ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

ΥΛΟΠΟΙΗΣΗ MAC ΠΡΩΤΟΚΟΛΛΟΥ 802.15.4 ΓΙΑ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ

Βετουλαδίτης Μιλτιάδης

Μεταπτυχιακή Διπλωματική Εργασία

Νοέμβριος 2006

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή κο Αθανάσιο Κανάτα για τη βοήθεια και τις σωστές παρατηρήσεις που οδήγησαν στην ολοκλήρωση της διπλωματική μου εργασίας. Ευχαριστίες επίσης οφείλονται στον υποψήφιο Διδάκτορα κο Γιώργο Μπράβο για τη βοήθεια που μου παρείχε στους τομείς της προσομοίωσης και του ελέγχου των αποτελεσμάτων.

Τέλος θα ήθελα να εκφράσω τη ευγνωμοσύνη μου στους γονείς μου, στον αδελφό μου, στην κα Καμέλια Μιχόκ και στον κο Αντώνη Γκότση για την αμέριστη συμπαράσταση τους σε όλη μου την προσπάθεια.

ΣΚΟΠΟΣ

Η παρούσα διπλωματική εργασία έχει ως σκοπό την υλοποίηση των υπηρεσιών του υποεπιπέδου ελέγχου πρόσβασης στο μέσο (Medium Access Control - MAC), όπως αυτό περιγράφεται από το πρωτόκολλου IEEE 802.15.4, προκειμένου να ελεγχθεί η χρησιμότητα του στα ασύρματα δίκτυα αισθητήρων. Στο πρωτόκολλο 802.15.4 αναφέρονται όλες οι προδιαγραφές που θα πρέπει να έχουν τα ασύρματα δίκτυα προσωπικής περιοχής και χαμηλού ρυθμού μετάδοσης δεδομένων (Low-rate Wireless Personal Area Networks ή LR-WPANs). Σκοπός της εργασίας είναι η επέκταση του προγράμματος SENSASIM, το οποίο προσομοιώνει τη λειτουργία ενός εικονικού ασυρμάτου δικτύου αισθητήρων, προσθέτοντας τις απαραίτητες υπηρεσίες του υποεπιπέδου MAC. Με αυτόν τον τρόπο θα εξαχθούν τα απαραίτητα συμπεράσματα ώστε να ελεγχθεί η χρησιμότητα του πρωτοκόλλου στα WSNs (Wireless Sensor Networks) κυρίως σε θέματα κατανάλωσης ενέργειας.

Αρχικά θα αναφέρουμε για ποιο λόγο χρησιμοποιούμε τα δίκτυα PAN. Θα διασαφηνίσουμε τι ακριβώς είναι ένα δίκτυο αισθητήρων, από τι αποτελείται και ποιες οι εφαρμογές του στη καθημερινή ζωή του ανθρώπου. Στη συνέχεια θα αναπτύξουμε τις βασικές υλοποιήσεις και αρχιτεκτονικές που χρησιμοποιεί το πρόγραμμα SENSASIM ώστε να διαπιστωθούν ευκολότερα στη συνέχεια τα όποια προβλήματα ενσωμάτωσης του υποεπιπέδου MAC.

Το επόμενο βήμα είναι η επακριβής ανάλυση και λειτουργία του πρωτοκόλλου 802.15.4, καθώς και η αναφορά των πλεονεκτημάτων και των μειονεκτημάτων του. Ακολουθεί μια ενδεικτική αναφορά στα απαιτούμενα interfaces που πρέπει να υλοποιηθούν ενώ παράλληλα αναφέρονται όλες οι οντότητες, οι αναγκαίες μέθοδοι και τα μηνύματα που ανταλλάσουν οι υπηρεσίες των επιπέδων PHY (φυσικό επίπεδο) και MAC.

Σημαντική είναι και η προσομοίωση ενός ασυρμάτου δικτύου αισθητήρων με το πρόγραμμα NS2. Το πρόγραμμα αυτό ενσωματώνει ήδη αρκετές λειτουργίες του 802.15.4 και είναι χρήσιμο εργαλείο για την κατεύθυνση που θα πρέπει να έχει η ανάπτυξη της εφαρμογής.

Στη συνέχεια ακολουθεί η ανάπτυξη της εφαρμογής. Η γλώσσα που χρησιμοποιείται είναι η Java και περιλαμβάνει είτε προσθήκες και μετατροπές σε

ήδη υπάρχοντα κώδικα, είτε καινούριες κλάσεις. Θα ακολουθήσουν διάφορα παραδείγματα εφαρμογής, έχοντας ως στόχο να προσομοιώσουμε όσο το δυνατόν πιο ρεαλιστικά σενάρια λειτουργίας ασυρμάτων δικτύων αισθητήρων. Ο κώδικας που χρησιμοποιείται στη διπλωματική εργασία παρουσιάζεται συνολικά στα παραρτήματα Α και Β.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

Συντομογραφία	**Επεξήγηση**
ACK	Ackowledge
ACL	Access Control List
AODV	Ad-hoc On-demand Distance Vector
API	Application Programming Interface
BEBA	Binary Exponential Backoff Aloha
BO	Beacon Order
CAP	Contention Access Period
CCA	Clear Channel Assessment
CFP	Contention – Free Period
CSMA	Carrier Sense Multiple Access
ED	Energy Detection
ESB	Embedded Sensor Board
FFD	Full Function Device
GTS	Guaranteed Time Slots
IEEE	Institute of Electrical and Electronics Engineers
LLC	Logical Link Control
LQI	Link Quality Indication
LR-WPAN	Low Rate Wireless Personal Area Network
MAC	Medium Access Control
MCPS	MAC Common Part Sublayer
MEMS	Micro-Electro-Mechanical Systems
MFR	MAC Footer
MHR	MAC Header
MIC	Message Integrity Code
MLME	MAC Sublayer Management Entity
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
O-QPSK	Offset Quadrature Phase-Shift Keying
OSI	Open Systems Interconnection
PAN	Personal Area Network
PHY	Physical Layer
PIB	PAN Information Base
PLME	Physical Layer Management Entity
POS	Personal Operating Space
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
QoS	Quality of Service
RFD	Reduced Function Device
SAP	Service Access Point
SO	Superframe Order
SSCS	Service Specific Convergence Sublayer
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

1. ΕΙΣΑΓΩΓΗ

1.1 Προεπισκόπηση

Το πρωτόκολλο IEEE 802.15.4 είναι σχεδιασμένο για συσκευές μικρού ρυθμού μετάδοσης δεδομένων, μειωμένης κατανάλωσης ισχύος και μικρού κόστους. Με αυτόν τρόπο θα φέρει απλές και αυτόνομες συσκευές στον κόσμο της δικτύωσης, ανοίγοντας τις 'πόρτες' σε ένα πλήθος εφαρμογών ενώ παράλληλα θα αξιοποιηθούν ήδη υπάρχουσες. Τα προηγούμενα χρόνια η ασύρματη δικτύωση ήταν προνόμιο μόνο συστημάτων υψηλών ταχυτήτων που παρείχαν ένα ευρύ φάσμα εφαρμογών (π.χ. ηλεκτρονικοί υπολογιστές). Η προσπάθεια για αύξηση του ρυθμού δεδομένων φαίνεται εύκολα στο πρωτόκολλο IEEE 802.11 στο οποίο από 1-2 Mbps φτάσαμε στα 54 Mbps και μελλοντικά στα 540 Mbps.

Από τα πιο γνωστά πρωτόκολλα που υποστηρίζουν χαμηλού ρυθμού δεδομένα είναι το Bluetooth (IEEE 802.15.1). Παρόλαυτα αγωνίζεται σκληρά για την επιβίωση του στην αγορά κυρίως λόγω του Wi-Fi (IEEE 802.11b). Ένας λόγος που συμβαίνει αυτό είναι η εξάπλωση του Wi-Fi που είχε ως αποτέλεσμα την μείωση της τιμής του και την προσέγγιση της τιμής του Bluetooth. Ένας άλλος παράγοντας που συντέλεσε σε αυτό το φαινόμενο είναι η προσπάθεια ώστε το Bluetooth να καλύψει παραπάνω εφαρμογές από αυτές που είχε στην αρχή σχεδιαστεί και να τους προσφέρει QoS ξεφεύγοντας από την αρχική απλότητα σχεδίασης. Με αυτόν τον τρόπο όμως έγινε ακατάλληλο για απλές εφαρμογές με χαμηλό κόστος και χαμηλή κατανάλωση ισχύος. Ένα επιπλέον πρόβλημα του Bluetooth είναι η έλλειψη ευελιξίας στις τοπολογίες που ακολουθεί.

Καθώς λοιπόν καινούριες συσκευές, χαμηλού κόστους και υψηλής ποιότητας, κατακλύζουν την αγορά υπάρχει επιτακτική ανάγκη για χαμηλού κόστους – χαμηλής κατανάλωσης ισχύος δικτύωση. Τα δίκτυα που επικρατούν σε αυτούς τους τομείς είναι τα ασύρματα δίκτυα προσωπικής περιοχής ή κατά την χρησιμοποιούμενη ορολογία wireless personal area networks - WPANs. Οι δύο πιο σημαντικές προσπάθειες της IEEE είναι το 802.15.3a, που είναι γνωστό και ως δίκτυα εξαιρετικά ευρείας ζώνης – ultra wideband (UWB), και το IEEE 802.15.4 που αφορά τα WPANs χαμηλού ρυθμού μεταφοράς (LR-WPANs). Στον επόμενο

πίνακα παρουσιάζονται πιο απλά οι διαφορές των WLANs, WPANs και LR-WPANs.

Πίνακας 1.1 Διαφορές πρωτοκόλλων IEEE 802

	WLAN (802.11)	**Bluetooth-based WPAN (802.15.1)**	**Low-rate WPAN (802.15.4)**
Εμβέλεια	~100 m	~10 - 100 m	~10 m
Throughput	~2 - 11 Mbs	~1 Mbs	~0.25 Mbs
Κατανάλωση ισχύος	Medium	Low	Ultra low
Μέγεθος	Μεγάλο	Μικρό	Αρκετά μικρό
Κόστος / Πολυπλοκότητα	>6	1	0.2

Προκειμένου να πετύχουμε χαμηλό κόστος και χαμηλή κατανάλωση ισχύος, όπως περιγράφεται στο IEEE 802.15.4 κάνουμε τις εξής ρυθμίσεις:

- Μειώνουμε την ποσότητα της μεταδιδόμενης πληροφορίας
- Μειώνουμε το duty cycle του πομποδέκτη, καθώς και τη συχνότητα της μετάδοσης δεδομένων
- Μειώνουμε την πολυπλοκότητα
- Ελαττώνουμε την εμβέλεια
- Εφαρμόζουμε μηχανισμούς εξοικονόμησης ενέργειας όπως κατάσταση ‘ύπνου’ (Sleep mode).

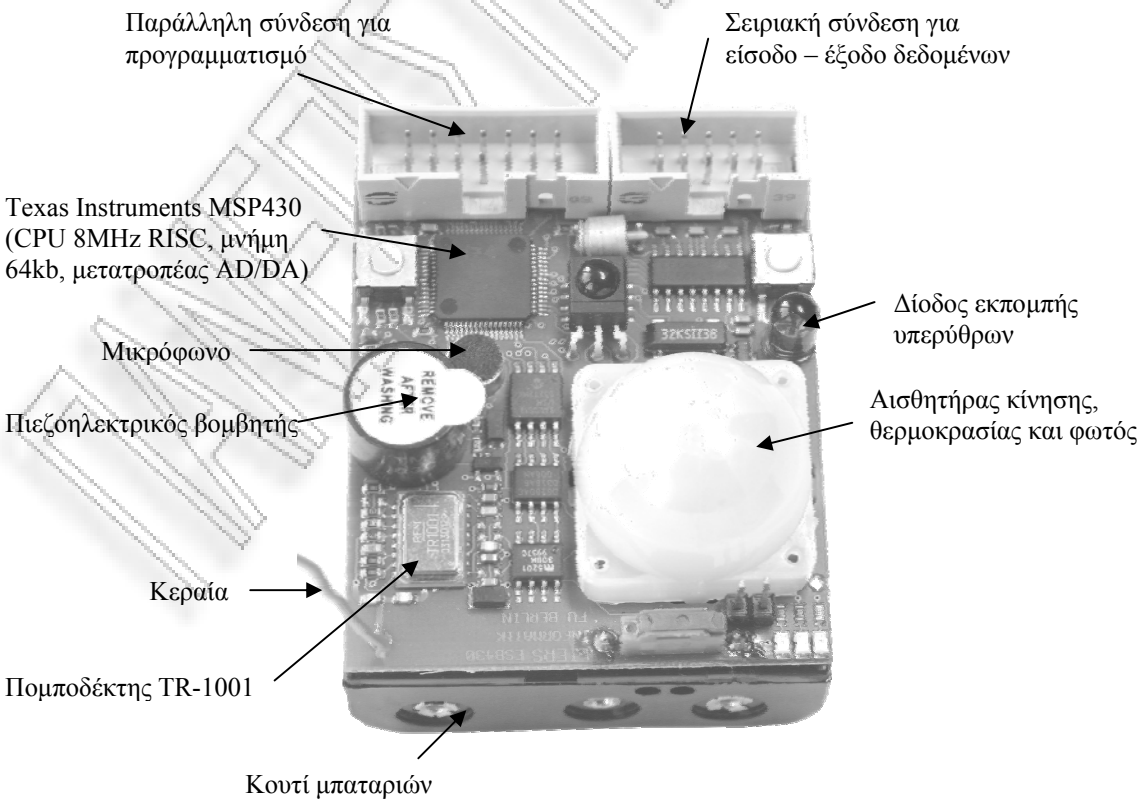
Είναι γεγονός ότι οι συσκευές που έχουν χαμηλό ρυθμό μετάδοσης βρίσκονται πιο κοντά στην καθημερινή μας ζωή. Το πρόβλημα αυτών των συσκευών είναι η έλλειψη εφικτής τεχνολογίας. Παρόλαυτα με τη βοήθεια του πρωτοκόλλου 802.15.4 και την πρόοδο στους μικροεπεξεργαστές, στα μικρο-ηλεκτρομηχανικά συστήματα (micro-electro mechanicals systems – MEMS), στις τεχνολογίες RF, οι συσκευές αυτές θα παίξουν σημαντικό ρόλο στη ζωή μας.

1.2 Κόμβος με Αισθητήρα

Το ασύρματο δίκτυο αισθητήρων είναι ένα δίκτυο υπολογιστικών συστημάτων που αποτελείται από αυτόνομες χωρικά κατανεμημένες συσκευές, οι οποίες

χρησιμοποιούν αισθητήρες προκειμένου να μετρήσουν και να καταχωρήσουν διάφορα φυσικά μεγέθη όπως: θερμοκρασία, ήχος, δόνηση, πίεση, κίνηση. Το δίκτυο αυτό, αρχικά αναπτύχθηκε για στρατιωτικές εφαρμογές όπως την επιτήρηση του πεδίου μάχης. Οι αισθητήρες ήδη χρησιμοποιούνται για περιβαλλοντικό ή οικιστικό έλεγχο, εφαρμογές υγείας, οικιακούς αυτοματισμούς και συγκοινωνιακό έλεγχο.

Σε ένα δίκτυο αισθητήρων ο κάθε κόμβος αποτελείται, εκτός από τον αισθητήρα, από έναν πομποδέκτη, έναν μικροελεγκτή και μια ενεργειακή πηγή (συνήθως μπαταρία) [13]. Το μέγεθος ενός αισθητήρα μπορεί να ποικίλει από το μέγεθος ενός κουτιού παπουτσιών μέχρι μιας μικροσκοπικής συσκευής. Αντίστοιχα και το κόστος ποικίλει ανάλογα με τη πηγή ενέργειας, την υπολογιστική ταχύτητα και το εύρος ζώνης. Στην επόμενη εικόνα παρουσιάζουμε ένα ενσωματωμένο σύστημα με αισθητήρα (Embedded Sensor Board – ESB) ο οποίος είναι κατασκευασμένος από την FU-Berlin και εκπέμπει στα 868 MHz. Το ESB είναι εξοπλισμένο με ένα chip της Texas Instruments με ενσωματωμένο επεξεργαστή, μνήμη και μετατροπέα AD/DA των 12-bit. Η ταχύτητα του ρολογιού είναι 8 MHz και περιλαμβάνει 64 KB μνήμη. Χρειάζεται μια μπαταρία 3 V, ενώ όταν βρίσκεται σε κατάσταση 'ύπνου' (sleep mode) η κατανάλωση είναι 1000 φορές μικρότερη.

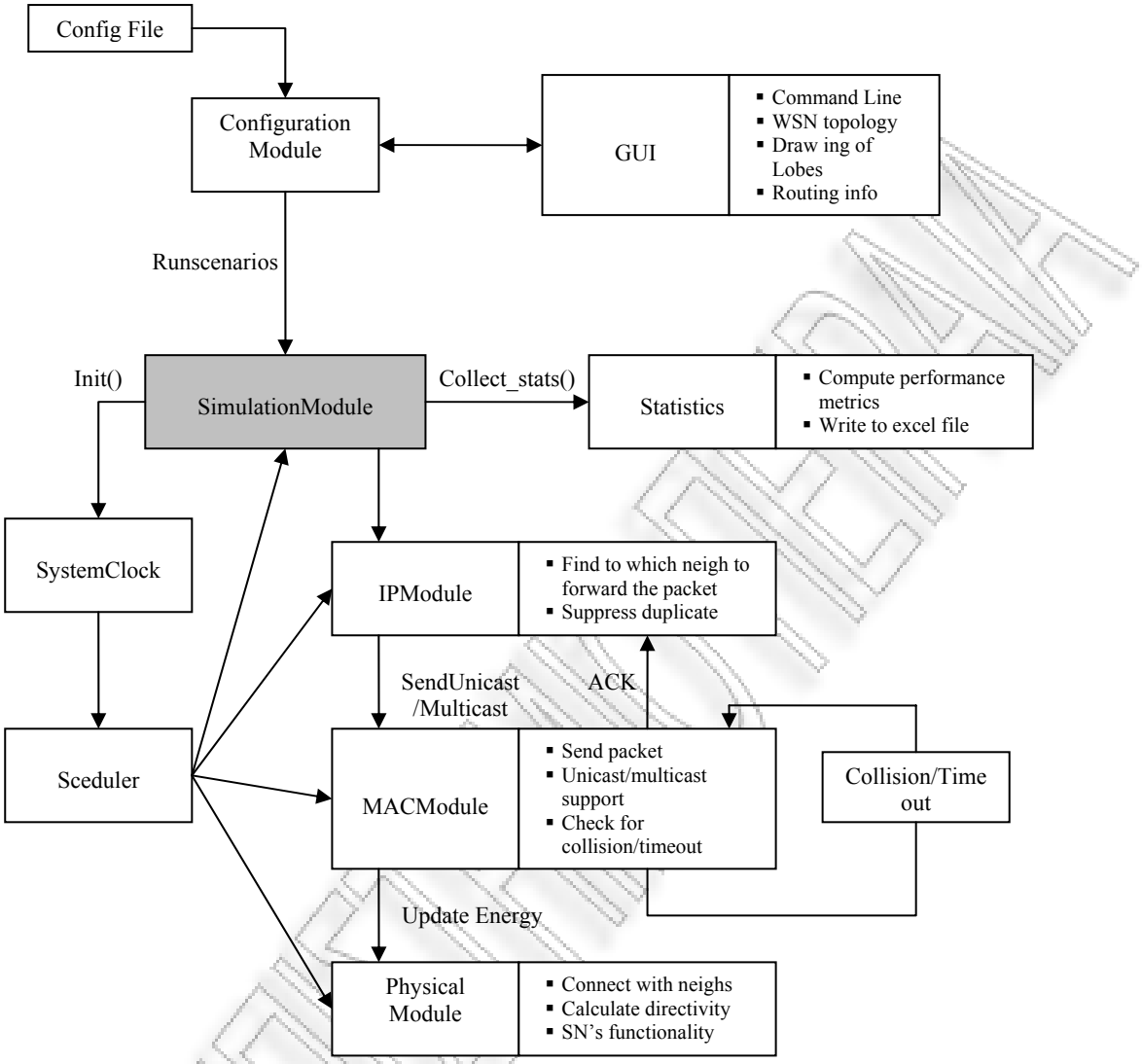


<u>Σχήμα 1.1</u> Σύστημα με αισθητήρα.

Οι εφαρμογές των ασύρματων δικτύων αισθητήρων είναι πάρα πολλές. Μικροί ασύρματοι κόμβοι σε ιατρικά θέματα πχ για την επιτήρηση ηλικιωμένων ανθρώπων. Άλλες συσκευές μπορούν να ενσωματωθούν στα ρούχα και να ελέγχουν την κατάσταση της υγείας του ατόμου που τα φορά έτσι ώστε αν προκύψει οποιοδήποτε πρόβλημα (πχ καρδιακή προσβολή) να καλείται ο γιατρός ή το ασθενοφόρο. Στο πεδίο των οικιακών αυτοματισμών μπορούμε να τοποθετήσουμε αισθητήρες σε κάθε δωμάτιο ώστε να συλλέγουν τη θερμοκρασία να δίνουν εντολές στους θερμοστάτες των κλιματιστικών μηχανημάτων. Ακόμα μπορούν να αναγνωρίζουν κίνηση και να ειδοποιούν τους ενοίκους του σπιτιού κατά την απουσία τους ενώ ταυτόχρονα να ενεργοποιούν το συναγερμό. Στο τομέα των επιχειρήσεων θα μπορούσαμε να κάνουμε έλεγχο του αποθέματος των προϊόντων, τοποθετώντας αισθητήρες στα πακέτα τους. Παράλληλα θα μπορούσαμε να ελέγξουμε αν τηρούνται οι απαιτούμενες τιμές θερμοκρασίας και υγρασίας. Σημαντικό αποτελεί το γεγονός, ότι με αυτόν τον τρόπο προστατεύονται τα προϊόντα από κλοπή.

1.3 Πρόγραμμα SENSASIM

Το Sensasim είναι ένας προσομοιωτής ασυρμάτων δικτύων, κατασκευασμένος εξολοκλήρου με τη γλώσσα προγραμματισμού Java [6]. Έχει υλοποιηθεί ξεχωριστά κάθε επίπεδο ενός WSN σαν μία κλάση Java. Αυτές οι κλάσεις επικοινωνούν μεταξύ τους χρησιμοποιώντας συγκεκριμένα interfaces. Στο sensasim μπορούν να αναπτυχθούν μέχρι 10000 κόμβοι (sensors) και λόγω του ότι η γλώσσα Java είναι αρκετά ‘βαριά’, κάθε γεγονός (όπως η μετάδοση πακέτου, η αναμονή για acknowledgement κτλ) μοντελοποιείται σαν ένα ανεξάρτητο νήμα (thread). Στη συνέχεια τα νήματα συγχρονίζονται, έτσι ώστε τα γεγονότα να γίνονται με τη σωστή χρονολογική σειρά. Το αντικείμενο το οποίο διαφυλάσσει την τελευταία λειτουργία που αναφέραμε είναι το scheduler. Στο επόμενο σχήμα παρουσιάζεται η αντικειμενοστραφής αρχιτεκτονική που ακολουθήθηκε.



Σχήμα1.2 Αρχιτεκτονική προσομοιωτή

Όσον αφορά το *φυσικό επίπεδο*, οι παράμετροι που χρησιμοποιεί ο προσομοιωτής είναι χωρισμένες σε τρεις κατηγορίες: basic, energy, directional antennas. Στην πρώτη κατηγορία οι πιο σημαντικές παράμετροι είναι οι: sensor name, position of the node, transmission speed, packet length. Στη δεύτερη κατηγορία οι σημαντικότερες παράμετροι είναι: initial energy, energy left, energy per transmitted packet, energy per received packet, processing energy. Όσον αφορά την τρίτη κατηγορία, κάθε κόμβος δημιουργεί έναν συγκεκριμένο αριθμό από κατευθυντικές δέσμες. Προκειμένου να γίνεται εξοικονόμηση της ενέργειας που καταναλώνεται ανά μετάδοση, η σχεδίαση έχει επικεντρωθεί στην κατευθυντικότητα (σε αντίθεση με το εύρος της μετάδοσης).

Τα πρωτόκολλα που χρησιμοποιούνται για να καθορισθεί ποιος έχει πρόσβαση σ' έναν δίαυλο πολλαπλής προσπέλασης ανήκουν σ' ένα υπόστρωμα του στρώματος ζεύξης δεδομένων που αποκαλείται MAC (Medium Access Control – Έλεγχος προσπέλασης στο μέσο μετάδοσης). Προκειμένου να υλοποιηθεί το *MAC επίπεδο*, ο προσομοιωτής περιλαμβάνει δύο εκδοχές του πρωτοκόλλου Aloha (μία απλή και την BEBA - Binary Exponential Backoff Aloha), καθώς και το πρωτόκολλο non-persistent (μη επίμονο) CSMA. Στο απλό ALOHA ακολουθείται η εξής διαδικασία: στέλνεται το πρώτο μήνυμα και ακολουθεί αναμονή κάποιας περιόδου πριν την επαναμετάδοση. Στο BEBA γίνεται αναμονή ακόμα και πριν την μετάδοση του πρώτου μηνύματος. Αυτή η έκδοση του πρωτοκόλλου είναι χρήσιμη σε περίπτωση multicast μεταδόσεων. Στο συγκεκριμένο βέβαια η αναμετάδοση των πακέτων δεν γίνεται άμεσα, αλλά περιμένουν μία τυχαία περίοδο που βασίζεται στον εκθετικό αλγόριθμο 'backoff'. Σύμφωνα με το πρωτόκολλο non-persistent CSMA, ο κόμβος που έχει στείλει δεδομένα ελέγχει το δίαυλο για να δει μήπως μεταδίδει κάποιος άλλος εκείνη τη στιγμή. Αν ο δίαυλος είναι απασχολημένος ο κόμβος περιμένει μέχρις ότου ελευθερωθεί ο δίαυλος. Αν συμβεί σύγκρουση, ο κόμβος περιμένει για κάποιο τυχαίο χρονικό διάστημα και μετά ξεκινάει πάλι. Αξίζει να τονιστεί ότι θεωρείται ότι κατά την αποστολή ACKs δεν δημιουργούνται συγκρούσεις.

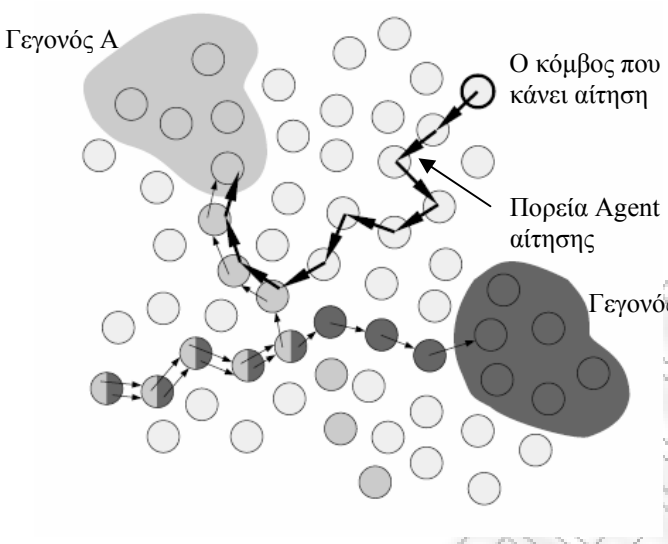
Το *network layer* αναλαμβάνει την δρομολόγηση των πακέτων. Πιο συγκεκριμένα αποτελείται από μία κλάση που έχει υλοποιημένους πολλούς αλγόριθμους δρομολόγησης: Broadcast Flooding (multicast), Normal Flooding, Directional Flooding, SPEED Broadcast (multicast), SPEED with duplicate suppression, SPEED with hole detection, SPEED with energy enhancement. Το επίπεδο αυτό επικοινωνεί με το MAC υποεπίπεδο με ένα ειδικά σχεδιασμένο interface. Η τοπολογία των sensors στο χώρο όπου αναπτύσσεται το δίκτυο επιλέγεται από τέσσερα είδη: τυχαία, τυχαία με κενό στο μέσο του χώρου, τυχαία με εξαιρετικά μεγάλο κενό στο μέσο του χώρου και τέλος σε grid (πλέγμα).

Κάθε σενάριο προσομοίωσης θα μοντελοποιείται και θα αρχικοποιείται μέσω μιας γραφικής διεπαφής (GUI). Το GUI χρησιμοποιείται προκειμένου ο χρήστης να παρακολουθήσει ή να διορθώσει τις παραμέτρους κάποιου σεναρίου. Αυτό μπορεί να γίνει και μέσω του αρχείου διαμόρφωσης παραμέτρων (configuration file). Ακόμη, το GUI προσφέρει τη δυνατότητα να δούμε την τοπολογία του δικτύου, την περιοχή κάλυψης κάθε κόμβου, καθώς και ποιοι κόμβοι είναι ενεργοί.

1.4 Ροή πληροφορίας

Η υλοποίηση των αιτήσεων και γενικότερα την αποστολής μηνυμάτων μεταξύ των κόμβων μπορεί να γίνει με τη ‘ροή’ των αιτήσεων ή των γεγονότων διαμέσου των κόμβων. Όταν το δίκτυο αισθανθεί να συμβαίνει μια πλειάδα από γεγονότα, αυτά δε θα κοινοποιηθούν στους κόμβους, ενώ θα ξεκινήσει η ροή των αιτήσεων. Ένα γεγονός αναμένεται να επηρεάσει μια ομάδα κόμβων. Σε αυτήν την ομάδα θα παραχθούν ένα ή περισσότερα πακέτα τα οποία θα κινούνται στο δίκτυο. Αυτά τα πακέτα ονομάζονται agents. Ο σκοπός ενός agent είναι καθώς κινείται στο δίκτυο, να ενημερώνει διάφορους κόμβους για το γεγονός που έχει πραγματοποιηθεί. Κάθε φορά που ένας agent εισέρχεται σε έναν καινούριο κόμβο, αφήνει τα ίχνη του στη βάση δεδομένων του κόμβου. Τα ίχνη αυτά αφορούν πόσα βήματα (θα χρησιμοποιήσουμε καλύτερα τον όρο hop) έχει διανύσει για να φτάσει και ποιο ήταν το αρχικό σημείο εκκίνησης. Οι κόμβοι οι οποίοι βρίσκονται στο πεδίο επιρροής του γεγονότος, έχουν μηδενικό μετρητή hop. Ο κινούμενος agent είναι στην πραγματικότητα ένα μικρό πακέτο που εμπεριέχει πληροφορία για το γεγονός.

Ένας κόμβος που κάνει αίτηση στέλνει έναν σχετικό agent προς μια αυθαίρετη κατεύθυνση. Καθώς κινείται τυχαία στο δίκτυο ελπίζει ότι αργά ή γρήγορα θα διασχίσει το μονοπάτι που ακολούθησε ένας agent γεγονότος. Όταν θα γίνει αυτό θα ελέγξει τη βάση δεδομένων σε κάθε κόμβο που επισκέπτεται για τις πληροφορίες που χρειάζεται. Σε περίπτωση που συναντήσει πρόβλημα, θα καταλάβει πως θα επιστρέψει αφού θα ακολουθήσει τα ‘ίχνη’ που άφησε στο πέρασμα του ο agent γεγονότος. Στο επόμενο σχήμα παρουσιάζουμε την παραπάνω διαδικασία. Ο κόμβος που κάνει αίτηση ξεκινάει την αναζήτηση του για το γεγονός Α. Ο agent αίτησης που αποστέλλει ο κόμβος, ταξιδεύει στο δίκτυο ακολουθώντας ένα τυχαίο μονοπάτι μέχρι να συναντήσει τον πρώτο κόμβο που περιέχει πληροφορίες σχετικά με το γεγονός Α. Στη συνέχεια ακολουθεί την αντίστροφη διαδρομή από αυτήν που ακολούθησε ο agent του γεγονότος Α.



Σχήμα 1.3 Κίνηση ενός agent.

1.5 Προβλήματα Υλοποίησης

Προβλήματα του πρωτοκόλλου 802.15.4

Θα αναφέρουμε κάποια σημαντικά προβλήματα που παρουσιάζει το πρωτόκολλο 802.15.4: Δεν υπάρχει προς το παρόν κάποια μέθοδος που να επιτρέπει στο IP να 'τρέχει' πάνω στο 802.15.4. Οι πληροφορίες που περιλαμβάνει το IP δεν μπορούν να προστεθούν σε ένα πλαίσιο του 802.15.4 διότι δε θα αφήσουν ελεύθερο χώρο για τα δεδομένα. Ακόμη δεν έχει διαπιστωθεί ότι όλοι οι μέθοδοι δρομολόγησης είναι κατάλληλοι με τα LR-WPANs. Υπάρχει έλλειψη ολοκληρωμένης διαχείρισης και επιλογής όλων των απαραίτητων ρυθμίσεων.

Προβλήματα με το πρόγραμμα Sensasim

Ένα από τα πρώτα θέματα που προκύπτουν μελετώντας το πρότυπο και συγκρίνοντας το με τους μηχανισμούς που υλοποιεί το sensasim, είναι ότι χρειάζεται να αναπτυχθεί και το φυσικό επίπεδο του 802.15.4 και ποιο συγκεκριμένα οι εξής υπηρεσίες: PHY data service και PHY management service. Η πρώτη για τη μεταφορά των MPDUs και η δεύτερη για τη μεταφορά των εντολών διαχείρισης μεταξύ MLME και PLME. Ακόμη, τα πακέτα πού δημιουργεί το

φυσικό επίπεδο πρέπει να προσαρμοστούν με τα πακέτα που δημιουργεί το πρότυπο (PPDUs).

Άλλο σημαντικό πρόβλημα είναι ότι τα υψηλότερα επίπεδα στο sensasim δεν έχουν σχεδιαστεί με τη λογική σχεδίασης των wireless PANs. Αυτό σημαίνει ότι δεν ορίζονται απαραίτητες παράμετροι όπως: η POS (η περιοχή όπου επικοινωνούν οι συσκευές που αποτελούν το PAN), δεν ορίζεται PAN coordinator. Κάτι άλλο πολύ σημαντικό, και το οποίο πρέπει να αναπτυχθεί στο πρόγραμμα, είναι ο ορισμός των τύπων των συσκευών (σύμφωνα με το πρότυπο) όπως αυτός αναφέρεται στην παράγραφο 2.1 σελίδα 15. Τα υψηλότερα επίπεδα του 802.15.4 έχουν αναπτυχθεί από την εταιρεία Zigbee. Τελικά, είναι ανάγκη να δημιουργηθούν κάποια interfaces τα οποία να διασυνδέουν ομαλά τα δύο επίπεδα.

2. IEEE 802.15.4 – LOW RATE WIRELESS PERSONAL AREA NETWORKS

2.1 Εισαγωγή

Ένα LR-WPAN είναι ένα χαμηλού κόστους δίκτυο επικοινωνίας που επιτρέπει την ασύρματη σύνδεση συσκευών με ελάχιστη κατανάλωση ισχύος. Τα βασικά χαρακτηριστικά ενός ενός LR-WPAN είναι [1]:

- Ρυθμός μετάδοσης δεδομένων 250,40 και 20 kbps
- Τοπολογίες αστέρα ή peer-to-peer
- 16-bit (short) ή 64-bit (extended) διευθύνσεις
- Guaranteed time slots (GTSs)
- CSMA-CA
- Πλήρες πρωτόκολλο για acknowledgments, για αξιόπιστη μεταφορά
- Χαμηλή κατανάλωση ισχύος
- Μηχανισμό ανίχνευση ενέργειας (Energy Detection – ED)
- Ένδειξη της ποιότητας της σύνδεσης
- 16 διαθέσιμα κανάλια στα 2450 MHz, 10 κανάλια στα 915 MHz, και 1 κανάλι στα 868 MHz.

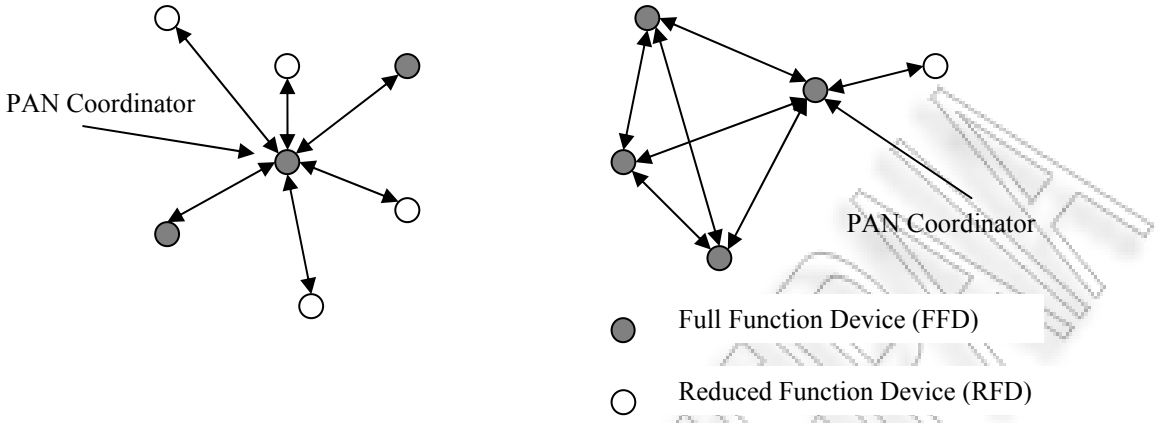
Σε ένα LR-WPAN θεωρείται ότι υπάρχουν δύο ειδών συσκευές: μία συσκευή με πλήρεις λειτουργίες (full function device – FFD) και μία με μειωμένες λειτουργίες (reduced function device – RFD). Μία FFD μπορεί να λειτουργήσει με τρεις τρόπους: ως ένας PAN συντονιστής (ο λεγόμενος PAN coordinator), ως ένας απλός coordinator ή ως μία απλή συσκευή. Μία FFD μπορεί να επικοινωνήσει τόσο με άλλα RFDs όσο και με άλλα FFDs. Όμως μία RFD μπορεί να επικοινωνήσει μόνο με μία FFD. Μία RFD προορίζεται για απλές εφαρμογές, σαν αυτές που κάνει ένας

sensor και δεν χρειάζεται να στέλνει μεγάλη ποσότητα δεδομένων. Αυτό έχει ως αποτέλεσμα μία RFD να πραγματοποιείται χρησιμοποιώντας ελάχιστες πηγές και μνήμη.

Να τονίσουμε ότι τα LR-WPANs μπορούν να λειτουργήσουν ικανοποιητικά σε εμβέλεια 10m. Ο χώρος αυτός ονομάζεται Personal Operating Space ή για συντομία POS. Υπάρχει περίπτωση να λειτουργήσουν και σε μεγαλύτερο εύρος μειώνοντας όμως τον ρυθμό μεταφοράς δεδομένων. Δύο ή περισσότερες συσκευές που βρίσκονται μέσα σε ένα POS και οι οποίες επικοινωνούν στο ίδιο φυσικό κανάλι αποτελούν ένα WPAN. Αυτό βέβαια προϋποθέτει ότι τουλάχιστον θα υπάρχει μία FFD, η οποία θα λειτουργεί σαν ένα συντονιστής του PAN.

2.2 Τοπολογίες Δικτύου

Όπως αναφέραμε και πιο πάνω, ένα LR-WPAN μπορεί να λειτουργεί σε δύο τοπολογίες: αστέρα και peer-to-peer. Οι τοπολογίες αυτές παρουσιάζονται στο σχήμα 2.1. Στην τοπολογία αστέρα η επικοινωνία εδραιώνεται μεταξύ των συσκευών και ενός κεντρικού ελεγκτή που καλείται PAN coordinator. Ένας PAN coordinator πέρα από την εφαρμογή που εκτελεί, πρέπει να εκτελεί και εργασίες όπως: εκκίνηση, τερματισμό και δρομολόγηση επικοινωνίας. Όλες οι συσκευές που βρίσκονται σε ένα τέτοιο δίκτυο εμπεριέχουν μία μοναδική διεύθυνση μήκους 64 bits. Στη συγκεκριμένη τοπολογία ο PAN coordinator πρέπει να είναι συνεχώς συνδεδεμένος με κάποια πηγή ενέργειας, ενώ οι υπόλοιπες συσκευές μπορούν να έχουν ως πηγή μπαταρίες.



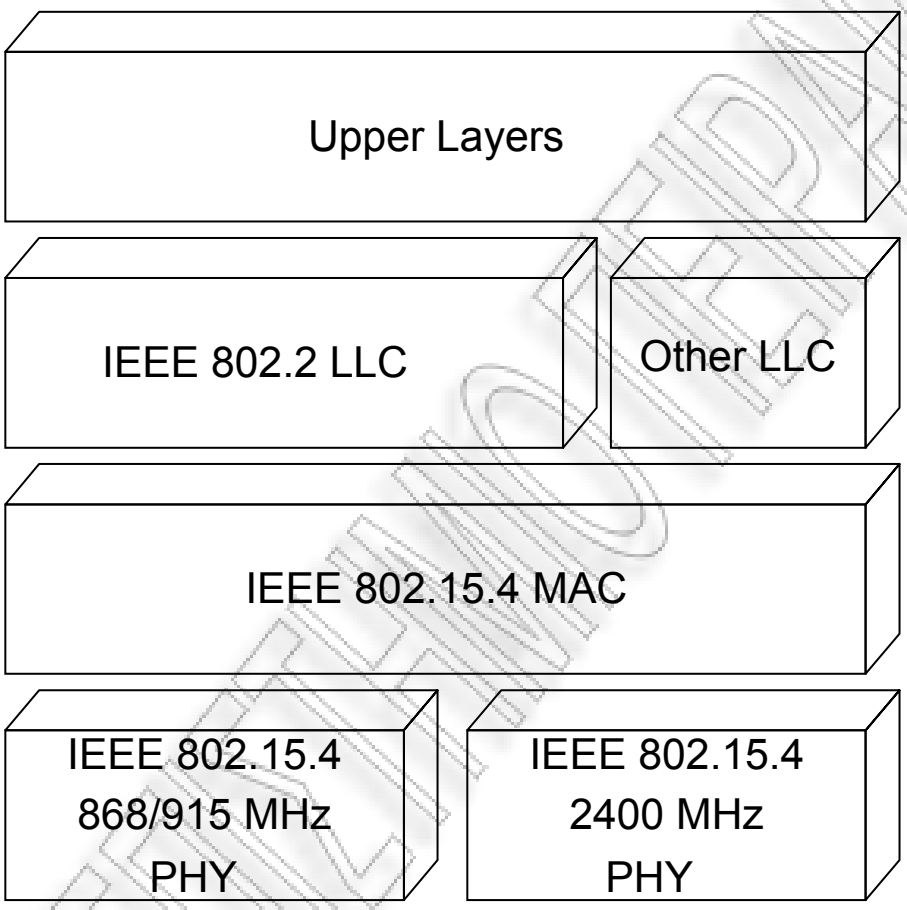
Σχήμα 2.1 Τοπολογίες αστέρα και peer-to-peer

Ο PAN coordinator υπάρχει και στην τοπολογία peer-to-peer. Η βασική διαφορά με την τοπολογία αστέρα είναι ότι όλες οι συσκευές μπορούν να επικοινωνούν μεταξύ τους, αρκεί να η μία να βρίσκεται στην εμβέλεια της άλλης. Η τοπολογία αυτή επιτρέπει σύνθετες υλοποιήσεις δικτύων και για αυτόν τον λόγο είναι κατάλληλη για Wireless Sensor Networks. Ένα δίκτυο σε αυτήν την μορφή μπορεί να είναι ad-hoc, να αυτοοργανώνεται και να αυτοθεραπεύεται. Μπορεί ακόμη να επιτρέπει τα πολλαπλά άλματα για τη δρομολόγηση των μηνυμάτων που ανταλλάσουν οι συσκευές. Να προσθέσουμε ακόμη, ότι κάθε PAN επιλέγει ένα μοναδικό προσδιοριστικό. Αυτό το προσδιοριστικό επιτρέπει την επικοινωνία μεταξύ των συσκευών ενός δικτύου χρησιμοποιώντας short addresses και επιτρέπει ακόμη τις μεταδόσεις μεταξύ συσκευών που βρίσκονται σε διαφορετικά δίκτυα.

2.3 Αρχιτεκτονική Δικτύου

Η αρχιτεκτονική ενός LR-WPAN [8] αποτελείται από επίπεδα (σχήμα 2.2) και βασίζεται στο OSI (open systems interconnection). Κάθε επίπεδο είναι υπεύθυνο για κάποιο μέρος του προτύπου και προσφέρει διάφορες υπηρεσίες προς τα υψηλότερα επίπεδα. Μια συσκευή που βρίσκεται σε ένα LR-WPAN περιλαμβάνει

το φυσικό επίπεδο (PHY), το οποίο εμπεριέχει τον πομποδέκτη RF καθώς και έναν χαμηλού επιπέδου μηχανισμό ελέγχου, και το MAC υποεπίπεδο το οποίο παρέχει πρόσβαση στο φυσικό κανάλι.



