah in IoT based applications,in Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on. IEEE, 2013, pp. 332337.

[17] L. Zheng, M. Ni, L. Cai, J. Pan, C. Ghosh, and K. Doppler, Performance analysis of group-synchronized dcf for dense IEEE 802.11 networks, Wireless Communications, IEEE Transactions on, vol. 13, no. 11, pp. 61806192, 2014.

[18] Y. Zhou, H. Wang, S. Zheng, and Z. Lei, Advances in IEEE 802.11ah standardization for machine-type communications in sub-1GHz WLAN,in Communications Workshops (ICC), 2013 IEEE International Conference on, June 2013, pp. 12691273.

[19] W. Sun, M. Choi, and S. Choi, Ieee 802.11 ah: A long range 802.11 wlan at sub 1 ghz,Journal of ICT Standardization, vol. 1, no. 1, pp. 83108, 2013. [20] S. Aust, R. V. Prasad, and I. G. Niemegeers, IEEE 802.11 ah: Advantages in standards and further challenges for sub 1 GHz Wi-Fi,in Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012, pp. 68856889. [21] T. Adame, A. Bel, B. Bellalta, J. Barceló, J. Gonzalez, and M. Oliver, Capacity analysis of IEEE 802.11 ah WLANs for M2M communications, in Multiple Access Communcations. Springer, 2013, pp. 139155.

[22] IEEE Standard for Information technologyTelecommunications and information exchange between systems Local and metropolitan area networksSpecic requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specications, IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007), pp. 12793, March 2012.

[23] M. Park, IEEE P802. 11 Wireless LANs Specication Framework for TGah Rev. 15, TGah Spec Framework, doc.: IEEE 802.11-11/1137r6, 2014. [24] D. Halasz, Sub 1 GHz license-exempt par and 5c, IEEE 802.11-10/0001r13, July 2010.

[25] R. Porat, S. Young, and K. Doppler, IEEE P802. 11 Wireless LANs, TGah Channel Model Proposed Text,DCN 11-11-0968-03-00ah-channel-model-text, Tech. Rep., 2011.

[26] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, A survey on ieee 802.11 ah: An enabling networking technology for smart cities,Computer Communications, vol. 58, pp. 5369, 2015.

[27] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, Performance anomaly of 802.11b, in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 2, March 2003, pp. 836843.

[28] Leung, K.K. & Kim, B.-J. (2003). Frequency assignment for IEEE 802.11 wireless networks. 3. 1422 - 1426 Vol.3. 10.1109/VETECF.2003.1285259.

[29] Song, Taewon & Kim, Taeyoon. (2019). Performance Analysis of Addressing Mechanisms in Inter-Operable IoT Device with Low-Power Wake-Up Radio. Sensors. 19. 5106. 10.3390/s19235106.

[30] Paul, Biswajit. (2021). An Overview of LoRaWAN. WSEAS TRANSACTIONS ON COMMUNICATIONS. 19. 231-239. 10.37394/23204.2020.19.27.

[31] Lavric, Alexandru & Petrariu, Adrian & Valentin, Popa. (2019). Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2903157.

[32] Sri Pavan, Badarla & Govindan, Hari. (2021). Restricted Access Window-Based Resource Allocation Scheme for Performance Enhancement of IEEE 802.11ah Multi-Rate IoT Networks. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3117836.

[33] Wu, Y., Sheu, T., & Chan, P.H. (2021). Dynamic Slot Allocation in Restricted Access Window for IEEE 802.11ah Networks. *J. Inf. Sci. Eng., 37*, 697-708.

