Διαχείριση Πληροφορίας σε Ασύρματα Δίκτυα Αισθητήρων

Χριστοφοράκη Μαρία

July 28, 2009

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Το παρόν κείμενο αποτελεί την διπλωματική μου εργασία για την απόκτηση του διπλώματος της σχολής Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών. Το θέμα της εργασίας είναι η αποδοτική διαχείριση πληροφορίας σε ασύρματα δίκτυα αισθητήρων. Ο επιβλέπων καθηγητής είναι ο κύριος Παύλος Σπυράκης και επιβλέπων διδάκτορας είναι ο κύριος Ιωάννης Χατζηγιαννάκης.

1.1 Ασύρματα Δίκτυα Αισθητήρων

Οι ασύρματες κινητές επικοινωνίες άλλαξαν δραστικά την καθημερινότητα του μέσου πολίτη, αποτέλεσαν παγκοσμίως την ατμομηχανή της ανάπτυξης και της αγοράς εργασίας τις τελευταίες δεκαετίες και έθεσαν ανεξίτηλα τη σφραγίδα τους στην ταυτότητα της σύγχρονης εποχής. Η ποιοτική, αδιάλειπτη διασύνδεση και επικοινωνία, οποτεδήποτε και οπουδήποτε, αποτελεί πλέον αδιαπραγμάτευτη απαίτηση κάθε συνδρομητή εφοδιασμένου με κινητές τερματικές συσκευές.

Ως άμεση συνέπεια αυτής της κατάστασης μια νέα πραγματικότητα διαμορφώνεται σήμερα μέσα από την ανάπτυξη των Ασύρματων Δικτύων Αισθητήρων είτε λειτουργούν αυτοτελώς, είτε διασυνδεδεμένα στα μεγαλύτερα δίκτυα τηλεπικοινωνιών ή στο διαδίκτυο. Τα δίκτυα αυτά αποτελούνται από μεγάλο αριθμό μικρών ηλεκτρονικών διατάξεων (αισθητήρων), κινητών ή μη, που αποστέλλουν σε μια κεντρική μονάδα πληθώρα δεδομένων προς επεξεργασία και λήψη αποφάσεων.

1.1.1 Εφαρμογές

Η ταχύτατη ανάπτυξη της μικροηλεκτρονικής και των υλικών επέτρεψε την κατασκευή πολύ μικρών αισθητήρων, οι οποίοι έχουν την ικανότητα να μετρούν και να καταγράφουν μια κυριολεκτικά ατέλειωτη σειρά από περιβαλλοντολογικά ή βιολογικά μεγέθη, όπως τη θερμοκρασία, την ατμοσφαιρική πίεση, την υγρασία, τη φωτεινότητα, τη στάθμη υδάτων, την ωρίμανση καρπών, την ανίχνευση χημικών στοιχείων, την πίεση αίματος, τους σφυγμούς καρδιάς, την κίνηση αντικειμένων και ανθρώπων και πολλές ακόμα παραμέτρους που προστίθενται διαρκώς στον παραπάνω κατάλογο. Όλες αυτές οι δυνατότητες που διαθέτουν οι αισθητήρες πλέον, έχουν οδηγήσει στη διεύρυνση πολλών τομέων εφαρμογών, καθώς και στη δημιουργία πληθώρας νέων. Μια επιλεγμένη λίστα των τομέων στους οποίους εφαρμόζονται με επιτυχία τα ασύρματα δίκτυα αισθητήρων περιγράφεται παρακάτω:

- Προστασία από καταστροφές(πυρκαγιές, πλημμύρες κλπ)
- Μετεωρολογία
- Ιατρική
- Τηλεματική
- Παρακολούθηση και συντήρηση μηχανημάτων
- Έξυπνα κτίρια
- Παρακολούθηση και διαχείριση της παραγωγικής διαδικασίας
- Αγροτικές καλλιέργειες

Η ευρεία χρήση των ασυρμάτων δικτύων αισθητήρων σε όλους αυτούς τους τομείς που περιγράψαμε παραπάνω, ώθησε και στην ραγδαία ανάπτυξη έρευνας για τη αποδοτική λειτουργία τους σε πολλά επίπεδα. Το είδος αυτών των δικτύων από τη μία, συνδυάζει χαρακτηριστικά από διάφορες κατηγορίες δικτύων όπως τα αδόμητα δίκτυα(ad-hoc networks), τα ασύρματα δίκτυα(wireless networks), τα κινητά δίκτυα(mobile networks), τα δίκτυα ομότιμων κόμβων (peer-to-peer networks) κτλ. Από την άλλη όμως είναι περίπλοκη η διαδικασία του συνδυασμού των μελετών που

έχουν γίνει πάνω σε όλα αυτά τα δίκτυα ώστε να εφαρμοστούν αποδοτικά στα ασύρματα δίκτυα αισθητήρων. Επιπλέον, ακόμα και αυτοί οι τομείς έρευνας είναι ακόμα σε σχετικά αρχικό στάδιο εξέλιξης.

1.1.2 Περιορισμοί

Σε αυτή την εργασία ασχολούμαστε με πρωτόκολλα αποδοτικής διανομής πληροφορίας σε ασύρματα δίκτυα αισθητήρων, πράγμα που απαιτεί φυσικά και τη χρήση αποδοτικών αλγορίθμων δρομολόγησης πακέτων. Βασικοί παράγοντες που πρέπει να βελτιστοποιηθούν για την αποδοτική λειτουργία τέτοιων πρωτοκόλλων είναι:

- Η Κλιμακωσιμότητα του δικτύου

  Τα δίκτυα αισθητήρων, αποτελούνται συνήθως απο πολύ μεγάλους αριθμούς κόμβων, γεγονός το οποίο καθιστά απαραίτητη την προσαρμοστικότητα των πρωτοκόλλων σε αυτή τους την ιδιότητα.

- Ενεργειακή κατανάλωση κόμβων

  Οι αισθητήρες είναι συσκευές με ελάχιστα διαθέσιμα αποθέματα ενέργειας, πράγμα το οποίο αποτελεί σημαντικό περιορισμό για την αποδοτική λειτουργία των πρωτοκόλλων. Αυτό γιατί θα πρέπει να είναι σχεδιασμένα με τέτοιο τρόπο, ώστε κάθε αισθητήρας ξεχωριστά, αλλά και ο συνδοιασμός όλων αισθητήρων, που αποτελούν το δίκτυο θα πρέπει να καταναλώνουν την ελάχιστη δυνατή ενέργεια. Περιορίζεται λοιπόν και η επικοινωνία μεταξύ τον κόμβων, αλλά και η πολυπλοκότητα των υπολογισμών που πραγματοποιούν στο ελάχιστο δυνατό.

- Μνήμη και υπολογιστική δύναμη κόμβων

  Το μέγεθος των αισθητήρων είναι άλλος ένας λόγος που περιορίζει την υπολογιστική τους δυνατότητα, όπως και η ενέργεια. Το μικρό τους μέγεθος επιπλέον όμως αποτελεί και περιοριστικό παράγοντα για τη μνήμη που έχουν στη διάθεσή τους οι αισθητήρες. Συνεπώς για τους αλγορίθμους δρομολόγησης πρέπει να χρησιμιποιούνται όσο το δυνατόν λιγότερες πληροφορίες για την τοπολογία του δικτύου και ο αποθηκευτικός φόρτος θα πρέπει να διαμοιράζεται ομοιόμορφα στο δίκτυο για την ελαχιστοποίηση της επιβάρυνσης μεμονομένων κόμβων με πληροφορία.

- Προσαρμοστικότητα σε αλλαγές και σφάλματα

  Τα ασύρματα δίκτυα αισθητήρων είναι δυναμικά. Συνεχώς προστίθενται νέοι κόμβοι για την επέκταση του δικτύου. Λόγω των περιορισμένω ενεργειακών αποθεμάτων, αλλά και δυσμενών περιβαλλοντικών συνθηκών υπάρχουν πάντα πιθανότητες κόμβοι να αποτυχάνουν. Και ο σημάντικότερος παράγοντας που καθιστά τα δίκτυα αυτά δυναμικά(τουλάχιστον υποκατήγορίες αυτών) είναι και η κίνηση των κόμβων. Συνεπώς αν δε μιλάμε για πρωτόκολλα κατακλυσμού*(flooding)*, τα οποία είναι ενεργειοβόρα, πρέπει να υπάρχει συνεχής ενημέρωση των κόμβων για τις αλλαγές της τοπικής τους τοπολογίας.

Όλοι αυτοί οι περιορισμοί αποτελούν ακόμα μεγάλη πρόκληση για το σχεδιασμό αποδοτικών αλγορίθμων δρομολόγησης και γενικότερα οποιουδήποτε είδους εφαρμογών στον τομέα των ασυρμάτων δικτύων αισθητήρων.

1.1.3 Δίκτυα αισθητήρων και DHT πρωτόκολλα

Η έρευνα που έχει πραγματοποιηθεί στον τομέα της διανομής πληροφορίας σε δίκτυα ομοτίμων κόμβων, και ειδικά τα πρωτόκολλα που έχουν σχεδιαστεί με βάση κατανεμημένους πίνακες κατακερματισμού*(Distributed Hash Tables - DHTs)*, αποτελούν ιδιαίτερα ενδιαφέροντα αντικείμενα μελέτης για την προσαρμογή τους στον τομέα των ασυρμάτων δικτύων αισθητήρων. Αυτό διότι προτείνουν διάφορες μεθόδους που επιτυγχάνουν τα εξής:

- *Κλιμακωσιμότητα των δικτύων(scalability)* με τη χρήση λίγης μονάχα τοπικής πληροφορίας για τη δρομολόγηση πακέτων,
- *Ευρωστία(robustness)* με την εγγύηση επιτυχίας των λειτουργιών τους σε περασμένο αριθμό βημάτων, και
- *Ανεκτικότητα σε αποτυχίες κόμβων και συνδέσμων(fault tolerance)* με διάφορους αλγορίθμους ανάνηψης από σφάλματα.

Παρ' όλες τις ομοιότητες όμως, με τα δίκτυα ομοτίμων κόμβων, υπάρχουν κάποια βασικά χαρακτηριστικά των δικτύων αισθητήρων που καθιστούν αδύνατη την άμεση εφαρμογή των DHT-πρωτοκόλλων σε αυτά. Τα κυριότερα αυτών είναι ο περιορισμός στην κατανάλωση της ενέργειας, η έλλειψη ονοματολογίας της μορφής IP-διεύθυνσης

και σε πολλές περιπτώσεις η κινητικότητα των κόμβων. Για την αντιμετώπιση αυτών των προβλημάτων που θα έχει ως αποτέλεσμα την αποδοτική προσαρμογή των πρωτοκόλλων αυτών σε δίκτυα αισθητήρων, έχουν γίνει κάποιες προσπάθειες που παρουσιάζονται σε αυτή την εργασία.

Κεφάλαιο 2

Πρωτόκολλα διανομής πληροφορίας σε ασύρματα δίκτυα αισθητήρων

2.1 Εισαγωγή

Στο παρόν κεφάλαιο θα παρουσιάσουμε κάποια γνωστά πρωτόκολλα δρομολόγησης και διαχείρισης δεδομένων σε ασύρματα δίκτυα αισθητήρων. Ο σχεδιασμός τους αποβλέπει κυρίως στην ελαχιστοποίηση της κατανάλωσης ενέργειας και την ομοιόμορφη κατανομή λειτουργιών, που είναι και από τα πιο κρίσιμα προβλήματα που απασχολούν την έρευνα στον τομέα των ασυρμάτων δικτύων αισθητήρων. Ένα άλλο ζήτημα με το οποίο ασχολούνται τα παρακάτω συστήματα είναι η ελαχιστοποίηση της πληροφορίας κατάστασης($state$) που χρησιμοποιεί κάθε αισθητήρας για τη διαδικασία της δρομολόγησης. Αυτό είναι πολύ σημαντικό γιατί, από τη μία οι αισθητήρες διαθέτουν πολύ περιορισμένη μνήμη και από την άλλη, όσο περισσότερη πληροφορία διατηρούν για την τοπολογία του δικτύου, τόσο περισσότερη ενέργεια καταναλώνεται για να τη συλλέξουν. Τέλος σε αυτή την ενότητα θα παρουσιάσουμε και ένα πρωτόκολλο γεωγραφικής δρομολόγησης δεδομένων. Αυτό, ενώ απευθύνεται στον ευρύτερο τομέα ασυρμάτων δικτύων, χρησιμοποιείται στην ανάπτυξη πολλών άλλων πρωτοκόλλων που αφορούν σε ασύρματα δίκτυα αισθητήρων.

2.2 LEACH (Low Energy Adaptive Clustering Hierarchy)

Το LEACH[1] είναι ένα πρωτόκολλο επικοινωνίας δεδομένων για ασύρματα δίκτυα αισθητήρων που σαν βασικό στόχο έχει την ομοιόμορφη κατανομή του ενεργειακού φόρτου σε όλους τους κόμβους που είναι συνδεδεμένοι στο δίκτυο. Για το μοντέλο δικτύου στο οποίο αναφέρονται οι συγγραφείς του [1], ισχύουν κάποιες βασικές υποθέσεις. Το δίκτυο αποτελείται από εκατοντάδες ή και χιλιάδες μικροαισθητήρες(ομογενείς κόμβοι με περιορισμένα αποθέματα ενέργειας). Υπάρχει μια βάση σε ένα αρκετά απομακρυσμένο σημείο από το δίκτυο, στην οποία όλοι οι κόμβοι πρέπει να στέλνουν τα δεδομένα που συλλέγουν. Όλοι οι κόμβοι έχουν τη δυνατότητα, με κατάλληλη ρύθμιση του του εύρους εκπομπής τους, να φτάσουν τη βάση. Κανένας κόμβος δεν έχει γνώση της γεωγραφικής του θέσης και οι κόμβοι στέλνουν συμπιεσμένα τα δεδομένα τους, όταν ο προορισμός είναι η βάση. Τέλος, υποθέτουμε ότι τα κανάλια επικοινωνίας είναι συμμετρικά, το οποίο σημαίνει ότι η ενέργεια που καταναλώνεται για την αποστολή ενός δυαδικού ψηφίου από ένα κόμβο u σ' ένα v είναι η ίδια που θα καταναλωθεί και για την αποστολή του, από τον κόμβο v στον u.

Το βασικό χαρακτηριστικό του LEACH είναι ότι δημιουργεί δυναμικά *συστάδες(clusters)* κόμβων σε κάθε γύρο λειτουργίας του. Αναθέτει την "αρχηγία" της συστάδας σε κάποιον από τους κόμβους της, ο οποίος είναι υπεύθυνος να συλλέξει τα δεδομένα από τους άλλους μικροαισθητήρες που είναι απλά μέλη της συστάδας του, να τα συμπιέσει και να τα αποστείλει στη βάση. Κάθε κόμβος *αρχηγός συστάδας(cluster head)* συλλέγει n μηνύματα k δυαδικών ψηφίων από τους n γειτονικούς του κόμβους και μετά στέλνει cn (όπου $c \leq 1$ ο συντελεστής συμπίεσης) μηνύματα k δυαδικών ψηφίων στην βάση συλλογής δεδομένων. Το LEACH, όπως αναφέρθηκε και παραπάνω λειτουργεί σε *γύρους (rounds)* και κάθε γύρος έχει τρεις φάσεις.

1. *Φάση διαγωνισμού(Advertisement phase)*: Σε αυτή τη φάση αποφασίζεται ποιοι κόμβοι θα είναι αρχηγοί συστάδων για τον τρέχοντα γύρο.

2. *Φάση συγκρότησης συστάδων(cluster set-up phase)*: Είναι η φάση κατά την οποία οι κόμβοι που έχουν ανακηρυχθεί ως αρχηγοί συστάδων, το ανακοινώνουν στους υπόλοιπους κόμβους του δικτύου. Αυτοί οργανώνονται και γίνονται μέλη

των κατάλληλων συστάδων.

3. *Σταθερή φάση(steady phase)*: Κατά τη διάρκεια αυτής της φάσης, τα δεδομένα που συλλέχθηκαν στέλνονται από τους κόμβους-μέλη στους αρχηγούς των συστάδων τους οι οποίοι τα συμπιέζουν και αποστέλλουν στη βάση.

Ορίζουμε το E_{elec} σαν την ενέργεια που καταναλώνεται ανά δυαδικό ψηφίο όταν λειτουργεί το κύκλωμα εκπομπής ή λήψης, και σαν E_{amp} την ενέργεια που καταναλώνει ο ενισχυτής μετάδοσης ανά δυαδικό ψηφίο που μεταφέρεται. Το κόστος αποστολής χαρακτηρίζεται από την ισότητα $E_{T\chi}(k,d) = E_{elec}k + E_{amp}kd^{\lambda}$, ενώ το κόστος λήψης από την $E_{R\chi}(k,) = E_{elec}k$, όπου k το μήκος του μηνύματος σε δυαδικά ψηφία, d η απόσταση μεταξύ αποστολέα και παραλήπτη, ενώ $\lambda \geq 2$ ο εκθέτης απώλειας μονοπατιού.

Στο LEACH οι κόμβοι-αρχηγοί των συστάδων επιλέγονται στοχαστικά κατά τη φάση διαγωνισμού. Σε κάθε γύρο, κάθε κόμβος n, αποφασίζει ένα τυχαίο αριθμό από το 0 έως το 1. Αν ο αριθμός είναι μικρότερος από ένα κατώφλι $T(n)$ τότε ανακήρυσσει τον εαυτό του ως αρχηγό συστάδας του τρέχοντος γύρου. Το κατώφλι ορίζεται ως εξής:

$$T(n) = \frac{P}{1 - P \times (rmod\frac{1}{P})} \qquad \forall n \in G$$

$$T(n) = 0 \qquad \forall n \notin G$$

με P, το επιθυμητό ποσοστό κόμβων αρχηγών στο δίκτυο, r τον τρέχοντα γύρο και G το σύνολο των κόμβων οι οποίοι ήταν αρχηγοί τους προηγούμενους $1/P$ γύρους. Αυτός ο αλγόριθμος εγγυάται ότι όλοι οι κόμβοι θα γίνουν αρχηγοί συστάδας ακριβώς μια φορά στους $1/P$ γύρους. Είναι επιθυμητό σε κάθε γύρο να υπάρχει περίπου ο ίδιος αριθμός κόμβων αρχηγών. Στον πρώτο γύρο εκλέγονται NP αρχηγοί, στο δεύτερο $N(1-P)P_1$ αρχηγοί. Συνεπώς $N(P-1)P_1 = NP \Rightarrow P_1 = \frac{P}{1-P} = T(n)$. Ενώ για τον τρίτο γύρο έχουμε $N(1-P)(1-P_1)P_2$ αρχηγούς, δηλαδή, $N(1-P)(1-P_1)P_2 = NP \Rightarrow P_2 = \frac{P}{1+PP_1+(P+P_1)} = \frac{1}{1-2P} = T(n)$. Έτσι, διαισθητικά έχουμε το κατώφλι $T(n)$. Αφού έχουν επιλεχθεί οι κόμβοι αρχηγοί, ενημερώνουν τους υπόλοιπους κόμβους του δικτύου για την αρχηγία τους μεταδίδοντας μηνύματα σε όλο το δίκτυο. Οι απλοί κόμβοι ακούγοντας πολλά μηνύματα από κόμβους αρχηγούς, αποφασίζουν τη συστάδα του αρχηγού στην οποία θα ενταχθούν σύμφωνα με τις εντάσεις των σημάτων που

έλαβαν. Τελικά θα ενταχθούν στη συστάδα του αρχηγού από τον οποίο έλαβαν το ισχυρότερο σήμα.

Στη φάση συγκρότησης συστάδων, όλοι οι αρχηγοί είναι ενήμεροι για τα μέλη των συστάδων τους. Δημιουργούν ένα σχεδιάγραμμα για τη διαμοιρασμό του χρόνου που θα διαθέτουν σε κάθε κόμβο-μέλος της συστάδας τους *(TDMA schedule)*. Έπειτα, προωθούν το χρονοδιάγραμμα αυτό σε όλα τα μέλη τους, για να είναι ενήμερα για το πότε μπορούν να στείλουν τα δεδομένα που συνέλεξαν στον αρχηγό τους.

Τέλος, στη σταθερή φάση οι μεν κόμβοι-μέλη των συστάδων, συλλέγουν τα δεδομένα τους από το περιβάλλον και τα αποστέλλουν στον αρχηγό τους, στο χρονικό διάστημα το οποίο τους έχει ανατεθεί μέσω του χρονοδιαγράμματος. Οι δε αρχηγοί των συστάδων λαμβάνουν τα δεδομένα των μελών τους, τα συμπιέζουν και τα προωθούν στη βάση συλλογής δεδομένων. Η διάρκεια της τελευταίας φάσης είναι προκαθορισμένη εξ' αρχής. Για να μην υπάρχουν παρεμβολές κατά τη διάρκεια μεταβιβάσεων μηνυμάτων από κοντινές συστάδες των οποίων τα μέλη μπορεί να στέλνουν ταυτόχρονα, κάθε συστάδα έχει μια διαφορετική κωδικοποίηση μηνυμάτων η οποία επιλέγεται τυχαία από τους αρχηγούς και μεταδίδεται και στα μέλη κατά την οργάνωση της συστάδας.

Παρατηρώντας ένα μοναδικό γύρο του LEACH, δεδομένου ενός δικτύου N κόμβων, είναι προφανές ότι η στοχαστική επιλογή των αρχηγών δεν οδηγεί αυτόματα και στην ελάχιστη κατανάλωση ενέργειας στη διαδικασία μετάδοσης των δεδομένων. Μπορεί για παράδειγμα να επιλεγούν ως αρχηγοί συστάδας κόμβοι που βρίσκονται πολύ κοντά ο ένας στον άλλον ή όλοι οι κόμβοι-αρχηγοί να βρίσκονται στα όρια του δικτύου. Σ' αυτές τις περιπτώσεις οι κόμβοι που είναι απλά μέλη των συστάδων θα καταναλώσουν μεγάλες ποσότητες ενέργειας για να στείλουν τα δεδομένα που έχουν συλλέξει στους αρχηγούς των συστάδων τους. Παρ'Α όλα αυτά, παρατηρώντας τους επόμενους γύρους, θα μπορούσε κανείς να παρατηρήσει ότι η επιλογή κατάλληλων κόμβων αρχηγών, μπορεί να οδηγήσει στην επιλογή "μη-κατάλληλων" κόμβων αρχηγών στους επόμενους γύρους. Τελικά, μπορεί να παρατηρηθεί ότι όπως και να γίνουν επιλογές των κόμβων αρχηγών με τη στοχαστική διαδικασία η συνολική κατανάλωση ενέργειας εξισορροπείται στο σύνολο των γύρων της λειτουργίας του πρωτοκόλλου.

Τέλος προτείνεται και μια ιεραρχική δομή, βασισμένη στον παραπάνω αλγόριθμο. Η ίδια διαδικασία που ακολουθείται παραπάνω, ακολουθείται σε περισσότερα επίπεδα.

Δηλαδή αφού επιλεχθούν οι κόμβοι αρχηγοί, λειτουργούν σαν απλοί κόμβοι και επιλέγουν αρχηγούς για τις *υπερσυστάδες(superclusters)* που θα σχηματίσουν κοκ. Η παραλλαγή αυτή από τη μία έχει σαν αποτέλεσμα εξοικονόμηση μεγαλύτερου ποσοστού ενέργειας και είναι αρκετά ρεαλιστική προσέγγιση για μεγάλης κλίμακας δίκτυα, από την άλλη όμως, έχει και πολύ μεγαλύτερο κόστος υλοποίησης και αυξάνεται ο χρόνος για να καταλήξουν τελικά τα δεδομένα που έχουν συλλεχθεί από τους μικροαισθητήρες στη βάση συλλογής δεδομένων.

Το μεγαλύτερο προτέρημα του LEACH είναι ότι κατανέμοντας ομοιόμορφα την ενεργειακή κατανάλωση όλου του συστήματος στους μικροαισθητήρες, αυξάνεται σημαντικά η *διάρκεια ζωής(lifetime)* του συστήματος σε σχέση με άλλα πρωτόκολλα δρομολόγησης για τη διαχείριση πληροφορίας σε δίκτυα αισθητήρων.

Δεν έχει υλοποιηθεί ακόμα από κάποια εφαρμογή, αλλά οι έρευνες συνεχίζονται για τη βελτιστοποίηση αυτού του πρωτοκόλλου, όπως βλέπουμε για παράδειγμα στα[2] και[3].

2.3 Directed Diffusion(Κατευθυνόμενη Διάχυση)

Το Directed Diffusion[4] είναι από τα πρώτα δημοφιλή πρωτόκολλα διανομής πληροφορίας που βασίζονται στα δεδομένα που μεταδίδονται*(datacentric protocols)*. Αυτός είναι και ο λόγος για τον οποίο οι μεταδιδόμενες πληροφορίες αντιστοιχίζονται σε μοναδικά ονόματα. Τα βασικά χαρακτηριστικά του [4], είναι η αποδοτικότητα όσον αφορά την κατανάλωση ενέργειας, ότι οι κόμβοι "γνωρίζουν" την εφαρμογή που χρησιμοποιεί το πρωτόκολλο(*application specific*), δηλαδή προσαρμόζεται στην εκάστοτε εφαρμογή, και φυσικά ότι είναι *datacentric*. Οι στόχοι της σχεδίασης αυτού του πρωτοκόλλου είναι η αποδοτική δρομολόγηση, που έχει σαν επακόλουθο την εξοικονόμηση ενέργειας, η κλιμακωσιμότητα, που αποτελεί πολύ σημαντική ιδιότητα των πρωτοκόλλων ασύρματων δικτύων αισθητήρων, καθώς είναι επί το πλείστον δίκτυα πολύ μεγάλης κλίμακας, και τέλος η ευρωστία του συστήματος και η ανεκτικότητα σε σφάλματα, που επίσης είναι πολύ σημαντικά σε δίκτυα αισθητήρων, μιας και οι αποτυχίες των κόμβων είναι πολύ σύνηθες φαινόμενο στα δίκτυα αυτά.

Τα βασικά "λογικά δομικά στοιχεία" δικτύου που χρησιμοποιεί το DD είναι:

- *Τα μηνύματα ενδιαφέροντος(Interest messages)*

- *Τα μηνύματα δεδομένων(Data messages)*
- *Οι σύνδεσμοι(Gradients)*
- *Οι ενισχυμένοι σύνδεσμοι(Reinforced gradients)*

Τα *ενδιαφέροντα (Interests)* περιλαμβάνουν την περιγραφή μιας αποστολής που πρέπει να εκτελέσουν οι αισθητήρες. Για παράδειγμα, ένα ενδιαφέρον μπορεί είναι η αναζήτηση της πληροφορίας για το που βρίσκονται κάποια συγκεκριμένου τύπου οχήματα ή για την ύπαρξη ενός συγκεκριμένου ζώου στην περιοχή. Οι αισθητήρες απαντούν σε μια τέτοια αποστολή εφόσον η περιγραφή ταυτίζεται με τα δεδομένα που συλλέγουν. Τα μηνύματα αυτά εκκινούνται και εισάγονται στο δίκτυο από κάποιους κόμβους(συνήθως τυχαίους), οι οποίοι λέγονται *sinks* και διαχέονται στο σύστημα. Κάθε *Interest* παράγει και μια αποστολή, και για κάθε ενεργή αποστολή ο κόμβος εκκίνησης ενδιαφέροντος παράγει και ένα διερευνητικό μήνυμα ενδιαφέροντος *Exploratory Interest Message*. Τα ενδιαφέροντα ανανεώνονται περιοδικά από κάθε κόμβο εκκίνησης, ενώ τα μηνύματα που διαχέουν στο δίκτυο δεν περιέχουν καμία πληροφορία για αυτούς. Τα ενδιαφέροντα που περιγράφουν ακριβώς την ίδια αποστολή συναθροίζονται και λειτουργούν ως ένα.

