ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ

ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Υλοποίηση Πρωτοκόλλου CSMA για Ασύρματα Δίκτυα Αισθητήρων σε Ενσωματωμένο Ραδιόφωνο Ελεγχόμενο από Μικροελεγκτή 8051

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Παπαμεντζελόπουλος

Ιούλιος 2012

ΕΠΙΤΡΟΠΗ

Επίκουρος Καθηγητής Άγγελος Μπλέτσας - Επιβλέπων Καθηγητής

Επίκουρος Καθηγητής Αντώνιος Δεληγιαννάκης

Επίκουρος Καθηγητής Πολυχρόνης Κουτσάκης

Περίληψη

Η εργασία αυτή έχει ως σκοπό την ανάλυση και υλοποίηση ενός πρωτοκόλλου, που θα διασφαλίζει την επικοινωνία ασύρματων κόμβων σε ένα δίκτυο αισθητήρων, βασισμένο σε πλατφόρμες τύπου 8051, με στόχο την χαμηλή κατανάλωση ενέργειας και την αξιόπιστη μετάδοση. Επιπλέον στόχος είναι να αποτελέσει μία βάση και ένα εργαλείο, για την δικτύωση και περαιτέρω ανάπτυξη ενός τέτοιου δικτύου. Στα πλαίσια αυτά υλοποιήθηκε ένα πρωτόκολλο ελέγχου πρόσβασης στο μέσο (MAC - Medium Access Control), με βάση το πρότυπο IEEE 802.15.4 και ένα σύστημα αφύπνισης και περιοδικής αποστολής πακέτων, απαραίτητο σε δίκτυα αισθητήρων, χαμηλής κατανάλωσης ενέργειας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή

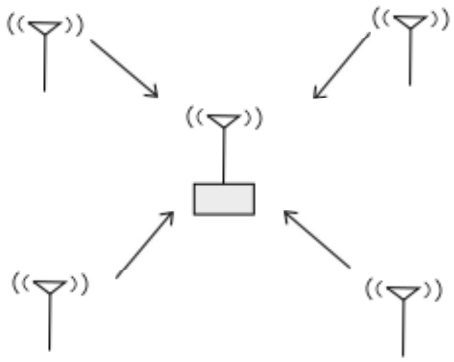
1.1 Ασύρματα Δίκτυα Προσωπικού Χώρου (WPANs) - IEEE 802.15.4

Τα τελευταία χρόνια έχει δοθεί μεγάλο ενδιαφέρον στην ανάπτυξη των ασύρματων δικτύων προσωπικού χώρου, γνωστά ως WPAN (Wireless Personal Area Networks). Πρόσφατα η επιστημονική κοινότητα της IEEE στα πλαίσια της δουλειάς της για τα πρότυπα πρωτόκολλα 802, ανακοίνωσε το παγκόσμιο πρότυπο 802.15.4 [1] για τα WPAN. Στο πρότυπο αυτό αναφέρονται λεπτομερώς, ο τρόπος με τον οποίο γίνεται πολλαπλή πρόσβαση στο κανάλι από πολλούς κόμβους (Medium Access Control - MAC), καθώς και τα χαρακτηριστικά του φυσικού επιπέδου (Physical Layer - PHY).

Τα WPANs χρησιμοποιούνται για την διάδοση πληροφορίας σε μικρή εμβέλεια και σε μικρούς ρυθμούς μέχρι 250kb/s. Λειτουργούν στην μπάντα συχνοτήτων στα 868/915 MHz ή στα 2450 MHz. Σε αντίθεση με τα WLANs, η σύνδεση των κόμβων απαιτεί ελάχιστη, ως και καμία επιπλέον υποδομή. Σκοπός σε αυτά τα δίκτυα δεν είναι τόσο ο μεγάλος ρυθμός πληροφορίας, όσο η αξιοπιστία.

Για αυτό το λόγο και τα WPAN δίκτυα αναφέρονται κυρίως σε εφαρμογές αισθητήρων και ελέγχου. Αυτά τα χαρακτηριστικά δίνουν την δυνατότητα για φτηνές σε κόστος και χαμηλής κατανάλωσης ενέργειας, λύσεις δικτύωσης πολλών ειδών συσκευών. Απώτερος σκοπός είναι αντίστοιχες τεχνολογίες τηλεπικοινωνιών, όπως IrDA, Bluetooth, Wireless USB, Z-Wave, ZigBee, να μπορούν να δικτυωθούν κάτω από μια κοινή σκοπιά και να ενοποιηθούν, βασιζόμενες σε ένα κοινό πρωτόκολλο.

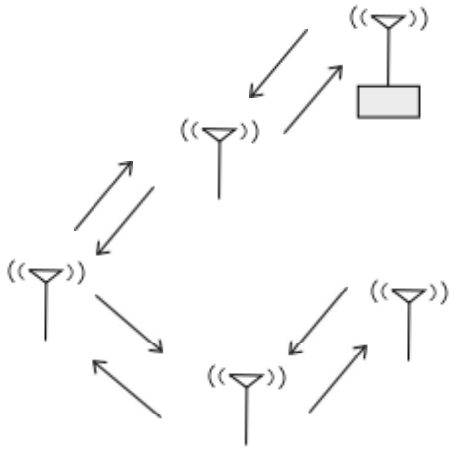
Το πρωτόκολλο καλύπτει την δικτύωση κόμβων που βρίσκονται σε τοπολογία αστέρα(star topology) και σε τοπολογία σύνδεσης κόμβου με κόμβο(peer-to-peer), όπως ορίζονται και στο πρότυπο, και φαίνονται στα σχήματα 1 και 2.



Σχήμα 1: Κόμβοι σε τοπολογία αστέρα (star topology)

Στην εφαρμογή του χωραφιού, θέλουμε να δικτυώσουμε διάφορα φυτά, με κόμβους-αισθητήρες οι οποίοι θα παίρνουν μετρήσεις υγρασίας , θερμοκρασίας κτλ. Και θα μεταδίδουν την πληροφορία αυτή σε έναν απομακρυσμένο κόμβο-Σταθμό Βάσης (Base Station).

Χρησιμοποιήσαμε λοιπόν την τοπολογία αστέρα, ως λύση για την δικτύωση των κόμβων-αισθητήρων, με σκοπό και τη μείωση των μεταδόσεων (όπως θα είχαμε από μια peer-to-peer τοπολογία, με πολλά μονοπάτια). Για την κάλυψη αποστάσεων, πέρα από τα όρια όμως του ενός χωραφιού, μπορεί να χρησιμοποιηθεί μια ιεραρχική δομή, από πολλά δίκτυα αστέρα, που θα συνδέονται μεταξύ τους, μέσω των κόμβων-βάσης τους.



Σχήμα 2: Κόμβοι σε τοπολογία peer-to-peer

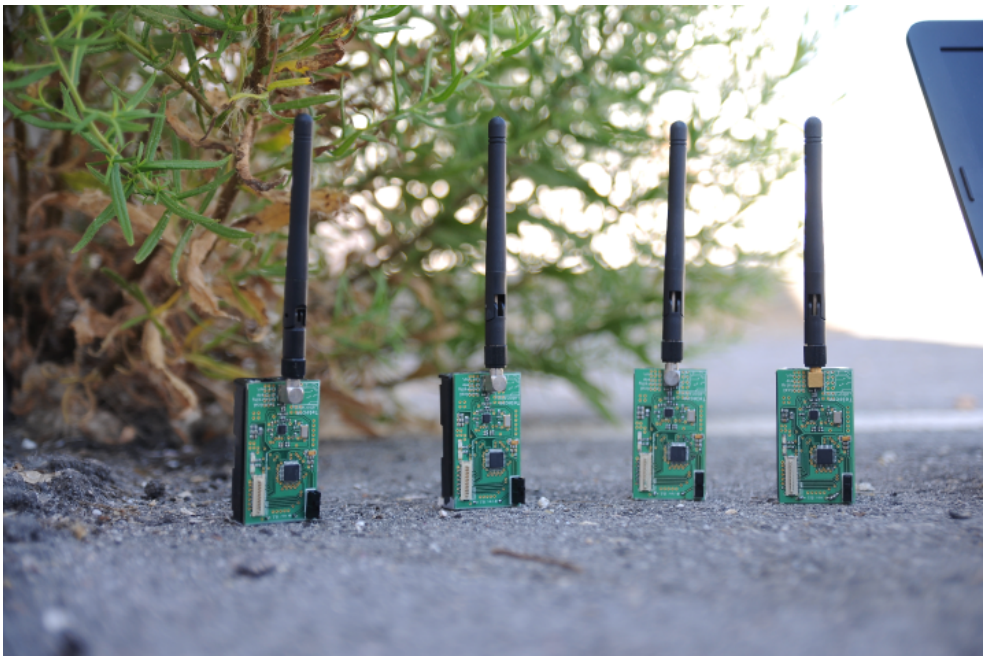
Η εργασία αυτή στηρίζεται σε αυτή την σκοπιά και προσπαθεί να ενσωματώσει τα χαρακτηριστικά εκείνα στα οποία βασίζεται το 802.15.4 πρότυπο, με κυριότερο χαρακτηριστικό τον τρόπο της πολλαπλής πρόσβασης κόμβων στο μέσο (MAC).

1.2 Ασύρματα Δίκτυα Αισθητήρων - iCubes

Το πεδίο των ασύρματων δικτύων αισθητήρων είναι ένα συνεχώς αναπτυσσόμενο πεδίο. Ως ασύρματο δίκτυο αισθητήρων (WSN - Wireless Sensor Network) ορίζεται ένα δίκτυο το οποίο αποτελείται από διάσπαρτους ανεξάρτητους κόμβους, οι οποίοι καταγράφουν διαφόρων ειδών μετρήσεις (θερμοκρασίας, ήχου, πίεσης κτλ.) και συνεργαζόμενοι διαδίδουν τα δεδομένα τους μέσα από το δίκτυο, σε μία περιοχή βάσης και καταγραφής αυτών.

Συνήθως οι κόμβοι αυτοί αποτελούνται από διάφορα συγκεκριμένα μέρη: έναν ασύρματο πομποδέκτη, είτε με ενσωματωμένη κεραία, είτε με σύνδεση με εξωτερική, έναν μικροελεγκτή , ένα ηλεκτρικό κύκλωμα για την διασύνδεση των αισθητήρων και κάποια μορφή παροχής αυτόνομης ηλεκτρικής ενέργειας (συνήθως μπαταρίες) ή ακόμη καλύτερα και κάποια μορφή συλλογής ενέργειας από εξωτερικές πηγές (energy harvesting). Το κόστος για τέτοιους κόμβους εξαρτάται κυρίως από τον προσανατολισμό της λειτουργίας τους, συνήθως όμως ο γενικός προσανατολισμός είναι το κόστος να είναι μικρό, ώστε το δίκτυο αισθητήρων να έχει πολλούς κόμβους.

Για τον σκοπό αυτό, χρησιμοποιήθηκαν τα, «iCubes». Τα iCubes είναι ασύρματοι τηλεπικοινωνιακοί κόμβοι, οι οποίοι δημιουργήθηκαν στα πλαίσια του μαθήματος «Ανάλυση και Σύνθεση Τηλεπικοινωνιακών Διατάξεων» (ΤΗΛ404), του τμήματος «Ηλεκτρονικών Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών» Πολυτεχνείου Κρήτης, το 2010 και μπορούν να αποτελέσουν αποτελεσματικούς κόμβους, δικτύου αισθητήρων, συνδέοντας κάποιον αισθητήρα επιλογής, σαν εξωτερική επέκταση.



Σχήμα 3: iCubes

Σκοπός ήταν να δημιουργηθούν οι κατάλληλοι κόμβοι για να μπορέσει να δικτυωθεί ένα χωράφι με αισθητήρες υγρασίας. Τα κίνητρα της εργασίας αυτής, είναι να θέσει μια βάση για το πρωτόκολλο επικοινωνίας των κόμβων αυτών, ώστε να διασφαλίζει την σωστή μετάδοση πληροφορίας αλλά και την ενεργειακή αυτοδυναμία των κόμβων, σε εφαρμογές που απαιτούν σποραδική μετάδοση, σε μεγάλη διάρκεια χρόνου, π.χ της παρακολούθησης υγρασίας των φυτών, σε ένα χωράφι.

Η σωστή διαχείριση της κατανάλωσης ενέργειας σε κάθε κόμβο, είναι ίσως από τα σημαντικότερα θέματα από τη σκοπιά των δικτύων αισθητήρων, καθώς αποτελεί σημαντικό κριτήριο για την βιωσιμότητα και την ύπαρξη του δικτύου. Η εργασία αυτή αναφέρεται και στο θέμα της κατανάλωσης ενέργειας στο επίπεδο της εφαρμογής και παραθέτει διάφορες λειτουργίες, που εστιάζουν στην ελαχιστοποίηση της κατανάλωσης ενέργειας.

2 Μικροελεγκτής (MCU) - C8051F320

2.1 Περιγραφή

Ο μικροελεγκτής που χρησιμοποιούν, οι κόμβοι iCubes του δικτύου μας, είναι ο C8051F320 της Silicon Laboratories. Είναι ένας μικροελεγκτής με πυρήνα τύπου CIP-51, ο οποίος είναι πλήρως συμβατός με το σετ εντολών του Intel MCS-51 (γνωστός απλά ως 8051) καθώς και με τους συμβολομεταφραστές τύπου 803x/805x.

Το μεγάλο όφελος της αρχιτεκτονικής του CIP-51 πυρήνα είναι ότι χρησιμοποιεί αρχιτεκτονική διοχέτευσης, η οποία αυξάνει τον ρυθμό διεκπεραίωσης εντολών σημαντικά. Πιο συγκεκριμένα, στην κλασσική αρχιτεκτονική 8051 όλες οι εντολές εκτός των MUL και DIV (πολλαπλασιασμού και διαίρεσης) απαιτούν 12 ή 24 κύκλους ρολογιού για να υλοποιηθούν, με ρολόι επεξεργαστή από 12 MHz - 24 MHz. Στην αρχιτεκτονική διοχέτευσης του CIP-51, το 70% των εντολών υλοποιούνται σε 1 με 2 κύκλους ρολογιού, όπως φαίνεται και στο αναλυτικό σχήμα παρακάτω.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	7	3	1	2	1

Σχήμα 4: Κύκλοι εντολών της αρχιτεκτονικής CIP-51 στο [4]

Τα βασικότερα τεχνικά χαρακτηριστικά του μικροελεγκτή μας είναι τα εξής:

- Πλήρως συμβατός με το σύνολο εντολών του πυρήνα 8051.
- Μέγιστη ταχύτητα διεκπεραίωσης εντολών 25 MIPS στα 25 MHz .
- Ρολόι πυρήνα έως 25 MHz.
- 256 bytes εσωτερικής μνήμης RAM και 16k bytes FLASH μνήμη.
- 25 θύρες I/O.
- Αυξημένες δυνατότητες χειριστή διακοπών (Interrupt Handler).

- Προγραμματιζόμενες λειτουργίες χαμηλής κατανάλωσης (περίπου 1 mA) Idle Mode - Stop Mode.

2.2 Εργαλεία Προγραμματισμού - Μεταγλώττισης

Ο προγραμματισμός του μικροελεγκτή έγινε με τον Keil C51 μεταφραστή/μεταγλωττιστή (compiler) της ARM Ltd, για την γλώσσα προγραμματισμού C. Είναι από τους πιο δημοφιλείς μεταφραστές στον κόσμο, για αρχιτεκτονική 8051. Παρέχει αρκετές δυνατότητες στον χρήστη για άμεση και πλήρη επαφή και διαχείριση όλων των πόρων ενός 8051 επεξεργαστή.

Τα βασικότερα χαρακτηριστικά του μεταφραστή Keil C51 είναι:

- Υποστηρίζει 9 βασικά είδη πληροφορίας, συμπεριλαμβάνοντας και 32-bit IEEE floating-point.
- Παρέχει ευέλικτη διαμόρφωση της διαθέσιμης μνήμης, ορίζοντας μεταβλητές διαφορετικού τύπου μνήμης data, bdata, idata, xdata, pdata.
- Οι συναρτήσεις διαχείρισης διακοπών (interrupt handler routines), μπορούν να γραφούν σε C.
- Δυνατότητα διευθυνσιοδότησης bit.

2.2.1 Σχεδίαση Στρώματος Περίληψης Υλικού - HAL

Το στρώμα περίληψης υλικού (HAL - Hardware Abstraction Layer) είναι ένα στρώμα κώδικα μεταξύ του φυσικού υλικού και του καθαρού λογισμικού. Ο σκοπός είναι να παραμένουν αφανείς οι λεπτομέρειες του υλικού (ονόματα καταχωρητών, διευθύνσεις μνήμης, περιφερειακά κτλ) έτσι ώστε να παραχθεί μια συνεπής πλατφόρμα/βιβλιοθήκη την οποία να μπορεί ο κάθε χρήστης, ανεξαρτήτως της γνώσης του και της επαφής του με το υλικό, να τη χρησιμοποιεί για να τρέχει διάφορες εφαρμογές/προγράμματα.

Αυτό έχει αρκετά πλεονεκτήματα, με βασικότερο ότι σε περίπτωση που το φυσικό υλικό αλλάζει κάθε φορά, ο χρήστης δεν θα χρειάζεται να αλλάζει την μορφή του κώδικα της εφαρμογής του και να τον τροποποιεί σε μεγάλο βαθμό, να εξαρτάται δηλαδή από το υλικό (hardware), αλλά θα χρησιμοποιεί τις εκάστοτε ανάλογες βιβλιοθήκες για το κατάλληλο υλικό του κάθε φορά.

Αυτή η λογική είναι πολύ σημαντική στα ενσωματωμένα συστήματα (embedded systems) έτσι ώστε να μπορεί το λογισμικό των εφαρμογών που υλοποιείται, να ενσωματώνεται και να εξελίσσεται γρήγορα και εύκολα σε διαφορετικές πλατφόρμες. Σε αυτήν την προσπάθεια έχει κινηθεί και η οργάνωση του κώδικα σε αυτή την εργασία, έτσι ώστε μελλοντικές εφαρμογές να μπορούν να βασιστούν, τροποποιήσουν ή χρησιμοποιήσουν κομμάτια αυτής της εργασίας χωρίς να πρέπει να αλλάζουν τον κώδικα της εφαρμογής τους.

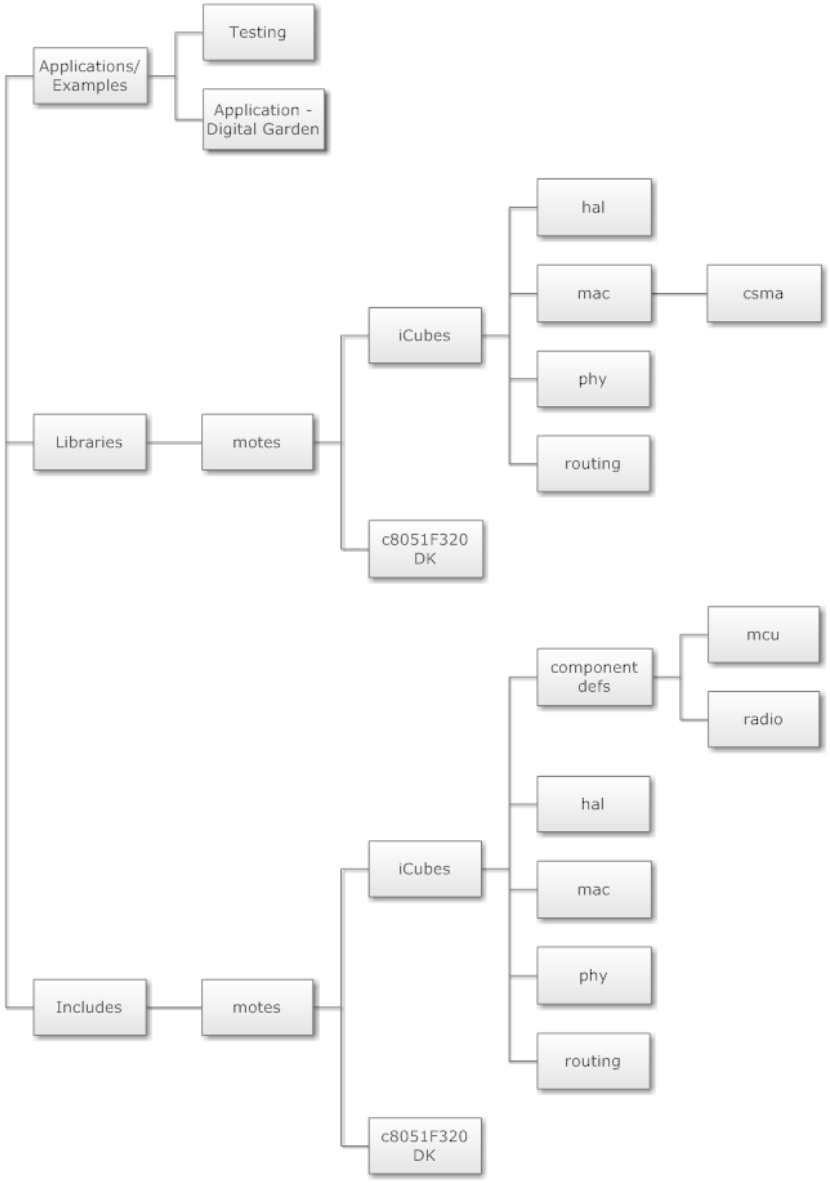
Μερικά από τα βασικά κομμάτια της οργάνωσης (HAL) που έγινε όπως φαίνεται στο σχήμα 5 είναι τα εξής:

- Examples-Applications : Εδώ βρίσκουμε υλοποιήσεις και λογισμικό εφαρμογών (αλγορίθμους, εφαρμογές, κτλ).

- Includes : Εδώ βρίσκουμε τους ορισμούς διαφόρων συναρτήσεων και τις περιγραφές σταθερών που έχουν να κάνουν με το επίπεδο υλικού και το χαμηλό επίπεδο γενικότερα (φυσικό επίπεδο, ονομασίες καταχωρητών-θυρών, κτλ)

- Libraries: Εδώ βρίσκουμε τις υλοποιήσεις των διαφόρων συναρτήσεων χαμηλού επιπέδου που βασίζονται τους ορισμούς που έχουν γίνει παραπάνω και υλοποιούν κυρίως

βασικές λειτουργίες χαμηλού επιπέδου, διασύνδεσης περιφερειακών κτλ.



Σχήμα 5: Παράδειγμα οργάνωσης κώδικα όπως στο [6]- HAL

3 Πομποδέκτης (RF tranceiver) - CC2500

3.1 Περιγραφή

Οι κόμβοι χρησιμοποιούν επίσης ραδιοφωνικό πομποδέκτη για την τηλεπικοινωνιακή ζεύξη και δικτύωση τους. Ο πομποδέκτης των iCubes είναι ο CC2500 της Texas Instruments. Είναι ένας πομποδέκτης χαμηλού κόστους, στα 2.4 GHz ειδικά σχεδιασμένος για ασύρματες εφαρμογές, χαμηλής κατανάλωσης ενέργειας.

Το υλικό αυτό είναι σχεδιασμένο για την μπάντα συχνοτήτων 2400-2483.5 MHz, η οποία χαρακτηρίζεται ως ISM (Industrial, Scientific, Medical) και SRD (Short Range Device). Το ραδιόφωνο CC2500 παρέχει εκτενή έλεγχο και διαχείριση των πακέτων, δυνατότητα στοίβας πακέτων (buffering), δυνατότητα εκπομπής εκρηκτικής ροής (burst transmission), ανίχνευση φέροντος και ελεύθερου καναλιού(CS - carrier sense /CCA - clear channel assessment), έλεγχο ποιότητας ζεύξης (LQI - link quality indication) και δυνατότητα λειτουργίας σε χαμηλή κατανάλωση και αφύπνισης μέσω του ραδιοφώνου (Wake-On-Radio). Τα βασικότερα τεχνικά χαρακτηριστικά του:

- Υψηλή ευαισθησία δέκτη μέχρι και -104dBm στα 2.4 KBaud(kilo symbols/sec) και 1% ποσοστού σφάλματος (packet error rate).
- Χαμηλή κατανάλωση ρεύματος 13.3 mA σε κατάσταση λήψης (RX state) για ισχυρά σήματα στην είσοδο.
- Ρυθμιζόμενη ισχύς εκπομπής από -50dBm μέχρι +1dBm.
- Προγραμματιζόμενος ρυθμός εκπομπής από 1.2 μέχρι 500 KBaud.
- Συχνότητα φέροντος εκπομπής 2400 - 2483.5 MHz και δυνατότητα γρήγορης αλλαγής φέροντος (frequency hopping).
- Ευέλικτες λειτουργίες για συστήματα βασισμένα σε πακέτα: Ανίχνευση προθέματος και συντονισμός (preamble, sync word detection) , έλεγχος διεύθυνσης (address check),

μεταβλητό μέγεθος πακέτων, αυτόματος κυκλικός έλεγχος πλεονασμού (cyclic redundancy check - CRC).

- Πλήρη συμβατότητα με την διασύνδεση τύπου SPI.
- Προγραμματιζόμενο πλάτος φίλτρου συχνοτήτων.
- Δυνατότητα λειτουργίας σε πολύ χαμηλή κατανάλωση (400 nA), κατάσταση SLEEP mode και γρήγορη επαναφορά (240 us) σε κατάσταση λήψης ή μετάδοσης (RX - TX state).
- Παροχή ένδειξης για την ισχύ του λαμβανόμενου σήματος (Received signal strength indicator - RSSI).