Παράρτημα

Κώδικας

Ακολουθεί ο κώδικας για την παραγωγή των γραφημάτων της προσομοίωσης

```
#
pytho
n lib
modul
es
        from __future__ import print_function
        import sys
        import shutil
        import types
        import optparse
        import os.path
        import re
        import shlex
        import subprocess
        import textwrap


        from utils import read_config_file
```

```
# WAF modules
from waflib import Utils, Scripting, Configure, Build, Options, TaskGen,
Context, Task, Logs, Errors
from waflib.Errors import WafError




# local modules
import wutils




# By default, all modules will be enabled, examples will be disabled,
# and tests will be disabled.
modules_enabled  = ['all_modules']
examples_enabled = False
tests_enabled    = False




                # λήψη πληροφοριών από τ αρχείο του NS-3 configuration file.
config_file_exists = False
(config_file_exists, modules_enabled, examples_enabled, tests_enabled) =
read_config_file()


                           sys.path.insert(0, os.path.abspath('waf-tools'))
try:
    import cflags # override the build profiles from waf
finally:
    sys.path.pop(0)


cflags.profiles = {
    # profile name: [optimization_level, warnings_level, debug_level]
                                                    'debug':     [0, 2, 3],
    'optimized': [3, 2, 1],
    'release':   [3, 2, 0],
    }
```

```
                                                cflags.default_profile = 'debug'


Configure.autoconfig = 0


# οι ακόλουθες 2 μεταβλητές χρησιμοποιούνται για το target "waf dist"
VERSION = open("VERSION", "rt").read().strip()
                                                                APPNAME = 'ns'


wutils.VERSION = VERSION
wutils.APPNAME = APPNAME



wutils.VNUM = None


top = '.'
                                                                 out = 'build'


def load_env():
    bld_cls = getattr(Utils.g_module, 'build_context', Utils.Context)
                                                            bld_ctx = bld_cls()
    bld_ctx.load_dirs(os.path.abspath(os.path.join (srcdir,'..')),
                      os.path.abspath(os.path.join (srcdir,'..', blddir)))
                                                            bld_ctx.load_envs()
    env = bld_ctx.get_env()
    return env


def get_files(base_dir):
    retval = []
    reference=os.path.dirname(base_dir)
                                for root, dirs, files in os.walk(base_dir):
        if root.find('.hg') != -1:
            continue
        for file in files:
            if file.find('.hg') != -1:
                continue
            fullname = os.path.join(root,file)
            # we can't use os.path.relpath because it's new in python 2.6
            relname = fullname.replace(reference + '/','')
                                            retval.append([fullname,relname])
```

```
    return retval



def dist_hook():
    import tarfile
    shutil.rmtree("doc/html", True)
    shutil.rmtree("doc/latex", True)
    shutil.rmtree("nsc", True)


# εμφάνιση ταξινομημένη λίστας
def print_module_names(names):
    # Sort the list of module names.
    names.sort()


                              # Εμφάνιση της λίστας των module σε 3 στήλες.
                                                                    i = 1
    for name in names:
        print(name.ljust(25), end=' ')
        if i == 3:
                print()
                i = 0
                                                                  i = i+1


    if i != 1:
        print()


                                                       def maybe_decode(input):
    if sys.version_info < (3,):
        return input
    else:
        try:
            return input.decode('utf-8')
        except:
            sys.exc_clear()
            return input


def options(opt):
    # options provided by the modules
                                                         opt.load('compiler_c')
```

```
    opt.load('compiler_cxx')
    opt.load('cflags')
    opt.load('gnu_dirs')


    opt.add_option('--check-config',
                   help=('Print the current configuration.'),
                   action="store_true", default=False,
                   dest="check_config")

    opt.add_option('--cwd',
                   help=('Set the working directory for a program.'),
                   action="store", type="string", default=None,
                                                    dest='cwd_launch')


    opt.add_option('--enable-gcov',
                   help=('Enable code coverage analysis.'
                         ' WARNING: this option only has effect '
                         'with the configure command.'),
                   action="store_true", default=False,
                                                  dest='enable_gcov')


    opt.add_option('--no-task-lines',
                   help=("Don't print task lines, i.e. messages saying
which tasks are being executed by WAF."
                         "  Coupled with a single -v will cause WAF to
output only the executed commands,"
                         " just like 'make' does by default."),
                   action="store_true", default=False,
                   dest='no_task_lines')


    opt.add_option('--lcov-report',
                                    help=('Generate a code coverage report '
                         '(use this option at build time, not in
configure)'),
                   action="store_true", default=False,
                   dest='lcov_report')


    opt.add_option('--run',
                   help=('Run a locally built program; argument can be a
program name,'
                           ' or a command starting with the program name.'),
```

```
                    type="string", default='', dest='run')
    opt.add_option('--visualize',
                    help=('Modify --run arguments to enable the
visualizer'),
                    action="store_true", default=False, dest='visualize')
    opt.add_option('--command-template',
                    help=('Template of the command used to run the program
given by --run;'
                          ' It should be a shell command string containing
                                                              %s inside,'
                          ' which will be replaced by the actual program.'),
                    type="string", default=None, dest='command_template')
    opt.add_option('--pyrun',
                    help=('Run a python program using locally built ns3
python module;'
                          ' argument is the path to the python program,
optionally followed'
                          ' by command-line options that are passed to the
                                                              program.'),
                    type="string", default='', dest='pyrun')
    opt.add_option('--valgrind',
                    help=('Change the default command template to run
programs and unit tests with valgrind'),
                    action="store_true", default=False,
                    dest='valgrind')
    opt.add_option('--shell',
                    help=('DEPRECATED (run ./waf shell)'),
                    action="store_true", default=False,
                    dest='shell')
    opt.add_option('--enable-sudo',
                    help=('Use sudo to setup suid bits on ns3
executables.'),
                    dest='enable_sudo', action='store_true',
                    default=False)
    opt.add_option('--enable-tests',
                    help=('Build the ns-3 tests.'),
                    dest='enable_tests', action='store_true',
                    default=False)
    opt.add_option('--disable-tests',
                    help=('Do not build the ns-3 tests.'),
                    dest='disable_tests', action='store_true',
                    default=False)
    opt.add_option('--enable-examples',
                    help=('Build the ns-3 examples.'),