Σχήμα 2.2 Αρχιτεκτονική LR-WPAN

Το PHY παρέχει δύο υπηρεσίες: την υπηρεσία δεδομένων PHY και την υπηρεσία διαχείρησης PHY η οποία διασυνδέεται με το PLME (Physical Layer Management Entity). Η πρώτη επιτρέπει τη μετάδοση και την παραλαβή PPDUs (PHY Protocol Data Units) διαμέσου του φυσικού καναλιού. Άλλα χαρακτηριστικά του PHY είναι η ενεργοποίηση και η απενεργοποίηση του πομποδέκτη, η ανίχνευση ισχύος (ED), το LQI (Link Quality Indication), η επιλογή καναλιού και τέλος η αποδοχή και μετάδοση πακέτων.

Το MAC υποεπίπεδο παρέχει δύο υπηρεσίες: την υπηρεσία δεδομένων MAC και την υπηρεσία διαχείρισης MAC η οποία διασυνδέεται με την οντότητα MLME-SAP (MAC Sublayer Management Entity Service Access Point). Η πρώτη επιτρέπει τη μετάδοση και την παραλαβή MPDUs (MAC Protocol Data Units) προς την υπηρεσία δεδομένων PHY. Άλλα χαρακτηριστικά του MAC είναι η διαχείρηση των beacons (αναγνωριστικά σήματα), η πρόσβαση σε κανάλι, η διαχείριση των GTS (Guaranteed time slots), η επικύρωση των frames, η αναγνώριση παράδοσης των frames, η σύνδεση και η αποσύνδεση με το PAN.

Τα υψηλά επίπεδα αποτελούνται από το network layer, το οποίο παρέχει τη διαμόρφωση του δικτύου, τον χειρισμό και τη δρομολόγηση των μηνυμάτων και το επίπεδο εφαρμογών. Το LLC (Logical Link Control) μπορεί να προσεγγίσει το υποεπίπεδο MAC χρησιμοποιώντας το SSCS (Service Specific Convergence Sublayer).

Στον επόμενο πίνακα παρουσιάζονται συνοπτικά οι γενικές λειτουργίες ενός LR-WPAN. Στην πρώτη στήλη αναφέρεται η λειτουργία και στη δεύτερη στήλη, διάφοροι επιμέρους μηχανισμοί (δεν αναλύονται στη παρούσα αναφορά):

Πίνακας 2.1 Συνοπτικές λειτουργίες ενός LR-WPAN

Δομή superframe	Χρήση beacons
	Μεταξύ beacons υπάρχει η Contention Access Period – Χρήση μηχανισμού CSMA
	Χρήση guaranteed time slots
Μοντέλο μεταφοράς δεδομένων	Μεταφορά δεδομένων προς έναν coordinator
	Μεταφορά δεδομένων από έναν coordinator
	Peer-to-peer PANs – Χρήση μηχανισμού CSMA
Δομή frame	Beacon frame
	Data frame
	Acknowledgment frame
	MAC command frame
Αξιοπιστία στη μεταφορά δεδομένων	Μηχανισμός CSMA-CA
	Frame acknowledgment
	Επαλήθευση δεδομένων
Εκτιμήσεις κατανάλωσης ισχύος	Χρήση μπαταριών στις συσκευές
	Κατάσταση Sleep για αρκετό χρόνο μεταξύ πολλαπλών beacons
Ασφάλεια	Υπηρεσίες ασφαλείας (έλεγχος πρόσβασης, κρυπτογράφηση δεδομένων, ακεραιότητα frame, απόρριψη frames που επαναλήφθηκαν)
	Security modes: unsecured, ACL (Access Control List)

2.4 Συνοπτική Περιγραφή του MAC

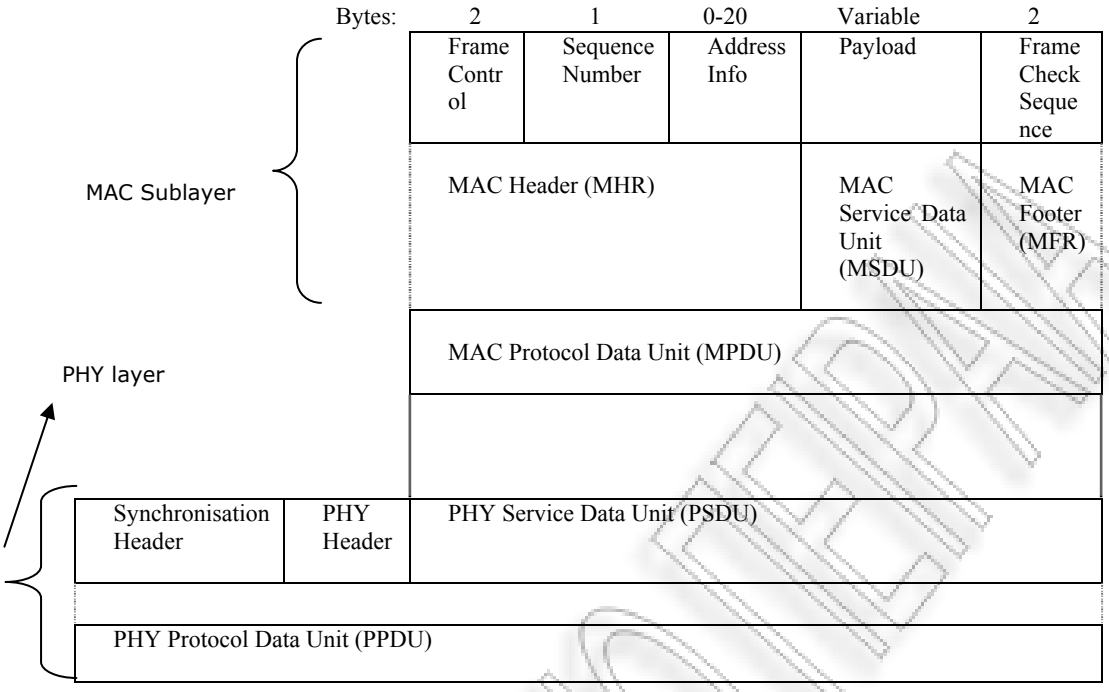
Υπάρχουν δύο μηχανισμοί για πρόσβαση στο κανάλι. Η πρόσβαση με τον contention-based μηχανισμό επιτρέπει στις συσκευές να χρησιμοποιούν έναν αλγόριθμο CSMA-CA για back off. Η πρόσβαση χωρίς contention μηχανισμό ελέγχεται πλήρως από τον PAN coordinator, ο οποίος χρησιμοποιεί τα GTSs.

Ακολουθεί η έναρξη ενός καινούριου PAN. Μια συσκευή ξεκινάει την ανίχνευση του καναλιού, ελέγχοντας αρχικά για όλα τα beacons (σήματα συγχρονισμού) που βρίσκονται στο POS (Personal Operating Space) ή εντοπίζει κάποιο συγκεκριμένο beacon το οποίο έχεο αποσυγχρονιστεί. Πριν ξεκινήσει το νέο PAN, τα

αποτελέσματα της ανίχνευσης καναλιού χρησιμοποιούνται για την επιλογή κατάλληλου λογικού καναλιού, αλλά και την επιλογή συγκεκριμένου προσδιοριστικού PAN. Υπάρχει η πιθανότητα το POS δύο διαφορετικών PAN να επικαλύπτεται. Για αυτό το πρόβλημα έχει προβλεφθεί συγκεκριμένη διαδικασία λύσης. Αφού πραγματοποιηθούν οι παραπάνω ενέργειες ένα FFD μπορεί να λειτουργήσει σαν PAN coordinator. Στο MAC περιγράφονται ακόμη οι μηχανισμοί που επιτρέπουν στις συσκευές να ενωθούν με ένα PAN ή να φύγουν από αυτό. Η διαδικασία ενσωμάτωσης περιγράφει τις συνθήκες κάτω από τις οποίες μία συσκευή μπορεί να ενωθεί στο PAN και τις συνθήκες που μπορεί ένας coordinator να επιτρέψει την ένωση αυτή. Υπάρχουν μηχανισμοί για την αποσύνδεση οι οποίοι ξεκινούν τη λειτουργία τους είτε από την συσκευή (προς θέλει να αποσυνδεθεί) είτε από τον coordinator.

Στη συνέχεια ένας coordinator δημιουργεί beacon frames προκειμένου να επιτευχθεί συγχρονισμός . Σε περίπτωση που δεν είναι ενεργοποιημένη η επιλογή για δημιουργία beacons, υπάρχει άλλος μηχανισμός για την απόκτηση συγχρονισμού στις μεταδόσεις. Άλλες διαδικασίες που υλοποιεί το MAC είναι η μετάδοση και αποδοχή frames αλλά και η αποστολή acknowledgments, καθώς και η αναμετάδοση frames. Η διάθεση και η απελευθέρωση των GTS μπορεί να προκαλέσει τεμαχισμό του διαστήματος που έχει ορισθεί για τα GTSs.

Η δομή ενός MAC frame είναι πολύ ευέλικτη προκειμένου να καλύψει τις ανάγκες για διαφορετικές εφαρμογές [10]. Η γενική μορφή ενός MAC frame παρουσιάζεται στο σχήμα 2.3. Η γενική ονομασία που έχει είναι MPDU (Mac protocol data unit) και αποτελείται από την επικεφαλίδα MAC – MAC header (MHR), το MSDU (MAC service data unit) και το MAC footer (MFR). Το MHR αποτελεί το πεδίο ελέγχου για το MAC frame. Σε αυτό εμφανίζεται ο τύπος του frame που μεταδίδεται, ρυθμίζει τη μορφή των διευθύνσεων και ελέγχει τα acknowledgements. Το μήκος του πεδίου – διεύθυνση κυμαίνεται μεταξύ 0 και 20 bytes. Για παράδειγμα, κάποιο frame δεδομένων μπορεί να περιέχει πληροφορίες και για την πηγή και για τον προορισμό, ενώ ένα acknowledgment frame δεν περιλαμβάνει καμία τέτοια πληροφορία. Από την άλλη πλευρά ένα beacon frame μπορεί να περιέχει μόνο τη διεύθυνση της πηγής. Επιπρόσθετα, μπορούν να χρησιμοποιηθούν 8-bit ή 64-bit διευθύνσεις.



Σχήμα 2.3 Η γενική δομή του MAC frame

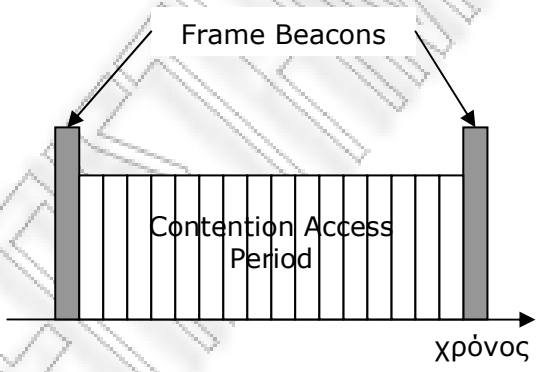
Το πεδίο payload έχει μεταβλητό μήκος. Τα δεδομένα που υπάρχουν στο Payload εξαρτώνται από τον τύπο του frame. Το MAC του προτύπου 802.15.4 έχει τέσσερις διαφορετικούς τύπους frame: beacon frame, data frame, acknowledgment frame, MAC command frame. Μόνο τα data και beacon frames περιέχουν πληροφορίες που στέλνονται από τα υψηλότερα επίπεδα. Τα acknowledgment και MAC command frames δημιουργούνται στο MAC.

2.5 Διαμόρφωση Δικτύου Peer-to-Peer

Σε ένα δίκτυο peer-to-peer, κάθε συσκευή μπορεί να επικοινωνήσει με οποιαδήποτε άλλη συσκευή που βρίσκεται στη σφαίρα επιρροής της. Μια συσκευή θα είναι υποψήφια για PAN coordinator όταν για παράδειγμα θα είναι η πρώτη που θα επικοινωνήσει στο κανάλι. Μπορούν να δημιουργηθούν περαιτέρω διαμορφώσεις δικτύου που βασίζονται στην τοπολογία peer-to-peer και αυτές στη συνέχεια να επιβάλλουν περιορισμούς στην τοπολογία του δικτύου.

2.6 Δομή Superframe

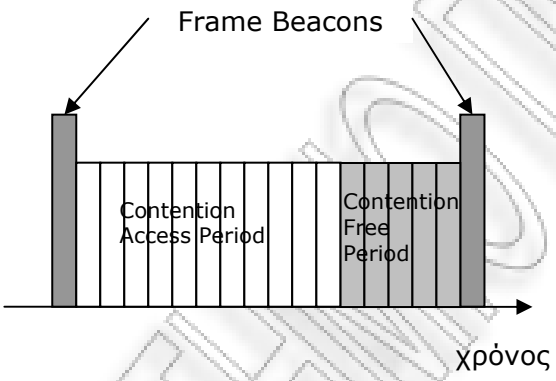
Το πρότυπο για τα LR-WPAN επιτρέπει την προαιρετική χρήση της δομής superframe (υπερ-πλαίσιο). Η μορφή ενός superframe ορίζεται από τον διαχειριστή (coordinator). Το superframe οριοθετείται από beacons, στέλνεται από το διαχειριστή και είναι χωρισμένο σε 16 θυρίδες (slots) ίσου μεγέθους. Ένα πλαίσιο – beacon μεταδίδεται στην πρώτη θυρίδα του κάθε Superframe. Αν ένας διαχειριστής δεν επιθυμεί να χρησιμοποιήσει δομή superframe μπορεί να σταματήσει τις μεταδόσεις των beacons. Τα beacons χρησιμοποιούνται ώστε να συγχρονίζονται οι συνδεμένες συσκευές, να γίνεται αναγνώριση του PAN και στην περιγραφή της δομής των superframes. Κάθε συσκευή που επιθυμεί να επικοινωνήσει κατά τη διάρκεια της CAP (Contention Access Period) μεταξύ δύο beacons θα αναμετρηθεί με τις άλλες συσκευές χρησιμοποιώντας ένα μηχανισμό CSMA-CA με χρονοθυρίδες. Όλες οι συναλλαγές θα πρέπει να έχουν πραγματοποιηθεί τη στιγμή που θα μεταδοθεί το επόμενο beacon.



Σχήμα 2.4 Δομή superframe χωρίς GTSs

Ένα superframe μπορεί να έχει ένα ενεργό και ένα μη ενεργό τμήμα. Κατά τη διάρκεια του μη ενεργού τμήματος, ο διαχειριστής δε θα αλληλεπιδρά με το PAN που ορίζει και θα μπορεί να εισέρχεται σε κατάσταση χαμηλής ισχύος. Όσον αφορά εφαρμογές μικρής λανθάνουσας κατάστασης ή εφαρμογές που απαιτούν συγκεκριμένο εύρος ζώνης δεδομένων, ο διαχειριστής PAN μπορεί να αφιερώσει τμήματα του ενεργού superframe σε αυτές τις εφαρμογές. Αυτά τα τμήματα ονομάζονται GTSs (Guaranteed Time Slots). Τα GTSs αποτελούν την περίοδο CFP (Contention – Free Period), η οποία εμφανίζεται πάντα στο τέλος του ενεργού

superframe, ξεκινώντας με τη θυρίδα που ακολουθεί το πέρας της CAP περιόδου. Ο διαχειριστής PAN μπορεί να αναθέτει μέχρι επτά GTSs και κάθε GTS μπορεί να απασχολεί περισσότερες από μία χρονοθυρίδες. Παρόλαυτα, ένα συγκεκριμένο τμήμα της περιόδου CAP θα παραμένει για πρόσβαση άλλων δικτυακών συσκευών ή συσκευών που θέλουν να ενσωματωθούν στο δίκτυο. Όλες αυτές οι διεργασίες θα πρέπει να έχουν διεκπεραιωθεί πριν την έναρξη της περιόδου CFP. Ακόμα, κάθε συσκευή που θα μεταδίδει σε κάποιο GTS θα πρέπει να εξασφαλίσει ότι η συναλλαγή ολοκληρώθηκε πριν από την έναρξη του επόμενου GTS ή του τέλους του CFP.



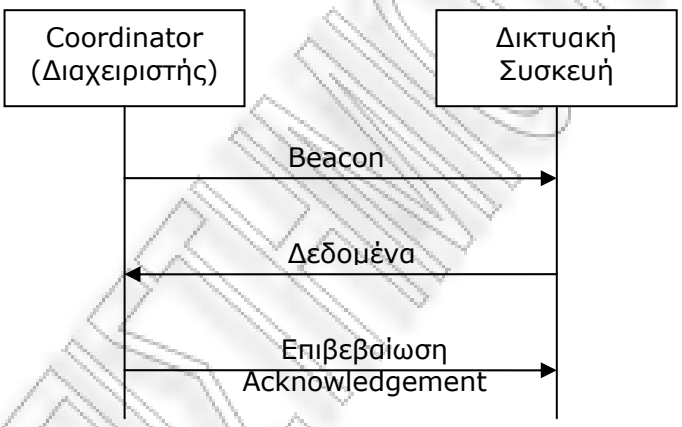
Σχήμα 2.5 Δομή superframe με GTSs

2.7 Μοντέλο μεταφοράς δεδομένων

Υπάρχουν τρεις τύποι μεταφοράς δεδομένων. Ο πρώτος τύπος αφορά στη μεταφορά δεδομένων από τη συσκευή στο διαχειριστή. Στο δεύτερο τύπο ο διαχειριστής στέλνει δεδομένα προς μία συσκευή. Τέλος υπάρχει η μεταφορά δεδομένων μεταξύ δύο ομότιμων συσκευών. Σε ένα δίκτυο peer-to-peer τα δεδομένα μπορούν να σταλούν μεταξύ δύο οποιονδήποτε συσκευών του δικτύου. Αυτό έχει ως αποτέλεσμα να μπορούν να χρησιμοποιηθούν όλες οι παραπάνω συναλλαγές που αναφέρθηκαν.

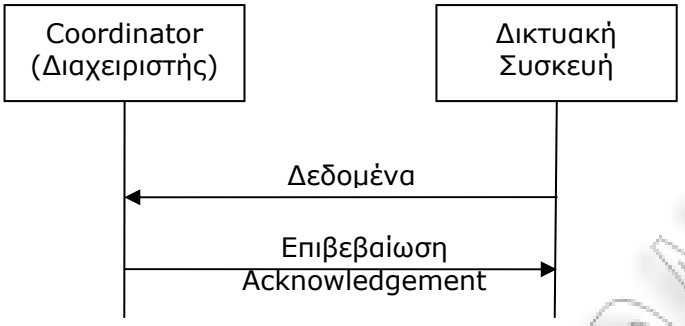
Τα δεδομένα μεταφέρονται προς ένα διαχειριστή

Όταν μια συσκευή, που αποτελεί μέρος ενός δικτύου χρησιμοποιεί beacons, επιθυμεί να μεταφέρει δεδομένα σε έναν διαχειριστή αρχικά 'ακούει' για beacons. Όταν βρεθεί κάοιο beacon, η συσκευή συγχρονίζεται με τη δομή superframe. Σε κάποιο συγκεκριμένο χρονικό σημείο, η συσκευή μεταδίδει το πλαίσιο δεδομένων προς το διαχειριστή, χρησιμοποιώντας τον μηχανισμό CSMA-CA με θυρίδες (slotted). Ο διαχειριστής αποδέχεται την επιτυχή είσοδο πλαισίων και μεταδίδει (προαιρετικά) ένα πλαίσιο επιβεβαίωσης (acknowledgement frame). Με αυτόν τον τρόπο η συναλλαγή ολοκληρώνεται. Η αλληλουχία των γεγονότων παρουσιάζεται στο σχήμα.



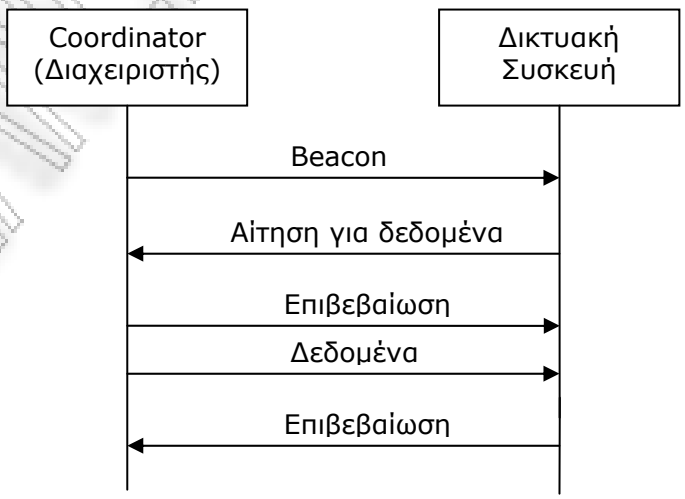
Σχήμα 2.6 Επικοινωνία με έναν διαχειριστή με τη χρήση beacons.

Όταν μια συσκευή θέλει να μεταφέρει δεδομένα σε ένα δίκτυο που δεν έχει ενεργοποιημένο τον μηχανισμό με τα beacons, μεταδίδει τα δεδομένα προς το διαχειριστή χρησιμοποιώντας το μηχανισμό CSMA-CA χωρίς θυρίδες (unslotted). Ο διαχειριστής αποδέχεται την επιτυχή είσοδο πλαισίων και μεταδίδει (προαιρετικά) ένα πλαίσιο επιβεβαίωσης (acknowledgement frame).



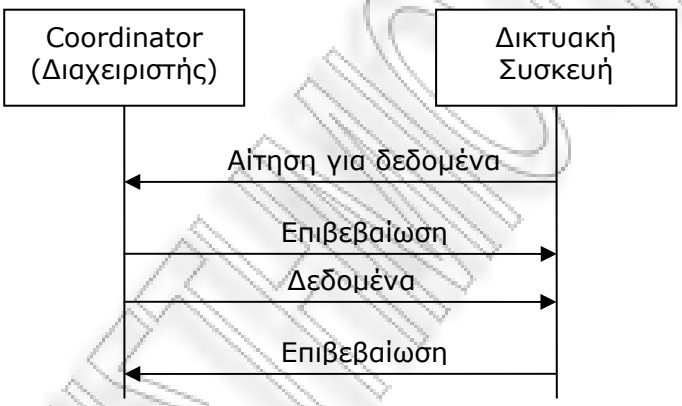
Σχήμα 2.7 Επικοινωνία με έναν διαχειριστή χωρίς τη χρήση beacons. Τα δεδομένα μεταφέρονται από ένα διαχειριστή

Όταν ένας διαχειριστής επιθυμεί να μεταφέρει δεδομένα σε μια συσκευή με τη χρήση beacons, υποδεικνύει στο beacon ότι κάποιο μήνυμα δεδομένων εκκρεμεί. Η συσκευή 'ακούει' περιοδικά για μηνύματα τύπου beacon και εφόσον κάποιο μήνυμα εκκρεμεί, μεταδίδει μια εντολή MAC και ζητάει τα δεδομένα χρησιμοποιώντας το μηχανισμό CSMA-CA με θυρίδες. Ο διαχειριστής αποδέχεται την επιτυχή είσοδο της αίτησης δεδομένων και στέλνει ένα πλαίσιο επιβεβαίωσης (acknowledgement frame). Στη συνέχεια το πλαίσιο δεδομένων που εκκρεμούσε, στέλνεται και αυτό χρησιμοποιώντας το μηχανισμό CSMA-CA με θυρίδες. Η συσκευή αποδέχεται την επιτυχή είσοδο του πλαισίου δεδομένων και στέλνει ένα πλαίσιο επιβεβαίωσης. Όταν ληφθεί το πλαίσιο επιβεβαίωσης, το μήνυμα διαγράφεται από τη λίστα των εκκρεμών μηνυμάτων.



Σχήμα 2.8 Επικοινωνία από έναν διαχειριστή προς μία συσκευή με τη χρήση beacons.

Όταν ένας διαχειριστής επιθυμεί να μεταφέρει δεδομένα σε μια συσκευή χωρίς τη χρήση beacons, αποθηκεύει τα δεδομένα μέχρι μια συσκευή να κάνει αίτηση για τα δεδομένα. Μια συσκευή δημιουργεί μια επαφή μεταδίδοντας μια εντολή MAC και ζητώντας με αυτόν τον τρόπο τα δεδομένα. Ο διαχειριστής αποδέχεται την επιτυχή είσοδο της αίτησης δεδομένων και στέλνει ένα πλαίσιο επιβεβαίωσης. Αν εκκρεμούν δεδομένα, ο διαχειριστής μεταδίδει το πλαίσιο δεδομένων το μηχανισμό CSMA-CA χωρίς θυρίδες. Εάν δεν εκκρεμούν δεδομένα ο διαχειριστής στέλνει ένα πλαίσιο δεομένων με μηδενικό ωφέλιμο φορτίο. Η συσκευή αποδέχεται την επιτυχή είσοδο των δεδομένων και στέλνει ένα πλαίσιο επιβεβαίωσης. Η συναλλαγή ολοκληρώνεται.



Σχήμα 2.9 Επικοινωνία από έναν διαχειριστή προς μία συσκευή χωρίς τη χρήση beacons.

Μεταφορά δεδομένων peer-to-peer

Σε ένα PAN με τοπολογία peer-to-peer, κάθε συσκευή επικοινωνεί με μια άλλη στη σφαίρα της ραδιο-επιρροής της. Οι συσκευές που επιθυμούν να επικοινωνήσουν θα χρειαστεί είτε να λαμβάνουν ταυτόχρονα είτε να συγχρονίσει η μία την άλλη. Στην πρώτη περίπτωση η συσκευή μπορεί απλά να μεταδώσει τα δεδομένα της χρησιμοποιώντας το μηχανισμό CSMA-CA χωρίς θυρίδες. Στη δεύτερη περίπτωση χρειάζονται ειδικές μετρήσεις ώστε να επιτευχθεί συγχρονισμός.

2.8 Δομή Πλαισίου

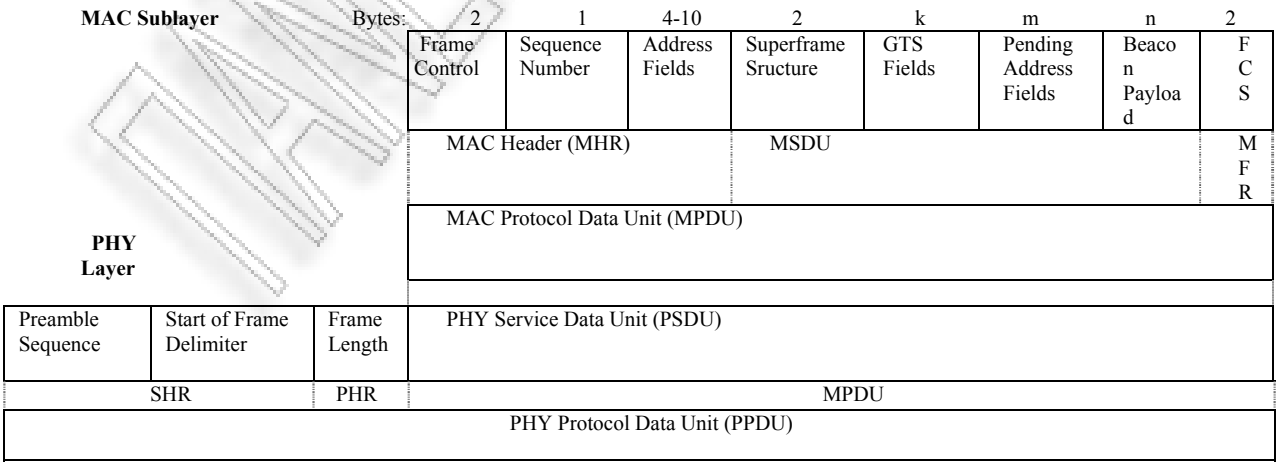
Η δομή των πλαισίων σχεδιάστηκε έτσι ώστε να διατηρείται η πολυπλοκότητα στο ελάχιστο ενώ την ίδια στιγμή η μετάδοση να είναι αρκετά ανθεκτική σε θορυβώδη κανάλια. Τα δίκτυα LR-WPAN προσδιορίζουν τέσσερις δομές πλαισίων:

- Πλαίσιο beacon (beacon frame), ο διαχειριστής μεταδίδει beacons
- Πλαίσιο δεδομένων, για μεταφορά δεδομένων
- Πλαίσιο επιβεβαίωσης (acknowledgement frame), για επιβεβαίωση επιτυχούς λήψης κάποιου πλαισίου
- Πλαίσιο εντολής MAC, για τη διαχείριση των MAC οντοτήτων

Η δομή του κάθε είδους πλαισίου παρουσιάζεται αναλυτικότερα παρακάτω.

Πλαίσιο beacon

Το επόμενο σχήμα παρουσιάζει τη δομή του πλαισίου beacon, το οποίο παράγεται από το υποεπίπεδο MAC. Ένας διαχειριστής μπορεί να στείλει τέτοιου είδους πλαίσια μόνο σε ένα δίκτυο που έχει ενεργοποιημένα τα beacons. Το MSDU (MAC service data unit) περιλαμβάνει τα εξής: προδιαγραφές Superframe, προδιαγραφές εκκρεμών διευθύνσεων, λίστα διευθύνσεων, και ωφέλιμα πεδία. Περικλείεται από τα MHR (MAC header) και MFR (MAC footer)

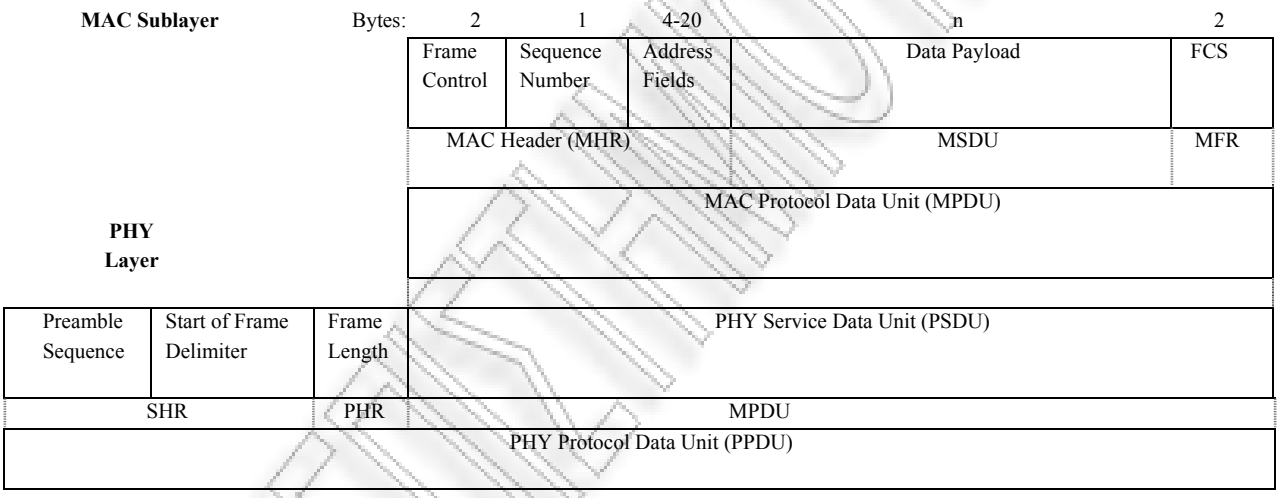


Σχήμα 2.10 Σχηματική απεικόνιση ενός πλαισίου beacon

Τα MHR, MSDU και MFR αποτελούν το πλαίσιο beacon του MAC (MPDU). Το MPDU στη συνέχεια περνάει στο φυσικό επίπεδο σαν το ωφέλιμο φορτίο ενός πλαισίου beacon του PHY (PSDU). Τα SHR, PHR και PSDU μαζί αποτελούν το πλαίσιο beacon του PHY (PPDU).

Πλαίσιο δεδομένων

Το επόμενο σχήμα παρουσιάζει τη δομή του πλαισίου δεδομένων, το οποίο παράγεται από το υψηλότερα επίπεδα.

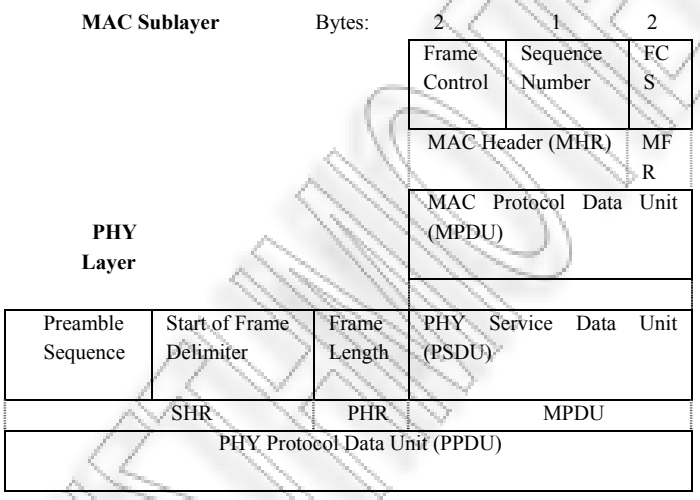


Σχήμα 2.11 Σχηματική απεικόνιση ενός πλαισίου δεδομένων

Τα ωφέλιμα δεδομένα περνούν προς το υποεπίπεδο MAC και αναφέρονται ως MSDU. Τα MHR, MSDU και MFR αποτελούν το πλαίσιο δεδομένων του MAC (MPDU).). Το MPDU στη συνέχεια περνάει στο φυσικό επίπεδο σαν το ωφέλιμο φορτίο ενός πλαισίου δεδομένων του PHY (PSDU). Τα SHR, PHR και PSDU μαζί αποτελούν το πλαίσιο δεδομένων του PHY (PPDU).

Πλαίσιο επιβεβαίωσης

Το επόμενο σχήμα παρουσιάζει τη δομή του πλαισίου επιβεβαίωσης, το οποίο παράγεται από το υποεπίπεδο MAC. Το πλαίσιο αυτό κατασκευάζεται από τα MHR και MFR. Τα MHR και MFR αποτελούν το πλαίσιο επιβεβαίωσης του MAC (MPDU). Το MPDU στη συνέχεια περνάει στο φυσικό επίπεδο σαν το ωφέλιμο φορτίο ενός πλαισίου επιβεβαίωσης του PHY (PSDU). Τα SHR, PHR και PSDU μαζί αποτελούν το πλαίσιο επιβεβαίωσης του PHY (PPDU).

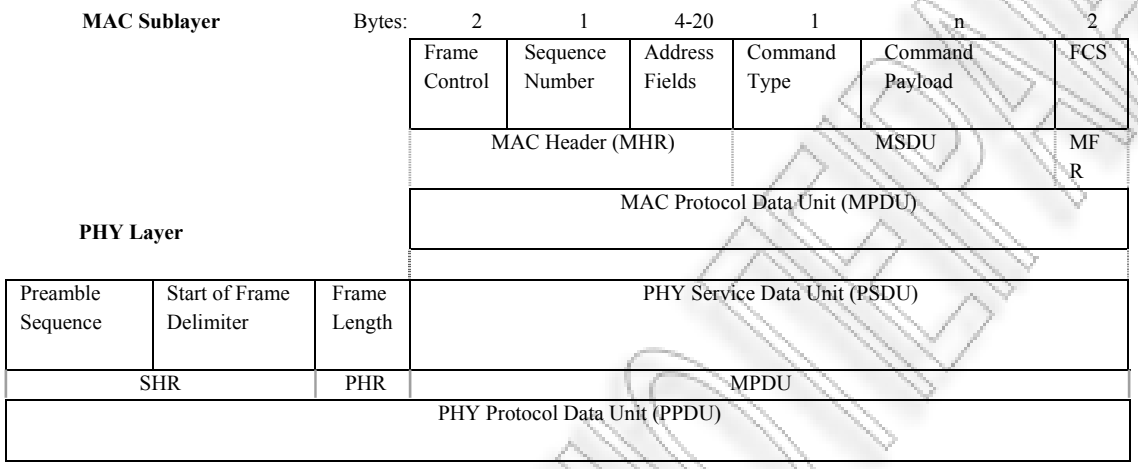


Σχήμα 2.12 Σχηματική απεικόνιση ενός πλαισίου επιβεβαίωσης

Πλαίσιο εντολής MAC

Το επόμενο σχήμα παρουσιάζει τη δομή του πλαισίου εντολής MAC, το οποίο παράγεται από το υποεπίπεδο MAC. Το MSDU περιλαμβάνει τα εξής: το πεδίο τύπου εντολής, και τα δεδομένα που προσδιορίζουν την εντολή και που ονομάζονται ωφέλιμο φορτίο εντολής. Περικλείεται από τα MHR (MAC header) και MFR (MAC footer). Το MHR περιλαμβάνει τα εξής: MAC frame control, data sequence number, addressing fields. Το MHR αποτελείται από ένα 16-bit FCS. . Τα MHR, MSDU και MFR αποτελούν το πλαίσιου εντολής MAC. Το MPDU στη συνέχεια περνάει στο φυσικό επίπεδο σαν το ωφέλιμο φορτίο ενός πλαισίου

εντολής MAC του PHY (PSDU). Τα SHR, PHR και PSDU μαζί αποτελούν το πλαίσιο εντολής MAC του PHY (PPDU).



Σχήμα 2.13 Σχηματική απεικόνιση ενός πλαισίου εντολής MAC

2.9 Επιβεβαίωση πλαισίου

Η επιτυχής παραλαβή και επικύρωση των δεδομένων ή του πλαισίου εντολής MAC μπορεί να επιβεβαιωθεί προαιρετικά με ένα μήνυμα επιβεβαίωσης. Αν για οποιοδήποτε λόγο η συσκευή – δέκτης είναι ανήμπορη να χειριστεί ή να λάβει το πλαίσιο, το μήνυμα δεν επιβεβαιώνεται. Αν ο δημιουργός του πλαισίου δεν λάβει την επιβεβαίωση μέσα σε συγκεκριμένο χρόνο, αναμεταδίδει το πλαίσιο.

2.10 Θέματα εξοικονόμησης ενέργειας

Το πρωτόκολλο αυτό θα χρησιμοποιηθεί από πολλές συσκευές που λαμβάνουν ενέργεια από μπαταρίες. Αυτό σημαίνει ότι η αλλαγή των μπαταριών σε τακτά διαστήματα είναι αδύνατη. Οι συσκευές αυτές, προκειμένου να ελαττώσουν την

κατανάλωση ενέργειας θα χρειαστούν duty-cycling. Τον περισσότερο χρόνο θα βρίσκονται σε κατάσταση ‘ύπνου’ (sleep state), ενώ ταυτόχρονα θα ακούν περιοδικά τη ραδιοσυχνότητα σε περίπτωση που εκκρεμεί κάποιο μήνυμα. Αυτός ο μηχανισμός δίνει τη δυνατότητα στο σχεδιαστή εφαρμογών, να αποφασίσει αν θα θέλει εξοικονόμηση ενέργειας ή ασφάλεια στα μηνύματα. Οι συσκευές που τροφοδοτούνται συνεχώς από κάποια πηγή θα έχουν τη δυνατότητα να ακούν συνεχώς τη ραδιοσυχνότητα.

2.11 Ασφάλεια

Το πρωτόκολλο έχει ως στόχο να χρησιμοποιηθεί σε διαφορετικές συσκευές, οπότε θέτονται διάφοροι περιορισμοί στα θέματα ασφαλείας του υποστρώματος MAC. Σαν βασικό εργαλείο είναι η δυνατότητα να υπάρχει μια λίστα ελέγχου πρόσβασης (ACL – Access Control List), καθώς και η χρήση συμμετρικής κρυπτογραφίας προκειμένου να προστατευθούν τα μεταδιδόμενα πλαίσια. Αυτό δε σημαίνει όμως ότι αυτοί οι μηχανισμοί ασφαλείας θα χρησιμοποιούνται παντού και πάντα. Το ζήτημα αυτό θα καθορίζεται από τα υψηλότερα επίπεδα, τα οποία θα προμηθεύουν το υπόστρωμα MAC με όλα απαραίτητα κλειδιά ασφαλείας. Αυτό σημαίνει ότι τα υψηλότερα στρώματα παρέχουν λειτουργίες όπως: διαχείριση κλειδιών, διακρίβωση συσκευής και προστασία.

Έλεγχος πρόσβασης: πρόκειται για την υπηρεσία που παρέχει τη δυνατότητα σε μια συσκευή να επιλέξει άλλες συσκευές πρόθυμες για επικοινωνία. Αν παρέχεται έλεγχος πρόσβασης, η κάθε συσκευή θα συγκροτήσει μια λίστα συσκευών (ACL) από τις οποίες θα αναμένει να λάβει πλαίσια.

Κρυπτογράφηση δεδομένων: πρόκειται για την υπηρεσία που χρησιμοποιεί έναν συμμετρικό αλγόριθμο ώστε να προστατεύσει τα δεδομένα από μη επιτρεπτή ανάγνωση (συσκευών που δεν έχουν το κρυπτογραφικό κλειδί). Μια ομάδα συσκευών θα μπορεί να κρυπτογραφήσει τα δεδομένα χρησιμοποιώντας ένα κλειδί, το οποίο θα το μοιράζουν μεταξύ τους. Η υπηρεσία αυτή θα μπορεί να παρέχεται στο ωφέλιμο μέρος των beacons, των δεδομένων και των εντολών MAC.

Ακεραιότητα πλαισίου: η υπηρεσία αυτή χρησιμοποιεί έναν κώδικα ακεραιότητας μηνύματος (MIC – message integrity code) προκειμένου να προστατέψει τα δεδομένα από το να αλλοιωθούν λόγω της επέμβασης συσκευών που δεν κατέχουν

το κρυπτογραφικό κλειδί. Επιπρόσθετα παρέχει εγκυρότητα στο αν στάλθηκαν τα δεδομένα από συσκευή που κατέχει το κρυπτογραφικό κλειδί. Η ακεραιότητα παρέχεται σε όλα τα είδη πλαισίου.

Διαδοχικός έλεγχος: πρόκειται για την υπηρεσία που χρησιμοποιεί μια συγκεκριμένη ακολουθία εισόδων προκειμένου να απορριφθούν πλαίσια που επαναλαμβάνονται. Όταν μια συσκευή δέχεται ένα πλαίσιο, συγκρίνεται η τιμή ελέγχου με τη προγενέστερη τιμή ελέγχου. Αν η τιμή αυτή είναι νεότερη από την προγενέστερη τιμή, ο έλεγχος πετυχαίνει και ανανεώνεται η τιμή. Αν η τιμή δεν είναι νεότερη ο έλεγχος αποτυγχάνει.

2.12 Προδιαγραφές Φυσικού Επιπέδου

Στο κεφάλαιο αυτό αναλύουμε τις προδιαγραφές του φυσικού επιπέδου. Το φυσικό επίπεδο (για λόγους συντομίας θα το αναφέρουμε PHY) είναι υπεύθυνο τα εξής διεργασίες:

- Ενεργοποίηση / Απενεργοποίηση του ραδιο-πομποδέκτη
- Ανίχνευση ενέργειας στο κανάλι μετάδοσης
- LQI (link quality indication) – ένδειξη ποιότητας σύνδεσης
- CCA (clear channel assessment) – σαφής αξιολόγηση του καναλιού
- Επιλογή συχνότητας καναλιού
- Μετάδοση και υποδοχή δεδομένων

Απαιτήσεις και ορισμοί

Μια οποιαδήποτε συσκευή μπορεί να λειτουργεί σε μία ή περισσότερες ζώνες συχνοτήτων χρησιμοποιώντας τις παρακάτω διαμορφώσεις και παραμέτρους διάδοσης (πίνακας)

Πίνακας 2.2 Ζώνες συχνοτήτων και παράμετροι δεδομένων

PHY (MHz)	Ζώνη Συχνοτήτων (MHz)	Παράμετροι διάδοσης		Παράμετροι Δεδομένων		
		Ρυθμός Chip (kchip/s)	Διαμόρφωση	Ρυθμός Bit (kb/s)	Ρυθμός Συμβόλων (ksymbol/s)	Σύμβολα
868/915	868-868,6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
2450	2400-2483,5	2000	O-QPSK	250	62.5	16-ary Orthogonal

Οι τρεις ζώνες συχνοτήτων μας παρέχουν ένα σύνολο από 27 κανάλια, αριθμημένα από 0 έως 26. 16 κανάλια είναι διαθέσιμα στη ζώνη των 2450 MHz, 10 στη 915 MHz και 1 στη 868 MHz. Η κεντρική συχνότητα αυτών των καναλιών καθορίζεται ως εξής:

$F_c = 868.3$ MHz, με $k = 0$
$F_c = 906 + 2(k-1)$ MHz, με $k = 1,2,...,10$
$F_c = 2405 + 5(k-11)$ MHz, με $k = 11,12,...,26$

Όπου k είναι ο αριθμός καναλιού.