3.2 Φυσικό επίπεδο

Είναι σημαντικό να παραθέσουμε ορισμένα στοιχεία που αφορούν το φυσικό επίπεδο ζεύξης για τον πομποδέκτη CC2500 και να τα συγκρίνουμε, με τα αντίστοιχα στοιχεία του προτύπου 802.15.4, όπως ορίζονται στο [1]. Στα πλαίσια αυτής της εργασίας έγινε προσπάθεια να γίνει όσο το δυνατό μεγαλύτερη σύγκλιση, του δικτύου μας με το πρότυπο, ωστόσο μερικά σημεία του υλικού μας δεν μας παρέχουν την δυνατότητα αυτή και σε άλλα σημεία η εφαρμογή μας απαιτούσε άλλες ρυθμίσεις.

Στο [1] ορίζονται κάποια συγκεκριμένα χαρακτηριστικά του φυσικού επιπέδου για τις μπάντες συχνοτήτων. Στον πίνακα 1 γίνεται μια σύγκριση σε τεχνικά χαρακτηριστικά, για την μπάντα συχνοτήτων που χρησιμοποιεί το δικό μας WSN, σε σχέση με όσα ορίζονται στο πρότυπο.

Τα κύρια χαρακτηριστικά που προβλέπει το πρότυπο 802.15.4 για το φυσικό του επίπεδο (PHY) είναι:

- Δυνατότητα ενεργοποίησης και απενεργοποίησης του ραδιοφώνου.
- Ανίχνευση της ενέργειας του σήματος στο τρέχον κανάλι, ED (Energy Detection).

- Ακριβής έλεγχος του καναλιού για την αποφυγή συγκρούσεων σε ταυτόχρονη μετάδοση, CCA (Clear Channel Assessment).
- Δυνατότητα επιλογής συχνότητας.
- Δυνατότητα ελέγχου ορθότητας των πακέτων που λαμβάνονται και αποφυγή λαθών, CRC (cyclic redundancy check).
- Αποστολή και λήψη πληροφορίας.

Αυτές τις δυνατότητες και λειτουργίες καλύπτει και το δικό μας ραδιόφωνο. Ορίζονται ακόμη και τα χαρακτηριστικά της διαμόρφωσης και ρυθμού μετάδοσης, για συγκεκριμένες μπάντες συχνοτήτων. Το δικό μας ραδιόφωνο λειτουργεί γύρω από την συχνότητα 2.4GHz, οπότε θα εστιάσουμε εκεί. Βασική διαφορά είναι ότι το υλικό μας (ραδιόφωνο CC2500), δεν μας

	IEEE Std-802.15.4	iCubes
Band	2400-2483 (MHz)	2400-2483 (MHz)
Modulation	O-QPSK	OOK, 2-FSK, GFSK, MSK
Data Rate	62.5 kB/s	2.4 - 500 kB/s

Πίνακας 1: Χαρακτηριστικά Φυσικού Επιπέδου - (Physical Layer)

παρέχει την δυνατότητα να έχουμε διαμόρφωση τύπου O-QPSK, οπότε σε σχέση με τις υπόλοιπες επιλογές, θα έπρεπε να διαλέξουμε μια διαμόρφωση ή οποία να ταιριάζει στις εκάστοτε απαιτήσεις της εφαρμογής μας.

Στην τωρινή εφαρμογή του δικτύου αισθητήρων δεν απαιτείται υψηλός ρυθμός αποστολής, καθώς η πληροφορία είναι σποραδική και μικρή σε μέγεθος (16 bit για κάθε μέτρηση είναι αρκετά), αλλά μας ενδιαφέρει η κάλυψη μεγάλης απόστασης. Για αυτό το σκοπό χρειαζόμαστε εκείνες τις ρυθμίσεις του CC2500, για τις οποίες επιτυγχάνεται μέγιστη ευαισθησία και υψηλές αποστάσεις.

Για το CC2500 αυτές οι ρυθμίσεις είναι οι παρακάτω, όπως ορίζονται στο εγχειρίδιο του.

Parameter	Min	Typ	Max	Unit	Condition/Note
Digital channel filter bandwidth	58		812	kHz	User programmable. The bandwidth limits are proportional to crystal frequency (given values assume a 26.0 MHz crystal).
2.4 kBaud data rate, sensitivity optimized, `MDMCFG2.DEM_DCFILT_OFF=0` (2-FSK, 1% packet error rate, 20 bytes packet length, 203 kHz digital channel filter bandwidth)					
Receiver sensitivity		–104		dBm	The RX current consumption can be reduced by approximately 1.7 mA by setting `MDMCFG2.DEM_DCFILT_OFF=1`. The typical sensitivity is then -102 dBm and the temperature range is from 0°C to +85°C. The sensitivity can be improved to typically –106 dBm with `MDMCFG2.DEM_DCFILT_OFF=0` by programming registers `TEST2` and `TEST1` (see page 82). The temperature range is then from 0°C to +85°C.
Saturation		–13		dBm	
Adjacent channel rejection		23		dB	Desired channel 3 dB above the sensitivity limit. 250 kHz channel spacing
Alternate channel rejection		31		dB	Desired channel 3 dB above the sensitivity limit. 250 kHz channel spacing
					See Figure 22 for plot of selectivity versus frequency offset
Blocking					Wanted signal 3 dB above sensitivity level.
±10 MHz offset		64		dBm	Compliant with ETSI EN 300 440 class 2 receiver requirements.
±20 MHz offset		70		dBm	
±50 MHz offset		71		dBm	

Σχήμα 6: Απόσπασμα από τις πρότυπες ρυθμίσεις του CC2500 στο [3]

Επιλέχθηκε δηλαδή διαμόρφωση 2-FSK και ρυθμός μετάδοσης στα 2.4Kbps

4 Πρόσβαση στο Μέσο - CSMA/CA

4.1 Περιγραφή

Η πολλαπλή πρόσβαση στο μέσο (MAC - Medium Access Control) απαιτεί κάποιο πρωτόκολλο, έτσι ώστε να αποφεύγονται οι συγκρούσεις κόμβων που μεταδίδουν στην ίδια μπάντα συχνοτήτων. Στο πρότυπο 802.15.4 όπως ορίζεται στο [1], αυτός ο μηχανισμός είναι ο CSMA-CA (carrier sense multiple access - collision avoidance), όπου προτείνονται δύο εκδοχές του, μία για PANs με beacon-enabled δομή και μία για non-beaconed δίκτυα.

Ο μηχανισμός αυτός ανήκει στην κατηγορία listen-before-talk πρωτοκόλλων. Η λογική του αλγορίθμου στηρίζεται στην ικανότητα του κάθε κόμβου, να μπορεί να ακούει πιθανές μεταδόσεις που βρίσκονται σε εξέλιξη (CCA - clear channel assessment) όταν αυτός θέλει να μεταδώσει ένα πακέτο και να αναβάλει την προσπάθεια του, για τυχαία χρονικά διαστήματα (backoff slots), μέχρι να βρει κάποια στιγμή το κανάλι αδρανές, για να μεταδώσει το πακέτο του.

Το πρότυπο ορίζει δύο πιθανές επιλογές για την μορφή του CSMA-CA αλγορίθμου:

- Η εκδοχή με σχισμές (slotted CSMA-CA), η οποία χρησιμοποιείται όταν στο δίκτυο μας χρησιμοποιούνται beacons.
- Η εκδοχή χωρίς σχισμές (unslotted CSMA-CA), η οποία χρησιμοποιείται όταν στο δίκτυο δεν χρησιμοποιούνται beacons. Σε κάθε περίπτωση ο CSMA-CA αλγόριθμος υλοποιείται και στηρίζεται στη μονάδα χρόνου backoff period, όπου ορίζεται στο [1] να είναι ίση με μία σταθερά aUnitBackoffPeriod (20 περίοδοι συμβόλου).

Και στις δύο περιπτώσεις το CSMA-CA είναι τύπου non-persistent, δεν συνεχίζει δηλαδή την διαδικασία ελέγχου του καναλιού διαρκώς, μέχρι να το βρει κάποια στιγμή ελεύθερο, παρά αναμένει για περιόδους backoff slots όπως προαναφέρθηκε.

Τα PANs με beacons-enabled απαιτούν από το επίπεδο MAC, να υπάρχει στο δίκτυο μία σαφώς ορισμένη και σταθερή οριοθέτηση των χρονοθυρίδων (time slots), κοινή για όλους τους κόμβους του δικτύου, όπως αυτή που φαίνεται στο αντίστοιχο σχήμα 7. Την οποία θα

φροντίζει ένας κόμβος στο δίκτυο να συγχρονίζει και να κρατάει σταθερή (PAN - coordinator).

Τότε θα μπορεί να ορίζει μια δομή από περιόδους εκπομπής και μη-εκπομπής (superframe structure) για τους υπόλοιπους κόμβους, οι οποίοι συγχρονισμένοι θα πρέπει να τηρούν τα περιθώρια αυτά, να επιδιώκουν μεταδόσεις ή να παραμένουν αδρανείς μέσα σε αυτά τα όρια και οι σχισμές των backoff slots να βρίσκονται σε **πλήρη ευθυγράμμιση** με τις σχισμές των superframes.

Σε αντίθεση με αυτή την εκδοχή, όταν δεν χρησιμοποιούνται beacons και έχουμε την μορφή του unslotted-CSMA αλγορίθμου, τα backoff slots του κάθε κόμβου **δεν** συσχετίζονται στο χρόνο, με τα backoff slots των άλλων κόμβων του δικτύου.

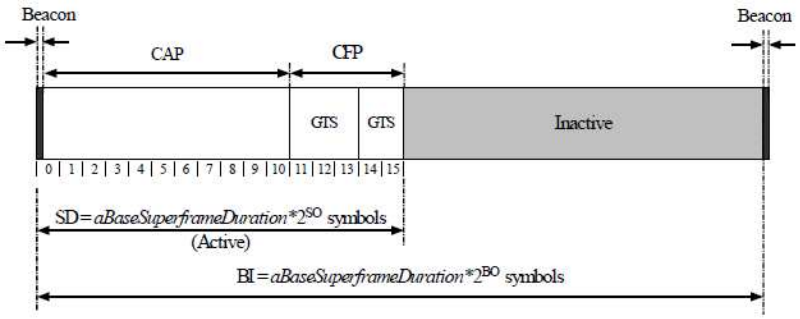


Figure 66—An example of the superframe structure

Σχήμα 7: Δομή ενός superframe όπως ορίζεται στο [1]

Στα πλαίσια αυτής της εργασίας αυτός ο συγχρονισμός δεν επιδιώχθηκε και για αυτό το λόγο χρησιμοποιήθηκε η εκδοχή του unslotted CSMA-CA αλγορίθμου, με βάση όσα ορίζονται από το πρότυπο, για μια απλούστερη λύση στο δίκτυο αισθητήρων του χωραφιού. Στον unslotted CSMA-CA αλγόριθμο, τα βήματα του οποίου αναλύονται στο παρακάτω σχήμα, για κάθε μετάδοση ενός πακέτου, κάθε κόμβος κρατάει δύο μεταβλητές, σε κάθε προσπάθεια μετάδοσης.

- NB: Είναι ο αριθμός των φορών που ο κόμβος αναγκάστηκε να αναβάλει την μετάδοση του.

- BE : Ο αριθμός του εκθέτη της εκθετικής υποχώρησης (exponential backoff) από τον

οποίο εξαρτάται και ο αριθμός των backoff σχισμών, που κάθε κόμβος θα περιμένει, εάν βρει το κανάλι απασχολημένο. Στην αρχή κάθε ξεχωριστής μετάδοσης πακέτου, αυτός ο αριθμός έχει την τιμή macMinBE (η οποία για το πρότυπο είναι ίση με 3 σαν default τιμή). Εάν δοθεί τιμή 0, σαν αρχική τιμή, για το BE, ουσιαστικά η αποφυγή σύγκρουσης ακυρώνεται, για την πρώτη επανάληψη του αλγορίθμου.

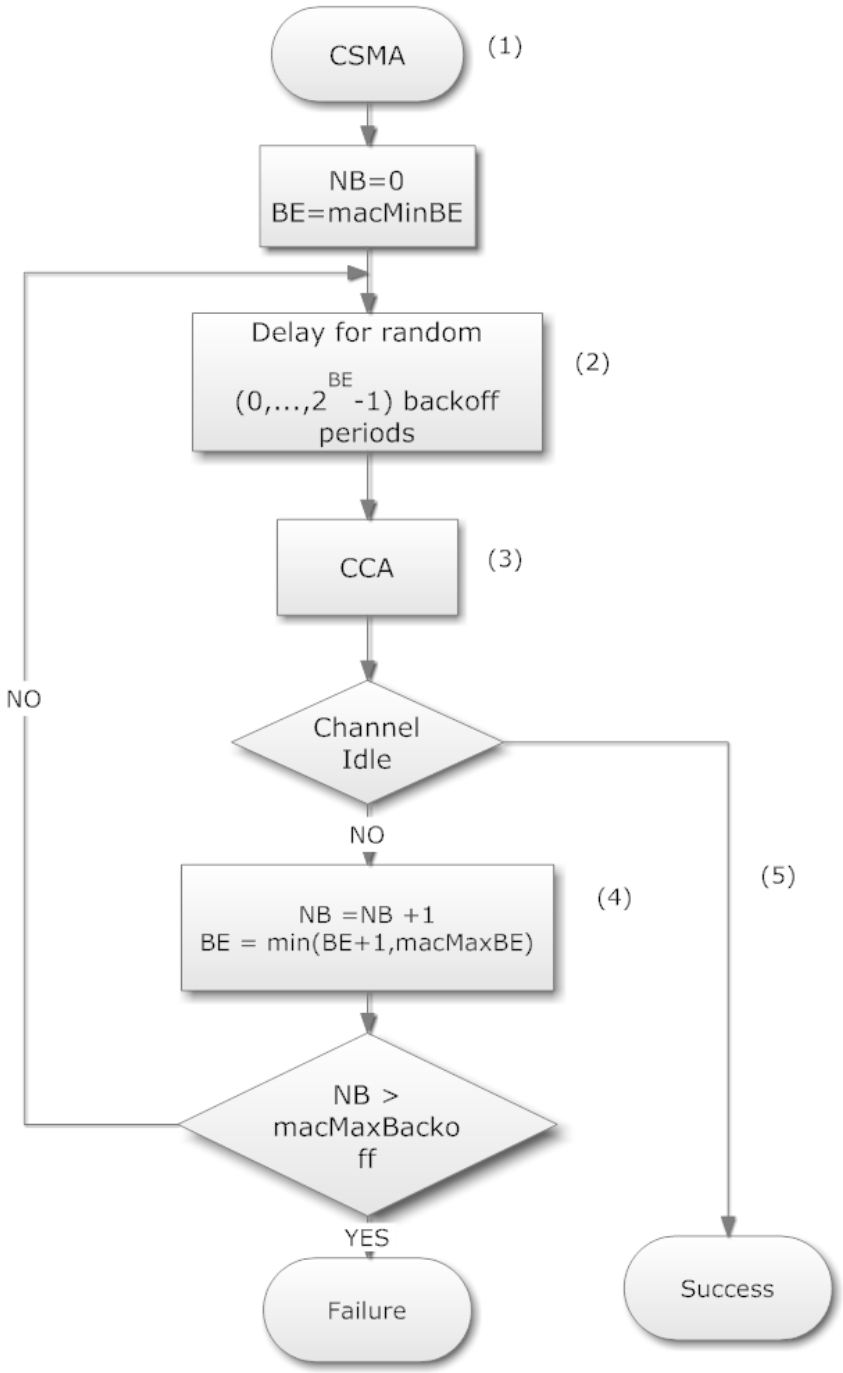
Αρχικά ο αλγόριθμος για το unslotted CSMA-CA αρχικοποιεί τις μεταβλητές NB και BE και προχωράει στο βήμα (2). Στο βήμα (2) ο αλγόριθμος υποχρεώνει τον κάθε κόμβο σε αναμονή για τυχαίο αριθμό backoff slots,τον οποίο αριθμό επιλέγει από ένα διάστημα ακέραιων τιμών ομοιόμορφα κατανεμημένων στο $[0, 2^{BE} - 1]$.

Στην συνέχεια προχωράει στο βήμα (3) όπου ζητάει από το φυσικό επίπεδο (PHY) να εφαρμόσει έλεγχο διαθεσιμότητας του καναλιού (CCA - clear channel assessment). Εάν το κανάλι κριθεί ως απασχολημένο, τότε το MAC επίπεδο αυξάνει τους αριθμούς NB και BE κατά 1 μονάδα, στο βήμα (4), τσεκάροντας κάθε φορά ότι το BE δεν θα ξεπεράσει την τιμή macMaxBE.

Εάν η τιμή NB ξεπεράσει την προγραμματισμένη τιμή macMaxCSMABackoffs, τότε ο αλγόριθμος τερματίζει και δηλώνει αποτυχία πρόσβασης στο μέσο. Εάν το κανάλι βρεθεί σε κατάσταση αδρανή, τότε ο κόμβος προχωρεί σε μετάδοση στο βήμα (5) και δηλώνει επιτυχία.

Όπως ορίζεται στο πρότυπο, η χρήση επιβεβαιώσεων (ACKs - acknowledgements) για τα πακέτα δεδομένων (data frames), είναι προαιρετική. Θα γίνεται από τον PAN - coordinator προς τους κόμβους και δεν θα χρησιμοποιεί τον αλγόριθμο CSMA-CA, για τον μηχανισμό πρόσβασης στο μέσο.

Στην υλοποίηση της εργασίας αυτής, συμπεριλαμβάνεται ο μηχανισμός των επιβεβαιώσεων (ACKs) για τα πακέτα δεδομένων, πάνω στην ίδια λογική, ότι δηλαδή κάθε επιβεβαίωση γίνεται έπειτα από την λήψη πακέτου, από τον παραλήπτη, κυρίως στην περίπτωση που αυτός είναι ο σταθμός βάσης, αντίστοιχος με τον PAN - coordinator του προτύπου δηλαδή. Το διάγραμμα που ακολουθεί δείχνει την ροή του αλγορίθμου.



Σχήμα 8: Δομή του unslotted CSMA-CA όπως ορίζεται στο [1]

Επίσης στα πλαίσια της εργασίας αυτής υλοποιήθηκε και η απαραίτητη λογική για την αποφυγή του προβλήματος του κρυμμένου κόμβου(hidden node problem). Στη περίπτωση

αυτού του προβλήματος ένας ή περισσότεροι κόμβοι, δεν μπορούν να ακούσουν τους γείτονες τους, λόγω απόστασης ή γενικότερα άλλων συνθηκών του περιβάλλοντος. Ως εκ τούτου κατά την περίοδο ελέγχου του καναλιού, αυτοί οι κόμβοι δεν μπορούν να ακούσουν ο ένας, τη μετάδοση του άλλου, οπότε θεωρούν λανθασμένα το κανάλι ελεύθερο.

Η λογική που ακολουθήθηκε είναι αυτή των RTS/CTS πακέτων για την αποφυγή τέτοιων συγκρούσεων και η χρήση των επιβεβαιώσεων. Δεν θα επεκταθούμε όμως στην ανάλυση τους παρακάτω, καθώς είναι προαιρετικές, όπως και το πρότυπο 802.15.4 ορίζει.

4.2 Θεωρητική Ανάλυση

4.2.1 Εισαγωγή - Παραδοχές

Η θεωρητική σκοπιά που περιγράφεται παρακάτω έχει παρουσιαστεί και αναλυθεί στο [2]. Αρχικά τονίζεται ότι ο αλγόριθμος του unslotted CSMA-CA πρωτοκόλλου, που υλοποιήθηκε(και βασίζεται στο 802.15.4), διαφέρει σημαντικά από άλλα αντίστοιχα πρότυπα, όπως αυτό του 802.11. Η βασική διαφορά είναι ότι στο 802.11, ο μετρητής της δυαδικής εκθετικής υποχώρησης μικραίνει, όσο το κανάλι βρίσκεται αδρανές και παγώνει κάθε φορά που βρίσκεται μία μετάδοση. Παρακολουθεί δηλαδή το κανάλι κάθε κόμβος συνεχώς. Στο 802.15.4 πρότυπο, ο κόμβος δεν παρακολουθεί συνεχώς το κανάλι, ταυτόχρονα δηλαδή με την περίοδο που υποχωρεί (backoff), παρά ελέγχει ξανά το κανάλι στο τέλος αυτής της περιόδου.

Το μαθηματικό μοντέλο στο [2], διαφέρει σε μεγάλο βαθμό από αντίστοιχα μοντέλα. Για την ανάλυση του, κάνει τις εξής βασικές παραδοχές:

- Στο δίκτυο υπάρχουν N κόμβοι όπου ο κάθε κόμβος μεταδίδει ένα πακέτο $D \times 10$ *bytes*. Ο χρόνος που απαιτείται να αποσταλεί ένα τέτοιο πακέτο είναι $D \times d\_b$ όπου $d\_b = 1 \times$ backoff period.

- Θεωρούνται ιδανικές συνθήκες καναλιού: Όλοι οι κόμβοι «ακούνε» ο ένας τον άλλο, δεν υπάρχει δηλαδή πρόβλημα κρυφών κόμβων (hidden node problem). Συγκρούσεις μεταξύ κόμβων μπορεί να συμβούν στην περίπτωση που δύο ή περισσότεροι κόμβοι ξεκινήσουν ταυτόχρονα τον έλεγχο καναλιού προς διαθεσιμότητα καναλιού (CCA). Ε-

πίσης δεν θεωρείται ενεργοποιημένος, κάποιος μηχανισμός επιβεβαιώσεων (ACK) και αναμετάδοσης.

- Όλοι οι κόμβοι ξεκινούν τον αλγόριθμο πρόσβασης στο μέσο την ίδια στιγμή, όταν στέλνει ο κόμβος βάσης το σήμα, για την έναρξη αποστολής μετρήσεων (δεν υπάρχει δηλαδή μεταβλητότητα καθυστέρησης, στην διάδοση πακέτων και οι κόμβοι λαμβάνουν ταυτόχρονα την εντολή από τον κόμβο βάσης για την έναρξη της αποστολής). Επίσης κάθε κόμβος σε κάθε γύρο προσπαθεί να μεταδώσει , ένα πακέτο ακριβώς, όπως στις περισσότερες περιπτώσεις των WSN σεναρίων. Έτσι ο αριθμός των κόμβων που ανταγωνίζονται για την πρόσβαση στο μέσο, μειώνεται καθώς αυξάνεται ο χρόνος σε έναν γύρο.

Κάθε κόμβος, ο οποίος επιχειρεί πρόσβαση στο μέσο, σε κάθε γύρο, μπορεί να βρίσκεται σε τέσσερεις πιθανές καταστάσεις: κατάσταση υποχώρησης (backoff), κατάσταση ελέγχου καναλιού (CCA), κατάσταση μετάδοσης (transmit) και κατάσταση αδράνειας (idle). Εάν μετά από την φάση του ελέγχου καναλιού (CCA), το κανάλι βρεθεί ελεύθερο τότε ο κόμβος προχωρά σε μετάδοση και στην συνέχεια παραμένει αδρανής ως το τέλος του γύρου. Οι στόχοι λοιπόν του θεωρητικού μοντέλου στο [2] είναι να μοντελοποιηθεί η κατάσταση του ελέγχου του καναλιού και της υποχώρησης (backoff).

Οι καταστάσεις αυτές μοντελοποιούνται από μία διαδικασία Q δύο διαστάσεων $Q(t) = \{BO_c(t), BO_s(t)\}$, όπου t είναι ακέραιος αριθμός που αναπαριστά τον χρόνο σε σχισμές. Μία σχισμή είναι ίση με μία backoff period. Έτσι τα $BO_c(t)$ και $BO_s(t)$ αναπαριστούν τον backoff μετρητή και το backoff επίπεδο(δηλαδή πόσες φορές έχει γίνει backoff) αντίστοιχα.

Και οι δύο είναι διακριτού-χρόνου στοχαστικές διαδικασίες, θεωρώντας διακριτές τιμές. Οπότε η διαδικασία Q είναι μια αλυσίδα. Δεν είναι όμως μία Markovian αλυσίδα, αφού η $BO_c(t)$ δεν είναι μία χωρίς μνήμη διαδικασία, αλλά η τιμή της εξαρτάται από πόσες φορές ο κόμβος, έχει προσπαθήσει να κάνει πρόσβαση στο κανάλι χωρίς επιτυχία.

Η αρχική τιμή του backoff μετρητή ($BO_c(0)$) είναι ομοιόμορφα κατανεμημένη, στο διάστημα $[0, W_{NB}-1]$, όπου $W_{NB} = 2^{BE}$ είναι το μέγεθος του παραθύρου ανταγωνισμού (contention

window) και NB $[0, NB_{max}]$. Η τιμή του BE εξαρτάται από την δεύτερη διαδικασία $BO_s(t)$.

Οπότε ορίζονται $NB_{max}+1$ διαφορετικά στάδια υποχώρησης (backoff stages) παρατηρώντας τα διαφορετικά ζευγάρια τιμών (NB, BE). Το πρότυπο 802.15.4 όπως ορίζεται στο [1] ορίζει ότι οι βασικές τιμές των NB και BE θα αρχικοποιούνται στην αρχή του αλγορίθμου, σε $NB^0 = 0 kai BE^0 = 3$ (αντιστοιχεί σε παράθυρο $W_0 = 8$). Κάθε φορά που το κανάλι βρίσκεται κατειλημμένο θα αυξάνονται αυτές οι δύο τιμές κατά μια μονάδα. Η περίπτωση $BO_s = 4$, είναι η τελευταία, στις βασικές ρυθμίσεις του 802.15.4.

Επειδή υπάρχει μία μέγιστη τιμή για το NB, υπάρχει και μια μέγιστη καθυστέρηση που μπορεί να έχει ένα πακέτο στη μετάδοση του. Αυτό το μέγιστο επιτυγχάνεται, στην περίπτωση που κάποιος κόμβος εξαντλήσει όλες τις προσπάθειες του, για πρόσβαση στο κανάλι και φτάσει το υψηλότερο backoff stage. Σε αυτή την περίπτωση, θα ισχύει ότι ο κόμβος θα βρίσκεται σε κατάσταση backoff για αριθμό $\sum_{k=0}^{NB_{max}} (W_k - 1)$ από slots και για αριθμό $NB + 1$ slots σε κατάσταση ελέγχου του καναλιού.