Όλοι οι κόμβοι του δικτύου διατηρούν προσωρινή μνήμη (cache) στην οποία αποθηκεύουν τα μηνύματα ενδιαφέροντος που έχουν λάβει. Κάθε φορά που λαμβάνουν ένα νέο μήνυμα ενδιαφέροντος δημιουργούν μια νέα εγγραφή στην προσωρινή τους μνήμη. Η εγγραφή αυτή περιέχει τη χρονοσφραγίδα του τελευταίου μηνύματος ενδιαφέροντος από αυτά που ταυτίζονται, καθώς και το πολύ ένα *σύνδεσμο(gradient)* για κάθε γείτονα του κόμβου. Κάθε ένας από τους συνδέσμους περιγράφεται από την τιμή του ρυθμού μετάδοσης και τη διάρκειά του. Αν ένας κόμβος λάβει το ίδιο ενδιαφέρον από κάποιον άλλο γείτονα, απλά δημιουργεί για την ίδια εγγραφή ακόμα ένα σύνδεσμο. Μια εγγραφή διαγράφεται από την προσωρινή μνήμη του κόμβου, μόνο αφού λήξουν όλοι του οι σύνδεσμοι.

Όταν ένας κόμβος λάβει μήνυμα ενδιαφέροντος το οποίο μπορεί να εξυπηρετήσει, ξεκινά τη συλλογή δεδομένων από το περιβάλλον. Αν αυτά ταιριάζουν με τα ζητούμενα, τα προωθεί στους αιτούντες. Για να γίνει αυτό, πρώτα υπολογίζει το μέγιστο ρυθμό μεταφοράς από τους συνδέσμους που σχετίζονται με το συγκεκριμένο ενδιαφέρον και αρχίζει να παράγει *γεγονότα (events)* από τα δεδομένα που συνέλεξε

και τα προωθεί στους γείτονες, με τους οποίους έχει σύνδεσμο για το συγκεκριμένο αίτημα με το μέγιστο ρυθμό που υπολόγισε. Η αποστολή γίνεται σε κάθε ένα ξεχωριστά(unicast).

Κάθε κόμβος διατηρεί μια προσωρινή μνήμη και για τα δεδομένα. Μόλις λάβει μήνυμα δεδομένων, αν αυτό ανταποκρίνεται στα ενδιαφέροντα που διατηρεί στη μνήμη του και δεν υπάρχει ήδη στη μνήμη δεδομένων, το καταχωρεί στην προσωρινή μνήμη δεδομένων. Σε κάθε άλλη περίπτωση το αγνοεί. Για να προωθήσει μετά ο κόμβος τα μηνύματα δεδομένων που έχει λάβει, εξετάζει το ρυθμό μετάδοσης των δεδομένων αυτών, έστω R_D. Έπειτα εξετάζει και τους ζητούμενους ρυθμούς μετάδοσης στους συνδέσμους του ενδιαφέροντος στο οποίο απαντά το μήνυμα δεδομένων $R_{I1}, R_{I2}, ..., R_{Im}$ όπου m ο αριθμός των συνδέσμων (γειτόνων) που ενδιαφέρθηκαν για τα συγκεκριμένα δεδομένα. Αν $R_D \leq R_{Ii} \quad \forall i = 1, ..., k$,προωθεί τα δεδομένα στους κόμβους-γείτονες για τους οποίους έχει συνδέσμους με ρυθμό R_D, διαφορετικά μειώνει κατάλληλα το ρυθμό μετάδοσης για να τον προσαρμόσει στους αντίστοιχους ζητούμενους και προωθεί το μήνυμα δεδομένων μέσω των συνδέσμων μικρότερου ρυθμού μετάδοσης.

Ο κόμβος εκκίνησης ενδιαφερόντων(sink) διαχέει περιοδικά μηνύματα ενδιαφέροντος σε χαμηλούς ρυθμούς μετάδοσης, τα οποία καλούνται *διερευνητικά μηνύματα ενδιαφέροντος(exploratory interest messages)*. Ενώ οι σύνδεσμοι που δημιουργούνται μέσω αυτών των μηνυμάτων ονομάζονται *διερευνητικοί σύνδεσμοι(exploratory gradients)* και έχουν χαμηλούς ρυθμούς μετάδοσης. Μόλις εντοπιστεί κάποιο γεγονός*(event)*, που ταιριάζει με τα ζητούμενα ενδιαφέροντα, παράγονται τα λεγόμενα *διερευνητικά γεγονότα(exploratory events)* και δρομολογούνται πάλι πίσω προς τον κόμβο(sink) που εξέφρασε το ενδιαφέρον. Όταν αυτός λάβει τα διερευνητικά γεγονότα ενισχύει έναν από τους γείτονές του, στέλνοντας σε αυτόν το ίδιο ενδιαφέρον αλλά με αυξημένο ρυθμό μετάδοσης, με σκοπό να λάβει τελικά τα δεδομένα μέσω αυτού του γείτονα. Ο παραλήπτης του ενισχυμένου ενδιαφέροντος ανανεώνει το ρυθμό μετάδοσης που έχει αποθηκεύσει προηγουμένως για το συγκεκριμένο ενδιαφέρον. Αν ο ζητούμενος ρυθμός μετάδοσης δεδομένων είναι υψηλότερος από αυτόν της μετάδοσης εισερχομένων γεγονότων, ο κόμβος ενισχύει κάποιον από τους συνδέσμους του με ένα γείτονά του ώστε να ανταποκρίνεται στο ζητούμενο. Η επιλογή για το ποιος από τους συνδέσμους ενός κόμβου θα ενισχυθεί για προώθηση βασίζεται σε τοπικά

κριτήρια. Παραδείγματος χάριν, ενισχύεται ο γείτονας που πρώτος ανέφερε ένα νέο γεγονός. Η προσωρινή μνήμη δεδομένων χρησιμοποιείται για να αποφασίσει ποια κριτήρια ικανοποιούνται.

Με το πέρασμα του χρόνου, μπορεί κάποια μονοπάτια να υποβαθμίζονται(λόγω των επιλογών που παρουσιάσαμε παραπάνω). Ένας κόμβος μπορεί να εντοπίσει μια τέτοια υποβάθμιση διαπιστώνοντας για παράδειγμα ότι λαμβάνει μειωμένο αριθμό γεγονότων από κάποιο/κάποιους συγκεκριμένους κόμβους. Έτσι, σε ένα μονοπάτι, όταν οι ενδιάμεσοι κόμβοι εντοπίσουν μια τέτοια υποβάθμιση, επιβάλλουν τους κανόνες ενίσχυσης και διορθώνουν το πρόβλημα.

Παρόλο που το Directed Diffusion δεν μπορεί να εφαρμοστεί αποδοτικά σε συστήματα υψηλών δυναμικών, είναι ένα πρωτόκολλο το οποίο μπορεί να χρησιμοποιηθεί για την υλοποίηση πολλών διαφορετικών εφαρμογών και έχει σημαντική αποδοτικότητα όσον αφορά την ελάχιστη κατανάλωση ενέργειας, και αυξάνει έτσι και τη διάρκεια ζωής ενός συστήματος.

2.4 PFR(Probablilistic Forwarding Protocol)

Το PFR [5] ασχολείται με την αποδοτική διάδοση δεδομένων σε ασύρματα δίκτυα αισθητήρων. Τα δεδομένα διαδίδονται στο δίκτυο με σκοπό να καταλήξουν σε ένα κέντρο ελέγχου(sink) και βασικός στόχος του πρωτοκόλλου είναι η αποφυγή του κατακλυσμού του δικτύου με μηνύματα(flooding). Για την επίλυση του προβλήματος αυτού, το [5] χρησιμοποιεί τη στοχαστική επιλογή συγκεκριμένων (κοντά στο βέλτιστο) μεταδόσεων δεδομένων. Υποθέτουμε ότι οι κόμβοι χρησιμοποιούν μόνο τοπική πληροφορία και μπορούν να λειτουργήσουν χωρίς κανένα συντονισμό μεταξύ τους. Για το σχεδιασμό του PFR έχουν γίνει κάποιες βασικές υποθέσεις. Κάθε κόμβος μπορεί να εκτιμήσει την κατεύθυνση ενός σήματος που λαμβάνει μέσω της τεχνολογίας της κεραίας του, καθώς και υπολογίσει την απόσταση από το στοιχείο απ' όπου έλαβε το σήμα μέσω της έντασής του. Επιπλέον, κάθε κόμβος γνωρίζει την κατεύθυνση του κέντρου ελέγχου. Στην αρχή της οργάνωσης του δικτύου, το κέντρο ελέγχου μεταδίδει σήματα σε όλο το δίκτυο για αυτό το σκοπό.

Όλα τα πρωτόκολλα μετάδοσης δεδομένων πρέπει να έχουν τις παρακάτω ιδιότητες:

- **Ορθότητα(Correctness)**: Πρέπει να εγγυώνται ότι τα δεδομένα θα φτάνουν στον προορισμό τους, δεδομένου ότι το δίκτυο είναι λειτουργικό.

- **Ευρωστία(Robustness)**: Πρέπει να εγγυώνται ότι τα δεδομένα θα φτάσουν σε αρκετά σημεία σε μια περιοχή κοντά στον προορισμό, έτσι ώστε να υπάρχουν αρκετές εναλλακτικές διαδρομές σε περίπτωση που μέρος του δικτύου αποτύχει.

- **Αποδοτικότητα(Efficiency)**: Εδώ αναφερόμαστε σε δίκτυα αισθητήρων, συνεπώς η απόδοση αφορά κυρίως την ελαχιστοποίηση της κατανάλωσης ενέργειας. Σαν μετρική απόδοσης, εδώ θα χρησιμοποιηθεί το ποσοστό ενεργών κόμβων του δικτύου. Δηλαδή αν N, ο συνολικός αριθμός κόμβων του δικτύου και k ο αριθμός των ενεργών. Η απόδοση του συστήματος θα μετριέται με το $r = \frac{k}{N}$. Όσο μικρότερο το r, τόσο πιο αποδοτικό θα θεωρείται το δίκτυο.

Το PFR στοχαστικά ευνοεί περισσότερες μεταδόσεις δεδομένων στην περιοχή(λεπτή λωρίδα) γύρω από το ευθύγραμμο τμήμα που ενώνει τον αποστολέα των δεδομένων(αισθητήρας που συλλέγει τα δεδομένα) έστω E με τον προορισμό S. Τα δεδομένα μεταδίδονται με μια κατάλληλα επιλεγμένη πιθανότητα p και δεν μεταδίδονται με πιθανότητα $1-p$. Για την προσέγγιση των βέλτιστων μεταδόσεων χρησιμοποιείται η πιθανότητα $P_{FW} = \frac{\varphi}{\pi}$.

Η λειτουργία του PFR χωρίζεται σε δύο φάσεις:

1. *Φάση μπροστινής δόμησης(Front Creation Phase)*

2. *Φάση στοχαστικής προώθησης(Probabilistic Forwarding Phase)*

Κατά τη διάρκεια της πρώτης φάσης, ο αισθητήρας s που έχει συλλέξει τα δεδομένα και θέλει να τα στείλει στο κέντρο ελέγχου C, εκκινεί μια διαδικασία πλημμύρας(*flooding*) η οποία όμως έχει περιορισμένη έκταση. Δηλαδή, τα δεδομένα μεταδίδονται σε όλους τους κόμβους γείτονες του s και αυτοί με τη σειρά τους τα μεταδίδουν σε όλους τους δικούς τους γείτονες κοκ. Η διαδικασία αυτή σταματά μετά από πεπερασμένο αριθμό βημάτων, το οποίο μπορεί να επιτευχθεί για παράδειγμα με μια μεταβλητή του μηνύματος που θα μετράει τις μεταβάσεις του. Όταν ένας κόμβος λάβει ένα μήνυμα που θα έχει ξεπεράσει τον επιτρεπτό αριθμό μεταβάσεων θα εισέρχεται στην επόμενη φάση λειτουργίας του PFR. Η φάση αυτή σκοπεύει στο να δημιουργηθεί ένα αρκετά ισχυρό τμήμα του δικτύου που θα έχει στην κατοχή του την κρίσιμη πληροφορία,

γεγονός το οποίο ενισχύει την εγγύηση της επιτυχούς διανομής της πληροφορίας σε περιπτώσεις αποτυχιών κόμβων.

Στην δεύτερη φάση, όλοι οι κόμβοι πού έχουν λάβει την κρίσιμη πληροφορία αρχίζουν να την προωθούν στοχαστικά προς τον παραλήπτη. Η μετάδοση γίνεται προς τους γείτονες με πιθανότητα P_{FW}, με:

$$P_{FW} = \begin{cases} 1 & ,\text{αν } \varphi \geq \varphi_0 \\ \frac{\varphi}{\pi} & ,\text{αλλού} \end{cases}$$

όπου $\varphi_0 = 134^\circ$ και φ η γωνία που σχηματίζουν η ευθεία που περνά από τον τρέχοντα κόμβο με αυτόν που του έστειλε τα δεδομένα και η ευθεία που ενώνει τον τρέχοντα κόμβο με τον προορισμό των δεδομένων.

Όσον αφορά την ορθότητα του PFR, αποδεικνύεται στο [5] ότι για μια πληροφορία που στέλνεται από ένα κόμβο s σε ένα προορισμό c, η πληροφορία παραλαμβάνεται πάντα επιτυχώς όταν το δίκτυο είναι λειτουργικό. Ενώ όσον αφορά την αποδοτικότητά του σε σχέση με την ελαχιστοποίηση της κατανάλωσης ενέργειας, αποδεικνύεται ότι το ποσοστό ενεργών κόμβων r είναι:

$$r = \Theta\left(\left(\frac{n_0}{n}\right)^2\right)$$

όπου n_0 η απόσταση μεταξύ του κόμβου που ξεκινά να στέλνει την πληροφορία και του κόμβου παραλήπτη, ενώ $N = n^2$ ο συνολικός αριθμός κόμβων του δικτύου.

Τέλος, όσον αφορά την ευρωστία του δικτύου, στις περιπτώσεις που κάποιοι κόμβοι που βρίσκονται σε γωνίες μεγαλύτερες των 134° δεν είναι σε λειτουργία, αποδεικνύεται ότι το PFR μπορεί να μεταδώσει την πληροφορία στον προορισμό της μέσω μονοπατιών παράλληλων στο τμήμα που ενώνει παραλήπτη και αποστολέα, έστω $absSR$. Αν οι τα μονοπάτια αυτά έχουν απόσταση x από το τμήμα SR, η πιθανότητα να φτάσει η πληροφορία στον προορισμό της είναι:

$$P_{παραδοσης} \geq \left(1 - \frac{2x}{\pi n_0}\right) \approx \epsilon^{\frac{-2x}{\pi}}$$

Από τα παραπάνω συμπεραίνουμε ότι το PFR είναι ένα πρωτόκολλο που πληροί όλες τις βασικές προϋποθέσεις. Μπορεί να χρησιμοποιηθεί και για την άμεση επίλυση του προβλήματος εντοπισμού και μετάδοσης πληροφορίας, αλλά και σαν μέρος ενός γενικότερου παραδείγματος διανομής πληροφορίας όπως είναι το Directed Diffusion.

2.5 Greedy Perimeter Stateless Routing(GPSR)

Το GPSR[23] βασίζεται κυρίως στο γεγονός ότι τα ασύρματα δίκτυα(στην περίπτωση μας τα δίκτυα αισθητήρων) είναι δυναμικά. Νέοι κόμβοι εισέρχονται να επεκτείνουν το δίκτυο, ενώ πολλοί το εγκαταλείπουν λόγω της εξάντλησης της ενέργειάς τους ή δυσμενών καιρικών φαινομένων. Επιπλέον, στις περισσότερες περιπτώσεις οι κόμβοι δεν είναι στατικοί, δηλαδή κινούνται. Αυτά τα δεδομένα ώθησαν σε μια ερευνητική προσπάθεια που θα χρησιμοποιεί τη δρομολόγηση με βάση τη γεωγραφική θέση των κόμβων που προφανώς προσαρμόζεται σε κάθε περίπτωση. Οι κόμβοι ενός τέτοιο δικτύου μπορεί να διαθέτουν GPS, ή να εκτελούν περιοδικά αλγορίθμους για τον εντοπισμό της θέσης τους *(localization algorithms)*.

Στο [23], κάθε κόμβος έχει στη μνήμη του την ελάχιστη δυνατή πληροφορία για την κοντινή του περιοχή. Διατηρεί ένα πίνακα δρομολόγησης με τις γεωγραφικές θέσεις των κόμβων που απέχουν μόνο μια μετάβαση από αυτόν(*single hop neighbours*). Οι πίνακες αυτοί είναι πάντα ενημερωμένοι με την τρέχουσα πληροφορία μέσω σημάτων που ανταλλάσσουν περιοδικά όλοι κόμβοι με τους γειτονικούς τους. Έστω ότι ένας κόμβος u, δε λαμβάνει το σήμα αυτό(beacon) από ένα γείτονα του n_i του πίνακα δρομολόγησής του, για διάστημα μεγαλύτερο από το T. Για κάθε γείτονα n_i υπάρχει ξεχωριστός χρονιστής t_i, που τίθεται στην αρχική του τιμή όταν ο u λάβει το σήμα του από τον n_i. Αν μετά από χρόνο T ο χρονιστής t_i μηδενιστεί ο u υποθέτει ότι ο γείτονας δεν υπάρχει πλέον και τον διαγράφει από τον πίνακα δρομολόγησής του[1].

Όσον αφορά τη δρομολόγηση μηνυμάτων, που είναι και το βασικό θέμα μελέτης αυτού του πρωτοκόλλου, χρησιμοποιούνται δυο αλγόριθμοι δρομολόγησης. Αυτοί περιγράφονται παρακάτω.

Ίπληστη Προώθηση(Greedy Forwarding)

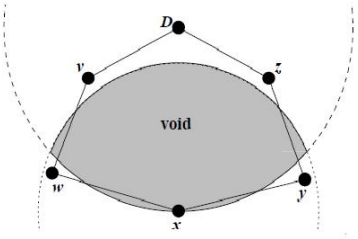
Όταν δημιουργείται ένα πακέτο αιτήματος(*request)* ή δεδομένων αρχικοποιείται αυτόματα σε αυτόν τον τρόπο δρομολόγησης. Τα πακέτα αυτά όταν μεταδίδονται μέσα στο δίκτυο περιέχουν τη διεύθυνση (γεωγραφικές συντεταγμένες) του προορισμού τους. Κάθε κόμβος u για τη μετάδοση ενός πακέτου, ελέγχει την ευκλείδεια απόσταση όλων των γειτόνων του από τον προορισμό του πακέτου. Επιλέγει έπειτα το

[1]Για την αποφυγή συγχρονισμού γειτονικών beacons παραμορφώνεται κάθε μετάδοση κατά 50% του διαστήματος B μεταξύ beacons , έτσι ώστε το μέσο διάστημα μετάδοσης να είναι B, ομοιόμορφα κατανεμημένο στο $[0.5B, 1.5B]$.

γείτονά του, έστω v που έχει την μικρότερη απόσταση από τον προορισμό του πακέτου. Αν ο κόμβος ο v είναι πιο κοντά στον προορισμό από ότι ο ίδιος ο u, τότε μεταδίδει το πακέτο στον v με άπληστη προώθηση(*greedy forwarding*). Διαφορετικά, θέτει τη λειτουργία του πακέτου σε περιμετρική δρομολόγηση(*perimeter routing*) και προσθέτει στο πακέτο τις συντεταγμένες του, για να είναι γνωστό το σημείο εκκίνησης της περιμετρικής δρομολόγησης.

Δρομολόγηση περιμέτρου(Perimeter Routing)

Επειδή υπάρχουν περιοχές κενές από αισθητήρες ή παρεμβάλλονται εμπόδια(voids, obstacles), όπου ο κόμβος που θέλει να προωθήσει το πακέτο του, δεν έχει γείτονα κοντινότερο στον προορισμό από τον ίδιο χρησιμοποιείται ένας εναλλακτικός αλγόριθμος. Κάθε κόμβος εάν δεν έχει κοντινότερο γείτονα στον προορισμό από ότι είναι ο ίδιος,προωθεί το πακέτο στον αριστερόστροφο κοντινότερο γείτονά του. Αυτό γίνεται με τη χρήση του κανόνα του δεξιού χεριού*(Right Hand Rule)*. Έστω ότι κόμβος u είναι ο πρώτος κόμβος που διαπιστώνει ότι κανένας γείτονάς του δεν είναι πιο κοντά στον προορισμό d ενός πακέτου από τον ίδιο. Θα θέσει το πακέτο σε τρόπο λειτουργίας δρομολόγησης περιμέτρου. Έπειτα, θα επιλέξει τον γείτονά του v, που η ακμή μεταξύ τους (u, v) έχει τη μικρότερη γωνία με το ευθύγραμμο τμήμα που δημιουργεί ο u με τον προορισμό του πακέτου d. Η φορά των γωνιών είναι αντίθετη από αυτή των δεικτών του ρολογιού. Αφού λοιπόν ο u κάνει τους κατάλληλους υπολογισμούς για να βρει το διάδοχο του πακέτου, του το προωθεί. Παράδειγμα για μια τέτοια περίπτωση αποτελεί το παρακάτω σχήμα.



Εικόνα 2.1: Παράδειγμα περιοχής void

Όπως βλέπουμε στο παραπάνω σχήμα, ο κόμβος που λαμβάνει το πακέτο σε λειτουργία άπληστης προώθησης και καλείται να επιλέξει το διάδοχο του πακέτου, είναι ο x, ενώ προορισμός είναι ο D. Το πακέτο σε αυτήν την περίπτωση θα ακολουθήσει τη διαδρομή $x \rightarrow w \rightarrow v \rightarrow D$, μέχρι τον προορισμό. Το σύνολο των ακμών που

ακολουθούνται κατά τη διάρκεια αυτής της λειτουργίας ονομάζονται περίμετρος, εξού και το όνομα της λειτουργίας. Όταν ένας κόμβος λάβει πακέτο σε λειτουργία περιμετρικής δρομολόγησης, ελέγχει πρώτα αν αυτός ή κάποιος γείτονάς του είναι πιο κοντά στον προορισμό από ότι το σημείο εκκίνησης περιμετρικής δρομολόγησης. Στην περίπτωση που ισχύει αυτό, επιστρέφει το πακέτο στη λειτουργία άπληστης προώθησης, διαφορετικά συνεχίζει την προώθηση περιμετρικά.

Η προσέγγιση αυτή απαιτεί ένα ευρετικό για την αποφυγή δρομολόγησης μέσω τεμνόμενων ακμών*(non-crosiing heuristc)*. Αυτό χρειάζεται για να αναγκάσει τον κανόνα του δεξιού χεριού να βρει περιμέτρους που εμπεριέχουν κενά*(voids)* ή εμπόδια*(obstacles)* σε περιοχές που οι ακμές του γράφου τέμνονται. Το ευρετικό αυτό αφαιρεί αυθαίρετα όποια ακμή καταμετρά δεύτερη σε ένα ζεύγος από τεμνόμενες ακμές. Αυτό φυσικά μπορεί να έχει σαν αποτέλεσμα ακόμα και τη διαίρεση του δίκτυο και έτσι ο αλγόριθμος δε θα βρει τα μονοπάτια που περιέχουν αυτή την ακμή. Συνεπώς, με τη χρήση ενός τέτοιου ευρετικού, μπορεί τα αποτελέσματα προσέγγισης τέτοιων περιοχών βελτιώνονται, αλλά ο αλγόριθμος δεν βρίσκει πάντα τα μονοπάτια όταν υπάρχουν.

Planarized Graphs:

Παρόλο που το non-crossing heuristic εμπειρικά βρίσκει την πλειοψηφία των μονοπατιών (πάνω από 99.5% από τα $n(n-1)$ μονοπάτια για n κόμβους), σε τυχαία δίκτυα δεν είναι αποδεκτό για έναν αλγόριθμος δρομολόγησης να αποτυγχάνει να βρει μονοπάτι προς ένα κόμβο, που μπορεί να προσεγγιστεί σε μια στατική και αμετάβλητη τοπολογία. Μία πρόταση λοιπόν που γίνεται στο [23] είναι η αναγωγή του γράφου που απεικονίζει το δίκτυο σε ένα *planar γράφο*[2]. Τα δίκτυα αισθητήρων των οποίων οι κόμβοι έχουν την ίδια ακτίνα εκπομπής r, περιγράφονται από το μοντέλο γράφου δίσκου *(disk graph)*. Κάθε κόμβος(αισθητήρας) του δικτύου είναι κορυφή του γράφου και κάθε ακμή (u, v) μεταξύ δυο κόμβων u και v, υπάρχει αν $d(u, v) \leq r$.

Ο Γράφος Σχετικής Γειτνίασης*(Relative Neighbourhood Graph-RNG)* και ο γράφος Gabriel*(Gablriel Graph-GG)* είναι δυο ευρέως διαδεδομένοι planar γράφοι. Ένας αλγόριθμος που πετυχαίνει την απαλοιφή ακμών από ένα γράφο που δεν είναι μέρος των RNG και GG θα απέδιδε ένα δίκτυο χωρίς τεμνόμενες ακμές. Ο αλγόριθμος που προτείνεται θα πρέπει να τρέχει με κατανεμημένο τρόπο από κάθε κόμβο και να

[2]Planar Graph: Γράφος στον οποίο δεν υπάρχουν ακμές που να διασταυρώνονται.

χρειάζεται πληροφορία μόνο για την κοντινή τοπολογία κάθε κόμβου σαν είσοδο για τον αλγόριθμο. Βασική προϋπόθεση για την ορθή λειτουργία του αλγορίθμου αυτού είναι η εγγύηση ότι η απαλοιφή ακμών από το γράφο, για να αναχθεί σε RR ή RNG, δεν οδηγεί σε αποσύνδεση του γράφου.

Η αναγωγή ενός γράφου δίσκου όπως είναι το δίκτυο αισθητήρων σε έναν RNG γίνεται ως εξής:

Κάθε κόμβος u εξετάζει τη λίστα L με τους γείτονές του. Για κάθε ζεύγος κόμβων $w, v \in L$, αν η απόσταση του v από u είναι μεγαλύτερη από τη μέγιστη απόσταση $max\{d(u,w), d(v,w)\}$, ο u απαλείφει την ακμή (u,v). Έτσι, μια ακμή (u,v), απαλείφεται μόνο αν υπάρχει κόμβος w μέσα στο εύρος*(range)* και των δύο κόμβων u, v. Επομένως, απαλείφουμε μια ακμή μόνο όταν υπάρχει ένας μάρτυρας *(witness)*. Έτσι, όλα τα συνδεδεμένα αντικείμενα σε ένα ασύρματης επικοινωνίας δίκτυο χωρίς εμπόδια, δεν θα αποσυνδεθεί με τις αφαιρέσεις των ακμών που δεν ανήκουν στον RNG.

Παρόμοια, η αναγωγή ενός γράφου δίσκου όπως είναι το δίκτυο αισθητήρων σε έναν GG γίνεται ως εξής:

Κάθε κόμβος u εξετάζει τη λίστα L με τους γείτονές του. Για κάθε ζεύγος κόμβων $w, v \in L$ αν η απόσταση του u από το μέσο M, του ευθύγραμμου τμήματος που σχηματίζουν οι κορυφές u και v είναι μεγαλύτερη ή ίση από την απόσταση του w από το σημείο M, δηλαδή αν $d(u,M) \geq (w,M)$, τότε η ακμή (u,v) απαλείφεται. Και σε αυτήν την περίπτωση αποδεικνύεται ότι με την αναγωγή σε GG δεν υπάρχουν περιπτώσεις αποσύνδεσης του γράφου.

Και οι δύο αλγόριθμοι έχουν χρονική πολυπλοκότητα , $O(\log deg^2$όπου deg είναι ο βαθμός σύνδεσης του γράφου.