Όλες οι μετρήσεις της ισχύος του RF (και μετάδοση και λήψη), πραγματοποιούνται στον συνδετήρα που ενώνει τον πομποδέκτη με την κεραία. Για συσκευές που δεν περιλαμβάνουν συνδετήρα κεραίας, οι μετρήσεις θα ερμηνεύονται ως μετρήσεις του EIRP (effective isotropic radiated power). Η μέγιστη ισχύς μετάδοσης θα πρέπει να συμμορφώνεται με τους τοπικούς περιορισμούς.

Προδιαγραφές υπηρεσιών φυσικού επιπέδου

Το PHY μας παρέχει την κατάλληλη διεπαφή μεταξύ του υποστρώματος MAC και του φυσικού ραδιο-καναλιού. Το PHY περιλαμβάνει μια οντότητα διαχείρισης η οποία ονομάζεται, όπως αναφέραμε και σε προηγούμενο σημείο, PLME. Αυτή η οντότητα παρέχει συναρτήσεις διαχείρισης. Το PLME είναι υπεύθυνο για τη

λειτουργία μιας βάσης δεδομένων που εμπεριέχει όλα τα αντικείμενα που αναφέρονται στο PHY. Η βάση δεδομένων ονομάζεται PIB (PAN information base). Το PHY διαθέτει δύο υπηρεσίες, οι οποίες είναι προσβάσιμες μέσω δύο SAPs: την υπηρεσία δεδομένων PHY (PHY data service) που είναι προσβάσιμη μέσω του PD-SAP (PHY data SAP) και την υπηρεσία διαχείρισης PHY η οποία διασυνδέεται με το PLME μέσω του PLME-SAP.

Υπηρεσία δεδομένων PHY

Το PD-SAP παρέχει μεταφορά των MPDUs μεταξύ δύο οντοτήτων MAC. Στο PD-SAP καθορίζονται οι εξής μέθοδοι υπηρεσιών (πίνακας): PD-DATA.request, PD-DATA.confirm, PD-DATA.indication. Η πρώτη χρησιμοποιείται για να αίτηση μεταφοράς ενός MPDU (ή PSDU) από το MAC προς την τοπική οντότητα του PHY. Η δεύτερη μέθοδος επιβεβαιώνει το τέλος της μετάδοσης του MPDU. Η τρίτη δείχνει ότι έγινε η μεταφορά ενός MPDU από το PHY προς την τοπική οντότητα του PHY.

Πίνακας 2.3 Οντότητες υπηρεσίας δεδομένων PHY

PD-SAP	Request	Confirm	Indication
PD-DATA	✓	✓	✓

Υπηρεσία διαχείρισης PHY

Το PLME-SAP επιτρέπει τη μεταφορά διαχειριστικών εντολών μεταξύ MLME και PLME. Ο επόμενος πίνακας περιλαμβάνει όλες τις μεθόδους της υπηρεσίας. Διευκρινίζουμε ότι στην πρώτη στήλη αναφέρονται όλες οι οντότητες του PLME και στις διπλανές στήλες αν περιέχουν μεθόδους αίτησης (Request) και επιβεβαίωσης (Confirm).

Έτσι έχουμε:

Η CCA (clear channel assessment) αναφέρεται στην δυνατότητα να πραγματοποιείται ‘σαφής αξιολόγηση του καναλιού’.

Η ED αναφέρεται στις μετρήσεις ενεργειακής ανίχνευσης.

Η GET αναφέρεται στην αναζήτηση πληροφορίας για κάποια μεταβλητή της PHY PIB.

Η SET-TRX-STATE αναφέρεται στη δυνατότητα αλλαγής της κατάστασης του πομποδέκτη. Ο πομποδέκτης έχει τρεις δυνατές θέσεις: πομποδέκτης απενεργοποιημένος, πομπός ενεργοποιημένος, δέκτης ενεργοποιημένος.

Η SET δίνει μια συγκεκριμένη τιμή σε κάποια μεταβλητή της PHY PIB

Πίνακας 2.4 Οντότητες υπηρεσίας διαχείρισης PHY

PLME	Request	Confirm
PLME-CCA	✓	✓
PLME-ED	✓	✓
PLME-GET	✓	✓
PLME-SET-TRX-STATE	✓	✓
PLME-SET	✓	✓

Δομή PPDU

Η δομή του πακέτου PPDU όπως είναι αποτυπωμένη στο πρωτόκολλο, ορίζει ότι το πεδίο που βρίσκεται αριστερότερα από τα υπόλοιπα είναι εκείνο που μεταδίδεται ή λαμβάνεται πρώτο. Όλα τα πεδία οκτάδων (από bits) θα μεταδίδονται ή θα λαμβάνονται με βάσει τη λιγότερη σημαντική οκτάδα και από κάθε οκτάδα θα μεταδίδεται ή θα λαμβάνεται το μικρότερο σημαντικό bit (LSB – least significant bit).

Το πακέτο PPDU αποτελείται από τα εξής βασικά συστατικά:

Ένα SHR, το οποίο επιτρέπει σε μια συσκευή που λαμβάνει δεδομένα να συγχρονίζεται με το ρεύμα από bits που δέχεται.

Ένα PHR, το οποίο περιέχει πληροφορίες για το μήκος του πλαισίου.

Ένα μεταβλητού μήκους ωφέλιμο φορτίο πληροφορίας, το οποίο μεταφέρει τα πλαίσια του MAC υποστρώματος.

Η δομή φαίνεται στο επόμενο σχήμα:

Πίνακας 2.5 Δομή PPDU

Οκτάδες: 4	**1**	**1**		**Μεταβλητό**
Προοίμιο	SFD	Μήκος πλαισίου (7 bits)	Δεσμευμένο (1 bit)	PSDU
SHR		PHR		PHY payload

Μεταβλητές της PHY PIB

Η PHY PIB περιλαμβάνει τις απαιτούμενες μεταβλητές για τη διαχείριση του PHY μιας συσκευής. Αυτές οι μεταβλητές μπορούν να διαβαστούν ή να εγγραφούν χρησιμοποιώντας τις μεθόδους PLME-GET.request και PLME-SET.request αντίστοιχα. Οι μεταβλητές αυτές παρουσιάζονται στον επόμενο πίνακα.

Πίνακας 2.6 Μεταβλητές της PHY PIB

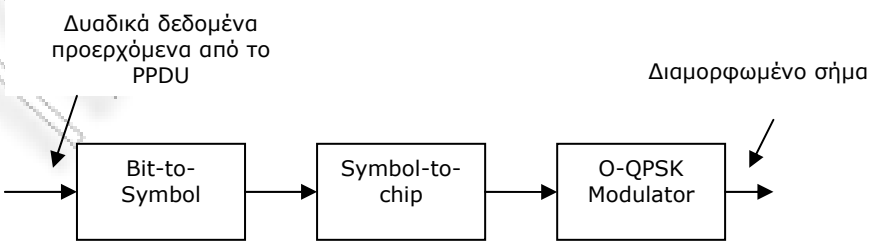
Attribute	**Identifier**	**Type**	**Range**	**Description**
phyCurrentChannel	0 x 00	Integer	0–26	The RF channel to use for all following transmissions and receptions (
phyChannelsSupported	0 x 01	Bitmap		The 5 most significant bits (MSBs) (b27,... , b31) of *phyChannelsSupported* shall be reserved and set to 0, and the 27 LSBs (b0, b1, ... b26) shall indicate the status (1=available, 0=unavailable) for each of the 27 valid channels
phyTransmitPower	0 x 02	Bitmap	0 x 00–0xbf	The 2 MSBs represent the tolerance on the transmit power:

				00 = ± 1 dB 01 = ± 3 dB 10 = ± 6 dB The 6 LSBs represent a signed integer in twos-complement format, corresponding to the nominal transmit power of the device in decibels relative to 1 mW. The lowest value of *phyTransmitPower* shall be interpreted as less than or equal to –32 dBm.
phyCCAMode	0 x 03	Integer	1–3	The CCA mode

Οι προδιαγραφές για τα 2450 MHz

Ο ρυθμός δεδομένων είναι 250 kb/s. Το πρωτόκολλο στη συγκεκριμένη συχνότητα έχει υιοθετήσει μια δεκαεξαδική τεχνική διαμόρφωσης. Κατά τη διάρκεια μιας περιόδου ενός συμβόλου δεδομένων χρησιμοποιούνται 4 bits, προκειμένου να επιλέξουν μία από τις 16 ψευδοτυχαίες ορθογώνιες ακολουθίες θορύβου (PN – pseudo-random sequence), η οποία και θα μεταδοθεί. Οι ακολουθίες PN είναι συνδεδεμένες και συνολικά αποτελούν μια ακολουθία chip, η οποία είναι διαμορφωμένη χρησιμοποιώντας την O-QPSK(offset quadrature phase-shift keying).

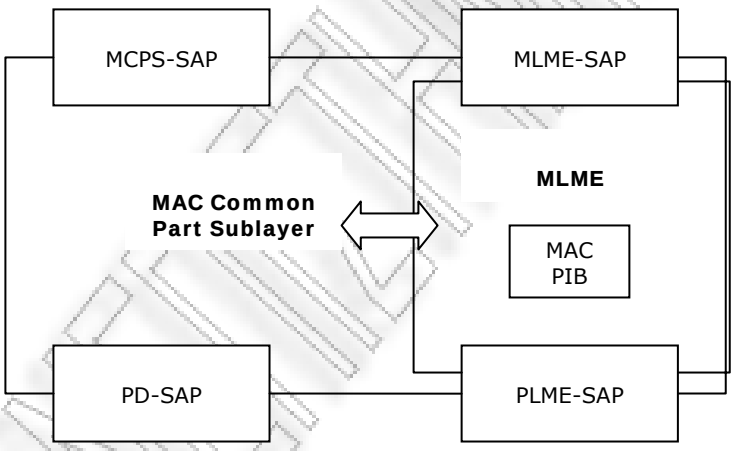
Το επόμενο σχήμα μας παρουσιάζει τις διαδικασίες διαμόρφωσης και διάδοσης.



Σχήμα 2.14 Διαμόρφωση και διάδοση.

2.13 Προδιαγραφές υπηρεσιών του MAC υποστρώματος

Το υπόστρωμα MAC παρέχει μία διεπαφή μεταξύ της υπηρεσίας SSCS (Service Specific Convergence Sublayer) και του PHY. Όπως αναφέραμε και στη συνοπτική περιγραφή του πρωτοκόλλου το υποεπίπεδο MAC παρέχει δύο υπηρεσίες: την υπηρεσία δεδομένων MAC (MCPS-SAP) και την υπηρεσία διαχείρισης MAC η οποία διασυνδέεται με την οντότητα MLME (MLME-SAP). Η πρώτη επιτρέπει τη μετάδοση και την παραλαβή MPDUs προς την υπηρεσία δεδομένων PHY. Η δεύτερη παρέχει τις κατάλληλες διεπαφές μέσω των οποίων μπορούν να επιτευχθούν οι συναρτήσεις διαχείρισης των επιπέδων. Η MLME είναι υπεύθυνη και για τη συντήρηση μιας βάσης δεδομένων που θα εμπεριέχει τα αντικείμενα που αφορούν το MAC υποεπίπεδο. Το επόμενο σχήμα συνοψίζει τα στοιχεία και τις διεπαφές του υποεπιπέδου.



Σχήμα 2.15 Διεπαφές υποεπιπέδου MAC.

Αυτές οι δύο υπηρεσίες παρέχουν την κατάλληλη διεπαφή μεταξύ του SSCS και του PHY, μέσω των διεπαφών PD-SAP και PLME-SAP. Επιπλέον, μεταξύ των MLME και MCPS υπάρχει μια διεπαφή που επιτρέπει να χρησιμοποιεί τις MAC υπηρεσίες δεδομένων το MLME.

Υπηρεσία δεδομένων του MAC

Το MCPS-SAP υποστηρίζει τη μεταφορά των SPDUs μεταξύ των οντοτήτων της SSCS. Ο επόμενος πίνακας παρουσιάζει της συναρτήσεις που υποστηρίζονται από το MCPS-SAP.

Πίνακας 2.7 Οντότητες υπηρεσίας δεδομένων του MAC

MCPS-SAP	Request	Confirm	Indication
MCPS-DATA	✓	✓	✓
MCPS-PURGE	✓	✓	

Οι μέθοδοι data αναφέρονται στη μεταφορά δεδομένων ενός MSDU. Οι μέθοδοι purge αναφέρονται στην εκκαθάριση των MSDUs από την ουρά μεταφοράς. Οι παράμετροι που χρησιμοποιούνται στις παραπάνω μεθόδους, φαίνονται στον επόμενο πίνακα.

Πίνακας 2.8 Παράμετροι υπηρεσίας δεδομένων του MAC

Name	Type	Valid range	Description
SrcAddrMode	Integer	0 x 00–0 x 03	The source addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0 x 00 = no address (addressing fields omitted). 0 x 01 = reserved. 0 x 02 = 16 bit short address. 0 x 03 = 64 bit extended address.
SrcPANId	Integer	0 x 000–0 x ffff	The 16 bit PAN identifier of the entity from which the MSDU is being transferred.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the MSDU is being transferred.

DstAddrMode	Integer	0 x 00–0 x 03	The destination addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0 x 00 = no address (addressing fields omitted). 0 x 01 = reserved. 0 x 02 = 16 bit short address. 0 x 03 = 64 bit extended address.
DstPANId	Integer	0 x 0000–0 x ffff	The 16 bit PAN identifier of the entity to which the MSDU is being transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.
msduLength	Integer	≤ aMaxMACFrameSize	The number of octets contained in the MSDU to be transmitted by the MAC sublayer entity.
msdu	Set of octets	—	The set of octets forming the MSDU to be transmitted by the MAC sublayer entity.
msduHandle	Integer	0 x 00–0 x ff	The handle associated with the MSDU to be transmitted by the MAC sublayer entity.
TxOptions	Bitmap	0000 xxxx (where x can be 0 or 1)	The transmission options for this MSDU. These are a bitwise OR of one or more of the following: 0 x 01 = acknowledged transmission. 0 x 02 = GTS transmission. 0 x 04 = indirect transmission. 0 x 08 = security enabled transmission.
status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, INVALID_GTS, NO_ACK, UNAVAILABLE_KEY, FRAME_TOO_LONG, FAILED_SECURITY_CHECK, or INVALID_PARAMETER	The status of the last MSDU transmission.

mpduLinkQuality	Integer	0 x 00–0 x ff	LQ value measured during reception of the MPDU. Lower values represent lower LQ
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Integer	0 x 00–0 x 08	The *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame. This value is set to 0 x 08 if the sender of the data frame was not found in the ACL.

Υπηρεσία διαχείρισης MAC

Η υπηρεσία MLME-SAP επιτρέπει τη μεταφορά των διαχειριστικών εντολών μεταξύ του αμέσως επόμενου επιπέδου και της MLME. Ο επόμενος πίνακας συνοψίζει τις υποστηριζόμενες μεθόδους της υπηρεσίας.

Πίνακας 2.9 Οντότητες υπηρεσίας διαχείρισης MAC

Name	Request	Indication	Confirm	Indication
MLME-ASSOCIATE	✓	✓	✓	✓
MLME-DISASSOCIATE	✓	✓		✓
MLME-BEACON-NOTIFY		✓		
MLME-GET	✓	✓		✓

MLME-GTS				
MLME-ORPHAN		✓	✓	
MLME-RESET	✓			✓
MLME-RX-ENABLE	✓			✓
MLME-SCAN	✓			✓
MLME-COMM-STATUS		✓		
MLME-SET	✓			✓
MLME-START	✓			✓
MLME-SYNC	✓			
MLME-SYNC-LOSS		✓		
MLME-POLL	✓			✓

Οι ASSOCIATE μέθοδοι διευκρινίζουν με ποιο τρόπο μπορεί να συνδεθεί με κάποιο PAN.

Οι DISASSOCIATION μας δείχνουν πως μπορεί μια συσκευή να αποσυνδεθεί από το PAN.

Οι μέθοδοι BEACON-NOTIFY καθορίζουν πως μπορεί μια συσκευή να ειδοποιηθεί ότι λαμβάνεται κάποιο beacon.

Οι GET αφορούν την ανάγνωση τιμών από την PIB.

Οι GTS διευκρινίζουν τις αιτήσεις και τη διατήρηση των GTSs. Οι ORPHAN καθορίζουν πως ένας διαχειριστής coordinator) μπορεί να εκδώσει μια ανακοίνωση για μία ‘ανεξάρτητη’ συσκευή.

Οι RESET επαναφέρουν τις αρχικές τιμές του υποεπιπέδου MAC.

Οι μέθοδοι RX-ENABLE παράγονται από το υψηλότερο επίπεδο και κοινοποιούνται στην MLME προκειμένου να ενεργοποιήσει το δέκτη για μια συγκεκριμένη χρονική διάρκεια, ξεκινώντας στην αρχή του παρόντος ή του αμέσως επόμενου superframe (για τα PANs που έχουν ενεργοποιημένα τα beacons), ή αμέσως για PANs που δεν στέλνουν beacons.

Οι SCAN μέθοδοι χρησιμοποιούνται στην αρχικοποίηση της σάρωσης ενός καναλιού. Μία συσκευή μπορεί να χρησιμοποιήσει τη σάρωση καναλιού προκειμένου να μετρήσει την ενέργειά του, να ψάξει για τον κατάλληλο διαχειριστή ή να ψάξει για όλους τους διαχειριστές που μεταδίδουν beacons στο πεδίο της συσκευής.

Οι COMM-STATUS αναφέρονται στον τρόπο με τον οποίο η MLME επικοινωνεί με το αμέσως υψηλότερο επίπεδο για θέματα ποιότητας της μετάδοσης, σε περίπτωση που γίνεται μετάδοση χωρίς να έχει δοθεί κατάλληλη εντολή ή στέλνονται λάθη ασφαλείας μέσω των μηνυμάτων.

Οι SET αφορούν την εγγραφή τιμών στην PIB.

Οι μέθοδοι START καθορίζουν με ποιον τρόπο μία FFD μπορεί να ζητήσει την έναρξη των ρυθμίσεων ενός superframe προκειμένου να αρχικοποιηθεί ένα PAN, να ξεκινήσει η μετάδοση beacons, να ανακαλυφθεί μία συσκευή και τέλος να σταματήσει η μετάδοση beacons.

Οι SYNC καθορίζουν πως μπορεί να γίνει συγχρονισμός με ένα διαχειριστή και το πώς ενημερώνεται το υψηλότερο επίπεδο σε περίπτωση που χαθεί ο συγχρονισμός.

Οι POLL μέθοδοι καθορίζουν με ποιον τρόπο μπορεί ένας διαχειριστής να ζητάει δεδομένα.

Στις παραπάνω μεθόδους χρησιμοποιούνται οι εξής παράμετροι (πιν.)

Πίνακας 2.10 Παράμετροι υπηρεσίας διαχείρισης MAC

Name	**Type**	**Valid range**	**Description**
LogicalChannel	Integer	Selected from the available channels supported by the PHY	The logical channel on which to attempt association.
CoordAddrMode	Integer	0 x 02–0 x 03	The coordinator addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 2=16 bit short address. 3=64 bit extended address.
CoordPANId	Integer	0 x 0000–0 x ffff	The identifier of the PAN with which to associate.

CoordAddress	Device address	As specified by the CoordAddrMode parameter.	The address of the coordinator with which to associate.
Capability Information	Bitmap		Specifies the operational capabilities of the associating device.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer orFALSE otherwise.
DeviceAddress	Device address	An extended 64 bit IEEE address.	The address of the device requesting association.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received MAC command frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Integer	0 x 00– 0 x 08	The *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame. This value is set to 0 x 08 if the sender of the data frame was not found in the ACL.
msduHandle	Integer	0 x 00–0 x ff	The handle of the MSDU to be purged from the transaction queue.
status	Enumeration	SUCCESS, INVALID_HANDLE, DENIED, NO_SHORT_ADDRESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK, or INVALID_PARAMETER.	The status of the request to be purged an MSDU from the transaction queue.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.
AssocShortAddress	Integer	0 x 0000–0 x ffff	The short device address allocated by the coordinator on successful association. This parameter is set to 0xffff if the association was unsuccessful.
DisassociateReason	Integer	0 x 00–0 x ff	The reason for the disassociation
PIBAttribute	Integer		The identifier of the PIB attribute to read.

PIBAttributeValue	Various	Attribute specific	The value of the indicated MAC PIB attribute that was read.
GTSCharacteristics	GTS characteristics		The characteristics of the GTS request.
AssociatedMember	Boolean	TRUE or FALSE	TRUE if the orphaned device is associated with this coordinator or FALSE otherwise.
OrphanAddress	Device address	Extended 64 bit IEEE address	The address of the orphaned device.
DeferPermit	Boolean	TRUE or FALSE	TRUE if the receiver enable can be deferred until during the next superframe if the requested time has already passed. FALSE if the receiver enable is only to be attempted in the current superframe. This parameter is ignored for nonbeacon-enabled PANs.
RxOnTime	Integer	0 x 000000–0 x ffffff	The number of symbols from the start of the superframe before the receiver is to be enabled. The precision of this value is a minimum of 20 bits, with the lowest 4 bits being the least significant. This parameter is ignored for nonbeacon-enabled PANs.
RxOnDuration	Integer	0 x 000000–0 x ffffff	The number of symbols for which the receiver is to be enabled.
ScanType	Integer	0 x 00─0 x 03	Indicates if the type of scan performed: 0 x 00 = ED scan (FFD only). 0 x 01 = active scan (FFD only). 0 x 02 = passive scan. 0 x 03 = orphan scan.
UnscannedChannels	Bitmap	32 bit field	Indicates which channels given in the request were not scanned (1 = not scanned, 0 = scanned or not requested). This parameter is only valid for passive or active scans.
ResultListSize	Integer	Implementation specific	The number of elements returned in the appropriate result lists. This value is 0 for the result of an orphan scan.
EnergyDetectList	List of integers	0 x 00–0 x ff for each integer	The list of energy measurements, one for each channel searched during an ED scan. This parameter is null for active, passive, and orphan scans.

PANDescriptorList	List of PAN descriptor values		The list of PAN descriptors, one for each beacon found during an active or passive scan. This parameter is null for ED and orphan scans.
BeaconOrder	Integer	0─15	How often the beacon is to be transmitted. The beacon order, *BO*, and the beacon interval, *BI*, are related as follows: for 0 δ *BO* δ 14, *BI* = *aBaseSuperframeDuration* * 2^{BO} symbols. If *BO* = 15, the coordinator will not transmit a beacon, and the SuperframeOrder parameter value is ignored.
SuperframeOrder	Integer	0─*BO* or 15	The length of the active portion of the superframe, including the beacon frame. The superframe order, *SO*, and the superframe duration, *SD*, are related as follows: for 0 δ *SO* δ BO δ 14, *SD* = *aBaseSuperframeDuration* * 2^{SO} symbols. If *SO* = 15, the superframe will not be active after the beacon.
CoordRealignment	Boolean	TRUE or FALSE	TRUE if a coordinator realignment command is to be transmitted prior to changing the superframe configuration or FALSE otherwise.
BatteryLifeExtension	Boolean	TRUE or FALSE	If this value is TRUE, the receiver of the beaconing device is disabled *macBattLifeExtPeriods* full backoff periods after the interframe spacing (IFS) period of the beacon frame. If this value is FALSE, the receiver of the beaconing device remains enabled for the entire CAP.
TrackBeacon	Boolean	TRUE or FALSE	TRUE if the MLME is to synchronize with the next beacon and attempt to track all future beacons. FALSE if the MLME is to synchronize with only the next beacon.
LossReason	Enumeration	PAN_ID_CONFLICT, REALIGNMENT, or BEACON_LOST	The reason that synchronization was lost.
ShortAddress	Integer	0 x 0000–0 x ffff	The short address allocated to the orphaned device if it is associated with this coordinator. The special short address 0 x fffe indicates that no short address was allocated, and the device will use its 64 bit extended address in all communications. If the device was not associated with this coordinator, this field will contain the value 0 x ffff and be ignored on receipt.

SetDefaultPIB	Boolean	TRUE or FALSE	If TRUE, the MAC sublayer is reset and all MAC PIB attributes are set to their default values. If FALSE, the MAC sublayer is reset but all MAC PIB attributes retain their values prior to the generation of the MLME-RESET.request primitive.

2.14 Περιγραφή των Λειτουργιών του MAC

Όπως αναφέραμε και στο υποκεφάλαιο 2.4, υπάρχουν δύο μηχανισμοί για πρόσβαση στο κανάλι. Η πρόσβαση με τον contention-based μηχανισμό και η πρόσβαση χωρίς contention μηχανισμό. Η πρώτη, επιτρέπει στις συσκευές να γίνουν προσβάσιμες προς το κανάλι χρησιμοποιώντας ένα μηχανισμό CSMA-CA για υπαναχώρηση (back-off). Στη δεύτερη περίπτωση η πρόσβαση καναλιού ελέγχεται πλήρως από το διαχειριστή μέσω GTSs (guaranteed time slots).

Δομή Superframe

Ένας διαχειριστής μπορεί να οριοθετεί προαιρετικά τη διάρκεια πρόσβασης στο κανάλι χρησιμοποιώντας δομή superframe. Ένα superframe οριοθετείται με τη μετάδοση ενός πλαισίου beacon και περιλαμβάνει δύο περιοχές, μια ενεργή και μία ανενεργή. Ένας διαχειριστής μπορεί να επικοινωνεί με το PAN του μόνο κατά τη διάρκεια της ενεργής περιοχής, ενώ κατά τη διάρκεια της ανενεργής οδηγείται σε κατάσταση χαμηλής ισχύος (κατάσταση ‘ύπνου’). Η δομή του superframe περιγράφεται από τις τιμές του MAC PIB: *macBeaconOrder, macSuperframeOrder*. Η πρώτη αφορά το διάστημα όπου ο διαχειριστής μεταδίδει τα πλαίσια beacons. Η δεύτερη περιγράφει το μήκος της ενεργούς περιοχής του superframe.

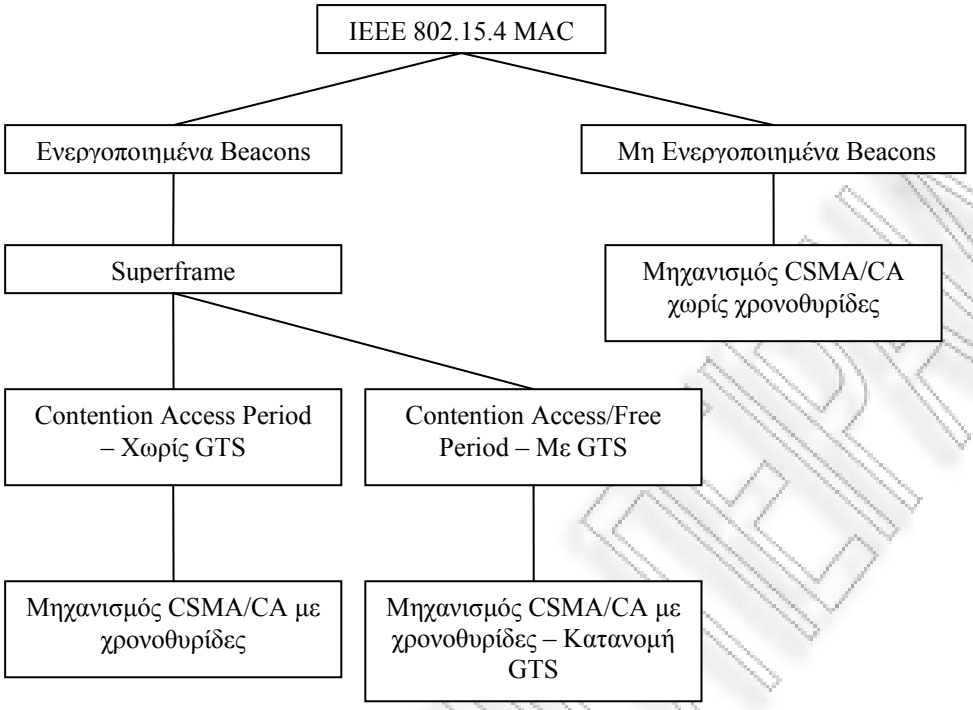
Τα PANs που επιθυμούν να χρησιμοποιήσουν τη δομή Superframe θα θέσουν στην *macBeaconOrder* μια τιμή μεταξύ 0 και 14 και στην *macSuperframeOrde*r μια

τιμή μεταξύ 0 και την τιμή της *macBeaconOrder*. Τα PANs που δεν επιθυμούν τη χρήση της δομής superframe θα θέσουν στις παραπάνω μεταβλητές την τιμή 15. Σε αυτήν την περίπτωση ο διαχειριστής δε θα μεταδίδει beacons και θα χρησιμοποιείται ο μηχανισμός CSMA-CA χωρίς χρονοθυρίδες προκειμένου να υπάρχει πρόσβαση στο κανάλι.

Μηχανισμός CSMA – CA

Ένα LR-WPAN χρησιμοποιεί δύο τύπους πρόσβασης στο κανάλι, ανάλογα με τις ρυθμίσεις του δικτύου. Τα δίκτυα που δεν χρησιμοποιούν beacons παρέχουν έναν μηχανισμό CSMA – CA χωρίς τη χρήση χρονοθυρίδων. Κάθε στιγμή που μια συσκευή επιθυμεί να μεταδώσει πλαίσια δεδομένων ή εντολές MAC, περιμένει για μια τυχαία περίοδο. Αν μετά τη τυχαία αυτή περίοδο βρεθεί ότι το κανάλι είναι ανενεργό, η συσκευή μεταδίδει τα δεδομένα. Αν μετά τη τυχαία περίοδο βρεθεί ότι το κανάλι είναι απασχολημένο, η συσκευή θα αναμείνει ακόμα μία τυχαία περίοδο και στη συνέχεια θα ξαναπροσπαθήσει να ελέγξει το κανάλι. Τα πλαίσια επιβεβαίωσης θα στέλνονται χωρίς τη χρήση του μηχανισμού CSMA – CA.

Τα δίκτυα που χρησιμοποιούν beacons παρέχουν έναν μηχανισμό CSMA – CA με τη χρήση χρονοθυρίδων, όπου οι θυρίδες υπαναχώρησης (back off) βρίσκονται στην αρχή της μετάδοσης των beacons. Κάθε φορά που μια συσκευή επιθυμεί να μεταδώσει πλαίσια δεδομένων κατά τη διάρκεια της περιόδου CAP, αφού προσδιορίσει τα όρια της επόμενης χρονοθυρίδας υπαναχώρησης, θα περιμένει για ένα τυχαίο αριθμό χρονοθυρίδων υπαναχώρησης. Αν μετά τη τυχαία περίοδο βρεθεί ότι το κανάλι είναι απασχολημένο, η συσκευή θα αναμείνει για ένα τυχαίο αριθμό από χρονοθυρίδες υπαναχώρησης πριν να δοκιμάσει να ελέγξει πάλι το κανάλι. Αν το κανάλι είναι ανενεργό, η συσκευή θα ξεκινήσει να μεταδίδει. Τα πλαίσια επιβεβαίωσης και beacon θα στέλνονται χωρίς τη χρήση του μηχανισμού CSMA – CA. Στο επόμενο σχήμα παρουσιάζονται συνοπτικά οι δύο μηχανισμοί λειτουργίας [12].



Σχήμα 2.16 Μηχανισμός λειτουργίας

Βήμα 1: Σε κάθε προσπάθεια μετάδοσης κάθε συσκευή θα καθορίζει τις μεταβλητές: NB, CW και BE. Η NB μας λέει πόσες φορές ο αλγόριθμος χρειάστηκε να υπαναχωρήσει ενώ προσπαθούσε να μεταδώσει. Αυτή η μεταβλητή αρχικοποιείται στην τιμή 0 πριν από κάθε προσπάθεια για μετάδοση. Η CW περιλαμβάνει το μήκος του παραθύρου 'contention'. Η αρχική του τιμή είναι 2 πριν από κάθε προσπάθεια για μετάδοση και αρχικοποιείται και πάλι στο 2 κάθε φορά που το κανάλι είναι κατειλημμένο. Το BE είναι ο εκθέτης για τον υπολογισμό των back off και σχετίζεται με το πόσες περιόδους back off μια συσκευή θα περιμένει μέχρι να ξαναπροσεγγίσει.

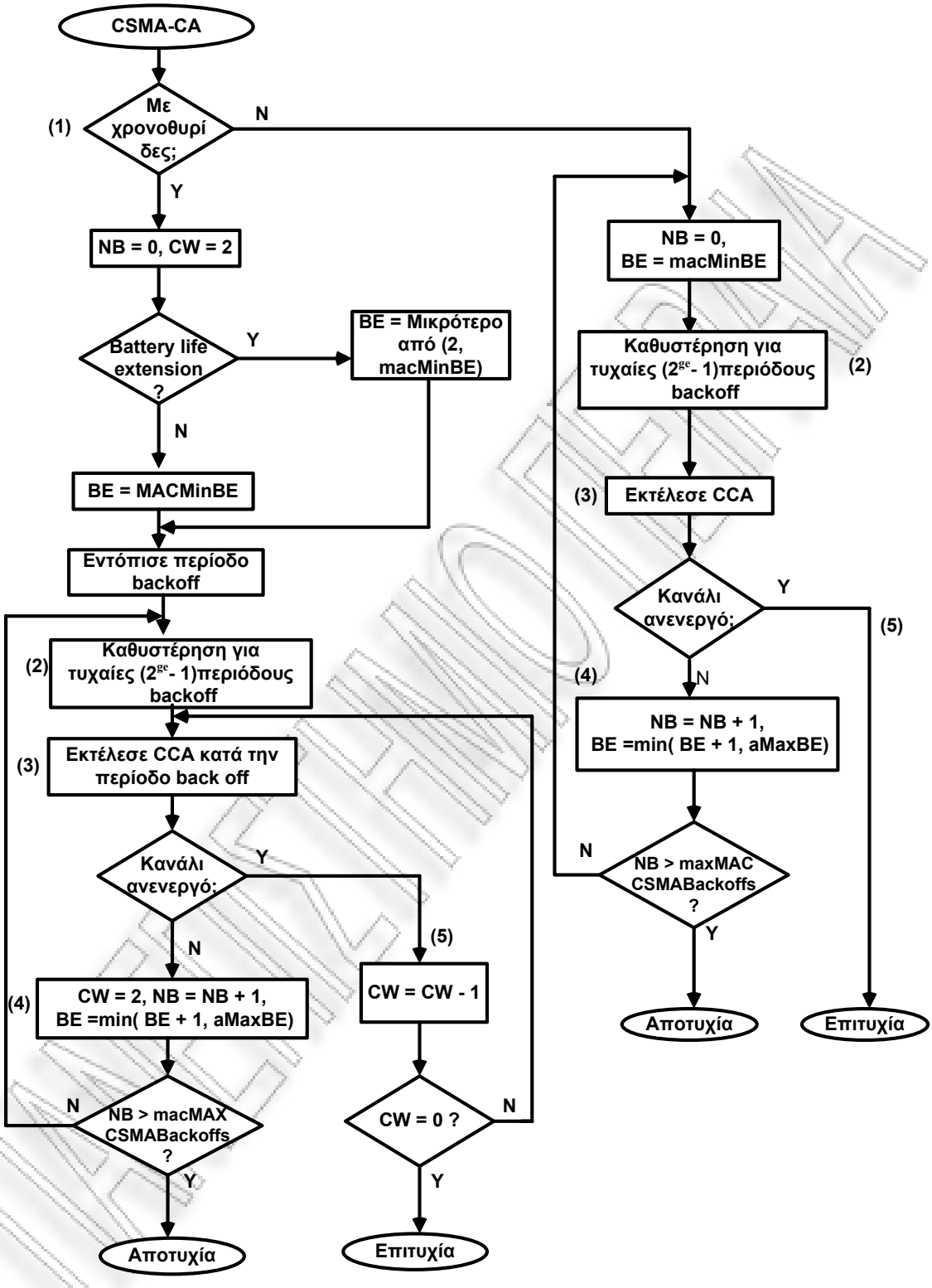
Βήμα 2: Στο CSMA-CA με χρονοθυρίδες, το NB, το CW και το BE αρχικοποιούνται στο όριο της επόμενης περιόδου backoff . Τα NB και BE αρχικοποιούνται στο βήμα 1. Το υποεπίπεδο MAC θα καθυστερήσει για έναν τυχαίο αριθμό πλήρων backoff περιόδων στη περιοχή 0 $2^{BE} - 1$.

Βήμα 3: Κατόπιν θα απαιτήσει από το PHY να εκτελέσει μία CCA Έπειτα θα προχωρήσει, μόνο εάν τα υπόλοιπα βήματα του αλγορίθμου CSMA-CA, δηλαδή η μετάδοση πλαισίων επιβεβαιώσεων, μπορούν να ολοκληρωθούν πριν από το τέλος

του CAP. Εάν το MAC δεν μπορεί να προχωρήσει, θα περιμένει μέχρι την έναρξη του CAP του επόμενου superframe και θα επαναλάβει την αξιολόγηση.

Βήμα 4: Αν το κανάλι αξιολογηθεί ως απασχολημένο, το MAC θα αυξήσει το NB και το BE κατά ένα, εξασφαλίζοντας ότι το BE θα είναι λιγότερο από το *aMaxBE*. Στο μηχανισμό CSMA-CA με χρονοθυρίδες, το CW θα επανέλθει στην τιμή 2. Εάν η τιμή του NB είναι μικρότερη ή ίση του *macMaxCSMABackoffs*, το CSMA-CA θα επιστρέψει στο βήμα 2, αλλιώς το CSMA-CA θα τερματίσει σε μια κατάσταση αποτυχημένης πρόσβασης στο κανάλι.

Βήμα 5: Αν το κανάλι αξιολογηθεί ως ανενεργό, το υποεπίπεδο MAC, στο μηχανισμό CSMA-CA με χρονοθυρίδες, εξασφαλίζει ότι το παράθυρο 'contention' λήγει πριν αρχίσει τη μετάδοση. Για αυτό το λόγο, το MAC αρχικά μειώνει το CW κατά ένα. Εάν το CW δεν είναι ίσο με 0, πηγαίνει στο βήμα 3 αλλιώς αρχίζει τη μετάδοση στο όριο της επόμενης περιόδου backoff. Το MAC ξεκινά αμέσως τη μετάδοση εάν το κανάλι αξιολογηθεί ανενεργό. Ο αλγόριθμος CSMA-CA παρουσιάζεται στο επόμενο σχήμα.



<u>Σχήμα 2.17</u> Μηχανισμός CSMA

Ξεκινώντας ένα καινούριο PAN

Όλες οι συσκευές είναι ικανές να πραγματοποιήσουν ορφανή και παθητική σάρωση σε μια συγκεκριμένη λίστα καναλιών. Μια συσκευή θα ξεκινήσει μια σάρωση καναλιού μέσω της μεθόδου MLME-SCAN.request. Κατά τη διάρκεια της σάρωσης, η συσκευή θα παγώσει τις μεταδόσεις beacons και λίγο πριν το τέλος της σάρωσης θα επαναφέρει τις μεταδόσεις. Το αποτέλεσμα της σάρωσης θα επιστρέψει μέσω της μεθόδου MLME-SCAN.confirm. Μία σάρωση ED (energy detection) επιτρέπει σε μία FFD να μετρήσει την μέγιστη τιμή της ενέργειας σε κάθε κανάλι. Η συγκεκριμένη μέτρηση μπορεί να χρησιμοποιηθεί από ένα διαχειριστή PAN και να επιλέξει κανάλι ελέγχου ώστε να ξεκινήσει το PAN. Κατά τη διάρκεια της σάρωσης ED το MAC θα απορρίπτει όλα τα πλαίσια που θα λαμβάνει από το PHY.

Μία ενεργή σάρωση επιτρέπει σε μία FFD να προσδιορίσει ποιος διαχειριστής μεταδίδει beacons, μέσα στην εμβέλεια του. Αυτό θα μπορούσε να χρησιμοποιηθεί από έναν διαχειριστή PAN ώστε να επιλέξει αναγνωριστικό ή από μία συσκευή προκειμένου να ζητήσει σύνδεση με το PAN. Πριν ξεκινήσει η ενεργή σάρωση το MAC αποθηκεύει την τιμή της *macPANId* και την θέτει στην τιμή 0xffff κατά τη διάρκεια της σάρωσης. Αυτό επιτρέπει στο δέκτη να δέχεται όλα τα beacons που στέλνονται και όχι μόνο από τον τωρινό του διαχειριστή. Η ενεργή σάρωση ζητείται με τη μέθοδο MLME-SCAN.request. Στη συνέχεια η συσκευή θα ενεργοποιήσει το δέκτη της. Μία συσκευή μπορεί να αποθηκεύσει από ένα μέχρι συγκεκριμένα προσδιοριστικά PANs.

Παρόμοια με την ενεργή σάρωση, μια παθητική σάρωση μπορεί να εντοπίσει όλους τους διαχειριστές που εκπέμπουν beacons, μέσα στην εμβάλεια της. Όμως δεν μεταδίδονται αιτήσεις για εντολές beacons. Η παθητική σάρωση σε συγκεκριμένα κανάλια ζητείται με τη μέθοδο MLME-SCAN.request. Μία συσκευή μπορεί να αποθηκεύσει από ένα μέχρι ένα μέγιστο προσδιοριστικό PAN.

Μία ‘ορφανή’ σάρωση επιτρέπει σε μία συσκευή να αναπροσδιορίσει το διαχειριστή της σε περίπτωση διακοπής του συγχρονισμού τους. Κατά τη διάρκεια της ‘ορφανής’ σάρωσης το MAC θα απορρίπτει όλα τα πλαίσια που θα λαμβάνει από το PHY. Η ‘ορφανή’ σάρωση σε συγκεκριμένα κανάλια ζητείται με τη μέθοδο MLME-SCAN.request. Όταν ο διαχειριστής λάβει μία ένδειξη με ‘ορφανή’ εντολή, θα ψάξει τη λίστα του με τις συσκευές, για να εντοπίσει αυτήν που τη στέλνει. Αν

την εντοπίσει στο αρχείο του, θα στείλει μια εντολή επαναπροσδιορισμού με την 'ορφανή' συσκευή.

Ένα καινούριο PAN θα ξεκινήσει τη λειτουργία του μόνο αφού έχει πραγματοποιηθεί μία ενεργή σάρωση και έχει επιλεγεί κατάλληλο αναγνωριστικό PAN. Η FFD θα ορίσει τη μεταβλητή *macShortAddress* με μια τιμή μικρότερη από 0 x ffff. Μία FFD θα ξεκινήσει τη διαχείρηση του PAN με τη χρήση της μεθόδου MLME-START.request και με τη μεταβλητή PANCoordinator ίση με true και τη μεταβλητή *CoordRealigment* ίση με false. Χρησιμοποιώντας πάλι την προηγούμενη μέθοδο θα ξεκινήσει η μετάδοση beacons. Όταν το MAC δεχθεί την μέθοδο αυτή θα θέσει το προσδιοριστικό PAN ως macPANId. Η ώρα μετάδοσης του πιο πρόσφατα σταλμένου beacon καταχωρείται στην *macBeaconTxTime*. Όλα τα πλαίσια beacons θα μεταδοθούν κατά την έναρξη κάθε superframe και στη διάρκεια ενός διαστήματος ίσου με *aBaseSuperframeDuration*$*2^n$ σύμβολα, όπου N είναι η τιμή της *macBeaconOrder*.

Μία FFD μπορεί να κάνει αισθητή την παρουσία της σε ένα PAN στέλνοντας πλαίσια beacons. Αυτό επιτρέπει στις υπόλοιπες συσκευές να εκτελέσουν 'ανακάλυψη συσκευής'. Η FFD πραγματοποιεί τις παραπάνω ενέργειες μόνο αν έχει συνδεθεί σε κάποιο PAN. Η μετάδοση πλαισίων beacons αρχικοποιείται με τη χρηση της μεθόδου MLME-START.request και θέτοντας την τιμή PANCoordinator ίση με false.

Σύνδεση με το PAN

Μία συσκευή θα προσπαθήσει να συνδεθεί με κάποιο PAN αφού πρώτα έχει επανεκκινήσει τις τιμές των μεταβλητών του υποεπιπέδου MAC με τη χρήση της μεθόδου MLME-RESET.request και αφού έχει ολοκληρώσει μια ενεργή σάρωση καναλιού. Τα αποτελέσματα της σάρωσης του καναλιού θα χρησιμοποιηθούν για την επιλογή του κατάλληλου PAN. Ένας διαχειριστής θα επιτρέψει τη σύνδεση με τη συσκευή μόνο αν η τιμή της μεταβλητής *macAssociationPermit* είναι TRUE.

Ακολούθως, το αμέσως υψηλότερο επίπεδο θα κάνει αίτηση προς το MLME ώστε το τελευταίο να καθορίσει τις παρακάτω τιμές των PIB των MAC και PHY:

- Στη *phyCurrentChannel* θα τεθεί το κατάλληλο λογικό κανάλι που θα συνδεθεί.
- Στη *macPANId* θα τεθεί το κατάλληλο ID του προς σύνδεση PAN
- Στις *macCoordExtendedAddress* or *macCoordShortAddress* θα τεθεί κατάλληλη τιμή βάσει του πλαισίου beacon του διαχειριστή.