                    dest='enable_examples', action='store_true',
                    default=False)
```

```
    opt.add_option('--disable-examples',
                   help=('Do not build the ns-3 examples.'),
                   dest='disable_examples', action='store_true',
                   default=False)
    opt.add_option('--check',
                   help=('DEPRECATED (run ./test.py)'),
                   default=False, dest='check', action="store_true")
    opt.add_option('--enable-static',
                   help=('Compile NS-3 statically: works only on linux,
without python'),
                   dest='enable_static', action='store_true',
                   default=False)
    opt.add_option('--enable-mpi',
                   help=('Compile NS-3 with MPI and distributed simulation
support'),
                   dest='enable_mpi', action='store_true',
                   default=False)
    opt.add_option('--doxygen-no-build',
                   help=('Run doxygen to generate html documentation from
source comments, '
                         'but do not wait for ns-3 to finish the full
build.'),
                   action="store_true", default=False,
                   dest='doxygen_no_build')
    opt.add_option('--enable-des-metrics',
                   help=('Log all events in a json file with the name of
the executable (which must call CommandLine::Parse(argc, argv)'),
                   action="store_true", default=False,
                   dest='enable_desmetrics')


    # options provided in subdirectories
    opt.recurse('src')
    opt.recurse('bindings/python')
    opt.recurse('src/internet')




def _check_compilation_flag(conf, flag, mode='cxx', linkflags=None):
    """
    Checks if the C++ compiler accepts a certain compilation flag or flags
    flag: can be a string or a list of strings
    """
    l = []
    if flag:
```

```
            l.append(flag)
        if isinstance(linkflags, list):
                                                            l.extend(linkflags)
        else:
            if linkflags:
                l.append(linkflags)
        if len(l) > 1:
                                              flag_str = 'flags ' + ' '.join(l)
                                                                    else:
            flag_str = 'flag ' + ' '.join(l)
        if len(flag_str) > 28:
            flag_str = flag_str[:28] + "..."


        conf.start_msg('Checking for compilation %s support' % (flag_str,))
        env = conf.env.derive()


        retval = False
        if mode == 'cc':
            mode = 'c'


        if mode == 'cxx':
            env.append_value('CXXFLAGS', flag)
        else:
            env.append_value('CFLAGS', flag)


        if linkflags is not None:
            env.append_value("LINKFLAGS", linkflags)


                                                                          try:
            retval = conf.check(compiler=mode, fragment='int main() { return 0;
    }', features='c', env=env)
        except Errors.ConfigurationError:
                                                                  ok = False
        else:
            ok = (retval == True)
        conf.end_msg(ok)
                                                                   return ok
```

```
def report_optional_feature(conf, name, caption, was_enabled,
reason_not_enabled):
    conf.env.append_value('NS3_OPTIONAL_FEATURES', [(name, caption,
was_enabled, reason_not_enabled)])




def check_optional_feature(conf, name):
                    for (name1, caption, was_enabled, reason_not_enabled) in
                                                conf.env.NS3_OPTIONAL_FEATURES:
        if name1 == name:
            return was_enabled
    raise KeyError("Feature %r not declared yet" % (name,))









                                  def _check_nonfatal(conf, *args, **kwargs):
                                                                        try:
        return conf.check(*args, **kwargs)
    except conf.errors.ConfigurationError:
                                                                  return None


# Write a summary of optional features status
def print_config(env, phase='configure'):
    if phase == 'configure':
        profile = get_build_profile(env)
    else:
                                             profile = get_build_profile()

    print("---- Summary of optional NS-3 features:")
    print("%-30s: %s%s%s" % ("Build profile", Logs.colors('GREEN'),
                             profile, Logs.colors('NORMAL')))
    bld = wutils.bld
    print("%-30s: %s%s%s" % ("Build directory", Logs.colors('GREEN'),
                             Options.options.out, Logs.colors('NORMAL')))


                      for (name, caption, was_enabled, reason_not_enabled) in
                    sorted(env['NS3_OPTIONAL_FEATURES'], key=lambda s : s[1]):
        if was_enabled:
```

```
            status = 'enabled'
            color = 'GREEN'
        else:
            status = 'not enabled (%s)' % reason_not_enabled
            color = 'RED'
        print("%-30s: %s%s%s" % (caption, Logs.colors(color), status,
Logs.colors('NORMAL')))


def configure(conf):
    conf.load('relocation', tooldir=['waf-tools'])


    # attach some extra methods
    conf.check_nonfatal = types.MethodType(_check_nonfatal, conf)
    conf.check_compilation_flag = types.MethodType(_check_compilation_flag,
conf)
    conf.report_optional_feature =
types.MethodType(report_optional_feature, conf)
    conf.check_optional_feature = types.MethodType(check_optional_feature,
conf)
    conf.env['NS3_OPTIONAL_FEATURES'] = []


    conf.load('compiler_c')
    cc_string = '.'.join(conf.env['CC_VERSION'])
    conf.msg('Checking for cc version',cc_string,'GREEN')
    conf.load('compiler_cxx')
    conf.load('cflags', tooldir=['waf-tools'])
                                conf.load('command', tooldir=['waf-tools'])
    conf.load('gnu_dirs')
    conf.load('clang_compilation_database', tooldir=['waf-tools'])


    env = conf.env


    if Options.options.enable_gcov:
        env['GCOV_ENABLED'] = True
        env.append_value('CCFLAGS', '-fprofile-arcs')
        env.append_value('CCFLAGS', '-ftest-coverage')
                            env.append_value('CXXFLAGS', '-fprofile-arcs')
        env.append_value('CXXFLAGS', '-ftest-coverage')
        env.append_value('LINKFLAGS', '-lgcov')
        env.append_value('LINKFLAGS', '-coverage')
```

```
    if Options.options.build_profile == 'debug':
        env.append_value('DEFINES', 'NS3_BUILD_PROFILE_DEBUG')
        env.append_value('DEFINES', 'NS3_ASSERT_ENABLE')
        env.append_value('DEFINES', 'NS3_LOG_ENABLE')


    if Options.options.build_profile == 'release':
                    env.append_value('DEFINES', 'NS3_BUILD_PROFILE_RELEASE')


    if Options.options.build_profile == 'optimized':
        env.append_value('DEFINES', 'NS3_BUILD_PROFILE_OPTIMIZED')


    env['PLATFORM'] = sys.platform
    env['BUILD_PROFILE'] = Options.options.build_profile
    if Options.options.build_profile == "release":
        env['BUILD_SUFFIX'] = ''
                                                                        else:
        env['BUILD_SUFFIX'] = '-'+Options.options.build_profile

    env['APPNAME'] = wutils.APPNAME
    env['VERSION'] = wutils.VERSION


    if conf.env['CXX_NAME'] in ['gcc', 'icc']:
        if Options.options.build_profile == 'release':
                        env.append_value('CXXFLAGS', '-fomit-frame-pointer')
        if Options.options.build_profile == 'optimized':
            if conf.check_compilation_flag('-march=native'):
                            env.append_value('CXXFLAGS', '-march=native')
            env.append_value('CXXFLAGS', '-fstrict-overflow')
            if conf.env['CC_VERSION'] >= gcc_version_warn_strict_overflow:
                env.append_value('CXXFLAGS', '-Wstrict-overflow=2')


        if sys.platform == 'win32':