Έτσι η τελευταία σχισμή, που θα μπορεί μια μετάδοση να συμβεί, είναι η $t_{max} = \sum_{k=0}^{NB_{max}} W_k = 120$, και η τελευταία μετάδοση θα τελειώσει σε $(tmax + D - 1)$. Ο έλεγχος του καναλιού είναι όμως εφικτός, για το διάστημα χρόνου $[0, tmax - 1]$.

4.2.2 Μαθηματικό μοντέλο - Μέρος 1

Στα παρακάτω, όπως ορίζονται στο [2], θα ορίζεται η γενική κατάσταση της αλυσίδας ως $Q(t) = \{BOc, BOs, t\}$ και η πιθανότητα να βρίσκεται κάποιος κόμβος σε μια κατάσταση ως $P\{BO_c = c, BO_s = i, t = j\} = P\{c, i, j\}$.

Ορίζεται επίσης η πιθανότητα p_b^j ότι στην j-th slot το κανάλι βρέθηκε κατειλημμένο, έπειτα από τον έλεγχο. Το μοντέλο εξετάζει τις εξής πιθανότητες:

- Την πιθανότητα ένας κόμβος να λήξει την μετάδοση του πακέτου του σε ένα slot j, η οποία θα ορίζεται ως $P\{T^j\}, j \in [0, t_{max} + D - 1]$.

- Την πιθανότητα επιτυχούς μετάδοσης ενός πακέτου, δηλαδή την πιθανότητα επιτυχίας της μετάδοσης σε οποιαδήποτε σχισμή, για ένα γύρο προσπαθειών, η οποία θα ορίζεται

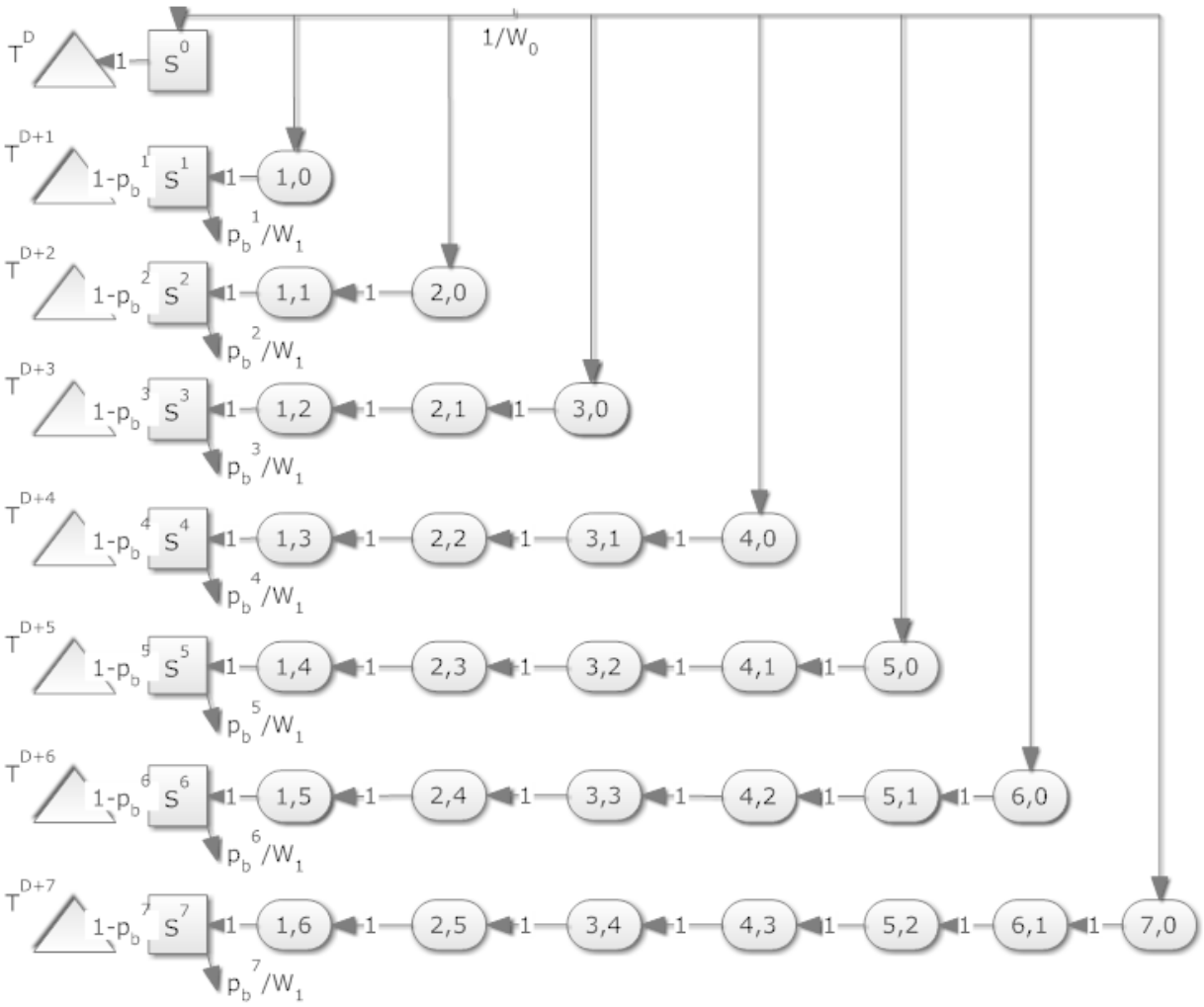
ως p_s.

- Την πιθανότητα ο κόμβος βάσης να λάβει την ουρά ενός πακέτου, προερχόμενη από οποιονδήποτε κόμβο, σε μια σχισμή j, η οποία θα ορίζεται ως
  $P\{R^j\}, j \in [0, t_{max} + D - 1]$.

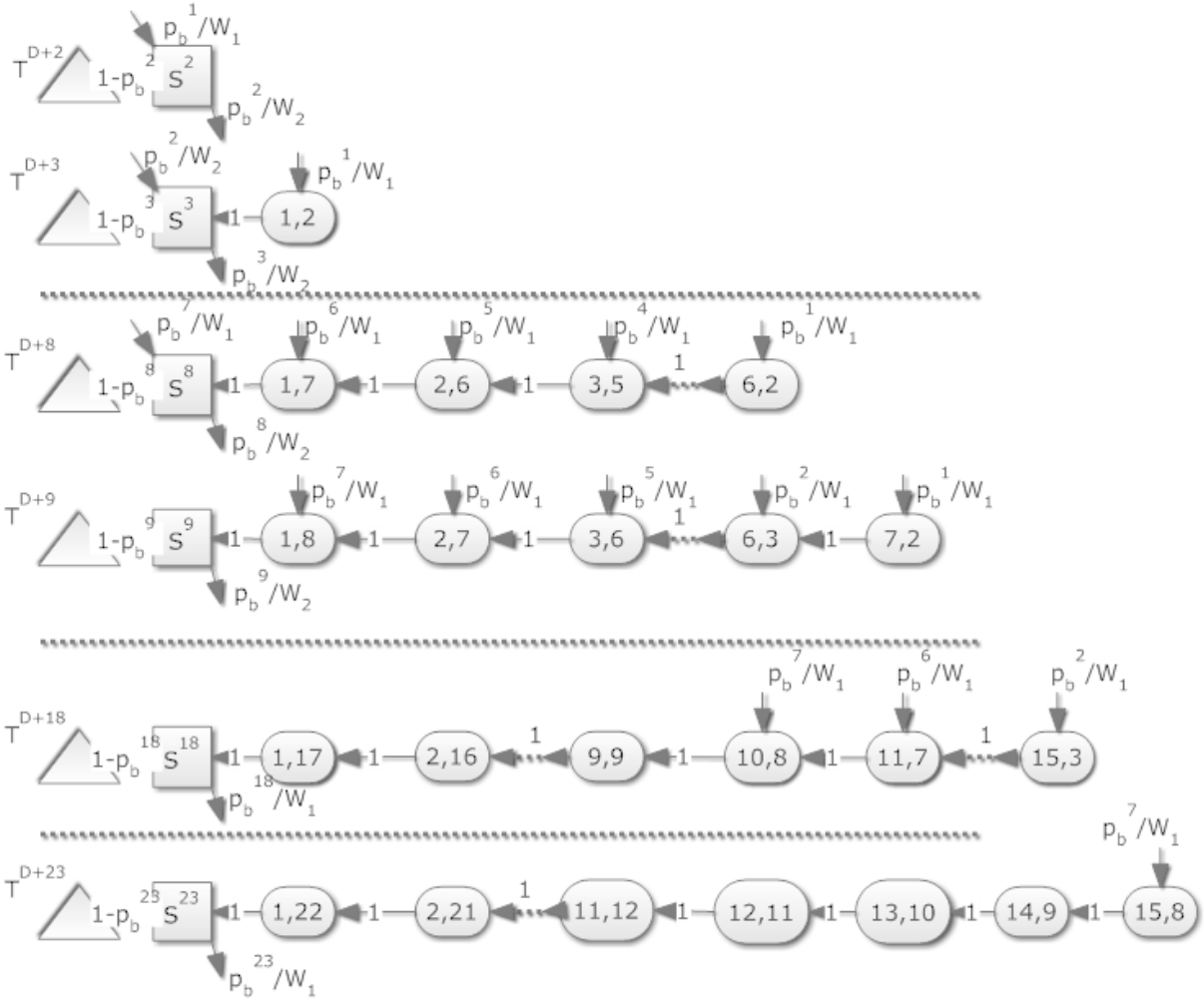
Η πιθανότητα $P\{T^j\}$ εξαρτάται από την πιθανότητα να ήταν κάποιος κόμβος την σχισμή $j-D$ σε κατάσταση ελέγχου-καναλιού (CCA). Γιατί ένα πακέτο όπως αναφέρθηκε καταλαμβάνει D σχισμές και ένας κόμβος που βρίσκει το κανάλι ελεύθερο την σχισμή $j-D$, θα ολοκληρώσει την μετάδοση του την σχισμή j.

Η πιθανότητα να βρίσκεται ένας κόμβος σε κατάσταση ελέγχου του καναλιού, την σχισμή j στο στάδιο υποχώρησης $i(BOs = i)$, ορίζεται στο [2] ως $P\{S_i^j\} = P\{0, i, j\}$, για i και j. Από αυτές τις πιθανότητες υπολογίζεται και η πιθανότητα, να βρίσκεται ένα κόμβος στην κατάσταση ελέγχου του καναλιού την σχισμή j, η οποία ορίζεται $P\{C^j\}, j \in [0, t_{max} - 1]$ και η πιθανότητα $P\{T^j\}$. Οι πιθανότητες p_s και $P\{R^j\}$, εξαρτώνται από την πιθανότητα $P\{T^j\}$ και τέλος υπολογίζεται και η πιθανότητα p_b^j.

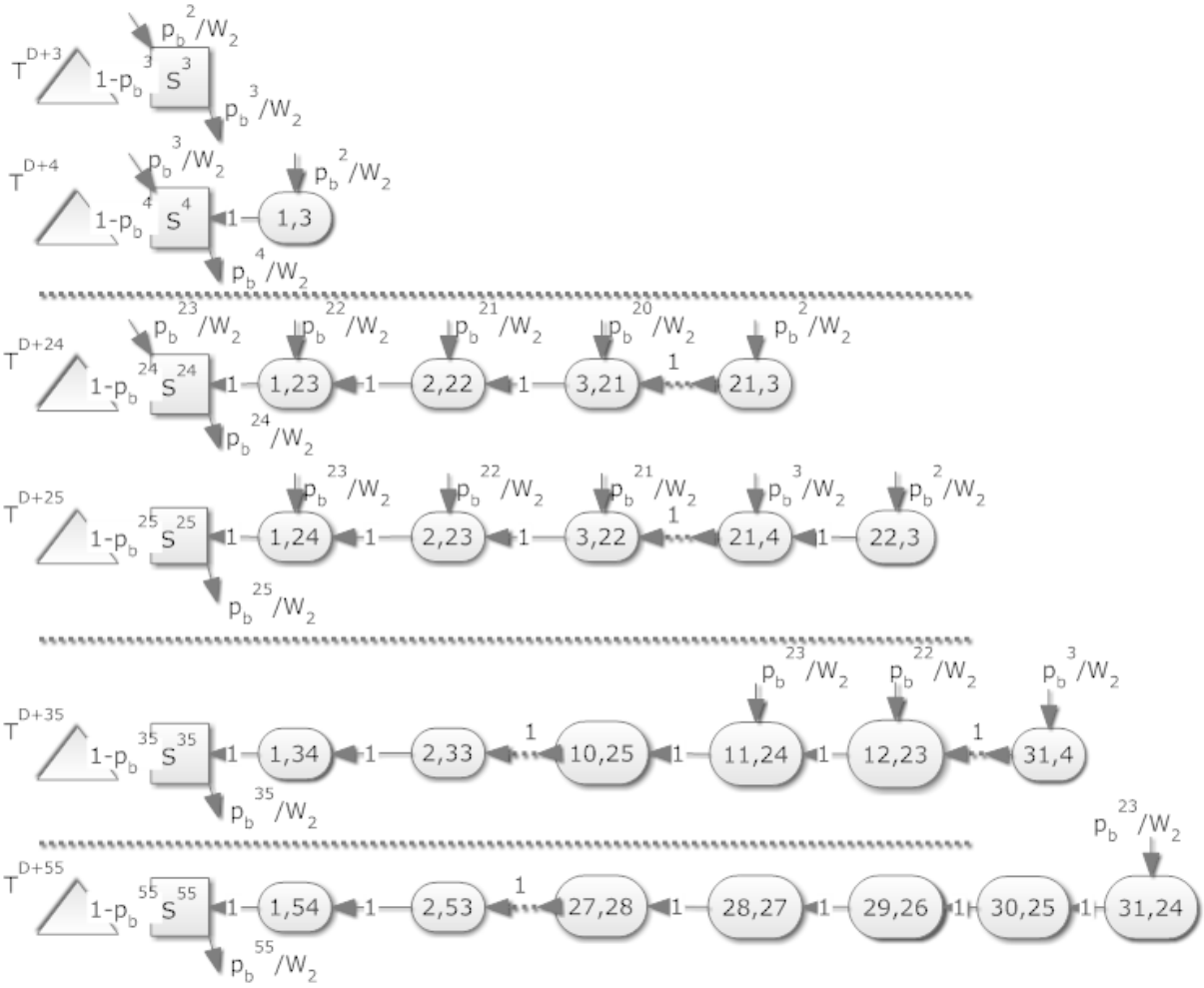
Το διάγραμμα μετάβασης καταστάσεων όπως αναφέρεται στο [2] φαίνεται παρακάτω στα σχήματα.



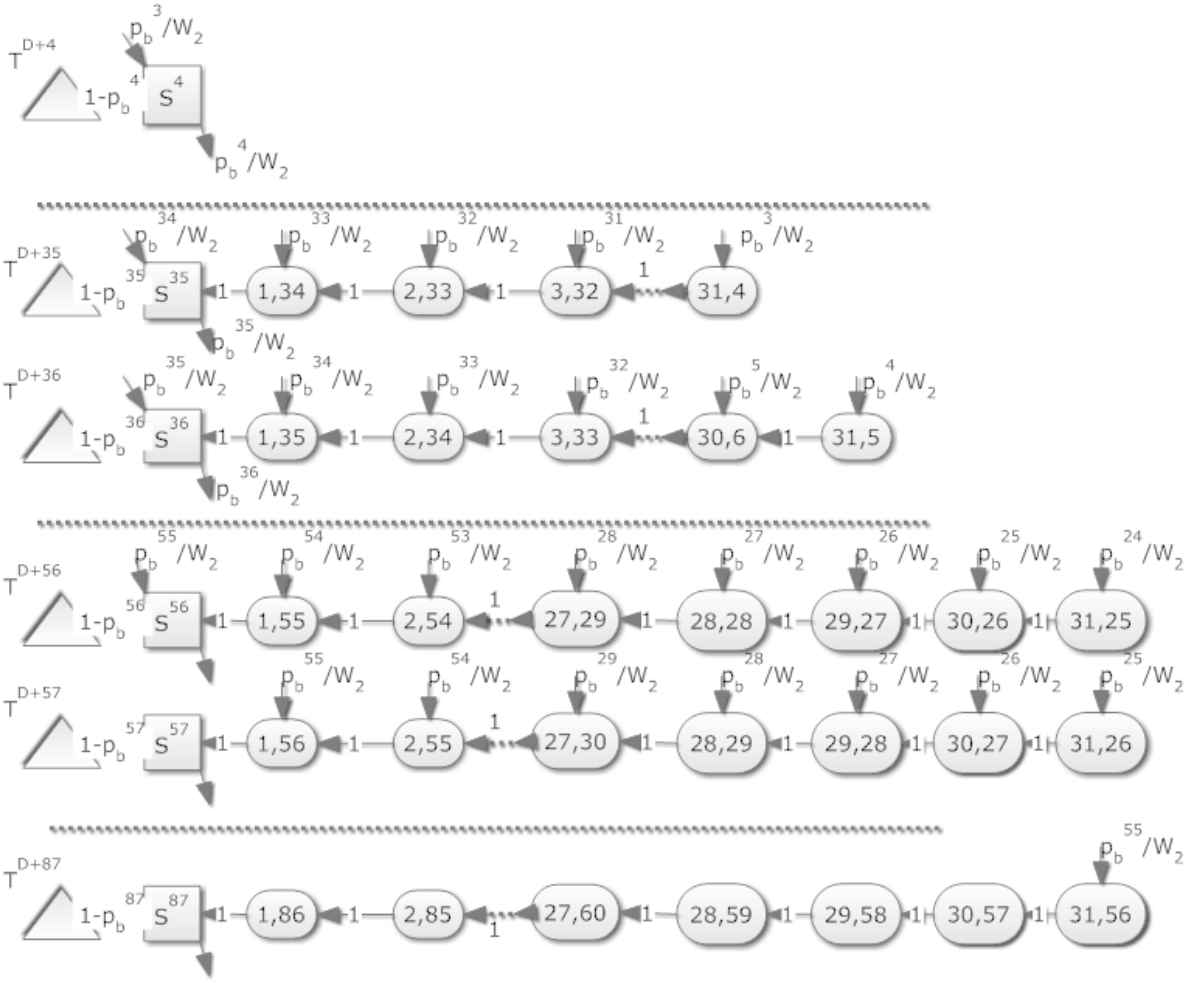
Σχήμα 9: Απόσπασμα από [2, Fig 2] για backoff stage 1



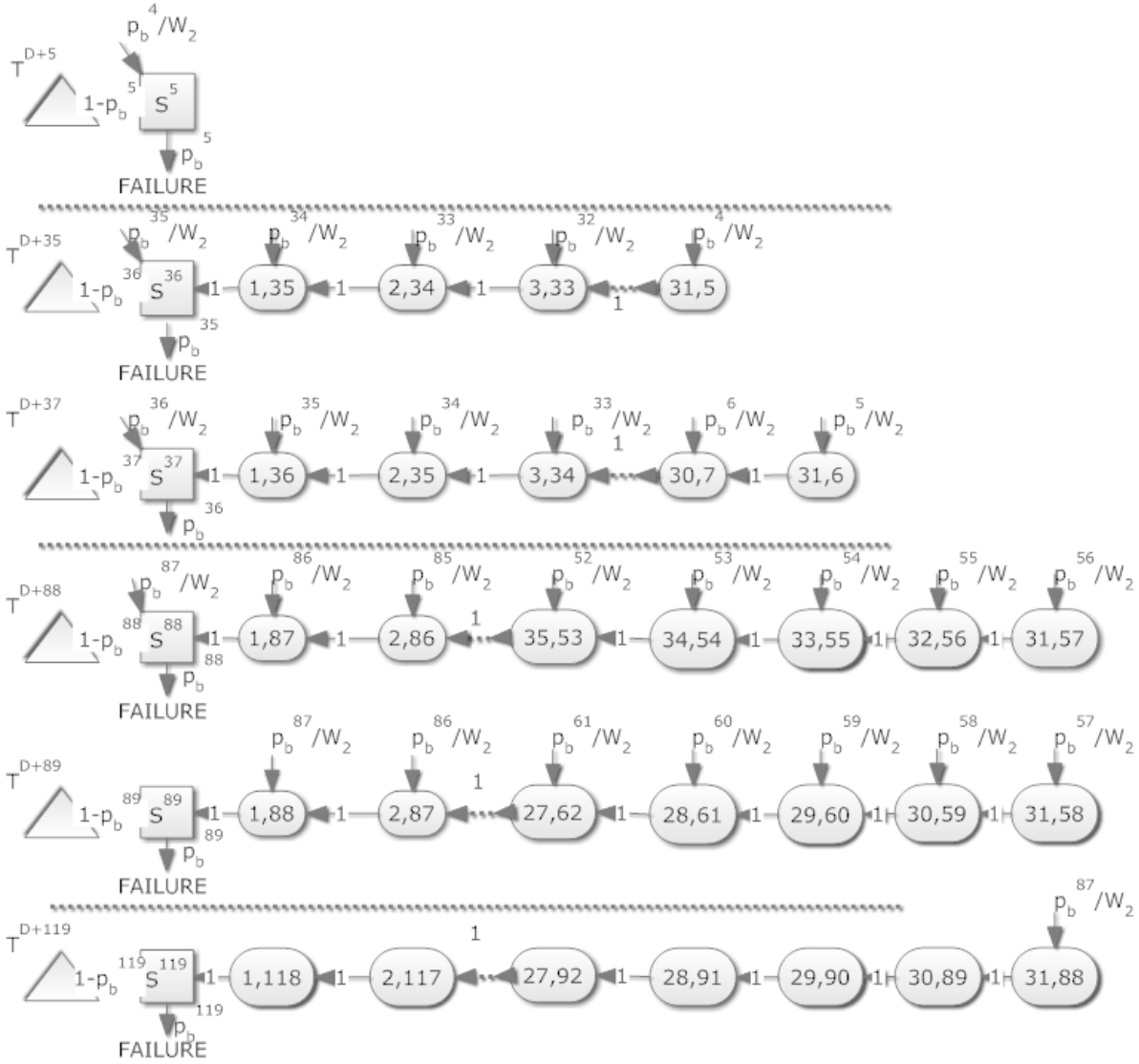
Σχήμα 10: Απόσπασμα από [2, Fig 3] για backoff stage 2



Σχήμα 11: Απόσπασμα από [2, Fig 4] για backoff stage 3



Σχήμα 12: Απόσπασμα από [2, Fig 5] για backoff stage 4



Σχήμα 13: Απόσπασμα από [2, Fig 6] για backoff stage 5

Από το διάγραμμα των καταστάσεων εξάγουμε τα εξής:

- **Πρώτο Στάδιο Υποχώρησης (First Backoff Stage)**:

  Στην αρχή κάθε κόμβος διαλέγει έναν αριθμό από ένα ομοιόμορφα κατανεμημένο σύνολο, στο διάστημα $\{0, W_0 - 1 = 7\}$. Στο χρόνο t=0, ένας κόμβος επιλέγει με πιθανότητα $1/W_0$, μια από τις καταστάσεις $\{c, 0, 0\}, c \in [0, 7]$. Εάν ο αριθμός επιλογής του είναι το 0, τότε ο κόμβος στην σχισμή 0, θα ελέγξει το κανάλι και στην σχισμή 1, θα μετα-

δώσει το πακέτο του. Αυτό γιατί δεν μπορεί να υπάρξει μετάδοση, στην πρώτη σχισμή ($P\{T^0 = 0\}$), και έτσι το κανάλι εκείνη τη σχισμή πάντα θα βρίσκεται αδρανές ($p_b^0 = 0$).

Σε περίπτωση που επιλέξει αριθμό μεγαλύτερο του 0 τότε, ο κόμβος θα μειώνει το μετρητή υποχώρησης σε κάθε backoff slot μέχρι την τιμή 0, όπου ο κόμβος θα κάνει έλεγχο του καναλιού. Μετά την περίοδο ελέγχου, ο κόμβος θα μεταδώσει αν το κανάλι βρέθηκε καθαρό αλλιώς θα περάσει στο επόμενο backoff stage, και μια άλλη τιμή από το διάστημα $\{0, W_1 - 1 = 15\}$ θα επιλεχθεί.