Η επιβεβαίωση που στέλνεται ως απάντηση σε μια αίτηση για σύνδεση, δεν συνεπάγεται και αποδοχή από το διαχειριστή. Ο διαχειριστής χρειάζεται χρόνο προκειμένου να διαπιστώσει αν υπάρχουν διαθέσιμοι πόροι στο PAN ώστε να συνδεθεί άλλη μία συσκευή. Ο διαχειριστής θα αποφασίσει μέσα στο χρονικό διάστημα που ορίζει η μεταβλητή *aResponseWaitTime*. Αν ο διαχειριστής διαπιστώσει ότι η συγκεκριμένη συσκευή ήταν και σε παλιότερη στιγμή συνδεδεμένη με το PAN του, θα σβήσει όλες τις πληροφορίες που σχετίζονται με τη συσκευή αυτήν. Αν διαπιστωθεί ότι τηρούνται όλες οι προϋποθέσεις, ο διαχειριστής θα ορίσει μια διεύθυνση για τη συσκευή και θα δημιουργήσει μια εντολή αποδοχής της σύνδεσης.

Όταν η συσκευή λάβει την επιβεβαίωση από την αίτηση για σύνδεση, θα περιμένει για περίοδο *aResponseWaitTime* προκειμένου ο διαχειριστής να αποφασίσει την είσοδο της ή όχι στο PAN. Όταν τελειώσει αυτή η περίοδος και δεν λάβει κάποια απάντηση από το διαχειριστή, θα ορίσει στη μέθοδο MLME-ASSOCIATE.confirm κατάσταση NO_DATA και η σύνδεση θα οδηγηθεί σε αποτυχία. Αν η συσκευή λάβει απάντηση για επιτυχή σύνδεση με το PAN, θα αποθηκεύσει τη διεύθυνση του διαχειριστή που συνδέθηκε.

Αποσύνδεση από το PAN

Η αποσύνδεση από κάποιο PAN αρχικοποιείται από το αμέσως υψηλότερο επίπεδο, στέλνοντας μια μέθοδο MLMEDISASSOCIATE.request προς το MLME. Όταν ο διαχειριστής θελήσει να αποσυνδεθεί από το PAN κάποια από τις συνδεδεμένες συσκευές, θα στείλει μια εντολή αποσύνδεσης χρησιμοποιώντας έμμεση μετάδοση. Αν η συσκευή ανταποκριθεί και λάβει σωστά την εντολή, θα αποδεχθεί τη λήψη του στέλνοντας ένα πλαίσιο επιβεβαίωσης. Ακόμα και αν ο

διαχειριστής δε λάβει ποτέ την επιβεβαίωση, θα θεωρεί τη συσκευή αποσυνδεδεμένη.

Στην περίπτωση όπου μία συσκευή θελήσει να αποδεσμευτεί από το PAN, θα στείλει μια εντολή αποσύνδεσης στο διαχειριστή. Αν αυτή ληφθεί σωστά από το διαχειριστή, εκείνος θα το αποδεχθεί στέλνοντας ένα πλαίσιο επιβεβαίωσης. Ακόμα και αν η συσκευή δε λάβει ποτέ την επιβεβαίωση, η συσκευή θα θεωρείται αποσυνδεδεμένη. Μια συνδεδεμένη συσκευή θα μπορέσει να αποδεσμεύσει τον εαυτό της, βγάζοντας όλες τις αναφορές προς το PAN. Με όμοιο τρόπο και ο διαχειριστής θα αποβάλλει όλες τις αναφορές προς τη συσκευή.

Συγχρονισμός με χρήση beacons

Όλες οι συσκευές που λειτουργούν σε ένα δίκτυο με ενεργοποιημένη τη χρήση beacons, είναι ικανές να συγχρονιστούν με τη χρήση beacons προκειμένου να διαπιστώσουν αν εκκρεμούν δεδομένα για αυτές. Οι συσκευές αυτές επιτρέπεται να συγχρονίζονται μόνο αν τα λαμβανόμενα beacons έχουν αναγνωριστικό PAN ίσο με αυτό της μεταβλητής *macPANId*. Η συσκευή προσπαθεί να αναγνωρίσει το beacon με τη μέθοδο MLME-SYNC.request. Προκειμένου να επιτευχθεί ο συγχρονισμός με beacons, θα ενεργοποιήσει το δέκτη της και θα ψάξει για διάρκεια *aBaseSuperframeDuration * (2n + 1)* συμβόλων, όπου n είναι η τιμή της *macBeaconOrder*.

Αν ληφθεί ένα πλαίσιο beacon, η συσκευή θα πρέπει να επιβεβαιώσει ότι το πλαίσιο αυτό προήλθε από το διαχειριστή με τον οποίο έχει συνδεθεί. Σε περίπτωση που η διεύθυνση που περιλαμβάνεται στο beacon δεν ταυτίζεται με εκείνη του διαχειριστή PAN, το MLME θα απορρίψει το πλαίσιο. Αν το πλαίσιο αυτό είναι έγκυρο και επιπρόσθετα η τιμή της *macAutoRequest* είναι false, το MLME θα υποδείξει τις παραμέτρους του beacon προς το επόμενο υψηλότερο επίπεδο με τη χρήση της MLME-BEACON-NOTIFY.indication.

Αν είναι ενεργοποιημένη η ανίχνευση beacons, το MLME θα ενεργοποιήσει τον πομπό σε ένα κοντινό χρόνο με την επόμενη μετάδοση των beacons, π.χ. λίγο πριν τη έναρξη του επόμενου superframe. Αν ο αριθμός των διαδοχικών beacons που χάνονται από το MLME φτάσει την τιμή *aMaxLostBeacons,* το MLME θα

απαντήσει με τη μέθοδο MLME-SYNC-LOSS.indication, θέτοντας ως λόγο απώλειας: BEACON_LOST.

Συγχρονισμός χωρίς τη χρήση beacons

Όλες οι συσκευές που λειτουργούν σε ένα δίκτυο που δεν έχει ενεργοποιημένη τη χρήση beacons, θα ζητήσουν για δεδομένα που εκκρεμούν από το διαχειριστή. Η αίτηση αυτή πραγματοποιείται με τη μέθοδο MLME-POLL.request.

Χειρισμός των συναλλαγών

Προκειμένου να έχουμε πολύ φτηνές συσκευές οι οποίες θα τροφοδοτούνται από μπαταρίες, οι συναλλαγές θα υποκινούνται από τις συσκευές και όχι από το διαχειριστή. Με άλλα λόγια ο διαχειριστής χρειάζεται να υποδεικνύει στα beacons που εκπέμπει αν εκκρεμούν μηνύματα για κάποιες συσκευές ή οι συσκευές από μόνες τους χρειάζεται να ρωτήσουν το διαχειριστή αν έχει μηνύματα που εκκρεμούν. Αυτές οι μεταφορές λέγονται έμμεσες.

Ο διαχειριστής θα ξεκινήσει να χειρίζεται μία συναλλαγή κατά την λήψη μιας αίτησης έμμεσης μετάδοσης με τη χρήση της μεθόδου MCPS-DATA.request. Μετά την ολοκλήρωση της συναλλαγής, το υποεπίπεδο MAC θα ενημερώσει για την κατάσταση το αμέσως επόμενο επίπεδο. Η πληροφορία η οποία περιέχεται σε μια αίτηση έμμεσης μετάδοσης αποτελεί μία συναλλαγή και ο διαχειριστής θα πρέπει να είναι ικανός να αποθηκεύσει τουλάχιστον μία συναλλαγή. Κατά τη λήψη μίας αίτησης έμμεσης μετάδοσης, εάν δεν υπάρχει άλλος χώρος για την αποθήκευση κάποιας άλλης συναλλαγής, το υποεπίπεδο MAC θα υποδείξει στο ανώτερο επίπεδο τη μέθοδο MLME-COMM-STATUS.indication με ένδειξη TRANSACTION_ OVERFLOW.

Μετάδοση, λήψη και επιβεβαίωση

Κάθε φορά που παράγονται δεδομένα ή πλαίσια εντολών MAC, το υποεπίπεδο MAC αντιγράφει την τιμή της *macDSN* στο πεδίο ακολουθίας αριθμών του MHR που βρίσκεται στο πλαίσιο που στέλνεται και στη συνέχεια αυξάνει την τιμή της κατά ένα. Παρόμοια, κάθε φορά που παράγεται ένα πλαίσιο beacon, το υποεπίπεδο MAC αντιγράφει την τιμή της *macBSN* στο πεδίο ακολουθίας αριθμών του MHR που βρίσκεται στο πλαίσιο που στέλνεται και στη συνέχεια αυξάνει την τιμή της κατά ένα. Το πεδίο source address (διεύθυνση προέλευσης) θα περιέχει τη διεύθυνση της συσκευής που στέλνει το πλαίσιο. Το πεδίο destination address (διεύθυνση προορισμού) θα περιέχει τη διεύθυνση της συσκευής όπου θα καταλήξει το πλαίσιο και μπορεί να είναι είτε 16-bit είτε 64-bit.

Το υποεπίπεδο MAC θα συγκρίνει τα προσδιοριστικά των PAN προέλευσης και προορισμού. Αν τα προσδιοριστικά είναι όμοια, το υπο-πεδίο intra-PAN του πεδίου ελέγχου πλαισίου θα τεθεί ίσο με 1 και θα παραληφθεί το αναγνωριστικό του PAN προέλευσης. Αν είναι διαφορετικά, το υπο-πεδίο intra-PAN του πεδίου ελέγχου πλαισίου θα τεθεί ίσο με 0 και θα συμπεριληφθούν στο μεταδιδόμενο frame και τα δύο αναγνωριστικά.

Κάθε συσκευή θα επιλέγει πότε το υποεπίπεδο MAC θα ενεργοποιεί το δέκτη κατά τη διάρκεια των ανενεργών περιόδων. Κατά τη διάρκεια αυτών των ανενεργών περιόδων το MAC δε θα σταματήσει να τροφοδοτεί τον πομποδέκτη με αιτήσεις από το υψηλότερο επίπεδο. Κάθε διεργασία του πομποδέκτη θα θεωρείται σαν μια αίτηση μετάδοσης με λήψη επιβεβαίωσης. Με την ολοκλήρωση της διεργασίας του πομποδέκτη, το υποεπίπεδο MAC θα αιτηθεί ώστε το PHY να ενεργοποιήσει ή να απενεργοποιήσει το δέκτη. Αυτό εξαρτάται από το αν η μεταβλητή *macRxOnWhenIdle* είναι true ή false.

Σε ένα δίκτυο που είναι ενεργοποιημένη η αποστολή beacons, μία συσκευή μπορεί να διαπιστώσει αν εκκρεμούν πλαίσια για αυτήν ελέγχοντας τα περιεχόμενα του πλαισίου beacon που έλαβε. Αν η διεύθυνση της συσκευής περιλαμβάνεται στο πεδίο διευθύνσεων του πλαισίου beacon, το MLME θα στείλει μια εντολή αίτησης δεδομένων στο διαχειριστή κατά τη διάρκεια του CAP. Υπάρχουν δύο ακόμα περιπτώσεις που το MLME στέλνει εντολή αίτησης δεδομένων στο διαχειριστή. Η πρώτη είναι όταν το MLME λαμβάνει μια μέθοδο MLME-POLL.request. Στη δεύτερη περίπτωση, μία συσκευή μπορεί να στείλει μια εντολή αίτησης δεδομένων

για διάρκεια *aResponseWaitTime* συμβόλων μετά την επιβεβαίωση για μια εντολή αίτησης.

Το πλαίσιο δεδομένων θα μεταδοθεί χρησιμοποιώντας έναν από τους επόμενους μηχανισμούς:

- Χωρίς τη χρήση CSMA-CA, αν το υποεπίπεδο MAC ξεκινήσει τη μετάδοση του πλαισίου δεδομένων μεταξύ των διαστημάτων *aTurnaroundTime* και (*aTurnaroundTime* + *aUnitBackoffPeriod*) και ακόμα, έχει περισσέψει αρκετός χρόνος στο CAP για το μήνυμα, το κατάλληλο IFS και την επιβεβαίωση. Αν δε ληφθεί πλαίσιο επιβεβαίωσης, όλα οι επόμενες μεταδόσεις θα μεταδοθούν με τη χρήση του μηχανισμού CSMA-CA.
- Με τη χρήση CSMA-CA.

Εάν η συσκευή που πραγματοποίησε την αίτηση δεν έλαβε το πλαίσιο δεδομένων από το διαχειριστή μέσα σε χρόνο *aMaxFrameResponseTime* σύμβολα της περιόδου CAP ή απλά σύμβολα (σε δίκτυο χωρίς beacons) ή αν την έλαβε αλλά έχει μηδενικό ωφέλιμο φορτίο, θα καταλάβει ότι δεν εκκρεμούν δεδομένα στο διαχειριστή. Αν τελικά λάβει το πλαίσιο δεδομένων, θα αποστείλει ένα παλίσιο επιβεβαίωσης αποδοχής.

Κατανομή των GTS και διαχείρηση

Τα GTS επιτρέπει σε μία συσκευή να λειτουργεί στο κανάλι μέσα σε ένα μερος του superframe που υπάρχει αποκλειστικά για αυτήν την συσκευή. Το GTS μπορεί να διατεθεί μόνο από το διαχειριστή του PAN, και μπορεί να χρησιμοποιηθεί μόνο για επικοινωνίες μεταξύ του διαχειριστή PAN και κάποιας συσκευής. Ένα απλό GTS μπορεί να επεκταθεί και πλέον του superframe. Ένας διαχειριστής PAN θα πρέπει να είναι ικανός να αποθηκεύσει όλη την απαραίτητη πληροφορία προκειμένου να διαχειριστεί εφτά GTSs. Για κάθε GTS ο διαχειριστής PAN θα μπορεί να αποθηκεύσει την αρχική θυρίδα του, το μήκος του, την κατεύθυνση του και την διεύθυνση της συνδεδεμένης συσκευής.

Διαγράμματα ακολουθίας μηνυμάτων που απεικονίζουν την αλληλεπίδραση MAC-PHY

Το MAC υποεπίπεδο παρέχει ένα interface προκειμένου να συνδέσει το SSCS[1] (Service Specific Convergence Sublayer) και το PHY. Για αυτό το σκοπό το MAC περιλαμβάνει μια οντότητα διαχείρισης που ονομάζεται MLME (MAC Sublayer Management Entity) και η οποία περιλαμβάνει τις υπηρεσίες για την διαχείριση των επιπέδων. Το MLME είναι ακόμα υπεύθυνο για την βάση δεδομένων που εμπεριέχει όλα τα αντικείμενα διαχείρισης (PIB – PAN Inforamation Base). Όπως αναφέρθηκε και στη σελίδα 6 το MAC υποεπίπεδο παρέχει τις εξής υπηρεσίες: την MCPS-SAP και την MLME-SAP οι οποίες αποτελούν το προαναφερόμενο interface.

Αξίζει να σημειωθεί ότι το φυσικό επίπεδο παρέχει αντίστοιχα και αυτό ένα interface μεταξύ του MAC και του φυσικού καναλιού. Περιλαμβάνει μία οντότητα διαχείρισης που ονομάζεται PLME (Physical Layer Management Entity). Εμπεριέχει επίσης την υπηρεσία δεδομένων PHY. Η δεύτερη υποστηρίζει τη μεταφορά των MPDUs (MAC protocol data units) μεταξύ των οντοτήτων του MAC. Η PLME-SAP επιτρέπει την ανταλλαγή εντολών μεταξύ MLME και PLME.

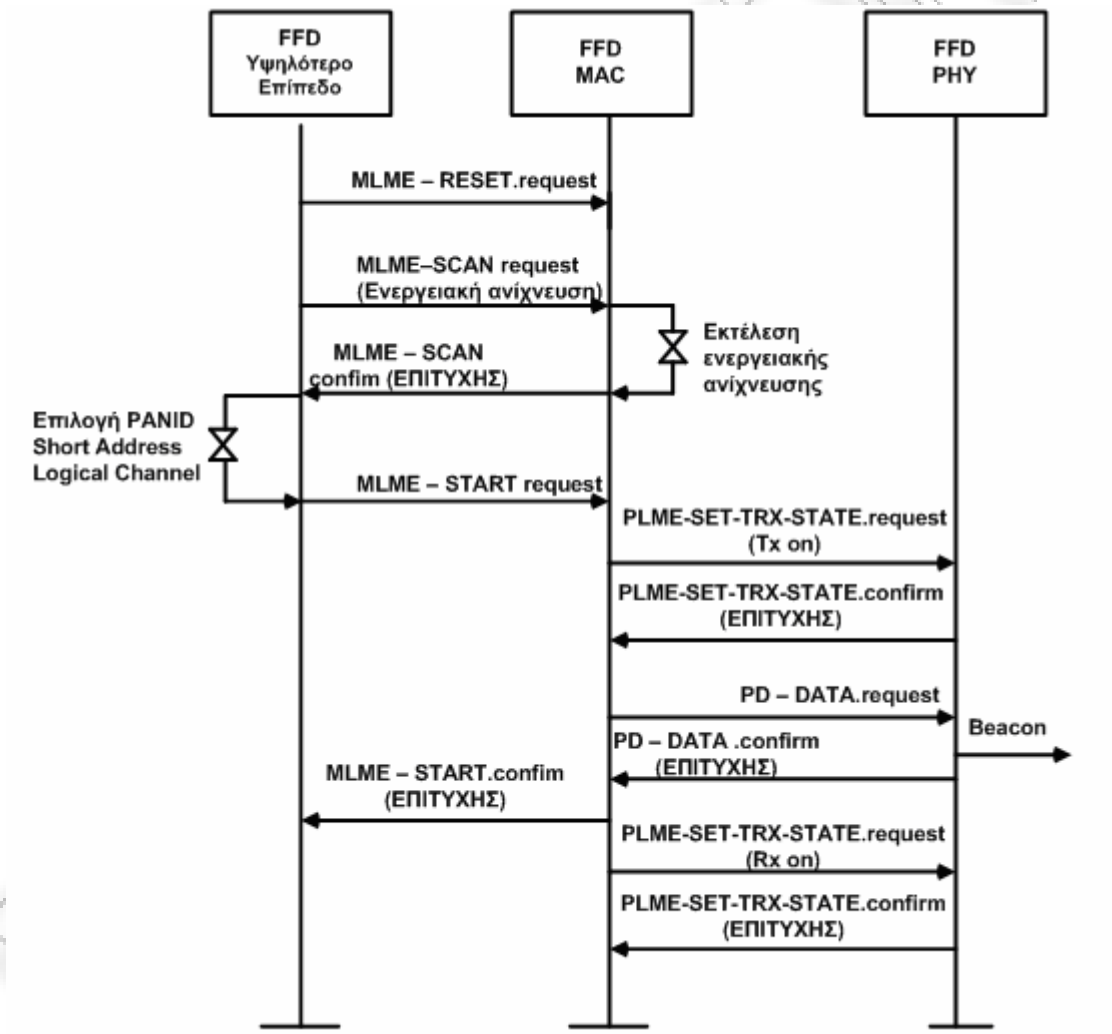
Ανακεφαλαιώνοντας τα παραπάνω, χρειάζονται να αναπτυχθούν τα εξής interfaces:

- MLME: Χρησιμοποιείται για όλες τις εντολές του 802.15.4 MAC.
- MCPS: Χρησιμοποιείται για τις εντολές που σχετίζονται με τα δεδομένα.
- Διασύνδεση μεταξύ MAC και υψηλότερου επιπέδου (π.χ. εφαρμογή).
- PLME (αναφέρεται παραπάνω).

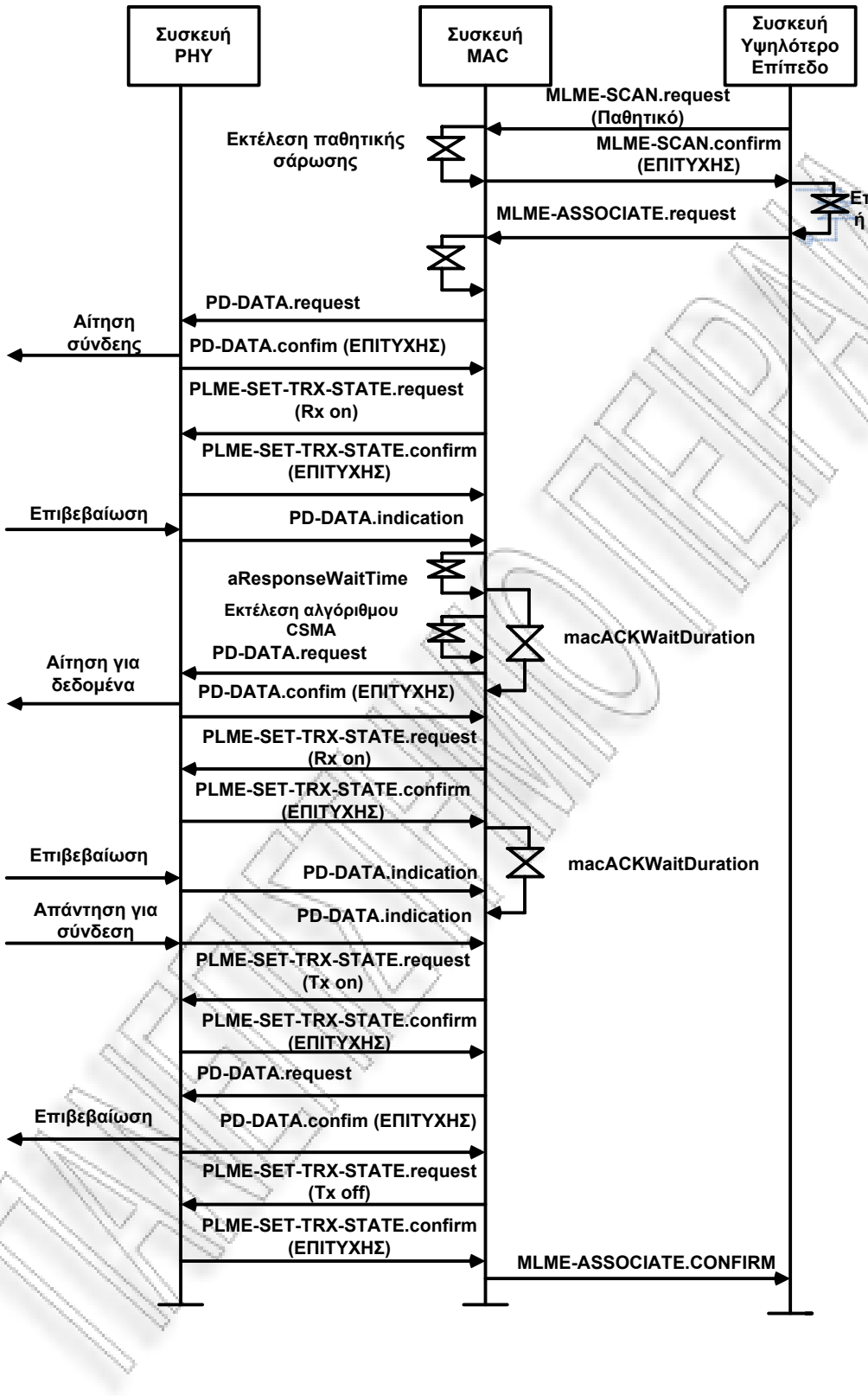
Τα επόμενα διαγράμματα ακολουθίας περιγράφουν την ανταλλαγή μηνυμάτων μεταξύ MAC και PHY, καθώς και MAC και υψηλότερου επιπέδου. Στο σχήμα 2.17 αναφέρονται όλα τα μηνύματα που χρειάζεται ο PAN coordinator προκειμένου να ξεκινήσει ένα καινούριο PAN. Από τη στιγμή που το νέο PAN δημιουργηθεί, ο

[1] Το SSCS διασυνδέει το LLC (logical link control) υποεπίπεδο με το MCPS (MAC common part sublayer) που ασχολείται με την υπηρεσία δεδομένων του MAC.

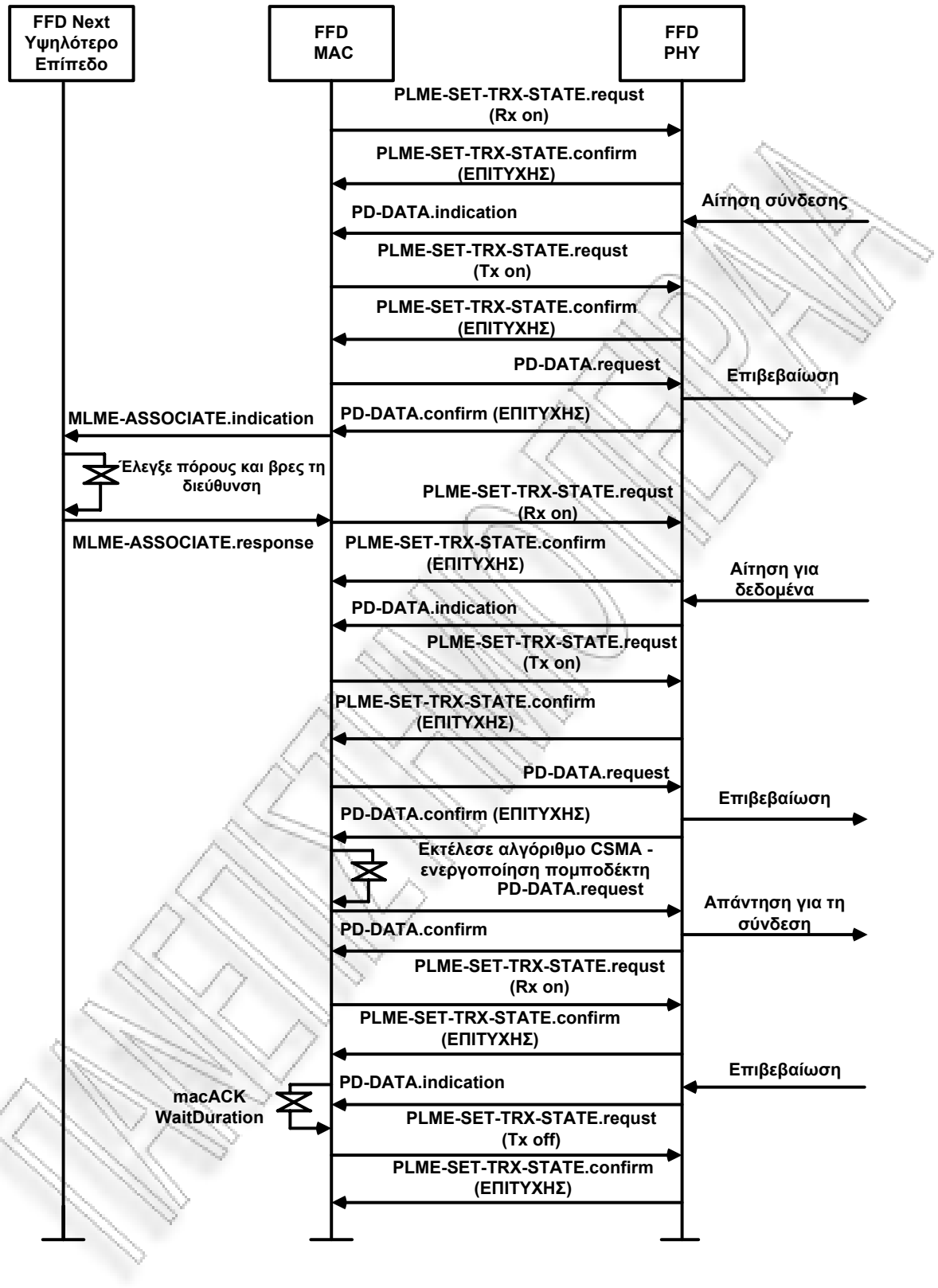
PAN coordinator θα είναι έτοιμος να δεχθεί αιτήσεις για ενσωμάτωση συσκευών (σχήματα 2.18 και 2.19). Στη συνέχεια ακολουθούν τα μηνύματα που πρέπει να ανταλλαχθούν προκειμένου να μεταδοθεί και να ληφθεί ένα απλό πακέτο δεδομένων. Στο σχήμα 2.20 παρουσιάζονται οι ενέργειες που ακολουθεί ο δημιουργός του πακέτου, ενώ στο σχήμα 2.21 οι ενέργειες του παραλήπτη.



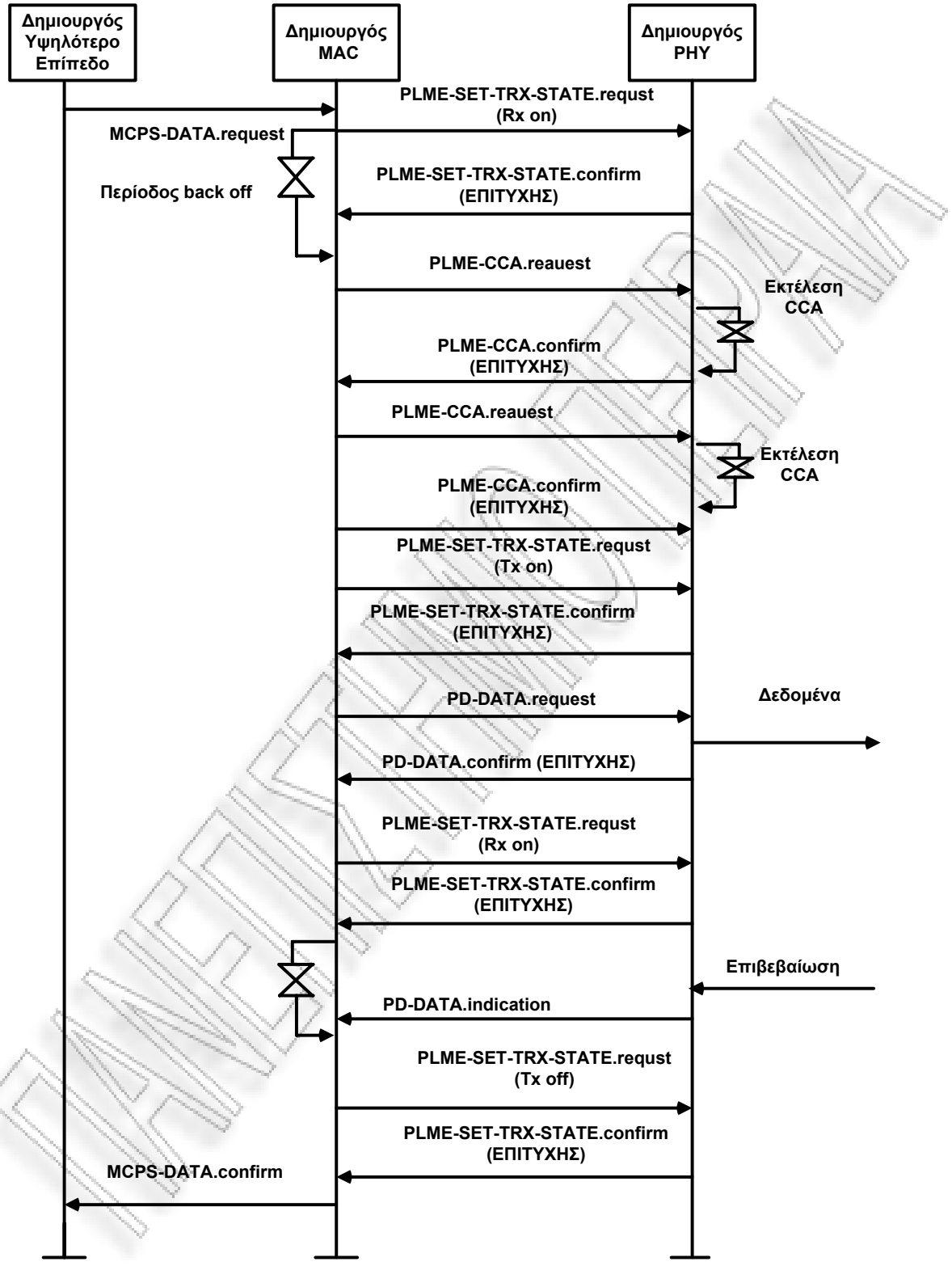
Σχήμα 2.18 Τα μηνύματα που ανταλλάζει ο PAN coordinator προκειμένου να ξεκινήσει ένα καινούριο PAN.



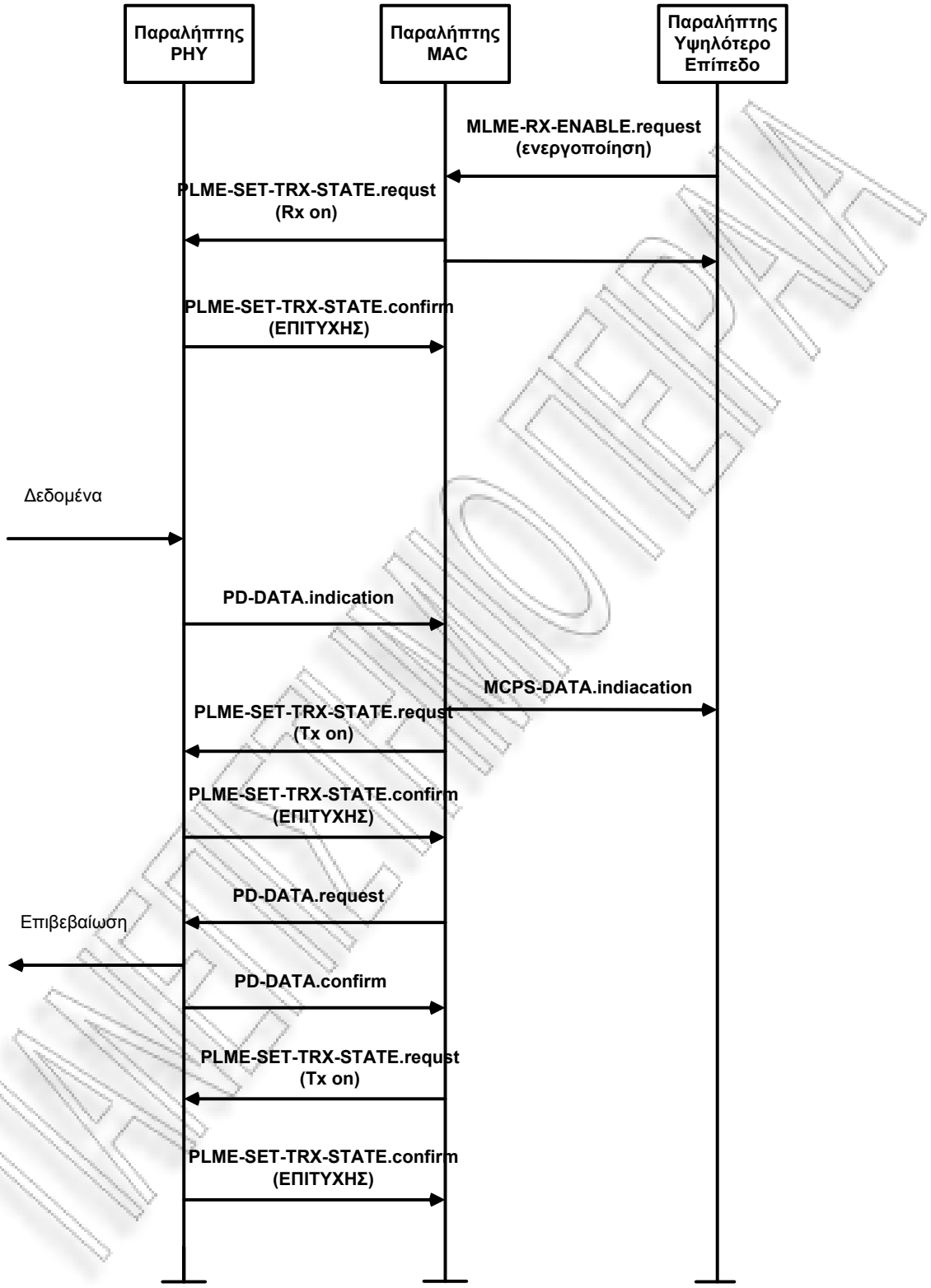
Σχήμα 2.19 Σύνδεση συσκευής με PAN (συσκευή)



Σχήμα 2.20 Σύνδεση συσκευής με PAN (coordinator)



<u>Σχήμα 2.21</u> Οι ενέργειες που ακολουθεί η συσκευή που στέλνει το πακέτο.



Σχήμα 2.22 Ενέργειες του παραλήπτη του πακέτου.

3. ΠΡΟΣΟΜΟΙΩΣΗ ΠΡΩΤΟΚΟΛΛΟΥ 802.15.4 ΜΕ ΤΟ ΠΡΟΓΡΑΜΜΑ NS-2

3.1 Προσομοιώσεις Ασυρμάτων Δικτύων στον NS-2

Η παρακάτω εισαγωγή στις προσομοιώσεις ασυρμάτων δικτύων με το πρόγραμμα NS, αποτελεί παράλληλα και μια εισαγωγή στον προγραμματισμό του SENSASIM. Για να καταλάβουμε πως δουλεύουν οι προσομοιώσεις αυτές, θα αναφέρουμε κάποια βασικά στοιχεία και ορισμούς και στη συνέχεια θα τα προσαρμόσουμε σε ένα απλό παράδειγμα δύο ασυρμάτων κινητών κόμβων.

Ένας κινητός ασύρματος κόμβος αποτελείται από στοιχεία όπως τα: Επίπεδο σύζευξης, Interface Queue (IfQ), υποεπίπεδο MAC, μετάδοση και λήψη σημάτων κτλ. Στην αρχή μιας προσομοίωσης ασυρμάτου δικτύου πρέπει να προσδιορίσουμε το είδος για το κάθε ένα από τα παραπάνω στοιχεία. Επιπρόσθετα χρειάζεται να προσδιορίσουμε και άλλες παραμέτρους όπως τον τύπο της κεραίας, το μοντέλο ραδιο-μετάδοσης, τον τύπο ad-hoc πρωτοκόλλου κ.ά. Στον επόμενο κώδικα του NS παρουσιάζουμε τις αναγκαίες παραμέτρους και την περιγραφή τους με τη μορφή σχολίου [2] [17].

```
# ======================================================================
# Kathorismos Epilogwn
# ======================================================================
set val(chan)         Channel/WirelessChannel  ;# channel type
set val(prop)         Propagation/TwoRayGround ;# radio-propagation model
set val(ant)          Antenna/OmniAntenna      ;# Antenna type
set val(ll)           LL                       ;# Link layer type
set val(ifq)          Queue/DropTail/PriQueue  ;# Interface queue type
set val(ifqlen)       50                       ;# max packet in ifq
set val(netif)        Phy/WirelessPhy          ;# network interface type
set val(mac)          Mac/802_11               ;# MAC type
set val(rp)           DSDV                     ;# ad-hoc routing protocol
set val(nn)           2                        ;# number of mobilenodes
```

Στη συνέχεια πηγαίνουμε στο κυρίως κομμάτι του προγράμματος και ξεκινούμε δημιουργώντας ένα αντικείμενο προσομοιωτή

```
set ns_    [new Simulator]
```

Στη συνέχεια ενεργοποιούμε τη δυνατότητα για 'ίχνη' (αφορά τους agents που αναφέραμε στο 1ο κεφάλαιο) ανοίγοντας το αρχείο simple.tr και καλώντας τη διαδικασία trace-all{}:

```
set tracefd     [open simple.tr w]
$ns_ trace-all $tracefd
```

Στη συνέχεια δημιουργούμε ένα αντικείμενο 'τοπολογίας' το οποίο κρατάει τις κινήσεις των κόμβων μέσα στα όρια της περιοχής δράσης τους.

```
set topo  [new Topography]
```

Στο συγκεκριμένο παράδειγμα οι κόμβοι βρίσκονται σε μια τοπολογία πλέγματος 500m X 500m.

```
$topo load_flatgrid 500 500
```

Στη συνέχεια δημιουργούμε ένα αντικείμενο GOD:

```
create-god $val(nn)
```

Το αντικείμενο GOD χρησιμεύει στην αποθήκευση καθολικών μεταβλητών σχετικά με τη κατάσταση του περιβάλλοντος, του δικτύου και των κόμβων, και οι οποίες δε γίνονται γνωστές προς τους χρήστες. Το αντικείμενο αυτό αποθηκεύει το συνολικό αριθμό από κόμβους και ένα πίνακα με το μικρότερο αριθμό hops που απαιτείται, προκειμένου να το πακέτο να φτάσει στο προορισμό του.

Στη συνέχεια δημιουργούμε τους κόμβους. Στην αρχή όμως πρέπει να τους διαμορφώσουμε κατάλληλα. Έτσι ορίζουμε τον τύπο διευθυνσιοδότησης, τύπο πρωτόκολλου ad-hoc, το επίπεδο σύζευξης, το επίπεδο MAC, κτλ. Για παράδειγμα έχουμε τις παρακάτω ρυθμίσεις:

```
# $ns_ node-config -addressingType flat or hierarchical or expanded
#                  -adhocRouting   DSDV or DSR or TORA
#                  -llType         LL
#                  -macType        Mac/802_11
#                  -propType       "Propagation/TwoRayGround"
#                  -ifqType        "Queue/DropTail/PriQueue"
#                  -ifqLen         50
#                  -phyType        "Phy/WirelessPhy"
#                  -antType        "Antenna/OmniAntenna"
#                  -channelType    "Channel/WirelessChannel"
#                  -topoInstance   $topo
#                  -energyModel    "EnergyModel"
#                  -initialEnergy  (in Joules)
#                  -rxPower        (in W)
#                  -txPower        (in W)
#                  -agentTrace      ON or OFF
#                  -routerTrace     ON or OFF
#                  -macTrace        ON or OFF
#                  -movementTrace   ON or OFF
```

Προκειμένου να δημιουργήσουμε κινητούς κόμβους, ορίζουμε το παρακάτω API:

```
# diamorfwsh komvwn
        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -topoInstance $topo \
                         -channelType $val(chan) \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF
```

Στη συνέχεια δημιουργούμε 2 κινητούς κόμβους ως ακολούθως:

```
for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node ]
```

```
                $node_($i) random-motion 0          ;# apenergopoihsh
tyxaias kinhshs
        }
```

Τώρα, θα ορίσουμε τις αρχικές θέσεις των κόμβων:

```
 #
 # parexontai arxikes (X,Y, kai Z=0) syntetagmenes gia toys node_(0) kai
node_(1)
 #
 $node_(0) set X_ 5.0
 $node_(0) set Y_ 2.0
 $node_(0) set Z_ 0.0

 $node_(1) set X_ 390.0
 $node_(1) set Y_ 385.0
 $node_(1) set Z_ 0.0
```

Ο κόμβος 0 ξεκινάει από τη θέση (5,2) και ο κόμβος 1 από τη θέση (390,385). Στη συνέχεια, αν το επιθυμούμε, παράγουμε κίνηση των κόμβων:

```
 #
 # o Node_(1) ksekinaei na kineitai pros ton node_(0)
 #
 $ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
 $ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"

 # o Node_(1) sth synexeia ksekinaei na apomakrynetai apo ton node_(0)
 $ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0"
```

Επόμενο βήμα είναι να ορίσουμε τη ‘ροή’ της κίνησης μεταξύ των δύο κόμβων:

```
 # syndeseis TCP metaksy twn node_(0) kai node_(1)

 set tcp [new Agent/TCP]
 $tcp set class_ 2
 set sink [new Agent/TCPSink]
 $ns_ attach-agent $node_(0) $tcp
 $ns_ attach-agent $node_(1) $sink
 $ns_ connect $tcp $sink
 set ftp [new Application/FTP]
 $ftp attach-agent $tcp
 $ns_ at 10.0 "$ftp start"
```

Ακολούθως, χρειάζεται να ορίσουμε το χρόνο λήξης της προσομοίωσης και να δώσουμε την εντολή για αναστοιχειοθέτηση.

```
#
# Leei stous komvous pote teleiwnei h prosomoiwsh
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0001 "stop"
$ns_ at 150.0002 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    close $tracefd
}
```

Ακολουθεί η τελική εντολή για να ξεκινήσει η προσομοίωση:

```
puts "Starting Simulation..."
$ns_ run
```

3.2 Ανάλυση Αποτελεσμάτων με τη Γλώσσα AWK

Με τη γλώσσα awk μπορούμε να αναλύσουμε αρχεία με δεδομένα. Ένα πρόγραμμα awk αποτελείται από τρία τμήματα. Το πρώτο τμήμα του ορίζεται με την εντολή BEGIN { } και περιλαμβάνει όλες τις εντολές που θα γίνουν μία φορά, κατά την εκκίνηση του προγράμματος. Εδώ μπορούν να αρχικοποιηθούν μεταβλητές, να ανοιχτούν αρχεία κτλ. Το δεύτερο τμήμα του προγράμματος αποτελείται από ένα σύνολο από κανόνες που θα εκτελεστούν για κάθε γραμμή του αρχείου εισόδου. Αυτοί οι κανόνες αποτελούνται από δύο κομμάτια. Το πρώτο κομμάτι ορίζει σε ποιες γραμμές του αρχείου εισόδου αναφέρεται ο κανόνας, και το δεύτερο ορίζει ποιες λειτουργίες θα πραγματοποιηθούν σε αυτές τις γραμμές.