Ένας από τους δύο αυτούς αλγόριθμους αναγωγής, σε συνδυασμό με τον αλγόριθμο δρομολόγησης που περιγράψαμε παραπάνω, αποτελούν ένα συνεπές και αποδοτικό πρωτόκολλο γεωγραφικής δρομολόγησης και σε ασύρματα δίκτυα αισθητήρων. Όταν τα δίκτυα είναι δυναμικά και οι αλγόριθμοι αναγωγής πρέπει να εκτελούνται περιοδικά.

Κεφάλαιο 3

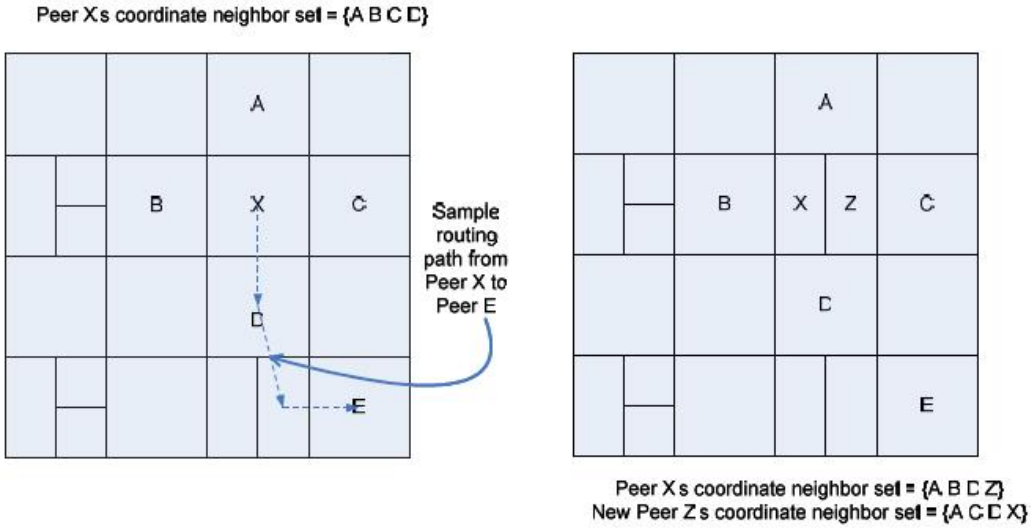
Πρωτόκολλα διαχείρισης πληροφορίας βασισμένα σε κατανεμημένους πίνακες κατακερματισμού(DHTs)

3.1 Εισαγωγή

Όπως αναφέρθηκε και στην εισαγωγή, η χρήση πρωτοκόλλων που βασίζονται σε κατανεμημένους πίνακες κατακερματισμού(DHTs) για την εξυπηρέτηση αναζητήσεων σε ασύρματα δίκτυα αισθητήρων αποτελεί ένα σημαντικό θέμα προς μελέτη. Αυτό, διότι είναι πολύ πιθανόν ένας τέτοιος συνδυασμός να οδηγήσει σε βελτιωμένα μοντέλα σε σχέση με τα ήδη υπάρχοντα όσον αφορά κάποιες μετρικές απόδοσης των δικτύων αισθητήρων. Τέτοιες μετρικές είναι η ευρωστία, η ταχύτητα απόκρισης σε διάφορα αιτήματα, η διάρκεια ζωής του συστήματος, η κλιμακωσιμότητα και η ανεκτικότητα σε σφάλματα . Η χρήση των παρακάτω πρωτοκόλλων και η προσαρμογή τους στον τομέα των αισθητήρων μπορεί να επιφέρει σημαντικές αλλαγές και βελτιστοποιήσεις. Θα περιγράψουμε τα τέσσερα πιο δημοφιλή πρωτόκολλα που βασίζονται σε DHTs και τα βασικά τους χαρακτηριστικά, ενώ στο επόμενο κεφάλαιο θα δούμε κάποια παραδείγματα εφαρμογής τους σε ασύρματα δίκτυα αισθητήρων.

3.2 CAN(Content Addressable Network)

Το CAN [6] είναι μια κατανεμημένη, αποκεντρικοποιημένη υποδομή που παρέχει τη λειτουργικότητα ενός πίνακα κατακερματισμού (hash table) σε κλίμακα διαδικτύου. Κάθε κόμβος αυτής της υποδομής διατηρεί ένα τμήμα (chunk) του hash table, καθώς και πληροφορίες για ένα μικρό αριθμό άμεσων γειτόνων του. Η υποδομή αυτή είναι σχεδιασμένη έτσι ώστε να έχει τρία βασικά χαρακτηριστικά: να είναι ανεκτική σε σφάλματα, κλιμακώσιμη και τέλος οι κόμβοι να είναι σε θέση να οργανωθούν μόνοι τους σε μια δομή. Οι λειτουργίες που εξυπηρετούνται είναι η εισαγωγή πληροφορίας στο δίκτυο, η διαγραφή της από αυτό και φυσικά η αναζήτησή της.



Εικόνα 3.1: Παράδειγμα CAN 2 διαστάσεων, πριν και μετά την άφιξη του κόμβου Z

Το [6] έχει σαν κεντρική ιδέα ένα εικονικό, d-διάστατο Καρτεσιανό χώρο συντεταγμένων πάνω σε έναν πολυ-δακτύλιο(d-$torus$). Οι συντεταγμένες συνιστούν εικονικές διευθύνσεις, εντελώς ανεξάρτητες της φυσικής τοπολογίας του δικτύου. Κάθε χρονική στιγμή, ολόκληρο το σύστημα συντεταγμένων διαμοιράζεται δυναμικά μεταξύ όλων των κόμβων που βρίσκονται σε αυτό(έστω N κόμβοι) και έτσι κάθε ένας από αυτούς έχειὅυπό την κατοχή του τη δική του διακριτή και ανεξάρτητη ζώνη αυτού του χώρου. Κάθε $n_i, i = 1, 2, ...N$ κόμβος διατηρεί ένα πίνακα δρομολόγησης που περιέχει την διεύθυνση IP καθώς και την εικονική ζώνη συντεταγμένων κάθε κόμβου που είναι γείτονάς του [1]στο εικονικό σύστημα. Τα μηνύματα που ανταλλάσσονται

[1]Δύο κόμβοι θεωρούνται γείτονες αν $d-1$ διαστάσεις επικαλύπτονται σε ένα σύστημα d διαστά-

περιέχουν τις συντεταγμένες του προορισμού τους. Με τη χρήση των εικονικών συντεταγμένων των γειτόνων του, κάθε κόμβος προωθεί άπληστα(greedily) το μήνυμα σε όποιον γείτονα του έχει τη μικρότερη ευκλείδεια απόσταση από τις συντεταγμένες του προορισμού. Η πολυπλοκότητα της δρομολόγησης ενός μηνύματος στο CAN είναι $O(d * N^{1overN}$ ενώ αυτή της μνήμης που χρειάζεται ένας κόμβος $O(2 * d)$. Όπως φαίνεται στο πρώτο σχήμα της εικόντας 3.1 το εικονικό σύστημα συντεταγμένων χρησιμοποιείται ώστε να αποθηκεύει ζεύγη {*κλειδί, τιμή*} ως εξής:

Αποθήκευση ζεύγους {K,T}:

- Το κλειδί K αντιστοιχίζεται ντετερμινιστικά σε ένα σημείο $P(x_1, X_2, .., x_d)$ μέσω μιας ομοιόμορφης συνάρτησης κατακερματισμού.
- Ο κόμβος που καλείται να ενθέσει το ζεύγος στο δίκτυο, δημιουργεί για αυτό ένα μήνυμα M με προορισμό το P και το προωθεί στο γείτονά που βρίσκεται πιο κοντά σε αυτό υπό την έννοια ευκλείδειας απόστασης
- Κάθε κόμβος που το λαμβάνει το προωθεί με τον ίδιο τρόπο στον επόμενο έως ότου το λάβει ο κόμβος στο οποίο ανήκει η ζώνη που περιέχει το σημείο P. Αυτός είναι ὄυπεύθυνος για την αποθήκευση του ζεύγους $\{K, T\}$.

Παρόμοια διαδικασία ακολουθείται τόσο για τη διαγραφή, όσο και την αναζήτηση ζευγών *(κλειδιού, τιμής)*. Πάντα χρησιμοποιείται η ίδια συνάρτηση κατακερματισμού για να υπάρχει ένα προς ένα αντιστοιχία πληροφορίας και σημείων όπου βρίσκονται.

Όταν ένας νέος κόμβος εισέρχεται στο σύστημα θα πρέπει να του ανατεθεί η δική του ζώνη (*zone*) στο χώρο συντεταγμένων. Αυτό μπορεί να επιτευχθεί με το χωρισμό της ζώνης ενός ήδη υπάρχοντος κόμβου στη μέση. Έτσι, στο νέο κόμβο θα ανατεθεί το ένα μισό, ενώ ο προηγούμενος κόμβος κάτοχος της ζώνης θα έχει πλέον υπό την κατοχή του το άλλο μισό. Το CAN διατηρεί ένα σχετικό DNS όνομα πεδίου *domain name* το οποίο αντιστοιχίζεται στις διευθύνσεις IP ενός οι περισσοτέρων κόμβων εκκίνησης του CAN, *bootstrap nodes*(οι οποίοι διατηρούν μέρος της λίστας των κόμβων όλου του συστήματος). Όταν ένας νέος κόμβος θέλει να εισέλθει στο δίκτυο του CAN, αναζητά στον DNS ένα *domain name* CAN για να ανακτήσει την διεύθυνση IP ενός από τους bootstrap κόμβους. Ο bootstrap κόμβος παρέχει στο νέο κόμβο την διεύθυνση IP ενός τυχαία επιλεγμένου κόμβου του συστήματος, μέσω

σεων

του οποίου στέλνεται αίτημα εισαγωγής(join request) του νέου κόμβου στο διαδίκτυο. Το αίτημα έχει σαν προορισμό ένα τυχαίο σημείο P που επιλέγει ο νέος κόμβος και δρομολογείται έως τη ζώνη που το περιέχει με τον ήδη γνωστό μηχανισμό δρομολόγησης. Ο κόμβος που είναι κάτοχος της ζώνης στην οποία ανήκει το P, την χωρίζει στη μέση και αναθέτει το άλλο μισό στον νέο κόμβο. Για παράδειγμα, σε δισδιάστατο χώρο, μια ζώνη θα χωριζόταν πρώτα κατά την X διάσταση, μετά κατά την Y κοκ. Ο προηγούμενος κάτοχος παραχωρεί στο νέο κόμβο τα ζεύγη $\{κλειδου, τιμς\}$ που θα αντιστοιχούν στη ζώνη του. Φυσικά, γίνονται και οι αναγκαίες αλλαγές στον πίνακα δρομολόγησης του προηγούμενου κατόχου καθώς και η κατασκευή με τις κατάλληλες διευθύνσεις IP του πίνακα δρομολόγησης του νέου κόμβου.

Όταν ένας κόμβος εγκαταλείπει το CAN δίκτυο, ένας αλγόριθμος άμεσης αποκατάστασης εξασφαλίζει ότι κάποιος από τους γείτονες του κόμβου που έφυγε αναλαμβάνει τη ζώνη του και εκκινεί έναν χρονοδιακόπτη αποκατάστασης. Ο κόμβος που ανέλαβε τη ζώνη ανανεώνει τον πίνακα δρομολόγησής του διαγράφοντας τους κόμβους που δεν είναι πλέον γείτονές του. Έπειτα όλοι οι κόμβοι του συστήματος διοχετεύουν στο δίκτυο μηνύματα ανανέωσης κατάστασης για να εξασφαλίσουν ότι όλοι οι γείτονές του θα ενημερωθούν για την αλλαγή στο σύστημα και θα ανανεώσουν με τη σειρά τους τους δικούς τους πίνακες δρομολόγησης. Ο αριθμός των γειτόνων που διατηρεί κάθε κόμβος στον πίνακά του, είναι ανεξάρτητος από το μέγεθος του συστήματος αφού εξαρτάται μόνο από των αριθμό των διαστάσεων του εικονικού χώρου.

Στο παράδειγμα του δεύτερου σχήματος της εικόνας 3.1 φαίνεται ένα απλό μονοπάτι δρομολόγησης από τον κόμβο X στο σημείο E και η εισαγωγή ενός νέου κόμβου Z στο δίκτυο CAN. Για ένα δισδιάστατο χώρο διαμοιρασμένο σε n ισομεγέθεις ζώνες το μέσο μήκος μονοπατιού είναι $(d/4)x(n^{1overd})$ μεταβιβάσεις μηνυμάτων από κόμβο σε κόμβο $(hops)$, και κάθε ξεχωριστός κόμβος διατηρεί μια λίστα των $2d$ γειτόνων . Έτσι, η κλιμάκωση του συστήματος με την εισαγωγή νέων κόμβων μπορεί να επιτευχθεί χωρίς την αύξηση της μνήμης που χρησιμοποιείται ανά κόμβο, ενώ το μέσο μήκος μονοπατιού αυξάνεται με ρυθμό $O(n^{1overd})$. Καθώς υπάρχουν πολλά διαφορετικά πιθανά μονοπάτια μεταξύ δύο σημείων στο χώρο, αν ένας οι περισσότεροι γείτονες ενός κόμβου αποτύχουν αυτός μπορεί ακόμα να προωθήσει το μήνυμά του μέσω των υπόλοιπων διαθέσιμων μονοπατιών.

Μια από τις βελτιώσεις που προτείνονται στο [6] πάνω στο βασικό πρωτόκολλο του CAN είναι οι διατήρηση πολλαπλών ανεξάρτητων χώρων συντεταγμένων, με κάθε κόμβο να κατέχει μια διαφορετική ζώνη σε καθέναν από αυτούς. Κάθε ένας από αυτούς του χώρους ονομάζεται *πραγματικότητα ή reality*. Για ένα CAN με r πραγματικότητες, σε κάθε κόμβο ανατίθενται r εικονικές ζώνες συντεταγμένων, μία σε κάθε πραγματικότητα καθώς και r λίστες γειτόνων. Τα περιεχόμενα του πίνακα κατακερματισμού, δηλαδή τα ζεύγη {*κλειδί, τιμή*}είναι αντιγραμμένα σε κάθε πραγματικότητα το οποίο βελτιώνει τη διαθεσιμότητα των δεδομένων.

Η διαθεσιμότητα δεδομένων μπορεί να βελτιωθεί επιπλέον με με τη χρήση k διαφορετικών συναρτήσεων κατακερματισμού που θα αντιστοιχίζουν ένα κλειδί σε k διαφορετικά σημεία του χώρου συντεταγμένων. Αυτό έχει ως αποτέλεσμα την αποθήκευση ενός ζεύγους {*κλειδί, τιμή*} σε k διακεκριμένους κόμβους του δικτύου. Συνεπώς, ένα ζεύγος δεν θα είναι διαθέσιμο, μόνο αν αποτύχουν ταυτόχρονα και οι k κόμβοι που το έχουν στη μνήμη τους. Έτσι, τα αιτήματα αναζήτησης για μία συγκεκριμένη εγγραφή του πίνακα κατακερματισμού μπορεί να δρομολογηθεί παράλληλα και στους k κόμβους. Αυτό έχει ως αποτέλεσμα τη μείωση του μέσου χρόνου αναμονής για την απάντηση του αιτήματος και οι ιδιότητες της αξιοπιστίας και της ανθεκτικότητας σε λάθη του συστήματος να ενισχύονται σημαντικά. Από την άλλη, γίνεται μεγαλύτερη χρήση μνήμης ανά κόμβο και περισσότερα μηνύματα ανά αναζήτηση που όμως δεν έχουν ιδιαίτερα σημαντική επίδραση στην συνολική αποδοτικότητα του συστήματος.

Στο [6] έχουν προταθεί και άλλες βελτιώσεις όπου πολλαπλοί κόμβοι μοιράζονται την ίδια ζώνη και λειτουργούν σαν κατανεμημένο υποσύστημα με τη λειτουργικότητα ενός κόμβου, αυξάνοντας πάλι τη διαθεσιμότητα δεδομένων με κόστος όμως πολυπλοκότερους μηχανισμούς συντήρησης της συνέπειας του δικτύου.

Τέλος, παρουσιάστηκαν και βελτιώσεις που λαμβάνουν υπόψιν τους την πραγματική τοπολογία του δικτύου και είτε κατασκευάζουν το εικονικό σύστημα βασισμένο στην πληροφορία αυτή, είτε επιλέγονται μονοπάτια που είναι συντομότερα στο φυσικό δίκτυο άρα είναι και πιο αποδοτικά όσον αφορά το χρόνο αναμονής για εξυπηρέτηση ενός αιτήματος. Στη δεύτερη περίπτωση, ακολουθείται η ίδια διαδικασία που έχει περιγραφεί από την αρχή για την κατασκευή της υποδομής, μόνο που στη δρομολόγηση των αιτημάτων γίνεται η επιλογή του μονοπατιού βασισμένη σε μια άλλη μετρική που συνδυάζει και την ευκλείδεια απόσταση στο εικονικό επίπεδο αλλά και τις αποστάσεις

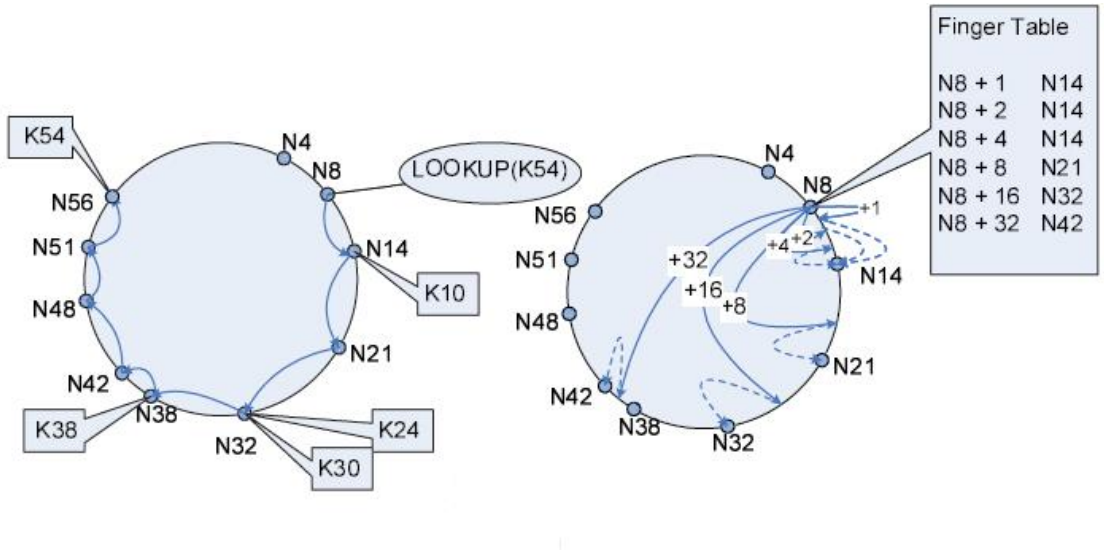
του φυσικού επιπέδου.

Το CAN με διάφορους συνδυασμούς βελτιώσεων, ανάλογα με την περίπτωση μπορεί να εφαρμοστεί σε διάφορα συστήματα πολύ μεγάλης κλίμακας που διαχειρίζονται την διανομή και την ανάκτηση πληροφορίας. Βασισμένο σε αυτό είναι και το πρωτόκολλο GHT [22] που χρησιμοποιείται για τη διαχείριση πληροφορίας σε ασύρματα δίκτυα αισθητήρων και οποίο περιγράφεται στην επόμενη ενότητα.

3.3 Chord

Το Chord [7] είναι ένα επίσης πολύ διαδεδομένο πρωτόκολλο που αποσκοπεί στην αποδοτική διαχείριση δεδομένων σε δίκτυα ομοτίμων κόμβων μεγάλης κλίμακας. Χρησιμοποιεί συνεπείς συναρτήσεις κατακερματισμού (*conststent hashing* για την αντιστοίχηση κλειδιών στους κόμβους του δικτύου. Οι συναρτήσεις αυτές είναι σχεδιασμένες έτσι ώστε να επιτρέπουν στους κόμβους να εισέρχονται στο δίκτυο και να το εγκαταλείπουν με ελάχιστη επιρροή στη συνολική υποδομή. Αυτός ο αποκεντρικοποιημένος σχεδιασμός τείνει να ισορροπεί το φόρτο στο σύστημα, καθώς κάθε κόμβος λαμβάνει περίπου τον ίδιο αριθμό κλειδιών και υπάρχει μικρή μετακίνηση κλειδιών καθώς κόμβοι εισέρχονται ή εγκαταλείπουν το σύστημα. Σε μία σταθερή κατάσταση N κόμβων στο σύστημα, κάθε κόμβος διατηρεί πληροφορία δρομολόγησης για μόνο $O(logN)$ άλλους κόμβους. Αυτό μπορεί να είναι αποδοτικό, αλλά η απόδοση μειώνεται με ήπιους ρυθμούς όταν η πληροφορία αυτή δεν είναι συνεπής.

Οι συνεπείς συναρτήσεις κατακερματισμού αντιστοιχίζουν σε κόμβους και κλειδιά δεδομένων ένα αναγνωριστικό των m δυαδικών ψηφίων με τη χρήση της SHA-1[11] σαν βασική συνάρτηση κατακερματισμού. Το μήκος m του αναγνωριστικού πρέπει να είναι αρκετά μεγάλο ώστε η πιθανότητα δύο κλειδιά να αντιστοιχιστούν στο ίδιο αναγνωριστικό να είναι αμελητέα. Τα αναγνωριστικά είναι ταξινομημένα σε ένα δακτύλιο δεικτών modulo $2m$. Ένα κλειδί k αντιστοιχίζεται στον πρώτο κόμβο του οποίου το αναγνωριστικό είναι ίσο ή ακολουθεί το k στον ταξινομημένο δακτύλιο. Ο κόμβος αυτός ονομάζεται διάδοχος του κλειδιού k και συμβολίζεται ως *διάδοχος(k)*(*successor(k)*). Αν τα αναγνωριστικά αναπαριστώνται σε ένα δακτύλιο αριθμών από το 0 έως το $2m-1$, τότε διάδοχος(k), είναι ο πρώτος κόμβος του δακτυλίου δεξιόστροφα του k. Ο δακτύλιος των δεικτών ονομάζεται *δακτύλιος χορδής ή chord*



Εικόνα 3.2: Παράδειγμα Μονοπατιού για την αναζήτηση του κλειδιού 54 σε δακτύλιο Chord 10 κόμβων και 5 κλειδιών

ring. Για τη διατήρηση συνεπούς αντιστοίχησης, όταν ένας κόμβος n εισέρχεται στο σύστημα, κάποια κλειδιά που είχαν προηγουμένως ανατεθεί στον διάδοχο του n τώρα θα πρέπει να μεταφερθούν στον n. Ενώ αν ένας κόμβος n εγκαταλείψει το σύστημα Chord όλα τα κλειδιά που του έχουν ανατεθεί θα πρέπει να μεταφερθούν στους διαδόχους του n. Για αυτό το λόγο, επιτυγχάνονται αφίξεις και αναχωρήσεις κόμβων με απόδοση $O(logN^2)$. Κατά τα άλλα, δεν χρειάζονται αλλαγές στην ανάθεση κλειδιών σε κόμβους. Στο πρώτο σχήμα της εικόνας 3.2 απεικονίζεται ο Chord δακτύλιος με $m = 6$. Ο συγκεκριμένος δακτύλιος έχει 10 κόμβους συνδεδεμένους στο σύστημα στους οποίους αποθηκεύονται 5 κλειδιά. Ο διάδοχος του αναγνωριστικού 10 είναι ο κόμβος 14 και έτσι το κλειδί 10 θα βρίσκεται στον κόμβο με το αναγνωριστικό 14. Παρομοίως αν ένας κόμβος με αναγνωριστικό 26 ζητήσει να εισέλθει στο σύστημα, θα μεταφερόταν σε αυτόν για αποθήκευση το κλειδί με αναγνωριστικό 24 που τώρα βρίσκεται στον κόμβο 32.

Κάθε κόμβος στον Chord δακτύλιο χρειάζεται να ξέρει πως να επικοινωνήσει με τον τρέχων διάδοχο του στο δακτύλιο των αναγνωριστικών. Τα αιτήματα αναζήτησης προϋποθέτουν το "ταίριασμα" κλειδιού και αναγνωριστικού του κόμβου. Δεδομένου ενός αναγνωριστικού, το αίτημα μπορεί να προωθηθεί γύρω από τον κύκλο μέσω των δεικτών των διαδόχων έως ότου βρεθεί ένα ζεύγος κόμβων που περιέχουν το ζητούμενο αναγνωριστικό. Ο δεύτερος στη σειρά του δακτυλίου από αυτό το ζεύγος

κόμβων θα είναι και αυτός που θα μπορεί να απαντήσει στο αίτημα για το συγκεκριμένο κλειδί. Στο δεύτερο σχήμα της εικόνας 3.2 παρουσιάζεται ένα παράδειγμα όπου ο κόμβος 8 εκκινεί ένα αίτημα αναζήτησης του κλειδιού 54. Ο κόμβος 8 καλεί τη λειτουργία *Αναζήτηση Διαδόχου* για αυτό το κλειδί, ο οποίος στο συγκεκριμένο παράδειγμα είναι ο 56. Το αίτημα "επισκέπτεται" όλους τους κόμβους του δακτυλίου μεταξύ του 8 και του 56. Η απάντηση στο αίτημα επιστρέφεται από το αντίστροφο μονοπάτι το οποίο ακολούθησε και το αίτημα.

Αν m είναι ο αριθμός των δυαδικών ψηφίων που στο χώρο των κλειδιών/αναγνωριστικών κόμβων, κάθε κόμβος n διατηρεί ένα πίνακα δρομολόγησης με το πολύ m εγγραφές. Ο πίνακας αυτός ονομάζεται *πίνακας δεικτών*(*finger table*). Η $i^{οστ}$ εγγραφή του πίνακα δεικτών ενός κόμβου n περιέχει την ταυτότητα το πρώτου κόμβου s που διαδέχεται τον n κατά τουλάχιστον 2^{i-1} θέσεις στον Chord δακτύλιο. Για παράδειγμα $s = διδοχος(n+2)^{i-1}$ όπου $i \leq i \leq; m$. Ο κόμβος s είναι ο $i^{οστς}$ δείκτης του κόμβου n ($n.finger[i]$). Μία εγγραφή του πίνακα δεικτών περιλαμβάνει και το αναγνωριστικό που ανατίθεται σε έναν κόμβο από το chord πρωτόκολλο, αλλά και την IP διεύθυνση (και τον αριθμό πύλης) του σχετικού κόμβου. Στο παράδειγμα του σχήματος 2 της εικόνας 3.2, φαίνεται ο πίνακας δεικτών του κόμβου με αναγνωριστικό 8. Όπως βλέπουμε, η πρώτη εγγραφή του πίνακα δεικτοδοτεί τον κόμβο 14, καθώς ο τελευταίος είναι ο πρώτος κόμβος που διαδέχεται τον κόμβο 9 αφού, $(8 + 20)mod26 = 9$. Παρόμοια, η τελευταία εγγραφή του πίνακα δεικτών του κόμβου 8 δεικτοδοτεί τον κόμβο 42, διότι είναι ο πρώτος κόμβος που διαδέχεται τον κόμβο $(8 + 25)mod26 = 40$. Κατά αυτόν τον τρόπο, οι κόμβοι αποθηκεύουν πληροφορία για ένα μικρό μόνο μέρος των άλλων κόμβων και ξέρουν περισσότερα για κόμβους που τους ακολουθούν άμεσα υπό την έννοια των αναγνωριστικών τους από ότι για άλλους που βρίσκονται αρκετά μακρυά από αυτούς. Βέβαια, ο πίνακας δεικτών δεν περιλαμβάνει αρκετή πληροφορία έτσι ώστε να προσδιορίσει ένας κόμβος απευθείας τον διάδοχο ενός τυχαίου κλειδιού k. Για παράδειγμα στο σχήματος ο κόμβος 8 δεν μπορεί να προσδιορίσει από μόνος του, το διάδοχος του κλειδιού 34, αφού ο πίνακας δεικτών του δεν δεικτοδοτεί σε καμία εγγραφή του τον 38 που είναι ο διάδοχος του 34. Η διαδικασία που ακολουθείται λοιπόν είναι να προωθήσει το αίτημα στον 32 που είναι ο διάδοχος του 42 με τη γνώση που έχει ο 8 και ο 42 με τη σειρά του να ακολουθήσει την ίδια διαδικασία μέχρι το αίτημα να φτάσει στον 38 που είναι ο τρέχων διάδοχος του κλειδιού 34. Συνεπώς,

ένα αίτημα να κάνει $O(logN)$ μεταβάσεις για να φτάσει στον προορισμό του.