Ορίζοντας ως $P\{BO_c = c_1, BO_s = i_1, t = j_1 | BO_c = c_0, BO_s = i_0, t = j_0\} = P\{c_1, i_1, j_1 | c_0, i_0, j_0\}$ την πιθανότητα μεταβολής της κατάστασης μας από την $\{c_0, i_0, j_0\}$ κατάσταση, στην $\{c_1, i_1, j_1\}$, οι πιθανότητες μετάβασης είναι

$$P\{c, 0, j + 1 | c + 1, 0, j\} = 1 \tag{1}$$

για $c \in [0, W_0 - 2]$ και $j \in [0, W_0 - 2]$. Αυτή η σχέση δείχνει ότι, στην αρχή κάθε σχισμής, ο backoff counter, φθίνει κατά 1 μονάδα μέχρι να φτάσει το 0, με πιθανότητα 1. Οι πιθανότητες να βρίσκεται ένας κόμβος στην κατάσταση ελέγχου του καναλιού είναι:

$$P\{S_0^j\} = \begin{cases} \frac{1}{W_0}, & \text{για } \ j \in [0, W_0 - 1] \\ 0, & \text{για } \ j > W_0 - 1 \end{cases} \tag{2}$$

- **Δεύτερο Στάδιο Υποχώρησης (Second Backoff Stage)**: Επειδή σε αυτό το στάδιο ένας κόμβος, δεν μπορεί να φτάσει πριν την σχισμή $t = 2$, γι΄ αυτό και στο διάγραμμα του σχήματος 10, δεν υπάρχουν τα S_1^0 και S_1^1, όπως και τα T αντίστοιχα. Όπως και στην προηγούμενη περίπτωση, οι πιθανότητες μετάβασης δίνονται από την σχέση:

$$P\{c, 1, j + 1 | c + 1, 1, j\} = 1 \tag{3}$$

για $c \in [0, W_1 - 2]$ και $j \in [2, W_{0,1} - 2]$ όπου $W_{0,1} = W_0 + W_1$. Παρακάτω θα αναφέρεται το $W_{x,y,z}$, το οποίο ορίζεται έως το άθροισμα $W_x + W_y + W_z$. Οι πιθανότητες μετάβασης για τις καταστάσεις του πρώτου σταδίου ($BO_s = 0$) και του δεύτερου σταδίου

$(BO_s = 1)$ δίνονται από την σχέση:

$$P\{c, 1, j+1 | 0, 0, j\} = \frac{p_b^j}{W_1} \tag{4}$$

για $c \in [0, W_1 - 1]$ και $j \in [1, W_0 - 1]$. Αυτή η σχέση δείχνει ότι, στην περίπτωση που το κανάλι στη σχισμή j βρεθεί κατειλημμένο, ο κόμβος θα μεταβεί σε μία από τις καταστάσεις $\{c, 1, j+1\}$, με $c \in [0, W_1 - 1]$, με την ίδια πιθανότητα $1/W_1$. Οι πιθανότητες να βρίσκεται ο κόμβος σε κατάσταση ελέγχου του καναλιού είναι:

$$P\{S_1^j\} = \begin{cases} 0, & \text{για } \ j < 2 \\ \sum_{u=1}^{j-1} P\{S_0^u\} \cdot \frac{p_b^u}{W_1}, & \text{για } \ j \in [2, W_0] \\ P\{S_1^{W_0}\}, & \text{για } \ j \in [W_0 + 1, W_1 + 1] \\ P\{S_1^{W_0}\} - \sum_{u=1}^{j-W_1-1} P\{S_0^u\} \cdot \frac{p_b^u}{W_1}, & \text{για } \ j \in [W_1 + 2, W_{0,1} - 1] \\ 0, & \text{για } \ j > W_{0,1} - 1 \end{cases} \tag{5}$$

Η δεύτερη σχέση πηγάζει από το γεγονός ότι για $j <= W_0$, η πιθανότητα να βρίσκεται ένας κόμβος σε κατάσταση ελέγχου του καναλιού στο δεύτερο στάδιο υποχώρησης, εξαρτάται από τις πιθανότητες, να βρίσκεται σε κατάσταση ελέγχου του καναλιού στο πρώτο στάδιο και να έχει βρει το κανάλι κατειλημμένο.

Για παράδειγμα ένας κόμβος μπορεί να φτάσει στην κατάσταση S_1^3, εάν ήταν στην κατάσταση S_0^1, βρει το κανάλι κατειλημμένο και επιλέξει τυχαία την τιμή 1 για το δεύτερο στάδιο του backoff ή αν ήταν στο στην κατάσταση S_0^2, βρήκε το κανάλι κατειλημμένο και επέλεξε τον αριθμό 0 για το δεύτερο στάδιο.

Η τρίτη σχέση οφείλεται στο γεγονός ότι για $j > W_0$, δεν υπάρχουν μεταβάσεις μεταξύ των σταδίων $(BO_s = 0)$ και $(BO_s = 1)$, επειδή η τελευταία σχισμή που μπορεί να ελέγχει το κανάλι ένας κόμβος, στο πρώτο στάδιο είναι η $j = W_0 - 1 = 7$. Τελικώς, όταν ισχύει $j = W_1 + 2 = 18$, η $P\{S_1^{18}\}$ υπολογίζεται αφαιρώντας από την πιθανότητα $P\{S_1^{W_0}\}$ την $P\{S_0^1\}(p_b^1/W_1)$.

Έτσι έχουμε ότι, $P\{S_1^{18}\} = \sum_{u=2}^{W_0-1} P\{S_0^u\} \cdot (p_b^u/W_1)$. Εάν ένας κόμβος είναι στην κατάσταση S_0^1 τότε μεταβαίνει σε καταστάσεις $\{c, 1, 2\}$ με $c \in [0, 15]$. Γι' αυτό και στο σχήμα 10, η κατάσταση {16,1,2} δεν υπάρχει. Η τελευταία κατάσταση ελέγχου

καναλιού που μπορεί να βρεθεί ένας κόμβος σε αυτό το στάδιο είναι η S_1^{23}, το οποίο σημαίνει ότι το δεύτερο στάδιο θα έχει ολοκληρωθεί από κάποιο κόμβο μέχρι τη σχισμή $j = 24$.

- **Τρίτο Στάδιο Υποχώρησης (Third Backoff Stage)**: Ακολουθώντας την ίδια λογική όπως στα παραπάνω, έχουμε ότι οι πιθανότητες μετάβασης δίνονται από την σχέση:

$$P\{c, 2, j+1 | c+1, 2, j\} = 1 \tag{6}$$

για $c \in [0, W_2 - 2]$ και $j \in [3, W_{0,1,2} - 2]$. Οι πιθανότητες μετάβασης καταστάσεων του δεύτερου σταδίου ($BO_s = 1$) και του τρίτου ($BO_s = 2$) δίνονται από την σχέση:

$$P\{c, 2, j+1 | 0, 1, j\} = \frac{p_b^j}{W_2} \tag{7}$$

για $c \in [0, W_2 - 1]$ και $j \in [2, W_{0,1} - 1]$. Οι πιθανότητες να βρίσκεται κόμβος σε κατάσταση ελέγχου του καναλιού είναι:

$$P\{S_2^j\} = \begin{cases} 0, & \text{για } \ j < 3 \\ \sum_{u=2}^{j-1} P\{S_1^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [3, W_{0,1}] \\ P\{S_2^{W_{0,1}}\}, & \text{για } \ j \in [W_{0,1}+1, W_2+2] \\ P\{S_2^{W_{0,1}}\} - \sum_{u=2}^{j-W_2-1} P\{S_1^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [W_2+3, W_{0,1,2}-1] \\ 0, & \text{για } \ j > W_{0,1,2} - 1 \end{cases} \tag{8}$$

- **Τέταρτο Στάδιο Υποχώρησης (Fourth Backoff Stage)**: Ακολουθώντας την ίδια λογική όπως στα παραπάνω, έχουμε ότι οι πιθανότητες μετάβασης δίνονται από την σχέση:

$$P\{c, 3, j+1 | c+1, 3, j\} = 1 \tag{9}$$

για $c \in [0, W_2 - 2]$ και $j \in [4, W_{0,1,2,3} - 2]$. Οι πιθανότητες μετάβασης καταστάσεων του τρίτου σταδίου ($BO_s = 2$) και του τέταρτου ($BO_s = 3$) δίνονται από την σχέση:

$$P\{c, 3, j+1 | 0, 2, j\} = \frac{p_b^j}{W_2} \tag{10}$$

για $c \in [0, W_2 - 1]$ και $j \in [3, W_{0,1,2} - 1]$. Οι πιθανότητες να βρίσκεται κόμβος σε κατάσταση ελέγχου του καναλιού είναι:

$$P\{S_3^j\} = \begin{cases} 0, & \text{για } \ j < 4 \\ \sum_{u=3}^{j-1} P\{S_2^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [4, W_2 + 3] \\ \sum_{u=3}^{j-1} P\{S_2^u\} \cdot \frac{p_b^u}{W_2} - \sum_{u=3}^{j-W_2-1} P\{S_2^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [W_2 + 4, W_{0,1,2}] \\ P\{S_3^{W_{0,1,2}}\} - \sum_{u=W_{0,1}}^{j-W_2-1} P\{S_2^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [W_{0,1,2} + 1, W_{0,1,2,3} - 1] \\ 0, & \text{για } \ j > W_{0,1,2,3} - 1 \end{cases} \tag{11}$$

- **Πέμπτο Στάδιο Υποχώρησης (Fifth Backoff Stage)**: Ακολουθώντας την ίδια λογική όπως στα παραπάνω, έχουμε ότι οι πιθανότητες μετάβασης δίνονται από την σχέση:

$$P\{c, 4, j + 1 | c + 1, 4, j\} = 1 \tag{12}$$

για $c \in [0, W_2 - 2]$ και $j \in [5, W_{0,1,2,3,4} - 2]$. Οι πιθανότητες μετάβασης καταστάσεων του τρίτου σταδίου ($BO_s = 3$) και του τέταρτου ($BO_s = 4$) δίνονται από την σχέση:

$$P\{c, 4, j + 1 | 0, 3, j\} = \frac{p_b^j}{W_2} \tag{13}$$

για $c \in [0, W_2 - 1]$ και $j \in [4, W_{0,1,2,3} - 1]$. Οι πιθανότητες να βρίσκεται κόμβος σε κατάσταση ελέγχου του καναλιού είναι:

$$P\{S_4^j\} = \begin{cases} 0, & \text{για } \ j < 5 \\ \sum_{u=4}^{j-1} P\{S_3^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [5, W_2 + 4] \\ \sum_{u=4}^{j-1} P\{S_3^u\} \cdot \frac{p_b^u}{W_2} - \sum_{u=4}^{j-W_2-1} P\{S_3^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [W_2 + 5, W_{0,1,2,3}] \\ P\{S_4^{W_{0,1,2,3}}\} - \sum_{u=W_{0,1,2}}^{j-W_2-1} P\{S_3^u\} \cdot \frac{p_b^u}{W_2}, & \text{για } \ j \in [W_{0,1,2,3} + 1, W_{0,1,2,3,4} - 1] \\ 0, & \text{για } \ j > W_{0,1,2,3,4} - 1 \end{cases} \tag{14}$$

4.2.3 Μαθηματικό μοντέλο - Μέρος 2

Ο σκοπός όπως προαναφέρθηκε είναι να εκτιμηθεί η πιθανότητα ένας κόμβος να βρεθεί να ολοκληρώνει την μετάδοση του σε μία σχισμή j, με $P\{T^j\}$, με $j \in [0, t_{max} + D - 1]$.Κάθε

κόμβος τελειώνει την μετάδοση του στη σχισμή j, εάν στη σχισμή $j - D$ βρει το κανάλι ελεύθερο. Οπότε η πιθανότητα αυτή δίνεται από την σχέση:

$$P\{T^j\} = P\{C^{j-D}\} \cdot (1 - p_b^{j-D}) \tag{15}$$

Επειδή η μετάδοση κρατάει D slots, συνδέουμε την πιθανότητα $P\{T^j\}$ με την σχισμή στην οποία τελειώνει η μετάδοση, άρα για $j < D, P\{T^j\} = 0$. Η πιθανότητα λοιπόν να βρίσκεται ο κόμβος σε κατάσταση ελέγχου του καναλιού, για διάφορες σχισμές j είναι:

$$P\{C^j\} = \begin{cases} P\{S_0^0\}, & \text{για } \ j = 0 \\ P\{S_0^1\}, & \text{για } \ j = 1 \\ \sum_{i=0}^{j-1} P\{S_i^j\}, & \text{για } \ j = (2, \ldots, 4) \\ \sum_{i=0}^{NB_{max}} P\{S_i^j\}, & \text{για } \ j = (5, \ldots, W_0 - 1) \\ \sum_{i=1}^{NB_{max}} P\{S_i^j\}, & \text{για } \ j = (W_0, \ldots, W_{0,1} - 1) \\ \sum_{i=2}^{NB_{max}} P\{S_i^j\}, & \text{για } \ j = (W_{0,1}, \ldots, W_{0,1,2} - 1) \\ \sum_{i=3}^{NB_{max}} P\{S_i^j\}, & \text{για } \ j = (W_{0,1,2}, \ldots, W_{0,1,2,3} - 1) \\ P\{S_4^j\}, & \text{για } \ j = (W_{0,1,2,3}, \ldots, W_{0,1,2,3,4} - 1) \end{cases} \tag{16}$$

Οι πρώτες δύο εξισώσεις προέρχονται από το γεγονός ότι, στις σχισμές 0 και 1, ένας κόμβος μπορεί να βρίσκεται μόνο στο πρώτο στάδιο υποχώρησης (backoff stage). Εάν βρίσκεται στη σχισμή 2, τότε μπορεί να ελέγχει το κανάλι είτε στο πρώτο στάδιο, είτε στο δεύτερο και ομοίως για μεγαλύτερα slots.

Από την σχισμή j=8 και μετά ένας κόμβος δεν μπορεί να είναι στο πρώτο στάδιο πλέον, οπότε στο άθροισμα στην παραπάνω τα όρια ξεκινούν στο i=1, ομοίως και για υψηλότερες σχισμές ισχύει η ίδια λογική και στα υπόλοιπα αθροίσματα. Για σχισμές από την $W_{0,1,2,3,} = 88$ και μέχρι την τελευταία $W_{0,1,2,3,4} = 119$, ένας κόμβος μπορεί να βρίσκεται σε κατάσταση ελέγχου του καναλιού στο τελευταίο στάδιο μόνο.

Για να εκτιμηθούν και οι υπόλοιπες πιθανότητες που ζητούνται από το μοντέλο θα πρέπει να οριστεί ο αριθμός N_c^j. Το N_c^j είναι ο αριθμός των κόμβων οι οποίοι δεν έχουν μεταδώσει ακόμη στο τέλος της σχισμής $j - 1$, και θα ανταγωνιστούν στη σχισμή j. Πιο συγκεκριμένα στο slot 0, ο αριθμός των κόμβων που ανταγωνίζονται για το κανάλι είναι ίσος με N και επειδή

κανείς δεν μπορεί να μεταδώσει στο αρχικό slot j=0, $PT^0 = 0$ και N_c^1 είναι ίσο με N επίσης. Ενώ στην σχισμή 1, κάποιοι κόμβοι μπορεί να ολοκληρώσουν την μετάδοση τους ο καθένας με πιθανότητα $P\{T^1\}$ και στο τέλος της σχισμής, ο αριθμός των κόμβων που δεν θα έχουν ακόμη καταφέρει να μεταδώσουν N_c^2, εξαρτάται από την πιθανότητα $P\{T^1\}$ και μπορεί να είναι μικρότερος από N.

Στην περίπτωση λοιπόν που $D = 1$, η πιθανότητα k κόμβοι από τους $\hat{N}$ την σχισμή j να μην έχουν ακόμη μεταδώσει, υπό την προϋπόθεση ότι στη σχισμή j-1, $\hat{N}$ κόμβοι ανταγωνίζονται για το κανάλι ($N_c^{j-1} = \hat{N}$), είναι ίση με:

$$\begin{aligned} P\{N_c^j = k | N_c^{j-1} = \hat{N}\} &= B^j(k, \hat{N}) \\ &= \binom{\hat{N}}{k} (1 - p_b^{j-2})(P\{C^{j-2}\})^{\hat{N}-k} \\ &\quad \cdot \prod_{i=0}^{NB_{max}} (1 - P\{S_i^{j-2}\})^k \end{aligned}$$

όπου $(1 - p_b^{j-2})(P\{C^{j-2}\})^{\hat{N}-k}$ είναι η πιθανότητα $\hat{N} - k$ κόμβοι να μεταδώσουν την σχισμή j και $\prod_{i=0}^{NB_{max}}(1 - P\{S_i^{j-2}\})^k$ είναι η πιθανότητα οι υπόλοιποι k κόμβοι που απομένουν να μην μεταδώσουν,αφού δεν ελέγχουν το κανάλι στη σχισμή $j - 2$.

Όπως αναφέρεται στο παράρτημα του [2] ο αριθμός N_c^{j-1}, είναι μια τυχαία μεταβλητή, με κατανομή διωνυμική, όπου εξαρτάται από τις τιμές $P\{T^l\}$ με $l \in (1, \ldots, j-2)$. Αυξάνοντας τον αρχικό αριθμό των κόμβων στο δίκτυο N_c^0 και τον αριθμό των σχισμών j, η πολυπλοκότητα αυξάνεται για τον υπολογισμό του αριθμού αυτού. Εισάγεται λοιπόν η προσέγγιση ότι, η μεταβλητή N_c^{j-1} δεν μοντελοποιείται σαν μια τυχαία μεταβλητή, αλλά ορίζεται σαν μία μεταβλητή k για την οποία, μεγιστοποιείται η πιθανότητα $B^{j-1}(k, \hat{N})$.

$$P\{N_c^{j-1}\} = \underset{k}{\operatorname{argmax}}\, B^{j-1}(k, \hat{N}) \tag{17}$$

Η μοντελοποίηση για τιμές του N_c^j στην περίπτωση που το $D > 1$ είναι ακόμη πιο σύνθετες. Αλλά όσο το D μεγαλώνει και τα πακέτα διαρκούν περισσότερο να μεταδοθούν στο κανάλι, το N_c^j θα μικραίνει με χαμηλό ρυθμό κατά την διάρκεια του χρόνου. Μπορεί λοιπόν να παρθεί η τιμή $N_c^j = N$, για οποιαδήποτε σχισμή j, για $D > 1$.

Τώρα μπορεί να οριστεί και η πιθανότητα p_s, επιτυχούς μετάδοσης ενός πακέτου στο κανάλι, η οποία δίνεται από τον τύπο:

$$p_s = \sum_{j=0}^{t_{max}+D-1} P\{Z^j\} \tag{18}$$

όπου $P\{Z^j\}$ είναι η πιθανότητα μια επιτυχή μετάδοση να τελειώσει στην σχισμή j, το οποίο σημαίνει ότι μία και μόνο μία μετάδοση αρχίζει την σχισμή $j - D + 1$. Επειδή θεωρείται ότι δεν υπάρχει περίπτωση hidden node και όλοι οι κόμβοι ακούνε τις μεταδόσεις των άλλων, εάν ένας μόνο κόμβος μεταδώσει την σχισμή $j - D + 1$ ο κόμβος βάσης θα λάβει σωστά το πακέτο του. Από τον νόμο της ολικής πιθανότητας έχουμε ότι:

$$\begin{aligned} P\{Z^j\} = P\{1 \text{ tx in } (j-D+1) \mid \text{channel free in } (j-D)\} \\ \cdot P\{\text{channel free in } (j-D)\} \\ + P\{1 \text{ tx in } (j-D+1) \mid \text{channel busy in } (j-D)\} \\ \cdot P\{\text{channel busy in } (j-D)\} \end{aligned} \tag{19}$$

Όπου $P\{1 \text{ tx in } (j-D+1) \mid \text{channel free in } (j-D)\}$ και $P\{1 \text{ tx in } (j-D+1) \mid \text{channel busy in } (j-D)\}$ είναι οι πιθανότητες που μία και μόνο μία μετάδοση ξεκινά την σχισμή $j-D+1$ δεδομένου ότι το κανάλι την σχισμή $j-D$ είναι ελεύθερο ή κατειλημμένο αντίστοιχα.

Επειδή μόνο μία μετάδοση, ξεκινά τη σχισμή $j - D + 1$, αν ένας μόνο κόμβος από τούς N_c^{j-D}, ελέγχει το κανάλι τη σχισμή $j - D$ και αν δεν ξεκινήσουν μεταδόσεις τη σχισμή $j - D + 1$, εάν το κανάλι την σχισμή $j - D$ είναι κατειλημμένο, τότε η πιθανότητα $P\{Z^j\}$ δίνεται από τον τύπο:

$$\begin{aligned} P\{Z^j\} = (1 - p_b^{j-D}) P\{C^{j-D}\} \\ \cdot \prod_{i=0}^{NB_{max}} (1 - P\{S_i^{j-D}\})^{N_c^{j-D}-1} \end{aligned} \tag{20}$$

Όπου η $P\{C^{j-D}\}$ πιθανότητα είναι η πιθανότητα ένας κόμβος να ελέγχει το κανάλι τη σχισμή $j - D$, και $\prod_{i=0}^{NB_{max}} (1 - P\{S_i^{j-D}\})^{N_c^{j-D}-1}$ είναι η πιθανότητα οι υπόλοιποι $N_c^{j-D} - 1$ κόμβοι να μην ελέγχουν το κανάλι τη σχισμή $j - D$.

Τελικώς η πιθανότητα PR^j ο κόμβος βάσης να λάβει την ουρά ενός πακέτου, από οποιονδήποτε κόμβο, την j σχισμή, δίνεται από τον τύπο:

$$P\{R^j\} = N_c^j \cdot P\{Z^j\} \tag{21}$$

Τέλος ορίζοντας ως p_f^j την πιθανότητα ότι το κανάλι είναι ελεύθερο την σχισμή j, μπορούμε να ορίσουμε επίσης την πιθανότητα

$$p_b^j = 1 - p_f^j \tag{22}$$

Από τον νόμο της ολικής πιθανότητας, εκφράζουμε το p_f^j ως:

$$\begin{aligned} p_f^j = P\{\text{no tx in } \ j \ \mid \ \text{channel free in } \ (j-1)\} \\ \cdot P\{\text{channel free in } \ (j-1)\} \\ + P\{\text{no tx in } \ j \ \mid \ \text{channel busy in } \ (j-1)\} \\ \cdot P\{\text{channel busy in } \ (j-1)\} \end{aligned} \tag{23}$$

Όπου $P\{\text{no tx in}j|\text{channel free in}(j-1)\}$ και $P\{\text{no tx in}j|\text{channel busy in}(j-1)\}$ είναι οι πιθανότητες να μην συμβούν μεταδόσεις στην σχισμή j, δεδομένου ότι το κανάλι στη σχισμή j-1 είναι ελεύθερη ή κατειλημμένη, αντίστοιχα.

Όταν $D = 1$, $P\{\text{no tx in}j|\text{channel free in}(j-1)\} = \prod_{i=0}^{NB_{max}}(1-P\{S_k^{j-1}\})^{N_c^{j-1}-1}$, επειδή όταν το κανάλι στη σχισμή $j-1$ είναι ελεύθερο, καμία μετάδοση δεν συμβαίνει τη σχισμή j, εάν κανείς κόμβος δεν ελέγχει το κανάλι τη σχισμή j-1.

Αντίθετα όταν, το κανάλι τη σχισμή $j-1$ είναι κατειλημμένο, καμία μετάδοση δεν θα συμβεί τη σχισμή j. Άρα σε αυτή την περίπτωση η πιθανότητα p_f^j δίνεται από τον τύπο:

$$p_f^j = (1 - p_b^{j-1}) \prod_{i=0}^{NB_{max}} (1 - P\{S_i^{j-1}\})^{N_c^{j-1}-1} + p_b^{j-1} \tag{24}$$

Όταν, $D > 1$ ο δεύτερος όρος της εξίσωσης (23) συμπίπτει με την πιθανότητα ότι στη σχισμή $j-1$, μία μετάδοση τελειώνει, δηλαδή είναι η πιθανότητα ότι τουλάχιστον μία μετάδοση ξεκινά τη σχισμή $j-D$, η οποία δίνεται από τον τύπο: $(1-p_b^{j-D-1})\cdot[1-\prod_{k=0}^{NB_{max}}\left[(1-P\{S_k^{j-D-1}\})^{N_c^{j-D-1}-1}\right]$

Σε αυτή την περίπτωση η πιθανότητα p_f^j, δίνεται από τον τύπο:

$$\begin{aligned} p_f^j = (1-p_b^{j-1}) \prod_{i=0}^{NB_{max}} (1-P\{S_i^{j-1}\})^{N_c^{j-1}-1} \\ +(1-p_b^{j-D-1}) \cdot \left[1 - \prod_{k=0}^{NB_{max}} (1-P\{S_i^{j-D-1}\})^{N_c^{j-D-1}-1}\right] \end{aligned} \tag{25}$$

Όταν $j \leq D$ τότε το δεύτερο κομμάτι της σχέσης (25) γίνεται 0.

5 Υλοποίηση - Εφαρμογή

5.1 Περιγραφή

Η εφαρμογή μας απαιτεί την δικτύωση αισθητήρων ενός χωραφιού, για αυτό το σκοπό δόθηκε έμφαση ώστε το πρωτόκολλο να προσαρμοστεί, στα πλαίσια των ρυθμίσεων του ραδιοφώνου μας και του υλικού μας γενικότερα, ώστε να καλύπτονται δύο βασικοί στόχοι:

- Να καλύπτει η μετάδοση μας την μέγιστη δυνατή απόσταση. Θέλουμε μια χαμηλή σε κόστος λύση, δηλαδή με λίγους κόμβους να καλύψουμε μεγάλη εδαφική περιοχή. Γι' αυτό το σκοπό ρυθμίσαμε το ραδιόφωνο σε χαμηλό data rate. Αυτό δεν συμβαδίζει με το πρότυπο 802.15.4, όμως η ρύθμιση αυτή, ήταν αναγκαία.

- Οι κόμβοι μας, αν και αποτελούμενοι, από χαμηλής κατανάλωσης ενέργειας υλικό, θα πρέπει να μπορούν να είναι ενεργειακά αυτόνομοι, για μεγάλα χρονικά διαστήματα, καθώς η τοποθέτηση και η αλλαγή τους μέσα στο χωράφι, θα θέλαμε να μην γίνεται συχνά.

Στα πλαίσια αυτά έγινε χρήση όλων των δυνατοτήτων του ραδιοφώνου CC2500, έτσι ώστε να επιτευχθούν οι στόχοι μας. Οι σημαντικότερες ρυθμίσεις, που χρησιμοποιήθηκαν για το ραδιόφωνο, και τον μικροελεγκτή ήταν η εξής:

- **Wake-On-Radio**: Έγιναν οι απαραίτητες ρυθμίσεις, έτσι όπως περιγράφονται και στο σημείωμα [5] στο ραδιόφωνο, έτσι ώστε το τελευταίο να λειτουργεί τον περισσότερο χρόνο σε πολύ χαμηλή κατανάλωση, όπως θα δούμε και στη συνέχεια.