Ο παρακάτω κώδικας μας αναφέρει ότι όταν συναντήσουμε μία γραμμή του αρχείου εισόδου που να ξεκινάει με τον χαρακτήρα “r” (/^r/) και (&&) που περιλαμβάνει τη λέξη cbr (/cbr/), τότε η τιμή της μεταβλητής packets αυξάνεται

κατά 1, η τιμή της μεταβλητής data αυξάνεται κατά την τιμή που βρίσκεται στην έκτη λέξη ($6) της γραμμής την οποία εξετάζουμε, και η αύξηση της τιμής της μεταβλητής sumDelay ισούται με τη διαφορά της τιμής που βρίσκεται στην δεύτερη θέση της εξεταζόμενης γραμμής, μείον την τιμή του πίνακα sendtimes στην θέση $12%bufferspace.

```
/^r/&&/cbr/ {
    data+=$6;
    packets++;
    sumDelay += $2 - sendtimes[$12%bufferspace];
}
```

Για να μπορούμε να υπολογίσουμε την χρησιμοποίηση (utilization) του καναλιού, πρέπει να μετρήσουμε την ποσότητα των δεδομένων που ελήφθησαν κατά τη διάρκεια της προσομοίωσης. Στη συνέχεια αυτός αριθμός θα διαιρεθεί με την διάρκεια της προσομοίωσης, ώστε να υπολογιστεί ο ρυθμός διέλευσης δεδομένων από το κανάλι (throughput), και αυτή η τιμή θα διαιρεθεί με τον ονομαστικό ρυθμό μετάδοσης του καναλιού ώστε να προκύψει η χρησιμοποίηση (utilization). Η καθυστέρηση κάθε πακέτου προκύπτει αν αφαιρεθεί ο χρόνος λήψης κάθε πακέτου από τον χρόνο αποστολής του. Ως χρόνο αποστολής ενός πακέτου θεωρούμε την χρονική στιγμή κατά την οποία το πακέτο εξέρχεται από την ουρά αναμονής στον κόμβο αποστολής.

Για την εκτέλεση προγραμμάτων awk χρησιμοποιείται ο διερμηνέας (interpreter) awk με παραμέτρους το όνομα του αρχείου που περιγράφει τις διαδικασίες της ανάλυσης και το όνομα του αρχείου που περιλαμβάνει τα δεδομένα που θα αναλυθούν. Εάν ο παραπάνω κώδικας έχει αποθηκευτεί στο αρχείο test.awk και τα δεδομένα βρίσκονται στο αρχείο test.tr, τότε εκτελούμε την εντολή:

```
awk.exe -f test.awk < test.tr
```

Αυτή η εντολή θα εκτυπώσει στην οθόνη τον αριθμό των πακέτων και το πλήθος των δεδομένων που ελήφθησαν. Για να αποθηκεύσουμε τα αποτελέσματα της ανάλυσης στο αρχείο results.txt εκτελούμε την εντολή

```
awk.exe -f test.awk < test.tr > results.txt
```

3.3 Σενάρια Προσομοίωσης

Στο πρόγραμμα NS-2 έχει υλοποιηθεί μεγάλο μέρος του πρωτοκόλλου IEEE 802.15.4, ενώ από την πλευρά της δρομολόγησης των πακέτων έχει χρησιμοποιηθεί η ZBR (ZigBee Routing). Προκειμένου να 'τρέξει' το πρόγραμμα χρειαζόμαστε την έκδοση 2.26 ή μεταγενέστερη του NS και την έκδοση P802.15.4/D18 του πρωτοκόλλου. Η έκδοση αυτή περιλαμβάνει τις εξής λειτουργίες:

- Μηχανισμός CSMA-CA
- Συμβατότητα με το 802.11b
- Τοπολογίες αστέρα και peer-to-peer
- Ενεργή / μη ενεργή μετάδοση beacons
- Ανακάλυψη beacon και συγχρονισμός
- Σύνδεση / αποσύνδεση με PAN
- Διαμορφώσεις δέντρων peer-to-peer και συστάδων
- Άμεσες / έμμεσες μεταδόσεις
- Ενεργειακή ανίχνευση
- Αξιολόγηση καθαρού καναλιού (Clear Channel Assessment - CCA)
- Ανίχνευση ποιότητας ζεύξης (Link Quality Detection - LQD)
- Υποστήριξη πολλαπλών καναλιών
- Σάρωση καναλιού
- Φιλτράρισμα
- Γραφική απεικόνιση με το πρόγραμμα NAM

Η επόμενη εντολή χρησιμοποιείται για να ξεκινήσουμε ένα καινούριο PAN. Ο κόμβος που θα ανταποκριθεί θα αρχίσει να δρα ως διαχειριστής:

```
$node sscs startPANCoord <txBeacon = 1> <beaconOrder = 3>
<SuperframeOrder = 3>
```

Παράδειγμα:

```
$node_(0) sscs startPANCoord
$node_(0) sscs startPANCoord 1 2 2
```

Η επόμενη εντολή εκκινεί μια συσκευή ή ένα διαχειριστή:

```
$node sscs startDevice <isFFD = 1> <assoPermit = 1> <txBeacon
= 0> <beaconOrder = 3> <SuperframeOrder = 3>
```

Παράδειγμα:

```
$node_(0) sscs startDevice 0 //device
$node_(0) sscs startDevice //coor., non-beacon
$node_(0) sscs startDevice 1 1 1 //coor., beacon enabled
```

Με τον επόμενο κώδικα δίνουμε την εντολή να ξεκινήσει ένα PAN σε διάταξη cluster tree (δέντρο σε συστάδες):

```
$node sscs startCTPANCoord <txBeacon = 1>
<beaconOrder = 3> <SuperframeOrder = 3>
```

Ο επόμενος κώδικας ξεκινάει μια συσκευή σε διάταξη cluster tree:

```
$node sscs startCTDevice <isFFD = 1> <assoPermit =
1> <txBeacon = 0> <beaconOrder = 3>
<SuperframeOrder = 3>
```

Εκκίνηση μετάδοσης beacons σε κατάσταση δικτύου χωρίς beacons ή αλλαγή της σειράς των τιμών ‘beacon order’ και ‘superframe order’ σε κατάσταση δικτύου με beacons:

```
$node sscs startBeacon <beaconOrder = 3>
<SuperframeOrder = 3>
```

Λήξη της μετάδοσης beacons:

```
$node sscs stopBeacon
```

Στο σενάριο που εκτελέσαμε έχουμε ορίσει ως αλγόριθμο δρομολόγησης την AODV (Ad-hoc On-demand Distance Vector) η οποία δρομολογεί δεδομένα σε δίκτυα πλέγματος. Αναφερόμαστε σε δίκτυο peer-to-peer με ενεργοποιημένη την αποστολή beacons. Θα προσομοιώσουμε 11 κόμβους αισθητήρα και θα δούμε

γραφικά τα αποτελέσματα με το πρόγραμμα NAM. Ορίζουμε έναν διαχειριστή PAN, 5 διαχειριστές και 5 απλές συσκευές. Η περιοχή που βρίσκονται οι κόμβοι έχει έκταση 50m x50m και η απόσταση μεταξύ γειτονικών κόμβων είναι περίπου 10m. Η διάρκεια της επικοινωνίας είναι 900 sec, ενώ έχουμε ενεργοποιήσει τη χρήση beacons. Η εμβέλεια του κάθε κόμβου είναι 15m. Στον επόμενο κώδικα διαμορφώνουμε το εικονικό μας δίκτυο:

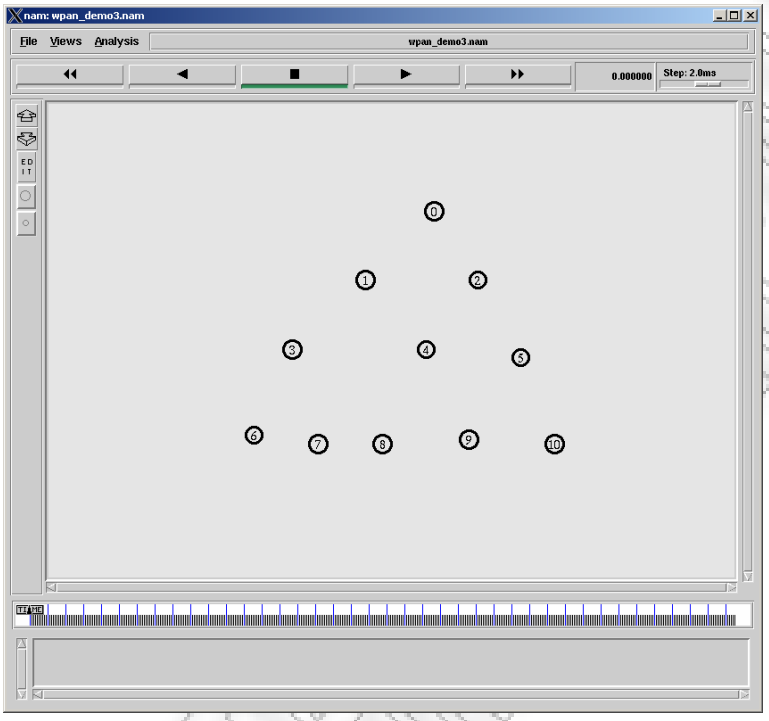
```
set val(chan)        Channel/WirelessChannel   ;# Channel Type
set val(prop)        Propagation/TwoRayGround  ;# radio-propagation model
set val(netif)       Phy/WirelessPhy/802_15_4
set val(mac)         Mac/802_15_4
set val(ifq)         Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)          LL                        ;# link layer type
set val(ant)         Antenna/OmniAntenna       ;# antenna model
set val(ifqlen)      50                        ;# max packet in ifq
set val(nn)          11                        ;# number of mobilenodes
set val(rp)          AODV                      ;# routing protocol
set val(x)           50
set val(y)           50
set val(nam)         wpan_demo3.nam
set val(traffic)     ftp                       ;# cbr/poisson/ftp
```

Σε περιβάλλον Linux (ή Cygwin αν έχουμε λειτουργικό σύστημα Windows) εκτελούμε την εντολή `$ startx` και στη συνέχεια μετακινούμαστε στο φάκελο που έχουμε το σενάριο `$ cd /usr/local/demo` και εκτελούμε την εντολή `$ ns wpan_demo3.tcl`.

Ακολουθεί η σειρά μηνυμάτων που ανταλλάσσεται μεταξύ των κόμβων αισθητήρων. Αρχικά εμφανίζεται η ενέργεια που εκτελείται π.χ. εκκίνηση διαχειριστή PAN `--- startPANCoord [0] ---` Στη συνέχεια εμφανίζεται η χρονική στιγμή εκτέλεσης του γεγονότος πχ. `[0.140800],` ποιος κόμβος δημιουργεί το γεγονός `(node 0)` και τι ακριβώς εκτελεί `performing active channel scan.` Για παράδειγμα η πρόταση `[1.787648](node 1) association successful (beacon enabled) [channel:11] [PAN_ID:0]` μας αναφέρει ότι τη χρονική στιγμή 1.787648 ο κόμβος αισθητήρα με αναγνωριστικό 1, συνδέθηκε επιτυχώς με το PAN που έχει αναγνωριστικό ίσο με 0.

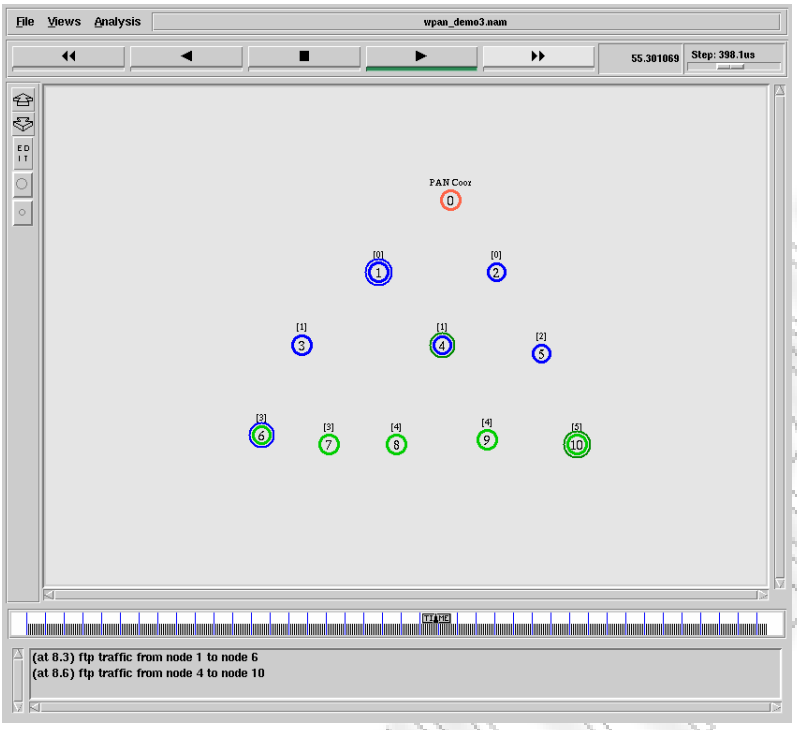
Πιο αναλυτικά ο κώδικας παρουσιάζεται στο Παράρτημα Α.1 και η ακολουθία μηνυμάτων στο Παράρτημα Α.2.

Στη συνέχεια εμφανίζεται στην οθόνη του υπολογιστή το περιβάλλον του NAM, και εμφανίζονται γραφικά οι κόμβοι.



Σχήμα 3.1 Εμφάνιση των κόμβων αισθητήρα στο πρόγραμμα NAM

Ορίζεται ως διαχειριστής PAN ο κόμβος 0 και ξεκινά η ανταλλαγή μηνυμάτων. Πάνω από κάθε κόμβο εμφανίζεται το ID του προηγούμενου κόμβου που κινούνταν το πακέτο.



Σχήμα 3.2 Ανταλλαγή μηνυμάτων

Στο περιβάλλον του Cygwin εκτελούμε την εντολή:

```
$ awk.exe -f delay.awk < wpan_demo3.tr > results.txt
```

Στην παραπάνω εντολή εκτελούμε το αρχείο delay.awk το οποίο όπως αναφέραμε και στο κεφάλαιο 3.2 χρησιμεύει στην επεξεργασία των αποτελεσμάτων. Με τη χρήση του αρχείου αυτού υπολογίζουμε το συνολικό throughput και τις μέσες καθυστερήσεις στην πρόσβαση. Το throughput (S) αφορά το μέρος της κυκλοφορίας που λαμβάνεται σωστά, κανονικοποιημένο με τη χωρητικότητα του δικτύου (250kbps), δηλαδή στην ουσία αποτελεί ένα μέτρο της απόδοσης του δικτύου. Η μέση καθυστέρηση (D) αφορά τη καθυστέρηση που αντιμετωπίζει ένα πλαίσιο δεδομένων από τη στιγμή που παράγεται μέχρι τη στιγμή που θα παραληφθεί. Τα παραπάνω μεγέθη συγκρίνονται σε σχέση με το προσφερόμενο φορτίο (G).

Ελάττωση της κατανάλωσης ισχύος μπορεί να επιτευχθεί, όταν οι συσκευές λειτουργούν με χαμηλό duty cycle (ο λόγος της διάρκειας κατά την οποία μια συνάρτηση δεν είναι μηδέν, προς την περίοδο της συνάρτησης) , κάτω από 0.1%.

Για να είναι πιο αποδοτικό το σύστημα πρέπει να μειωθεί η μέση καθυστέρηση (delay) και να βελτιωθεί η κατάσταση sleep (κατάσταση 'ύπνου').

Ο κώδικας του αρχείου delay.awk παρουσιάζεται αναλυτικά στο Παράρτημα Α.3.

Θα μεταβάλλουμε τη διάρκεια της προσομοίωσης, μεταβάλλοντας με αυτόν τον τρόπο το φορτίο κίνησης, καθώς το τελευταίο εξαρτάται από το πόσα πακέτα στάλθηκαν και για πόση διάρκεια. Στη συνέχεια θα μεταβάλλουμε τις τιμές των 'beacon order'(BO) και 'superframe order'(SO) (στο αρχείο 'wpan_demo3.tcl') ώστε να δείξουμε πόσο οι μεταβολές αυτές επηρεάζουν το throughput και τη μέση καθυστέρηση:

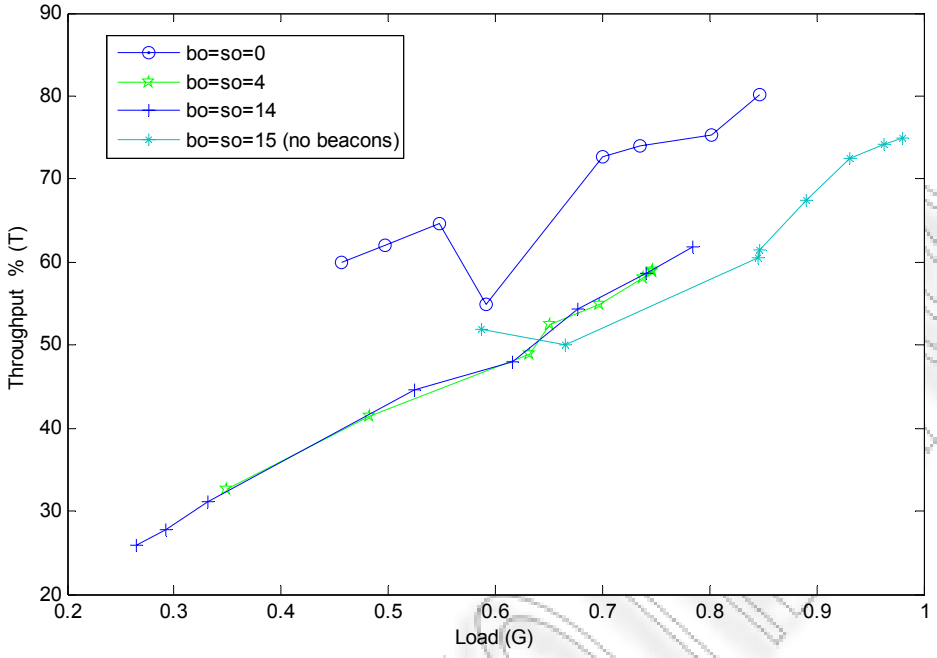
```
$ns_ at 9.0 "$node_(5) sscs startBeacon 0 0"  ;# change beacon order and
superframe order
```

Με αυτόν τρόπο θα μπορέσουμε να διαπιστώσουμε για ποιες τιμές της μέσης καθυστέρησης, μπορεί να μειωθεί το duty cycle που έχει ως αποτέλεσμα τη μείωση της κατανάλωσης ενέργειας [11]. Στους παρακάτω πίνακες το throughput είναι κανονικοποιημένο ως προς τη χωρητικότητα 250 kbps και η μέση καθυστέρηση εκφράζεται σε msec.

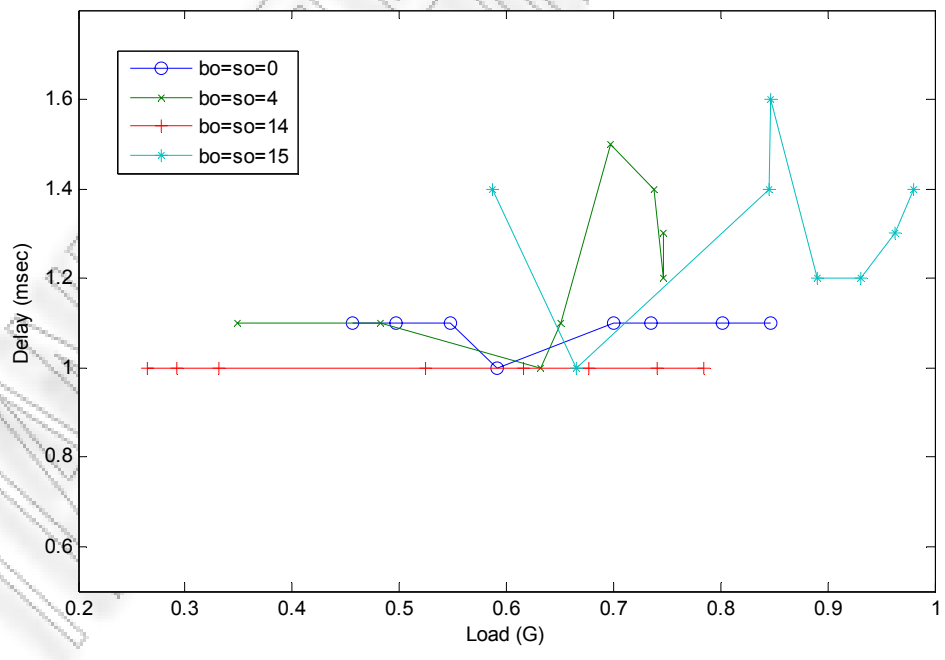
Πίνακας 3.1 Αποτελέσματα προσομοιώσεων

BO = SO = 0								
Runtime	20	40	50	70	100	200	300	500
Offered Load	0.5916	0.8016	0.8467	0.7357	0.7009	0.5475	0.4965	0.4556
Av.Throughput	55%	75,40%	80.1%	74%	72,60%	64,66%	62%	59.9%
Av.Delay (msec)	1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
BO = SO = 4								
Offered Load	0.6319	0.6967	0.7376	0.7470	0.7460	0.6506	0.4830	0.3489
Av.Throughput	49%	55%	58%	58.8%	59.1%	52.4%	41.37%	32.55%
Av.Delay (msec)	1	1.5	1,4	1.3	1.2	1.1	1.1	1.1
BO = SO = 14								
Offered Load	0.6153	0.7414	0.7842	0.6764	0.5241	0.3319	0.2925	0.2648
Av.Throughput	48%	58.6%	61.8%	54.4%	44.6%	31,12%	27.8%	25.9%
Av.Delay (msec)	1	1	1	1	1	1	1	1
BO = SO = 15 (χωρίς beacons)								
Offered Load	0.6655	0.9810	0.9632	0.9310	0.8911	0.8466	0.8460	0.5875
Av.Throughput	50%	74.9%	74,10%	72,50%	67,40%	61,50%	60.6%	52%
Av.Delay (msec)	1	1.4	1.3	1.2	1.2	1.6	1.4	1.4

Για BO=SO=0 και για φορτίο περίπου ίσο με 0.74 παρατηρούμε ότι παίρνουμε καλύτερα αποτελέσματα σε σχέση με BO=SO=4 ή BO=SO=14. Για BO=SO=14 και φορτίο μικρότερο από 0.7 παρατηρούμε ότι το μέσο throughput είναι πολύ χαμηλό αλλά ταυτόχρονα είναι χαμηλή και η μέση καθυστέρηση. Αν απενεργοποιήσουμε τα beacons, το μέσο throughput διατηρείται σε υψηλά επίπεδα. Στα επόμενα διαγράμματα παρουσιάζεται το throughput σε σχέση με το φορτίο και η καθυστέρηση σε σχέση με το φορτίο.



Σχήμα 3.3 Throughput σε συνάρτηση με το προσφερόμενο φορτίο.



Σχήμα 3.4 Η καθυστέρηση σε συνάρτηση με το προσφερόμενο φορτίο.

4. ΠΡΟΣΟΜΟΙΩΣΗ MAC 802.15.4 ΜΕ ΤΟ ΠΡΟΓΡΑΜΜΑ SENSASIM

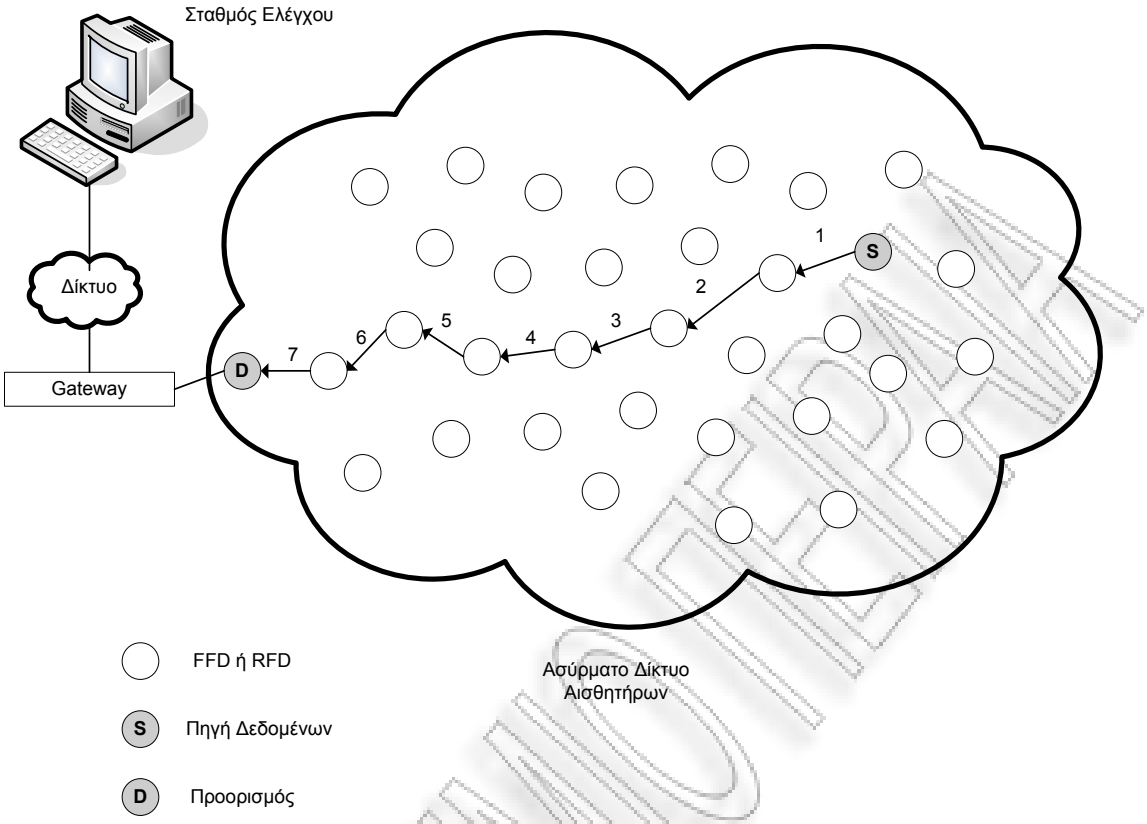
4.1 Επιλογή Κατάλληλων Παραμέτρων και Λειτουργιών

Προβλήματα και λύσεις στη σχεδίαση

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο η κατάλληλη μορφή επικοινωνίας είναι η σημείο-προς-σημείο (ή καλύτερα peer-to-peer). Σε αυτήν τη μορφή επικοινωνίας και όταν αναφερόμαστε σε ασύρματα δίκτυα αισθητήρων χρησιμοποιείται συνήθως μετάδοση προς όλους τους 'γείτονες' ενός κόμβου (ή αλλιώς broadcast). Επιπρόσθετα, οι εφαρμογές των προαναφερόμενων δικτύων είναι πολύ μεγάλης κλίμακας και δημιουργείται η ανάγκη για εξελιξιμότητα. Τα δίκτυα που δεν έχουν ενεργοποιημένο το μηχανισμό για αποστολή beacons παρουσιάζουν μεγαλύτερη προσαρμογή στην εξελιξιμότητα του δικτύου σε σχέση με τα δίκτυα που έχουν ενεργοποιημένα τα beacons [12].

Στην πραγματικότητα, απενεργοποιώντας την κατάσταση 'αποστολής beacons', είναι πιο εύκολη η δημιουργία ενός δικτύου peer-to-peer. Αυτό συμβαίνει διότι οι κόμβοι είναι ανεξάρτητοι από το διαχειριστή PAN και η επικοινωνία είναι αποκεντρωμένη. Επιπρόσθετα, σε δίκτυα αισθητήρων μεγάλης κλίμακας αν επιλεγεί μηχανισμός CSMA χωρίς τη χρήση χρονοθυρίδων, θα επιτραπεί μεγαλύτερη ευελιξία στην επικοινωνία του δικτύου peer-to-peer καθώς δεν απαιτεί συγκεκριμένο συγχρονισμό, σε αντίθεση με το μηχανισμό CSMA με χρήση χρονοθυρίδων.

Το πρόβλημα βέβαια με την επιλογή μηχανισμού CSMA χωρίς τη χρήση χρονοθυρίδων, είναι ότι δεν υπάρχει εγγυημένη αποστολή δεδομένων μέσα σε κάποια χρονικά πλαίσια. Ας θεωρήσουμε λοιπόν το απλό σενάριο της αποστολής δεδομένων από κάποιον κόμβο S (source) προς έναν άλλο κόμβο D (destination) με δρομολόγηση αρκετών βημάτων.



<u>Σχήμα 4.1</u> Απλό σενάριο αποστολής δεδομένων

Ένα πακέτο δεδομένων όταν ακολουθεί την παραπάνω πορεία θα παρουσιάσει καθυστέρηση η οποία σχετίζεται με τον αριθμό των hops που πραγματοποιεί το πακέτο, την απόσταση, το ρυθμό δεδομένων, τη πρόσβαση στο μέσο και πρωτόκολλα δρομολόγησης. Στα μεγάλα δίκτυα αισθητήρων ο αριθμός των hops είναι αρκετά μεγάλος. Αν λάβουμε υπόψη μας το χαμηλό ρυθμό μετάδοσης και την ανάγκη για χαμηλή κατανάλωση ενέργειας, προκύπτει ότι η καθυστέρηση μετάδοσης δεν είναι αμελητέα. Αυτό μπορούμε να το ελαχιστοποιήσουμε με τη χρήση των GTS και των beacons, τα οποία όμως επιτρέπουν επικοινωνία μεταξύ των κόμβων και το διαχειριστή PAN. Με αυτόν τον τρόπο όμως, μεταφερόμαστε από τοπολογία peer-to-peer σε τοπολογία αστέρα, με όλα τα μειονεκτήματα που αναφέραμε στις προηγούμενες παραγράφους.

Βήματα λειτουργίας

Σύμφωνα με την ανάλυση του υποκεφαλαίου 4.1, η κατάλληλη επικοινωνία δικτύου για τα ασύρματα δίκτυα αισθητήρων είναι η peer-to-peer. Παρόλαυτα οι παρακάτω ενέργειες είναι η γενικότερη μορφή επικοινωνίας σε ένα PAN. Με αυτόν τον τρόπο όταν ενεργοποιηθεί το 802.15.4 MAC στον προσομοιωτή, θα πρέπει να γίνονται ενδεικτικά:

- Γίνεται αρχικοποίηση του 802.15.4 MAC.
- Ο coordinator πραγματοποιεί ανίχνευση με Energy Detection προκειμένου να επιλέξει λογικό κανάλι. Αυτό πραγματοποιείται πριν γίνει ακόμα διαχειριστής PAN.
- Γίνεται εκκίνηση του διαχειριστή PAN. Τώρα το MAC απαντάει σε active scans από τις συσκευές.
- Παράλληλα οι συσκευές πραγματοποιούν ενεργές σαρώσεις για την εύρεση κατάλληλου διαχειριστή PAN.
- Ο διαχειριστής απαντάει σε αιτήματα διασύνδεσης από τις συσκευές.
- Παράλληλα οι συσκευές αιτούνται για σύνδεση με τον coordinator.
- Ο διαχειριστής λαμβάνει δεδομένα από τη συσκευή.
- Παράλληλα η συσκευή στέλνει δεδομένα στον διαχειριστή.
- Ο διαχειριστής στέλνει έμμεσα δεδομένα στη συσκευή (δεδομένα που είναι ήδη αποθηκεμένα στον coordinator).
- Η συσκευή περιμένει δεδομένα που εκκρεμούν στον διαχειριστή.
- Ο διαχειριστής ξεκινάει ένα δίκτυο με beacons. Αυτό σημαίνει ότι υπάρχει συγχρονισμός και η συσκευή λαμβάνει άμεσα τα δεδομένα από τον coordinator.

Λόγω της επικοινωνίας peer-to-peer οι συσκευές θα μπορούν να στέλνουν άμεσα δεδομένα μεταξύ τους και να μην εξαρτώνται από τον διαχειριστή, οπότε τα τελευταία πέντε βήματα μπορούν να μην εφαρμοστούν.

4.2 Επεξήγηση Κώδικα Java

Οι βασικές κλάσεις του προγράμματος περιέχονται στα αρχεία SensorNode.java, Simulation.java, MacAloha.java, IPClass.java, testscenarios.java. Η συνηθισμένη ακολουθία μεθόδων και κλάσεων πχ για δρομολόγηση Broadcast Flood είναι συνοπτικά:

- Εκτέλεση εφαρμογής testscenarios.java – π.χ. επιλέγουμε το σενάριο για Broadcast Flood άρα τη μέθοδο Scenario_IP_Broadcast_Flood.
- Στη συνέχεια εκτελείται η μέθοδος Run_ IP_Broadcast_Flood που βρίσκεται στην κλάση Simulation.java. Η μέθοδος αυτή καλεί τη κλάση IP_Send προκειμένου να ξεκινήσει τη δρομολόγηση των πακέτων.
- Με τη σειρά της καλεί τη κλάση IPClass.
- Η IPClass μέσω των μεθόδων IP_Receive και IP_Broadcast_Flood δρομολογεί τα δεδομένα και καλεί τη κλάση MacAloha για τον έλεγχο της σωστής λήψης από τον παραλήπτη.

Η κλάση SensorNode περιέχει όλες τις απαραίτητες μεθόδους και μεταβλητές ώστε να προσδιοριστεί με ακρίβεια ένας ασύρματος κόμβος με αισθητήρα. Περιλαμβάνει μεθόδους για προσδιορισμό του φυσικού επιπέδου, υπολογισμό της κατευθυντικότητας, και τους κατάλληλους ελέγχους για το ποιοι κόμβοι βρίσκονται στην εμβέλεια του. Θέλουμε η κλάση αυτή να κληρονομήσει χαρακτηριστικά που να την προσδιορίζουν ως FFD ή RFD [4] [5]. Για αυτό το λόγο δημιουργούμε τη κλάση Mac_802_15_4 η οποία ‘κληροδοτεί’ την SensorNode με όλα τα απαραίτητα στοιχεία:

```
public class SensorNode extends MAC_802_15_4{
```

Στη συνέχεια ορίζουμε ένα καινούριο κατασκευαστή της κλάσης που ορίζει αν ο κόμβος έχει τη δυνατότητα να γίνει PAN Coordinator ή όχι.

```
    public SensorNode(int index, String name, double coorX, double
coorY, NodeConfig pNodeConfig, boolean PANCoordinator_) {
```

```
PANCoordinator = PANCoordinator_;
```

Το πρώτο βήμα του κώδικα αφορά στη δημιουργία των κόμβων μέσα σε κατάλληλη διάταξη. Όλες οι παράμετροι που επιλέγουμε είναι ίδιες με εκείνες που επιλέξαμε και στο σενάριο που εκτελέσαμε στο Κεφάλαιο 3.3. Έτσι αναφερόμαστε σε δίκτυο πλέγματος με διαστάσεις 50x50 και σε αριθμό κόμβων ίσο με 11. Τους κόμβους τους δημιουργούμε με τη μέθοδο TestCreateNodes2, την οποία προσθέτουμε στη κλάση Simulation. Εκεί ορίζουμε και το ποιοι κόμβοι έχουν τη δυνατότητα να γίνουν διαχειριστές του PAN.

Στη συνέχεια μεταφερόμαστε στη μέθοδο SetupSimulation της κλάσης testscenarios. Η μέθοδος αυτή αρχικοποιεί και εκκινεί μια καινούρια προσομοίωση. Στην επιλογή της τοπολογίας και πιο συγκεκριμένα στο σενάριο για grid (πλέγματος) προσθέτουμε τη κλήση της μεθόδου που δημιουργήσαμε πριν, της TestCreateNodes2. Ακολούθως, καλείται η μέθοδος TestConnectNodes της κλάσης Simulation, η οποία διασυνδέει τους κόμβους μεταξύ τους αναλόγως με την εμβέλεια τους, τη θέση τους και τις επιμέρους ρυθμίσεις τους. Στην ουσία εκτελείται μια *ενεργειακή ανίχνευση – Energy Detection (ED)*, την οποία τη χρειαζόμαστε για την υλοποίηση της εκκίνησης του PAN και της σύνδεσης των συσκευών με το διαχειριστή. Αφου πραγματοποιηθεί ο παραπάνω έλεγχος, ο κάθε κόμβος προσπαθεί να συνδεθεί με κάποιον άλλο καλώντας τη μέθοδο tryToConnectWith της κλάσης SensorNode.

Η μέθοδος tryToConnectWith μελετώντας τη τοπολογία και τις θέσεις των κόμβων, υπολογίζει την απαραίτητη εμβέλεια για σύνδεση. Σε περίπτωση που κάποιος κόμβος είναι εκτός εμβέλειας σε σχέση με κάποιον άλλο κόμβο, η σύνδεση δεν μπορεί να πραγματοποιηθεί.

```
 int ono = otherNode.getSensorIndex();
 System.out.println("Node Address: "+ ono+" out of range of (node
"+sensorIndex+") ");
```

Σε περίπτωση που πληρούνται όλες οι ενεργειακές προϋποθέσεις εκτελούμε τη μέθοδο associate. Η μέθοδος αυτή ελέγχει αν ο κόμβος που ελέγχεται αυτή τη στιγμή έχει τη δυνατότητα να λειτουργήσει ως διαχειριστής (είναι FFD δηλαδή). Αν η μεταβλητή bAdd είναι true τότε έχει τις δυνατότητες.

```
if (bAdd ) {
    if(dMaxGain == 0) dMaxGain = 1;
    this.addNeighbor(otherNode, iLobe, dMaxGain);
    this.associate(otherNode);
    //this.associate(otherNode, iLobe, dMaxGain);
  }
```

Στη συνέχεια ελέγχει αν ο άλλος κόμβος (με τον οποίο θέλει ο πρώτος κόμβος να συνδεθεί) είναι ήδη συνδεδεμένος σε κάποιο PAN, είτε αν είναι ένας διαφορετικός διαχειριστής PAN (με διαφορετικό PAN id). Αν για όλα τα παραπάνω ισχύει: ότι ο πρώτος κόμβος είναι όντως FFD (δηλαδή ικανός για PAN coordinator), ο δεύτερος κόμβος δεν είναι συνδεμένος με κάποιο άλλο PAN και τέλος ο δεύτερος κόμβος δεν είναι διαχειριστής άλλου PAN, θα ξεκινήσει η ανταλλαγή μηνυμάτων για σύνδεση του δεύτερου στο PAN που διαχειρίζεται ο πρώτος κόμβος.

```
if (this.PANCoordinator == true && otherNode.isConnected() == false &&
otherNode.getPANCoordinator() == false){
```

Θα σταλεί μήνυμα που θα περιλαμβάνει την αίτηση του δεύτερου για σύνδεση με το PAN (association request)[2]. Στη συνέχεια στέλνουμε μέσω MAC και του μηχανισμού CSMA το μήνυμα του δεύτερου και περιμένουμε αν θα μας απαντήσει θετικά ο πρώτος κόμβος. Αν πάρουμε θετική απάντηση (association response) ξεκινούν οι διαδικασίες ενσωμάτωσης του δεύτερου κόμβου στο PAN που ορίζει ο πρώτος κόμβος με τη χρήση της μεθόδου addAssociateDev.

```
            if (ACK == true ){

                System.out.print("(Node    "+sensorIndex+")    Ack    for
association request received by (node "+ono+")");
                System.out.println();
                otherNode.setCoordPANId( sensorIndex );
                System.out.println("Association:  successful  -     Node
Address: "+ ono+", PAN Address: "+ sensorIndex );
                System.out.println(" ");
                this.addAssociatedDev(otherNode);
```

[2] Έχουμε προσθέσει κατάλληλο κώδικα στη κλάση Message ώστε τα μηνύματα να δέχονται εντολές και αντικείμενα του MLME.

```
                }
                else  {
                   System.out.println("Ack    for    association    request
received");
                   System.out.println("Association: unsuccessful" );
                }
```

Στη συνέχεια τρέχουμε το σενάριο όπου ο κόμβος 4 θέλει να στείλει δεδομένα στον κόμβο 3 μέσω του διαχειριστή PAN. Ακολούθως θα τρέξουμε το σενάριο όπου ο κόμβος 4 στέλνει άμεσα τα δεδομένα στον κόμβο 3. Θα συγκρίνουμε τις τιμές της υπολειπόμενης ενέργειας του διαχειριστή PAN καθώς και των δύο συναλλασσόμενων κόμβων και στις δύο περιπτώσεις. Για την πρώτη περίπτωση, στη μέθοδο SetupSimulation() της κλάσης testscenarios καλούμε τη μέθοδο dataTransmission() της κλάσης Simulation. Η μέθοδος dataTransmission() προσομοιώνει την επικοινωνία των δύο κόμβων που προαναφέραμε έχοντας ως κεντρικό ρυθμιστή, το διαχειριστή PAN. Αρχικοποιούμε τον αποστολέα, τον παραλήπτη και 'ψάχνουμε' ποιος κόμβος είναι διαχειριστής του PAN.

```
int numberofnodes = this.sensorNetworkNodes.size();
SensorNode sender = (SensorNode)this.sensorNetworkNodes.elementAt(4);
int senderPANid = sender.getCoordPANId();
SensorNode PANcoord =
(SensorNode)this.sensorNetworkNodes.elementAt(senderPANid);
SensorNode receiver = (SensorNode)this.sensorNetworkNodes.elementAt(3);
 int receiverId = receiver.getSensorIndex();
```

Στη συνέχεια ο κόμβος 4 στέλνει τα δεδομένα στο διαχειριστή PAN:

```
Message Msg1 = new Message(1, "Run_IP_Broadcast_Flood","Data
sending...",receiverId, 0);
this.ResetSimulation(this.getONodeConfig());
IPSend oIPSend = new IPSend(0, sender, PANcoord, Msg1, sender,
this.oIPClass.eBroadcastFlooding);
oIPSend.setDaemon(true);
oIPSend.start();
```

ο οποίος τα αποθηκεύει:

```
PANcoord.setDataQueue( Msg1 );
```

Ο κόμβος 3 στέλνει μήνυμα στο διαχειριστή του και τον ρωτά αν εκκρεμούν δεδομένα για εκείνον. Εκείνος του απαντά θετικά σε περίπτωση που εκκρεμούν δεδομένα για αυτόν τον κόμβο.

```
if (pend == true ){
    System.out.println("Data is pending for (Node 3): PAN coord
    sending pending data to receiver (Node:3)");
    System.out.println();

    IPSend oIPSend3 = new IPSend(0, PANcoord, receiver,
    PANcoord.getDataQueue(), PANcoord, this.oIPClass.eBroadcastFlooding);
    oIPSend3.setDaemon(true);
    oIPSend3.start();
    this.setDone(true);
    this.WaitForSimulationToFinish();
 }
    else{
          System.out.println("There is no data pending...");
    }
```

Η μεταβλητή *pend* που μας δείχνει αν εκκρεμούν ή όχι δεδομένα για κάποιον κόμβο ορίζεται στην κλάση IPClass.

```
try{
  if  ( msg.getMlmeObj().equals("Ask for Pending Data..." ) ){
          this.oSimulation.setPend(true);
   }
    else this.oSimulation.setPend(false);
    }catch(NullPointerException npe){ }
```

Για το δεύτερο σενάριο θα χρησιμοποιήσουμε με τον ίδιο τρόπο, τη μέθοδο dataTransmission2() της κλάσης Simulation.

Για τα καταχώρηση των διαφόρων μεγεθών που μας ενδιαφέρουν, χρησιμοποιούμε τη κλάση tempStats.

Ο χρησιμοποιούμενος κώδικας, καθώς και οι προσθήκες σε κλάσεις που ήδη ενσωματώνει το Sensasim παρουσιάζονται στο Παράρτημα Β.

4.3 Οδηγός Χρήσης του Sensasim και Προσομοίωση Δικτύου

Όταν ξεκινήσει το πρόγραμμα, θα εμφανιστεί ένα παράθυρο που θα μας δείχνει τις παραμέτρους που μπορούμε να ορίσουμε. Οι παράμετροι είναι χωρισμένες σε έξι κατηγορίες:

- Topology
- Physical
- MAC
- IP
- General
- Compilation

Θα αναλύσουμε λοιπόν τις παραμέτρους για κάθε κατηγορία [7]. Για το σενάριο που θα τρέξουμε, επιλέγουμε παρόμοιες τιμές με αυτές που επιλέξαμε και στο πρόγραμμα NS-2 στο κεφάλαιο 3.