Με την άφιξη ενός νέου κόμβου στο σύστημα οι δείκτες σε διαδόχους κάποιων κόμβων πρέπει να ανανεωθούν για να είναι συμβατοί με τη νέα κατάσταση του συστήματος. Είναι πολύ σημαντικό οι εγγραφές του πίνακα δεικτών να είναι πάντα προσαρμοσμένοι στην τρέχουσα κατάσταση κάθε χρονική στιγμή, διότι σε κάθε άλλη περίπτωση δεν υπάρχει εγγύηση για την εγκυρότητα των αιτημάτων αναζήτησης που είναι και το βασικό χαρακτηριστικό του chord πρωτοκόλλου. Το Chord χρησιμοποιεί έναν αλγόριθμο σταθεροποίησης που καλείται περιοδικά στο παρασκήνιο για να ανανεώνει τους δείκτες σε διαδόχους και τις εγγραφές των πινάκων δεικτοδότησης. Η ορθότητα του Chord στηρίζεται στο γεγονός ότι κάθε κόμβος γνωρίζει ανά πάσα χρονική στιγμή τους διαδόχους του. Όταν ένας κόμβος αποτυγχάνει, είναι πιθανόν ένας κόμβος να μην γνωρίζει ποιος είναι ο νέος του διάδοχος και να μην έχει και καμία πιθανότητα να τον βρει. Για να αποφευχθεί αυτή η κατάσταση, οι κόμβοι διατηρούν μια λίστα διαδόχων μεγέθους r, που περιέχει τους r πρώτους διαδόχους του κόμβου. Αν ο διάδοχος ενός κόμβου n δεν επιβεβαιώσει ότι έλαβε το μήνυμα που του έστειλε ο n, τότε ο n απλά προωθεί το μήνυμά του ξανά, αλλά αυτή τη φορά στον επόμενο στη σειρά διάδοχό του. Υποθέτοντας ότι οι αποτυχίες κόμβων συμβαίνουν με πιθανότητα p, η πιθανότητα όλοι οι κόμβοι της λίστας των r διαδόχων να αποτύχουν είναι p^r. Είναι ξεκάθαρο ότι αυξάνοντας το r το σύστημα αυξάνει και την ευρωστία του. Με την ρύθμιση του r μπορεί να επιτευχθεί καλή ευρωστία και ανεκτικότητα σε σφάλματα.

Το Chord μπορεί να χρησιμοποιηθεί για την υλοποίηση διάφορων εφαρμογών . Μια από αυτές είναι το CSF[12](*΄Συνεργατικό Σύστημα Αρχείων*) όπου πολλαπλοί παροχείς δεδομένων συνεργάζονται για την αποθήκευση και την αναζήτηση των δεδομένων τους. Μια ακόμα είναι ένας DNS[13] βασισμένος στο Chord με υπηρεσίες αναζήτησης, όπου τα ονόματα των εξυπηρετητών αντιστοιχούν στα κλειδιά και οι διευθύνσεις IP στις οποίες αναφέρονται αντιστοιχούν στις τιμές του Chord συστήματος που περιγράψαμε παραπάνω. Βασισμένα στο Chord είναι και τα πρωτόκολλα CSN [21] για ασύρματα δίκτυα αισθητήρων καθώς και το Mesh Chord[35] για ασύρματα δίκτυα γενικότερα. Το CSN Chord περιγράφεται ολοκληρωμένα στην επόμενη ενότητα.

3.4 Tapestry

Το Tapestry [8], ένα πρωτόκολλο που παρουσιάστηκε την ίδια χρονική περίοδο με τα παραπάνω , έχει χρησιμοποιηθεί σε αρκετές εφαρμογές και έχει κοινές ιδιότητες με αυτές του Pastry[9]. Χρησιμοποιεί αποκεντρικοποιημένη τυχαιότητα για να πετύχει από τη μία την ομοιόμορφη κατανομή του φόρτου, και από την άλλη την ανάγκη μόνο τοπικής πληροφορίας για τη διαδικασία της δρομολόγησης. Η διαφορά μεταξύ του Parsty και του Tapestry βρίσκεται στον τρόπο με τον οποίο διαχειρίζονται την τοπικότητα του δικτύου καθώς και την δημιουργία αντιγράφων των αντικειμένων των δεδομένων. Η διαφορά τους περιγράφεται εκτενέστερα στην παρακάτω υποενότητα που αφορά το Pastry.

Το Tapestry όσον αφορά στην αρχιτεκτονική του, χρησιμοποιεί μια παραλλαγή της κατανεμημένης τεχνικής αναζήτησης που προτείνεται στο Plaxton et al. [10]. Στη δημοσίευση αυτή παρουσιάζεται μια κατανεμημένη δομή δεδομένων γνωστή ως *Πλέγμα Plaxton*, βελτιστοποιημένη με τέτοιο τρόπο ώστε να υποστηρίζει ένα επιπλέον στρώμα της ιεραρχίας των επιπέδων του διαδικτύου που βρίσκεται πάνω από το επίπεδο δικτύου. Το στρώμα αυτό είναι υπεύθυνο για τον εντοπισμό ονοματισμένων αντικειμένων δεδομένων, που συνδέονται με έναν *ὅγονικό κόμβο ή root peer*. Στο σχεδιασμό του Tapestry χρησιμοποιούνται επιπλέον μηχανισμοί που συμβάλουν στη βελτίωση της διαθεσιμότητας των δεδομένων, της κλιμακωσιμότητας και της προσαρμογής του συστήματος σε τυχόν αποτυχίες ή επιθέσεις στο δίκτυο.

Η παραλλαγή που προτείνεται στο [8], χρησιμοποιεί, όχι μόνο έναν αλλά πολλαπλούς γονικούς κόμβους για κάθε αντικείμενο δεδομένων και έτσι αποφεύγει το μοναδικό σημείο πιθανής αποτυχίας. Στο Πλέγμα Plaxton οι κόμβοι παίζουν το ρόλο των εξυπηρετητών (όπου αποθηκεύονται τα αντικείμενα δεδομένων), των δρομολογητών (προωθούν μηνύματα) και των πελατών (εκκινούν αιτήματα στο δίκτυο). Επίσης, χρησιμοποιούνται τοπικοί *χάρτες δρομολόγησης* σε κάθε κόμβο, ώστε να δρομολογούνται αυξητικά τα μηνύματα μέσω του υποστρώματος Plaxton στο αναγνωριστικό (ID) του προορισμού τους, ψηφίο-ψηφίο. Για παράδειγμα, $***7 => **97 => *297 => 3297$ όπου το $'*'$ παίζει το ρόλο του "μπαλαντέρ". Η επιλογή των ψηφίων που θα χρησιμοποιηθούν από δεξιά προς τα αριστερά ή από αριστερά προς τα δεξιά είναι τυχαία. Ο τοπικός πίνακας δρομολόγησης ενός κόμβου έχει πολλαπλά επίπεδα, το καθένα από τα οποία αντιπροσωπεύει το κατάλληλο επίθεμα αυξανόμενου αριθμού ψηφίων στο χώρο

των αναγνωριστικών. Ο $n^{στος}$ κόμβος στον οποίο έχει φτάσει το μήνυμα έχει κοινό επίθεμα τουλάχιστον n ψηφίων με το αναγνωριστικό του προορισμού του μηνύματος. Για τον εντοπισμό του επόμενου δρομολογητή στον οποίο θα σταλεί το μήνυμα, εξετάζεται το $(n+1)^{στο}$ επίπεδο του χάρτη δρομολόγησης, ώστε να βρεθεί η εγγραφή που ταιριάζει και στο επόμενο στη σειρά ψηφίο του επιθέματος με το αναγνωριστικό του προορισμού. Αυτή η μέθοδος δρομολόγησης εγγυάται ότι κάθε μοναδικός κόμβος που υπάρχει στο σύστημα μπορεί να εντοπιστεί σε το πολύ $log_B N$ μεταβάσεις στο λογικό επίπεδο του υποστρώματος που δημιουργεί το Tapestry, όταν στο σύστημα συμμετέχουν N κόμβοι ο οποίοι χρησιμοποιούν αναγνωριστικά βάσης B. Μιας και ο τοπικός χάρτης δρομολόγησης των κόμβων προϋποθέτει ότι όλα τα προηγούμενα ψηφία του επιθέματος ταυτίζονται με το τρέχον επίθεμα του κόμβου που λαμβάνει το ένα μήνυμα, ο κόμβος αυτός δεν χρειάζεται παρά να κρατάει μια μικρού και σταθερού μεγέθους B εγγραφή για κάθε επίπεδο δρομολόγησης του τοπικό του χάρτη, με αποτέλεσμα ο πίνακας δρομολόγησης να έχει και αυτός σταθερό μέγεθος και ίσο με: (εγγραφές/χάρτη)x αριθμό χαρτών $= B \log_B N$.

Οι μηχανισμοί αναζήτησης και δρομολόγησης που χρησιμοποιεί το Tapestry είναι παρεμφερείς με αυτούς που χρησιμοποιούνται στο Πλέγμα Plaxton, οι οποίοι βασίζονται στο ὅταίριασμα των επιθεμάτων των αναγνωριστικών όπως περιγράψαμε παραπάνω. Οι χάρτες δρομολόγησης είναι οργανωμένοι και εδώ σε επίπεδα δρομολόγησης, καθένα από τα οποία περιέχει εγγραφές που δεικτοδοτούν ένα σύνολο κόμβων πιο κοντά στην απόσταση και με το επίθεμα αναγνωριστικού που ταυτίζεται με αυτό του επιθέματος του αντίστοιχου επιπέδου. Επίσης, κάθε κόμβος διατηρεί και μια ακόμα λίστα με ένα σύνολο κόμβων που αναφέρονται ως γείτονες. Το Tapestry αποθηκεύει τις θέσεις όλων των αντιγράφων που έχουν δημιουργηθεί για τα αντικείμενα των δεδομένων, με αποτέλεσμα να αυξάνεται η προσαρμοστικότητα αλλά και να επιτρέπεται στο επίπεδο της εφαρμογής που τρέχει από πάνω να επιλέγει με βάση τα δικά της κριτήρια, όπως παραδείγματος χάριν την ημερομηνία, ποιο από τα αντίγραφα των δεδομένων θέλει να χρησιμοποιήσει. Κάθε αντικείμενο δεδομένων, εκτός από τη μετρική της απόστασης μπορεί να εμπεριέχει και επιπλέον μετρικές που να καθορίζονται από την εφαρμογή που χρησιμοποιεί το Tapestry. Για παράδειγμα, η γενική αρχιτεκτονική αποθήκευσης του Ocean Store [14], βρίσκει το κοντινότερο αντίγραφο αρχείου που έχει αποθηκευτεί προσωρινά σε διάφορους κόμβους(έχει γίνει cached),

το οποίο ικανοποιεί και το κριτήριο της κοντινότερης απόστασης. Αυτά τα αιτήματα αναζήτησης αποκλίνουν αρκετά από τις απλές λειτουργίες *ὅβρες το πρώτο ή ὅfind first*, και το Tapestry θα δρομολογήσει το μήνυμα στα κοντινότερα k διακριτά αντικείμενα δεδομένων.

Το Tapestry διαχειρίζεται το πρόβλημα της *αποτυχίας από μοναδικό σημείο(single point failure)* που συμβαίνει όταν μόνο ένας γονικός κόμβος είναι υπεύθυνος για ένα αντικείμενο δεδομένων, αναθέτοντας το ίδιο αντικείμενο δεδομένων(με αντίγραφά του) σε πολλαπλούς γονικούς κόμβους. Το Tapestry χρησιμοποιεί την τεχνική της *"θετής" δρομολόγησης (surrogate routing)*, με την οποία επιλέγονται αυξητικά γονικοί κόμβοι, κατά τη διαδικασία της δημοσίευσης/εισαγωγής αντικειμένων δεδομένων στο σύστημα, ώστε να εισάγεται ταυτόχρονα και πληροφορία εντοπισμού τους στο σύστημα. Η θετή δρομολόγηση παρέχει μια τεχνική με την οποία κάθε αναγνωριστικό μπορεί να αντιστοιχιστεί μοναδικά με ένα κόμβο του δικτύου. Ο γονικός κόμβος ενός αντικειμένου ή αλλιώς *θετός κόμβος(surrogate peer)*επιλέγεται να είναι ο κόμβος που ταυτίζεται με το αναγνωριστικό, X, που έχει το αντικείμενο δεδομένου. Αυτό μπορεί να συμβεί με μικρή πιθανότητα, δεδομένης της αραιής φύσης του χώρου των αναγνωριστικών των κόμβων. Παρόλα αυτά το Tapestry, υποθέτει την ύπαρξη του κόμβου με αναγνωριστικό X, επιχειρώντας να δρομολογήσει το μήνυμα με προορισμό τον κόμβο X. Ένα μονοπάτι με προορισμό ένα μη υπάρχον αναγνωριστικό κόμβου θα καταμετρήσει κενές εγγραφές στη λίστα των γειτόνων σε πολλές θέσεις της διαδρομής. Ο στόχος είναι η επιλογή ενός υπάρχοντος συνδέσμου που θα μπορεί να λειτουργήσει σαν εναλλακτική του επιθυμητού συνδέσμου, όπως π.χ. αυτό που θα είναι σχετικό με ένα ψηφίο X. Η δρομολόγηση τερματίζεται όταν καταλήξει σε ένα κόμβο με τοπικό χάρτη δρομολόγησης του οποίου η μόνη κενή εγγραφή ανήκει στον ίδιο τον κόμβο. Ο κόμβος αυτός αναδεικνύεται ως ο *θετός γονικός κόμβος* για το συγκεκριμένο αντικείμενο δεδομένων. Η θετή δρομολόγηση μπορεί μεν να προσθέτει κάποιες επιπλέον λογικές μεταβάσεις στο μονοπάτι δρομολόγησης, αλλά σε σύγκριση με τον αλγόριθμο Plaxton, οι πρόσθετες μεταβάσεις είναι πολύ λίγες. Έτσι ο αλγόριθμος θετής δρομολόγησης που χρησιμοποιεί το Tapestry έχει ελάχιστη επιβάρυνση δρομολόγησης σχετικά με το στατικό καθολικό αλγόριθμο του Plaxton.

Το Tapestry υπογραμμίζει το θέμα της προσαρμοστικότητας σε σφάλματα και διατηρεί τα chahced δεδομένα για ανάνηψη από αποτυχίες κόμβων, βασιζόμενο σε TCP

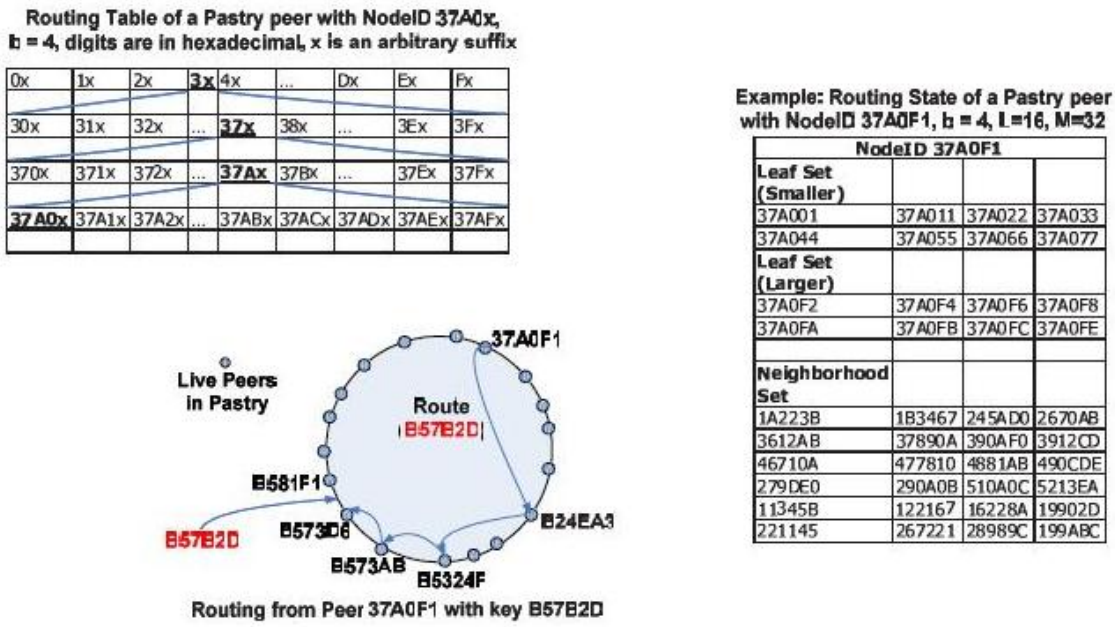
διακοπές(TCP timeouts) και την αποστολή περιοδικών UDP heartbeats πακέτων που ανιχνεύουν αποτυχίες συνδέσμων και εξυπηρετητών κατά τη διάρκεια κανονικών διαδικασιών και επανεκκινούν τη δρομολόγηση των μηνυμάτων μέσω των γειτόνων τους. Κατά τη διάρκεια μια ὅλανθασμένης κατάστασης, κάθε εγγραφή στον τοπικό χάρτη δρομολόγησης ενός γείτονα διατηρεί 2 εναλλακτικούς γείτονες εκτός από τον κοντινότερο που σε μια κανονική κατάσταση είναι η πρώτη του επιλογή. Ὅπως φαίνεται από τις γραφικές παραστάσεις του [8] που είναι αποτέλεσμα προσομοίωσης δικτύων 1000 κόμβων το Tapestry δίνει καλές ταχύτητες δρομολόγησης και διατηρεί τη ρυθμαπόδοση σε ακαριαίες αποτυχίες και συνεχείς αφίξεις και αναχωρήσεις κόμβων από το δίκτυο.

Υπάρχει πληθώρα διαφορετικών εφαρμογών που έχουν σχεδιαστεί και υλοποιηθεί για να λειτουργούν πάνω από το λογικό επίπεδο που δημιουργεί το Tapestry. Ὅπως και στα παραπάνω πρωτόκολλα οι κόμβοι που συμμετέχουν στο σύστημα οργανώνονται μόνοι τους στην επιθυμητή δομή. Είναι επίσης ανεκτικό σε σφάλματα και προσαρμόζεται σε μεγάλους φόρτους αιτημάτων. Τα χαρακτηριστικά αυτά είναι σημαντικά χαρακτηριστικά που πρέπει να ικανοποιούνται και στο σχεδιασμό συστημάτων διαχείρισης πληροφορίας σε ασύρματα δίκτυα αισθητήρων. Συνεπώς τέτοια πρωτόκολλα μπορούν να βασιστούν στο Tapestry παρόλο που δεν έχει γίνει ακόμα κάποια προσπάθεια προσαρμογής του σε δίκτυα αισθητήρων, όπως με τα υπόλοιπα πρωτόκολλα που παρουσιάζονται.

3.5 Pastry

Το Pastry [9], όπως και το Tapestry χρησιμοποιεί μια διαδικασία δρομολόγησης που βασίζεται σε επιθέματα αναγνωριστικών και είναι παρόμοια με τον αλγόριθμο του Plaxton [10], για να δημιουργήσει μια λογική υποδομή που θα βρίσκεται πάνω από το επίπεδο του δικτύου και οι κόμβοι που συμμετέχουν στο σύστημα θα είναι σε θέση να οργανωθούν χωρίς οποιαδήποτε κεντρική βοήθεια στην υποδομή αυτή. Σε αυτή την λογική δομή οι κόμβοι αποστέλλουν αιτήματα πελατών στο δίκτυο και αλληλεπιδρούν με τοπικές οντότητες μιας ή περισσοτέρων εφαρμογών. Σε κάθε κόμβο του Pastry ανατίθεται ένα αναγνωριστικό κόμβου 128 δυαδικών ψηφίων (*NodeID*). Το αναγνωριστικό αυτό χρησιμοποιείται για να δώσει στον κόμβο τη θέση του μέσα σε ένα

κυκλικό χώρο αναγνωριστικών που έχει εύρος 0 - $2^{128} - 1$. Το αναγνωριστικό ενός κόμβου προσδίδεται σε αυτόν με τυχαίο τρόπο όταν εισέρχεται στο δίκτυο. Βασική υπόθεση είναι ότι τα αναγνωριστικά παράγονται κατά τέτοιο ώστε στο σύνολο των κόμβων που βρίσκονται σε μια τυχαία χρονική στιγμή στο σύστημα να έχουν ανατεθεί αναγνωριστικά που στο σύνολό τους θα είναι ομοιόμορφα κατανεμημένα στον χώρο των 128 δυαδικών ψηφίων των αναγνωριστικών. Για ένα δίκτυο N κόμβων, το Pastry δρομολογεί στον αριθμητικά κοντινότερο κόμβο σε σχέση με ένα δοσμένο κλειδί k σε λιγότερο από $log_B N$ μεταβάσεις όταν η λειτουργία του συστήματος είναι κανονική (όπου $B = 2^b$ είναι μια παράμετρος που ορίζεται κατά τη δημιουργία του συστήματος με πιο σύνηθης τιμή $b = 4$). Τα αναγνωριστικά των κόμβων, καθώς και τα κλειδιά θεωρούνται σαν ακολουθίες ψηφίων με βάση B. Το Pastry προωθεί τα μηνύματα στους κόμβους που έχουν το αριθμητικά κοντινότερο αναγνωριστικό στο δοσμένο κλειδί. Ένας κόμβος v_1 κανονικά προωθεί το μήνυμα με κλειδί k στον κόμβο v_2 που το αναγνωριστικό μοιράζεται με το κλειδί k ένα πρόθεμα, το οποίο είναι τουλάχιστον ένα ψηφίο (b δυαδικά ψηφία) μεγαλύτερο από το πρόθεμα που μοιραζόταν ο v_1 με το k.



Εικόνα 3.3: Παράδειγμα πινακάς δρομολόγησης, συνόλου φύλλων και μονοπατιού δρομολόγησης ενός κόμβου Pastry

Όπως φαίνεται στο Σχήμα 3.3 κάθε κόμβος διατηρεί ένα πίνακα δρομολόγησης,

ένα σύνολο γειτόνων και ένα *σύνολο φύλλων* ή *leaf set*. Ο πίνακας δρομολόγησης κάθε κόμβου έχει σχεδιαστεί να έχει $log_B N$ γραμμές, η κάθε μια από τις οποίες διατηρεί $B-1$ εγγραφές. Κάθε μια από αυτές οι $B-1$ εγγραφές μιας γραμμής n του πίνακα δρομολόγησης ενός κόμβου n_i αναφέρεται στους κόμβους των οποίων αναγνωριστικά έχουν κοινά τα πρώτα n ψηφία με το αναγνωριστικό του n_i, αλλά και που το $(n+1)^{στο}$ τους ψηφίο έχει μια από τις υπόλοιπες $B-1$ τιμές διάφορες από το $(n+1)^{στο}$ ψηφίο του αναγνωριστικού του n_i. Κάθε εγγραφή του πίνακα δρομολόγησης περιέχει την διεύθυνση IP των κόμβων των οποίων το αναγνωριστικό, έχει το κατάλληλο πρόθεμα, το οποίο επιλέγεται με βάση μια προσεγγιστική μετρική. Η επιλογή του b αντικατοπτρίζει μια προσπάθεια εξισορρόπησης μεταξύ του χώρου που θα χρησιμοποιηθεί στον πίνακα δρομολόγησης [προσεγγιστικά $log_B N x(B-1)$ εγγραφές] και του μέγιστου αριθμού μεταβάσεων που θα κάνει ένα μήνυμα μεταξύ οποιουδήποτε τυχαίου ζεύγους κόμβων [$log_B N$]. Για το b με τιμή $b=4$ και 10^6 κόμβους, ένας πίνακας δρομολόγησης περιέχει κατά μέσο όρο 75 εγγραφές και ο αναμενόμενος αριθμός μεταβάσεων είναι 5. Το σύνολο των γειτόνων ενός κόμβου, M, περιέχει τα αναγνωριστικά και τις IP διευθύνσεις $absM$ κόμβων που είναι κατά προσέγγιση κοντινότεροι τοπικά στον τρέχοντα κόμβο. Η προσέγγιση τοπικότητας που χρησιμοποιεί το Pastry, βασίζεται σε μια βαθμωτή μετρική όπως ο αριθμός της IP γεωγραφικής απόστασης δρομολόγησης. Το σύνολο φύλλων L είναι ένα σύνολο κόμβων με τα $absL/2$ μεγαλύτερα αριθμητικά κοντινότερα αναγνωριστικά και $absL/2$ με τα αντίστοιχα μικρότερα σε σχέση πάντα με τον τρέχοντα κόμβο. Τυπικές τιμές για τα $absM$ και τα $absL$ είναι το B και το $Bx2$. Ακόμα και με πολλαπλές και ταυτόχρονες αποτυχίες κόμβων εγγυάται η σίγουρη διανομή και η ανάνηψη από σφάλματα, εκτός αν και οι $absL/2$ κόμβοι με γειτονικά αναγνωριστικά αποτύχουν ταυτόχρονα.

Όταν ένας κόμβος εισέρχεται στο δίκτυο πρέπει να αρχικοποιήσει τη δική του κατάσταση (πίνακες δρομολόγησης κλπ) και να ενημερώσει τους άλλους κόμβους για την παρουσία του. Για να συνδεθεί πρώτα από ολα πρέπει να ξέρει την διεύθυνση ενός από τους υπάρχοντες κόμβους του δικτύου. Η λίστα αυτή παρέχεται σαν υπηρεσία που επιστρέφει ανάλογα με τον αιτών κόμβο (που έχει αναγνωριστικό X) μια μικρή λίστα με υπάρχοντες κόμβους του δικτύου που βρίσκονται κοντά σε αυτόν. Έστω ότι ο X επιλέγει από τη λίστα τον A, ο οποίος με τη σειρά του μετα αίτημα του X δρομολογεί ένα ειδκό μήνυμα, *ὅ αίτηση σύνδεσηςὺ* με κλειδί το X στο δίκτυο. Το αίτημα φτάνει

στον κόμβο Z και σαν απόκριση στην αίτηση ο A, ο Z και όλοι οι υπόλοιποι κόμβοι του μονοπατιού που ακολουθήθηκε στέλνουν στον X τους πίνακες δρομολόγησής τους. Τέλος, ο X ενημερώνει τους κατάλληλους κόμβους για την παρουσία του. Αφού ο X θεωρείται ότι βρίσκεται τοπολογικά κοντά στον A το σύνολο τον γειτόνων του A αρχικοποιεί το σύνολο των γειτόνων του X. Λαμβάνοντας υπόψιν τη γενική περίπτωση όπου το αναγνωριστικό του X δεν έχει κανένα κοινό πρόθεμα με το αναγνωριστικό του A, θέτουμε τη γραμμή του $i^{στο}$ επιπέδου του πίνακα δρομολόγησης του κόμβου A, A_i. Το A_0 περιέχει τις κατάλληλες τιμές για το του κόμβου X_0 καθώς οι εγγραφές της γραμμής 0 είναι ανεξάρτητες του αναγνωριστικού του κόμβου. Οι υπόλοιπες γραμμές του πίνακα δρομολόγησης του κόμβου A δεν ενδιαφέρουν τον X αφού δεν μοιράζονται κοινό επίθεμα. Σύμφωνα με την προηγούμενη θεωρία αφού το μονοπάτι που ακολουθήθηκε από το μήνυμα με κλειδί το X, ο κόμβος στον οποίο εστάλη το μήνυμα από τον A, έστω B θα αρχικοποιήσει το X_1 με το B_1 κοκ μέχρι τον Z. Αφού αρχικοποιηθεί ο πίνακας δρομολόγησης του X, αυτός θα στείλει ένα αντίγραφο της δικιάς του πληροφορίας σε όλους τους κόμβους που βρίσκονται στο σύνολο των γειτόνων του, στο σύνολο των φύλλων και στον πίνακα δρομολόγησής του, ώστε αυτοί να ανανεώσουν κατάλληλα τους αντίστοιχους δικούς τους πίνακες.