- **Idle Mode - MCU**: Εκτός του ραδιοφώνου και ο μικροελεγκτής θα λειτουργεί σε κατάσταση χαμηλής κατανάλωσης. Θα δούμε στην συνέχεια ότι αυτός θα λειτουργεί μόνο όταν πρέπει.

- **CRC - LQI**: Χρησιμοποιήθηκε κάθε δυνατότητα ελέγχου των πακέτων, αφού πρέπει να εγγυάται η ακρίβεια στην λήψη των πακέτων, όπως ορίζει και το πρότυπο 802.15.4 και απαιτεί και η εφαρμογή μας.

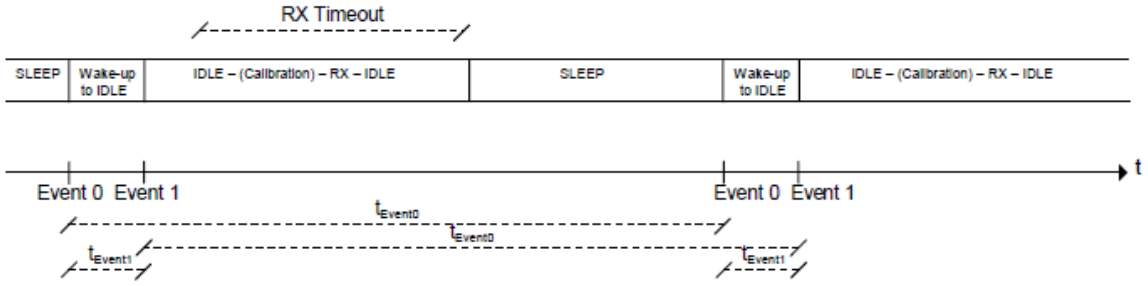
- **Sync Word detection**: Άλλο ένα μέσο διασφάλισης της ορθότητας, για την λήψη των πακέτων το οποίο όπως θα δούμε ήταν και σημαντική παράμετρος του Wake-On-Radio σεναρίου αφύπνισης.

5.2 Χαμηλή Κατανάλωση (Wake-On-Radio)

Όπως προαναφέρθηκε, ίσως από τα σημαντικότερα ζητήματα στον τομέα των ασύρματων δικτύων αισθητήρων, είναι η χαμηλή κατανάλωση και η ενεργειακή αυτονομία των κόμβων του δικτύου. Γι΄ αυτόν το σκοπό έπρεπε να εφαρμοστεί, η λειτουργία Wake-on-Radio(WOR) του ραδιοφώνου.

Σε αυτήν την λειτουργία, το ραδιόφωνο μπορεί να βρίσκεται στην κατάσταση SLEEP και ανά καθορισμένο χρονικό διάστημα Event0, να μεταβαίνει από κατάσταση SLEEP, σε IDLE (κάνοντας και ένα calibration του ταλαντωτή συχνοτήτων του) και τέλος σε κατάσταση λήψης RX. Να ελέγχει το κανάλι (Rx polling) για καθορισμένο διάστημα (RX Timeout), και αν βρίσκει σήμα στο κανάλι ή αναγνωρίσει πακέτο τότε να πράττει ανάλογα (συνήθως παραμένει σε κατάσταση RX, έως ότου λάβει το πακέτο). Εάν δεν ανιχνευθεί σήμα στο κανάλι, έπειτα απο χρόνο (RX Timeout), τότε το ραδιόφωνο μεταβαίνει ξανά σε κατάσταση SLEEP. Και αυτή η διαδικασία επαναλαμβάνεται.

Τα παραπάνω διαστήματα φαίνονται αναλυτικά στο σχήμα 14 από το [5].



Σχήμα 14: Απόσπασμα από [5] για Wake-On-Radio Events

Σύμφωνα με τους κατασκευαστές το ραδιόφωνο στην κατάσταση SLEEP και όταν είναι ενεργοποιημένο σωστά το WOR, έχει πολύ χαμηλή κατανάλωση ρεύματος, ίση με 900 nA.

Θα ορίσουμε μια βασική έννοια για εφαρμογές σε WSNs, η οποία ονομάζεται εποχή (**epoch**). Μία εποχή ορίζεται, στα πλαίσια αυτής της εργασίας, ως το χρονικό διάστημα κατά το οποίο αφυπνίζονται οι κόμβοι από τον σταθμό βάσης, εκτελούν τα καθήκοντα τους (παίρνουν μετρήσεις κτλ) και κάνουν πρόσβαση στο μέσο, ώστε να στείλουν την πληροφορία τους στο σταθμό βάσης.

Η λογική του αλγορίθμου για το WOR σενάριο λοιπόν, στην εφαρμογή μας, για το WSN μας, είναι η εξής:

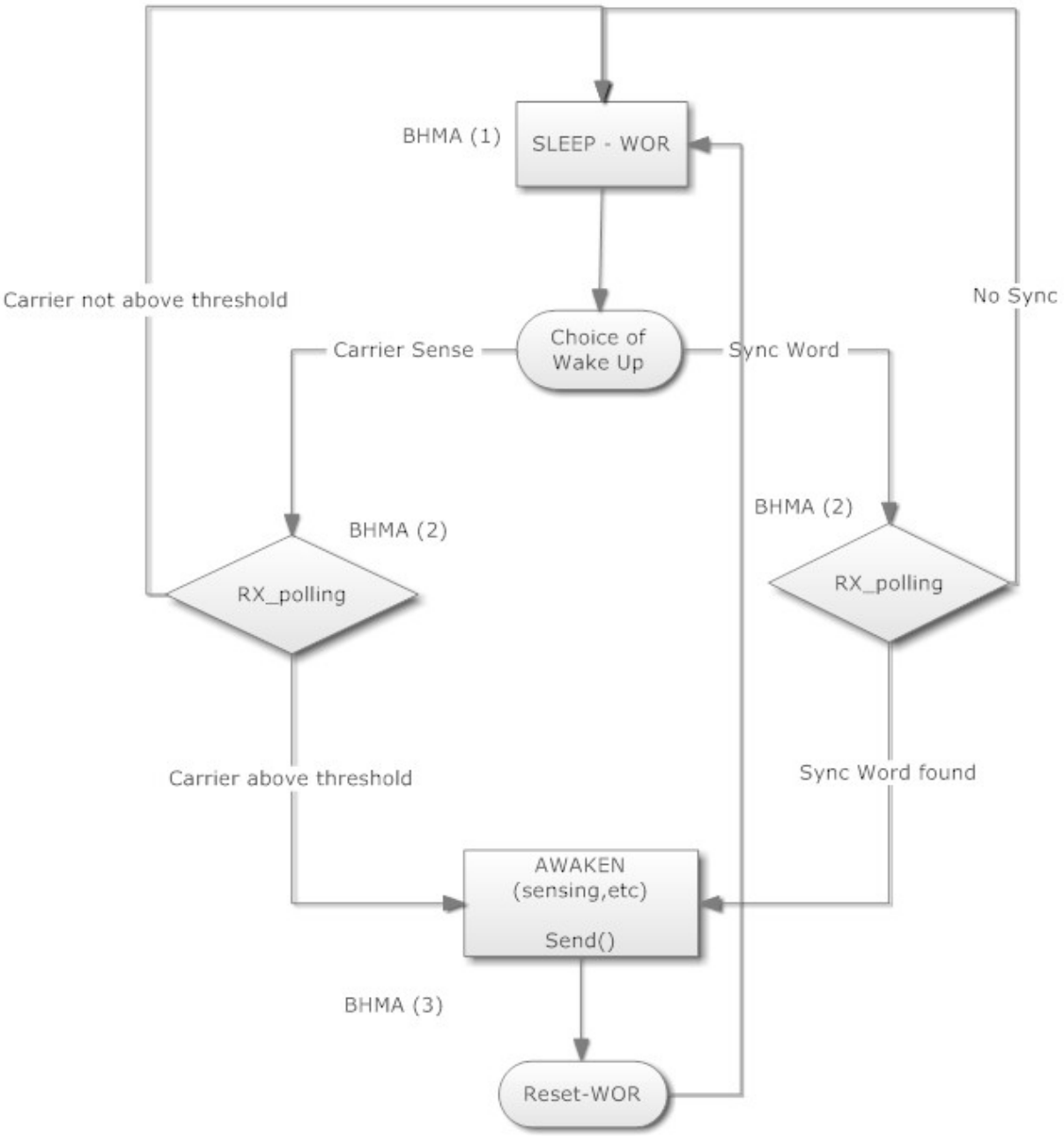
- **Βήμα 1**: Κάθε κόμβος πριν την αρχή μιας περιόδου μετρήσεων, θα βρίσκεται σε κατάσταση WOR ραδιοφώνου και θα περιμένει την αφύπνιση του από τον κόμβο βάσης, ελέγχοντας το κανάλι, όπως είναι προγραμματισμένο ανάλογα με τις περιόδους του WOR. Θα έχει επίσης τον μικροελεγκτή του σε κατάσταση (Idle Mode), για ακόμη μικρότερη κατανάλωση. Ο μικροελεγκτής C8051F320 όταν μπει σε αυτή την κατάσταση χαμηλής κατανάλωσης, μπορεί να παραμένει σε αυτήν όσο δεν του έρχονται αιτήσεις διακοπών interrupts. Αυτό όπως θα δούμε επηρεάζει το επόμενο βήμα.

- **Βήμα 2**: Κάθε κόμβος θα πρέπει να μπορεί να επανέρχεται σε κανονική λειτουργία, όταν έρθει η περίοδος των μετρήσεων και να αποστέλλει την πληροφορία που απαιτείται, στον σταθμό βάσης σύμφωνα με το πρωτόκολλο που έχουμε προαναφέρει. Η περίοδος των μετρήσεων είναι αρκετά σποραδική (όπως συνήθως σε εφαρμογές WSN με low-power χαρακτηριστικά) και ο μικροελεγκτής μας μπορεί να παραμένει σε κατάσταση χαμηλής λειτουργίας όσο δεν έχει αιτήσεις διακοπών interrupts.

  Επειδή οι εσωτερικοί μετρητές του μικροελεγκτή C8051F320, κάθε φορά που υπερχειλίζουν προκαλούν μία διακοπή σε αυτόν, και επειδή είναι στην καλύτερη περίπτωση 16 bit θα μας προκαλούσαν αρκετές διακοπές (interrupts), κατά την διάρκεια του (Idle Mode) και θα μας έβγαζαν από αυτή την κατάσταση. Κάτι το οποίο φεύγει από τα πλαίσια της χαμηλής κατανάλωσης. Άρα καταλήγουμε ότι οι κόμβοι και ο μικροελεγκτής τους, θα πρέπει να αφυπνίζονται και να συντονίζονται, από τον κόμβο βάσης όταν έρχεται η περίοδος μετρήσεων. Θα δούμε στη συνέχεια τους τρόπους με τους οποίους θα γίνει αυτό.

- **Βήμα 3**: Σε αυτό το βήμα οι κόμβοι έχουν ενεργοποιηθεί από τον σταθμό βάσης και πράττουν αναλόγως με τις απαιτήσεις της εφαρμογής (συνήθως παίρνουν μετρήσεις) και στέλνουν τα αποτελέσματα τους, προς τον σταθμό, κάνοντας πρόσβαση στο μέσο σύμφωνα με τον αλγόριθμο CSMA/CA, όπως ορίστηκε προηγουμένως. Στο τέλος της διαδικασίας αυτής πρέπει να ξαναμπούν σε κατάσταση WOR (Reset WOR) του ραδιοφώνου, να βάλουν τον μικροελεγκτή τους σε κατάσταση (Idle Mode) και να μπουν ξανά στο βήμα 1.

Ακολουθεί το διάγραμμα που φαίνονται οι καταστάσεις του αλγορίθμου.



Σχήμα 15: Αλγόριθμος για Wake-On-Radio εφαρμογή

Ο τρόπος με τον οποίο θα ξυπνήσει ένας κόμβος είναι επίσης σημαντικός. Γενικότερα ο μικροελεγκτής όταν βρίσκεται σε κατάσταση (Idle Mode), έχει σταματήσει την ροή του κώδικα στο σημείο που καλέστηκε η αντίστοιχη halSleep() συνάρτηση και περιμένει μόνο μία εξωτερική διακοπή, ώστε να συνεχίσει την λειτουργία του, από εκείνο το σημείο.

Η εφαρμογή μας λοιπόν σε αυτό το σημείο θα πρέπει να τον ξυπνήσει όταν έρθει η κατάλληλη στιγμή με κάποιο σήμα στον αέρα, από τον σταθμό βάσης (αφού το ραδιόφωνο λειτουργεί σε WOR και ελέγχει το κανάλι περιοδικά).

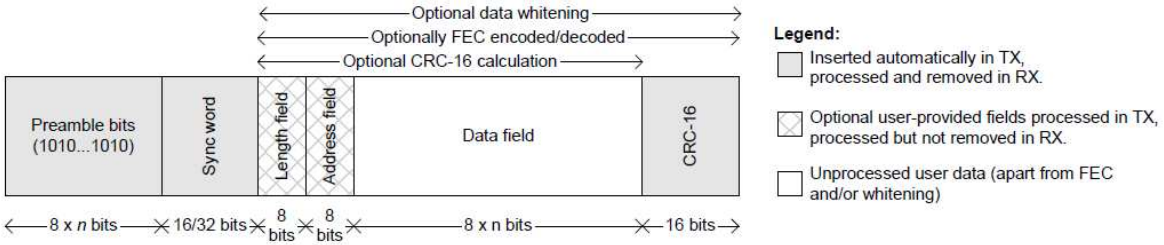
Παρακάτω βρίσκονται πιθανές επιλογές που μπορούν να χρησιμοποιηθούν και τα χαρακτηριστικά τους:

- **Αφύπνιση με σήμα καναλιού (Carrier Sense)**: Όπως προαναφέρθηκε το ραδιόφωνο δίνει την δυνατότητα έτσι ώστε να λαμβάνεται η ισχύς του σήματος του καναλιού (RSSI) και να μπορούμε να γνωρίζουμε πότε υπάρχει κάποιο σήμα στον αέρα (Carrier Sense). Αυτές ακριβώς οι λειτουργίες χρησιμοποιούνται και στην υλοποίηση του CSMA για τον CCA έλεγχο.

  Συνεπώς σε μία απλή υλοποίηση, ο σταθμός βάσης στέλνει κάποια πακέτα, όταν πρέπει να ξυπνήσουν οι κόμβοι και επειδή η ισχύς του σήματος στο κανάλι θα αλλάξει, οι κόμβοι θα δουν την αλλαγή, κατά τον έλεγχο τους και αν υπερβαίνει ένα προγραμματιζόμενο όριο (carrier threshold) θα ξυπνήσουν.

  Αυτό το μοντέλο προϋποθέτει την σωστή ρύθμιση του (carrier threshold), το οποίο προφανώς αλλάζει από περιβάλλον σε περιβάλλον, είναι ευαίσθητο σε μεταβολές και σε τυχόν παρεμβολές και το κυριότερο είναι ότι, μπορεί να εντείνει τις διακυμάνσεις στις στιγμές, τις οποίες οι κόμβοι ξυπνάνε. Έως και να ασυγχρονίσει σε μεγάλο βαθμό την περίοδο αφύπνισης του δικτύου (μπορεί δηλαδή ένας κόμβος να ξυπνήσει λόγω ισχυρότερου σήματος, κοντινού ή άλλου κόμβου και όχι από τον σταθμό βάσης).

- **Αφύπνιση με λήψη πακέτου (Sync Word detection)**: Το ραδιόφωνο παρέχει την δυνατότητα να υπάρχει πρόθεμα στο πακέτο μας (μορφή πακέτου στο σχήμα 16), κατά την αποστολή του, έτσι ώστε να υπάρχει ένα πρώτο ποιοτικό μέτρο σύγκρισης και περιορισμού των λαθών. Πακέτα με λανθασμένη την λέξη συγχρονισμού (sync word) δεν θα γίνονται δεκτά και θα απορρίπτονται. Μπορεί λοιπόν να χρησιμοποιηθεί η λογική, ότι για να ξυπνήσει ένας κόμβος θα πρέπει να λάβει ένα σωστό πακέτο, από τον σταθμό βάσης την ώρα που εξετάζει το κανάλι.



Σχήμα 16: Απόσπασμα απο το [3] - Μορφή ενός πακέτου

Αυτό όπως θα δούμε προϋποθέτει μία συγκεκριμένη λειτουργία κατά την περίοδο αφύπνισης από τον σταθμό βάσης, η οποία με την σειρά της προϋποθέτει από τον σταθμό να έχει αυτονομία ενέργειας, λύνει όμως το πρόβλημα ότι μπορεί να ξυπνήσουν οι κόμβοι, όταν δεν πρέπει και εισάγει μια μορφή συγχρονισμού στο σύστημα μας. Αυτή ήταν και η τελική μας επιλογή για την εφαρμογή μας. Τα σημεία που μας περιορίζει αυτή η επιλογή, καθώς και η τακτική της αφύπνισης με πακέτα αναλύονται παρακάτω.

Μία sync word στο ραδιόφωνο CC2500 μπορεί να επιλεχθεί με μέγεθος μεταξύ 16-32 bit. Αυτό σημαίνει ότι όταν χρησιμοποιείται, η ρύθμιση για τον τερματισμό της κατάστασης λήψης RX, όπως στην περίπτωση του WOR. Θα πρέπει η ρύθμιση του (RX Timeout) να είναι τέτοια ώστε να μπορεί να ληφθεί μία ολόκληρη sync word. Αλλιώς κανένα πακέτο δεν προλαβαίνει να ληφθεί και στην περίπτωση της αφύπνισης του WOR σεναρίου, κανείς κόμβος δεν θα αφυπνίζεται.

Θα πρέπει λοιπόν να ισχύει ότι:

$$t_{\text{RX\_ timeout}} \geq t_{\text{sync\_ detect}} = \frac{(N_{\text{sync word bits}} + N_{\text{preamble bits}})}{TX_{\text{rate}}} \tag{26}$$

αι ακόμη καλύτερα θα πρέπει να ισχύει: $t_{\text{RX\_ timeout}} = 2 \times t_{\text{sync\_ detect}}$, για μεγαλύτερη πιθανότητα αφύπνισης. Ο χρόνος (RX Timeout) δυστυχώς δεν ορίζεται απευθείας σαν τιμή, αλλά ως ποσοστό % του χρόνου Event0. Ουσιαστικά, το ποσοστό αυτό αναπαριστά και το duty cycle του WOR, αφού μόνο σε αυτό το ποσοστό του χρόνου θα καταναλώνει ρεύμα το ραδιόφωνο, λόγω της κατάστασης λήψης, τον υπόλοιπο χρόνο θα παραμένει σε χαμηλή

κατανάλωση. Οι επιλογές του ποσοστού αυτού είναι συγκεκριμένες και ενδέχεται λοιπόν να πρέπει να οριστεί μεγαλύτερο RX Timeout από το θεμιτό.

Αποτελεί το RX Timeout δηλαδή ένα tradeoff ανάμεσα στην επιτυχία του αλγορίθμου αφύπνισης και της κατανάλωσης ενέργειας. Καθώς όσο το ραδιόφωνο βρίσκεται σε κατάσταση RX καταναλώνει από 14.5mA - 17mA, στα 2.4 KBaud.

Η τιμή που ορίζει το Event0, βρίσκεται σε δυο καταχωρητές μεγέθους ενός byte, του ραδιοφώνου, τον WOREVT1 και WOREVT0. Μαζί μπορούν να δώσουν μέγιστη τιμή 65535. Ο χρόνος αυτός δίνεται από την σχέση:

$$t_{\text{Event0}} = \frac{750}{f_{XOSC}} \cdot EVENT0 \cdot 2^{5 \cdot WOR_{RES}} \tag{27}$$

όπου f_{XOSC} είναι το ρολόι του ραδιοφώνου (26MHz) και WOR_{RES} ένα εκθετικό για την παραγωγή αρκετά μεγάλης περιόδου t_{Event0}. Βάζοντας το μέγιστο δυνατό EVENT0 και γιά $WOR_{RES} = 0$ έχουμε ότι $t_{\text{Event0}} = 1,89sec$. Βρίσκουμε στην συνέχεια, το ελάχιστο ποσοστό χρόνου, για το RX Timeout που μπορούμε να επιτύχουμε, από τις επιλογές του ραδιοφώνου, έτσι ώστε να ισχύει η σχέση(26).

Με προσεγγιστικούς υπολογισμούς για $t_{\text{Event0}} = 1,89 \ \ sec$ και ποσοστό RX Timeout $= 3,125 \ \% \ \ t_{\text{Event0}} = 59 \ \ msec$, υπολογίζουμε οτι με μπαταρίες τύπου AA (2000 mAh), υπάρχει αυτοδυναμία των κόμβων για 82 ημέρες. Ενώ εάν οι κόμβοι π.χ άκουγαν συνεχώς το κανάλι, χωρίς το WOR σενάριο, θα είχαν αυτοδυναμία για 4 ημέρες.

Για την διαδικασία αφύπνισης με πακέτα, σε κάθε εποχή (epoch). Θα γίνονται τα εξής:

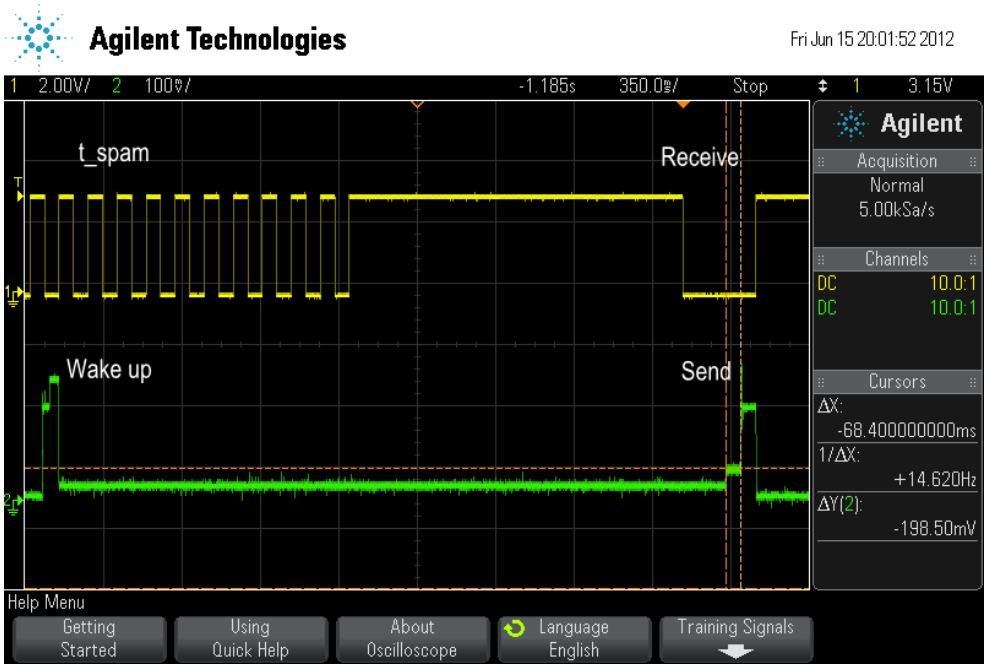
- **Ο Σταθμός Βάσης** (Base Station)στέλνει συνεχώς πακέτα για χρόνο $t_{spam} \geq t_{\text{Event0}}$. Κατά προτίμηση $t_{spam} = 2 \times t_{\text{Event0}}$. Αυτό αποτελεί και tradeoff μεταξύ κατανάλωσης ενέργειας και επιτυχίας της αφύπνισης, καθώς όπως περιγράφεται παρακάτω οι κόμβοι που έχουν αφυπνιστεί, αναμένουν κάποιο χρόνο για να ξεκινήσουν να μεταδίδουν. Κατά τη διάρκεια αυτού του χρόνου δεν έχουν το ραδιόφωνο τους σε κατάσταση SLEEP.

- **Ο κάθε κόμβος** (Node) που αφυπνίζεται μέσα στο t_{spam} διάστημα, περιμένει για διάστημα t_{spam}, πριν προσπαθήσει να μεταδώσει. Αυτό γίνεται ώστε να αποφευχθεί,

να προσπαθήσει κάποιος κόμβος να μεταδώσει, ενώ ακόμη στέλνει πακέτα αφύπνισης ο κόμβος βάσης. Η χειρότερη περίπτωση προφανώς είναι να ξυπνήσει κάποιος κόμβος στην αρχή ακριβώς της περιόδου t_{spam}, με το πρώτο πακέτο του σταθμού βάσης.

Όταν ένας κόμβος λάβει επιτυχώς ένα πακέτο, προκαλείται διακοπή στον μικροελεγκτή οπότε ξυπνάει και ελέγχει, εάν ήταν πακέτο του σταθμού βάσης. Εάν ήταν προχωράει στην εκτέλεση των καθηκόντων του, ενώ αν δεν ξύπνησε από τέτοιο πακέτο ξανά κοιμάται. Έτσι αποφεύγουμε το φαινόμενο να ξυπνήσει κάποιος κόμβος από αποστολή άλλου κόμβου και όχι από τον σταθμό βάσης.

Ένα διάστημα μίας τέτοιας εποχής (epoch) φαίνεται στο παρακάτω σχήμα 17, από παλμογράφο σε πραγματικό χρόνο.