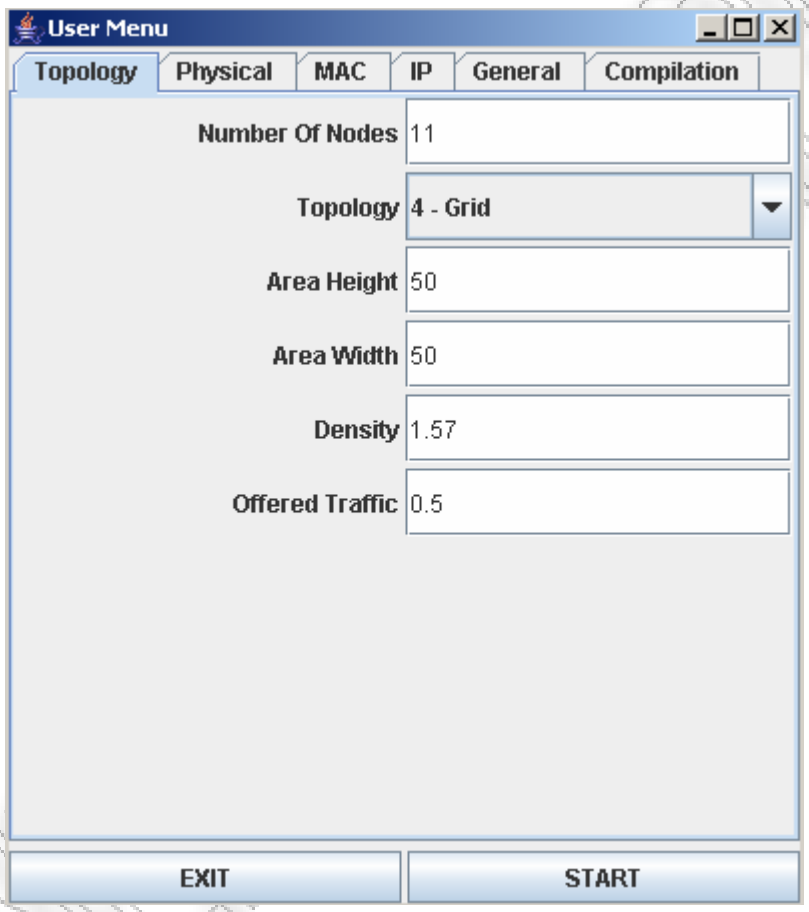
Topology

- Number of Nodes: Αναφέρεται στον αριθμό των κόμβων με αισθητήρα που περιλαμβάνει το δίκτυο.
- Topology: Υπάρχουν 4 διαφορετικές κατηγορίες τοπολογιών στο δίκτυο. Τυχαία, τυχαία με κενό στο κέντρο, τυχαία με μεγάλο κενό στο κέντρο, και τέλος η τοπολογία πλέγματος.
- Area Height: Αναφέρεται στο μέγεθος της περιοχής όπου υπάρχει το δίκτυο και πιο συγκεκριμένα στο μήκος του.
- Area Width: Αναφέρεται στο μέγεθος της περιοχής όπου υπάρχει το δίκτυο και πιο συγκεκριμένα στο πλάτος του.
- Density: Στην παρούσα έκδοση του προγράμματος ο μέσος αριθμός γειτονικών κόμβων, καθορίζεται από την πυκνότητα και όχι από τη παράμετρο της εμβέλειας που καθορίζεται στη κατηγορία ‘Physical’. Τρεις συνηθισμένες τιμές πυκνότητας είναι οι: 1.57 (οδηγεί σε περίπου 7

‘γείτονες’), 2.04 (οδηγεί σε περίπου 12 ‘γείτονες’), 2.48 (οδηγεί σε περίπου 17 ‘γείτονες’).

- Offered Traffic: Αυτή η παράμετρος χρησιμοποιείται αν η κατηγορία ‘scenario’ έχει ενεργοποιημένη την παράμετρο “ALOHA validation”.

Στην επόμενη εικόνα παρουσιάζουμε τις τιμές για την κατηγορία ‘Topology’.



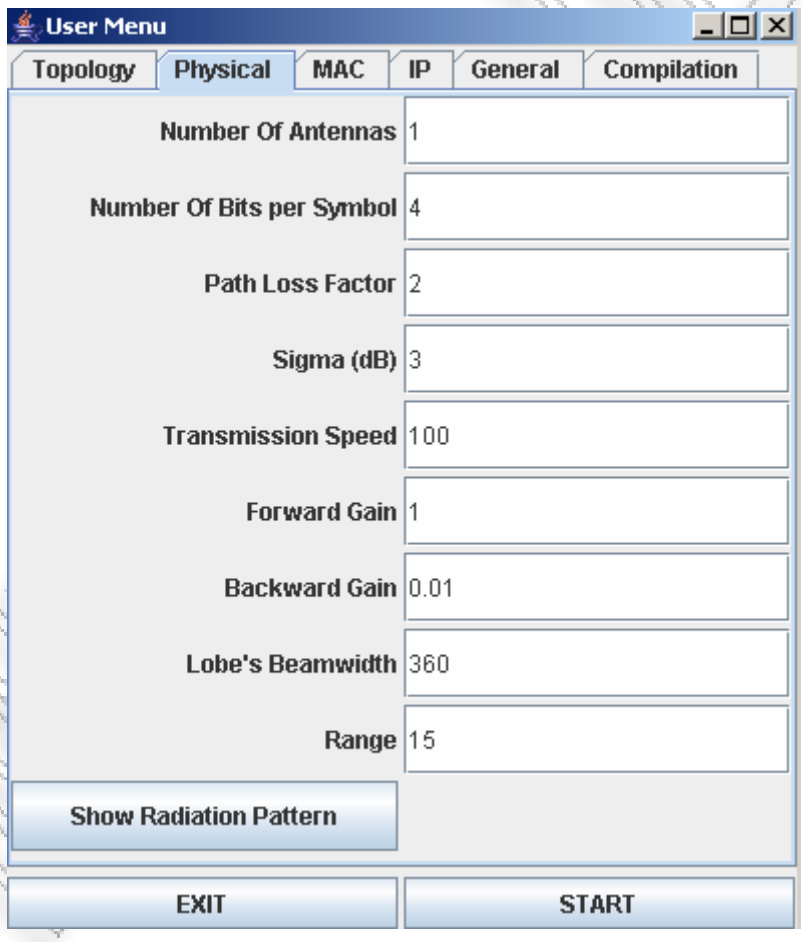
Σχήμα 4.2 Παράμετροι κατηγορίας ‘topology’.

Physical

- Number of Antennas: ο αριθμός των κεραιών κάθε κόμβου
- Forward Gain: ο λόγος του εύρους του κατευθυντικού λοβού προς εκείνον μιας ομοιοκατευθυντικής κεραίας. Συνήθως η τιμή αυτή τίθεται στο 1.

- Backward Gain: ο λόγος του εύρους του κατευθυντικού λοβού προς την αντίθετη κατεύθυνση. Η τιμή αυτή είναι πολύ μικρή.
- Lobe's Beamwidth: Αυτή η παράμετρος καθορίζει τη γωνία όπου η ισχύς του μεταδιδόμενου σήματος είναι τουλάχιστον το μισό του μέγιστου κέρδους.
- Range: Καθορίζει την απόσταση μέσα στην οποία ένας κόμβος μπορεί να εγκαθιδρύσει μια ραδιο-σύνδεση.
- Show Radiation Button: Πατώντας αυτό το κουμπί εμφανίζεται στην οθόνη το διάγραμμα ακτινοβολίας κάποιου κόμβου.

Στην επόμενη εικόνα παρουσιάζουμε τις τιμές για την κατηγορία 'Physical'.

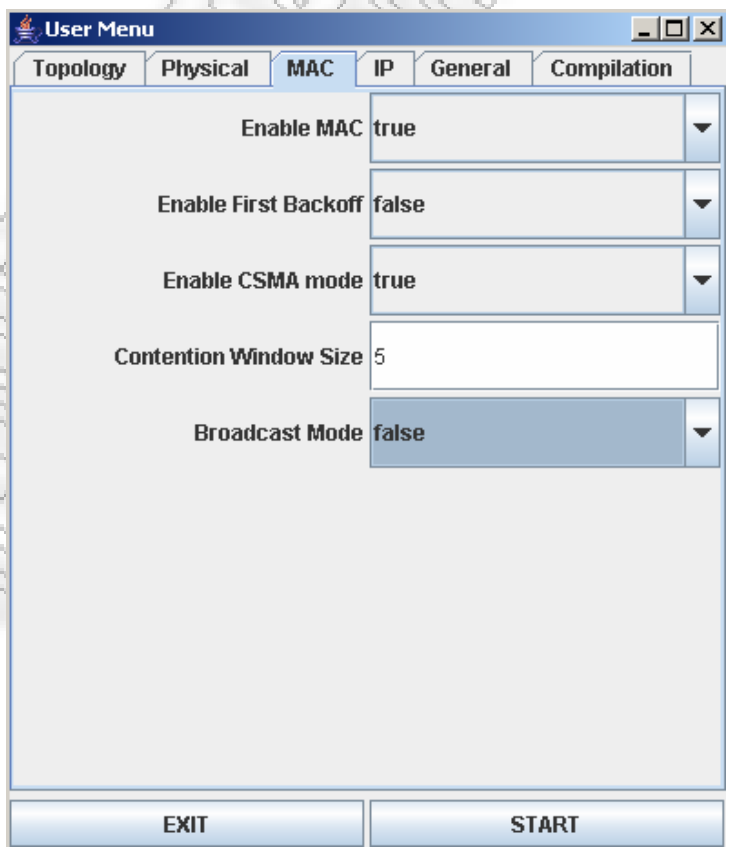


Σχήμα 4.3 Παράμετροι κατηγορίας 'Physical'.

MAC

- Enable MAC: θέτοντας την τιμή true ενεργοποιούμε το υποεπίπεδο ελέγχου πρόσβασης μέσου.
- Enable First Backoff: με αυτήν την παράμετρο επιλέγουμε αν οι κόμβοι θέλουμε να περιμένουν ένα τυχαίο χρονικό διάστημα μέχρι να μεταδώσουν κάποιο πακέτο.
- Enable CSMA mode: επιλέγουμε αν θέλουμε να ενεργοποιήσουμε το μηχανισμό CSMA.
- Contention Window size: επιλέγουμε το μέγεθος του 'παραθύρου' του εκθετικού αλγόριθμου για back off.
- Broadcast Mode: κάθε μήνυμα στέλνεται προς όλους τους γειτονικούς κόμβους και δεν λειτουργεί ο μηχανισμός αναμετάδοσης σε περίπτωση σύγκρουσης πακέτων.

Στην επόμενη εικόνα παρουσιάζουμε τις τιμές για την κατηγορία 'MAC'.

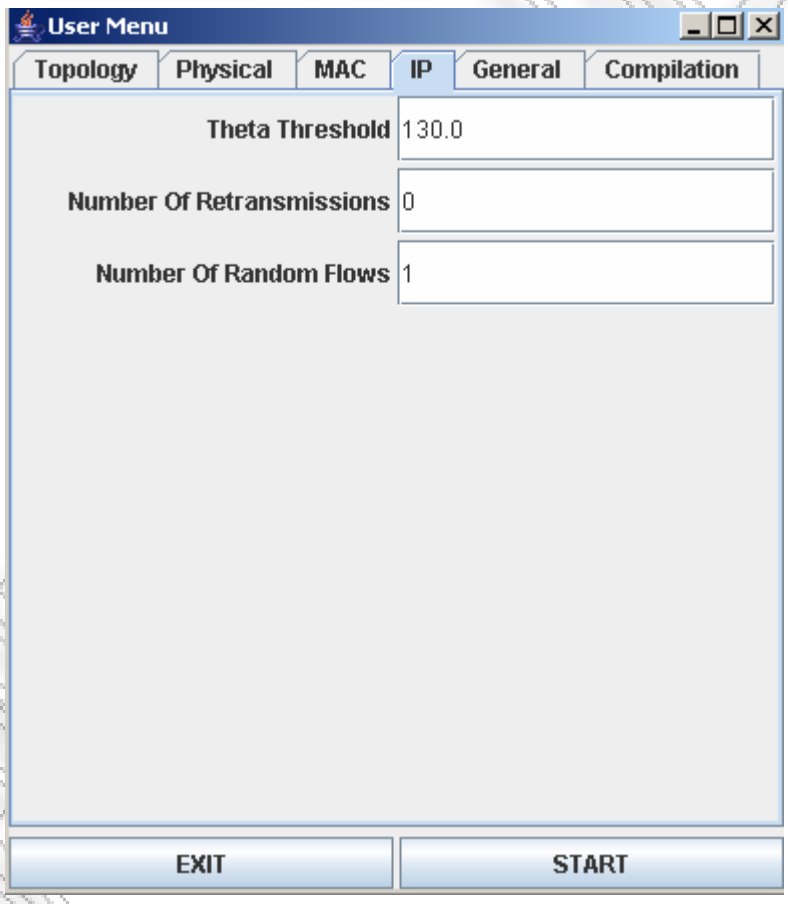


Σχήμα 4.4 Παράμετροι κατηγορίας 'MAC'.

IP

- Theta Threshold: είναι η γωνία του κατώτερου ορίου που χρησιμοποιεί ο αλγόριθμος δρομολόγησης SPEED. Η τιμή ορίζεται σε μοίρες.
- Number of retransmissions: ο αριθμός των αναμεταδόσεων ενός πακέτου σε περίπτωση σύγκρουσης.
- Number of Random Flows: με αυτήν την τιμή καθορίζουμε τον αριθμό των ροών κίνησης που παράγει τυχαία το δίκτυο.

Στην επόμενη εικόνα παρουσιάζουμε τις τιμές για την κατηγορία 'IP'

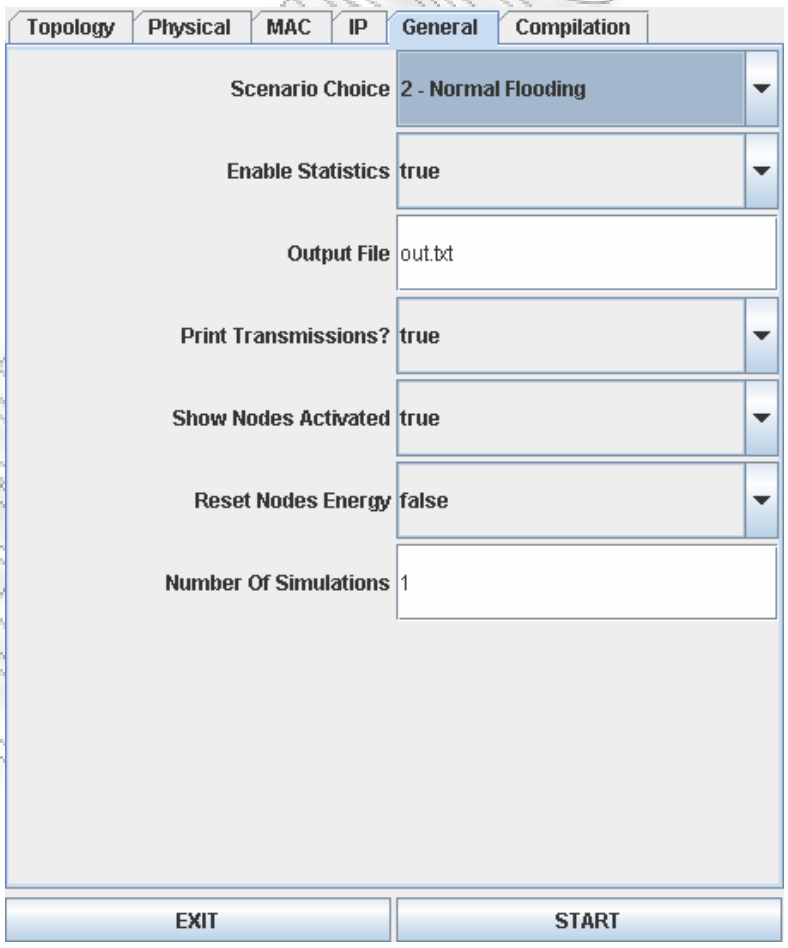


Σχήμα 4.5 Παράμετροι κατηγορίας 'IP'.

General

- Scenario Choice: επιλογή του σεναρίου. Αναφέρεται κυρίως σε μεθόδους δρομολόγησης.
- Enable Statistics: ενεργοποίηση της αποστολής των παραγόμενων στοιχείων, σε κάποιο εξωτερικό αρχείο.
- Output file: όνομα του αρχείου που συγκρατεί τα στοιχεία.\
- Print Transmissions?: επιλέγουμε το αν θέλουμε να εμφανίζονται στην οθόνη τα μηνύματα του MAC.
- Show Nodes Activated: επιλέγουμε να θέλουμε την εμφάνιση των κόμβων που δρομολογούν δεδομένα.

Στην επόμενη εικόνα παρουσιάζουμε τις τιμές για την κατηγορία 'General'.

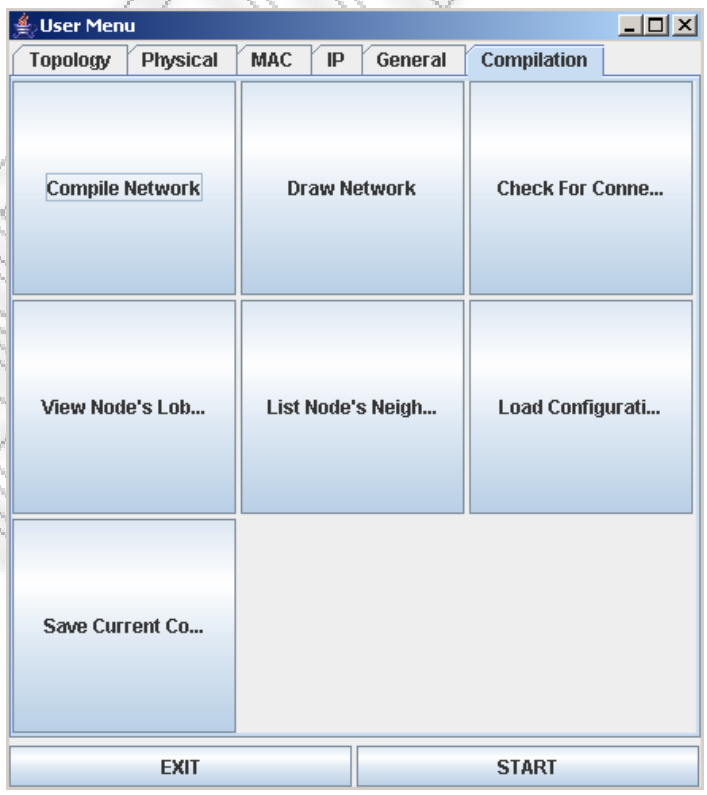


Σχήμα 4.6 Παράμετροι κατηγορίας 'General'.

Compilation

- Compile Network button: ενεργοποίηση και αρχικοποίηση του δικτύου. Εκτελείται πριν πατήσουμε οποιοδήποτε άλλου κουμπί.
- Draw Network button: οι κόμβοι εμφανίζονται γραφικά στην οθόνη.
- Check For Connectivity button: προς το παρόν δεν χρησιμοποιείται.
- View Node's lobes button: το κουμπί αυτό το πατάμε, αφού έχουμε πατήσει πρώτα το κουμπί 'Draw Network'. Σχεδιάζει στην οθόνη το διάγραμμα ακτινοβολίας ενός συγκεκριμένου κόμβου.
- List Node's neighbors button: το κουμπί αυτό το πατάμε, αφού έχουμε πατήσει πρώτα το κουμπί 'Draw Network'. Εμφανίζει τους γειτονικούς κόμβους κάποιου κόμβου που επιλέγουμε.
- Load Configuration File button: φόρτωση ενός αρχείου κειμένου, με προκαθορισμένες τιμές παραμέτρων.
- Save Current Configuration button: αποθήκευση των παραμέτρων που επιλέξαμε σε ένα αρχείο κειμένου.

Στην επόμενη εικόνα παρουσιάζουμε την κατηγορία 'Compilation'.



Σχήμα 4.7 Παράμετροι κατηγορίας 'Compilation'.

Πατάμε το κουμπί ‘Compile Network’ και αρχικά ξεκινάει η διαδικασία δημιουργίας καινούριων PAN και σύνδεσης συσκευών (κόμβων με αισθητήρα) με τα PANs. Τα αποτελέσματα που λαμβάνουμε είναι για τον πρώτο κόμβο:

```
Starting PAN Coordinator, PAN id: 0
Scanning...
Node Range: 27.0
Node 0 Energy Left: 10.0
 (Node : 1) sending association request to (PAN id: 0) at [3052]
[3592]: (Node 0) Ack for association request received by (node 1)
Association: successful -  Node Address: 1, PAN Address: 0
Node 0 Energy Left: 10.0
```

όπως παρατηρούμε εμφανίζονται και το ενεργειακό υπόλοιπο κάθε κόμβου.

Ξεκινάει ένα καινούριο PAN με id = 0. Ο κόμβος 1 στέλνει αίτηση για σύνδεση προς το διαχειριστή με id = 0 τη χρονική στιγμή [2840]. Αφού ελεγχθεί η εντολή association response που λαμβάνεται τη στιγμή [3380], αν είναι θετική εμφανίζεται μήνυμα επιτυχούς σύνδεσης. Η νέα διεύθυνση του κόμβου είναι η 0 1 ([PAN address] [Device Address]). Μπορεί κάποια στιγμή να σταλεί το μήνυμα μέσω άλλων κόμβων:

```
(Node : 2) sending association request to (PAN id: 0) at
SENDING: 5     at Timestamp: 1576      FROM: 0         TO: 1
RECVING: 2       at Timestamp: 2116      FROM: 0         TO: 1
SENDING: 6       at Timestamp: 5309      FROM: 0         TO: 2
RECVING: 3       at Timestamp: 5849      FROM: 0         TO: 2
 (Node 0) Ack for association request received by (node 2)
ssociation: successful -  Node Address: 2, PAN Address: 0
```

Άλλο παράδειγμα: επιτυχής σύνδεση του κόμβου 9 με το διαχειριστή με id ίσο με 10.

```
(Node 10) Ack for association request received by (node 9)
Association: successful -  Node Address: 9, PAN Address: 10
Node 10 Energy Left: 9.999692711999998
```

Θα παρουσιάσουμε συνοπτικά τα αποτελέσματα της αποστολής δεδομένων από τον κόμβο 4 στον κόμβο 3, με κεντρικό ρυθμιστή το διαχειριστή PAN:

```
(Node : 4) sending data to (Node: 3)
(Node : 4) sending data to (PAN id: 0)
SENDING: 84      at Timestamp: 345       FROM: 4         TO: 0
.
.
.

(Node: 4) successfully sent data to (PAN: 0)

(Node: 3) Asks (PAN: 0) for Pending Data.
SENDING: 95      at Timestamp: 805       FROM: 3         TO: 0
.
.
.
Data is pending for (Node 3): PAN coord sending pending data to receiver
(Node:3)

Node 3 Energy Left: 9.998610982026769
Node 4 Energy Left: 9.999071914026771
PAN Coordinator Energy Left: 9.999321477610707
```

Στο τέλος εμφανίζονται τα συνολικά αποτελέσματα των μεγσθών που μας ενδιαφέρουν (όπου EOS είναι ο συνολικός χρόνος προσομοίωσης):

```
Total Messages Sent: 492
Total Messages Received: 600
Total Energy Left: 0.07800961797066797
EOS: 955745
```

Θα παρουσιάσουμε συνοπτικά τα τελικά αποτελέσματα της αποστολής δεδομένων από τον κόμβο 4 στον κόμβο 3, χωρίς να έχουμε κεντρικό ρυθμιστή το διαχειριστή PAN:

```
(Node: 4) successfully sent data to (Node: 3)
Node 3 Energy Left: 9.998950243221415
Node 4 Energy Left: 9.999289504416062
PAN Coordinator Energy Left: 9.999481559416063
```

Πατάμε το κουμπί ‘List Node’s Neighbors’ και επιλέγουμε τον κόμβο 0. Εμφανίζονται πληροφορίες για τους γειτονικούς κόμβους του 0 και οι συσκευές που είναι συνδεδεμένες με αυτόν.

```
Neighbor Lobes:
Lobe: 0, 1
Lobe: 0, 2
Lobe: 0, 3
Lobe: 0, 4
Lobe: 1, 3
Lobe: 1, 4

Interference Neighbor Lobes:
Lobe: 0, 1
Lobe: 0, 2
Lobe: 0, 3
Lobe: 0, 4
Lobe: 0, 7
Lobe: 1, 3
Lobe: 1, 4
Lobe: 1, 7

Node 0 is a PAN Coordinator. Associated Devices:
Sensor: 1
Sensor: 2
Sensor: 3
Sensor: 4
```

4.4 Σενάρια Προσομοίωσης

Τα αποτελέσματα που μας ενδιαφέρουν είναι η συνολική ενέργεια που καταναλώθηκε κατά τη διάρκεια λειτουργίας των κόμβων, ο συνολικός χρόνος προσομοίωσης, τα μηνύματα που στάλθηκαν και τέλος τα μηνύματα που μεταδόθηκαν με επιτυχία. Τα μηνύματα αυτά εκτυπώνονται στο αρχείο ‘results.txt’. Με βάση αυτά τα αποτελέσματα θα μπορέσουμε να υπολογίσουμε τα εξής μεγέθη: throughput, φορτίο, ενέργεια που καταναλώθηκε ανά sec και ενέργεια που καταναλώθηκε ανά επιτυχώς σταλμένο bit. Για τον υπολογισμό αυτών των μεγεθών θα μας βοηθήσουν οι παρακάτω σχέσεις:

- Φορτίο (πακέτα / διάρκεια πακέτου)

$$Load = \frac{Message \quad Sent}{\frac{End \quad of \quad Simulation \quad Time(EOS)}{Packet \quad Time}}$$

- Throughput (bits / sec)

$$Throughput = \frac{(Message \quad \text{Re}ceived)\cdot(Packet \quad Size)\cdot 10^6}{EOS \quad Time}$$

- Ενέργεια που καταναλώθηκε προς χρόνο (J / sec)

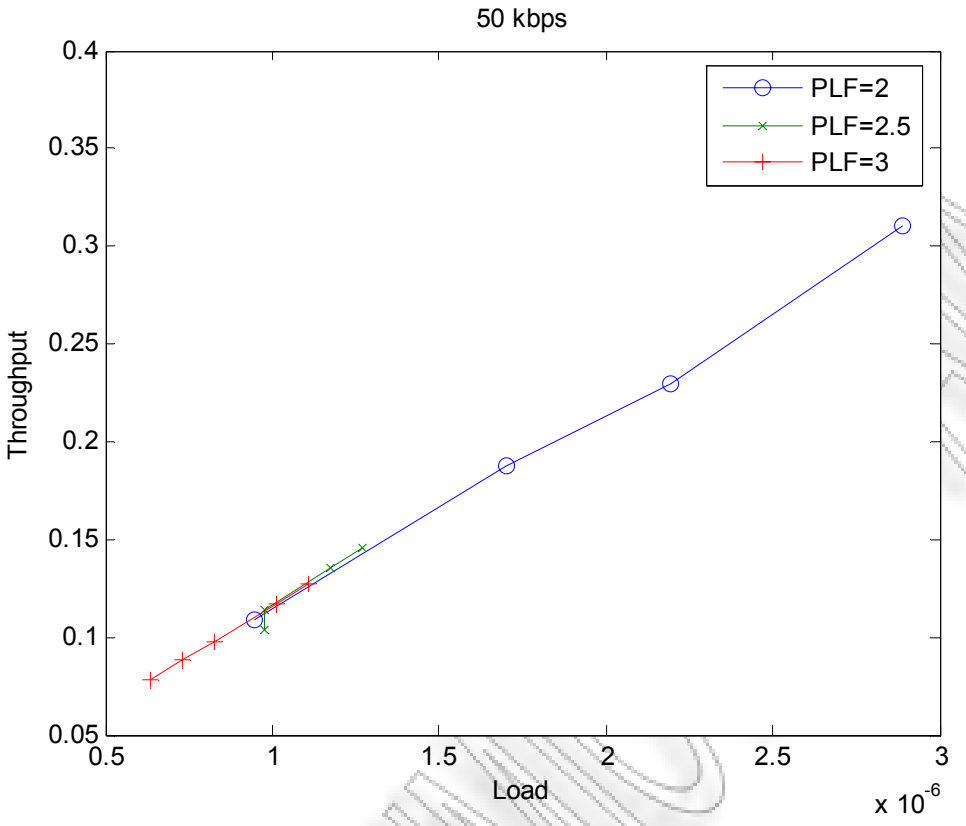
$$Energy \quad Consumed / Time = \frac{(Initial \quad Energy - Energy \quad Left)\cdot 10^6}{EOS \quad Time}$$

- Ενέργεια που καταναλώθηκε προς bit που μεταδόθηκε με επιτυχία (J / bit)

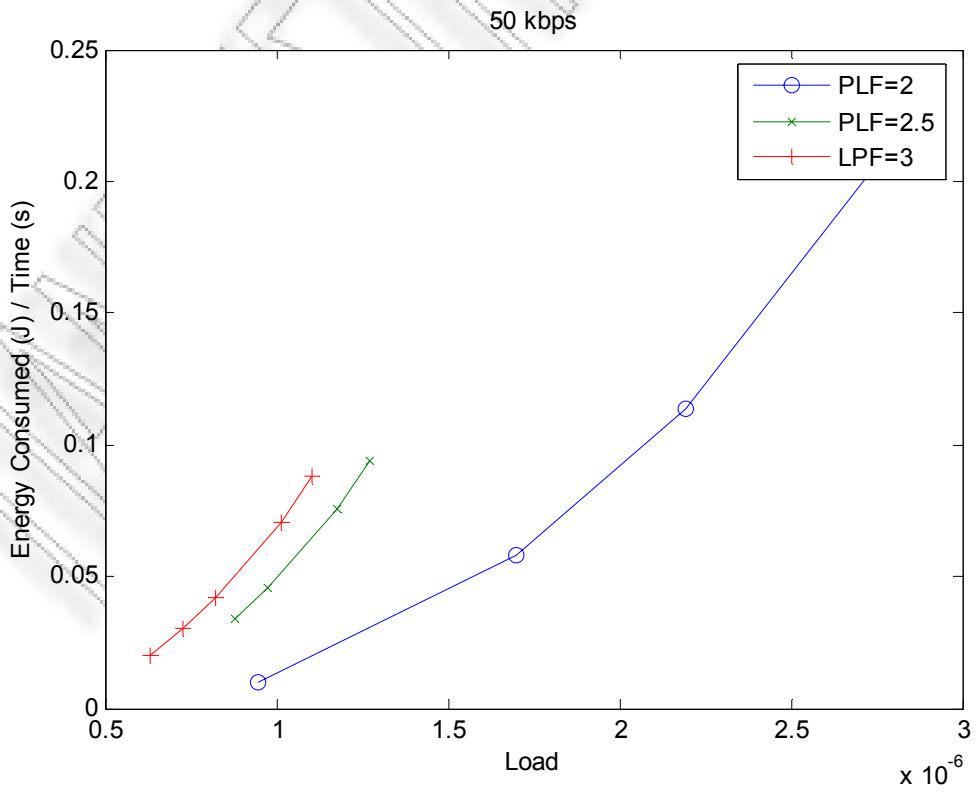
$$Energy \quad Consumed / Succes.Trans.Bit = \frac{(Initial \quad Energy - Energy \quad Left)\cdot 10^6}{(Messages \quad \text{Re}ceived)\cdot(Packet \quad Size)}$$

Τα σενάρια έχουν υλοποιηθεί για διαφορετικό ρυθμό μετάδοσης: 50kbps, 100kbps, 250kbps, 500kbps και ρυθμίζοντας κάθε φορά το path loss factor - PLF[3] (σχήματα 4.8 – 4.19). Στη συνέχεια κρατώντας σταθερό το PLF συγκρίνουμε τους διάφορους ρυθμούς μετάδοσης (σχήματα 4.20 – 4.22). Στα σχήματα 4.23 και 4.24 παρουσιάζονται δύο συγκριτικές γραφικές παραστάσεις με τα αποτελέσματα που προκύπτουν χρησιμοποιώντας την υλοποίηση του πρωτοκόλλου 802.15.4 και χωρίς τη χρήση αυτού. Η σύγκριση έγινε για ρυθμό μετάδοσης 250 kbps (που προτείνει το πρωτόκολλο) και για περιβάλλον με PLF ίσο με 2. Τα αποτελέσματα των προσομοιώσεων παρουσιάζονται στις παρακάτω γραφικές παραστάσεις.

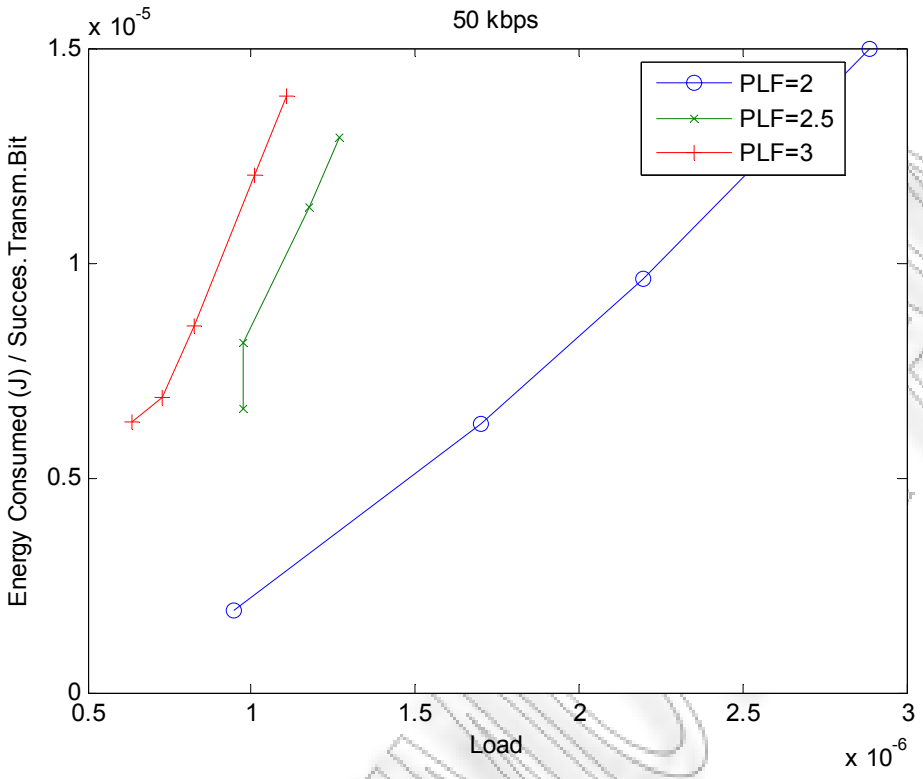
[3] Εξαρτάται από το περιβάλλον μετάδοσης.



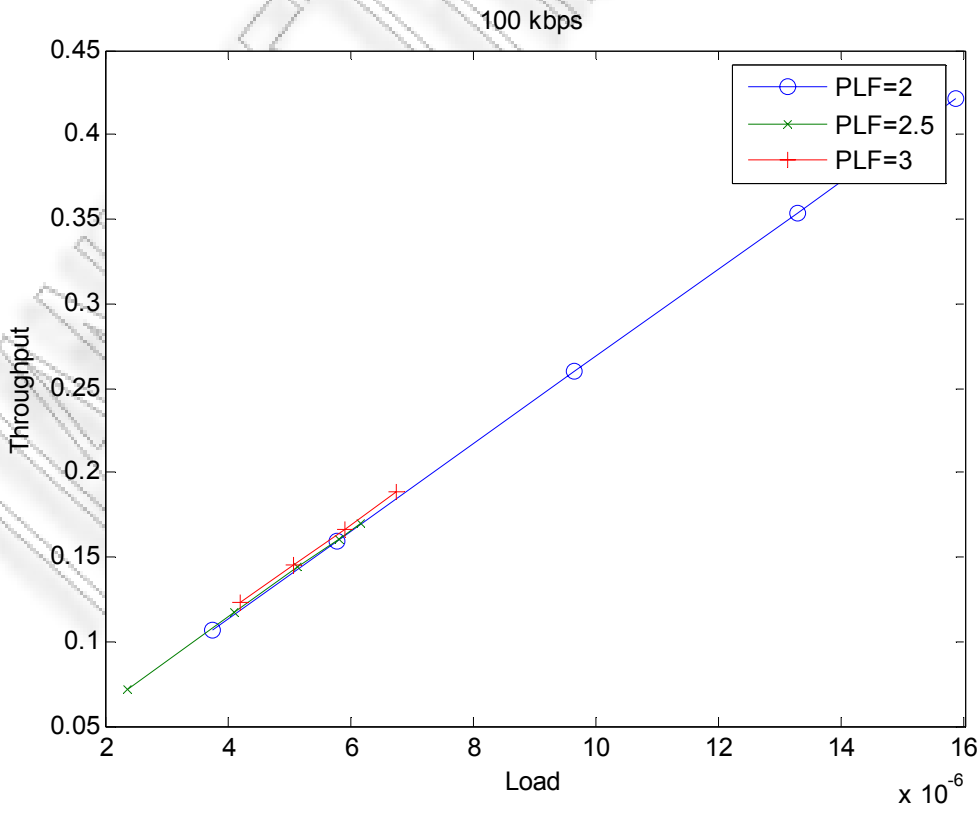
Σχήμα 4.8 Throughput vs. Load, Ρυθμός μετάδοσης: 50kbps



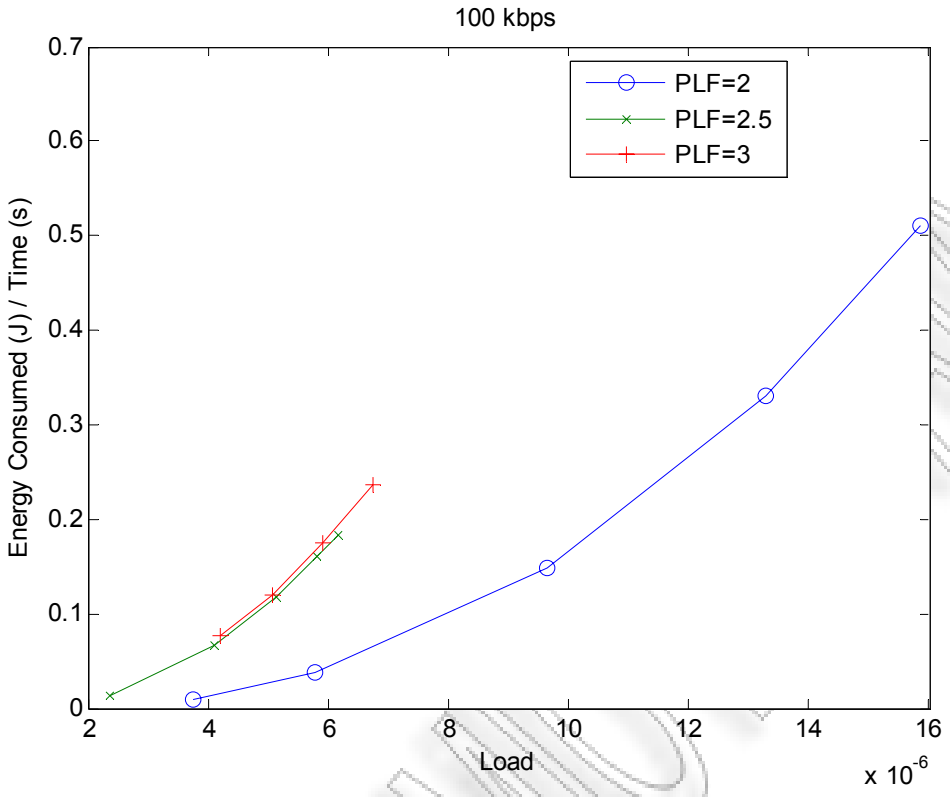
Σχήμα 4.9 (Energy Consumed / Time) vs. Load, Ρυθμός μετάδοσης: 50kbps



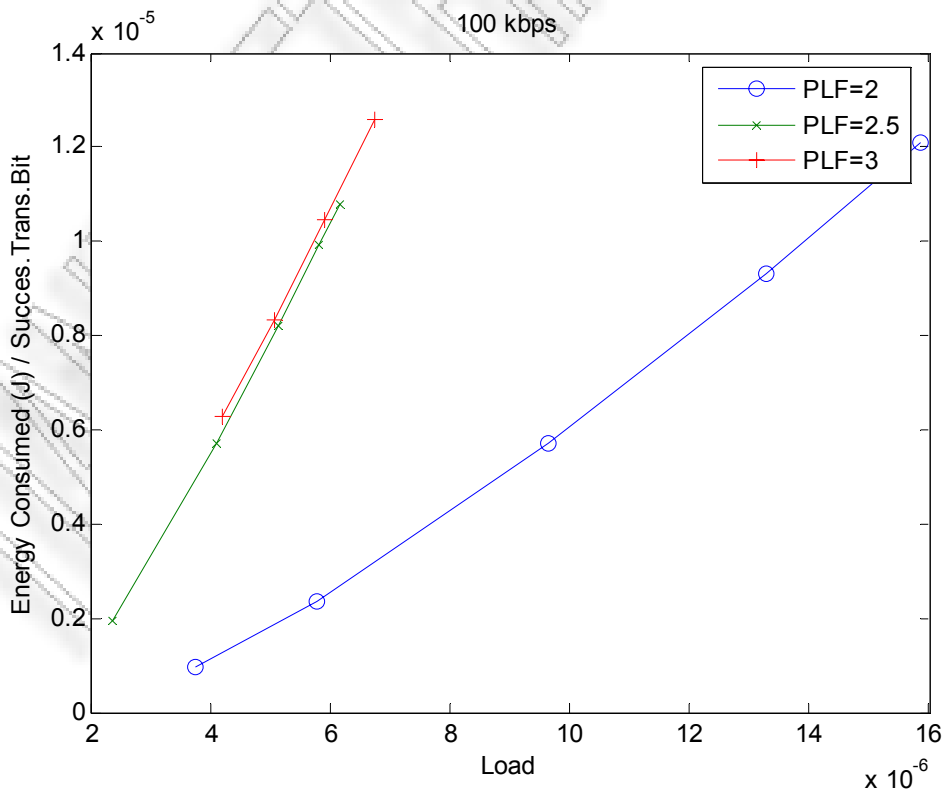
Σχήμα 4.10 (Energy Consumed / Bit) vs. Load, Ρυθμός μετάδοσης: 50kbps



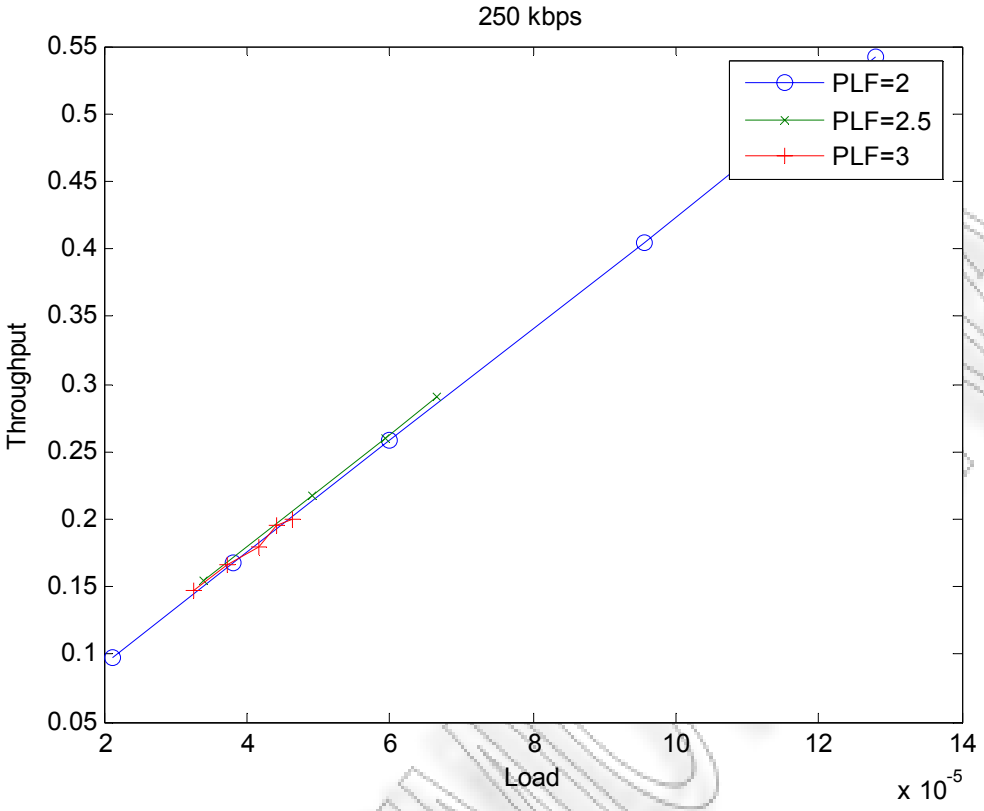
Σχήμα 4.11 Throughput vs. Load, Ρυθμός μετάδοσης: 100kbps