Ένας κόμβος του Pastry θεωρείται ότι απέτυχε όταν οι άμεσοι γείτονές του στο χώρο των αναγνωριστικών δεν μπορούν πλέον να επικοινωνήσουν μαζί του. Για την αντικατάσταση ενός κόμβου στο σύνολο των φύλλων ενός κόμβου, ο γείτονάς του στο χώρο των αναγνωριστικών επικοινωνεί με τον ὅζωντανό κόμβο από αυτούς που βρίσκονται στο σύνολο των φύλλων του με το μεγαλύτερο δείκτη προς τη τον κόμβο που απέτυχε και του ζητά το δικό του πίνακα φύλλων. Για παράδειγμα αν ο L_i για $absL/2leqileq0$ αποτύχει, ζητά τον πίνακα φύλλων του $L - absL/2$. Ας θέσουμε L' τον πίνακα φύλλων που έλαβε και αυτός επικαλύπτει τον πίνακα φύλλων L και περιέχει κοντινούς κόμβους(σχετικά με τα αναγνωριστικά) που δεν υπάρχουν στον L. Ο κατάλληλος κόμβος επιλέγεται να εισαχθεί στον L επαληθεύοντας ότι ο κόμβος είναι όντως ακόμα στο δίκτυο, με το να επικοινωνήσει μαζί του. Για τη διόρθωση της άκυρης εγγραφής του πίνακα δρομολόγησης $R_{d(levell}$ ενός κόμβου, αυτός επικοινωνεί με τον κόμβο που αναφέρεται σε μια άλλη εγγραφή της ίδιας γραμμής του πίνακα δρομολόγησης $R_{i(levell}$ με $d \neq i$ και του ζητά τη δική του εγγραφή για το R_d. Αν καμία από τις εγγραφές της γραμμής l δεν δεικτοδοτούν κόμβους που βρίσκονται

στο δίκτυο με το κατάλληλο πρόθεμα, ο κόμβος επικοινωνεί με αυτόν της εγγραφής $R_{i(levell+1}$ με $d \neq i$, καλύπτοντας έτσι μια ακόμα πιο ευρεία περιοχή αναζήτησης. Ο πίνακας γειτνίασης δεν χρησιμοποιείται για τη δρομολόγηση μηνυμάτων, αλλά πρέπει να παραμένει και αυτός ανανεωμένος καθώς παίζει καθοριστικό ρόλο στην ανταλλαγή πληροφορίας με κοντινούς κόμβους. Για αυτό το λόγο, κάθε κόμβος επικοινωνεί περιοδικά με όλους τους γείτονες για να δει αν είναι ακόμα συνδεδεμένοι στο σύστημα. Αν από κάποιον δε λάβει απάντηση, ο κόμβος ζητά τους πίνακες γειτνίασης των άλλων γειτόνων του και ελέγχει την τοπικότητα όλων αυτών και με τον κοντινότερο κάνει τις απαραίτητες αλλαγές στον δικό του πίνακα γειτνίασης.

Το Pastry είναι το πρωτόκολλο διανομής πληροφορίας σε δίκτυα ομοτίμων κόμβων που έχει τη λειτουργικότητα ενός κατανεμημένου πίνακα κατακερματισμού, το οποίο σε σχέση με όλα τα παραπάνω πρωτόκολλα που περιγράψαμε έχει εφαρμοστεί πρακτικά σε πολύ περισσότερες υλοποιήσεις εφαρμογών. Οι γνωστότερες αυτών το [15], το PAST [16],[17], το SplitStream [18], το Squirrel [19]και το Patiche [20]. Φυσικά όπως και τα προηγούμενα συστήματα το Pastry αποτελεί έμπνευση για την προσαρμογή του σε συστήματα ασυρμάτων δικτύων αισθητήρων λόγω των ιδιοτήτων του. Η προσπάθεια κατασκευής ενός τέτοιο συστήματος είναι το Scatter Pastry [24] που παρουσιάζεται στην επόμενη ενότητα.

Κεφάλαιο 4

Πρωτόκολλα διαχείρισης πληροφορίας σε δίκτυα αισθητήρων βασισμένα σε DHT πρωτόκολλα

4.1 Εισαγωγή

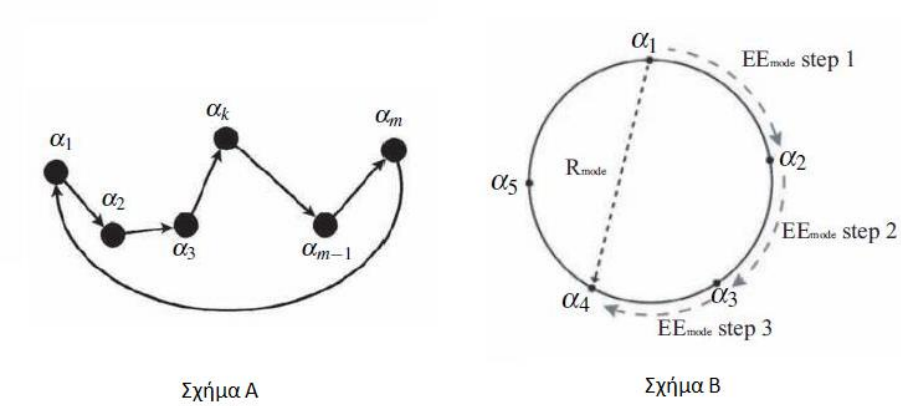
Ενώ τα πρωτόκολλα που εξυπηρετούν τη διανομή πληροφορίας και βασίζονται σε κατανεμημένους πίνακες κατακερματισμού, αποτελούν μια πολύ ενδιαφέρουσα προσέγγιση και για την εφαρμογή τους σε ασύρματα δίκτυα αισθητήρων, υπάρχουν ακόμα αρκετά θέματα που θα πρέπει να μελετηθούν για να μπορούν να προσαρμοστούν αποδοτικά. Τα πρωτόκολλα που περιγράψαμε στην προηγούμενη ενότητα και όμοια αυτών, τυπικά διασυνδέουν τους κόμβους του δικτύου σε ένα λογικό επίπεδο, ανεξάρτητα από τη φυσική τοπολογία του δικτύου. Προφανώς, μια τέτοια προσέγγιση, τουλάχιστον χωρίς κατάλληλες προσαρμογές, θα δημιουργούσε μεγάλο πρόβλημα στην εφαρμογή της σε ασύρματα δίκτυα αισθητήρων, όπου οι κόμβοι έχουν πολύ περιορισμένα αποθέματα ενέργειας. Το πλήθος των φυσικών μεταβάσεων που οδηγούν σε μια λογική μετάβαση στα παραπάνω συστήματα, στην περίπτωση των ασυρμάτων δικτύων αισθητήρων, θα κοστίσει πάρα πολύ στα ενεργειακά αποθέματα των αισθητήρων. Ένα άλλο θέμα που θα πρέπει να μελετηθεί, γιατί αποτελεί σημαντική διαφορά στα δυο είδη δικτύων είναι ότι κάποιοι από τους παραπάνω αλγορίθμους χρησιμοποιούν τις διευθύνσεις IP των κόμβων για την αντιστοίχηση δεδομένων στους κόμβους. Στα ασύρματα δίκτυα αισθητήρων δεν υπάρχει η ίδια έννοια ταυτοποίησης των κόμβων, συνεπώς είτε θα

πρέπει να υπάρχει και μια διαδικασία παροχής αναγνωριστικών στους κόμβους, είτε να μη χρησιμοποιούνται καθόλου τέτοιου είδους αναγνωριστικά. Παρακάτω θα δούμε κάποια πρωτόκολλα που επιχειρούν να ξεπεράσουν αυτά τα προβλήματα, ενώ θα ασχοληθούμε εκτενέστερα με το ένα από αυτά, πιο συγκεκριμένα με το GHT(Geographic Hash Table ή Γεωγραφικός Πίνακας Κατακερματισμού) το οποίο θεωρούμε ότι αντικατοπτρίζει μια πιο πραγματική και πειστική προσέγγιση με πολύ ενδιαφέρουσες ιδέες.

4.2 CSN(Chord Sensor Network)

Το CSN[21], είναι ένα πρωτόκολλο διανομής πληροφορίας σε ασύρματα δίκτυα αισθητήρων, βασισμένο στο Chord[7]. Το περιβάλλον λειτουργίας του CSN έχει τέσσερα βασικά χαρακτηριστικά. Οι αισθητήρες είναι διασκορπισμένοι τυχαία σε μια περιοχή και έτσι το δίκτυο που προκύπτει είναι αδόμητο. Η βάση συλλογής δεδομένων βρίσκεται μακριά από τους αισθητήρες. Το μοντέλο διανομής πληροφορίας κινείται από τον παρατηρητή *(observer initiated)*. Και τέλος, υποθέτουμε ότι οι αισθητήρες δεν κινούνται. Το δίκτυο αισθητήρων χορδής έχει δύο τρόπους λειτουργίας.

1. **Λειτουργία εξοικονόμηση ενέργειας(Energy Efficient Mode-EE_{mode})**
2. **Λειτουργία ευρωστίας(Robust Mode-R_{mode})**



Εικόνα 4.1: Σχήμα Α: Δρομολόγηση πακέτων στη μέθοδος αλυσίδας, Σχήμα Β: Δρομολόηση σε EE_{mode} και R_{mode}

Για την αποδοτική λειτουργία του πρωτοκόλλου, κάθε κόμβος πρέπει να επικοινωνεί με τον κοντινότερό του γεωγραφικά. Με αυτό τον τρόπο ελαχιστοποιείται η

ενέργεια που καταναλώνει για κάθε μετάδοση. Αφού λοιπόν το CSN βασίζεται στο Chord, τίθεται το εξής πρόβλημα:

Πρόβλημα Δακτυλίου: Η κατασκευή ενός λογικού δακτυλίου από κόμβους αισθητήρες, όπου ο δεξιόστροφος διάδοχος κάθε κόμβου θα είναι και ο κοντινότερος (γεωγραφικά) κόμβος σε αυτόν.

Η μελέτη που είχε ως αποτέλεσμα το [21] προτείνει δύο τρόπους για την επίλυση αυτού του NP-πλήρους προβλήματος. Τη Μέθοδο αλυσίδας και τη Μέθοδο ανάθεσης μέσου. Το CSN χωρίζει το δίκτυο σε συστάδες και για κάθε μια από αυτές δημιουργεί και ένα λογικό δακτύλιο. Σταδιακά δημιουργείται μια ιεραρχία συστάδων. Οι συστάδες δημιουργούνται στη φάση οργάνωσης του δικτύου, ανάλογα με την επιλογή της μεθόδου.

- **Μέθοδος Αλυσίδας(Chain Method**: Οι κόμβοι ομαδοποιούνται σε συστάδες "σειριακά". Η διαδικασία εκκινείται από τον κοντινότερο κόμβο στην βάση. Αυτός θέτει τον εαυτό του στην $1^η$ θέση του λογικού δακτυλίου. Στη $2^η$ θέση τοποθετείται ο κοντινότερος γεωγραφικά σε αυτόν κόμβος κοκ. μέχρι να συμπληρωθεί ο αριθμός των θέσεων λ_i στο λογικό δακτύλιο του επιπέδου i. Σε κάθε επίπεδο μια συστάδα έχει συγκεκριμένο αριθμό κόμβων, λ_i. Η διαδικασία συνεχίζεται μέχρι να ομαδοποιηθούν όλοι οι κόμβοι του δικτύου σε συστάδες. Έπειτα ο κόμβος που βρίσκεται στην $1^η$ θέση του δακτυλίου τίθεται ως αρχηγός της συστάδας. Οι κόμβοι αρχηγοί των συστάδων ακολουθούν την ίδια διαδικασία για την οργάνωση του $2^ο$υ επιπέδου συστάδων κοκ. Έτσι κάθε κόμβος α_i σε μια συστάδα επικοινωνεί με τον κοντινότερο σε αυτόν α_{i+1}. Εξαίρεση αποτελεί ο κόμβος που βρίσκεται στην τελευταία θέση του δακτυλίου α_m (m ο αριθμός των θέσεων του λογικού δακτυλίου)όπως φαίνεται στο σχήμα Α της εικόνας 4.1. Για να μην καταναλώνονται μεγάλα ποσά ενέργεια για την επικοινωνία του α_m με τον α_1, ο α_m μεταδίδει αριστερόστροφα στο δακτύλιο μέχρι το μήνυμα να φτάσει στον α_1. Η μέθοδος αυτή συνδυάζεται πιο αποδοτικά με τη λειτουργία εξοικονόμηση ενέργειας(EE_{mode}).

- **Μέθοδος Ανάθεσης Μέσου(Set-Average Method**: Και αυτή η μέθοδος ομαδοποιεί τους αισθητήρες σε συστάδες. Σε αυτή την περίπτωση όμως ακολουθείται μία διαδικασία ώστε κάθε συστάδα να αποτελείται από τους κοντινότερους m κόμβους προσεγγίζοντας τη κοντινότερη μέση απόσταση μεταξύ όλων των

κόμβων της συστάδας. Έπειτα οι κόμβοι όλων των συστάδων οργανώνονται σε δακτύλιους με τον ίδιο τρόπο, όπως και στην προηγούμενη μέθοδο. Και σε αυτή την περίπτωση η συσταδοποίηση είναι ιεραρχική.

Η ανάθεση αρχηγών συστάδων δε γίνεται όπως στο LEACH με στοχαστικό τρόπο σε γύρους. Σε αυτή την περίπτωση κάθε κόμβος αρχηγός συστάδας (οποιουδήποτε επιπέδου) μεταβιβάζει την αρχηγία στον δεξιόστροφο διάδοχό του στο λογικό δακτύλιο, μόλις καταναλώσει. απόθεμα ενέργειας $E = \frac{E_{max}}{\mu}$, όπου E_max η μέγιστη ενέργεια που έχει ο αισθητήρας και μ αναφέρεται στη συχνότητα αλλαγής αρχηγού.

Για να μπορέσει το CSN να προσεγγίσει τα χαρακτηριστικά που χρησιμοποιεί το chord, ακολουθεί και μια διαδικασία κατά την οποία προσδίδει αναγνωριστικά σε κάθε κόμβο. Κάθε αναγνωριστικό έχει τρία μέρη. Το πρώτο δείχνει το επίπεδο ιεραρχίας στο οποίο ανήκει ο κόμβος, το δεύτερο ταυτοποιεί τη συστάδα του συγκεκριμένου επιπέδου στην οποία βρίσκεται, ενώ το τρίτο ταυτοποιεί τον κόμβο μέσα στη συτάδα του.

Και το CSN όπως και το Chord χρησιμοποιεί τη συνάρτηση κατακερματισμού SHA-1[11]. Με αυτήν αντί να αντιστοιχεί όπως το Chord τις διευθύνσεις IP των κόμβων σε μοναδικά αναγνωριστικά, αντιστοιχεί τα μοναδικά ονόματα των κόμβων. Αυτά παράγονται με τη διαδικασία που περιγράψαμε στην προηγούμενη παράγραφο. Κάθε κόμβος του επιπέδου i διατηρεί πληροφορία για $O(\log \lambda_i)$ κόμβους. Δηλαδή διατηρεί στον πίνακα δρομολόγησής του, πληροφορία για όλους τους κόμβους της συστάδας του ή των συστάδων του αν ανήκει σε περισσότερα επίπεδα. Τα δεδομένα που συλλέγουν οι αισθητήρες και διανέμονται στο δίκτυο αντιστοιχίζονται επίσης σε μοναδικά αναγνωριστικά μέσω της συνάρτησης κατακερματισμού. Κάθε κόμβος Θ_i που συμμετέχει στο ανώτατο επίπεδο ιεραρχίας, δέχεται αιτήματα για αναζήτηση δεδομένων από τους χρήστες. Κάθε χρήστης στέλνει στον Θ_i που βρίσκεται πιο κοντά σε αυτόν.

Όσον αφορά τη *λειτουργία ευρωστίας*, η χρονική πολυπλοκότητα αναζήτησης είναι $O(\log N)$. Σε αυτή την περίπτωση ο αισθητήρας Θ_i που λαμβάνει το αίτημα, ελέγχει πρώτα στη μνήμη του για το ζητούμενο κλειδί, έστω k. Αν δεν βρίσκεται στη μνήμη του, ψάχνει στον τοπικό πίνακα δρομολόγησής του για κάποιο κόμβο της συστάδας του, που το αναγνωριστικό του να ταυτίζεται με το k. Αν βρει τέτοιο κόμβο του προωθεί το αίτημα. Διαφορετικά, το προωθεί στο διάδοχο του k, που είναι

ο αμέσως επόμενος υπάρχων κόμβος που έχει αναγνωριστικό $\geq k$. Η ίδια διαδικασία κατεβαίνει στα επίπεδα της ιεραρχίας μέχρι να βρεθεί ο κόμβος που έχει στην κατοχή του τα ζητούμενα δεδομένα που αντιστοιχούν στο k.

Στη *λειτουργία εξοικονόμησης ενέργειας* από την άλλη, η λογική διαδικασία διαχωρίζεται από τη φυσική, όπως ακριβώς συμβαίνει και στο Chord. Η λογική διαδικασία είναι ακριβώς η ίδια που ακολουθείται και στη λειτουργία ευρωστίας. Η διαφορά εδώ είναι ότι μόλις αποφασιστεί ο κόμβος στον οποίον θα προωθηθεί το αίτημα δεν αποστέλλεται απευθείας σε αυτόν στο φυσικό επίπεδο. Αν δηλαδή ένας κόμβος α_i υπολογίσει ότι πρέπει να προωθήσει το μήνυμά του στον α_{i+p}, όπου $p \geq 2$ το μήνυμα φτάνει τελικά στον α_{i+p} μέσω όλων των κόμβων που βρίσκονται μεταξύ τους στο λογικό δακτύλιο. Η διαδικασία θα γίνει πιο κατανοητή αν δούμε το σχήμα B της εικόνας 4.1. Ο α_1 έχει αποφασίσει σαν διάδοχο του μηνύματος του τον α_4. Ενώ στη λειτουργία ευρωστίας το μήνυμα θα σταλεί απευθείας στονα_4, στη λειτουργία εξοικονόμησης ενέργειας θα περάσει από τους κόμβους α_2 και α_3 μέχρι να καταλήξει στον α_4.

Παρόλο που η αρχική υπόθεση ήταν ότι η λειτουργία εξοικονόμησης ενέργειας θα κατανάλωνε μικρότερα ποσά ενέργειας από τη λειτουργία ευρωστίας, τα αποτελέσματα των προσομοιώσεων διαψεύδουν αυτή την υπόθεση. Το πλεονέκτημα όμως του EE_{mode} είναι ότι το σύστημα υπό τη λειτουργία του έχει μεγαλύτερη διάρκεια ζωής από το R_{mode}. Αυτό δείχνει ότι κατανέμεται πιο ομοιόμορφα η κατανάλωση ενέργειας στο σύστημα υπό τη λειτουργία του EE_{mode}.

Το CSN για να αντιμετωπίσει τα προβλήματα που προκαλούνται από αποτυχίες κόμβων, διατηρεί κάποιους επιπλέον κόμβους *(back-up nodes)* που διατηρούν αντίγραφα της πληροφορίας στη μνήμη τους. Οι κόμβοι αυτοί δε συμμετέχουν στην οργάνωση του συστήματος και τα αντίγραφα που κρατάνε ανήκουν σε κοντινούς τους αισιθητήρες. Αν ένας κόμβος αντιληφθεί την απουσία κάποιου άλλου κινεί τις κατάλληλες διαδικασίες για να μεταφερθούν τα δεδομένα του κόμβου που απέτυχε από τον back-up κόμβο στον διάδοχό του. Όπως και στο Chord, αποδεικνύεται ότι ακόμα και αν όλοι οι κόμβοι του συστήματος έχουν πιθανότητα αποτυχίας $\frac{1}{2}$, τα αιτήματα αναζήτησης κατά πάσα πιθανότητα (*with high probability*) θα επιστρέψουν το επιθυμητό αποτέλεσμα και μάλιστα σε $O(\log N)$ βήματα.

Ενώ το πρωτόκολλο αυτό παρουσιάζει αρκετές ενδιαφέρουσες ιδέες, έχει αρκετά σημεία που δεν είναι πολύ πειστικό. Οι συστάδες δημιουργούνται στην αρχική φάση οργάνωσης του δικτύου, στην οποία προσδίδονται και τα αναγνωριστικά στους κόμβους. Αυτό είναι πολύ περιοριστικό γιατί δεν αφήνει πολλά περιθώρια για άμεση προσαρμογή του δικτύου σε αφίξεις νέων κόμβων και την αποδοτική ενσωμάτωσή τους στο δίκτυο. Επίσης,η χρήση της ιεραρχίας στη συσταδοποίηση, έχει και αρνητικές συνέπειες. Κόμβοι που ανήκουν σε πολλά επίπεδα ιεραρχίας, καταναλώνουν περισσότερη ενέργεια αφού οι αποστάσεις μεταξύ των κόμβων είναι μεγαλύτερες. Επίσης, χρησιμοποιούν περισσότερη μνήμη για να διατηρούν τις επιπλέον πληροφορίες δρομολόγησης. Το [21] δεν είναι πολύ διαφωτιστικό όσον αφορά την ιεραρχική δόμηση των συστάδων. Αναφέρεται σε αυτήν χωρίς να την αξιολογεί πειραματικά, αφού τα πειράματα έχουν γίνει μόνο για ένα επίπεδο ιεραρχίας.

4.3 Scatter Pastry

Το ScatterPasrty[24], είναι ακόμα μια προσπάθεια για την αποδοτική προσαρμογή ενός πρωτοκόλλου διανομής πληροφορίας σε δίκτυα ομότιμων κόμβων(P2P) που βασίζονται σε κατανεμημένους πίνακες κατακερματισμού(DHTs), σε ασύρματα δίκτυα αισθητήρων. Στην περίπτωση αυτή, το πρωτόκολλο που προσεγγίζεται είναι το Pastry[9]. Το ScatterPastry ασχολείται με την επέκταση της λειτουργικότητα του Pastry, ώστε να μπορεί να χρησιμοποιηθεί σαν λογικό υπόστρωμα για τη δρομολόγηση μηνυμάτων στο ScatterWeb[26]. Το ScatterWeb, είναι ένας ιστός που προσφέρεται για την έρευνα στον τομέα των δικτύων αισθητήρων και όσον αφορά το υλικό αλλά και το λογισμικό.

Στο παρόν πρωτόκολλο, αρχικά όλοι οι κόμβοι παράγουν ένα λογικό αναγνωριστικό *(oevrlayID)*. Το αναγνωριστικό αυτό έχει μέγεθος 16 δυαδικών ψηφίων, σε αντίθεση με το Pastry που χρησιμοποιεί αναγνωριστικά 128 δυαδικών ψηφίων. Αυτό όπως είναι προφανές θα δίνει στο δίκτυο εύρος 65536 διαθέσιμων αναγνωριστικών. Τα αναγνωριστικά δίνονται στους κόμβους με τυχαίο τρόπο. Παρόλο που το εύρος είναι πολύ μικρότερο από αυτό του Pastry δεν τίθεται θέμα ταύτισης αναγνωριστικών, διότι το ScatterWeb διαθέτει μόνο 200 κόμβους. Η πιθανότητα λοιπόν να έχουν δύο ή παραπάνω κόμβοι το ίδιο αναγνωριστικό θεωρείται αμελητέα. Θα μπορούσαν

τα αναγνωριστικά να αποδίδονται στους κόμβους από κάποιον εξυπηρετητή, ώστε να μην τίθεται καν πρόβλημα ταύτισης, αλλά αυτό δεν είναι εφικτό στο συγκεκριμένο τύπο δικτύων.

Κάθε κόμβος, έχει στη μνήμη του δύο πίνακες. Τον πίνακα φύλλων*(leaf set table)* και τον πίνακα δρομολόγησης*(routing table)*. Ο πρώτος αποτελείται από $2 \times b$ εγγραφές. Κάθε εγγραφή περιέχει ένα αναγνωριστικό κόμβου. Οι μισές εγγραφές περιέχουν αναγνωριστικά μικρότερης τιμής από του τρέχοντα κόμβου, ενώ οι άλλες μισές μεγαλύτερης τιμής. Από την άλλη, ο πίνακας δρομολόγησης αποτελείται από *μήκος αναγνωρισιτικού/b* γραμμές και 2^b στήλες. Μια τυπική τιμή για το $b = 4$. Στην $i^{οστη}$ γραμμή του πίνακα δρομολόγησης περιέχει τα αναγνωριστικά των κόμβων που έχουν κοινό πρόθεμα i ψηφίων με τον τρέχοντα κόμβο.

Τα μηνύματα σε αυτό το πρωτόκολλο δρομολογούνται σύμφωνα με την ταύτιση του μέγιστου προθέματος αναγνωριστικού κόμβου και κλειδιού. Αν το κλειδί k είναι στο εύρος των αναγνωριστικών του πίνακα φύλλων προωθείται στον κόμβο που του αντιστοιχεί. Σε κάθε άλλη περίπτωση ο κόμβος που θέλει να προωθήσει το κλειδί, εξετάζει τον πίνακα δρομολόγησής του. Βρίσκει τον κόμβο με αναγνωριστικό που ταυτίζεται με μέγιστο πρόθεμα στο κλειδί και του το προωθεί. Ο αριθμός τον μεταδόσεων μέχρι να λάβει το μήνυμά του ο παραλήπτης είναι ($O(\log_{2^b} N)$, όπου N ο συνολικός αριθμός κόμβων του δικτύου.