            env.append_value("LINKFLAGS", "-Wl,--enable-runtime-pseudo-
reloc")
        elif sys.platform == 'cygwin':
            env.append_value("LINKFLAGS", "-Wl,--enable-auto-import")


                                                        cxx = env['CXX']
        cxx_check_libstdcxx = cxx + ['-print-file-name=libstdc++.so']
```

```
        p = subprocess.Popen(cxx_check_libstdcxx, stdout=subprocess.PIPE)
        libstdcxx_output = maybe_decode(p.stdout.read().strip())
        libstdcxx_location = os.path.dirname(libstdcxx_output)
        p.wait()
        if libstdcxx_location:
            conf.env.append_value('NS3_MODULE_PATH', libstdcxx_location)


        if Options.platform in ['linux']:
                        if conf.check_compilation_flag('-Wl,--soname=foo'):
                env['WL_SONAME_SUPPORTED'] = True


    # bug 2181 on clang warning suppressions
    if conf.env['CXX_NAME'] in ['clang']:
                                            if Options.platform == 'darwin':
            if conf.env['CC_VERSION'] >=
darwin_clang_version_warn_unused_local_typedefs:
                env.append_value('CXXFLAGS', '-Wno-unused-local-typedefs')
            if conf.env['CC_VERSION'] >=
darwin_clang_version_warn_potentially_evaluated:
                env.append_value('CXXFLAGS', '-Wno-potentially-evaluated-
expression')
        else:
            if conf.env['CC_VERSION'] >=
clang_version_warn_unused_local_typedefs:
                env.append_value('CXXFLAGS', '-Wno-unused-local-typedefs')
            if conf.env['CC_VERSION'] >=
clang_version_warn_potentially_evaluated:
                  env.append_value('CXXFLAGS', '-Wno-potentially-evaluated-
                                                                    expression')
    env['ENABLE_STATIC_NS3'] = False
    if Options.options.enable_static:
        if Options.platform == 'darwin':
                  if conf.check_compilation_flag(flag=[], linkflags=['-Wl,-
                                                                    all_load']):
                conf.report_optional_feature("static", "Static build",
True, '')
                env['ENABLE_STATIC_NS3'] = True
            else:
                conf.report_optional_feature("static", "Static build",
False,
                                             "Link flag -Wl,-all_load does
not work")
        else:
            if conf.check_compilation_flag(flag=[], linkflags=['-Wl,--
```

```
whole-archive,-Bstatic', '-Wl,-Bdynamic,--no-whole-archive']):
                conf.report_optional_feature("static", "Static build",
True, '')
                env['ENABLE_STATIC_NS3'] = True
            else:
                conf.report_optional_feature("static", "Static build",
False,
                                             "Link flag -Wl,--whole-
archive,-Bstatic does not work")


    # Enable C++-11 support
    env.append_value('CXXFLAGS', '-std=c++11')


    # Set this so that the lists won't be printed at the end of this
    # configure command.
    conf.env['PRINT_BUILT_MODULES_AT_END'] = False


    conf.env['MODULES_NOT_BUILT'] = []


    conf.recurse('bindings/python')


    conf.recurse('src')


    # Set the list of enabled modules.
    if Options.options.enable_modules:
                                       # Use the modules explicitly enabled.
        conf.env['NS3_ENABLED_MODULES'] = ['ns3-'+mod for mod in

Options.options.enable_modules.split(',')]
                                                                        else:
        # Use the enabled modules list from the ns3 configuration file.
        if modules_enabled[0] == 'all_modules':
            # Enable all modules if requested.
                  conf.env['NS3_ENABLED_MODULES'] = conf.env['NS3_MODULES']
        else:
                                         # Enable the modules from the list.
            conf.env['NS3_ENABLED_MODULES'] = ['ns3-'+mod for mod in
                                                           modules_enabled]
```

```
                # Add the template module to the list of enabled modules that
    # should not be built if this is a static build on Darwin.  They
    # don't work there for the template module, and this is probably
    # because the template module has no source files.
    if conf.env['ENABLE_STATIC_NS3'] and sys.platform == 'darwin':
        conf.env['MODULES_NOT_BUILT'].append('template')


    # Remove these modules from the list of enabled modules.
    for not_built in conf.env['MODULES_NOT_BUILT']:
        not_built_name = 'ns3-' + not_built
        if not_built_name in conf.env['NS3_ENABLED_MODULES']:
            conf.env['NS3_ENABLED_MODULES'].remove(not_built_name)
            if not conf.env['NS3_ENABLED_MODULES']:
                raise WafError('Exiting because the ' + not_built + '
module can not be built and it was the only one enabled.')


                                                    conf.recurse('src/mpi')


    # for suid bits
    try:
        conf.find_program('sudo', var='SUDO')
    except WafError:
        pass


    why_not_sudo = "because we like it"
    if Options.options.enable_sudo and conf.env['SUDO']:
        env['ENABLE_SUDO'] = True
    else:
        env['ENABLE_SUDO'] = False
        if Options.options.enable_sudo:
            why_not_sudo = "program sudo not found"
                                                                    else:
            why_not_sudo = "option --enable-sudo not selected"


    conf.report_optional_feature("ENABLE_SUDO", "Use sudo to set suid bit",
env['ENABLE_SUDO'], why_not_sudo)


    # Decide if tests will be built or not.
    if Options.options.enable_tests:
        # Tests were explicitly enabled.
```

```
                                                    env['ENABLE_TESTS'] = True
            why_not_tests = "option --enable-tests selected"
        elif Options.options.disable_tests:
            # Tests were explicitly disabled.
            env['ENABLE_TESTS'] = False
            why_not_tests = "option --disable-tests selected"
        else:
            # Enable tests based on the ns3 configuration file.
            env['ENABLE_TESTS'] = tests_enabled
            if config_file_exists:
                                why_not_tests = "based on configuration file"
            elif tests_enabled:
                                        why_not_tests = "defaults to enabled"
            else:
                why_not_tests = "defaults to disabled"


        conf.report_optional_feature("ENABLE_TESTS", "Tests",
    env['ENABLE_TESTS'], why_not_tests)


        # Decide if examples will be built or not.
        if Options.options.enable_examples:
            # Examples were explicitly enabled.
            env['ENABLE_EXAMPLES'] = True
            why_not_examples = "option --enable-examples selected"
        elif Options.options.disable_examples:
            # Examples were explicitly disabled.
            env['ENABLE_EXAMPLES'] = False
            why_not_examples = "option --disable-examples selected"
        else:
            # Enable examples based on the ns3 configuration file.
            env['ENABLE_EXAMPLES'] = examples_enabled
            if config_file_exists:
                                why_not_examples = "based on configuration file"
            elif examples_enabled:
                                        why_not_examples = "defaults to enabled"
            else:
                why_not_examples = "defaults to disabled"


        conf.report_optional_feature("ENABLE_EXAMPLES", "Examples",
    env['ENABLE_EXAMPLES'],
                                    why_not_examples)
        try:
            for dir in os.listdir('examples'):
```

```
            if dir.startswith('.') or dir == 'CVS':
                continue
            conf.env.append_value('EXAMPLE_DIRECTORIES', dir)
    except OSError:
        return


    env['VALGRIND_FOUND'] = False
    try:
        conf.find_program('valgrind', var='VALGRIND')