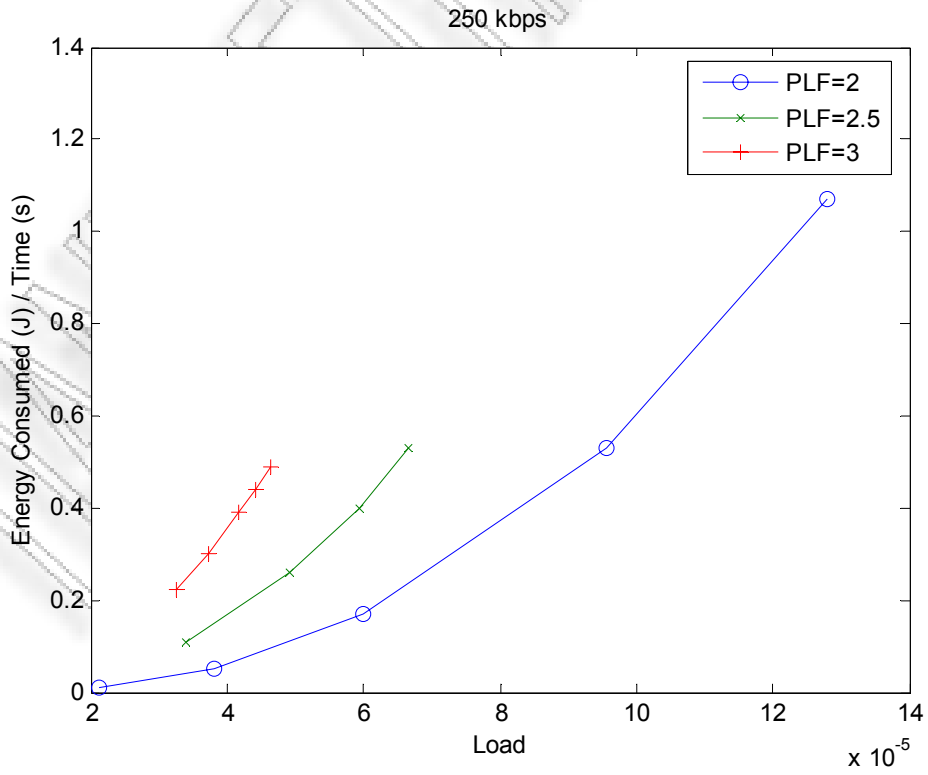
Σχήμα 4.12 (Energy Consumed / Time) vs.Load, Ρυθμός μετάδοσης: 100kbps



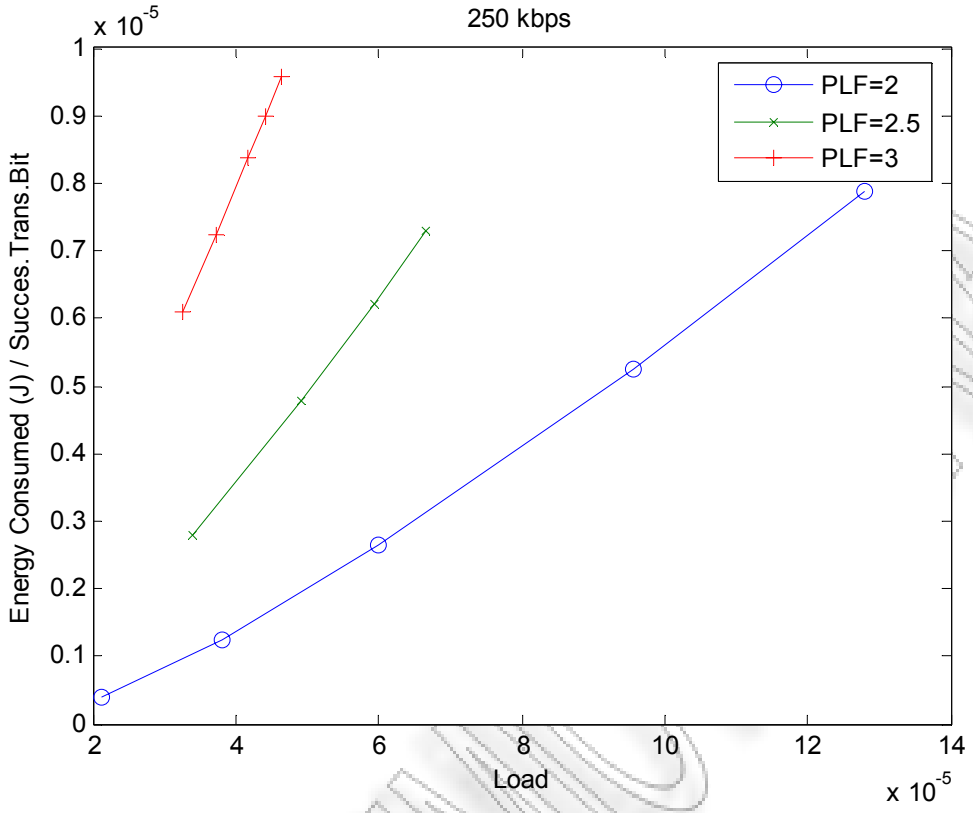
Σχήμα 4.13 (Energy Consumed / Bit) vs. Load, Ρυθμός μετάδοσης: 100kbps



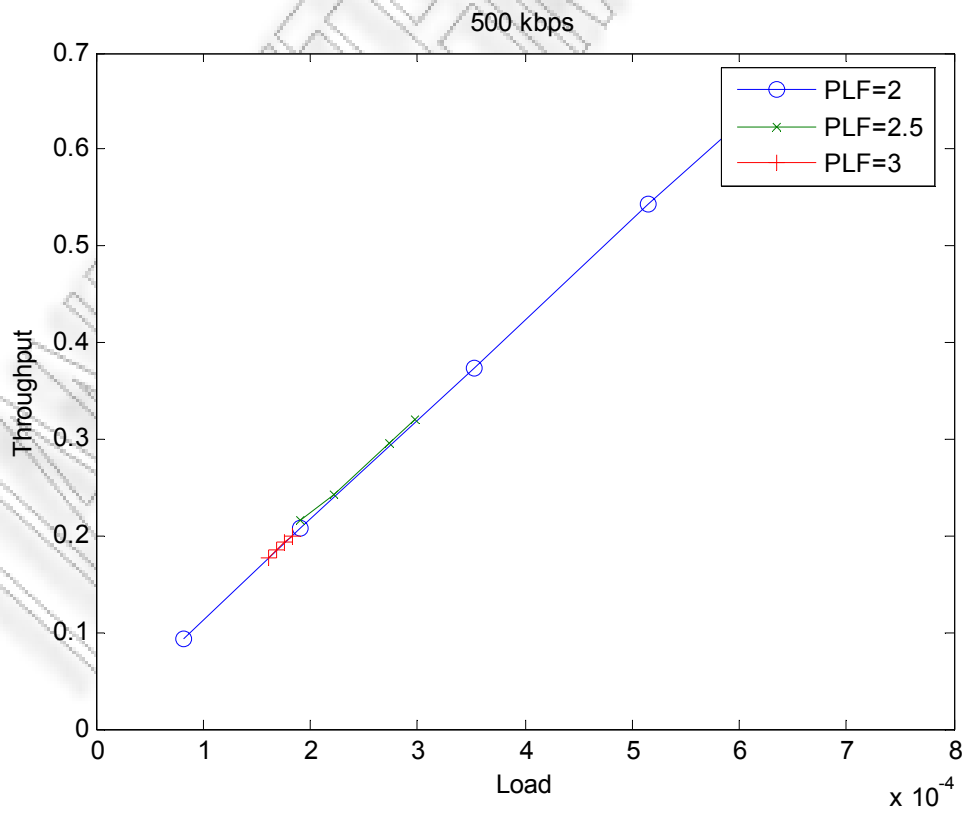
Σχήμα 4.14 Throughput vs. Load, Ρυθμός μετάδοσης: 250kbps



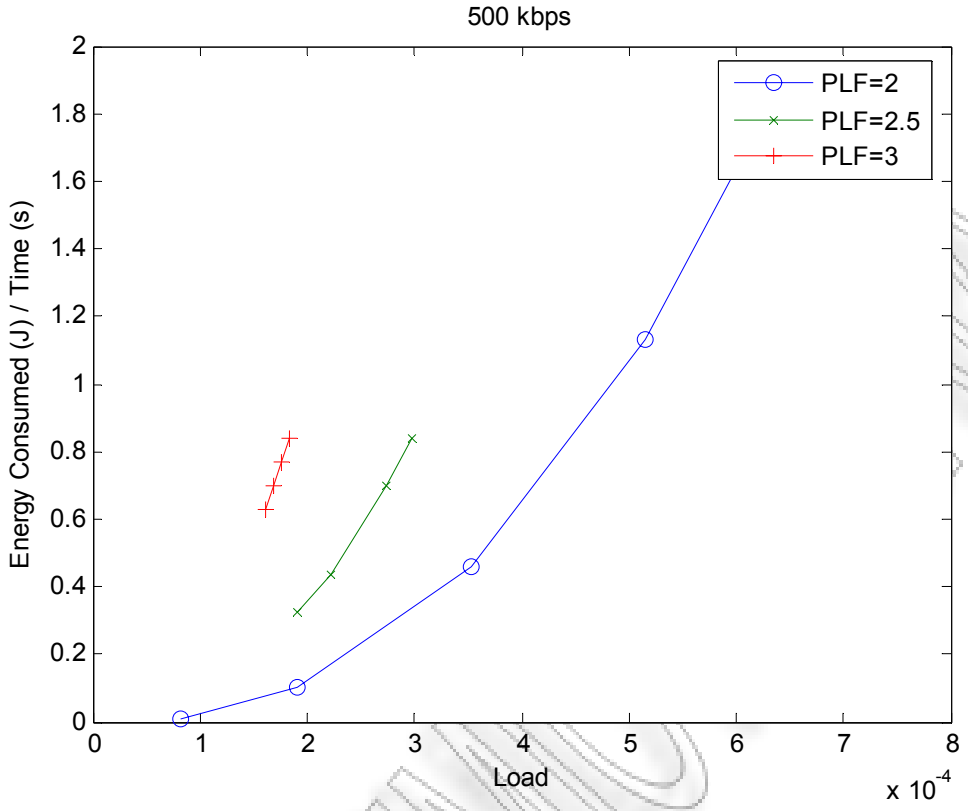
Σχήμα 4.15 (Energy Consumed / Time) vs. Load, Ρυθμός μετάδοσης: 250kbps



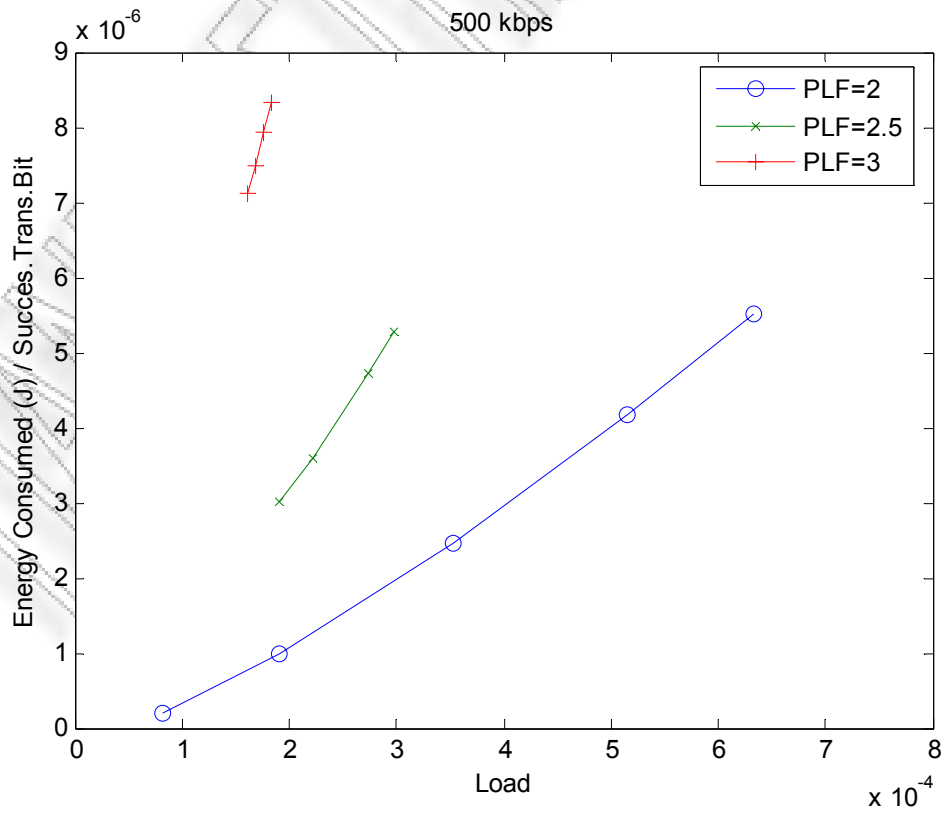
Σχήμα 4.16 (Energy Consumed / Bit) vs. Load, Ρυθμός μετάδοσης: 250kbps



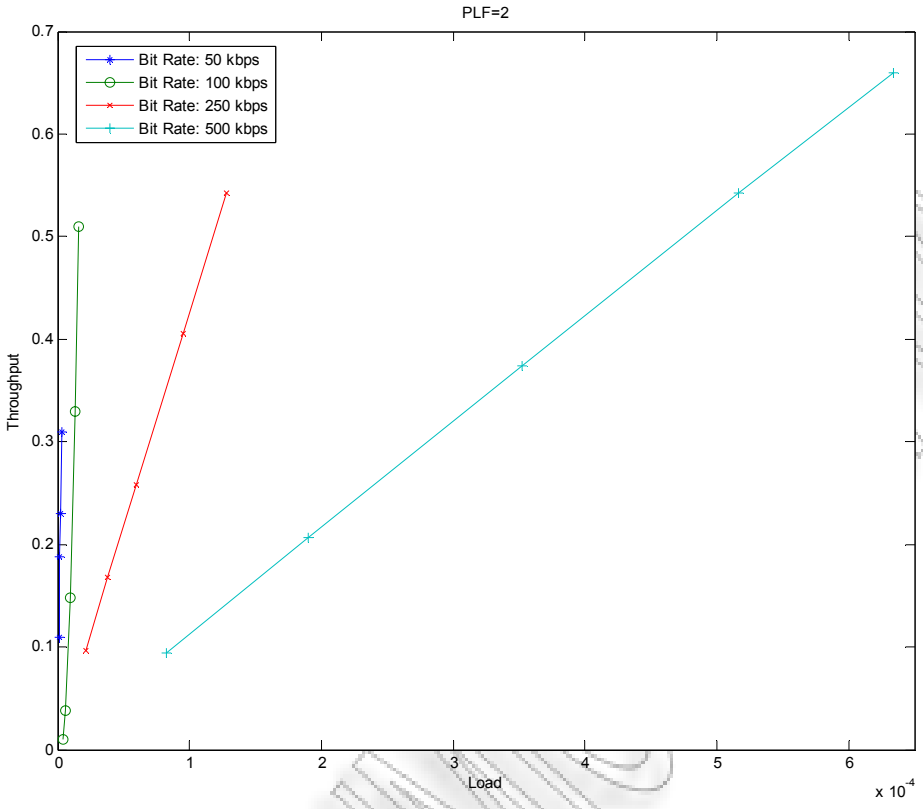
Σχήμα 4.17 Throughput vs. Load, Ρυθμός μετάδοσης: 500kbps



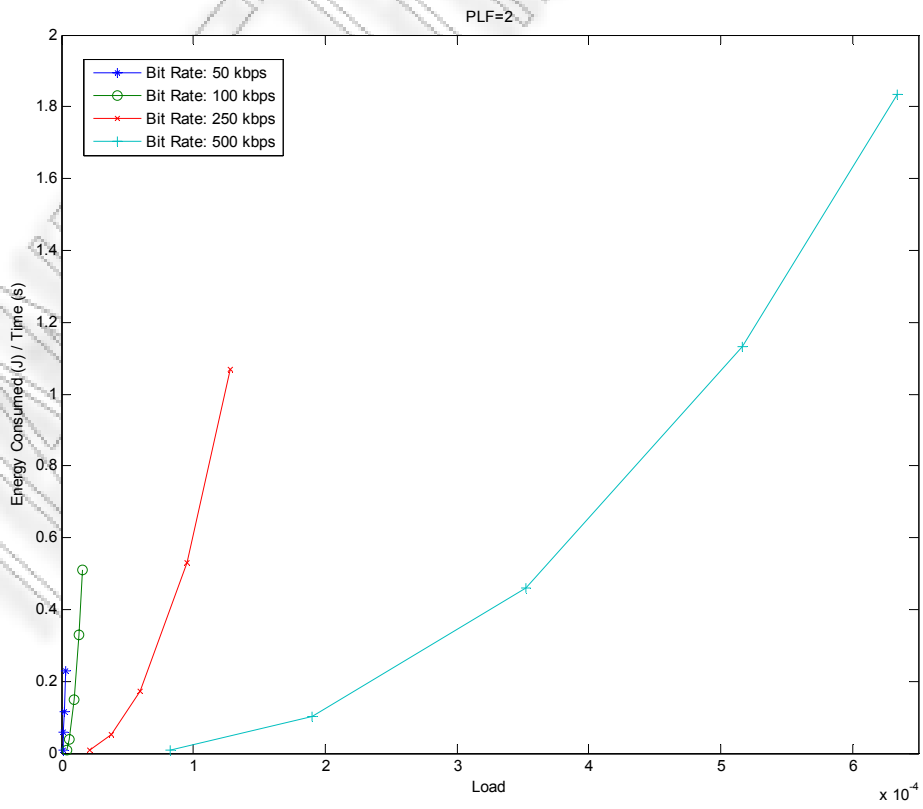
Σχήμα 4.18 (Energy Consumed / Time) vs. Load, Ρυθμός μετάδοσης: 500kbps



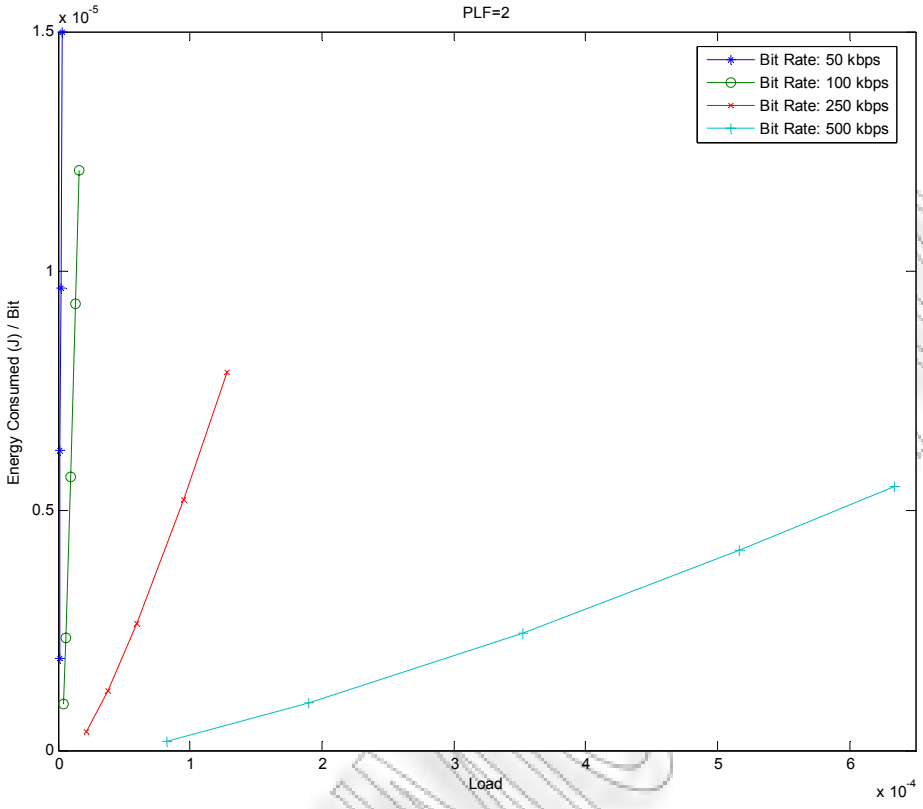
Σχήμα 4.19 (Energy Consumed / Bit) vs. Load, Ρυθμός μετάδοσης: 500kbps



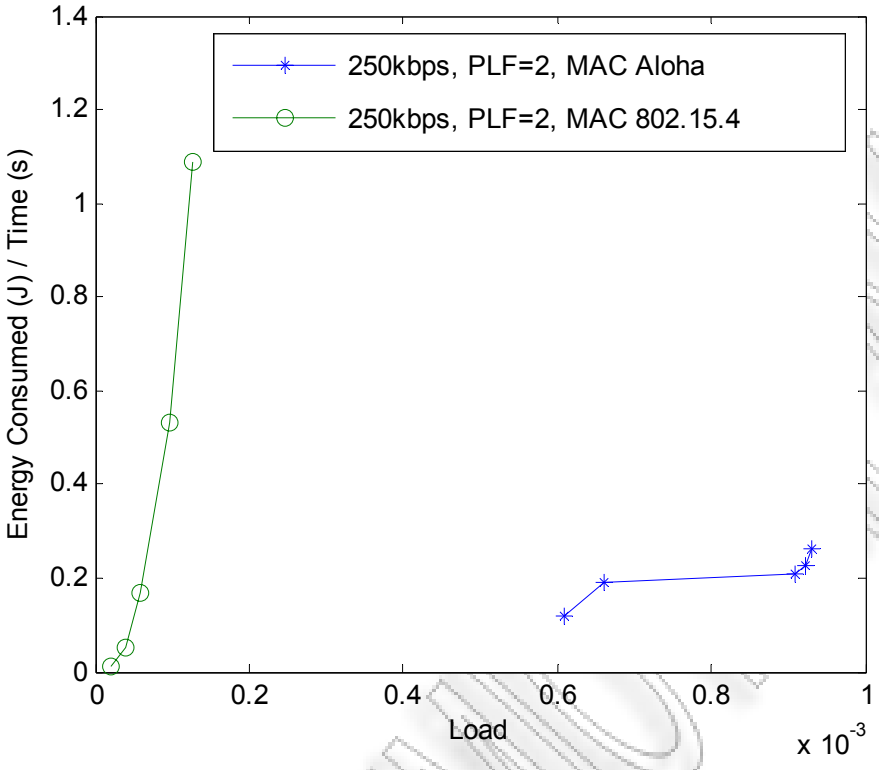
Σχήμα 4.20 Throughput vs. Load, PLF=2



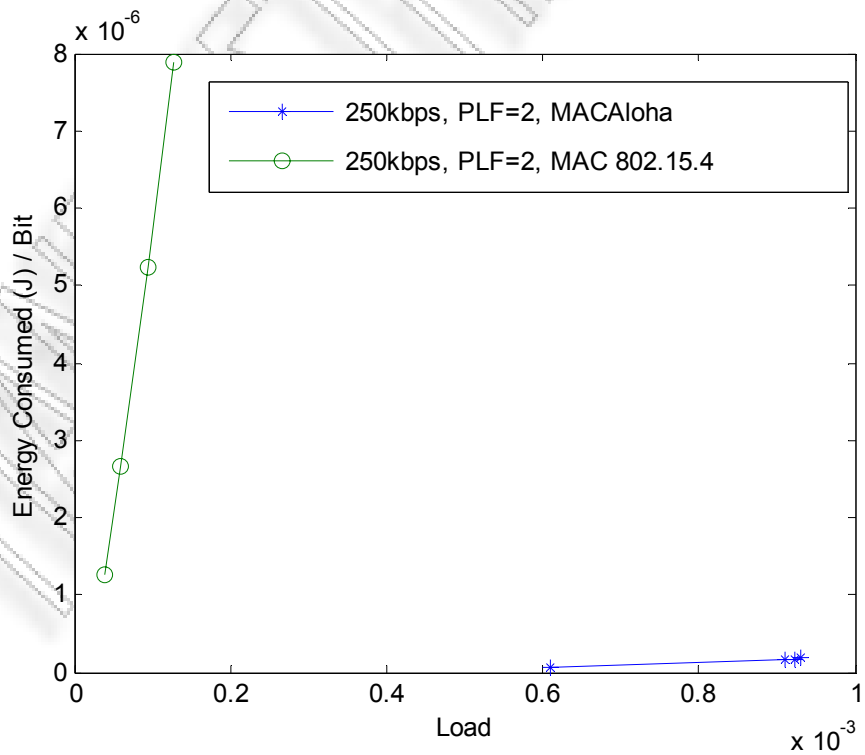
Σχήμα 4.21 (Energy Consumed / Time) vs. Load, PLF=2



<u>Σχήμα 4.22</u> (Energy Consumed / Bit) vs. Load, PLF=2



Σχήμα 4.23 (Energy Consumed / Time) vs. Load, Ρυθμός μετάδοσης: 250kbps. Σύγκριση αποτελεσμάτων με και χωρίς τη χρήση του πρωτοκόλλου 802.15.4



Σχήμα 4.24 (Energy Consumed / Bit) vs. Load, Ρυθμός μετάδοσης: 250kbps. Σύγκριση αποτελεσμάτων με και χωρίς τη χρήση του πρωτοκόλλου 802.15.4

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Το βασικό πρόβλημα με τα δίκτυα ασύρματων αισθητήρων παρουσιάζεται όταν τροφοδοτούνται με μπαταρίες και δεν έχουν σταθερή πηγή ενέργειας. Κύριο μέλημά μας είναι να βρούμε όλες τις κατάλληλες μεθόδους ώστε να επιμηκύνουμε τη διάρκεια ζωής των κόμβων, αλλά χωρίς να ελαττώσουμε την επιτυχημένη μετάδοση των δεδομένων που στέλνονται. Για αυτό το λόγο παρουσιάσαμε αρχικά τις λειτουργίες του πρωτοκόλλου 802.15.4, καθώς και τα βασικά πλεονεκτήματα που προκύπτουν από αυτές (κεφάλαιο 2): Μηχανισμός CSMA-CA, τοπολογίες αστέρα και peer-to-peer, ενεργή / μη ενεργή μετάδοση beacons, διαμορφώσεις δέντρων peer-to-peer και συστάδων, ενεργειακή ανίχνευση, ανίχνευση ποιότητας ζεύξης, σάρωση καναλιού, επιβεβαίωση αποστολής, ασφάλεια. Ακόμη έγινε λεπτομερής ανάλυση του προσομοιωτή ασυρμάτων δικτύων αισθητήρων Sensasim (κεφάλαιο 1), ενώ παρουσιάστηκε και το εγχειρίδιο χρήσης του προγράμματος (κεφάλαιο 4).

Χρησιμοποιήσαμε το πρόγραμμα NS-2 για τον έλεγχο της ενέργειας που καταναλώνεται μέσα από επιμέρους ρυθμίσεις του επιπέδου MAC του πρωτοκόλλου (κεφάλαιο 3). Με αυτόν τον τρόπο προσπαθήσαμε να διαπιστώσουμε, μεταβάλλοντας τις τιμές των μεταβλητών ‘beacon order’ και ‘Superframe order’ που ορίζονται από το πρωτόκολλο 802.15.4, για ποιες τιμές της μέσης καθυστέρησης μπορεί να μειωθεί το duty cycle που έχει ως αποτέλεσμα τη μείωση της κατανάλωσης ενέργειας. Το αποτέλεσμα ήταν ότι για τιμές των δύο μεταβλητών ίσες με BO=SO=14 να έχουμε σταθερή καθυστέρηση (σχήμα 3.4) αλλά να πετυχαίνουμε πιο σταθερό throughput για BO=SO=15 (σχήμα 3.3), δηλαδή μη χρησιμοποιώντας σήματα beacons.

Αποδείξαμε ακόμη, ότι το δίκτυο έχει καλύτερη προσαρμογή στο περιβάλλον που εφαρμόζεται, όταν ακολουθείται η τοπολογία peer-to-peer. Παρόλαυτα παρατηρήσαμε ότι χρησιμοποιώντας το πρωτόκολλο 802.15.4 (τοπολογία αστέρα με ένα κεντρικό διαχειριστή της επικοινωνίας και το δίκτυο χωρισμένο σε PANs), πετυχαίνουμε με μεγαλύτερη επιτυχία τη σωστή παράδοση-παραλαβή των διαφόρων δεδομένων (σχήματα 4.8, 4.11, 4.14) με κόστος όμως την μεγαλύτερη

κατανάλωση ενέργειας σε σχέση με το αν δεν χρησιμοποιούσαμε το πρωτόκολλο (σχήματα 4.23, 4.24) . Όπως παρατηρούμε στη γραφική 4.23 η ενέργεια που καταναλώνεται ανά μονάδα χρόνου αυξάνει πάρα πολύ με την αύξηση του προσφερόμενου φορτίου. Μεταβολή των παραπάνω τιμών παρατηρείται και στην υλοποίηση χωρίς τη χρήση του πρωτοκόλλου, αλλά ο ρυθμός αύξησης είναι πολύ μικρός. Ακόμα παρατηρούμε, ότι με τη χρήση του πρωτοκόλλου, το throughput αυξάνει, με σχεδόν σταθερό ρυθμό, ενώ οι τιμές που παίρνει είναι πολύ ικανοποιητικές. Με αυτόν τον τρόπο παρατηρούμε ότι στην πρώτη περίπτωση (χρήση πρωτοκόλλου) και σε εφαρμογές που απαιτούνται συνεχείς λήψεις μετρήσεων από τους αισθητήρες που λειτουργούν με μπαταρία, όπως π.χ. σε γεωργικές εκτάσεις, το πρωτόκολλο είναι ασύμφορο ενεργειακά. Σε άλλες περιπτώσεις όπου υπάρχει σταθερή ενεργειακή πηγή των κόμβων με αισθητήρα προτείνεται η χρήση του, καθώς προσφέρει μεγαλύτερη ασφάλεια στη σωστή μετάδοση των δεδομένων.

Ένα ακόμα συμπέρασμα σχετίζεται με το συντελεστή path loss. Αυξάνοντας το συντελεστή προκύπτει το συμπέρασμα ότι η ενέργεια που καταναλώνεται ανά δευτερόλεπτο λειτουργίας ή ανά μεταδιδόμενο bit, αυξάνει κατά πολύ (σχήματα 4.9, 4.10, 4.12, 4.13, 4.15, 4.16, 4.18, 4.19) . Τέλος, το throughput για ρυθμό μετάδοσης 250 kbps είναι κατά πολύ μεγαλύτερο από αυτό για ρυθμό μετάδοσης 500 kbps και για το ίδιο φορτίο. Κρατώντας σταθερό το συντελεστή path loss (PLF=2) συγκρίναμε για διάφορα φορτία τα γραφήματα που προκύπτουν για διαφορετικούς ρυθμούς μετάδοσης (σχήματα 4.20, 4.21, 4.22). Για μεγαλύτερο συντελεστή Path Loss η μέτρηση ήταν αδύνατη, καθώς η διάρκεια που διαρκεί κάθε προσομοίωση, για κάθε ρυθμό μετάδοσης. δημιουργεί διαφορετικής τάξης φορτίου, επιτρέποντας την άμεση σύγκριση μεταξύ των bit rates μόνο για PLF ίσο με 2. Σε αυτές τις συγκρίσεις η διάρκεια κάθε πακέτου που στέλνεται είναι διαφορετική, καθώς για μικρότερους ρυθμούς μετάδοσης η διάρκεια αυξάνει ενώ για μεγαλύτερους ρυθμούς η διάρκεια ελαττώνεται. Τελικά διαπιστώνουμε ότι καθώς το φορτίο αυξάνει έχουμε και ταυτόχρονη αύξηση της κατανάλωσης ενέργειας για ρυθμό 500 kbps (σχήμα 4.21).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] IEEE Standards, 802.15.4 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR - WPANs), 2003.

[2] "The *ns* Manual", Kevin Fall, Kannan Varadhan, 2005

[3] "Wireless Sensor Network Designs", Anna Hac, John Wiley & Sons Ltd, 2003.

[4] "802.15.4 MAC PHY Software Reference Manual", Freescale Semiconductor, 2005.

[5] "802.15.4 MAC PHY Software User's Guide", Freescale Semiconductor, 2005.

[6] "Java – based simulator for wireless multihop networks using directional antennas", Ghadi Rayess, Ioannis Tsirilakis, Antonis Kalis, Athens Information Technology.

[7] "User Manual" – Sensasim, Athens Information Technology.

[8] "802.15.4 and ZigBee Routing Simulation", Samsung/CUNY.

[9] "Σχεδιασμός και Υλοποίηση Πρωτοκόλλου Διασύνδεσης Δεδομένων (Data Link Layer) για Ασύρματα Δίκτυα Αισθητήρων", Ασβεστά Αργυρώ, ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, 2005

[10] "Home Networking with IEEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks", Ed Callaway, Paul Corday, Lance Hester et al., IEEE Communications Magazine August 2002.

[11] “IEEE 802.15.4: a wireless communication technology for large-scale ubiquitous computing applications”, Anis Koubâa, Mario Alves, Eduardo Tovar, 2006.

[12] “Technical Report: IEEE 802.15.4: a Federating Communication Protocol for Time-Sensitive Wireless Sensor Network”, Anis Koubâa, Mário Alves, Eduardo Tovar, 2006

[13] “Sensor Networks”, Thomas Haenselmann, 2006.

[14] “J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks”, Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, 2004

[15] “An Event-Driven Wireless MAC Protocol Simulator“, George R.J. Linnenbank, Paul J.M. Havinga, University of Twente, Department of Computer Science.

[16] “JavaSim, User’s Guide, Public Release 0.3, Version 1.0”, Object-Oriented Discrete-Event Simulation in Java, Department of Computing Science, Computing Laboratory, 1999.

[17] Marc Greis' Tutorial for the UCB/LBNL/VINT Network Simulator "ns".

[18] “Delay Efficient Sleep Scheduling in Wireless Sensor Networks”, Gang Lu, Narayanan Sadagopan†, Bhaskar Krishnamachari_†, Ashish Goel.

[19] “An Energy-Efficient MAC Protocol for Wireless Sensor Networks”, Wei Ye, John Heidemann, Deborah Estrin.

[20] “Energy Efficient & Delay Optimized MAC for Wireless Sensor Networks”, Changsu Suh, Young-Mi Song and Young-Bae Ko, We Duke Cho.

[21] "Configuring the IEEE 802.15.4 MAC Layer for Single-sinkWireless Sensor Network Applications", Joe Hoffert, Kevin Klues, Obi Orjih, Washington University in St. Louis.

ΑΝΑΦΟΡΕΣ ΣΤΟ INTERNET:

[1] "Low Rate Wireless Personal Area Networks", NS2 Simulation Platform. http://ees2cy.engr.ccny.cuny.edu/zheng/pub/

[2] "Advanced Network Technologies Division", National Institute of Standards and Technology. http://w3.antd.nist.gov/net_pc.shtml

[3] "Freescale Semiconductors". http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC13193&nodeId=01752003088166

[4] "Zigbee Aliance". http://www.zigbee.org/en/

[5] "IEEE 802.15 TUTORIALS", http://grouper.ieee.org/groups/802/15/pub/Tutorials.html

[6] "Linux Wireless Sensor LAN Project", http://linux-802-15-4.sourceforge.net/

[7] "A Guide For the Clueless: IEEE 802.15.4 Standard for Low-Rate Wireless Personal Area Networks (LR-WPAN)", http://lecs.cs.ucla.edu/~adparker/EE202A/hw2/index.html

[8] "Tutorial On Event Driven Simulations of Network Protocols", http://www-unix.ecs.umass.edu/~ece671/simulation/sim_instructions.html

[9] "Low Rate Wireless Personal Area Networks (LR-WPANs)", http://ees2cy.engr.ccny.cuny.edu/zheng/pub/

[10] "ALOHA PROTOCOL", http://www.laynetworks.com/ALOHA%20PROTOCOL.htm

ΠΑΡΑΡΤΗΜΑ Α

Α.1 Κώδικας Προσομοίωσης για το Πρόγραμμα NS-2

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# Channel Type
set  val(prop)              Propagation/TwoRayGround    ;# radio-propagation
model
set val(netif)          Phy/WirelessPhy/802_15_4
set val(mac)            Mac/802_15_4
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             11                          ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)          50
set val(y)          50

set val(nam)        wpan_demo3.nam
set val(traffic)    ftp                         ;# cbr/poisson/ftp

#read command line arguments
proc getCmdArgu {argc argv} {
        global val
        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
                set name [string range $arg 1 end]
                set val($name) [lindex $argv [expr $i+1]]
        }
}
getCmdArgu $argc $argv

set appTime1              8.3    ;# in seconds
set appTime2              8.6    ;# in seconds
set stopTime            100      ;# in seconds
```

```
# Initialize Global Variables
set ns_             [new Simulator]
set tracefd     [open ./wpan_demo3.tr w]
$ns_ trace-all $tracefd
if { "$val(nam)" == "wpan_demo3.nam" } {
        set namtrace     [open ./$val(nam) w]
        $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
}

$ns_ puts-nam-traceall {# nam4wpan #}          ;# inform nam that this is a
trace file for wpan (special handling needed)

Mac/802_15_4 wpanCmd verbose on
Mac/802_15_4 wpanNam namStatus on       ;# default = off (should be turned
on before other 'wpanNam' commands can work)
#Mac/802_15_4 wpanNam ColFlashClr gold        ;# default = gold

# For model 'TwoRayGround'
set dist(5m)  7.69113e-06
set dist(9m)  2.37381e-06
set dist(10m) 1.92278e-06
set dist(11m) 1.58908e-06
set dist(12m) 1.33527e-06
set dist(13m) 1.13774e-06
set dist(14m) 9.81011e-07
set dist(15m) 8.54570e-07
set dist(16m) 7.51087e-07
set dist(20m) 4.80696e-07
set dist(25m) 3.07645e-07
set dist(30m) 2.13643e-07
set dist(35m) 1.56962e-07
set dist(40m) 1.20174e-07
Phy/WirelessPhy set CSThresh_ $dist(15m)
Phy/WirelessPhy set RXThresh_ $dist(15m)

# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create God
set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]
```

```
# configure node

$ns_ node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
             -ifqType $val(ifq) \
             -ifqLen $val(ifqlen) \
             -antType $val(ant) \
             -propType $val(prop) \
             -phyType $val(netif) \
             -topoInstance $topo \
             -agentTrace OFF \
             -routerTrace OFF \
             -macTrace ON \
             -movementTrace OFF \
               #-energyModel "EnergyModel"\
               #-initialEnergy 1\
               #-rxPower 0.3\
               #-txPower 0.3\
             -channel $chan_1_


for {set i 0} {$i < $val(nn) } {incr i} {
       set node_($i) [$ns_ node]
       $node_($i) random-motion 0       ;# disable random motion
}

source ./wpan_demo3.scn

$ns_ at 0.0  "$node_(0) NodeLabel PAN Coor"
$ns_ at 0.0  "$node_(0) sscs startPANCoord 1"        ;#         startPANCoord
<txBeacon=1> <BO=3> <SO=3>
$ns_ at 0.5  "$node_(1) sscs startDevice 1 1 1"            ;#    startDevice
<isFFD=1> <assoPermit=1> <txBeacon=0> <BO=3> <SO=3>
$ns_ at 1.5  "$node_(2) sscs startDevice 1 1 1"
$ns_ at 2.5  "$node_(3) sscs startDevice 1 1 1"
$ns_ at 3.5  "$node_(4) sscs startDevice 1 1 1"
$ns_ at 4.5  "$node_(5) sscs startDevice 1 1 1"
$ns_ at 5.5  "$node_(6) sscs startDevice 0"
$ns_ at 5.8  "$node_(7) sscs startDevice 0"
$ns_ at 6.5  "$node_(8) sscs startDevice 0"
$ns_ at 6.8  "$node_(9) sscs startDevice 0"
$ns_ at 7.0  "$node_(10) sscs startDevice 0"

$ns_ at 6.0 "$node_(3) sscs stopBeacon"
```

```
$ns_ at 8.0 "$node_(3) sscs startBeacon"
$ns_ at 9.0 "$node_(5) sscs startBeacon 4 4"         ;# change beacon order
and superframe order
$ns_ at 10.0 "$node_(4) sscs stopBeacon"

Mac/802_15_4 wpanNam PlaybackRate 3ms

$ns_ at $appTime1 "puts \"\nTransmitting data ...\n\""

# Setup traffic flow between nodes

proc cbrtraffic { src dst interval starttime } {
   global ns_ node_
   set udp_($src) [new Agent/UDP]
   eval $ns_ attach-agent \$node_($src) \$udp_($src)
   set null_($dst) [new Agent/Null]
   eval $ns_ attach-agent \$node_($dst) \$null_($dst)
   set cbr_($src) [new Application/Traffic/CBR]
   eval \$cbr_($src) set packetSize_ 70
   eval \$cbr_($src) set interval_ $interval
   eval \$cbr_($src) set random_ 0
   #eval \$cbr_($src) set maxpkts_ 10000
   eval \$cbr_($src) attach-agent \$udp_($src)
   eval $ns_ connect \$udp_($src) \$null_($dst)
   $ns_ at $starttime "$cbr_($src) start"
}

proc poissontraffic { src dst interval starttime } {
   global ns_ node_
   set udp($src) [new Agent/UDP]
   eval $ns_ attach-agent \$node_($src) \$udp($src)
   set null($dst) [new Agent/Null]
   eval $ns_ attach-agent \$node_($dst) \$null($dst)
   set expl($src) [new Application/Traffic/Exponential]
   eval \$expl($src) set packetSize_ 70
   eval \$expl($src) set burst_time_ 0
   eval \$expl($src) set idle_time_ [expr $interval*1000.0-70.0*8/250]ms ;#
idle_time + pkt_tx_time = interval
   eval \$expl($src) set rate_ 250k
   eval \$expl($src) attach-agent \$udp($src)
   eval $ns_ connect \$udp($src) \$null($dst)
   $ns_ at $starttime "$expl($src) start"
}

if { ("$val(traffic)" == "cbr") || ("$val(traffic)" == "poisson") } {
```

```
   puts "\nTraffic: $val(traffic)"
   #Mac/802_15_4 wpanCmd ack4data on
   puts [format "Acknowledgement for data: %s" [Mac/802_15_4 wpanCmd
ack4data]]
   $ns_ at $appTime1 "Mac/802_15_4 wpanNam PlaybackRate 0.50ms"
   $ns_ at [expr $appTime1 + 0.5] "Mac/802_15_4 wpanNam PlaybackRate 1.5ms"
   $val(traffic)traffic 1 6 0.2 $appTime1
   $val(traffic)traffic 4 10 0.2 $appTime2
   $ns_ at $appTime1 "$node_(1) add-mark m1 blue circle"
   $ns_ at $appTime1 "$node_(6) add-mark m2 blue circle"
   $ns_ at $appTime1 "$ns_ trace-annotate \"(at $appTime1) $val(traffic)
traffic from node 1 to node 6\""
   $ns_ at $appTime2 "$node_(4) add-mark m3 green4 circle"
   $ns_ at $appTime2 "$node_(10) add-mark m4 green4 circle"
   $ns_ at $appTime2 "$ns_ trace-annotate \"(at $appTime2) $val(traffic)
traffic from node 4 to node 10\""
   Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
   Mac/802_15_4 wpanNam FlowClr -p ARP -c green
   Mac/802_15_4 wpanNam FlowClr -p MAC -c navy
   if { "$val(traffic)" == "cbr" } {
       set pktType cbr
   } else {
       set pktType exp
   }
   Mac/802_15_4 wpanNam FlowClr -p $pktType -s 1 -d 6 -c blue
   Mac/802_15_4 wpanNam FlowClr -p $pktType -s 4 -d 10 -c green4
}

proc ftptraffic { src dst starttime } {
   global ns_ node_
   set tcp($src) [new Agent/TCP]
   eval \$tcp($src) set packetSize_ 50
   set sink($dst) [new Agent/TCPSink]
   eval $ns_ attach-agent \$node_($src) \$tcp($src)
   eval $ns_ attach-agent \$node_($dst) \$sink($dst)
   eval $ns_ connect \$tcp($src) \$sink($dst)
   set ftp($src) [new Application/FTP]
   eval \$ftp($src) attach-agent \$tcp($src)
   $ns_ at $starttime "$ftp($src) start"
}

if { "$val(traffic)" == "ftp" } {
   puts "\nTraffic: ftp"
   #Mac/802_15_4 wpanCmd ack4data off
```