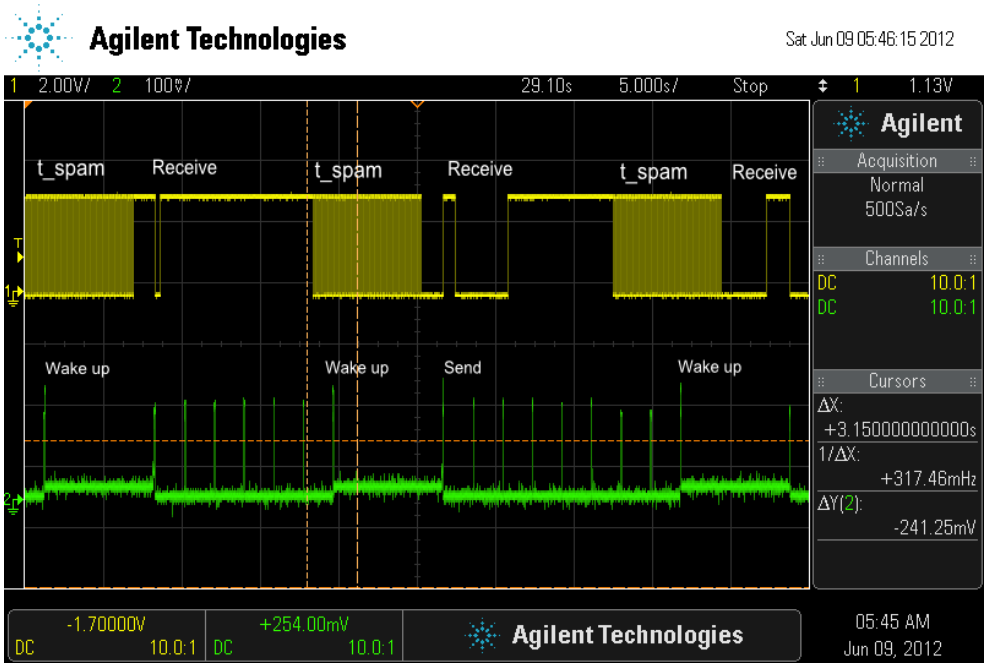
Σχήμα 17: Εικόνα Παλμογράφου - Μία Εποχή

Με μία έξοδο του μικροελεγκτή, μπορούμε να ανιχνεύουμε την πορεία των λειτουργιών κατά την διάρκεια μίας εποχής. Και με αυτό τον τρόπο, για τον σταθμό βάσης (πάνω κανάλι) καταγράφουμε με κάθε αλλαγή (0 - 1), στην αρχή τις μεταδόσεις πακέτων αφύπνισης για

χρόνο t_{spam} και στη συνέχεια την λήψη ενός πακέτου. Το κάτω κανάλι δείχνει την τάση στα άκρα ενός κόμβου, η οποία μετριέται από κατάλληλο κύκλωμα στην είσοδο τροφοδοσίας του.

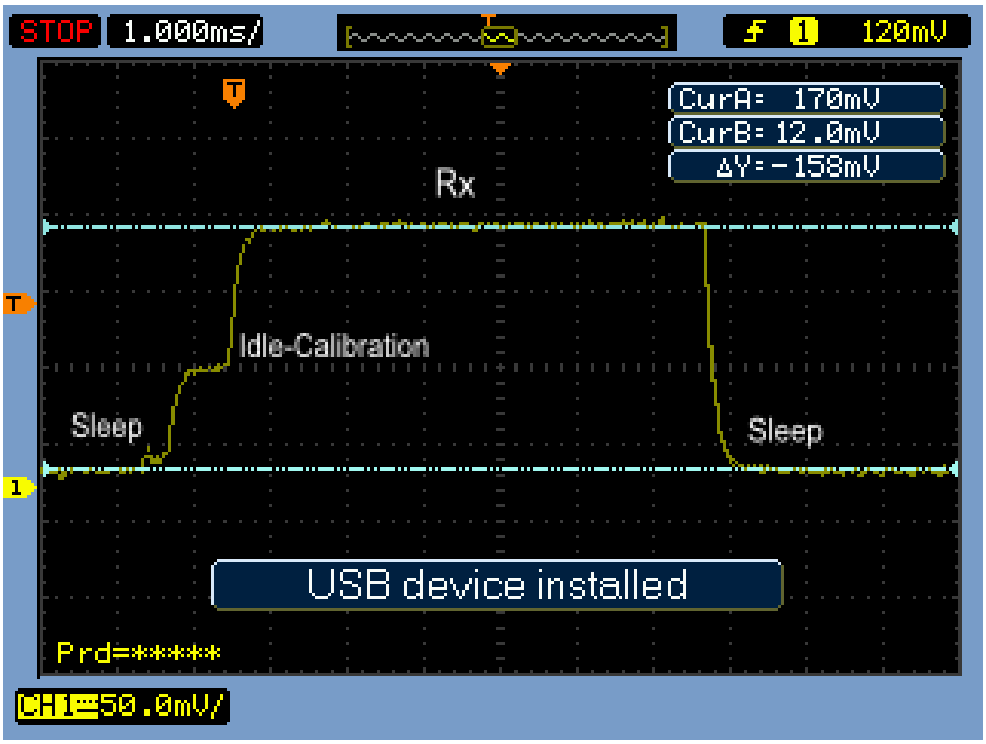
Επίσης παρατηρούμε σε αυτό το σχήμα την όλη διαδικασία, που ο κόμβος ξυπνά μέσα στο διάστημα t_{spam} από ένα πακέτο και στη συνέχεια αναμένει, για τον ίδιο χρόνο μέχρι που να στείλει το πακέτο του και ο σταθμός βάσης ακριβώς το λαμβάνει (σημειώνουμε ότι υπήρχε και άλλος κόμβος που ξύπνησε, ο σταθμός λαμβάνει 2 πακέτα, το δεύτερο είναι του κόμβου του οποίου μετράμε την κατανάλωση, καθώς συμπίπτει με την ακμή της μετρούμενης τάσης, για την αποστολή του πακέτου).

Μία ίδια εικόνα αλλά για μεγαλύτερη διάρκεια χρόνου, για τρείς εποχές, είναι η παρακάτω:



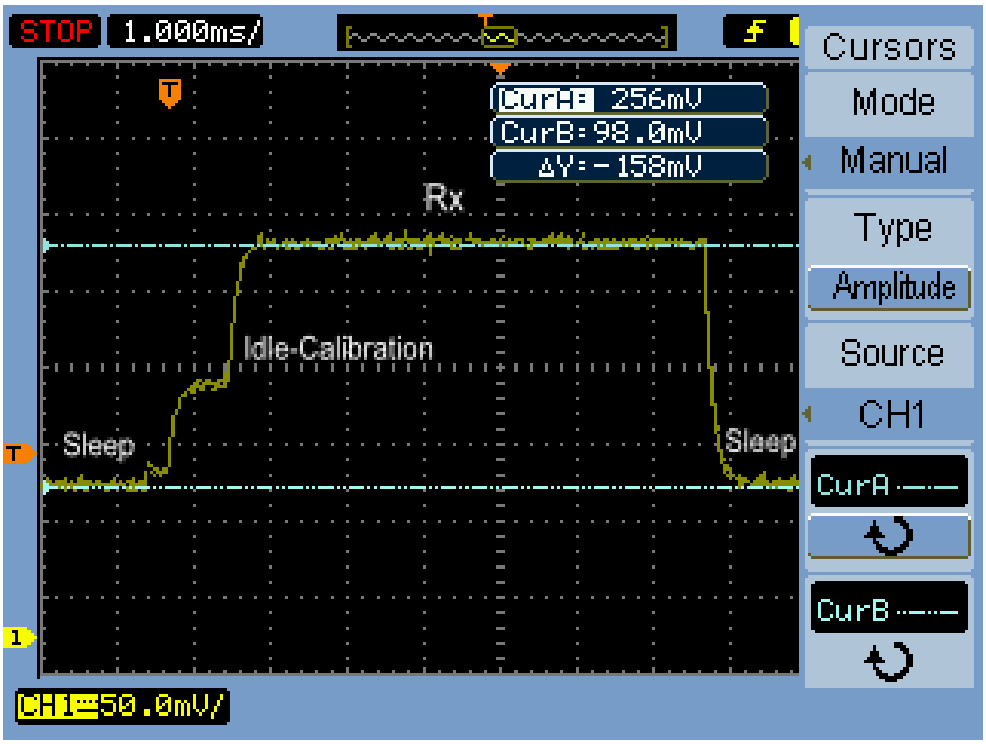
Σχήμα 18: Εικόνα Παλμογράφου - Τρείς Εποχές (epochs)

Τέλος παραθέτουμε σε μεγέθυνση χρόνου, δύο εικόνες για την κατανάλωση ρεύματος των κόμβων στις ακμές που γίνεται ο έλεγχος του καναλιού περιοδικά (Rx polling). Οι εικόνες παρουσιάζουν την τάση στα άκρα του κόμβου, οπότε για να βρεθεί η κατανάλωση σε ρεύμα, διαιρούμε την τιμή της τάσης με την τιμή της αντίστασης στην είσοδο τροφοδοσίας (στην περίπτωση μας 10 Ωhm).



Σχήμα 19: Εικόνα Παλμογράφου - Κατανάλωση ρεύματος με Idle Mode

Παρατηρούμε σε στάδια, την κατανάλωση ρεύματος να ανεβαίνει καθώς ο κόμβος μεταβαίνει από κατάσταση SLEEP του ραδιοφώνου σε κατάσταση RX. Η διαφορά είναι στα 15,8 μA. Όλη την υπόλοιπη διάρκεια ο κόμβος βρίσκεται σε κατάσταση SLEEP και με επεξεργαστή σε κατάσταση Idle Mode, με αποτέλεσμα να καταναλώνει μόνο 1,2 mA (κάτω επίπεδο).



Σχήμα 20: Εικόνα Παλμογράφου - Κατανάλωση ρεύματος χωρίς Idle Mode

Σε αυτή την περίπτωση ισχύουν τα ίδια, εκτός του ότι ο επεξεργαστής δεν μπαίνει σε κατάσταση Idle - Mode. Η διαφορά είναι μεγάλη, η κατανάλωση του κόμβου, όταν είναι σε Sleep το ραδιόφωνο μόνο (κάτω επίπεδο), είναι στα 9,8 mA σε σχέση με το 1,2 mA προηγουμένως.

5.3 Μετρήσεις - Πειράματα

Οι μετρήσεις που ακολουθούν πάρθηκαν σε πραγματικό χρόνο, από τον σταθμό βάσης και αποτελούν μερικές μετρικές, για την απόδοση του συστήματος μας. Θα χρησιμοποιούμε την ονομασία **εποχή** για κάθε επανάληψη της διαδικασίας ξυπνήματος των κόμβων και αποστολής πακέτων, όπως ορίστηκε παραπάνω.

Όπου αναφέρονται επιβεβαιώσεις (ACKs), αναφέρονται στον αριθμό των επιβεβαιώσεων που στάλθηκαν, από τον σταθμό βάσης προς τους αντίστοιχους κόμβους, επειδή τα πακέτα απαιτούσαν αυτή την λειτουργία (μπορεί μια εφαρμογή να απαιτούσε ροή πληροφορίας με σειρά κτλ).

Σχήμα 21: Δίκτυο σε λειτουργία & Σταθμός Βάσης

Οι ρυθμίσεις του πρωτοκόλλου ήταν οι πρότυπες για το 802.15.4, και στα πλαίσια που αναφέρονται στο [2]. Έτσι λοιπόν ισχύουν τα εξής:

Παράμετρος	Τιμή
macMaxNB	4
macMaxBE	5
macMinBE	3
Packet length	$2 \times D$
Backoff slot	$1 \times D$
D	10 bytes/$Tx_{rate} = 33msec$

Πίνακας 2: Χαρακτηριστικά Πρωτοκόλλου

Τονίζεται ότι στη προσπάθεια μας αυτή, δεν επιδιώκουμε να ταυτίσουμε τις μετρικές μας, με τα θεωρητικά αποτελέσματα που αναφέρονται στο [2]. Αυτό γίνεται για τους εξής λόγους, μερικοί από τους οποίους θα αναφερθούν και παρακάτω, εκτενέστερα.

- **Διαφορετικές Συνθήκες**: Υπάρχουν διαφορετικές συνθήκες καναλιού, δεν έχουμε ιδανικό κανάλι (hidden node problem, transmission errors-out of range).

- **Σφάλμα Χρόνου - Συγχρονισμού**: Υπάρχουν μικρές διαφορές στις σχισμές (backoff slots), που κάθε κόμβος έχει ώς αναφορά χρόνου, κόμβοι οι οποίοι θα διαλέγουν θεωρητικά, ίδιες σχισμές να μεταδώσουν, δεν θα προκαλούν απαραίτητα συγκρούσεις (όπως προϋποθέτει η θεωρητική απόδειξη στο σενάριο της).

  Ακόμη στο θεωρητικό μοντέλο, υπάρχει η προσέγγιση για τον αριθμό $N_c^j = N$ των κόμβων που ανταγωνίζονται το κανάλι σε κάθε σχισμή. Αυτός ο αριθμός στην πραγματικότητα μειώνεται, κατά την διάρκεια μίας εποχής, για το σενάριο όπου κάθε κόμβος μεταδίδει ένα μόνο πακέτο. Η θεώρηση του ως σταθερό αριθμό, ίσο με N (αριθμός των κόμβων), είναι προσεγγιστική.

5.3.1 Αποτελέσματα για CSMA

Οι μετρικές πάρθηκαν με 5 κόμβους να μεταδίδουν προς τον σταθμό βάσης, ανά εποχή των 6 δευτερολέπτων και για να το πετύχουμε αυτό ακολουθήθηκε, η λογική οι κόμβοι να περιμένουν το έναυσμα, από σήμα από τον σταθμό βάσης. Κάθε κόμβος πάλι μεταδίδει κάθε εποχή, ένα μόνο πακέτο (δεν περιμένει κάποια επιβεβαίωση) και αυξάνει το packet id μόνο όταν όντως στέλνει πακέτο.

Τα αποτελέσματα ήταν τα εξής:

Προέλευση	Ληφθέντα Πακέτα	Χαμένα Πακέτα
Node 2	149	11
Node 3	126	34
Node 4	130	30
Node 6	155	5
Node 7	139	21

Πίνακας 3: Μετρήσεις - Εποχές 160 (Περίοδος εποχής: 6 δευτερόλεπτα)

Από αυτά τα αποτελέσματα μπορούμε να βγάλουμε τις εξής μετρικές. Το ποσοστό επιτυχούς μετάδοσης (PDR - packet delivery ratio) ήταν:

$$PDR = \frac{\sum_{i=1}^{N} Send(i)}{\text{total\_ packs\_ transmitted}} = 0,87$$

Για την πιθανότητα επιτυχίας p_s, ισχύει ότι:

$$p_s \leq PDR = 0.87$$

Αυτό επειδή δεν εμπεριέχεται στο PDR, η πιθανότητα του κόμβου να μην κάνει πρόσβαση στο κανάλι.

Από το διάγραμμα στο [2, Fig 14] του θεωριτικού μοντέλου, παρατηρούμε ότι η πιθανότητα επιτυχίας p_s, ανάμεσα στις καμπύλες για μεγέθη πακέτων $D = 1$ έως $D = 3$, είναι στο διάστημα $0.65 - 0.73$.

5.3.2 Αποτελέσματα για CSMA & WOR

Σε αυτές τις μετρήσεις όταν αναφέρονται επιβεβαιώσεις, ο σταθμός βάσης καταγράφει κάθε φορά όσα πακέτα λαμβάνει, καταγράφει όμως τον αριθμό των επιβεβαιώσεων, μόνο όταν λαμβάνει διαφορετικά πακέτα. Οι κόμβοι που δεν έλαβαν επιτυχώς επιβεβαίωση, επαναμεταδίδουν τα ίδια πακέτα.

Έτσι έγινε μια προσπάθεια, να δούμε την επίδραση των επιβεβαιώσεων στο σύστημα μας, και πόσες αναμεταδόσεις, έγιναν λόγω συγκρούσεων ή χαμένων ACKs, που όπως αναφέρθηκε και προηγουμένως και ορίζει και το πρότυπο [1], δεν χρησιμοποιείται CSMA-CA για την μετάδοση τους.

Προέλευση	Ληφθέντα Πακέτα	Επιβεβαιώσεις (ACKs)
Node 2	528	528
Node 3	541	537
Node 4	522	520
Node 6	523	523
Node 7	526	524

Πίνακας 4: Μετρήσεις - Εποχές 614 (Περίοδος εποχής: 30 δευτερόλεπτα)

Παρατηρούμε ότι η επιβάρυνση του δικτύου μας, από επιβεβαιώσεις είναι μικρή και χάθηκαν ελάχιστες. Αυτό μας δείχνει ότι οι επιβεβαιώσεις δεν είναι απαγορευτικές. Το εξετάζουμε γιατί οι ACKs αποτελούν και μια ελαφριά, από άποψη κατανάλωσης ενέργειας εκδοχή, για την λύση του προβλήματος των κρυμμένων κόμβων, από το να χρησιμοποιηθεί, η πλήρης λύση πακέτων RTS/CTS. Επίσης δεν επιβαρύνεται το δίκτυο, με συγκρούσεις πακέτων, παρόλο που δεν χρησιμοποιείται για την αποστολή των ACKs, το πρωτόκολλο CSMA (όπως ορίζει και το πρότυπο 802.15.4).

Στην συνέχεια θέλουμε να δούμε την επιρροή (άν υπάρχει), που έχει στο σύστημα μας, ο χρόνος της περιόδου μέτρησης. Επίσης επειδή θέλουμε σε αυτή την μέτρηση να έχουμε ακριβώς την περίπτωση ενός δικτύου WSN, με ελάχιστη κατανάλωση ενέργειας δηλαδή, απενεργοποιούμε τις ACKs, ώστε να υπάρχει μόνο μία ή καμία μετάδοση ανά εποχή.

Επειδή σε μια πραγματική εφαρμογή χωραφιού, όπως η δική μας, θα χρειάζονταν μετρήσεις ανά περίπου 1 ώρα, και θέλουμε σίγουρα μέσα στη 1 ώρα αν είναι δυνατό, να έχουμε καινούργια μέτρηση, η περίοδος της εποχής μας θα είναι 30 λεπτά. Το πείραμα έτρεξε για 2 μέρες.

Προέλευση	Ληφθέντα Πακέτα
Node 2	72
Node 3	72
Node 4	88
Node 6	88
Node 7	80

Πίνακας 5: Μετρήσεις - Εποχές 96 (Περίοδος εποχής: 30 λεπτά)

Παρατηρούμε ότι ο αλγόριθμος του WOR σεναρίου, λειτούργησε και για την σποραδική αφύπνιση, των 30 λεπτών. Δεν παρουσιάστηκαν κόμβοι με πολύ μικρότερη αποστολή πακέτων από τους υπόλοιπους, το δίκτυο συμπεριφέρθηκε ανάλογα με τα προηγούμενα αποτελέσματα και οι κόμβοι αφυπνίζονταν κανονικά.

6 Συμπεράσματα

6.1 Δυσκολίες Υλοποίησης

Η εργασία αντιμετώπισε ορισμένα προβλήματα, κατά την υλοποίηση της, μερικά από τα βασικότερα, αναφέρουμε παρακάτω.

- **Μνήμη**: Η γρήγορα προσβάσιμη RAM μνήμη του μικροελεγκτή είναι μεγέθους 256 bytes και απο αυτά μόνο τα 128 bytes είναι διαθέσιμα για αποθήκευση πληροφορίας απο τον χρήστη, καθώς τα υπόλοιπα αναφέρονται στις διευθύνσεις των καταχωρητών του μικροελεγκτή. Ο μικροελεγκτής επίσης, αναγκάζει τον χρήστη στην περίπτωση, της μέτρησης χρόνου (απαραίτητη), να γίνεται αυτή μέσω επανάληψης των διακοπών υπερχείλισης μετρητών. Ο επεξεργαστής όμως, τηρεί σειρά προτεραιότητας στην εξυπηρέτηση των διακοπών.

  Αυτό σημαίνει ότι όταν έρχονται, υψηλότερης σημασίας/προτεραιότητας διακοπές, όπως για παράδειγμα η διακοπή από το περιφερειακό του ραδιοφώνου, κατά την διάρκεια λήψης ενός πακέτου, ο επεξεργαστής μπορεί και πρέπει, ορισμένες φορές να δίνει προτεραιότητα σε αυτές. Αυτό έχει σαν αποτέλεσμα ένα σφάλμα, το οποίο όμως με την αύξηση του χρόνου λειτουργίας, το οποίο μπορεί να προκαλεί σοβαρό πρόβλημα σε θέματα συγχρονισμού.

- **Απουσία λειτουργικού συστήματος (OS)**: Κάθε μορφή συγχρονισμού και μέτρησης χρόνου έγινε κυρίως με την χρήση διακοπών που δημιουργούν οι μετρητές του μικροελεγκτή και άρα υπήρχαν λάθη στην μέτρηση του χρόνου. Με την εφαρμογή ενός λειτουργικού συστήματος τέτοια προβλήματα θα λύνονταν και δεν θα απασχολούσαν την διαδικασία της υλοποίησης. Επίσης θέματα αναφοράς σε μνήμη και διαχείρισης αυτής θα γίνονταν από το λειτουργικό και όχι από τον χρήστη.

  Για παράδειγμα το κυριότερο ζήτημα διαχείρισης της μνήμης είναι ότι ο compiler αλλά και ο ίδιος ο μικροελεγκτής επιτρέπουν την άμεση αναφορά και διαχείριση, στη μνήμη RAM. Στο μεγαλύτερο κομμάτι της μνήμης δηλαδή της FLASH, γίνεται αναφορά μόνο

με την εντολή μηχανής (MOVX). Αυτή η διαχείριση μνήμης δεν προτείνεται για χρήση αποθήκευσης διαρκώς μεταβλητών δεδομένων και δημιουργεί ασυνέπεια στην μνήμη. Ένα λειτουργικό σύστημα θα μπορούσε να προσφέρει μία αποτελεσματική μορφή διαχείρισης της μνήμης.

6.2 Μελλοντική Δουλειά

Σαν στόχους μελλοντικούς η εργασία αυτή, έχει τα εξής:

- **Εξακρίβωση του θεωρητικού μοντέλου**: Επειδή οι μετρήσεις στα πλαίσια αυτής της εργασίας δεν επαρκούν για να μπορούν να έχουν κάποιο συμπέρασμα και σύγκριση με τη θεωρία, θα πρέπει να παρθούν μετρήσεις για μεγάλο διάστημα χρόνου, πολλών εποχών και θα πρέπει να δημιουργηθεί το κατάλληλο περιβάλλον, όπως περιγράφεται στο θεωρητικό μοντέλο, έτσι ώστε να παρθούν ακριβείς μετρήσεις. Βασικότερο σημείο για αυτό αλλά και για άλλες θεωρητικές αναλύσεις είναι να υπάρχει μία εξωτερική ροή πακέτων/πληροφορίας, από ένα επίπεδο εφαρμογής application layer, σταθερή και πλήρως ρυθμιζόμενη.

- **Αφύπνιση με Carrier Sense**: Η αφύπνιση μέσω πακέτων όπως είδαμε απαιτεί την αποστολή για αρκετό χρόνο και αυτό καταναλώνει ενέργεια στον σταθμό βάσης. Θα πρέπει λοιπόν να αναλυθούν, περισσότερες μέθοδοι για την σκοπιά της αφύπνισης με λίγα πακέτα ίσως και 1, και την χρήση του Carrier Sense. Να εφαρμοστεί δηλαδή ένας αποτελεσματικός αλγόριθμος, για την περίπτωση που ο σταθμός βάσης, δεν μπορεί να είναι ενεργειακά αυτόνομος.

- **Υλοποίηση Beaconed - enabled δικτύου** : Όπως προαναφέρθηκε το πρότυπο 802.15.4 ορίζει και την επιλογή για beaconed-enabled δίκτυα τα οποία βασίζονται στον slotted-CSMA αλγόριθμο. Για εφαρμογές περισσότερο συγχρονισμένες. Για να επιτευχθεί κάτι τέτοιο, απαιτείται δουλειά στο πεδίο του συγχρονισμού των κόμβων (να έχουν δηλαδή οι μετρητές, του χρόνου τους, κοινή βάση και να είναι ευθυγραμμισμένοι), είτε αυτός γίνει με την εισαγωγή λειτουργικού συστήματος, είτε με εφαρμογή άλλων

αλγορίθμων, όπου υπάρχουν πολλοί. Από άποψη υλοποίησης του Beaconed - enabled πρωτοκόλλου, η εργασία αυτή έχει θέσει όλα τα απαραίτητα θεμέλια και δεν απαιτείται σημαντική δουλειά.

- **Μνήμη**: Να γίνουν δοκιμές αποθήκευσης δεδομένων και προσωπικής διαχείρισης και αξιοποίησης της μνήμης FLASH, έτσι ώστε ο κόμβος βάσης, ή διάφοροι κόμβοι που θέλουμε στο δίκτυο μας, να μπορούν για παράδειγμα να αποθηκεύουν περισσότερη πληροφορία.

Α΄ CSMA-CA - Κώδικας

Σε αυτό το παράρτημα αναφέρονται οι κώδικες packet.h, CSMA.h, csmaSendPacket.c, csmaReceivePacket.c, csma_ init.c, οι οποίοι αποτελούν τον βασικότερο κορμό για την υλοποίηση του πρωτοκόλλου στο δικό μας υλικό. Στο packet.h υπάρχουν ορισμοί για την μορφοποίηση ενός πακέτου, τύποι λαθών κτλ. Στο CSMA.h υπάρχουν οι απαραίτητοι ορισμοί για το πρωτόκολλο CSMA. Τα csmaSendPacket.c και csmaReceivePacket.c παρέχουν τις βασικές λειτουργίες αποστολής και λήψης για το πρωτόκολλο.