        env['VALGRIND_FOUND'] = True
    except WafError:
                                                                    pass


    # These flags are used for the implicitly dependent modules.
    if env['ENABLE_STATIC_NS3']:
        if sys.platform == 'darwin':
            env.STLIB_MARKER = '-Wl,-all_load'
        else:
            env.STLIB_MARKER = '-Wl,--whole-archive,-Bstatic'
            env.SHLIB_MARKER = '-Wl,-Bdynamic,--no-whole-archive'




    have_gsl = conf.check_cfg(package='gsl', args=['--cflags', '--libs'],
                              uselib_store='GSL', mandatory=False)
    conf.env['ENABLE_GSL'] = have_gsl
    conf.report_optional_feature("GSL", "GNU Scientific Library (GSL)",
                                 conf.env['ENABLE_GSL'],
                                 "GSL not found")


    conf.find_program('libgcrypt-config', var='LIBGCRYPT_CONFIG',
msg="python-config", mandatory=False)
    if env.LIBGCRYPT_CONFIG:
        conf.check_cfg(path=env.LIBGCRYPT_CONFIG, msg="Checking for
libgcrypt", args='--cflags --libs', package='',
                                     define_name="HAVE_CRYPTO",
global_define=True, uselib_store='GCRYPT', mandatory=False)
    conf.report_optional_feature("libgcrypt", "Gcrypt library",
                                 conf.env.HAVE_GCRYPT, "libgcrypt not
found: you can use libgcrypt-config to find its location.")
```

```
                                why_not_desmetrics = "defaults to disabled"
                                      if Options.options.enable_desmetrics:
            conf.env['ENABLE_DES_METRICS'] = True
            env.append_value('DEFINES', 'ENABLE_DES_METRICS')
            why_not_desmetrics = "option --enable-des-metrics selected"
        conf.report_optional_feature("DES Metrics", "DES Metrics event
    collection", conf.env['ENABLE_DES_METRICS'], why_not_desmetrics)




                             # for compiling C code, copy over the CXX* flags
        conf.env.append_value('CCFLAGS', conf.env['CXXFLAGS'])


        def add_gcc_flag(flag):
            if env['COMPILER_CXX'] == 'g++' and 'CXXFLAGS' not in os.environ:
                if conf.check_compilation_flag(flag, mode='cxx'):
                    env.append_value('CXXFLAGS', flag)
                 if env['COMPILER_CC'] == 'gcc' and 'CCFLAGS' not in os.environ:
                if conf.check_compilation_flag(flag, mode='cc'):
                    env.append_value('CCFLAGS', flag)


        add_gcc_flag('-Wno-error=deprecated-declarations')
        add_gcc_flag('-fstrict-aliasing')
        add_gcc_flag('-Wstrict-aliasing')


                                                                        try:
            conf.find_program('doxygen', var='DOXYGEN')
        except WafError:
                                                                        pass


        # append user defined flags after all our ones
        for (confvar, envvar) in [['CCFLAGS', 'CCFLAGS_EXTRA'],
                                  ['CXXFLAGS', 'CXXFLAGS_EXTRA'],
                                  ['LINKFLAGS', 'LINKFLAGS_EXTRA'],
                                  ['LINKFLAGS', 'LDFLAGS_EXTRA']]:
            if envvar in os.environ:
                                        value = shlex.split(os.environ[envvar])
                conf.env.append_value(confvar, value)


        print_config(env)
```

```
class SuidBuild_task(Task.Task):
    """task that makes a binary Suid
    """
                                                            after = 'link'
    def __init__(self, *args, **kwargs):
        super(SuidBuild_task, self).__init__(*args, **kwargs)
        self.m_display = 'build-suid'
        try:
            program_obj = wutils.find_program(self.generator.name,
self.generator.env)
        except ValueError as ex:
            raise WafError(str(ex))
        program_node = program_obj.path.find_or_declare(program_obj.target)
                            self.filename = program_node.get_bld().abspath()




                                                            def run(self):
        print('setting suid bit on executable ' + self.filename,
file=sys.stderr)
        if subprocess.Popen(['sudo', 'chown', 'root',
self.filename]).wait():
            return 1
        if subprocess.Popen(['sudo', 'chmod', 'u+s',
self.filename]).wait():
            return 1
        return 0


    def runnable_status(self):
        "RUN_ME SKIP_ME or ASK_LATER"
        try:
            st = os.stat(self.filename)
        except OSError:
            return Task.ASK_LATER
                                                          if st.st_uid == 0:
                                                        return Task.SKIP_ME
        else:
            return Task.RUN_ME


def create_suid_program(bld, name):
```

```
    grp = bld.current_group
    bld.add_group() # this to make sure no two sudo tasks run at the same
time
    program = bld(features='cxx cxxprogram')
                                                program.is_ns3_program = True
    program.module_deps = list()
    program.name = name
    program.target = "%s%s-%s%s" % (wutils.APPNAME, wutils.VERSION, name,
bld.env.BUILD_SUFFIX)


    if bld.env['ENABLE_SUDO']:
        program.create_task("SuidBuild")


    bld.set_group(grp)


    return program


def create_ns3_program(bld, name, dependencies=('core',)):
    program = bld(features='cxx cxxprogram')


    program.is_ns3_program = True
    program.name = name
    program.target = "%s%s-%s%s" % (wutils.APPNAME, wutils.VERSION, name,
bld.env.BUILD_SUFFIX)
    # Each of the modules this program depends on has its own library.
    program.ns3_module_dependencies = ['ns3-'+dep for dep in dependencies]
                                                      program.includes = "#"
    program.use = program.ns3_module_dependencies
    if program.env['ENABLE_STATIC_NS3']:
                                                 if sys.platform == 'darwin':
            program.env.STLIB_MARKER = '-Wl,-all_load'
                                                                        else:
            program.env.STLIB_MARKER = '-Wl,-Bstatic,--whole-archive'
               program.env.SHLIB_MARKER = '-Wl,-Bdynamic,--no-whole-archive'
    else:
        if program.env.DEST_BINFMT == 'elf':
              # All ELF platforms are impacted but only the gcc compiler has
                                                           a flag to fix it.
            if 'gcc' in (program.env.CXX_NAME, program.env.CC_NAME):
                program.env.append_value ('SHLIB_MARKER', '-Wl,--no-as-
needed')
```

```
    return program


def register_ns3_script(bld, name, dependencies=('core',)):
    ns3_module_dependencies = ['ns3-'+dep for dep in dependencies]
    bld.env.append_value('NS3_SCRIPT_DEPENDENCIES', [(name,
ns3_module_dependencies)])


def add_examples_programs(bld):
    env = bld.env
    if env['ENABLE_EXAMPLES']:
        try:
            for dir in os.listdir('examples'):
                if dir.startswith('.') or dir == 'CVS':
                    continue
                if os.path.isdir(os.path.join('examples', dir)):
                    bld.recurse(os.path.join('examples', dir))
                                                            except OSError:
            return


def add_scratch_programs(bld):
    all_modules = [mod[len("ns3-"):] for mod in
bld.env['NS3_ENABLED_MODULES']]
    try:
                                      for filename in os.listdir("scratch"):
            if filename.startswith('.') or filename == 'CVS':
                continue
            if os.path.isdir(os.path.join("scratch", filename)):
                obj = bld.create_ns3_program(filename, all_modules)
                obj.path = obj.path.find_dir('scratch').find_dir(filename)
                obj.source = obj.path.ant_glob('*.cc')
                obj.target = filename
                obj.name = obj.target
                obj.install_path = None