```
   puts [format "Acknowledgement for data: %s" [Mac/802_15_4 wpanCmd
ack4data]]
   $ns_ at $appTime1 "Mac/802_15_4 wpanNam PlaybackRate 0.17ms"
   $ns_ at [expr $appTime1 + 0.5] "Mac/802_15_4 wpanNam PlaybackRate 1.5ms"
   ftptraffic 1 6 $appTime1
   ftptraffic 4 10 $appTime2
   $ns_ at $appTime1 "$node_(1) add-mark m1 blue circle"
   #$ns_ at $stopTime "$node_(1) delete-mark m1"
   $ns_ at $appTime1 "$node_(6) add-mark m2 blue circle"
   $ns_ at $appTime1 "$ns_ trace-annotate \"(at $appTime1) ftp traffic from
node 1 to node 6\""
   $ns_ at $appTime2 "$node_(4) add-mark m3 green4 circle"
   $ns_ at $appTime2 "$node_(10) add-mark m4 green4 circle"
   $ns_ at $appTime2 "$ns_ trace-annotate \"(at $appTime2) ftp traffic from
node 4 to node 10\""
   Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
   Mac/802_15_4 wpanNam FlowClr -p ARP -c green
   Mac/802_15_4 wpanNam FlowClr -p MAC -c navy
   Mac/802_15_4 wpanNam FlowClr -p tcp -s 1 -d 6 -c blue
   Mac/802_15_4 wpanNam FlowClr -p ack -s 6 -d 1 -c blue
   Mac/802_15_4 wpanNam FlowClr -p tcp -s 4 -d 10 -c green4
   Mac/802_15_4 wpanNam FlowClr -p ack -s 10 -d 4 -c green4
}

# defines the node size in nam
for {set i 0} {$i < $val(nn)} {incr i} {
       $ns_ initial_node_pos $node_($i) 2
}

# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stopTime "$node_($i) reset";
}

$ns_ at $stopTime "stop"
$ns_ at $stopTime "puts \"\nNS EXITING...\n\""
$ns_ at $stopTime "$ns_ halt"

proc stop {} {
    global ns_ tracefd appTime val env
    $ns_ flush-trace
    close $tracefd
    set hasDISPLAY 0
    foreach index [array names env] {
        #puts "$index: $env($index)"
```

```
        if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
                set hasDISPLAY 1
        }
    }
    if { ("$val(nam)" == "wpan_demo3.nam") && ("$hasDISPLAY" == "1") } {
       exec nam wpan_demo3.nam &
    }
}

puts "\nStarting Simulation..."
$ns_ run
```

Α.2 Ακολουθία Μηνυμάτων

```
Acknowledgement for data: on

Starting Simulation...
--- startPANCoord [0] ---
[0.000000](node 0) performing active channel scan
[0.000000](node 0) scanning channel 11
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 35.9
SORTING LISTS ...DONE!
[0.140800](node 0) scanning channel 12
[0.282240](node 0) scanning channel 13
[0.421760](node 0) begin to transmit beacons
[0.422528](node 0) successfully started a new PAN (beacon enabled)
[channel:11]
[PAN_ID:0]
--- startDevice [1] ---
[0.500000](node 1) performing active channel scan ...
[0.500000](node 1) scanning channel 11
[0.763680](node 1) scanning channel 12
[1.026400](node 1) scanning channel 13
[1.289760](node 1) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:0] ...
[1.290848](node 1) sending association request command ...
[1.292384](node 1) ack for association request command received
```

```
--- startDevice [2] ---
[1.500000](node 2) performing active channel scan ...
[1.500000](node 2) scanning channel 11
[1.763040](node 2) scanning channel 12
[1.783904](node 1) sending data request command ...
[1.785184](node 1) ack for data request command received
[1.787648](node 1) association response command received
[1.787648](node 1) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:0]
[1.787648](node 1) begin to synchronize with the coordinator
[1.787648](node 1) begin to transmit beacons
[1.788416](node 1) beacon transmission successful [channel:11] [PAN_ID:0]
[2.026080](node 2) scanning channel 13
[2.289760](node 2) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:0] ...
[2.291488](node 2) sending association request command ...
[2.293024](node 2) ack for association request command received
--- startDevice [3] ---
[2.500000](node 3) performing active channel scan ...
[2.500000](node 3) scanning channel 11
[2.763040](node 3) scanning channel 12
[2.784544](node 2) sending data request command ...
[2.785824](node 2) ack for data request command received
[2.788288](node 2) association response command received
[2.788288](node 2) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:0]
[2.788288](node 2) begin to synchronize with the coordinator
[2.788288](node 2) begin to transmit beacons
[2.789056](node 2) beacon transmission successful [channel:11] [PAN_ID:0]
[3.026400](node 3) scanning channel 13
[3.288800](node 3) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:1] ...
[3.288928](node 3) sending association request command ...
[3.290592](node 3) ack for association request command received
--- startDevice [4] ---
[3.500000](node 4) performing active channel scan ...
[3.500000](node 4) scanning channel 11
[3.764320](node 4) scanning channel 12
[3.782112](node 3) sending data request command ...
[3.783392](node 3) ack for data request command received
[3.785856](node 3) association response command received
```

```
[3.785856](node 3) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:1]
[3.785856](node 3) begin to synchronize with the coordinator
[3.785856](node 3) begin to transmit beacons
[3.786624](node 3) beacon transmission successful [channel:11] [PAN_ID:0]
[4.026720](node 4) scanning channel 13
[4.291040](node 4) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:1] ...
[4.291808](node 4) sending association request command ...
[4.293472](node 4) ack for association request command received
--- startDevice [5] ---
[4.500000](node 5) performing active channel scan ...
[4.500000](node 5) scanning channel 11
[4.764320](node 5) scanning channel 12
[4.784992](node 4) sending data request command ...
[4.786272](node 4) ack for data request command received
[4.788736](node 4) association response command received
[4.788736](node 4) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:1]
[4.788736](node 4) begin to synchronize with the coordinator
[4.788736](node 4) begin to transmit beacons
[4.789504](node 4) beacon transmission successful [channel:11] [PAN_ID:0]
[5.026720](node 5) scanning channel 13
[5.289120](node 5) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:2] ...
[5.290528](node 5) sending association request command ...
[5.292192](node 5) ack for association request command received
--- startDevice [6] ---
[5.500000](node 6) performing active channel scan ...
[5.500000](node 6) scanning channel 11
[5.762080](node 6) scanning channel 12
[5.783712](node 5) sending data request command ...
[5.784992](node 5) ack for data request command received
[5.787456](node 5) association response command received
[5.787456](node 5) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:2]
[5.787456](node 5) begin to synchronize with the coordinator
[5.787456](node 5) begin to transmit beacons
[5.788224](node 5) beacon transmission successful [channel:11] [PAN_ID:0]
--- startDevice [7] ---
```

```
[5.800000](node 7) performing active channel scan ...
[5.800000](node 7) scanning channel 11
[6.025760](node 6) scanning channel 13
[6.063360](node 7) scanning channel 12
[6.121344](node 3) beacon transmission stopped [channel:11] [PAN_ID:0]
[6.288160](node 6) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:3] ...
[6.289248](node 6) sending association request command ...
[6.290912](node 6) ack for association request command received
[6.325760](node 7) scanning channel 13
--- startDevice [8] ---
[6.500000](node 8) performing active channel scan ...
[6.500000](node 8) scanning channel 11
[6.588480](node 7) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:3] ...
[6.590208](node 7) sending association request command ...
[6.591712](node 7) ack for association request command received
[6.764000](node 8) scanning channel 12
[6.782432](node 6) sending data request command ...
[6.783712](node 6) ack for data request command received
[6.786176](node 6) association response command received
[6.786176](node 6) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:3]
[6.786176](node 6) begin to synchronize with the coordinator
--- startDevice [9] ---
[6.800000](node 9) performing active channel scan ...
[6.800000](node 9) scanning channel 11
--- startDevice [10] ---
[7.000000](node 10) performing active channel scan ...
[7.000000](node 10) scanning channel 11
[7.027680](node 8) scanning channel 13
[7.064320](node 9) scanning channel 12
[7.083232](node 7) sending data request command ...
[7.084512](node 7) ack for data request command received
[7.262080](node 10) scanning channel 12
[7.291680](node 8) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:4] ...
[7.292128](node 8) sending association request command ...
[7.293600](node 8) ack for association request command received
[7.327360](node 9) scanning channel 13
[7.524288](node 10) scanning channel 13
```

```
<!>[7.576032](node 7) association failed -> NO_DATA (beacon enabled)
[channel:11
] [PAN_ID:0] [CoordAddr:3]
[7.591040](node 9) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:4] ...
[7.592448](node 9) sending association request command ...
[7.594080](node 9) ack for association request command received
[7.785120](node 8) sending data request command ...
[7.786400](node 8) ack for data request command received
[7.788608](node 10) sending association request to [channel:11] [PAN_ID:0]
[Coor
dAddr:5] ...
[7.788864](node 8) association response command received
[7.788864](node 8) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:4]
[7.788864](node 8) begin to synchronize with the coordinator
[7.789696](node 10) sending association request command ...
[7.791200](node 10) ack for association request command received
[8.000768](node 3) beacon transmission successful [channel:11] [PAN_ID:0]
[8.085600](node 9) sending data request command ...
[8.086880](node 9) ack for data request command received
[8.282720](node 10) sending data request command ...
[8.284000](node 10) ack for data request command received
[8.286464](node 10) association response command received
[8.286464](node 10) association successful (beacon enabled) [channel:11]
[PAN_ID
:0] [CoordAddr:5]
[8.286464](node 10) begin to synchronize with the coordinator

Transmitting data ...

--- startDevice [7] ---
[8.576032](node 7) performing active channel scan ...
[8.576032](node 7) scanning channel 11
<!>[8.578400](node 9) association failed -> NO_DATA (beacon enabled)
[channel:11
] [PAN_ID:0] [CoordAddr:4]
[8.841440](node 7) scanning channel 12
[9.103520](node 7) scanning channel 13
[9.105984](node 5) beacon transmission successful [channel:11] [PAN_ID:0]
[9.366240](node 7) sending association request to [channel:11] [PAN_ID:0]
[Coord
Addr:3] ...
```

```
[9.368608](node 7) sending association request command ...
[9.370528](node 7) sending association request command ...
[9.372064](node 7) ack for association request command received
--- startDevice [9] ---
[9.578400](node 9) performing active channel scan ...
[9.578400](node 9) scanning channel 11
[9.842080](node 9) scanning channel 12
[9.863584](node 7) sending data request command ...
[9.864864](node 7) ack for data request command received
[9.867328](node 7) association response command received
[9.867328](node 7) association successful (beacon enabled) [channel:11]
[PAN_ID:
0] [CoordAddr:3]
[9.867328](node 7) begin to synchronize with the coordinator
[10.073344](node 4) beacon transmission stopped [channel:11] [PAN_ID:0]
[10.105760](node 9) scanning channel 13
[10.369440](node 9) sending association request to [channel:11] [PAN_ID:0]
[Coor
dAddr:4] ...
[10.369568](node 9) sending association request command ...
[10.371232](node 9) ack for association request command received
[10.862752](node 9) sending data request command ...
[10.864032](node 9) ack for data request command received
[10.866496](node 9) association response command received
[10.866496](node 9) association response command received
[10.866496](node 9) association successful (beacon enabled) [channel:11]
[PAN_ID
:0] [CoordAddr:4]
[10.866496](node 9) begin to synchronize with the coordinator

NS EXITING...
```

A.3 Κώδικας AWK

```
#! /usr/bin/awk -f
#
# Parse a ns2 wireless trace file and generate the following stats:
# - number of flows (senders)
# - time of simulation run
# - number of packets sent (at the Application)
# - number of packets received (at the Application)
```

```
# - number of packets dropped (at the Application)
# - number of collisions (802.11)
# - average delay
# - average throughput
# - average traffic rate (measured)
#
# Last updated: April 5, 2002 0113 mm
function average (array) {
    sum = 0;
    items = 0;
    for (i in array) {
        sum += array[i];
        items++;
    }
#    printf("DEBUG                sum is %d, items is %d\n", sum, items);
    if (sum == 0 || items == 0)
        return 0;
    else
        return sum / items;
}

function max( array ) {
    for (i in array) {
        if (array[i] > largest)
            largest = array[i];
    }
    return largest;
}

function min(array) {
    for (i in array) {
        if (0 == smallest)
            smallest = array[i];
        else if (array[i] < smallest)
            smallest = array[i];
    }
    return smallest;
}
BEGIN {
    total_packets_sent = 0;
    total_packets_received = 0;
    total_packets_dropped =0;
    first_packet_sent = 0;
    last_packet_sent = 0;
    last_packet_received = 0;
}
{
    event = $1;
    time = $2;
    node = $3;
    type = $4;
    reason = $5;
    packetid = $6;

# strip leading and trailing _ from node
    sub(/^_*/, "", node);
    sub(/_*$/, "", node);

    if ( time < simulation_start || simulation_start == 0 )
        simulation_start = time;
    if ( time > simulation_end )
        simulation_end = time;

    if ( reason == "COL" )
            total_collisions++;

    if ( type == "AGT" || type == "RTR" || type == "MAC") {
        nodes[node] = node; # to count number of nodes
```

```
        if ( time < node_start_time[node] || node_start_time[node] == 0 )
            node_start_time[node] = time;

        if ( time > node_end_time[node] )
            node_end_time[node] = time;

        if ( event == "s" ) {

            flows[node] = node; # to count number of flows
            if ( time < first_packet_sent || first_packet_sent == 0 )
                first_packet_sent = time;
            if ( time > last_packet_sent )
                last_packet_sent = time;
            # rate
            packets_sent[node]++;
            total_packets_sent++;

            # delay
            pkt_start_time[packetid] = time;
        }
       else if ( event == "r" ) {

            if ( time > last_packet_received )
                last_packet_received = time;
            # throughput
            packets_received[node]++;
            total_packets_received++;

            # delay
            pkt_end_time[packetid] = time;
        }
        else if ( event == "D" ) {

            total_packets_dropped++;
#           pkt_end_time[packetid] = time; # EXPERIMENTAL
        }
    }
}
END {
    print "" > "throughput.dat";
    print "" > "rate.dat";
    number_flows = 0;
    for (i in flows)
        number_flows++;

    # find dropped packets
    if ( total_packets_sent != total_packets_received ) {
        printf("***OUCH*** Dropped Packets!\n\n");
        for ( packetid in pkt_start_time ) {
            if ( 0 == pkt_end_time[packetid] ) {
                total_packets_dropped++;
#               pkt_end_time[packetid] = simulation_end; # EXPERIMENTAL
            }
        }
    }

    for (i in nodes) {
        if ( packets_received[i] > 0 ) {
            end = node_end_time[i];
            start = node_start_time[i - number_flows];
            runtime = end - start;
            if ( runtime > 0 ) {
                throughput[i] = packets_received[i]*8000 / runtime;
                printf("%d %f %f %d\n", i, start, end, throughput[i]) >>
"throughput.dat";
            }
        }
        # rate - not very accurate
```

```
        if ( packets_sent[i] > 2 ) {
            end = node_end_time[i];
            start = node_start_time[i];
            runtime = end - start;
            if ( runtime > 0 ) {
                rate[i] = (packets_sent[i]*8000) / runtime;
                printf("%d %f %f %d\n", i, start, end, rate[i]) >>
"rate.dat";
            }
        }
    }

    # delay
    for ( pkt in pkt_end_time) {
        end = pkt_end_time[pkt];
        start = pkt_start_time[pkt];
        delta = end - start;
        if ( delta > 0 ) {
            delay[pkt] = delta;
            printf("%d %f %f %f\n", pkt, start, end, delta) > "delay.dat";
        }
    }

    # offered load
    total_runtime = last_packet_sent - first_packet_sent;
    if ( total_runtime > 0 && total_packets_sent > 0)
        load = ((total_packets_sent * 8000)/ (total_runtime * 2000000)); #
no overhead

#printf("  \n");

    printf("\nFlows : %5d \nRuntime: %5.1f \nLoad: %7.4f \nPkt Sent: %8d
\nPkt Rxd: %8d \nPkts DRpd: %8d \nColisions: %10d \nAvg. Delay: %10.4f \nMax
Delay: %10.4f \nmin Delay: %10.4f \nAvg Thput: %12d \nRate: %12d\n",
           number_flows,
           total_runtime,
           load,
           total_packets_sent,
           total_packets_received,
           total_packets_dropped,
           total_collisions,
           average(delay),
           max(delay),
           min(delay),
           average(throughput),
           average(rate));

    printf("%5d %5.1f %7.4f %8d %8d %8d %10d %10.4f %10.4f %10.4f %12d
%12d\n",
           number_flows,
           total_runtime,
           load,
           total_packets_sent,
           total_packets_received,
           total_packets_dropped,
           total_collisions,
           average(delay),
           max(delay),
           min(delay),
           average(throughput),
           average(rate)) >> "stats.dat";
```

ΠΑΡΑΡΤΗΜΑ Β

Β.1 Mac_802_15_4.java

```
package Version1;
import java.util.*;

public class MAC_802_15_4
{

        //========  Parameters  =================
    int LogicalChannel;
    int CoordAddrMode = 0x02;
    int CoordPANId = 0x0000;
    String CoordAddress;
    long CapabilityInformation;
    boolean SecurityEnable = false;
    String DeviceAddress;
    boolean SecurityUse = false;
    int ACLEntry = 0x00;
    int AssocShortAddress = 0x0000;
    String status = "SUCCESS";
    boolean DisassociateEnable = false;
    int DisassociateReason;
    int BSN = 0x00;
    long SupeframeSpec;
    boolean GTSPermit = false;
    int LinkQuality = 0x00;
    int TimeStamp = 0x000000;
    boolean SecurityFailure = false;
    int DevAddress;
    long OrphanAddress;
    boolean SetDefaulPIB;
    boolean DeferPermit;
    int RxOnTime;
    int RxOnDuration;
    int ScanType;
    long ScanChannels;
    int ScanDuration;
    long UnscannedChannels;
    int ResultListSize;
    Vector EnergyDetectList;
    int BeaconOrder;
    int SuperframeOrder;
    boolean PANCoordinator;
    boolean BatteryLifeExtension;
    boolean CoordRealignment;
    boolean TrackBeacon;
    String LossReason;
    String PenAddrSpec;
    Vector AddrList;
    int sduLength;
    long sdu;
    Object PIBAttributeValue;
    long GTSCharacteristics;
    int ShortAddress;
    boolean AssociateEnable;
    boolean SetDefaultPIB = true;
    int PANId;
    boolean Connected = false;
```

```
    //=======  MAC sublayer constants  ======
    public static final int aBaseSlotDuration = 60;
    public static final int aBaseSuperframeDuration = 960;
    //public static final int aExtendedAddress;
    public static final int aMaxBE = 5;
    public static final int aMaxBeaconOverhead = 75;
    public static final int aMaxBeaconPayloadLength = 52;
    public static final int aGTSDescPersistenceTime = 4;
    public static final int aMaxFrameOverhead = 25;
    public static final int aMaxFrameResponseTime = 1220;
    public static final int aMaxFrameRetries = 3;
    public static final int aMaxLostBeacons = 4;
    public static final int aMaxMACFrameSize = 102;
    public static final int aMaxSIFSFrame = 18;
    public static final int aMinCAPLength =440;
    public static final int aMinLIFSPeriod = 40;
    public static final int aMinSIFSPeriod = 12;
    public static final int aNumSuperframeSlots = 16;
    public static final int aResponseWaitTime = 30720;
    public static final int aUnitBackoffPeriod = 20;

    //=======  MAC PIB attributes
    int macAckWaitDuration = 54;
    boolean macAssociationPermit = false;
    boolean macAutoRequest = true;
    boolean macBattLifeExt = false;
    int macBattLifeExtPeriods = 6;
    long macBeaconPayload;
    int macBeaconPayloadLength = 0;
    int macBeaconOrder = 15;
    int macBeaconTxTime = 0x000000;
    int macBSN ;
    long macCoordExtendedAddress;
    int macCoordShortAddress = 0xffff;
    int macDSN;
    boolean macGTSPermit = true;
    int macMaxCSMABackoffs = 4;
    int macMinBE = 3;
    int macPANId = 0xffff;
    boolean macPromiscuousMode = false;
    boolean macRxOnWhenIdle = false;
    int macShortAddress = 0xffff;
    int macSuperframeOrder = 15;
    int macTransactionPersistenceTime = 0x01f4;
    // ACL descriptor macACLEntryDescriptorSet = null;
    // int macACLEntryDescriptorSetSize = 0x00;
    boolean macDefaultSecurity = false;
    int macDefaultSecurityMaterialLength = 0x15;
    String macDefaultSecurityMaterial = "";
    int macDefaultSecuritySuite=0x00;
    int macSecurityMode = 0x00;
    int count = 0;
    //=======================================================

//============= MCPS ======================================

    int  SrcAddrMode = 0x00;
    int  SrcPANid    = 0x000;
    long SrcAddr;
    int  DstAddrMode = 0x00;
    int  DstPANId    = 0x0000;
    long DstAddr;
    int  msduLength;
    Object msdu;
    int  msduHandle  = 0x00;
    long TxOptions;
    int  mpduLinkQuality = 0x00;
```

```
  //============================================================

     String mlmeObj;
     String message;
     SensorNode sensor;

     public MAC_802_15_4( ){
     }


  //------------- Set methods  ----------------------------------------
-------

    public void setCoordPANId(int CoordPANId_) {
      CoordPANId = CoordPANId_;
      this.SetIsConnected(true) ;
    }

    public void setSetDefaultPIB(boolean SetDefaultPIB_){
     SetDefaultPIB = SetDefaultPIB_;
    }

    public void setBeaconOrder(int BeaconOrder_){
     BeaconOrder = BeaconOrder_;
    }

    public void setSuperframeOrder(int SuperframeOrder_){
     SuperframeOrder = SuperframeOrder_;
    }


    public void setBatLifeExtension(boolean BatteryLifeExtension_) {
      BatteryLifeExtension = BatteryLifeExtension_;
    }


    public void setDeviceAddress(String DeviceAddress_){
     DeviceAddress = DeviceAddress_;
    }

    public void setSrcPANid(int SrcPANid_){
     SrcPANid = SrcPANid_;
    }

    public void setSrcAddr(long SrcAddr_){
     SrcAddr = SrcAddr_;
    }

    public void setDstPANId(int DstPANId_){
     DstPANId = DstPANId_;
    }

    public void setDstAddr(long DstAddr_){
     DstAddr = DstAddr_;
    }

    public void SetIsConnected(boolean Connected_){
     Connected = Connected_;
    }


  //------------- Get methods  ----------------------------------------
-------
    public int getCoordPANId() {
      return CoordPANId;
    }

    public boolean getSetDefaultPIB(){
```

```
    return SetDefaultPIB;
   }

   public int getBeaconOrder(){
    return BeaconOrder;
   }

   public int getSuperframeOrder(){
    return SuperframeOrder;
   }


   public boolean getBatLifeExtension() {
     return BatteryLifeExtension;
   }


   public String getDeviceAddress(){
    return DeviceAddress;
   }

   public int getSrcPANid(){
    return SrcPANid;
   }

   public long getSrcAddr(){
    return SrcAddr;
   }

   public int getDstPANId(){
    return DstPANId;
   }

   public long getDstAddr(){
    return DstAddr;
   }

   public String getMLMEObj(){
    return mlmeObj;
   }

   public boolean isConnected(){
          return Connected;
   }

 //-------------------------------------------------------------------------
-------


 }
```

B.2 tempStats.java

```
package Version1;
import java.util.*;
import java.lang.Object.*;


class tempStats
{
    int msgSent;
```

```
    int msgRec;
    long EOSTime;
    double TotalEnerLeft;


    public tempStats(){
    }

    public void setMsgSent(int msgSent_){
           msgSent = msgSent_;
    }

    public void setMsgRec(int msgRec_){
           msgRec = msgRec_;
    }

    public void setEOSTime(long EOSTime_){
           EOSTime = EOSTime_;
    }

    public void setTotalEnerLeft(double TotalEnerLeft_){
           TotalEnerLeft = TotalEnerLeft_;
    }


    public int getMsgSent(){
      return msgSent;
    }

    public int getMsgRec(){
      return msgRec;
    }

    public long getEOSTime(){
      return EOSTime;
    }

    public double getTotalEnerLeft(){
      return TotalEnerLeft;
    }

}
```

Β.3 Πρόσθετες Μέθοδοι στην κλάση Simulation

```
  public tempStats TestConnectNodes() {
      // Establish connectivity between nodes according to range - Like
Initialization phase
      if (this.sensorNetworkNodes.size() > 0) {
        for (int j = 0; j < this.sensorNetworkNodes.size(); j++) {
          for (int k = 0; k < this.sensorNetworkNodes.size(); k++) {
            if ( ( ( (SensorNode)this.sensorNetworkNodes.elementAt(k)).
                  getCoordinateX() >=
                  ( (SensorNode)this.sensorNetworkNodes.elementAt(j)).
                  getCoordinateX() -
                  1.3 *
                  (
(SensorNode)this.sensorNetworkNodes.elementAt(j)).getONodeConfig().
                  getMaxRange()) && (
                ( (SensorNode)this.sensorNetworkNodes.elementAt(k)).
                getCoordinateX() <=
                ( (SensorNode)this.sensorNetworkNodes.elementAt(j)).
                getCoordinateX() +
                1.3 *
                (
(SensorNode)this.sensorNetworkNodes.elementAt(j)).getONodeConfig().
                getMaxRange())            &&             (              (
(SensorNode)this.sensorNetworkNodes.elementAt(k)).
                                  getCoordinateY() >=
                                  (
(SensorNode)this.sensorNetworkNodes.elementAt(j)).
                                  getCoordinateY() -
                                  1.3 *
                                  (
(SensorNode)this.sensorNetworkNodes.elementAt(j)).getONodeConfig().
                                  getMaxRange()) && (
                ( (SensorNode)this.sensorNetworkNodes.elementAt(k)).
                getCoordinateY() <=
                ( (SensorNode)this.sensorNetworkNodes.elementAt(j)).
                getCoordinateY() +
                1.3 *
                (
(SensorNode)this.sensorNetworkNodes.elementAt(j)).getONodeConfig().
                getMaxRange()))
              if (k != j) {
               tmpStatistics                        =                     (
(SensorNode)this.sensorNetworkNodes.elementAt(j)).
```

```
                    tryToConnectWith(
(SensorNode)this.sensorNetworkNodes.elementAt(k) );

                    msgSentNew = tmpStatistics.getMsgSent() + msgSentNew;
                    msgRecNew = tmpStatistics.getMsgRec() + msgRecNew;

                    eos = tmpStatistics.getEOSTime() + eos;
                }

            }
               tel =  tel - tmpStatistics.getTotalEnerLeft();
          }
        }
        tmpStatistics.setMsgSent(msgSentNew);
        tmpStatistics.setMsgRec(msgRecNew);
        tmpStatistics.setTotalEnerLeft(tel);
        tmpStatistics.setEOSTime(eos);
        return tmpStatistics;
      }


  public void TestCreateNodes2(int Width, int Height, int n) {

      if (Width != Height) {
        System.out.println("in a Grid, Width should be equal to height ");
        System.exit(0);
      }
      int number = (int) Math.sqrt(n);
      System.out.println("Grid " + number + "x" + number);
      int increment = Width / number;
      System.out.println("Increment " + increment);
      String Name = "";
      int i = 0;
      for (int X = 100; X < number * increment + 150; X += increment) {
        for (int Y = 100; Y < number * increment + 100; Y += increment) {
          //Name = "Node(" + X + "," + Y+")";
          Name = "" + i + "";
          if ( (i==0) || (i== 5) || (i==10) ){
            SensorNode   oNode   =   new   SensorNode(i,   Name,   X,   Y,
this.oNodeConfig, true);
            oNode.SetParent(this);
            this.InsertSensorYCoor(oNode, i);
            this.InsertSensorXCoor(oNode, i);
            this.sensorNetworkNodes.addElement(oNode);
          }
```

```
          else  {
              SensorNode   oNode   =   new   SensorNode(i,   Name,   X,   Y,
this.oNodeConfig, false);
            oNode.SetParent(this);
            this.InsertSensorYCoor(oNode, i);
            this.InsertSensorXCoor(oNode, i);
            this.sensorNetworkNodes.addElement(oNode);
          }

          i++;
          if (i == n)break;
        }
        if (i == n)break;
      }

    }


  public tempStats dataTransmission(){

      int numberofnodes = this.sensorNetworkNodes.size();
      SensorNode sender = (SensorNode)this.sensorNetworkNodes.elementAt(4);
      int senderPANid = sender.getCoordPANId();
      SensorNode                          PANcoord                          =
(SensorNode)this.sensorNetworkNodes.elementAt(senderPANid);
      SensorNode                          receiver                          =
(SensorNode)this.sensorNetworkNodes.elementAt(3);
      int receiverId = receiver.getSensorIndex();

      System.out.println(" (Node : 4) sending data to (Node: 3)");
      System.out.println("   (Node   :   4)   sending   data   to   (PAN   id:
"+senderPANid+")");

      Message   Msg1   =   new   Message(1,   "Run_IP_Broadcast_Flood","Data
sending...",receiverId, 0);
      this.ResetSimulation(this.getONodeConfig());
      IPSend  oIPSend  =  new  IPSend(0,  sender,  PANcoord,  Msg1,  sender,
this.oIPClass.eBroadcastFlooding);
       oIPSend.setDaemon(true);
       oIPSend.start();
       this.setDone(true);
       this.WaitForSimulationToFinish();
     //         System.out.print("     mexri     stigmhs     exoume     steilei:
"+this.getOMacAloha().oMacstats.getINumMsgSent()+" mhnymata  " );
```

```
      boolean ACK = this.getOIPClass().getIP_ACK();

      if (ACK == true ){
            System.out.println(" ");
            System.out.print("(Node: 4) successfully sent data to (PAN:
"+senderPANid+")");
          System.out.println();
          PANcoord.setDataQueue( Msg1 );
      }
      else{
            System.out.println(" ");
          System.out.println("Incomplete data transfer...");
          PANcoord.setDataQueue( null );
      }

      double tempEner1 = 10 - sender.getEnergyLeft() + 10 -
PANcoord.getEnergyLeft();
      long tempEOS1 = this.getOStatistics().getEndOfSimulationTime();
      int tempSent1 = this.getOMacAloha().oMacstats.getINumMsgSent();
      int tempRec1 = this.getOMacAloha().oMacstats.getINumMsgReceived();

      System.out.println(" ");
      System.out.print("(Node: 3) Asks (PAN: "+senderPANid+") for Pending
Data.");
      System.out.println();
      Message Msg2 = new Message(1, "Run_IP_Broadcast_Flood","Ask for
Pending Data...",receiverId, 0);
      this.ResetSimulation(this.getONodeConfig());
     IPSend oIPSend2 = new IPSend(0, receiver, PANcoord, Msg2, receiver,
this.oIPClass.eBroadcastFlooding);
      oIPSend2.setDaemon(true);
      oIPSend2.start();
      this.setDone(true);
      this.WaitForSimulationToFinish();
     //     System.out.print("     mexri     stigmhs     exoume     steilei:
"+this.getOMacAloha().oMacstats.getINumMsgSent()+" mhnymata  " );

      boolean ACK2 = this.getOIPClass().getIP_ACK();

      double tempEner2 = 10 - sender.getEnergyLeft() + 10 -
PANcoord.getEnergyLeft();
      long tempEOS2 = this.getOStatistics().getEndOfSimulationTime();
      int tempSent2 = this.getOMacAloha().oMacstats.getINumMsgSent();
      int tempRec2 = this.getOMacAloha().oMacstats.getINumMsgReceived();
```

```
      double tempEner3 = 0;
      long tempEOS3 = 0;
      int tempSent3 = 0;
      int tempRec3 = 0;

      this.ResetSimulation(this.getONodeConfig());
      if (pend == true ){
          System.out.println("Data  is  pending  for  (Node  3):  PAN  coord
sending pending data to receiver (Node:3)");
          System.out.println();

            IPSend   oIPSend3   =   new   IPSend(0,   PANcoord,   receiver,
PANcoord.getDataQueue(), PANcoord, this.oIPClass.eBroadcastFlooding);
            oIPSend3.setDaemon(true);
            oIPSend3.start();
            this.setDone(true);
            this.WaitForSimulationToFinish();
            //System.out.print("    mexri    stigmhs    exoume    steilei:
"+this.getOMacAloha().oMacstats.getINumMsgSent()+" mhnymata  " );

            System.out.println(     "Node     3     Energy     Left:     "+
sender.getEnergyLeft() );
            System.out.println(     "Node     4     Energy     Left:     "+
receiver.getEnergyLeft() );
            System.out.println(   "PAN   Coordinator   Energy   Left:   "+
PANcoord.getEnergyLeft() );

            tempEner3   =   10   -   sender.getEnergyLeft()   +   10   -
PANcoord.getEnergyLeft();
            tempEOS3 = this.getOStatistics().getEndOfSimulationTime();
            tempSent3 = this.getOMacAloha().oMacstats.getINumMsgSent();
            tempRec3  =  this.getOMacAloha().oMacstats.getINumMsgReceived();

      }
      else{
            System.out.println("There is no data pending...");
      }

      tmpStatistics2.setMsgSent(tempSent1 + tempSent2 + tempSent3);
      tmpStatistics2.setMsgRec(tempRec1 + tempRec2 + tempRec3);
      tmpStatistics2.setTotalEnerLeft(tempEner1 + tempEner2 + tempEner3);
      tmpStatistics2.setEOSTime(tempEOS1 + tempEOS2 + tempEOS3);

      return tmpStatistics2;
   }
```

```
   public  void dataTransmission2(){
       SensorNode sender = (SensorNode)this.sensorNetworkNodes.elementAt(4);
       int senderId = sender.getSensorIndex();
       SensorNode                          receiver                          =
(SensorNode)this.sensorNetworkNodes.elementAt(3);
       int receiverId = receiver.getSensorIndex();
       System.out.println();
       System.out.println(" (Node : 4) sending data to (Node: 3)");
       System.out.println();

       Message   Msg1   =   new   Message(1,   "Run_IP_Broadcast_Flood","Data
sending...",receiverId, 0);
      this.ResetSimulation(this.getONodeConfig());
      IPSend  oIPSend  =  new  IPSend(0,  sender,  receiver,  Msg1,  sender,
this.oIPClass.eBroadcastFlooding);
       oIPSend.setDaemon(true);
       oIPSend.start();
       this.setDone(true);
       this.WaitForSimulationToFinish();

       boolean ACK = this.getOIPClass().getIP_ACK();

       int senderPANid = sender.getCoordPANId();
      SensorNode                          PANcoord                          =
(SensorNode)this.sensorNetworkNodes.elementAt(senderPANid);

       if (ACK == true ){
             System.out.println(" ");
             System.out.print("(Node:  4)  successfully  sent  data  to  (Node:
3)");
          System.out.println();
          System.out.println(      "Node      3      Energy      Left:      "+
sender.getEnergyLeft() );
             System.out.println(     "Node     4     Energy     Left:     "+
receiver.getEnergyLeft() );
             System.out.println(    "PAN    Coordinator    Energy    Left:    "+
PANcoord.getEnergyLeft() );
       }
       else{
             System.out.println(" ");
          System.out.println("Incomplete data transfer...");
       }

   }
```

```
  public void setPend(boolean pend_){
    pend = pend_;
  }
```

Β.4 Πρόσθετες Μέθοδοι στη Κλάση SensorNode

```
 public void addAssociatedDev(SensorNode otherNode){

       Vector vAssociatedDev = new Vector();
       vAssociatedDev.add(otherNode);
       this.vNeighbors2.add(vAssociatedDev);
     }



 public tempStats associate(SensorNode otherNode){

          Message        Msg1        =        new        Message(1,
"Run_IP_Broadcast_Flood","Association Request", 0);
      //   IPSend oIPSend = new IPSend(0, otherNode,otherNode, Msg1, this,
this.oIPClass.eBroadcastFlooding);
        if (this.PANCoordinator == true && otherNode.isConnected() ==
false && otherNode.getPANCoordinator() == false){


          if ( this.GetCheckIfCoord() == false){
                System.out.println("Starting PAN Coordinator, PAN id: "+
sensorIndex);
                System.out.println("Scanning...");
                System.out.println("Node Range: "+ this.getMyRange());
                System.out.println( "Node "+sensorIndex+" Energy Left:
"+this.getEnergyLeft() );
                this.SetCheckIfCoord(true);
          }
```

```
                boolean other;
                other =  otherNode.getPANCoordinator();
                int ono = otherNode.getSensorIndex();


this.oSimulation.ResetSimulation(this.oSimulation.getONodeConfig());
                System.out.print(" (Node : "+ ono+") sending association
request to (PAN id: "+ sensorIndex+") at ");
           //    try{ ;
          // this.oSimulation.Run_IP_Flood2(1, "Association Request");
               IPSend oIPSend = new IPSend(0, otherNode, this, Msg1,
otherNode, this.oIPClass.eBroadcastFlooding);
                oIPSend.setDaemon(true);
                oIPSend.start();
             // }catch(Exception e){}
            //       MACSend oMACSend = new MACSend(otherNode, otherNode,
this,Msg1, 0, otherNode.oIPClass.eBroadcastFlooding);
             //           oMACSend.setDaemon(true);
             //           oMACSend.start();
             //      boolean                    ACK                    =
this.getOSimulation().getOMacAloha().MacSend(0, otherNode, otherNode, this,
Msg1, otherNode.oIPClass.eBroadcastFlooding);
             //
this.getOSimulation().getOMacAloha().ReceiveTheMessage(otherNode,
otherNode, this,Msg1, 0,oIPClass.eBroadcastFlooding);
             this.oSimulation.setDone(true);
                  this.oSimulation.WaitForSimulationToFinish();

                   boolean                    ACK                    =
this.oSimulation.getOIPClass().getIP_ACK();

                if (ACK == true ){
                  System.out.print("(Node    "+sensorIndex+")    Ack    for
association request received by (node "+ono+")");
                  System.out.println();
                  otherNode.setCoordPANId( sensorIndex );
                  System.out.println("Association:  successful  -     Node
Address: "+ ono+", PAN Address: "+ sensorIndex );
                  System.out.println(  "Node  "+sensorIndex+"  Energy  Left:
"+this.getEnergyLeft() );
                  System.out.println(" ");
                  this.addAssociatedDev(otherNode);
                }
                else  {
```

```
                System.out.println("Ack    for    association    request
received");
                System.out.println("Association: unsuccessful" );
                System.out.println(  "Node  "+sensorIndex+"  Energy  Left:
"+this.getEnergyLeft() );
              }

        // this.oSimulation.setDone(false);
       //  this.oSimulation.setEndSystemClock(false);
       //  this.oSimulation.WaitForSimulationToFinish();


          //   this.getOSimulation().WaitForSimulationToFinish();

          //

        }

tmpst1.setEOSTime(otherNode.oSimulation.getOStatistics().getEndOfSimulation
Time() );
          tmpst1.setTotalEnerLeft( 10 - otherNode.getEnergyLeft() );
         tmpst1.setMsgSent(
otherNode.oSimulation.getOMacAloha().oMacstats.getINumMsgSent()) ;

tmpst1.setMsgRec(otherNode.oSimulation.getOMacAloha().oMacstats.getINumMsgR
eceived()                                                              -
otherNode.oSimulation.getOMacAloha().oMacstats.getINumofSensing()-
otherNode.oSimulation.getOMacAloha().oMacstats.getINumofSensing1()      -
otherNode.oSimulation.getOMacAloha().oMacstats.getINumofSensing2()-
otherNode.oSimulation.getOMacAloha().oMacstats.getINumofSensing3() );


        return tmpst1;
  }

  public void SetCheckIfCoord(boolean check_){
     check = check_;
   }

   public boolean GetCheckIfCoord(){
     return check;
   }

  public void setDataQueue(Message data_){
```

```
    data = data_;
  }

  public Message getDataQueue(){
    return data;
  }
```

Β.5 Πρόσθετες Μέθοδοι στη κλάση Message

```
   public Message(int ID, int type,String msg, String mlmeObj_, long
pTimeStamp) {
     UniqueId = counter++;
     FixedId = ID;
     Payload = msg;
     msgType = type;    // So that we can discriminate between each kind
of messages
     mlmeObj = mlmeObj_;
     this.msgTimeStamp = pTimeStamp;
   }

   public Message(int type,String msg,String mlmeObj_/*mv*/,int destId_,
long pTimeStamp) {
     UniqueId = counter++;
     FixedId = this.getUniqueTrafficID();
     Payload = msg;
     msgType = type;    // So that we can discriminate between each kind
of messages
     destID = destId_;
     this.msgTimeStamp = pTimeStamp;
         mlmeObj = mlmeObj_; //mv
   }

   public String getMlmeObj() { //mv
     return mlmeObj;                              //mv
   }

  public void setMlmeObj(String MlmeObj_) { //mv
     mlmeObj = MlmeObj_;                            //mv
```

```
    }
```

Β.6 Πρόσθετες Μέθοδοι στη κλάση IPClass

```
    public void IP_Broadcast_Flood(int pLobe, SensorNode pSource,
SensorNode pDestination,Message msg, SensorNode pPrevious, int
iTypeOfRouting) {

      iTypeOfRouting = this.eBroadcastFlooding;

      //System.out.println("plobe: " + pLobe);
      long CurrentTime = msg.getMsgTimeStamp();

      if (pSource.getSensorIndex() == pDestination.getSensorIndex()) {
        SuppressDuplicates(pSource, msg);
        this.SetIfTrafficReachedDest(msg.getFixedId(), 1);

this.oSimulation.getOStatistics().setDeliveryTime(msg.getMsgTimeStamp());
        this.oSimulation.setDone(true);
       return;
      }
      // suppress any duplicates
      if (SuppressDuplicates(pSource, msg) == true) {
       return;
      }

      this.oSimulation.getOSystemTime().setFutureEvent(CurrentTime        +
this.oSimulation.getONodeConfig().getProcessingTime(),
true,"IP_Processing");
      while(this.oSimulation.getOSystemTime().getISemaphore()!=1){}
      this.oSimulation.getOSystemTime().CallEvent(CurrentTime             +
this.oSimulation.getONodeConfig().getProcessingTime(),"IP_Processing");
      msg.setMsgTimeStamp(CurrentTime                                     +
this.oSimulation.getONodeConfig().getProcessingTime());

      //int                          BackofPeriod                         =
this.oSimulation.getOMacAloha().CalculateBackoffPeriod(0);
      // send broadcast through the mac Layer


            Message    uniqueMsg    =    new    Message(msg.getFixedId(),
msg.getMsgType(),msg.getPayload(),/*mv*/msg.getMlmeObj(),CurrentTime);
```

```
              IP_ACK    =    this.oSimulation.getOMacAloha().MacSend(pLobe,
pSource,null, pDestination, uniqueMsg, iTypeOfRouting);
              System.out.print("["+uniqueMsg.getMsgTimeStamp()+"]:");
              System.out.print(" ");
              this.setAck(IP_ACK);

 try{
                  if   ( msg.getMlmeObj().equals("Ask for Pending Data..."
) ){
                      this.oSimulation.setPend(true);
               }
               else this.oSimulation.setPend(false);
              }catch(NullPointerException npe){ }
             if (IP_ACK == true) {

                  CurrentTime = uniqueMsg.getMsgTimeStamp();
             }

 }

   public void setAck(boolean ack_){
    IP_ACK = ack_;
   }
   public boolean getIP_ACK() {
     return IP_ACK;
   }
```

Β.7 Πρόσθετες Μέθοδοι στη κλάση testscenarios.java

```
   public void PrintToFile(String sString) {

       try {
         FileOutputStream foutstream = new FileOutputStream("results.txt",
true);
         BufferedWriter      out      =      new      BufferedWriter(new
OutputStreamWriter(foutstream));
         try {
           out.write(sString);
           out.newLine();
           out.close();
         }
```

```
        catch (IOException ex1) {
          System.err.println(ex1);
        }
      }
      catch (FileNotFoundException ex) {
        System.err.println(ex);
      }

  }
```

Στη μέθοδο SetupSimulation() έχει προστεθεί ο εξής κώδικας:

```
    msgSent1 = this.oSimulation.TestConnectNodes().getMsgSent();
    msgRec1 = this.oSimulation.TestConnectNodes().getMsgRec();
    tel1 = this.oSimulation.TestConnectNodes().getTotalEnerLeft();
    eos1 = this.oSimulation.TestConnectNodes().getEOSTime();

    int totSent;
    int totRec;
    long totEOS;
    double totEner;

    msgSent2 = msgSent2 + msgSent1;
   msgRec2 = msgRec2 + msgRec1;
   eos2 = eos2 + eos1;
   tel2 = tel1 - tel2;
    tempStats newStat = new tempStats();
    int index = 0;
   while (index < i){
      newStat = this.oSimulation.dataTransmission();
        msgSent2 = newStat.getMsgSent() + msgSent2;
        msgRec2 = newStat.getMsgRec() + msgRec2;
        tel2 = tel2 - newStat.getTotalEnerLeft();
        eos2 = newStat.getEOSTime() + eos2;
       index = index +1;

       System.out.println("Total Messages Sent: " + msgSent2);
        System.out.println("Total Messages Received: " + msgRec2);
        System.out.println("Total Energy Left: " + tel2);
        System.out.println("EOS: " + eos2);

        this.PrintToFile("\t" );
        this.PrintToFile("\t" + "Simulation Num: " + index);
        this.PrintToFile("\t" + "Total Messages Sent: " + msgSent2);
```

```
        this.PrintToFile("\t" + "Total Messages Received: " + msgRec2);
        this.PrintToFile("\t" + "Total Energy Left: " + tel2);
        this.PrintToFile("\t" + "EOS: " + eos2);

    }
```