Στα ασύρματα δίκτυα αισθητήρων, οι αφίξεις κόμβων που αποσκοπούν στην επέκτασή του και οι αποχωρήσεις των κόμβων λόγω ενεργειακής εξάντλησης είναι πολύ συχνά φαινόμενα. Οι αλλαγές αυτές αντιμετωπίζονται από το ScatterPastry ως εξής:

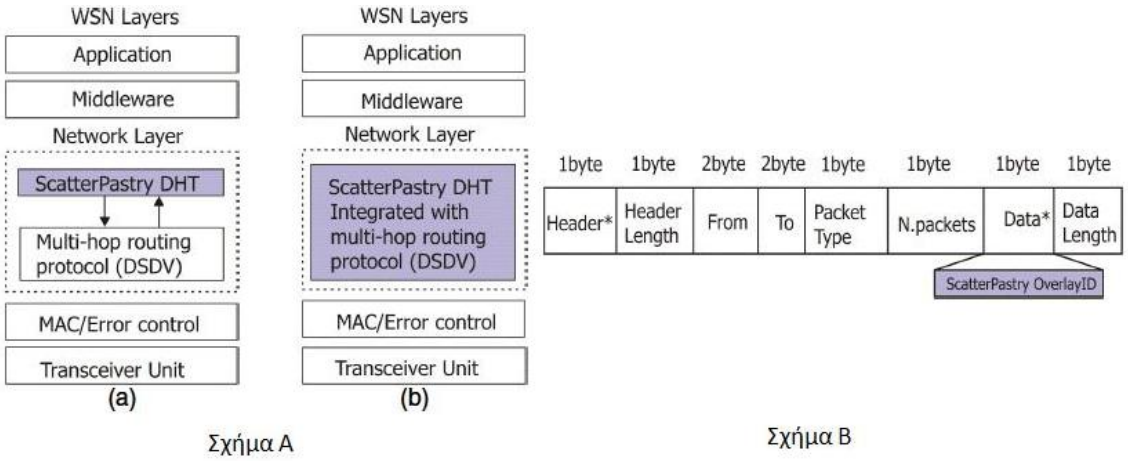
Αφίξεις κόμβων

Όταν ένας νέος κόμβος επιθυμεί να εισέλθει στο δίκτυο πρώτον επιλέγει τυχαία ένα αναγνωριστικό(ScatterPastryID), έστω X. Έπειτα, πρέπει να αρχικοποιήσει τους δυο πίνακές του. Για την περίπτωση του πίνακα φύλλων,εκπέμπει ένα σήμα και ανάλογα με τις αποκρίσεις συλλέγει τους γείτονές του. Επιλέγει τυχαία έναν από αυτούς, έστω τον A. Μέσω του A στέλνει μια "αίτηση-συμμετοχής"*(join request)* στο δίκτυο. Το μήνυμα αυτό περιέχει σαν κλειδί το X και τον πίνακα δρομολόγησής του κενό. Ο A μεταδίδει το μήνυμα μέσω ενός συνόλου κόμβων στον Z. Z θα είναι ο κοντινότερος κόμβος στο αναγνωριστικό X. Κάθε $i^{στος}$ κόμβος του μονοπατιού $A \rightarrow Z$ θα προσθέτει στον πίνακα δρομολόγησης την $i^{στη}$ γραμμή του δικού του πίνακα δρομολόγησης.

Αφού συμπληρωθεί ο πίνακας και το αίτημα έχει φτάσει πλέον στον Z, το μήνυμα επιστρέφει στον X. Αυτός κάνει τις αντίστοιχες ενημερώσεις στους πίνακές του. Τέλος, ο X ενημερώνει όλους τους κόμβους, τους οποίους αφορά για την άφιξή του. Η άφιξη ενός νέου κόμβου κοστίζει στο δίκτυο $O(\log_{2^b} N$ μεταβάσεις μηνυμάτων.

Αναχωρήσεις κόμβων

Η περιοδική ανταλλαγή ενημερωτικών μηνυμάτων*keep-alive messages* μεταξύ των γειτονικών κόμβων αποσκοπεί στην άμεση ενημέρωση του δικτύου για την απώλεια κόμβων. Όταν ένας κόμβος u πάψει να λαμβάνει τέτοια μηνύματα από κάποιον από τους γείτονές του, αντιλαμβάνεται ότι αυτός δεν υπάρχει πλέον στο δίκτυο και τον διαγράφει από τους πίνακές του. Στην περίπτωση που κάποιος κόμβος προσπαθήσει να προσπελάσει κενή εγγραφή από τον πίνακά του, αποφασίζει να προωθήσει το κλειδί στον αμέσως επόμενο(σύμφωνα με τα κριτήρια του ταύτισης προθεμάτων) κοντινότερο κόμβο του πίνακα δρομολόγησής του. Με αυτόν τον τρόπο, το ScatterPastry εγγυάται ότι και μετά από αναχωρήσεις κόμβων δεν θα υπάρχουν καθυστερήσεις στην παράδοση των μηνυμάτων στον κατάλληλο προορισμό.



Εικόνα 4.2: Σχήμα Α: Επίπεδα Αρχιτεκτονικής για τις δύο περιπτώσης εφαρμογής του ScatterPastry, Σχήμα Β: Δομή ScatterPastry πακέτου

Δύο τρόποι προτείνονται για την τελική εφαρμογή του ScatterPastry σε ασύρματα δίκτυα αισθητήρων. Στη μια περίπτωση θα υλοποιηθεί πάνω από ένα φυσικό πρωτόκολλο δρομολόγησης πολλαπλών μεταβάσεων*(physical multihop protocol)*. Στη συγκεκριμένη μελέτη χρησιμοποιείται το (DSDV-Destination Sequenced Distance Vector[25]). Διαφορετικά μπορεί να ενσωματωθεί μαζί με το κάποιο πρωτόκολλο

δρομολόγησης στο επίπεδο δικτύου του συστήματος. Στο σχήμα Α της εικόνας 4.2 φαίνονται οι δύο αυτές επιλογές, ενώ στο σχήμα της εικόνας 4.3 παρουσιάζεται μια πιθανή δομή του ScatterPastry πακέτου στο επίπεδο δικτύου. Το πεδίο *Header** δεικτοδοτεί τα δεδομένα.

Ουσιαστικά, το ScatterPastry δεν παρουσιάζει ιδιαίτερες διαφορές με το Pastry πέραν του μεγέθους κάποιον οντοτήτων που χρησιμοποιούν. Η μελέτη αυτή απλά δείχνει την απόδοση μιας πιθανής εφαρμογής του πρωτοκόλλου Pastry. Γίνεται σύγκριση με έναν απλό αλγόριθμο πλημμύρας*flooding* και όπως είναι αναμενόμενο το scatterPastry είναι αποδοτικότερο ειδικά σε περιπτώσεις κλιμάκωσης του δικτύου. Αποδοτικότερο και όσον αφορά την κατανάλωση ενέργειας αλλά και την καθυστέρηση παράδοσης των μηνυμάτων στον αποστολέα. Όπως το CSN, έτσι και το ScatterPastry δεν ασχολούνται καθόλου με την περίπτωση δικτύων αισθητήρων με κινούμενους κόμβους, ζήτημα πολύ σημαντικό για τη μελέτη των δικτύων αισθητήρων.

4.4 GHT(Geographic Hash Tables)

Το GHT[22] έχει κάποια κοινά χαρακτηριστικά με το CAN[6] και αποτελεί πιθανόν την πιο ενδιαφέρουσα από τις προσεγγίσεις που παρουσιάστηκαν. Αυτός ήταν και ο λόγος που μας ώθησε να ασχοληθούμε εκτενέστερα με αυτό το πρωτόκολλο και να το υλοποιήσουμε στον προσομοιωτή ασυρμάτων δικτύων αισθητήρων Shawn.

4.4.1 Βασικές Υποθέσεις και Μετρικές του GHT

Τα δίκτυα τα οποία οι σχεδιαστές του πρωτοκόλλου προτείνουν ως κατάλληλα για την εφαρμογή του Γεωγραφικού Πίνακα Κατακερματισμού, πρέπει να τηρούν κάποιες βασικές προϋποθέσεις.

- Υποθέτουμε δίκτυα αισθητήρων είναι μεγάλης κλίμακας με κόμβους διασκορπισμένους τυχαία σε περιοχή με γνωστά κατά προσέγγιση όρια.

- Οι κόμβοι γνωρίζουν τη γεωγραφική τους θέση. Αυτό είτε με τη χρήση ενός GPS, είτε με τη βοήθεια κάποιου αλγορίθμου εντοπισμού της θέσης τους στο χώρο(*localization algorithm*).

- Το δίκτυο συνδέεται με τον "εξωτερικό κόσμο" με ένα μικρό αριθμό από σημεία πρόσβασης (*access points*). Αυτή η υπόθεση δεν είναι απαραίτητη για την υλοποίηση του μηχανισμού που περιγράφεται αλλά χρησιμοποιείται για την σύγκριση του πρωτοκόλλου με άλλους μηχανισμούς διάδοσης.

- Οι κόμβοι κινούνται, όχι όμως συχνά από ένα όριο T_h, που περιγράφεται παρακάτω.

4.4.2 Η Προσέγγιση του GHT (Geographic Hash Table)

Οι δύο λειτουργίες που εξυπηρετούνται από ένα τέτοιο σύστημα είναι :

- **Put (k,v)** : Αποθηκεύει τα δεδομένα (που παρατήρησαν οι αισθητήρες) ανάλογα με το κλειδί k, το όνομα των δεδομένων.

- **Get (k)** : Ανακτά όποια τιμή είναι αποθηκευμένη στο δίκτυο και σχετίζεται με το κλειδί k.

Και οι δύο λειτουργίες αντιστοιχίζουν το k μέσω μιας ομοιόμορφης συνάρτησης κατακερματισμού στην ίδια γεωγραφική θέση. Κάθε ζεύγος αποθηκεύεται στον κόμβο που βρίσκεται πιο κοντά σε αυτή τη θέση. Η "συνεπής" επιλογή αυτού του κόμβου είναι το κεντρικό κομμάτι για την δόμηση ενός GHT(Γεωγραφικό πίνακα κατακερματισμού). Οι υπηρεσίες που παρέχονται από το GHT είναι παρόμοιες με αυτές που προσφέρουν τα DHT συστήματα που παρουσιάσαμε και είναι επηρεασμένο κυρίως από το CAN[6] . Επιπλέον το GHT προσφέρει τόσο κλιμακωσιμότητα, όσο και ευρωστία στο επίπεδο πολλών ειδών σφαλμάτων που μπορούν να προκύψουν σε ένα κατανεμημένο σύστημα. Αυτό επιτυγχάνεται με το *Πρωτόκολλο ανανέωσης Περιμέτρου(Perimeter Refresh Protocol)*. Με τη χρήση αυτού, το GHT εγγυάται διατήρηση και συνέπεια στο δίκτυο, όταν οι κόμβοι αποτυγχάνουν ή κινούνται.

Το πρωτόκολλο αυτό, αντιγράφει τα αποθηκευμένα δεδομένα που αντιστοιχούν σε ένα κλειδί k, σε κόμβους γύρω από την γεωγραφική θέση στην οποία αντιστοιχίζεται το κλειδί k με τη χρήση της συνάρτησης κατακερματισμού. Σημαντικό γεγονός είναι ότι εγγυάται ότι ένας κόμβος επιλέγεται με συνέπεια σαν *"κόμβος-εστία"(home node)* για ένα κλειδί k, έτσι ώστε όλα τα αιτήματα που αναζητούν το k να μπορούν να δρομολογηθούν πάντα σε αυτό τον κόμβο. Βασικό χαρακτηριστικό του ght είναι η αποδοτικότητά του. Χρησιμοποιεί τοπική επικοινωνία, ειδικά όταν το δίκτυο

είναι πυκνό. Με τη χρήση του κατακερματισμού διαμοιράζει το φόρτο αποθήκευσης και επικοινωνίας ομοιόμορφα μέσα στο δίκτυο. Για την αποθήκευση πολλών *γεγονότων(events)* με το ίδιο κλειδί k, το GHT αποφεύγει τη δημιουργία σημείων συμφόρησης(hotspots) στον home node. Αυτό το επιτυγχάνει με τη χρήση της *δομημένης αντιγραφής(structured replication)*, όπου τα γεγονότα που αντιστοιχίζονται στο ίδιο σημείο μπορούν να διαμοιραστούν μεταξύ πολλαπλών *κατοπτρικών κόμβων(mirrors)*.

4.4.3 Αλγόριθμοι

Στο GHT συνδυάζονται μια σειρά νέων αλλά και ήδη γνωστών αλγορίθμων.

- **Δρομολόγηση - GPSR(Greedy Perimeter Statless Routing[23]**

  Για καθαρά το κομμάτι της δρομολόγησης μηνυμάτων στον προορισμό τους χρησιμοποιείται το πρωτόκολλο GPSR το οποίο περιγράψαμε στο πρώτο κεφάλαιο αναλυτικά. Εν συντομία το ελαφρώς διαφοροποιημένο GPSR που χρησιμοποιεί το GHT, έχει τα εξής χαρακτηριστικά:

  Πίνακες Δρομολόγησης Κόμβων

  Κάθε κόμβος διατηρεί μια λίστα με τους γειτονικούς κόμβους που απέχουν από αυτόν μόνο μια μετάδοση(hop). Για καθένα από αυτούς, αποθηκεύει το μοναδικό τους αναγνωριστικό και τις γεωγραφικές τους συντεταγμένες. Η διαδικασία δημιουργίας των πινάκων περιγράφεται στο GPSR[23].

  Πακέτα GHT

  Τα πακέτα GHT έχουν την ίδια δομή με αυτά του GPSR. Περιέχουν τις συντεταγμένες προορισμού. Αυτές παράγονται με την αντιστοίχηση του ενός κλειδιού k σε ένα σημείο, μέσω της ομοιόμορφης συνάρτησης κατακερματισμού. Το κλειδί αυτό σχετίζεται είτε με δεδομένα προς αποθήκευση, είτε με δεδομένα προς αναζήτηση.

  Άπληστη Προώθηση(Greedy Forwarding Mode)

  Όταν δημιουργούνται τα πακέτα τίθενται αρχικά σε λειτουργία άπληστης προώθησης και όσο είναι σε αυτή τη λειτουργία προωθούνται πάντα στον γείτονα που η ευκλείδεια απόστασή του από τον προορισμό του πακέτου είναι μικρότερη από αυτή όλων των άλλων γειτόνων. Αν αυτός ο γείτονας είναι πιο μακρυά από τον προορισμό από τον ίδιο τον κόμβο το πακέτο τίθεται σε λειτουργία Περιμετρικής Δρομολόγησης*(Perimeter Routing mode)*. Ο κόμβος που θέτει το πακέτο σε

(Perimeter Routing mode) σημειώνει στο πακέτο τις γεωγραφικές του συντεταγμένες.

Περιμετρική Δρομολόγηση(Perimeter Routing mode)

Στην περίπτωση λοιπόν που η άπληστη προώθηση αποτύχει όπως είπαμε παραπάνω ο κόμβος, έστω u που θέλει να προωθήσει το πακέτο εξετάζει και πάλι τον πίνακα δρομολόγησής του. Υπολογίζει τον πιο κοντινό του αριστερόστροφο γείτονα με αρχή το ευθύγραμμο τμήμα μεταξύ του u και του προορισμού και προωθεί σε αυτόν το πακέτο. Κάθε φορά που ένας κόμβος λαμβάνει πακέτο που είναι σε perimeter mode, ελέγχει αν ο ίδιος ή κάποιος γείτονάς,του είναι πιο κοντά στον προορισμό του πακέτου, και αν δεν είναι εργάζεται όπως και παραπάνω συνεχίζοντας το perimeter mode. Αν δεν ξεκινάει από τον τρέχοντα κόμβο η περιμετρική δρομολόγηση, το ευθύγραμμο τμήμα που θεωρείται ως αρχή για να βρει τον κοντινότερο αριστερόστροφο γείτονά του είναι η ακμή από την οποία έλαβε το πακέτο. Η ακμές που τέμνουν το τμήμα που δημιουργούν τα σημεία εκκίνησης περιμετρικής δρομολόγησης και προορισμού, δεν χρησιμοποιούνται. Αν βρεθεί κοντινότερος κόμβος από τον κόμβο εκκίνησης, επιστρέφει το πακέτο σε greedy mode.

- **Home Node και Home Perimeter**

  Σαν κόμβος-εστία*(homde node)*ενός ζεύγους (k, v) ορίζεται ως ο γεωγραφικά κοντινότερος κόμβος στη θέση στην οποία αντιστοιχίζεται το κλειδί k.

  Διαδικασία επιλογής home node Όταν ένας κόμβος θέλει να κάνει $Put(k, v)$, κάνει αντιστοιχίζει το k σε γεωγραφικές συντεταγμένες ενός σημείου $P(x, y)$ και δημιουργεί πακέτο με προορισμό το σημείο αυτό . Έπειτα το πακέτο δρομολογείται με τη χρήση του GPSR στον προορισμό του. Όταν το πακέτο φτάσει στον κόμβο, έστω w ο οποίος βρίσκεται γεωγραφικά πιο κοντά στο P από όλους τους άλλους κόμβους του δικτύου, το πακέτο θα μπει σε Perimeter Routing και ο w θα σημειώσει στο πακέτο τις συντεταγμένες του ως συντεταγμένες αρχής perimeter routing mode. Το πακέτο ακολουθώντας τον κανόνα του δεξιού χεριού(right hand rule), μετά από μερικά hops, θα επιστρέψει στον κόμβο w, ο οποίος θα αναγνωρίσει ότι αυτός ξεκίνησε το perimeter mode. Αν συμβεί αυτό ο w θα ορίζεται ως ο *κόμβος εστία(home node)* του k. Ενώ η διαδρομή που

ακολούθησε το πακέτο από την στιγμή που μπήκε σε περιμετρική δρομολόγηση μέχρι να επιστρέψει στον w θα είναι η *εστιακή περίμετρος(home perimeter) του* k.

Ο μηχανισμός αυτός λειτουργεί σωστά μόνο για στατικά δίκτυα χωρίς αποτυχίες. Για την εξασφάλιση της σωστής λειτουργίας του GHT σε περιπτώσεις κίνησης, αποτυχίας ή εισαγωγής νέων κόμβων στο δίκτυο χρησιμοποιείται το *Πρωτόκολλο Ανανέωσης Περιμέτρου(Perimeter Refresh Protocol).*

- **Perimeter Refresh Protocol (PRP)**

  Το PRP χρησιμοποιείται για την συνεπή τοποθέτηση των ζευγών$(κλειδ, τιμ)$ στους κατάλληλους home nodes παρ'Α όλες τις αλλαγές της τοπολογίας του δικτύου. Οι κόμβοι που βρίσκονται στο στην εστιακή περίμετρο ενός ζεύγους $(κλειδ, τιμ)$ είναι και οι κόμβοι αντιγράφων*(replica nodes)* του ζεύγους αυτού. Η συνθήκη την οποία αναγνωρίζει ένας κόμβος για να αποφασίσει ότι είναι ο κόμβος-εστία ενός ζεύγους διαπιστώνεται από τα δεδομένα του πακέτου, όπως περιγράψαμε στην προηγούμενη παράγραφο. Το PRP παράγει περιοδικά μηνύματα ανανέωσης*(refresh messages)* με τη χρήση ενός απλού σχήματος χρονισμού. Κάθε T_h δευτερόλεπτα, ο κόμβος-εστία ενός κλειδί k δημιουργεί ένα πακέτο ανανέωσης με προορισμό το σημείο που αντιστοιχεί στο k. Το πακέτο αυτό περιέχει και τα αντίστοιχα δεδομένα (τιμή), του ζεύγους και δρομολογείται ακριβώς με τον ίδιο τρόπο που δρομολογούνται και τα Put και Get πακέτα στο GHT. Αυτό το πακέτο θα ακολουθήσει την "σωστή", δηλαδή την τρέχουσα εστιακή περίμετρο του κλειδιού k, άσχετα από τις αλλαγές στην τοπολογία του δικτύου από τη στιγμή της εισαγωγής του.

  Αν ένας κόμβος λάβει ένα πακέτο ανανέωσης και βρίσκεται πιο κοντά στον προορισμό του από ότι η πηγή του του πακέτου, το καταναλώνει και δημιουργεί ένα καινούριο γιαυτό το κλειδί αυτό. Αν πάλι δεν είναι , εισάγει τα δεδομένα του πακέτου στην λανθάνουσα μνήμη του*(cache)* του και θέτει για αυτά, έναν χρονιστή ανάληψης*(takeover timer)* T_t, οποίος ανανεώνεται κάθε φορά που φτάνει ένα πακέτο ανανέωσης για αυτό το κλειδί από οποιονδήποτε κόμβο. Μόλις αυτός ο χρονιστής μηδενιστεί ο κόμβος αναλαμβάνει να εκκινήσει ο ίδιος ένα πακέτο ανανέωσης για το κλειδί αυτό. Και στις δύο περιπτώσεις ο κόμβος που λαμβάνει το πακέτο ανανέωσης του προσθέτει και όλα τα ζεύγη $(κλειδ, τιμ)$ που

σχετίζονται με αυτό το κλειδί τα οποία έχει ο ίδιος στη λίστα δεδομένων του. Όταν ένα πακέτο ανανέωσης επιστρέψει στον κόμβο-πηγή του και αυτός δεν είναι ο προηγούμενος κόμβος-εστία του ζεύγους αυτού, τότε το πακέτο καταναλώνεται και ο κόμβος θέτει τον εαυτό του ως νέο κόμβο-εστία του κλειδιού και θέτει τον δικό του χρονιστή ανανέωσης*(Refresh timer)*, T_h για το κλειδί αυτό.

Όλοι οι κόμβοι διατηρούν για τα δεδομένα που έχουν στην στην λανθάνουσα μνήμη τους, ακόμη έναν χρονιστή *(exmpiration timer)*T_d, που όταν μηδενιστεί διαγράφεται και το αντίστοιχο ζεύγος $(κλειδ, τιμ)$. Ο χρονιστής αυτός ανανεώνεται κάθε φορά που λαμβάνεται πακέτο ανανέωσης για το κλειδί του. Προφανώς πρέπει να ισχύει $T_d < T_h$ και $T_t < T_h$. Με αυτόν τον τρόπο διασφαλίζεται η σωστή τοποθέτηση των δεδομένων στον κόμβο που είναι γεωγραφικά πιο κοντά στις συντεταγμένες που παράγει η συνάρτηση κατακερματισμού για το κλειδί. Παρομοίως διασφαλίζεται και η ανάκτηση οποιασδήποτε πληροφορίας ζητείται ακόμα και αν ο κόμβος που αρχικά την αποθήκευε πέσει ή μετακινηθεί. Τέλος, ο μηχανισμός αυτός ανταποκρίνεται επιτυχημένα σε αφίξεις νέων κόμβων, τους οποίους συμπεριλαμβάνει σχεδόν αμέσως στη δρομολόγηση και την αποθήκευση δεδομένων.

- **Δομημένη Αντιγραφή(Structured Replication-SR)**

  Το πρωτόκολλο που περιγράψαμε μέχρι στιγμής, αποθηκεύει όλα τα γεγονότα που αντιστοιχούν στο ίδιο κλειδί στο ίδιο σημείο. Αν όμως συλλεχθούν πολλά δεδομένα για το ίδιο κλειδί αυτό μπορεί να δημιουργήσει πρόβλημα συμφόρησης και αποθηκευτικού χώρου, αλλά και επικοινωνιακού φορτου*(hotspot nodes)*. Το πρόβλημα αυτό επιχειρεί να επιλύσει η SR με την εξής τεχνική. Προσαυξάνει τα ονόματα των γεγονότων με ένα ιεραρχικό βάθος και χρησιμοποιεί ιεραρχική αποσύνθεση στων χώρο που ανήκουν τα κλειδιά. Έστω ότι ονομάζουμε ρίζα*(root)* ενός κλειδιού k, το σημείο στο οποίο αντιστοιχίζεται με τη χρήση της ομοιόμορφης συνάρτησης κατακερματισμού. Δεδομένης της ρίζας r και βάθους ιεραρχίας d μπορούν να υπολογιστούν $4^d - 1$ κατοπτρικές εικόνες του k.
  Ένας κόμβος που θέλει να εισάγει πληροφορία στο δίκτυο, την τοποθετεί στο κοντινότερο σε αυτόν κατοπτρικό σημείο που αντιστοιχεί στο κλειδί της πληροφορίας. Με αυτόν τον τρόπο το SR μειώνει και το κόστος αποθήκευσης

που αντιστοιχεί σε ένα κλειδί για το οποίο έχουν εντοπιστεί n γεγονότα, από $O(\sqrt{n})$ σε $O(\sqrt{n}/2^d)$. από την άλλη πλευρά το κόστος αναζήτησης αυξάνεται από $O(\sqrt{n})$ σε $O(2^d\sqrt{n})$. Αυτό συμβαίνει διότι τώρα πλέον η αναζήτηση περνά αναδρομικά από όλα τα d επίπεδα ιεραρχίας μέχρι να βρεθεί το ζητούμενο αποτέλεσμα. Η αναζήτηση ξεκινά από τη ρίζα του κλειδιού, έπειτα προχωρά στα $4^d-1 = 4-1 = 3$ σημεία του πρώτου επιπέδου ιεραρχίας, τα $4^d-1 = 4^2-1 = 15$ του δευτέρου κοκ.

Το GHT λοιπόν παρουσιάζει πολλά ενδιαφέροντα χαρακτηριστικά και μια ποικιλία αλγορίθμων για την αντιμετώπιση διαφόρων προβλημάτων. Τα αποτελέσματα των πειραμάτων που παρουσιάζονται στο [22] είναι αρκετά κατατοπιστικά και αρκετά κοντά στα αναμενόμενα. Οι λόγοι αυτοί μας οδήγησαν στην προσπάθεια της υλοποίησης του πρωτοκόλλου σε ένα προσομοιωτή, που προορίζεται για ασύρματα δίκτυα αισθητήρων. Ο προσομοιωτής αυτός διαφέρει από τα παραδοσιακά εργαλεία προσομοίωσης και θεωρήσαμε ότι είναι ο πιο κατάλληλος για τη μελέτη ενός τέτοιου πρωτοκόλλου.

Κεφάλαιο 5

Ο Προσομοιωτής Ασυρμάτων Δικτύων Αισθητήρων Shawn

Τα ασύρματα δίκτυα αισθητήρων έχουν πολλά ιδιαίτερα χαρακτηριστικά που τα ξεχωρίζουν από άλλες κατηγορίες δικτύων. Λόγω αυτού, έχουν εντελώς διαφορετικές ανάγκες από τα εργαλεία προσομοίωσης των πρωτοκόλλων τους. Ο προσομοιωτής Shawn[27], υλοποιήθηκε για ακριβώς αυτό το σκοπό. Είναι ένα εργαλείο υλοποιημένο στη γλώσσα προγραμματισμού C++ που απευθύνεται αποκλειστικά σε προσομοιώσεις που αφορούν ασύρματα δίκτυα αισθητήρων.

Ο προσομοιωτής αυτός διαφέρει με ποικίλους τρόπους από τα περισσότερα εργαλεία προσομοιώσεων όπως είναι ο NS2 και ο TOSSIM. Η σημαντικότερη διαφορά που έχει με αυτά τα γνωστά εργαλεία είναι ότι δεν είναι εξειδικευμένος. Προφανώς, δεν μπορεί να ανταγωνιστεί τα εργαλεία αυτά σε προσομοιώσεις πρωτοκόλλων που περιγράφουν τα χαρακτηριστικά τους σε μεγάλη λεπτομέρεια. Αντιθέτως, έχει αναδυθεί από τον αλγοριθμικό τομέα και οι κύριοι σχεδιαστικοί στόχοι του είναι:

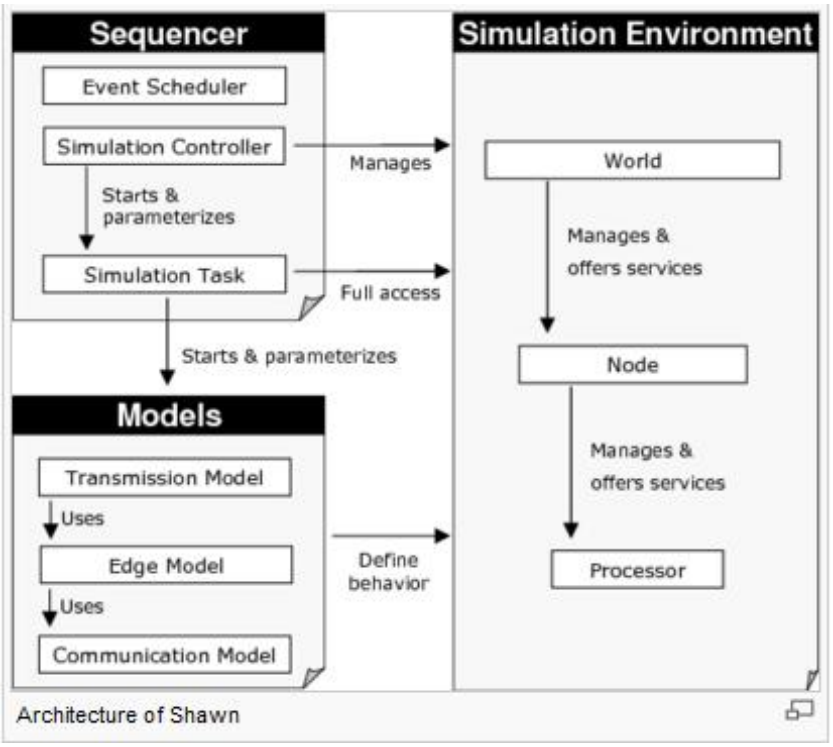
- Να προσομοιώνει τα αποτελέσματα που μπορούν να προκληθούν από κάποια φαινόμενα και όχι τα ίδια τα φαινόμενα.