            elif filename.endswith(".cc"):
                name = filename[:-len(".cc")]
                            obj = bld.create_ns3_program(name, all_modules)
                obj.path = obj.path.find_dir('scratch')
                obj.source = filename
                obj.target = name
                obj.name = obj.target
                obj.install_path = None
```

```
        except OSError:
            return


    def _get_all_task_gen(self):
        for group in self.groups:
            for taskgen in group:
                yield taskgen




    # ok, so WAF does not provide an API to prevent an
    # arbitrary taskgen from running; we have to muck around with
    # WAF internal state, something that might stop working if
    # WAF is upgraded...
    def _exclude_taskgen(self, taskgen):
        for group in self.groups:
            for tg1 in group:
                if tg1 is taskgen:
                    group.remove(tg1)
                                                                        break
            else:
                continue
            break




                                            def _find_ns3_module(self, name):
        for obj in _get_all_task_gen(self):
            # disable the modules themselves
            if hasattr(obj, "is_ns3_module") and obj.name == name:
                return obj
        raise KeyError(name)


    # Parse the waf lockfile generated by latest 'configure' operation
    def get_build_profile(env=None):
        if env == None:
            lockfile = os.environ.get('WAFLOCK', '.lock-waf_%s_build' %
    sys.platform)
            profile = "not found"
            with open(lockfile, "r") as f:
                for line in f:
                                             if line.startswith("options ="):
```

```
                                                    key, val = line.split('=')
                        arr = val.split(',')
                        for x in arr:
                            optkey,optval = x.split(':')
                            if (optkey.lstrip() == '\'build_profile\''):
                                profile = str(optval.lstrip()).replace("'","")
        else:
                                        profile = Options.options.build_profile
        return profile


def build(bld):
    env = bld.env


    if Options.options.check_config:
        print_config(env, 'build')
    else:
        if Options.options.check_profile:
            profile = get_build_profile()
            print("Build profile: %s" % profile)

    if Options.options.check_profile or Options.options.check_config:
        raise SystemExit(0)
        return


    # If --enabled-modules option was given, then print a warning
                                            # message and exit this function.
    if Options.options.enable_modules:
        Logs.warn("No modules were built.  Use waf configure --enable-
modules to enable modules.")
                                                                        return


    bld.env['NS3_MODULES_WITH_TEST_LIBRARIES'] = []
    bld.env['NS3_ENABLED_MODULE_TEST_LIBRARIES'] = []
    bld.env['NS3_SCRIPT_DEPENDENCIES'] = []
    bld.env['NS3_RUNNABLE_PROGRAMS'] = []
    bld.env['NS3_RUNNABLE_SCRIPTS'] = []


    wutils.bld = bld
    if Options.options.no_task_lines:
        from waflib import Runner
                                                        def null_printout(s):
```

```
        pass
    Runner.printout = null_printout


Options.cwd_launch = bld.path.abspath()
bld.create_ns3_program = types.MethodType(create_ns3_program, bld)
   bld.register_ns3_script = types.MethodType(register_ns3_script, bld)
bld.create_suid_program = types.MethodType(create_suid_program, bld)
bld.__class__.all_task_gen = property(_get_all_task_gen)
bld.exclude_taskgen = types.MethodType(_exclude_taskgen, bld)
bld.find_ns3_module = types.MethodType(_find_ns3_module, bld)


  # Clean documentation build directories; other cleaning happens later
if bld.cmd == 'clean':
    _cleandocs()


# process subfolders from here
bld.recurse('src')


      # If modules have been enabled, then set lists of enabled modules
# and enabled module test libraries.
if env['NS3_ENABLED_MODULES']:
    modules = env['NS3_ENABLED_MODULES']


    # Find out about additional modules that need to be enabled
    # due to dependency constraints.
    changed = True
                                                        while changed:
        changed = False
        for module in modules:
            module_obj = bld.get_tgen_by_name(module)
                                                  if module_obj is None:
                raise ValueError("module %s not found" % module)
            # Each enabled module has its own library.
                                             for dep in module_obj.use:
                if not dep.startswith('ns3-'):
                    continue
                if dep not in modules:
                    modules.append(dep)
                                                        changed = True
```

```
        env['NS3_ENABLED_MODULES'] = modules


        # If tests are being built, then set the list of the enabled
        # module test libraries.
        if env['ENABLE_TESTS']:
            for (mod, testlib) in
bld.env['NS3_MODULES_WITH_TEST_LIBRARIES']:
                if mod in bld.env['NS3_ENABLED_MODULES']:

bld.env.append_value('NS3_ENABLED_MODULE_TEST_LIBRARIES', testlib)


    add_examples_programs(bld)
    add_scratch_programs(bld)


    if env['NS3_ENABLED_MODULES']:
        modules = env['NS3_ENABLED_MODULES']


         # Exclude the programs other misc task gens that depend on disabled
                                                                        modules
        for obj in list(bld.all_task_gen):


            # check for ns3moduleheader_taskgen
            if 'ns3moduleheader' in getattr(obj, "features", []):
                if ("ns3-%s" % obj.module) not in modules:
                    obj.mode = 'remove' # tell it to remove headers instead
of installing


            # check for programs
                                if hasattr(obj, 'ns3_module_dependencies'):
                # this is an NS-3 program (bld.create_ns3_program)
                program_built = True
                                    for dep in obj.ns3_module_dependencies:
                    if dep not in modules: # prog. depends on a module that
isn't enabled?
                        bld.exclude_taskgen(obj)
                                                program_built = False
                        break


                                # Add this program to the list if all of its
```

```
                # dependencies will be built.
                if program_built:
                    object_name = "%s%s-%s%s" % (wutils.APPNAME,
wutils.VERSION,
                                                 obj.name,
bld.env.BUILD_SUFFIX)


                    # Get the relative path to the program from the
                    # launch directory.
                    launch_dir = os.path.abspath(Context.launch_dir)
                    object_relative_path = os.path.join(
                        wutils.relpath(obj.path.get_bld().abspath(),
launch_dir),
                        object_name)


                    bld.env.append_value('NS3_RUNNABLE_PROGRAMS',
object_relative_path)


                                            # disable the modules themselves
            if hasattr(obj, "is_ns3_module") and obj.name not in modules:
                bld.exclude_taskgen(obj) # kill the module


            # disable the module test libraries
            if hasattr(obj, "is_ns3_module_test_library"):
                       if not env['ENABLE_TESTS'] or (obj.module_name not in
                                                                  modules):
                    bld.exclude_taskgen(obj) # kill the module test library