Πολλοί από αυτούς χρησιμοποιούν ίσως και κώδικα από άλλες βιβλιοθήκες, όπως αναφέρθηκε σύμφωνα και με την αρχιτεκτονική HAL(Hardware Abstraction Layer) αλλά εδώ θα αναφέρουμε μόνο τον βασικό κορμό.

```
1  //-----------------------------------------------------------------------
   // packet.h
3  //-----------------------------------------------------------------------
   // AUTH: Ioannis Papamentzelopoulos
5  // DATE: 4 Ioun 2012
   //
7  // Target: C8051F32x
   // Tool chain: KEIL C51 7.06
9  //
   // Release 1.0
11 //
   //-----------------------------------------------------------------------
13 // Includes
   //-----------------------------------------------------------------------
15 #ifndef PACKET_H_
   #define PACKET_H_
17 #include "other/compiler_defs.h"
   #include "other/common.h"
19 //-----------------------------------------------------------------------
```

```
// Structures, Unions, Enumerations, and Type Definitions
//-------------------------------------------------------------------
//===================================//
//  Header Parameters    //
//===================================//
#define TX_IDX_DESTINATION 1  // Index of Receiver address  in
    TxBuffer
#define TX_IDX_PACKET_TYPE 2  // Index of packet id1 in TxBuffer
#define TX_IDX_SOURCE  3  // Index of Transmitter address in TxBuffer
#define TX_IDX_PACKET_ID 4  // Index of packet id2  in TxBuffer
#define TX_IDX_ACK_REQUIRE 5  // Index of ACK request in TxBuffer
#define TX_IDX_QLENGTH  6  // Index of Queue length for BackPressure
    Routing
#define RX_IDX_DESTINATION 0  // Index of Receiver address  in
    Rxbuffer
#define RX_IDX_PACKET_TYPE 1  // Index of packet id1 in Rxbuffer
#define RX_IDX_SOURCE  2  // Index of Transmitter address in Rxbuffer
#define RX_IDX_PACKET_ID 3  // Index of packet id2  in Rxbuffer
#define RX_IDX_ACK_REQUIRE 4  // Index of ACK request in Rxbuffer
#define RX_IDX_QLENGTH  5  // Index of Queue length for BackPressure
    Routing
//===================================//
//   Packet Types    //
//===================================//
#define PTYPE_DATA   0
#define PTYPE_ACK    1
#define PTYPE_RTS  2
#define PTYPE_CTS  3
#define PTYPE_FORWARD 4
//===================================//
```

```
//  Packet Error Codes     //
//=======================================//
#define PACKET_TIMEOUT      0
#define PACKET_SUCCESS      1
#define PACKET_CORRUPTED    2
#define PACKET_WRONG_NODEID   3
#define PACKET_WRONG_LENGTH   4
#define PACKET_WRONG_TYPE   5
#define PACKET_WRONG_PACKID   6
#define PACKET_ERROR_OTHER   254
//=======================================//
//  Buffer Information     //
//=======================================//
#define BUFF_ACK_REQUIRED     1
#define BUFF_ACK_NOT_REQUIRED    0
//-----------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------
#endif /* PACKET_H_ */
```

```
//-------------------------------------------------------------------
// CSMA.h
//-------------------------------------------------------------------
// AUTHOR: Ioannis Papamentzelpoulos
// DATE: Jun 1, 2012
//
// Target: C8051F32x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//
//-------------------------------------------------------------------
// Includes
//-------------------------------------------------------------------
#ifndef CSMA_H_
#define CSMA_H_
#include "other/compiler_defs.h"
#include "other/common.h"
#include "other/math_lib.h"
#include "motes/icubes/phy/halRF.h"
#include "motes/icubes/hal/halSleep.h"
#include "motes/icubes/hal/halTimers.h"
#include "motes/icubes/component_defs/icubes_defs.h"
#include "NODEID.h"
#include <stdlib.h>
//-------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-------------------------------------------------------------------
//====================================//
//  CSMA - Parameters    //
```

```
31 //=========================================//
   #define CCA_SLOTS     1  // Number of slots node will wait for CCA
33 #define ACK_SLOTS     2  // Number of slots node will wait for ACK
   #define MAX_BE      5  // Max number of back-off exponential window
35 #define MAX_RETRANSMITS   5  // Max number of retransmissions when
       waiting for ACK
   #define MAX_BACKOFFS    5  // MAx number of tries to access channel
37 #define INIT_BE      3  // Initialize Back-off exponential window
   #define TX_SLOTS     2  // Period of Tx of your packet in slots
39 #define SLOT_TIME_INTS    330  // Back-off Slot time          //duration
       in interrupts of timer selected in csmaInit() routine.
   //=========================================//
41 //     Clear Channel Assessment  //
   //=========================================//
43 #define TYPE_CCA_CARRIER_BASED  0  // Polling the channel for any
       signal
   #define TYPE_CCA_SYNC_BASED  1  // Polling the channel for packets
45           //with sync word
   #define TYPE_CCA TYPE_CCA_CARRIER_BASED
47 //-------------------------------------------------------------------
       // Exported Function Prototypes
   //-------------------------------------------------------------------
      TODO question ? Why not by reference txBuff or rxBuff
49 BOOL csmaSendPacket(BYTE packet_type, BYTE destination, BYTE packet_id
       , BYTE ack_requirement, BYTE init_BE);
   BYTE csmaReceivePacket(BYTE packet_type, UINT16 timeout, BYTE
       packet_id);
51 bit csmaInit(BYTE sync_timer_num);
   //utility functions in csmaInit.c
53 void csmaBackOff(BYTE backoff_exp);
```

```
BOOL csmaCCA(BYTE cca_type,BYTE timeout);
#endif /* CSMA_H_ */
```

```
/*
 * csmaSendPacket.c
 *
 *  Created on: Jun 1, 2012
 *      Author: Ioannis Papamentzelopoulos
 */
//-----------------------------------------------------------------
// Includes
//-----------------------------------------------------------------
#include "other/compiler_defs.h"
#include "other/common.h"
#include <stdio.h>
#include "motes/icubes/mac/CSMA.h"
#include "motes/icubes/mac/packet.h"
//-----------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------
//-----------------------------------------------------------------
// csmaSendPacket()
//
// Basic Routine used to send any type of packet(data,ACK,RTS,CTS).
// It uses the CSMA access mechanism, to access channel
//-----------------------------------------------------------------
BOOL csmaSendPacket(BYTE packet_type, BYTE destination, BYTE packet_id
    , BYTE ack_requirement, BYTE init_BE){
 BOOL send_finished;
 BYTE backoff_exp;
 BYTE retransmits;
 BYTE num_Backoffs;
 BYTE ack_received;
```

```
//------------------ INITIALIZATION ---------------------//
txBuffer[TX_IDX_DESTINATION] = destination;
txBuffer[TX_IDX_PACKET_TYPE] = packet_type;
txBuffer[TX_IDX_PACKET_ID]   = packet_id;
txBuffer[TX_IDX_ACK_REQUIRE]  = ack_requirement;
//Setup header of txBuffer
if(packet_type != PTYPE_FORWARD){
 txBuffer[TX_IDX_SOURCE]    = NODEID;
}
//Parameters
send_finished       = FALSE;
if(init_BE==DONT_CARE){
 backoff_exp      = INIT_BE;
}else{
 backoff_exp      = init_BE;
}
retransmits       = MAX_RETRANSMITS;
num_Backoffs      = MAX_BACKOFFS;
//-----------------SEND ROUTINE----------------------//
if(packet_type == PTYPE_DATA || packet_type == PTYPE_RTS ||
   packet_type == PTYPE_FORWARD){ //-------------------- DATA / RTS
 csmaBackOff(backoff_exp); //an initial Backoff for the first access
 do{
  if(csmaCCA(TYPE_CCA, SLOT_TIME_INTS * CCA_SLOTS)){
   //here channel is clear
   backoff_exp = INIT_BE; //re-initialize backoff threshold channel
      was clear
   retransmits --;   //decrease maximum number of retransmissions
   halRfSendPacket(txBuffer, sizeof(txBuffer));
   if(ack_requirement == BUFF_ACK_REQUIRED){
```

```
      //enter RX-mode to receive ACK
59    //------------WAIT ACK--------------------
      do{
61     if(packet_type==PTYPE_RTS){
        ack_received = csmaReceivePacket(PTYPE_CTS,(SLOT_TIME_INTS*
            TX_SLOTS*ACK_SLOTS),packet_id);
63     }else{
        ack_received = csmaReceivePacket(PTYPE_ACK,(SLOT_TIME_INTS*
            TX_SLOTS*ACK_SLOTS),packet_id);
65     }
      }while(     ack_received == PACKET_CORRUPTED
67      || ack_received == PACKET_WRONG_LENGTH
        || ack_received == PACKET_WRONG_NODEID
69      || ack_received == PACKET_WRONG_TYPE
        || ack_received == PACKET_WRONG_PACKID
71      || ack_received == PACKET_ERROR_OTHER);
      if(ack_received == PACKET_TIMEOUT){
73     continue;
      }else if(ack_received == PACKET_SUCCESS){
75     send_finished = TRUE;
      }
77    //-------------------------------------------------------------
     }else{
79    send_finished = TRUE;
     }
81  }else{
     num_Backoffs--;  //decrease number of left tries to access channel
83   if(backoff_exp<MAX_BE){
      backoff_exp++;
85   }
```

```
    /*If Channel is busy the node is waiting in IDLE for a rand time
       */
87    csmaBackOff(backoff_exp);
    }
89  }while((send_finished == FALSE) && (retransmits != 0) && (
      num_Backoffs !=0));
    return send_finished;
91  }else{ //--------------- ACK / CTS
    txBuffer[TX_IDX_DESTINATION]  = destination;
93  txBuffer[TX_IDX_SOURCE]    = NODEID;
    txBuffer[TX_IDX_PACKET_ID]    = packet_id;
95  txBuffer[TX_IDX_PACKET_TYPE]  = packet_type;
    halRfSendPacket(txBuffer, sizeof(txBuffer));
97  return TRUE;
  }
99}
```

```
/*
 * csmaReceivePacket.c
 *
 *  Created on: Jun 1, 2012
 *      Author: Ioannis Papamentzelopoulos
 */
//-------------------------------------------------------------------------
// Includes
//-------------------------------------------------------------------------
#include "other/compiler_defs.h"
#include "other/common.h"
#include "motes/icubes/phy/halRF.h"
#include "motes/icubes/mac/CSMA.h"
#include "motes/icubes/mac/packet.h"
#include "NODEID.h"
//-------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
// Global Constants
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
// csmaReceivePacket()
//-------------------------------------------------------------------------
BYTE csmaReceivePacket(BYTE packet_type, UINT16 timeout, BYTE
    packet_id){
 BYTE transmitter_id;
 BYTE length;
 BYTE receive_status;
 //set up the packet length in the txBuffer
```

```
length = sizeof(rxBuffer);
receive_status = halRfReceivePacketTimeout(rxBuffer,&length,timeout);
if(receive_status != PACKET_SUCCESS){
 return receive_status;
}
switch (packet_type){
case PTYPE_DATA:
 if((rxBuffer[RX_IDX_PACKET_TYPE] != PTYPE_DATA)&&(rxBuffer[
    RX_IDX_PACKET_TYPE] != PTYPE_FORWARD)&&(rxBuffer[
    RX_IDX_PACKET_TYPE] != PTYPE_RTS)){
   return PACKET_WRONG_TYPE;
 }
 transmitter_id = rxBuffer[RX_IDX_SOURCE];
 return PACKET_SUCCESS;
 break;
case PTYPE_ACK:
 if(rxBuffer[RX_IDX_PACKET_TYPE] != PTYPE_ACK){
  return PACKET_WRONG_TYPE;
 }
 /* note that halRfReceivePacket will break and return false if
    received packet
 is discarded due to address checking.THis is how it is handled by
    cc25000 radio's
 hardware(Our address checking is hardware based).  */
 if(rxBuffer[RX_IDX_PACKET_ID]== packet_id){    //check if ACK
    arrived in correct
  return PACKET_SUCCESS;
 }else{  //if ACK is is for another send packet,(waited for #2 but
    received #3)
  return PACKET_WRONG_PACKID;
```

```
    }
55  /* At this point it is certain that halRfReceivePacket has returned
       false
     which means it either broke due to timeout event or due to address
         check.
57   If it is due to address check it means that another node or the
        base station
        speaks in the channel so the node has to listen to the channel
           again until
59   it receives a valid ACK or has a timeout event.On a timeout event
     we clear the timer and we print out for debug reasons.
61  */
    break;
63  case PTYPE_CTS:
    if(rxBuffer[RX_IDX_PACKET_TYPE] != PTYPE_CTS){
65   return PACKET_WRONG_TYPE;
    }
67  return PACKET_SUCCESS;
    break;
69  default :
    return PACKET_ERROR_OTHER;
71  break;
  }
73 return FALSE;
}
```

```
/*
 * csma_init.c
 *
 *  Created on: Jun 2, 2012
 *      Author: Ioannis Papamentzelopoulos
 */
//-----------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------
#include "other/compiler_defs.h"
#include "other/common.h"
#include "motes/icubes/mac/CSMA.h"
//-----------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------
UINT16 cca_counter = 0;
bit cca_event_enabled = 0;
void csmaCounterEvent(void);
//-----------------------------------------------------------------------
// csma_init()
//-----------------------------------------------------------------------
bit csmaInit(BYTE sync_timer_num)
{
 bit status = 0;
 //initialize with transmitter's ID srand to get numbers
 srand(NODEID);
 // Init physical layer
```

```
  if(!init_halRfReceivePacketTimeout(sync_timer_num)){
32    while(1);
  }
34  //init sleep mode of mcu
  if(!init_halRfSleepTimer(sync_timer_num)){
36   while(1);
  }
38  //TODO setup for multiple timers
  switch (sync_timer_num){
40  case TIMER_0:
   NOP();
42   break;
  case TIMER_1:
44   NOP();
   break;
46  case TIMER_2:
   status = acquire_timer2_event( csmaCounterEvent );
48   break;
  case TIMER_3:
50   NOP();
   break;
52  default:
   return 0;
54  }
  return status;
56 }
  //TODO Comments
58 void csmaBackOff(BYTE backoff_exp){
  UINT16 rand_slot;
60  UINT16 window;
```

```
 UINT16 back_off_time;
62 /* Compute the Slot time interval to wait*/
 window = pow2(backoff_exp);  //window is computed from current
    backoff-num each time
64 rand_slot = (rand() % window); //pseudo-random value of window
 back_off_time = (window-1 - rand_slot)*SLOT_TIME_INTS; //compute the
    number of slots to Idle
66 // back_off_time = rand_slot * SLOT_TIME_INTS;
 //WARNING halSleepInit() has to be called first in main!!!
68 halSleep1(back_off_time);        //sleep for back_off_time (this is
    in ints of timer selected in csmaInit)
}
70 //-------------------------------------------------------------------
// Function used to perform CCA(clear channel assessment)
72 // There are 2 options:
// -Based on receiving a packet (SYNC_BASED)[resilient,not accurate]
74 // -Based on sensing the channel(CARRIER_BASED)[sensitive,accurate]
//-------------------------------------------------------------------
76 BOOL csmaCCA(BYTE cca_type, BYTE timeout){
 halSpiStrobe(CCxxx0_SRX); //enter RX-mode to listen channel
78 //timeout = (slot duration * number of slots) * timer interrupt
    period
 //TODO calculate the min time of sync word to receive
80 cca_counter = timeout;
 cca_event_enabled = 1;
82 do{   // Poll the channel for a period of time equal to CCA_SLOTS.
  // if even once the GD0O_PIN is 1 ,which means someone transmits
84  // the channel is dimmed occupied and the node will have to wait
  //WARNING: GDO0 PIN MUST BE SET AS SYNC WORD POLLING MODE IN RADIO
     SETTINGS (value:0x06)
```

```
86    if(cca_type == TYPE_CCA_SYNC_BASED){
       if(P_GDO0){
88      halSpiStrobe(CCxxx0_SIDLE);
        halSpiStrobe(CCxxx0_SFRX);
90      cca_event_enabled = 0;
        return FALSE;
92     }
     //WARNING: GDO2 PIN MUST BE SET AS POLLING CCA MODE IN RADIO
        SETTINGS (value:0x0E)
94    }else if (cca_type == TYPE_CCA_CARRIER_BASED){
       if(P_GDO2){
96      halSpiStrobe(CCxxx0_SIDLE);
        halSpiStrobe(CCxxx0_SFRX);
98      cca_event_enabled = 0;
        return FALSE;
100    }
      }else{
102    //HALT
       while(1);
104   }
    }while(cca_counter > 0);
106 cca_event_enabled = 0;
    //if during the time slot no transmittion was found
108 // then we assume the channel free and the node will be able
    // to transmit.
110 //TODO check if this is necessary
    halSpiStrobe(CCxxx0_SIDLE);//return radio to idle mode
112 halSpiStrobe(CCxxx0_SFRX); // flush receive fifo
    return TRUE;
114 }
```

```
//-------------------------------------------------------
// Routine used for to measure time using the
// interrupts of a timer chosen
//-------------------------------------------------------
void csmaCounterEvent(void){
 if(cca_event_enabled && cca_counter > 0){
  cca_counter--;
 }
}
```

Β′ Wake-On-Radio Κώδικας

Σε αυτό το παράρτημα αναφέρονται οι κώδικες WOR_ INIT,WOR_ RESET, halWakeOnInt οι οποίοι περιγράφουν βασικές λειτουργίες, πού έχουν να κάνουν με λειτουργίες τις λειτουργίες του WOR. Πολλοί από αυτούς χρησιμοποιούν ίσως και κώδικα από άλλες βιβλιοθήκες, όπως αναφέρθηκε σύμφωνα και με την αρχιτεκτονική HAL(Hardware Abstraction Layer), αλλά εδώ θα αναφέρουμε μόνο τον βασικό κορμό.

```
//------------------------------------------------------------------------------
//  WOR_INIT()
//
//  DESCRIPTION:
//      This macro is used to configure the WOR function of the radio
//
// Note:
//      wor_event
//   A 16bit value for the EVENT0 time configuration. The period each
    //   SLEEP-TO-IDLE-RX happens. Default max value 65535 gives 1,86
   sec for
//   26 Mhz osci of cc2500.
// ARGUMENTS:
//  RX_duty
//   The percentage of the EVENT time the radio stays in RX to listen
   for //   packets.
//------------------------------------------------------------------------------
#define RX_DUTY_12_5 0x00
#define RX_DUTY_6_2  0x01
#define RX_DUTY_3_1  0x02
#define RX_DUTY_1_5  0x03
#define WOR_INIT(x)\
  do{\
```

```
    halSpiWriteReg(CCxxx0_WORCTRL,0x38);\
    halSpiWriteReg(CCxxx0_WOREVT1,0xFF);\
    halSpiWriteReg(CCxxx0_WOREVT0,0xFF);\
    halSpiWriteReg(CCxxx0_MCSM2,x);\
    WOR_RESET();\
  }while(0)
```

```
//-------------------------------------------------------------------
//  WOR_RESET()
//
//  DESCRIPTION:
//      This macro is used to reset the WOR configuration and re-enter
    radio in WOR mode
//
//  ARGUMENTS:
//      None
//-------------------------------------------------------------------
#define WOR_RESET()\
 do{\
  halSpiStrobe(CCxxx0_SWORRST);\
  halSpiStrobe(CCxxx0_SWOR);\
 }while(0)
```

```
/*
 * halSleepWakeOnInt.c
 *
 *  Created on: Jun 6, 2012
 *      Author: Ioannis Papamentzelopoulos
 */
//-----------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------
#include "other/compiler_defs.h"
#include "other/common.h"
#include "motes/icubes/hal/halInterrupts.h"
#include "motes/icubes/hal/halSleep.h"
//-----------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// halSleepWakeOnInt()
//
// This function is used to keep the C8051F320 at low-power mode by
   setting
// its clock to the lower frequency, disabling all external/interrupts
    interrupts
// but one (the one that will wake him up), and entering Idle Mode.
//
// ARGUMENTS:
//   BYTE int_num
//     The num of INT1 or INT0 external interrupts, or any other
//     interrupt that WON'T BE DISABLED(this will wake you up).
//
```

```
//-----------------------------------------------------------------------
// Block until an int wakes me up
// TODO documetation
void halSleepWakeOnInt(BYTE int_num){

BYTE old_int_enables;
BYTE old_clksel;
BYTE old_oscicn;
// Keep all int and clock states
old_int_enables = IE;
old_clksel = CLKSEL;
old_oscicn = OSCICN;
//Disable all interrupts
IE = 0x00;
// Set up the wake up interrupt
INT_ENABLE(int_num, INT_ON);
// Enable global ints
ENABLE_GLOBAL_INT(INT_ON);
// Set the clock to its lowest frequency (select USB clock @ internal
   /2)
CLKSEL = 0x00;
OSCICN &= ~0x03;
// Set mcu to idle mode
PCON |= 0x01;
/*************************************************************/
// At this point, the mcu has been waken up by the selected interrupt
/*************************************************************/
// Reset the int and clock states
IE = old_int_enables;
CLKSEL = old_clksel;
```

```
58 OSCICN = old_oscicn;
   }
```