- Η συμπεριφορά των αλγορίθμων κατά την κλιμάκωση των δικτύων. Μπορεί να προσομοιώσει δίκτυα ιδιαίτερα μεγάλης κλίμακας.

- Να είναι ελεύθερη η επιλογή του μοντέλου υλοποίησης.

5.1 Αρχιτεκτονική

Η αρχιτεκτονική του Shawn φαίνεται στην εικόνα 5.1 και αποτελείται από τρία βασικά μέρη:



Εικόνα 5.1: Η αρχιτεκτονική του προσομοιωτή Shawn

1. Τα μοντέλα

2. Ο ακολουθητής(Sequencer)

3. Το περιβάλλον προσομοίωσης

Κάθε πλευρά του Shawn είναι επηρεασμένη από ένα ή περισσότερα μοντέλα, τα οποία είναι το κλειδί για την προσαρμοστικότητα και την κλιμακωσιμότητά του. Ο ακολουθητής είναι η κεντρική μονάδα συντονισμού του Shawn αφού ρυθμίζει τις παραμέτρους των προσομοιώσεων, εκτελεί σειριακά τις αποστολές(*tasks*) και ελέγχει όλη την προσομοίωση. Τέλος, το περιβάλλον υλοποίησης είναι η εστία για τον εικονικό κόσμο μέσα στον οποίο διανέμονται οι κόμβοι-αισθητήρες που προσομοιώνονται.

5.1.1 Μοντέλα

Ο προσομοιωτής Shawn κάνει ένα διαχωρισμό μεταξύ των μοντέλων και των σχετικών με αυτά υλοποιήσεων. Ένα μοντέλο αποτελεί τη διεπαφή που χρησιμοποιεί ο Shawn για να διατηρεί τον έλεγχο της προσομοίωσης, χωρίς να έχει καμία γνώση για τις λεπτομέρειες κάποιας συγκεκριμένης υλοποίησης. Ο Shawn διατηρεί ένα αρχείο από υλοποιήσεις μοντέλων που μπορούν να χρησιμοποιηθούν και να αποτελέσουν τις απαραίτητες ρυθμίσεις με την επιλογή των επιθυμητών συμπεριφορών(behaviors). Η υλοποίηση ενός μοντέλου μπορεί να είναι απλοποιημένη και γρήγορη, ή μπορεί να παρέχει πολύ καλές προσεγγίσεις στην πραγματικότητα. Έτσι, αφήνεται στο χρήστη η επιλογή της πιο κατάλληλης υλοποίησης κάθε μοντέλου που θα ρυθμίσει τελικά και την αποστολή της προσομοίωσης. Τα τρία μοντέλα που αποτελούν την υποδομή του Shawn είναι τα εξής:

- Μοντέλο Επικοινωνίας*(Communication Model)*

  Το μοντέλο αυτό καθορίζει ποιοι είναι οι υποψήφιοι παραλήπτες των μηνυμάτων που στέλνει κάθε κόμβος. Δεν καθορίζει τις ιδιότητες κάθε μετάδοσης. Ορίζει όμως τα ζεύγη των κόμβων που μπορούν να επικοινωνήσουν μεταξύ τους. Ο προσομοιωτής έχει υλοποιημένα τα παρακάτω μοντέλα. Το μοντέλο Unit Disk Graph(UDG) στο οποίο κάθε κόμβος u επικοινωνεί με ένα κόμβο v αν ο v βρίσκεται εντός του δίσκου ακτίνας R που έχει κέντρο τον κόμβο u. Το μοντέλο Quasi-Unit Disk Graph(QUDG) το οποίο χαρακτηρίζεται από 2 παραμέτρους R_1 και R_2. Έχει ίδια συμπεριφορά με το UDG με για $R = R_1$, ενώ για το διάστημα $R_1 < R < R_2$ η πιθανότητα παραλαβής μηνύματος μειώνεται γραμμικά από 1 σε 0. Το μοντέλο Radio Irregularity Model(RIM), το οποίο προτείνει ένα εύρος εκπομπής για κάθε κόμβο που εξαρτάται από τη γωνία μετάδοσης και είναι ένα μοντέλο που ίσως αντικατοπτρίζει περισσότερο την πραγματικότητα σε σχέση με τα υπόλοιπα. Τέλος, υπάρχει υλοποιημένο στον Shawn και το Chained Communication Model, το οποίο χρησιμοποιείται για το συνδυασμό κάποιων ή όλων τα παραπάνω μοντέλων.

- Μοντέλο Ακμών*(Edge Model)*

  Αυτό το μοντέλο προσφέρει μια αναπαράσταση γράφου του δικτύου. Η κόμβοι που προσομοιώνονται είναι οι κορυφές, και οι ακμές είναι οι υπαρκτοί σύνδεσμοι

που επιστρέφει το μοντέλο επικοινωνίας. Μέσω αυτού μπορούν να προσπελαστούν οι γείτονες ενός κόμβου, αφού ενημερώνεται συνεχώς για την κατάσταση του δικτύου από το μοντέλο επικοινωνίας. Τα βασικά μοντέλα που έχουν υλοποιηθεί είναι το απλό(simple), το πλέγμα(grid), η λίστα(list) και η ταχεία λίστα(fast list).

- Μοντέλο Μετάδοσης*(Transmission Model)*
  Κάθε φορά που ένας κόμβος μεταδίδει ένα μήνυμα, η συμπεριφορά του καναλιού μετάδοσης μπορεί να είναι εντελώς διαφορετική από ότι σε προηγούμενες μεταδόσεις μηνυμάτων. Για τη μοντελοποίηση αυτών των διαφορετικών συμπεριφορών, ο Shawn διαθέτει το μοντέλο μετάδοσης. Τυχαία μπορεί να καθυστερήσει, να τροποποιήσει ή να προκαλέσει την απώλεια μηνυμάτων.

5.1.2 Ακολουθητής

Ο ακολουθητής αποτελεί το κέντρο ελέγχου της προσομοίωσης: προετοιμάζει τον "κόσμο" που θα φιλοξενήσει τους κόμβους που θα προσομοιωθούν, παραμετροποιεί και ενεργοποιεί τις υλοποιήσεις των μοντέλων που θα χρησιμοποιηθούν από την προσομοίωση, και γενικότερα έχει τον έλεγχο όλης της διαδικασίας της προσομοίωσης. Τα τμήματα από τα οποία αποτελείται ο ακολουθητής είναι τα παρακάτω:

- Αποστολές Προσομοίωσης*(Simulation Tasks)*
  Αυτές είναι κομμάτια από κώδικά του χρήστη, που καλούνται από τη διαμόρφωση(configuration) της προσομοίωσης. Δεν σχετίζονται άμεσα με την εφαρμογή που προσομοιώνεται, αλλά έχουν πρόσβαση σε ολόκληρο το περιβάλλον προσομοίωσης και έτσι είναι σε θέση να εκτελέσουν μεγάλο εύρος λειτουργιών.

- Ελεγκτής Προσομοίωσης*(Simulation Controller)*
  Αυτός λειτουργεί ως κεντρική μονάδα διαχείρισης όλων των διαθέσιμων υλοποιήσεων μοντέλων. Εκτελεί την προσομοίωση μεταμορφώνοντας τη διαμορφωμένη είσοδο σε παραμετροποιημένες κλήσεις των Αποστολών προσομοίωσης. Έτσι αποτελεί δίοδο επικοινωνίας μεταξύ του πυρήνα της προσομοίωσης και του χρήστη.

- Δρομολογητής γεγονότων*(Event Scheduler)*

Ο Shawn χρησιμοποιεί έναν δρομολογητή διακεκριμένων γεγονότων για να μοντελοποιήσει το χρόνο. Αντικείμενα που χρειάζονται να αλληλεπιδρούν με την έννοια του χρόνου, αρχικοποιούνται στο δρομολογητή ώστε να μπορούν να εντοπιστούν σε κάθε τυχαία χρονική στιγμή. Αυτή η προσέγγιση του Shawn πλεονεκτεί έναντι άλλων παραδοσιακών προσεγγίσεων που χρησιμοποιούν σταθερά χρονικά διαστήματα.

5.1.3 Περιβάλλον

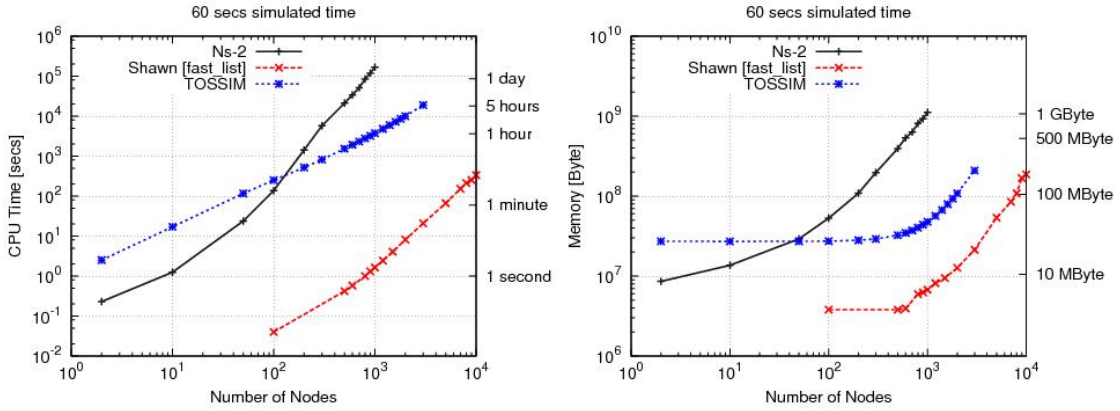
Το περιβάλλον προσομοίωσης ουσιαστικά αποτελεί τον εικονικό κόσμο μέσα στον οποίο αναπτύσσονται τα αντικείμενα προσομοίωσης. Οι κόμβοι που προσομοιώνονται αποτελούν τον λεγόμενο κόσμο. Και κάθε κόμβος "φιλοξενεί" τις οντότητες που καλούνται Επεξεργαστές(Processors).

Οι χρήστες του Shawn υλοποιούν τη λογική των εφαρμογών τους με τη χρήση των οντοτήτων Επεξεργαστών. Το Shawn, αποσυνδέοντας ουσιαστικά κάθε επεξεργαστή από κάθε κόμβο, προσφέρει ένα μεγάλο πλεονέκτημα. Επιτρέπει τον συνδυασμό πολλαπλών εφαρμογών σε μια προσομοίωση. Μπορεί για παράδειγμα ένας επεξεργαστής να τρέχει ένα πρωτόκολλο, ενώ παράλληλα κάποιος άλλος να συλλέγει στατιστικά δεδομένα.

5.2 Απόδοση

5.2.1 Σύγκριση με τους προσομοιωτές NS2 και TOSSIM

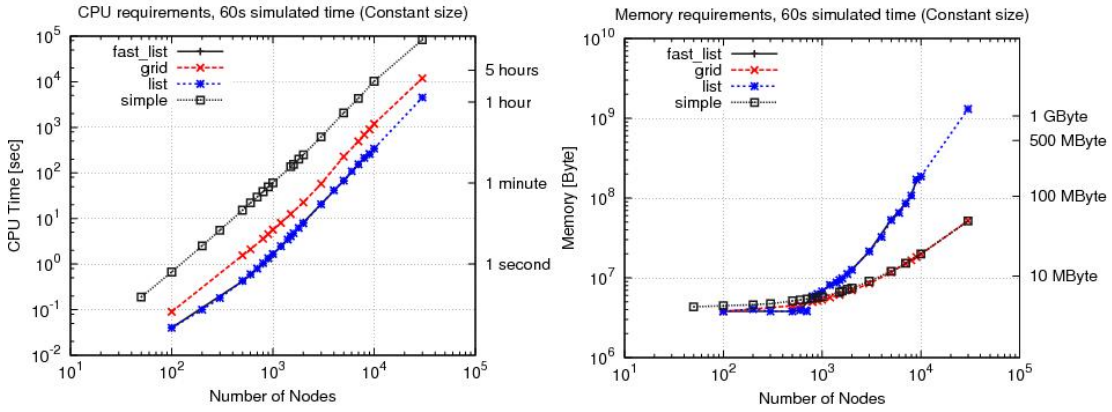
Μιας και η ανταλλαγή ασύρματων μηνυμάτων είναι το βασικό χαρακτηριστικό των ασυρμάτων δικτύων αισθητήρων, η ικανότητα ενός προσομοιωτή να παραδίδει τα μηνύματα στους παραλήπτες του καθορίζει και την ταχύτητα των προσομοιώσεων. Πειράματα πλήθους προσομοιώσεων δικτύων με τα ίδια χαρακτηριστικά με τη χρήση των εργαλείων προσομοίωσης SN2, TOSSIM και Shawn, έδειξαν ότι το Shawn απαιτεί εμφανώς λιγότερο χρόνο σε σχέση με τα άλλα δυο εργαλεία από την κεντρική μονάδα επεξεργασίας(CPU) αλλά και ότι έχει πολύ λιγότερες απαιτήσεις σε μνήμη (RAM). Οι συγκρίσεις φαίνονται στις γραφικές παραστάσης της εικόνας 5.2.



Εικόνα 5.2: Σχήμα Α: Σύγκριση χρόνου επεξεργασίας και χρήσης μνήμης των εργαλείων NS2, TOSSIM και Shawn

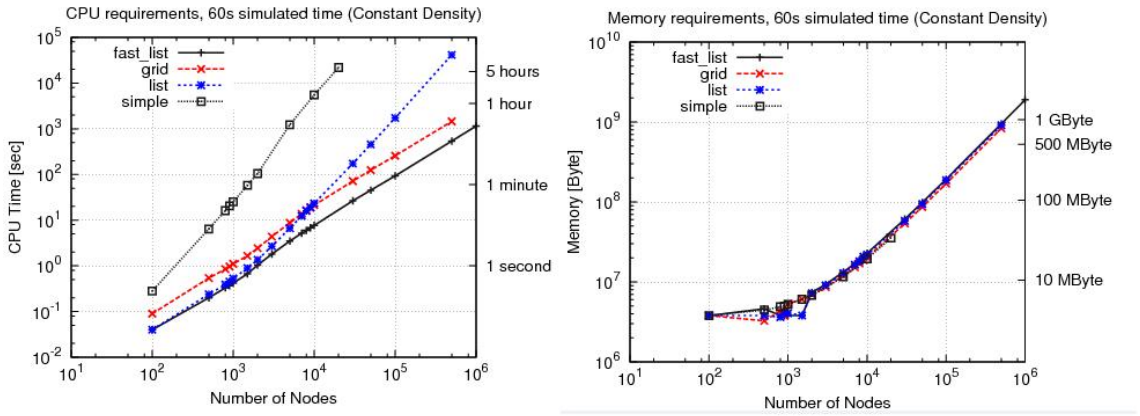
5.2.2 Προσαρμοστικότητα του Shawn

Η συμπεριφορά του Shawn κατά τη διάρκεια της εκτέλεσης των προσομοιώσεων επιρρεάζεται σημαντικά απο την επιλογή του μοντέλου υλοποίησης που θα χρησιμοποιηθεί(Μοντέλο ακμών,Μοντέλο επικοινωνίας ή Μοντέλο μετάδοσης). Ανάλογα με το σενάριο που προσομοιώνεται, μια διαφορετική επιλογή υλοποίησης, μπορεί να αλλάξει ριζικά την απόδοση και την κατανάλωση πόρων(μνήμη, χρόνος επεξεργασίας κλπ) που χρησιμοποιεί η προσομοίωση. Μιας και η επιλογή της κατάλληλης υλοποίησης, το μόνο που χρειάζεται είναι η αλλαγή μιας τιμής στο αρχείο διαμόρφωσης(configuration file), οι χρήστες μπορούν να επιλέξουν την υλοποίηση που ταιριάζει καλύτερα σε κάθε προσομοίωση.



Εικόνα 5.3: Σύγκριση μοντέλων ακμών για αυξανόμενες πυκνότητες δικτύων

Για την παρουσίαση των πλεονεκτημάτων και των μειονεκτημάτων που έχουν οι διαφορετικές υλοποιήσεις των μοντέλων ακμών, χρησιμοποιήθηκαν ακριβώς τα ίδια σενάρια και προσομοιώθηκαν με τη χρήση των διαφορετικών αυτών υλοποιήσεων. Το πρώτο σχήμα της εικόντας 5.3 δείχνει το απαιτούμενο χρόνο στη κεντρική μονάδα επεξεργασίας σε σχέση με το μέγεθος του δικτύου(αριθμός κόμβων) και για τα 4 μοντέλα. Βλέπουμε ότι τα μοντέλα λίστας και ταχείας λίστας είναι γρηγορότερα από το μοντέλο πλέγματος, το οποίο με τη σειρά του είναι πολύ γρηγορότερο από το απλό. Παρόλα αυτά, λαμβάνοντας υπόψιν και την μνήμη που χρησιμοποιεί το καθένα από αυτά(δεύτερο σχήμα εικόνας 5.3), αυτή η ταχύτητα έχει και κάποιο κόστος.



Εικόνα 5.4: Σύγκριση μοντέλων ακμών για αυξανόμενες σταθερή πυκνότητα και αυξανόμενο μέγεθος δικτύων

Το γεγονός ότι το μέγεθος της περιοχής του δικτύου που προσομοιώθηκε ήταν σταθερό, όπως και η ακτίνα εκπομπής, είχε σαν αποτέλεσμα την αυξανόμενη πυκνότητα του γράφου που περιγράφει το δίκτυο καθώς προστίθενται περισσότεροι κόμβοι. Προφανώς, αποτέλεσμα αυτού ήταν η αυξανόμενη ανάγκη για μνήμη που θα αποθηκεύσει τους γείτονες κάθε κόμβου. Το Grid είναι το μοντέλο που προσφέρει μια ισορροπία μεταξύ ταχύτητας και κατανάλωση μνήμης.

Παρόλα αυτά, η παραπάνω παρατήρηση δεν γενικεύεται. Έστω η περίπτωση όπου ο αριθμός των κόμβων που συμμετέχει στο δίκτυο αυξάνεται, αλλά η πυκνότητα του δικτύου παραμένει ίδια. Οι γραφικές παραστάσης της εικόνας 5.4 απεικονίζουν το αποτέλεσμα τέτοιων προσομοιώσεων. Παρατηρούμε ότι η Ταχεία Λίστα απαιτεί μια τάξη μεγέθους λιγότερο χρόνο στην κεντρική μονάδα επεξεργασίας από τα προηγούμενα σχήματα. Από την άλλη, οι απαιτήσεις σε μνήμη δε διαφέρουν και πολύ στις δύο

αυτές περιπτώσεις. Αυτό συμβαίνει, γιατί σε αντίθεση με το προηγούμενο σενάριο, το μέγεθος της "γειτονιάς" κάθε κόμβου παραμένει σταθερό και ο απαιτούμενος χώρος στη μνήμη για την αποθήκευσή της αυξάνεται λίγο και αργά. Όπως και στο προηγούμενο σενάριο, το Απλό μοντέλο είναι το πιο αργό ενώ το μοντέλο Ταχείας λίστας το ποιο γρήγορο. Στο παράδειγμα που απεικονίζεται στα σχήματα 16 και 17 το μοντέλο Πλέγματος είναι πιο αποδοτικό από αυτό της λίστας μόνο για δίκτυα με περισσότερους από $10,000$ κόμβους και αυτό λόγω του κόστους επεξεργασίας που έχει το μοντέλο λίστας στην αρχική οργάνωσή του. Παρόλα αυτά, το μοντέλο Ταχείας Λίστας είναι πιο αποδοτικό για όλα τα μεγέθη δικτύων, αφού μειώνει τον απαιτούμενο χρόνο επεξεργασίας που χρειάζεται για την κατασκευή της λίστας του, χρησιμοποιώντας το μοντέλο πλέγματος για να μειώσει το χρόνο αναζήτησης γειτονικών κόμβων.

5.2.3 Συμπεράσματα

Οι πειραματικές μετρήσεις που έγιναν δείχνουν ότι ο κεντρικός σχεδιαστικός στόχος του Shawn(δηλαδή η προσομοίωση του αποτελέσματος των φαινομένων και όχι τα ίδια τα φαινόμενα) όντως οδηγεί σε υψηλή κλιμακωσιμότητα συγκριτικά με άλλα παραδοσιακά εργαλεία προσομοίωσης. Επίσης, τα αποτελέσματα δείχνουν ότι η συμπεριφορά εκτέλεσης του Shawn διαμορφώνεται από τις αλλαγές στα configuration files. Αυτό δεν ισχύει μόνο για τα μοντέλα ακμών, αλλά και για όλα τα μοντέλα που χρησιμοποιούνται από το Shawn. Οι διαφορετικοί συνδυασμοί μοντέλων επηρεάζουν σημαντικά την απόδοση(ταχύτητα επεξεργασίας) του Shawn αλλά και την ποσότητα κατανάλωσης πόρων(μνήμη). Έτσι, η επιλογή του υλοποιημένου μοντέλου πρέπει να βασίζεται στα χαρακτηριστικά του σεναρίου που πρόκειται να υλοποιηθεί.

Σαν αποτέλεσμα, οι προγραμματιστές πρέπει να επιλέξουν πολύ προσεκτικά το εργαλείο προσομοίωσης ώστε να είναι κατάλληλο ανάλογα με τον τομέα της εφαρμογής που αναπτύσσουν. Εάν ασχολούνται με πολύ λεπτομερείς προσομοιώσεις που πρέπει να λαμβάνουν υπόψιν τους, για παράδειγμα ιδιότητες αναπαραγωγής σημάτων ή θέματα που αφορούν χαμηλά επίπεδα του της αρχιτεκτονικής του δικτύου, το Shawn δεν είναι η καταλληλότερη επιλογή. Από την άλλη πλευρά εάν μια μελέτη αφορά αλγορίθμους και πρωτόκολλα υψηλότερων επιπέδων για ασύρματα δίκτυα αισθητήρων, αυτή η προσοχή λεπτομέρεια στην προσομοίωση όχι απλά δεν είναι απαραίτητη, αλλά συχνά προσθέτει και κόστος στην προσομοίωση και "θολώνει" την εικόνα του πραγματικού

προβλήματος που μελετάται. Σε τέτοιες περιπτώσεις είναι που ο Shawn παρέχει την αναγκαία αφαιρετικότητα και απόδοση.

Ένα από τα μεγαλύτερα πλεονεκτήματα του προσομοιωτή Shawn είναι ότι τα πρωτόκολλα που δημιουργεί κάποιος χρήστης σε αυτόν μπορούν να ενσωματωθούν άμεσα μέσω στο iSense. Το iSense είναι μια πλατφόρμα υλικού και λογισμικού για ασύρματα δίκτυα που απευθύνεται σε ερευνητικές αλλά και σε εμπορικές εφαρμογές.

Κεφάλαιο 6

Υλοποίηση του GHT στο Shawn

Από τα πρωτόκολλα διανομής που μελετήσαμε, τα οποία βασίζονται σε γνωστά DHT πρωτόκολλα, ιδιαίτερο ενδιαφέρον παρουσίασε το GHT. Φαίνεται να τηρεί τα βασικά χαρακτηριστικά που καθιστούν ένα τέτοιο πρωτόκολλο αποδοτικό και παρουσιάζει πειστικές ιδέες για την αντιμετώπιση των βασικών προβλημάτων που δημιουργούνται στα ασύρματα δίκτυα αισθητήρων. Μελετώντας από την άλλη, τα χαρακτηριστικά του Shawn, θεωρήσαμε ότι το εργαλείο αυτό είναι κατάλληλο για να μελετήσουμε τη συμπεριφορά του πρωτοκόλλου και γιαυτό το λόγω προσπαθήσαμε να το ενσωματώσουμε στις βιβλιοθήκες του.

6.1 Μηνύματα(Messages)

Για την υλοποίηση του GHT δημιουργήσαμε τέσσερις τύπους μηνυμάτων.

1. **GhtFindNeighboursMessage**

   Τα μηνύματα αυτά, είναι απλά μηνύματα που περιέχουν τη γεωγραφική θέση κάθε κόμβου και χρησιμοποιούνται για την κατασκευή των τοπικών πινάκων δρομολόγησης κάθε αισθητήρα.

2. **GhtPutMessage**

   Τα μηνύματα αυτά χρησιμοποιούνται για την τοποθέτηση της πληροφορίας που συνέλεξε κάποιος αισθητήρας στον κατάλληλο κόμβο. Αποτελούνται από τα παρακάτω πεδία: τις συντεταγμένες του προορισμού του μηνύματος, το ID του διαδόχου του μηνύματος, τη λειτουργία του μηνύματος(Απληστη Προώθηση

ή Περιμετρική Δρομολόγηση), το κλειδί και την τιμή προς αποθήκευση. Αν το μήνυμα βρίσκεται σε λειτουργία περιμετρικής δρομολόγησης περιέχει και τις συντεταγμένες του σημείου στο οποίο μπήκε σε περιμετρική δρομολόγηση, καθώς και την τελευταία ακμή που προσπέλασε(ζεύγος συντεταγμένων).

3. **GhtGetMessage** Τα μηνύματα αυτά χρησιμεύουν για την αναζήτηση τις τιμής που αντιστοιχεί σε ένα κλειδί. Έχουν ακριβώς την ίδια δομή με τα παραπάνω, μόνο που δεν περιέχουν και την τιμή που αντιστοιχεί στο ζητούμενο κλειδί.

4. **GhtRefreshMessage** Τέλος, τα μηνύματα ανανέωσης τα οποία είναι μηνύματα που παράγονται περιοδικά από τους κόμβους για να εξασφαλίσουν το συνεπή εντοπισμό της πληροφορίας όταν συμβαίνουν αλλαγές στο δίκτυο. Τα μηνύματα αυτά, όσον αφορά τη δομή τους δεν έχουν καμία διαφορά με τα μηνύματα *GhtPutMessage*.