            # disable the ns3header_taskgen
            if 'ns3header' in getattr(obj, "features", []):
                if ("ns3-%s" % obj.module) not in modules:
                    obj.mode = 'remove' # tell it to remove headers instead
of installing


            # disable the ns3privateheader_taskgen
                      if 'ns3privateheader' in getattr(obj, "features", []):
                if ("ns3-%s" % obj.module) not in modules:
                    obj.mode = 'remove' # tell it to remove headers instead
of installing
```

```
                            # disable pcfile taskgens for disabled modules
        if 'ns3pcfile' in getattr(obj, "features", []):
            if obj.module not in bld.env.NS3_ENABLED_MODULES:
                bld.exclude_taskgen(obj)




if env['NS3_ENABLED_MODULES']:
    env['NS3_ENABLED_MODULES'] = list(modules)


# Determine which scripts will be runnable.
       for (script, dependencies) in bld.env['NS3_SCRIPT_DEPENDENCIES']:
    script_runnable = True
    for dep in dependencies:
        if dep not in modules:
            script_runnable = False
                                                                    break


    # Add this script to the list if all of its dependencies will
    # be built.
    if script_runnable:
                    bld.env.append_value('NS3_RUNNABLE_SCRIPTS', script)


bld.recurse('bindings/python')


        # Process this subfolder here after the lists of enabled modules
# and module test libraries have been set.
bld.recurse('utils')


# Set this so that the lists will be printed at the end of this
# build command.
bld.env['PRINT_BUILT_MODULES_AT_END'] = True


          # Do not print the modules built if build command was "clean"
if bld.cmd == 'clean':
    bld.env['PRINT_BUILT_MODULES_AT_END'] = False
```

```
    if Options.options.run:
                            # Check that the requested program name is valid
        program_name, dummy_program_argv =
wutils.get_run_program(Options.options.run,
wutils.get_command_template(env))


        # When --run'ing a program, tell WAF to only build that program,
        # nothing more; this greatly speeds up compilation when all you
        # want to do is run a test program.
             Options.options.targets += ',' + os.path.basename(program_name)
        if getattr(Options.options, "visualize", False):
            program_obj = wutils.find_program(program_name, bld.env)
            program_obj.use.append('ns3-visualizer')
                                                for gen in bld.all_task_gen:
            if type(gen).__name__ in ['ns3header_taskgen',
'ns3privateheader_taskgen', 'ns3moduleheader_taskgen']:
                gen.post()
        bld.env['PRINT_BUILT_MODULES_AT_END'] = False


    if Options.options.doxygen_no_build:
        _doxygen(bld)
        raise SystemExit(0)


def _cleandir(name):
    try:
        shutil.rmtree(name)
    except:
        pass


def _cleandocs():
    _cleandir('doc/html')
    _cleandir('doc/manual/build')
    _cleandir('doc/manual/source-temp')
    _cleandir('doc/tutorial/build')
                                      _cleandir('doc/tutorial-pt-br/build')
    _cleandir('doc/models/build')
    _cleandir('doc/models/source-temp')


                    # 'distclean' typically only cleans out build/ directory
# Here we clean out any build or documentation artifacts not in build/
def distclean(ctx):
```

```
    _cleandocs()
    # Now call waf's normal distclean
    Scripting.distclean(ctx)


def shutdown(ctx):
    bld = wutils.bld
    if wutils.bld is None:
        return
    env = bld.env


    # Only print the lists if a build was done.
    if (env['PRINT_BUILT_MODULES_AT_END']):


        # Print the list of built modules.
        print()
        print('Modules built:')
        names_without_prefix = []
        for name in env['NS3_ENABLED_MODULES']:
            name1 = name[len('ns3-'):]
                                if name not in env.MODULAR_BINDINGS_MODULES:
                name1 += " (no Python)"
            names_without_prefix.append(name1)
        print_module_names(names_without_prefix)
        print()


                    # Print the list of enabled modules that were not built.
        if env['MODULES_NOT_BUILT']:
            print('Modules not built (see ns-3 tutorial for explanation):')
                                print_module_names(env['MODULES_NOT_BUILT'])
            print()


        # Set this so that the lists won't be printed until the next
        # build is done.
        bld.env['PRINT_BUILT_MODULES_AT_END'] = False


    # Write the build status file.
    build_status_file = os.path.join(bld.out_dir, 'build-status.py')
    out = open(build_status_file, 'w')
    out.write('#! /usr/bin/env python\n')
    out.write('\n')
```

```
                                out.write('# Programs that are runnable.\n')
    out.write('ns3_runnable_programs = ' +
str(env['NS3_RUNNABLE_PROGRAMS']) + '\n')
    out.write('\n')
    out.write('# Scripts that are runnable.\n')
    out.write('ns3_runnable_scripts = ' + str(env['NS3_RUNNABLE_SCRIPTS'])
+ '\n')
    out.write('\n')
                                                                out.close()


    if Options.options.lcov_report:
        lcov_report(bld)


    if Options.options.run:
        wutils.run_program(Options.options.run, env,
wutils.get_command_template(env),
                           visualize=Options.options.visualize)
        raise SystemExit(0)


    if Options.options.pyrun:
        wutils.run_python_program(Options.options.pyrun, env,
                                  visualize=Options.options.visualize)
        raise SystemExit(0)


    if Options.options.shell:
        raise WafError("Please run `./waf shell' now, instead of `./waf --
shell'")


    if Options.options.check:
        raise WafError("Please run `./test.py' now, instead of `./waf --
check'")


    check_shell(bld)




                                    class CheckContext(Context.Context):
```

```
    """run the equivalent of the old ns-3 unit tests using test.py"""
    cmd = 'check'
    def execute(self):
        # first we execute the build
                                       bld = Context.create_context("build")
        bld.options = Options.options # provided for convenience
        bld.cmd = "build"
                                                              bld.execute()

        wutils.bld = bld
                    wutils.run_python_program("test.py -n -c core", bld.env)




                        class print_introspected_doxygen_task(Task.TaskBase):
                                                             after = 'cxx link'
    color = 'BLUE'


    def __init__(self, bld):
        self.bld = bld
        super(print_introspected_doxygen_task,
self).__init__(generator=self)

    def __str__(self):
        return 'print-introspected-doxygen\n'


    def runnable_status(self):
        return Task.RUN_ME


                                                                 def run(self):
                                     ## generate the trace sources list docs
        env = wutils.bld.env
        proc_env = wutils.get_proc_env()
        try:
             program_obj = wutils.find_program('print-introspected-doxygen',
                                                                        env)
        except ValueError: # could happen if print-introspected-doxygen is
                           # not built because of waf configure
                           # --enable-modules=xxx
            pass
        else:
            prog =
```

```
program_obj.path.find_or_declare(ccroot.get_target_name(program_obj)).get_b
ld().abspath(env)


            # Create a header file with the introspected information.
            doxygen_out = open(os.path.join('doc', 'introspected-
doxygen.h'), 'w')
                                if subprocess.Popen([prog], stdout=doxygen_out,