Γ′ Phy-Layer Κώδικας

Σε αυτό το παράρτημα αναφέρονται οι κώδικες halRfReceivePacketWOR.c, halRfReceivePacketTimeout.c, RFSettings.c, οι οποίοι περιγράφουν βασικές λειτουργίες, όπως λήψη πακέτων ενώ οι κόμβοι βρίσκονται σε κατάσταση WOR, είναι απαραίτητοι για την υλοποίηση του πρωτοκόλλου, και έχουν να κάνουν με λειτουργίες του φυσικού επιπέδου. Πολλοί από αυτούς χρησιμοποιούν ίσως και κώδικα από άλλες βιβλιοθήκες, όπως αναφέρθηκε σύμφωνα και με την αρχιτεκτονική HAL(Hardware Abstraction Layer), αλλά εδώ θα αναφέρουμε μόνο τον βασικό κορμό.

```
1  //------------------------------------------------------------------
   //  BOOL halRfReceivePacketWOR(BYTE *rxBuffer, UINT8 *length)
3  //  NOTE: This is the common halRFReceive() but without entering RX,
       used for WOR. Entering RX ruins WOR
   //          configuration and timing of cc2500. It also provides the
       choice to drop or keep the packet
5  //      received.
   //  DESCRIPTION:
7  //      This function can be used to receive a packet of variable
       packet length (first byte in the packet
   //      must be the length byte). The packet length should not exceed
       the RX FIFO size.
9  //      To use this function, GD00 must be configured to be asserted
       when sync word is sent and
   //      de-asserted at the end of the packet => halSpiWriteReg(
       CCxxx0_IOCFG0, 0x06);
11 //      Also, APPEND_STATUS in the PKTCTRL1 register must be enabled.
   //      The function implements polling of GDO0. First it waits for
       GD00 to be set and then it waits
13 //      for it to be cleared.
   //      After the GDO0 pin has been de-asserted, the RXBYTES register
```

```
//      is read to make sure that there
//      are bytes in the FIFO. This is because the GDO signal will
    indicate sync received even if the
//      FIFO is flushed due to address filtering, CRC filtering, or
    packet length filtering.
//
//  ARGUMENTS:
//      BYTE *rxBuffer
//          Pointer to the buffer where the incoming data should be
    stored
//      UINT8 *length
//          Pointer to a variable containing the size of the buffer
    where the incoming data should be
//          stored. After this function returns, that variable holds
    the packet length.
//  BOOL ignore_packet
//    Discard the packet received or keep it in the storage buffer.
//
//  RETURN VALUE:
//      BOOL
//          TRUE:   CRC OK
//          FALSE:  CRC NOT OK (or no packet was put in the RX FIFO
    due to filtering)
//-------------------------------------------------------------------
BOOL halRfReceivePacketWOR(BYTE *rxBuffer, UINT8 *length, BOOL
    ignore_packet) {
    BYTE status[2];
    UINT8 packetLength;
    // Wait for GDO0 to be set -> sync received
    while (!GDO0_PIN);
```

```
37      // Wait for GDO0 to be cleared -> end of packet
        while (GDO0_PIN);
39      if(!ignore_packet){
    // This status register is safe to read since it will not be updated
        after
41  // the packet has been received (See the CC1100 and 2500 Errata Note
        )
    if ((halSpiReadStatus(CCxxx0_RXBYTES) & BYTES_IN_RXFIFO) ) {
43   // Read length byte
     packetLength = halSpiReadReg(CCxxx0_RXFIFO);
45   // Read data from RX FIFO and store in rxBuffer
     if (packetLength <= *length) {
47    halSpiReadBurstReg(CCxxx0_RXFIFO, rxBuffer, packetLength);
      *length = packetLength;
49    // Read the 2 appended status bytes (status[0] = RSSI, status[1] =
          LQI)
      halSpiReadBurstReg(CCxxx0_RXFIFO, status, 2);
51    // MSB of LQI is the CRC_OK bit
      return (status[LQI] & CRC_OK);
53   } else {
      *length = packetLength;
55    // Make sure that the radio is in IDLE state before flushing the
         FIFO
      // (Unless RXOFF_MODE has been changed, the radio should be in
         IDLE state at this point)
57    halSpiStrobe(CCxxx0_SIDLE);
      // Flush RX FIFO
59    halSpiStrobe(CCxxx0_SFRX);
      return FALSE;
61   }
```

```
  } else
63   return FALSE;
    }else{
65  // Make sure that the radio is in IDLE state before flushing the
      FIFO
  // (Unless RXOFF_MODE has been changed, the radio should be in IDLE
      state at this point)
67  halSpiStrobe(CCxxx0_SIDLE);
  // Flush RX FIFO
69  halSpiStrobe(CCxxx0_SFRX);
  return TRUE;
71   }
}// halRfReceivePacketWOR
```

```
/*
 * halRfReceivePacketTimeout.c
 *
 *  Created on: Jun 1, 2012
 *      Author: Ioannis Papamentzelopoulos
 */
//------------------------------------------------------------------------------
// Includes
//------------------------------------------------------------------------------
#include "other/compiler_defs.h"
#include "other/common.h"
#include "motes/icubes/hal/halInterrupts.h"
#include "motes/icubes/hal/halTimers.h"
#include "motes/icubes/phy/halRF.h"
#include "motes/icubes/component_defs/icubes_defs.h"
#include "motes/icubes/mac/packet.h"
void timeout_counter_routine(void);
//------------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//------------------------------------------------------------------------------
#define CRC_OK              0x80
#define GDO0_PIN            P0_6
#define RSSI                0
#define LQI                 1
#define BYTES_IN_RXFIFO     0x7F
//------------------------------------------------------------------------------
// Global Constants
//------------------------------------------------------------------------------
UINT16 timeout_counter = 0;
bit rxtimeout_event_enabled = 0;
```

```
bit init_halRfReceivePacketTimeout(BYTE timer_num){
 bit status = 0;
 //TODO setup for multiple timers
 switch (timer_num){
  case TIMER_0:
   NOP();
   break;
  case TIMER_1:
   NOP();
   break;
  case TIMER_2:
   status = acquire_timer2_event( timeout_counter_routine );
   break;
  case TIMER_3:
   NOP();
   break;
  default:
   return 0;
 }
 return status;
}
//-------------------------------------------------------------------
// halRfReceivePacketTimeout()
//
//  NOTE:  This function is used to wait for a chosen amount of time
   to receive a
//     packet. After that radio switches back to IDLE state
//  DESCRIPTION:
//      This function can be used to receive a packet of variable
   packet length (first byte in the packet
```

```
//      must be the length byte). The packet length should not exceed
    the RX FIFO size.
//      To use this function, GD00 must be configured to be asserted
    when sync word is sent and
//      de-asserted at the end of the packet => halSpiWriteReg(
    CCxxx0_IOCFG0, 0x06);
//      Also, APPEND_STATUS in the PKTCTRL1 register must be enabled.
//      The function implements polling of GDO0. First it waits for
    GD00 to be set and then it waits
//      for it to be cleared.
//      After the GDO0 pin has been de-asserted, the RXBYTES register
    is read to make sure that there
//      are bytes in the FIFO. This is because the GDO signal will
    indicate sync received even if the
//      FIFO is flushed due to address filtering, CRC filtering, or
    packet length filtering.
//
//  ARGUMENTS:
//      BYTE *rxBuffer
//          Pointer to the buffer where the incoming data should be
    stored
//      UINT8 *length
//          Pointer to a variable containing the size of the buffer
    where the incoming data should be
//          stored. After this function returns, that variable holds
    the packet length.
//  UINT16 timeout
//   the amount of time to wait in RX measured in interrupts of a
    chosen timer.
//
```

```
78 //  RETURN VALUE:
   //      BOOL
80 //          TRUE:   CRC OK
   //          FALSE:  CRC NOT OK (or no packet was put in the RX FIFO
      due to filtering)
82 //-----------------------------------------------------------------
   //TODO return status[rssi]
84 BYTE halRfReceivePacketTimeout(BYTE *rxBuffer, UINT8 *length, UINT16
      timeout){
    BYTE status[2];
86  UINT8 packetLength;
    halSpiStrobe(CCxxx0_SRX);
88  timeout_counter = timeout;
    rxtimeout_event_enabled = 1;
90  // Wait for GDO0 to be set -> sync received
    while (  ( (!GDO0_PIN) && (timeout_counter > 0) ) );
92  rxtimeout_event_enabled = 0;
    if(timeout_counter > 0 && GDO0_PIN){
94   // Wait for GDO0 to be cleared -> end of packet
     while (GDO0_PIN);
96   // This status register is safe to read since it will not be updated
        after
     // the packet has been received (See the CC1100 and 2500 Errata Note
        )
98   if ((halSpiReadStatus(CCxxx0_RXBYTES) & BYTES_IN_RXFIFO)) {
      // Read length byte
100   packetLength = halSpiReadReg(CCxxx0_RXFIFO);
      // Read data from RX FIFO and store in rxBuffer
102   if (packetLength <= *length) {
       halSpiReadBurstReg(CCxxx0_RXFIFO, rxBuffer, packetLength);
```

```
104     *length = packetLength;
        // Read the 2 appended status bytes (status[0] = RSSI, status[1] =
            LQI)
106     halSpiReadBurstReg(CCxxx0_RXFIFO, status, 2);
        // MSB of LQI is the CRC_OK bit
108     if(status[LQI] & CRC_OK){
         return PACKET_SUCCESS;
110     }else{
         return PACKET_CORRUPTED;
112     }
      } else {
114     *length = packetLength;
        // Make sure that the radio is in IDLE state before flushing the
           FIFO
116     // (Unless RXOFF_MODE has been changed, the radio should be in
           IDLE state at this point)
        halSpiStrobe(CCxxx0_SIDLE);
118     // Flush RX FIFO
        halSpiStrobe(CCxxx0_SFRX);
120     return PACKET_WRONG_LENGTH;
      }
122   } else
      return PACKET_WRONG_NODEID;
124  }else  // control if
     return PACKET_TIMEOUT;
126 }
    //-------------------------------------------------------------------
128 // Routine to measure time in interrupts of a chosen timer
    //-------------------------------------------------------------------
130 void timeout_counter_routine(void){
```

```
  if(rxtimeout_event_enabled && timeout_counter>0){
132   timeout_counter--;
  }
134 }
```

```
/*
 * RFSettings.c
 *
 *  Created on: May 30, 2012
 *      Author: Ioannis Papamentzelopoulos
 */
//-----------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------
#include "other/compiler_defs.h"
#include "other/common.h"
#include "motes/c8051f320dk/phy/halRF.h"
#include "NODEID.h"
//-----------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// Radio Settings
//-----------------------------------------------------------------------
//#define RFSETTINGS_250KB  //High data rate settings - 250KBaud
#define RFSETTINGS_2_4KB  //Low data rate - high sensitivity
#ifdef RFSETTINGS_250KB
//radio settings
// Chipcon
// Product = CC2500
// Chip version = E   (VERSION = 0x03)
// Crystal accuracy = 10 ppm
// X-tal frequency = 26 MHz
// RF output power = 0 dBm
// RX filterbandwidth = 541.666667 kHz
```

```
// Phase = 1
// Datarate = 249.938965 kBaud
// Modulation = (7) MSK
// Manchester enable = (0) Manchester disabled
// RF Frequency = 2432.999908 MHz
// Channel spacing = 199.951172 kHz
// Channel number = 6
// Optimization = Sensitivity
// Sync mode = (3) 30/32 sync word bits detected
// Format of RX/TX data = (0) Normal mode, use FIFOs for RX and TX
// CRC operation = (1) CRC calculation in TX and CRC check in RX
   enabled
// Forward Error Correction = (0) FEC disabled
// Length configuration = (1) Variable length packets, packet length
   configured by the first received byte after sync word.
// Packetlength = 255
// Preamble count = (2)  4 bytes
// Append status = 1
// Address check = (0) No address check
// FIFO autoflush = 0
// Device address = 0
// GDO0 signal selection = ( 6) Asserts when sync word has been sent /
    received, and de-asserts at the end of the packet
// GDO2 signal selection = (41) CHIP_RDY
RF_SETTINGS code rfSettings = {
  0x0A,   // FSCTRL1   Frequency synthesizer control.
  0x00,   // FSCTRL0   Frequency synthesizer control.
  0x5D,   // FREQ2     Frequency control word, high byte.
  0x93,   // FREQ1     Frequency control word, middle byte.
  0xB1,   // FREQ0     Frequency control word, low byte.
```

```
58  0x2D,   // MDMCFG4   Modem configuration.
    0x3B,   // MDMCFG3   Modem configuration.
60  0x73,   // MDMCFG2   Modem configuration.
    0x22,   // MDMCFG1   Modem configuration.
62  0xF8,   // MDMCFG0   Modem configuration.
    0x06,   // CHANNR    Channel number.
64  0x00,   // DEVIATN   Modem deviation setting (when FSK modulation is
        enabled).
    0xB6,   // FREND1    Front end RX configuration.
66  0x10,   // FREND0    Front end TX configuration.
    0x18,   // MCSM0     Main Radio Control State Machine configuration.
68  0x1D,   // FOCCFG    Frequency Offset Compensation Configuration.
    0x1C,   // BSCFG     Bit synchronization Configuration.
70  0xC7,   // AGCCTRL2  AGC control.
    0x30,   // AGCCTRL1  AGC control.
72  0xB0,   // AGCCTRL0  AGC control.
    0xEA,   // FSCAL3    Frequency synthesizer calibration.
74  0x0A,   // FSCAL2    Frequency synthesizer calibration.
    0x00,   // FSCAL1    Frequency synthesizer calibration.
76  0x11,   // FSCAL0    Frequency synthesizer calibration.
    0x59,   // FSTEST    Frequency synthesizer calibration.
78  0x88,   // TEST2     Various test settings.
    0x31,   // TEST1     Various test settings.
80  0x0B,   // TEST0     Various test settings.
    0x07,   // FIFOTHR   RXFIFO and TXFIFO thresholds.
82  0x06,   // IOCFG2    GDO2 output pin configuration. 0x0E
    0x06,   // IOCFG0D   GDO0 output pin configuration.
84  0x06 | 0x08,   // PKTCTRL1  Packet automation control.(CRC autoflush
        )
    0x05,   // PKTCTRL0  Packet automation control.
```

```
86    NODEID,     // ADDR       Device address.
      0xFF   ,   // PKTLEN     Packet length.
88    0xFF // PATABLE Transmit power
  };
90 #endif
  #ifdef RFSETTINGS_2_4KB
92 //radio settings
  // Chipcon
94 // Product = CC2500
  // Datarate = 2.4 kBaud
96 // Modulation = 2-FSK
  // Manchester enable = (0) Manchester disabled
98 // RF Frequency = 2432.999908 MHz
  // Channel spacing = 199.951172 kHz
100 // Channel number = 6
  // Optimization = Sensitivity
102 // Sync mode = (3) 30/32 sync word bits detected
  // Format of RX/TX data = (0) Normal mode, use FIFOs for RX and TX
104 // CRC operation = (1) CRC calculation in TX and CRC check in RX
      enabled
  // Forward Error Correction = (0) FEC disabled
106 // Length configuration = (1) Variable length packets, packet length
      configured by the first received byte after sync word.
  // Packetlength = 255
108 // Preamble count = (2)  4 bytes
  // Append status = 1
110 // Address check = (0) No address check
  // FIFO autoflush = 0
112 // Device address = 0
```

```
// GDO0 signal selection = Asserts when sync word has been sent /
    received, and de-asserts at the end of the packet
// GDO2 signal selection = CS (Carrier Sense)
RF_SETTINGS code rfSettings = {
  0x08,   // FSCTRL1       Frequency Synthesizer Control
  0x00,   // FSCTRL0       Frequency Synthesizer Control
  0x5D,   // FREQ2         Frequency Control Word, High Byte
  0x93,   // FREQ1         Frequency Control Word, Middle Byte
  0xB1,   // FREQ0         Frequency Control Word, Low Byte
  0x86,   // MDMCFG4       Modem Configuration
  0x83,   // MDMCFG3       Modem Configuration
  0x03,   // MDMCFG2       Modem Configuration 0x03 is for 30/32 bits
      sync word
  0x22,   // MDMCFG1       Modem Configuration
  0xF8,   // MDMCFG0       Modem Configuration
  0x06,   // CHANNR        Channel Number 0x0F is for de-complexing ;)
  0x44,   // DEVIATN       Modem Deviation Setting
  0x56,   // FREND1        Front End RX Configuration
  0x10,   // FREND0        Front End TX configuration
  0x18,   // MCSM0         Main Radio Control State Machine
      Configuration
  0x16,   // FOCCFG        Frequency Offset Compensation Configuration
  0x6C,   // BSCFG         Bit Synchronization Configuration
  0x03,   // AGCCTRL2      AGC Control
  0x40,   // AGCCTRL1      AGC Control
  0x91,   // AGCCTRL0      AGC Control
  0xA9,   // FSCAL3        Frequency Synthesizer Calibration
  0x0A,   // FSCAL2        Frequency Synthesizer Calibration
  0x00,   // FSCAL1        Frequency Synthesizer Calibration
  0x11,   // FSCAL0        Frequency Synthesizer Calibration
```

```
140   0x59,   // FSTEST         Frequency Synthesizer Calibration Control
      0x81,   // TEST2          Various Test Settings 0x81 is for improved
         sensitivity
142   0x35,   // TEST1          Various Test Settings 0x35 is for improved
         sensitivity
      0x0B,   // TEST0          Various Test Settings
144   0x07,   // FIFOTHR        RX FIFO and TX FIFO Thresholds
      0x0E,   // IOCFG2         GDO2Output Pin Configuration
146   0x06,   // IOCFG0         GDO0Output Pin Configuration 0x06 is for
         sync word assert/deassert
      0x06 | 0x08,   // PKTCTRL1       Packet Automation Control 0x08 is for
          CRC_AUTOFLUSH
148   0x05,   // PKTCTRL0       Packet Automation Control 0x05 is for
         CRC_ENABLE and variable packet length
      NODEID,   // ADDR           Device Address
150   0xFF,   // PKTLEN         Packet Length
      0xFF // PATABLE Transmit power
152 };
    #endif
```

Δ΄ Application Layer(main) -Κώδικας

Σε αυτό το παράρτημα αναφέρεται ένα απλό δείγμα κώδικα εφαρμογής για ένα σταθμό βάσης και διάφορους κόμβους που στέλνουν πακέτα σε αυτόν, με δείγματα που λαμβάνουν από τον ADC(Analog-to-Digital Converter), σε ένα δίκτυο αισθητήρων. Πολλοί από αυτούς χρησιμοποιούν ίσως και κώδικα από άλλες βιβλιοθήκες, όπως αναφέρθηκε σύμφωνα και με την αρχιτεκτονική HAL(Hardware Abstraction Layer), αλλά εδώ θα αναφέρουμε μόνο τον βασικό κορμό.

```
1  //------------------------------------------------------------------
   // Includes
3  //------------------------------------------------------------------
   #include "other/compiler_defs.h"
5  #include "other/common.h"
   #include "motes/icubes/component_defs/mcu/c8051F320.h"
7  #include "motes/icubes/component_defs/radio/ccxxx0.h"
   #include "motes/icubes/component_defs/icubes_defs.h"
9  //#include "motes/icubes/hal/halFlash.h"
   #include "motes/icubes/hal/halClock.h"
11 #include "motes/icubes/hal/halUart.h"
   #include "motes/icubes/hal/halWait.h"
13 #include "motes/icubes/hal/halSpi.h"
   #include "motes/icubes/hal/halSleep.h"
15 #include "motes/icubes/hal/halTimers.h"
   #include "motes/icubes/hal/halInterrupts.h"
17 #include "motes/icubes/hal/halAdc.h"
   #include "motes/icubes/phy/halRF.h"
19 #include "motes/icubes/mac/packet.h"
   #include "motes/icubes/mac/CSMA.h"
21 #include "main.h"
   #include "NODEID.h"
```

```
23 #include <stdio.h>
   //-----------------------------------------------------------------------
25 // Structures, Unions, Enumerations, and Type Definitions
   //-----------------------------------------------------------------------
27 #define SYSCLK 12000000
   #define BAUDRATE 9600
29 //=======================
   // Application MODE
31 //=======================
   #define NO_ACK_APPLICATION
33 void sendFlagEvent(void);
   UINT8 maxBackoffDelay(UINT8 max_be);
35 //-----------------------------------------------------------------------
   // Global Constants
37 //-----------------------------------------------------------------------
   #define RX_IDX_DATA1 6
39 #define TX_IDX_DATA1 7
   #define RX_IDX_DATA2 7
41 #define TX_IDX_DATA2 8
   #define SPAM_TIME 30000
43 #define MEASUREMENT_PERIOD 60000
   #define TOGGLE_P2_0() P2_0 = !P2_0
45 #define TOGGLE_P2_3() P2_3 = !P2_3
   //-----------------------------------------------------------------------
47 // Function Prototypes
   //-----------------------------------------------------------------------
49 //-----------------------------------------------------------------------
   // Global Variables
51 //-----------------------------------------------------------------------
   bit bs_flag = 0;
```

```
53 UINT32 xdata bs_timer = 0;
   BYTE xdata node_data[2][TOTAL_NODES] = {0};
55 BYTE xdata txBuffer[11];
   BYTE xdata rxBuffer[11];     // Length byte  + 2 status bytes are not
      stored in this buffer
57 #if (NODEID == BASE_STATION)
   UINT16 xdata    packets_received[TOTAL_NODES] = {0}; //packets received
59 UINT16 xdata    packets_ACKed     [TOTAL_NODES] = {0}; //In ACK
      application we keep also
   UINT8  xdata packet_ID  [TOTAL_NODES] = {0}; //the last packet_id for
      check
61 #endif
   //-------------------------------------------------------------------
63 // Main routine
   //-------------------------------------------------------------------
65 void main (void) {
    BYTE length = sizeof(rxBuffer); //length of buffer
67  BYTE last_pack_id = 0;    //packet's id: 1-255
    BYTE xdata i = 1;     //vars..
69  BYTE xdata max_BE_delay=0;
    UINT16 adc_sample = 0;
71  BYTE xdata its =0;
    CLOCK_INIT(CLOCK_12MHZ); //Base clock at 12 Mhz
73  //Set up the crossbar and I/O ports to communicate with
    //the C8051F320 peripherals
75
    IO_PORT_INIT();
77 #if(NODEID == BASE_STATION)
    halUartSetup(SYSCLK,BAUDRATE); //initalize UART for Base Station
79 #endif
```

```
SPI_INIT(SCLK_6_MHZ); //SPI reference speed of communication
INT_PRIORITY(INUM_SPI0, INT_HIGH); //set high interrupts of SPI
    interface
POWER_UP_RESET_CCxxx0();  //enable radio
halRfWriteRfSettings(&rfSettings); //write RF setting of the radio
    cc2500
/* Setup timer to produce interrupts every 100us */

SET_RELOAD_VALUE_TIMER2_16BIT(100,1000);
halSetupTimer23(TIMER_2,SYSCLK_DIV_12,SYSCLK_DIV_12,MODE_0,INT_ON);
halAdcInit(ADC_POS_P1_5, ADC_NEG_GND); // init of ADC
if(!acquire_timer2_event(sendFlagEvent)){ //choose timer
 while(1);
}
if(!csmaInit(TIMER_2)){  // Important: choose a timer
 while(1);     // for CSMA
}
P2_3 = 0;
P2_2 = 0;
P2_6 = 0;
P2_7 = 0;
if(NODEID == BASE_STATION){
 P2_2 = 0;
}else if(NODEID != BASE_STATION ){
 P2_6 = 0;
}
txBuffer[0] = sizeof(txBuffer)-1; //Initialize first byte
//15/16 sync detected
halSpiWriteReg(CCxxx0_MDMCFG2,0x01);
//choose a sync_word (small for WOR reasons: les time in RX is needed
```

```
    )
#if NODEID!=BASE_STATION
 WOR_INIT(RX_DUTY_3_1);
 SETUP_GDO0_INT(HIGH,EDGE);
#endif
 max_BE_delay=maxBackoffDelay(MAX_BE);
 while (1) {
#if (NODEID == BASE_STATION)
  //------------------------ BASE STATION CODE-------------------------------//
  its++;
  //------------------- WAKE UP NODES ----------------
  bs_flag = 0;
  bs_timer = SPAM_TIME;
  while(!bs_flag){
   csmaSendPacket(PTYPE_ACK,0,DONT_CARE,DONT_CARE,DONT_CARE);
   P2_6 = !P2_6;
   P2_0 = !P2_0 ;
  }
  //------------------- RECEIVE FROM NODES ---------------------------------
  for(i=0; i<TOTAL_NODES;i++){
   bs_flag = 0;
   bs_timer = SPAM_TIME + (max_BE_delay*SLOT_TIME_INTS)   ;
   while(!bs_flag){
    if(csmaReceivePacket(PTYPE_DATA,65000,1) == PACKET_SUCCESS){
     // ------------------------ ACK ------------------------------
     if(rxBuffer[RX_IDX_ACK_REQUIRE]==BUFF_ACK_REQUIRED){
      if(rxBuffer[RX_IDX_PACKET_TYPE]!= PTYPE_RTS){
       csmaSendPacket(PTYPE_ACK, rxBuffer[RX_IDX_SOURCE], rxBuffer[
           RX_IDX_PACKET_ID],DONT_CARE,DONT_CARE);
      }else{
```

```
        csmaSendPacket(PTYPE_CTS, rxBuffer[RX_IDX_SOURCE], rxBuffer[
            RX_IDX_PACKET_ID],DONT_CARE,DONT_CARE);
      }
      P2_7 = !P2_7;
    }
    if(rxBuffer[RX_IDX_PACKET_TYPE]!= PTYPE_RTS){
      // Save the data
      node_data[0][rxBuffer[RX_IDX_SOURCE]-2] = rxBuffer[RX_IDX_DATA1
          ];
      node_data[1][rxBuffer[RX_IDX_SOURCE]-2] = rxBuffer[RX_IDX_DATA2
          ];
      packets_received[rxBuffer[RX_IDX_SOURCE]-2]++;
      if(rxBuffer[RX_IDX_PACKET_ID]!= packet_ID[rxBuffer[RX_IDX_SOURCE
          ]-2]){
        packets_ACKed[rxBuffer[RX_IDX_SOURCE]-2]++;
        packet_ID[rxBuffer[RX_IDX_SOURCE]-2] = rxBuffer[
            RX_IDX_PACKET_ID];
      }
      P2_0 = !P2_0;
    }
   }
  }
}
puts("|==========> Node ADC Data <==========|");
printf("Time elapsed: %u turns\n",(unsigned int)its);
for(i = 0 ; i < TOTAL_NODES ; i++){
 adc_sample = 0;
 adc_sample += node_data[0][i];
 adc_sample += node_data[1][i] << 8 ;
 printf("NODE %d : [%u]\n", (int)(i+2) , (unsigned int)adc_sample);
```

```
161     printf("Total Packets: %u \n",(unsigned int)packets_received[i]);
        printf("Successful ACKs: %u \n",(unsigned int)packets_ACKed[i]);
163     puts("-----------------------------------------------");
    }
165 //------- WAIT ---------------------
    //P2_2 = !P2_2;
167 for(i=1;i<60;i++){
     halSleep1(MEASUREMENT_PERIOD);
169  halSleep1(MEASUREMENT_PERIOD);
     halSleep1(MEASUREMENT_PERIOD);
171  halSleep1(MEASUREMENT_PERIOD);
     halSleep1(MEASUREMENT_PERIOD);
173 }
    RESET_CCxxx0();
175 halRfWriteRfSettings(&rfSettings);
    halSpiWriteReg(CCxxx0_MDMCFG2,0x01);
177#else
   P2MDOUT |= 0x20; //enable P2. ports (used for sensor)
179
    //---------------------------------- NODE CODE ---------------//
181 halSleepWakeOnInt(INUM_INT1);
    if(halRfReceivePacketWOR(rxBuffer,&length,IGNORE_PACKET_FALSE)){
183  P2_7 = 1;
     if(rxBuffer[RX_IDX_SOURCE]==BASE_STATION){
185   //------------------ BLINK -------------------
      //------------------ WAIT FOR WAKEUP SIG TO END-----------------+
187   OSCICN &= ~0x03;
      // ------------------ ADC -------------------
189   P2_5 = 1;
      adc_sample = halAdcSampleAvg(100);
```

```
191       P2_5 = 0;
          // ---------------- SEND DATA TO BASE STATION ----------------
193       txBuffer[TX_IDX_DATA1] = (0x00FF)&(adc_sample >> 0);
          txBuffer[TX_IDX_DATA2] = (0x00FF)&(adc_sample >> 8);
195       halSleep1( (UINT16) ((SPAM_TIME)/8));
          OSCICN |= 0x03;
197       if(last_pack_id==0){
           last_pack_id=1;
199       }
          if(csmaSendPacket(PTYPE_DATA,BASE_STATION,last_pack_id ,
              BUFF_ACK_NOT_REQUIRED,1)){
201        last_pack_id++;
           if(last_pack_id >255){
203         last_pack_id=1;
           }
205       }
        }
207    }
       P2_7 = 0;
209    WOR_RESET();
#endif
211  }//while
}//main
213 //-------------------------------------------------------------------
// A function used to approximate a maximum delay time in slots
215 //-------------------------------------------------------------------
UINT8 maxBackoffDelay(UINT8 max_be){
217  UINT8 x=0;
  UINT8 max_delay=0;
219  for (x=INIT_BE; x<=max_be ;x++){ //calculate sums of worst case
```

```
    max_delay = max_delay + pow2(x) -1;
221  }
   return max_delay;
223 }
void sendFlagEvent(void){
225  if(bs_timer > 0 && bs_flag == 0){
    bs_timer --;
227  }else{
    bs_flag = 1;
229  }
}
```

Ε΄ Αναφορές

[1] IEEE Computer Society, *Wireless Medium Access Control(MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks(WPANS)*, IEEE 802.15.4 Std, Σεπτέμβριος 2006.

[2] Chiara Buratti, Roberto Verdone, *"Performance Analysis of IEEE 802.15.4 Non Beacon-Enabled Mode"*, IEEE Transactions on Vehicular Technology, Vol 58, No.7, Σεπτέμβριος 2009.

[3] Texas Instruments, Datasheet *"Low-Cost Low-Power 2.4 GHz RF Transceiver (Rev. C)"*, 19 Μαΐου 2009.

[4] Silicon Labs, Datasheet C8051F32x, *Version 1.4*, 2009.

[5] Siri Namtvedt, *"Texas Instruments - Application Note AN047, Wake-on-Radio"*, 2009.

[6] Καμπιανάκης Λευτέρης, *"Custom Over The Air Programmable Embedded Radios"*, 2009.