                                                    env=proc_env).wait():
                raise SystemExit(1)
            doxygen_out.close()

            # Create a text file with the introspected information.
            text_out = open(os.path.join('doc', 'ns3-object.txt'), 'w')
            if subprocess.Popen([prog, '--output-text'], stdout=text_out,
env=proc_env).wait():
                raise SystemExit(1)
            text_out.close()


class run_python_unit_tests_task(Task.TaskBase):
    after = 'cxx link'
    color = 'BLUE'


    def __init__(self, bld):
        self.bld = bld
        super(run_python_unit_tests_task, self).__init__(generator=self)

    def __str__(self):
        return 'run-python-unit-tests\n'


    def runnable_status(self):
        return Task.RUN_ME


    def run(self):
                                           proc_env = wutils.get_proc_env()
        wutils.run_argv([self.bld.env['PYTHON'], os.path.join("..",
"utils", "python-unit-tests.py")],
                        self.bld.env, proc_env, force_no_valgrind=True)


                                                    def check_shell(bld):
    if ('NS3_MODULE_PATH' not in os.environ) or ('NS3_EXECUTABLE_PATH' not
```

```
in os.environ):
        return
    env = bld.env
    correct_modpath = os.pathsep.join(env['NS3_MODULE_PATH'])
    found_modpath = os.environ['NS3_MODULE_PATH']
    correct_execpath = os.pathsep.join(env['NS3_EXECUTABLE_PATH'])
                        found_execpath = os.environ['NS3_EXECUTABLE_PATH']
    if (found_modpath != correct_modpath) or (correct_execpath !=
found_execpath):
        msg = ("Detected shell (./waf shell) with incorrect
configuration\n"

               "==========================================================\n"
               "Possible reasons for this problem:\n"
               "  1. You switched to another ns-3 tree from inside this
shell\n"
               "  2. You switched ns-3 debug level (waf configure --
debug)\n"
               "  3. You modified the list of built ns-3 modules\n"
                     "You should correct this situation before running any
                                      program.  Possible solutions:\n"
               "  1. Exit this shell, and start a new one\n"
               "  2. Run a new nested shell")
        raise WafError(msg)




class Ns3ShellContext(Context.Context):
    """run a shell with an environment suitably modified to run locally
built programs"""
    cmd = 'shell'

    def execute(self):
        # first we execute the build
        bld = Context.create_context("build")
        bld.options = Options.options # provided for convenience
        bld.cmd = "build"
        bld.execute()


        # Set this so that the lists won't be printed when the user
        # exits the shell.
        bld.env['PRINT_BUILT_MODULES_AT_END'] = False

                                               if sys.platform == 'win32':
```

```
            shell = os.environ.get("COMSPEC", "cmd.exe")
        else:
            shell = os.environ.get("SHELL", "/bin/sh")


        env = bld.env
        os_env = {
            'NS3_MODULE_PATH': os.pathsep.join(env['NS3_MODULE_PATH']),
            'NS3_EXECUTABLE_PATH':
os.pathsep.join(env['NS3_EXECUTABLE_PATH']),
            }
        wutils.run_argv([shell], env, os_env)




def _doxygen(bld):
    env = wutils.bld.env
    proc_env = wutils.get_proc_env()


    if not env['DOXYGEN']:
        Logs.error("waf configure did not detect doxygen in the system ->
cannot build api docs.")
        raise SystemExit(1)
                                                                    return


    try:
        program_obj = wutils.find_program('print-introspected-doxygen',
env)
    except ValueError:
        Logs.warn("print-introspected-doxygen does not exist")
        raise SystemExit(1)
                                                                    return


    prog =
program_obj.path.find_or_declare(program_obj.target).get_bld().abspath()


    if not os.path.exists(prog):
             Logs.error("print-introspected-doxygen has not been built yet."
                   " You need to build ns-3 at least once before "
                   "generating doxygen docs...")
        raise SystemExit(1)
```

```
                # Create a header file with the introspected information.
    doxygen_out = open(os.path.join('doc', 'introspected-doxygen.h'), 'w')
    if subprocess.Popen([prog], stdout=doxygen_out, env=proc_env).wait():
        raise SystemExit(1)
    doxygen_out.close()


    # Create a text file with the introspected information.
                text_out = open(os.path.join('doc', 'ns3-object.txt'), 'w')
    if subprocess.Popen([prog, '--output-text'], stdout=text_out,
env=proc_env).wait():
                                                        raise SystemExit(1)
    text_out.close()


    _getVersion()
    doxygen_config = os.path.join('doc', 'doxygen.conf')
    if subprocess.Popen(env['DOXYGEN'] + [doxygen_config]).wait():
                            Logs.error("Doxygen build returned an error.")
        raise SystemExit(1)




def _getVersion():
    """update the ns3_version.js file, when building documentation"""


    prog = "doc/ns3_html_theme/get_version.sh"
    if subprocess.Popen([prog]).wait() :
        Logs.error(prog + " returned an error")
        raise SystemExit(1)


                                  class Ns3DoxygenContext(Context.Context):
    """do a full build, generate the introspected doxygen and then the
doxygen"""
    cmd = 'doxygen'
    def execute(self):
        # first we execute the build
        bld = Context.create_context("build")
                    bld.options = Options.options # provided for convenience
                                                        bld.cmd = "build"
        bld.execute()
```

```
            _doxygen(bld)


class Ns3SphinxContext(Context.Context):
    """build the Sphinx documentation: manual, tutorial, models"""

    cmd = 'sphinx'


    def sphinx_build(self, path):
        print()
        print("[waf] Building sphinx docs for " + path)
        if subprocess.Popen(["make", "SPHINXOPTS=-N", "-k",
                             "html", "singlehtml", "latexpdf" ],
                            cwd=path).wait() :
            Logs.error("Sphinx build of " + path + " returned an error.")
            raise SystemExit(1)


    def execute(self):
                                                                _getVersion()
        for sphinxdir in ["manual", "models", "tutorial", "tutorial-pt-br"]
:
            self.sphinx_build(os.path.join("doc", sphinxdir))



                                        class Ns3DocContext(Context.Context):
    """build all the documentation: doxygen, manual, tutorial, models"""

    cmd = 'docs'


    def execute(self):
        steps = ['doxygen', 'sphinx']
        Options.commands = steps + Options.commands


def lcov_report(bld):
    env = bld.env


    if not env['GCOV_ENABLED']:
        raise WafError("project not configured for code coverage;"
                                           " reconfigure with --enable-gcov")
```

```
os.chdir(out)
try:
    lcov_report_dir = 'lcov-report'
                        create_dir_command = "rm -rf " + lcov_report_dir
    create_dir_command += " && mkdir " + lcov_report_dir + ";"


    if subprocess.Popen(create_dir_command, shell=True).wait():
        raise SystemExit(1)


    info_file = os.path.join(lcov_report_dir, 'report.info')
    lcov_command = "../utils/lcov/lcov -c -d . -o " + info_file
                                    lcov_command += " -b " + os.getcwd()
    if subprocess.Popen(lcov_command, shell=True).wait():
        raise SystemExit(1)


         genhtml_command = "../utils/lcov/genhtml -o " + lcov_report_dir
    genhtml_command += " " + info_file
    if subprocess.Popen(genhtml_command, shell=True).wait():
        raise SystemExit(1)
finally:
    os.chdir("..")
```