Παρατίθεται ο κώδικας της κλάσης *GHTMessage* που υλοποιεί τη βασική κλάση *Message* του Shawn. Δεν παρατίθενται και οι υλοποιήσεις των μεθόδων γιατί καταλαμβάνουν πολύ χώρο.

```
1 #ifndef _SHAWN_GHT_GHT_MESSAGE_H
2 #define _SHAWN_GHT_GHT_MESSAGE_H
3
4 #include "_legacyapps_enable_cmake.h"
5 #ifdef ENABLE_GHT
6
7 #include <map>
8 #include "sys/vec.h"
9 #include "sys/message.h"
10
11 typedef std::pair<shawn::Vec,shawn::Vec> edge;
12 typedef std::pair<std::string, std::string> label_edge;
13 typedef std::pair<int, int> nodes_edge;
14
15 namespace ght
16 {
17
18   //Message type to create a neighbour list-nodes one hop away- for
       each node
```

```
19  class GhtFindNeighboursMessage
20        : public shawn::Message
21   {
22   public:
23     GhtFindNeighboursMessage(shawn::Vec& coordinates);
24     virtual ~GhtFindNeighboursMessage();
25
26   };
27
28
29   //Message type to containing the key value pair. Message is routed
         through the network to its right position
30   class GhtPutMessage
31        : public shawn::Message
32   {
33    public:
34      GhtPutMessage(shawn::Vec& destination, int succ, int hops, std::
           string& key,std::string& value);
35      GhtPutMessage(shawn::Vec& destination, int succ, shawn::Vec&
           start, nodes_edge first_edge, edge curr_edge, int hops,
                                 std::string& key,std::string& value);
36      virtual ~GhtPutMessage();
37           /*Getters */
38           inline shawn::Vec destination (void) const throw()
39           {return destination_;};
40           inline int successor (void) const throw()
41           {return successor_;};
42           inline shawn::Vec perimeter_start_point (void) const throw()
43           {return perimeter_start_point_;};
44           inline bool right_hand_rule(void) const throw()
45           {return right_hand_rule_;};
46           inline nodes_edge first_edge(void) const throw()
47           {return first_edge_;};
48           inline int hops(void) const throw()
49           {return hops_;};
50           inline edge current_edge(void) const throw()
51           {return curr_edge_;};
52           inline std::string key(void) const throw()
53           {return key_;};
```

```
54            inline std::string value(void) const throw()
55            {return value_;};
56
57     private:
58        shawn::Vec destination_;
59        int successor_;
60        shawn::Vec perimeter_start_point_;
61        edge curr_edge_;
62        nodes_edge first_edge_;
63        bool right_hand_rule_;
64        int hops_;
65        std::string key_;
66        std::string value_;
67    };
68
69    //Message type to containing the key value pair. Message is routed
          through the network to its current right position
70    //In case of topology changes, the new position may be different
          than the previous.
71    class GhtRefreshMessage
72        : public shawn::Message
73    {
74    public:
75      GhtRefreshMessage(shawn::Vec& destination, int succ, int hops, std
            ::string& key,std::string& value);
76            GhtRefreshMessage(shawn::Vec& destination, int succ, shawn::
                Vec& start, nodes_edge first_edge, edge curr_edge, int
                                                  hops, std::string& key,std
                ::string& value);
77      virtual ~GhtRefreshMessage();
78            /*Getters */
79            inline shawn::Vec destination (void) const throw()
80            {return destination_;};
81            inline int successor (void) const throw()
82            {return successor_;};
83            inline shawn::Vec perimeter_start_point (void) const throw()
84            {return perimeter_start_point_;};
85            inline bool right_hand_rule(void) const throw()
86            {return right_hand_rule_;};
```

```
87            inline nodes_edge first_edge(void) const throw()
88            {return first_edge_;};
89            inline int hops(void) const throw()
90            {return hops_;};
91            inline edge current_edge(void) const throw()
92            {return curr_edge_;};
93            inline std::string key(void) const throw()
94            {return key_;};
95            inline std::string value(void) const throw()
96            {return value_;};
97
98      private:
99         shawn::Vec destination_;
100        int successor_;
101        shawn::Vec perimeter_start_point_;
102        edge curr_edge_;
103        nodes_edge first_edge_;
104        bool right_hand_rule_;
105        int hops_;
106        std::string key_;
107        std::string value_;
108     };
109
110
111     //Message type to query for the value of a certain key. Message is
           routed through the network to its right position
112     class GhtGetMessage
113         : public shawn::Message
114     {
115     public:
116       GhtGetMessage(shawn::Vec& destination, int succ, int hops, std::
              string& key);
117            GhtGet1Message(shawn::Vec& destination, int succ, shawn::Vec
                  & start, nodes_edge first_edge, edge curr_edge, int
                                                hops, std::string& key);
118       virtual ~GhtGetMessage();
119            /*Getters */
120            inline shawn::Vec destination (void) const throw()
121            {return destination_;};
```

```
122             inline int successor (void) const throw()
123             {return successor_;};
124             inline shawn::Vec perimeter_start_point (void) const throw()
125             {return perimeter_start_point_;};
126             inline bool right_hand_rule(void) const throw()
127             {return right_hand_rule_;};
128             inline nodes_edge first_edge(void) const throw()
129             {return first_edge_;};
130             inline int hops(void) const throw()
131             {return hops_;};
132             inline edge current_edge(void) const throw()
133             {return curr_edge_;};
134             inline std::string key(void) const throw()
135             {return key_;};
136
137     private:
138         shawn::Vec destination_;
139         int successor_;
140         shawn::Vec perimeter_start_point_;
141         edge curr_edge_;
142         nodes_edge first_edge_;
143         bool right_hand_rule_;
144         int hops_;
145         std::string key_;
146     };
147
148 }
149
150 #endif
151 #endif
```

6.2 Επεξεργαστής(Processor)

Η κλάση *GhtProcessor* που υλοποιήσαμε και είναι υλοποιεί την κλάση *Processor* του Shawn, διαθέτει τις εξής βασικές μεθόδους:

- **boot(void)**

  Η συνάρτηση αυτή υλοποιεί τη μέθοδο *boot* της κλάσης *Processor* του Shawn.

Καθορίζει τις ενέργειες κάθε αισθητήρα μόλις εισέλθει στο δίκτυο. Στη συνάρτηση αυτή όλοι οι αισθητήρες στέλνουν τις συντεταγμένες τους για να αρχικοποιηθούν οι πίνακες δρομολόγησης όλων των κόμβων.

- **process_message(const shawn::ConstMessageHandle&)**

  Η συνάρτηση αυτή υλοποιεί τη μέθοδο *process_message* της κλάσης *Processor* του Shawn. Είναι η μέθοδος που αποφασίζει αν ο κόμβος θα παραλάβει το μήνυμα που άκουσε ή αν θα το απορρίψει. Επίσης αντιλαμβάνεται τον τύπο του μηνύματος και ανάλογα με αυτόν κάνει τις κατάλληλες ενέργειες.

- **work(void)**

  Η συνάρτηση αυτή υλοποιεί τη μέθοδο *work* της κλάσης *Processor* του Shawn. Η μέθοδος αυτή καλείται σε κάθε γύρο προσομοίωσης και είναι αυτή που κρατά ουσιαστικά τον έλεγχο της προσομοίωσης. Η μέθοδος στέλνει τυχαία αναζητήσεις στο δίκτυο για την ανάκτηση τιμών που αντιστοιχούν σε κάποια κλειδιά. Επίσης είναι υπεύθυνη για να καλέσει την update_content και την refresh, μιας και μέσω αυτής μόνο μπορεί να εφαρμοστεί η περιοδικότητα.

- **neighbours(void)**

  Η συνάρτηση αυτή επιστρέφει όταν κληθεί το τρέχον σύνολο των γειτόνων κάθε κόμβου.

- **replicas (void)**

  Η συνάρτηση αυτή επιστρέφει όταν κληθεί τον τρέχοντα πίνακα κατακερματισμού με τα αντίγραφα ζεύγη κλειδιών-τιμών που έχει ο κάθε κόμβος στη μνήμη του.

- **content (void)**

  Η συνάρτηση αυτή επιστρέφει όταν κληθεί τον τρέχοντα πίνακα κατακερματισμού με τα ζεύγη κλειδιών-τιμών για τα οποία κάθε κόμβος είναι κόμβος εστία.

- **handle_neighbour_message (const GhtFindNeighboursMessage& ght)**

  Η συνάρτηση αυτή καλείται από την process_message, όταν το μήνυμα που παρέλαβε είναι FindNeighbourMessage. Ελέγχει αν ο πίνακας δρομολόγησης περιέχει αυτόν τον γείτονα που έστειλε το μήνυμα. Εάν ήδη υπάρχει απλά ανανεώνει το χρονιστή λήξης του, διαφορετικά τον προσθέτει στον πίνακά του.

- **handle_get_message (const GhtGetMessage& ght)**

  Η συνάρτηση αυτή καλείται από την process_message, όταν το μήνυμα που παρέλαβε είναι GhtGetMessage. Ελέγχει τον τρόπο λειτουργίας του μηνύματος που παρέλαβε και το αντιμετωπίζει ανάλογα, όπως προδιαγράφει το πρωτόκολλο GHT. Είτε το προωθεί στον κατάλληλο διάδοχο, είτε αποφασίζει ότι ο τρέχων κόμβος έχει την ζητούμενη πληροφορία.

- **handle_put_message (const GhtPutMessage& ght)**

  Η συνάρτηση αυτή καλείται από την process_message, όταν το μήνυμα που παρέλαβε είναι GhtPutMessage. Ελέγχει τον τρόπο λειτουργίας του μηνύματος που παρέλαβε και το αντιμετωπίζει ανάλογα, όπως προδιαγράφει το πρωτόκολλο GHT. Είτε το προωθεί στον κατάλληλο διάδοχο, είτε αποφασίζει ότι ο τρέχων κόμβος είναι κόμβος εστία του ζεύγους κλειδιού τιμή που περιέχει το μήνυμα και τοποθετεί το ζεύγος στον πίνακα κατακερματισμού του.

- **handle_refresh_message (const GhtRefreshMessage& ght)**

  Η συνάρτηση αυτή καλείται από την process_message, όταν το μήνυμα που παρέλαβε είναι GhtRefreshMessage. Ελέγχει τον τρόπο λειτουργίας του μηνύματος που παρέλαβε και το αντιμετωπίζει ανάλογα, όπως προδιαγράφει το πρωτόκολλο GHT. Είτε το προωθεί στον κατάλληλο διάδοχο, είτε αποφασίζει ότι ο τρέχων κόμβος πρέπει να το καταναλώσει. Πάλι ανάλογα με την περίπτωση είτε παράγει νέο GhtRefreshMessage για το συγκεκριμένο ζεύγος κλειδί-τιμή, είτε αποφασίζει ότι εκείνος είναι ο κόμβος εστία του ζεύγους.

- **perimeter_successor(edge prev,shawn::Vec& start,shawn::Vec& destination,int pre)**

  Η συνάρτηση αυτή ελέγχει τις κατάλληλες γωνίες που σχηματίζει ο τρέχων κόμβος με όλους του τους γείτονες και αποφασίζει ποιος έχει τη μικρότερη για να του προωθήσει το μήνυμα σε λειτουργία περιμετρικής δρομολόγησης.

- **euclidean_distance(const shawn::Vec& v1, const shawn::Vec& v2)**

  Η συνάρτηση αυτή υπολογίζει και επιστρέφει την ευκλείδεια απόσταση μεταξύ δύο σημείων.

- **Put(std::string key, std::string value)**

  Η συνάρτηση αυτή όποτε καλείται αναλαμβάνει να παράγει ένα GhtPutMessage για ένα ζεύγος κλειδί-τιμή και να εκκινήσει τη διαδικασία μετάδοσής του στο δίκτυο.

- **Refresh(std::string key, std::string value)**

  Η συνάρτηση αυτή όποτε καλείται αναλαμβάνει να παράγει ένα GhtRefreshMessage για ένα ζεύγος κλειδί-τιμή και να εκκινήσει τη διαδικασία μετάδοσής του στο δίκτυο.

- **home_node(std::string& key,std::string& value)**

  Η συνάρτηση αυτή είναι υπεύθυνη να προσθέσει το ζεύγος κλειδί τιμή στον πίνακα κατακερματισμού δεδομένων του κόμβου. Καλείται όποτε ένας κόμβος αποφασίσει ότι είναι ο κόμβος-εστία για αυτό το ζεύγος και κάνει και τους κατάλληλους ελέγχους, ώστε να μην υπάρχουν διπλότυπα ή αν χρειάζεται να διαγραφεί το ίδιο ζεύγος από τον πίνακα κατακερματισμού αντιγράφων.

- **handle_refreshed_data(std::string& key,std::string& value)**

  Η συνάρτηση αυτή είναι υπεύθυνη να προσθέσει το ζέυγος κλειδί τιμή στον πίνακα κατακερματισμού αντιγράφων του κόμβου. Καλείται όποτε ένας κόμβος λάβει μήνυμα ανανέωσης για αυτό το ζεύγος και κάνει και τους κατάλληλους ελέγχους, ώστε να μην υπάρχουν διπλότυπα στον πίνακα κατακερματισμού αντιγράφων.

- **update_content(void)**

  Η συνάρτηση αυτή καλείται περιοδικά μέσα από τη work και ελέγχει όλους τους χρονιστές για να μπει στη διαδικασία, να παράγει μηνύματα ανανέωσης και να διαγράψει ληγμένα ζεύγη από τους πίνακες κατακερματισμού.

- **find_closest(const shawn::Vec& dest)**

  Η συνάρτηση αυτή ελέγχει τις αποστάσεις όλων των γειτόνων από τον προορισμό του μηνύματος και επιστρέφει τον κοντινότερο στον προορισμό.

- **angle (shawn::Vec& v1, shawn::Vec& v2)**

  Η συνάρτηση αυτή υπολογίζει την αριστερόστροφη γωνία που του ευθύγραμμου

τμήματος που σχηματίζει ο τρέχων κόμβος με το σημείο v_1, και του ευθύγραμμου τμήματος σχηματίζει ο τρέχων κόμβος με το σημείο v_2.

- **print_state(void)**

  Αυτή η συνάρτηση όποτε καλείται εκτυπώνει τα περιεχόμενα των πινάκων κατακερματισμού δεδομένων και αντιγράφων.

- **cross(edge e1, edge e2)**

  Τέλος ,η συνάρτηση αυτή ελέγχει κατά πόσον οι ακμές e_1 και e_2 τέμνονται. Αυτό χρειάζεται στην περίπτωση αναζήτησης του διαδόχου σε περιμετρική δρομολόγηση.

Παρατίθεται ο κώδικας της κλάσης *GHTMessage* που υλοποιεί τη βασική κλάση *Message* του Shawn. Δεν παρατίθενται και οι υλοποιήσεις των μεθόδων γιατί καταλαμβάνουν πολύ χώρο.

```
1 #ifndef _SHAWN_LEGACYAPPS_GHT_GHT_PROCESSOR_H
2 #define _SHAWN_LEGACYAPPS_GHT_GHT_PPROCESSOR_H
3 #include "_legacyapps_enable_cmake.h"
4 #ifdef ENABLE_GHT
5
6
7 #include "sys/processor.h"
8 #include "ght_message.h"
9 #include "sys/vec.h"
10 #include <set>
11 #include<vector>
12
13 #define REFRESH_INTERVAL 5;
14 #define TAKEOVER_INTERVAL 10;
15 #define EXPIRATION_INTERVAL 15;
16
17 typedef std::vector<int> timers;
18 typedef std::pair<std::string,timers> value;
19 typedef std::pair<std::string,value> key_value_pair;
20 typedef std::map<std::string, value> key_value_pairs;
21
22 namespace ght
23 {
```

```
24  class GhtProcessor
25      : public shawn::Processor
26        //public shawn::EventScheduler::EventHandler
27  {
28  public:
29      GhtProcessor();
30      virtual ~GhtProcessor();
31
32      virtual void boot( void ) throw();
33      virtual bool process_message( const shawn::ConstMessageHandle& )
            throw();
34      virtual void work( void ) throw();
35      inline std::set<const shawn::Node*> neighbours( void ) const
            throw()
36      {return neighbour_list;};
37      inline key_value_pairs replicas ( void ) const throw()
38      {return replicas_;};
39      inline key_value_pairs content ( void ) const throw()
40      {return content_;};
41
42  private:
43      //Message Handling, handles and forwards request message
44      void handle_get1_message ( const GhtGetMessage& ght) throw();
45      //Message Handling, finds the closest neighbour to send the
            message
46      void handle_put_message ( const GhtPutMessage& ght) throw();
47      //Message Handling,beacons to detect neighbours and construct
            routing table
48      void handle_neighbour_message ( const GhtFindNeighboursMessage&
            ght) throw();
49      // Message Handling, finds the closest neighbour to send the
            message
50      void handle_refresh_message ( const GhtRefreshMessage& ght) throw
            ();
51      //Forwards messages in perimeter mode
52      const shawn::Node* perimeter_successor(edge prev,shawn::Vec&
            start,shawn::Vec& destination,int pre) throw();
53      // Calculates the distance between node v1 and point v2 of the
            network
```

```
54      double eucledean_distance(const shawn::Node& v1, const shawn::Vec
            & v2) throw ();
55      // Calculates the distance between node v1 and point v2 of the
            network
56      double euclidean_distance(const shawn::Vec& v1, const shawn::Vec&
             v2) throw ();
57      //Constructs a putMessage for the sensed event and sends it out
            to the network
58      void Put(std::string key, std::string value) throw();
59      //Constructs a refresh paceket for the expired key value pair and
             sends it out into the network
60      void Refresh(std::string key, std::string value) throw();
61      //Handles replica data when it arrives
62      void home_node(std::string& key,std::string& value) throw();
63      //Handles replica data when it arrives
64      void handle_refreshed_data(std::string& key,std::string& value)
            throw();
65      //checks timers and handles data refreshes, expirations etc
66      void update_content( void ) throw();
67      //Calculates the closest neighbour to the destination (greedy
            mode)
68      const shawn::Node* find_closest(const shawn::Vec& dest) throw();
69      void store_random_data( void ) throw();
70      //calculates the angle between two edges(vectors)
71      double angle (shawn::Vec& v1, shawn::Vec& v2) throw();
72      //Prints out the state (content) the node holds
73      void print_state( void ) throw();
74      //Returns true if the given edges cross
75      bool cross(edge e1, edge e2) throw();
76
77      //Last time the node received a message
78      int last_time_of_receive_;
79      //a list with all neighbours(one hop away) of the node
80      std::set<const shawn::Node*> neighbour_list;
81      //hash table with all (key, value) pairs that for which it is
            home node
82      key_value_pairs content_;
83      //hash table with all (key, value) pairs of which the node holds
            replicas
```

```
84          key_value_pairs replicas_;
85          //Total number of bytes received
86          int bytes_received_;
87          //Total number of bytes sent
88          int bytes_sent_;
89      };
90
91  }
92
93  #endif
94  #endif
```

Κεφάλαιο 7

Μελλοντικές Κατευθύνσεις - Συμπεράσματα

Η έρευνα στον τομέα των ασυρμάτων δικτύων αισθητήρων, έχει εξελιχθεί ραγδαία τα τελευταία χρόνια. Ανάπτυξη νέων τεχνολογιών στον τομέα υλικού δημιουργεί συνεχώς νέα δεδομένα για την σχεδίαση και την υλοποίηση, αποδοτικότερων πρωτοκόλλων. Το πρόβλημα της αποδοτικής διανομής πληροφορίας σε αυτά τα δίκτυα είναι από τα πιο διαδεδομένα και συνεχώς αναπτύσσονται νέες ιδέες για την επίλυσή του. Το GHT είναι μια από αυτές, και θα μπορούσε να συνδυαστεί με πληθώρα άλλων μελετών και πιθανόν να οδηγήσει σε νέες.

Το GHT εξετάζει το πρόβλημα της αποδοτικής διανομής πληροφορίας σε ασύρματα δίκτυα αισθητήρων από μια πιο γενική σκοπιά. Δεν στοχεύει στη βελτιστοποίηση συγκεκριμένων μετρικών. Συνεπώς, θα μπορούσε να γίνει μια προσπάθεια προσαρμογής του GHT ώστε να επικεντρωθεί στη βελτιστοποίηση κάποιου χαρακτηριστικού ή να μελετηθεί ως προς τη συμπεριφορά του όταν εφαρμόζεται σε δίκτυα αισθητήρων συγκεκριμένων χαρακτηριστικών (settings). Μπορεί για παράδειγμα να γίνει η δοκιμή κάποιου διαφορετικού πρωτοκόλλου δρομολόγησης από το GPSR[23]. Μπορεί το GPSR να είναι πολύ αποδοτικό και ευρύτατα διαδεδομένο πρωτόκολλο, όμως υπάρχουν πολλοί εναλλακτικοί αλγόριθμοι δρομολόγησης . Μερικοί από αυτούς είναι οι: [29], [30], [31], [32], [33]. Πιθανότατα, η ενσωμάτωσή τους στο GHT να έχει ως αποτέλεσμα την επίτευξη καλύτερης απόδοσης σε θέματα κατανάλωσης ενέργειας ή ελαχιστοποίησης συμφόρησης. Για παράδειγμα το [34], παρουσιάζει μια μελέτη για την ομοιόμορφη κατανομή της μετάδοσης δεδομένων με τη χρήση επικαμπύλιας δρο-

μολόγησης(cuverball routing). Η χρήση αυτού με τις κατάλληλες προσαρμογές για τη δρομολόγηση μηνυμάτων στο GHT μπορεί να οδηγήσει στη διανομή και ανάκτηση πληροφορίας πετυχαίνοντας και μεγαλύτερη διάρκεια ζωής του δικτύου.

Το GHT διανέμει τα δεδομένα στο δίκτυο με τη χρήση της γεωγραφικής θέσης των κόμβων. Θα μπορούσαν να γίνουν προσπάθειες συνδυασμού του πρωτοκόλλου αυτού με αλγορίθμους εντοπισμού*(localization algorithms)*. Μια τέτοια προσέγγιση μπορεί να μειώσει σημαντικά την ενεργειακή σπατάλη των κόμβων από τα GPS, τα οποία θα προσδιόριζαν τη γεωγραφική του θέση. Πέραν της κατανάλωσης ενέργειας τίθεται και το θέμα κατά πόσον είναι εφικτό για κόμβους πολύ μικρού μεγέθους να διαθέτουν και GPS. Για παράδειγμα, η εργασία *Γεωγραφική Δρομολόγηση χωρίς τη χρήση πληροφορία τοποθεσίας(Geographic Routing without location Information)*[28] παρουσιάζει μια πολύ ενδιαφέρουσα προσέγγιση για τον εντοπισμό τη σχετικής θέσης ενός κόμβου σε ένα δίκτυο. Η περιοδική εφαρμογή ενός τέτοιου αλγορίθμου(για την περίπτωση κινούμενων κόμβων) μπορεί να συνδυαστεί με το GHT και να αποτελέσει μια ολοκληρωμένη δουλεία λαμβάνοντας υπόψιν και την ενέργεια που καταναλώνεται για να εντοπίσει κάθε κόμβος τη θέση του.

Πέραν όμως του GHT, και άλλα πρωτόκολλα όπως το CSN που περιγράψαμε παραθέτουν κάποιες ενδιαφέρουσες ιδέες. Για παράδειγμα η συσταδοποίηση με τις κεφαλές των συστάδων να εναλλάσσονται με βάση την ενεργειακή τους κατανάλωση είναι μια πρόταση που μπορεί να χρησιμοποιηθεί και σε άλλες προσεγγίσεις. Το MeshChord[35], ένα πρωτόκολλο διανομής πληροφορίας για γενικά ασύρματα δίκτυα με εξαιρετικά ταχεία απόκριση σε αναζητήσεις(Δεν έχει εφαρμοστεί πρακτικά, αλλά τα πειραματικά αποτελέσματα δείχνουν ότι ξεπερνά και αυτή του Chord[7]), μπορεί ίσως με κάποιες αλλαγές ή προσθήκες να χρησιμοποιηθεί και σε ασύρματα δίκτυα αισθητήρων.

Η λίστα των αλγορίθμων και των πρωτοκόλλων που σχετίζονται με αυτό τον τομέα είναι τεράστια. Παρόλα αυτά δεν έχει βρεθεί ακόμα κάποια επαναστατική μέθοδος η οποία θα συνδυάζει και θα προσφέρει αποδοτική λύση στο συνδυασμό των προβλημάτων που εμφανίζουν τα ασύρματα δίκτυα αισθητήρων. Για αυτό το λόγο ο τομέας αυτός εξακολουθεί να αποτελεί πρόκληση για το σύγχρονο ερευνητικό κόσμο.

Βιβλιογραφία

[1] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", in HICSS 2000.

[2] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "An architecture-specific protocol for Wireless Microsensor Networks ", in IEEE Transactions of Wireless Communications, vol 1, no. 4, pp.660-670, October 2002.

[3] M. J. Handy, M. Haase and D. Timmermann, "Low Energy Adaptive Clustering Hierarchy with deterministic cluster-head selection", in Proceedings of 4th International Workshop on Mobile and Wireless Communications Networks, 2002.

[4] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion:A Scalable and Robust Communication Paradigm for Sensor Networks", in MOBICOM 2000.

[5] I. Chatzigiannakis, T. Dimitriou, S. Nikoletseas, and P. Spirakis, "A Probabilistic Algorithm for Efficient and Robust Data Propagation in Smart Dust Networks", in the Proceedings of the 5th European Wireless Conference on Mobile and Wireless Systems beyond 3G (EW 2004), 2004. Also, in the Ad-Hoc Networks Journal, Elsevier, 2006.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable content addressable Network.", in Proceedings of the ACM/SIGCOMM, pp. 654-663, 2001.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications.", IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 17-32, 2003.

[8] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph and J. D. Kubiatowics, "Tapestry: A resilient global-scale overlay for service deployment.", IEEE Journal on Selected Areas in Communications, vol. 22, no 1, pp. 41-53, January 2004.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object localization and rouing for large-scale peer-to-peer systems.", in Proceedings of the Middleware, 2001.

[10] C.Plaxton, R.Rajaraman and A. Richa, "Accessing nearby copies of replicated objects in a distributed environment.", in Proceedings of the 9th annual Symposium on Parallel Algorithms and Architectures, 1997.

[11] "Secure hash standard", NIST, U.S. Dept of Commerce, National Technical Information Service FIPS 180-1, April 1995.

[12] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris and I. Stoica, "Wide area cooperative storge with csf.", in Proceedings of the 18th ACM symposium on Operating systems principles, pp. 202-215, 2001.

[13] R. Cox, A. Muthitacharoen and R. Morris, "Serving dns using Chord" in Proceedings of the 1st International Workshop on Peer-to-Peer Systems, March 2002.

[14] J. Kubiatoxicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Wiener, C. Wells and B. Zhao, "Oceanstore: An architecture for global scale persistent storage."' in Proceedings of the ACM ASPLOS, November 2002.

[15] A. Rowstron, A-M. Kermarrec, M. Castro and P. Druschel , "SCRIBE: The design of a large-scale event notification infrastructure ." in Proceedings of the 3rd International Workshop on Network Group Communications(NGC2001), pp. 30-43,UCL London, UK, November 2001.

[16] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", in Proceedings of th 18th

ACM Symposium on Operating systems principles (SOSP'01), pp. 188-201 Lake Louise, Alberta, Canada, October 2001.

[17] P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility", HotOS VIII, Schoss Elmau, Germany,ŕ May 2001.

[18] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment", SOSP'03,Lake Bolton, New York, October, 2003.ŕ

[19] S. Iyer, A. Rowstron and P. Drushel, "Suirrel: A decentralized peer-to-peer web cache.", in Proceedings of the 21st Symposium on Principles of Distributed Computing(PODC), Monterey, California, USA, July 2002.

[20] L. P. Cox, C. D. Murray and B.D Noble, "Patiche: Making backup cheap and easy.", SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pp. 258-298, 2002.

[21] Muneeb Ali and Zarash Afzal Uzmi, "CSN: A Network Protocol for Servicing Dynamic Queries in Large-Scale Wireless Sensor Networks", CNSR 2004.

[22] S. Ratnasamy, B. Karp, Li Yin, Fang Yu,D. Estrin,R. Govindan and S. Shenker"GHT: A Geographic Hash Table for DataCentric Storage", 1st ACM International Workshop on Wireless Sensor Networks and Applications(WSNA), 2002.

[23] B.Karp, H.T.Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Network", ACM Mobicom, 2000.

[24] Abd Al-Basset AL-Mamou and Houda Labiod, "ScatterPastry: An Overlay Routing Using a DHT over Wireless Sensor Networks", International Conference on Intelligent Pervasive Computing(PERCOM), 2007.

[25] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers", SIGCOMM, 1994.

[26] http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/

[27] http://shawnwiki.coalesenses.com/

[28] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker and I. Stoica, "Geographic Routing without Location Information", in Proceedings of the 9^{th} annual international conference on Mobile computing and networking(MOBICOM), 2003.

[29] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (DREAM)", in Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom ő98, (Dallas, Texas), August 1998.

[30] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks, In Wireless Networks", Vol. 7, pp. 609ů 616, 2001.

[31] J. Gao, L. J. Guibas, J. Hershburger, L. Zhang, A. Zhu,"Geometric Spanner for Routing in Mobile Networks ", In Proceedings of the 2nd ACM, In Proceedings of the 2^{nd} ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), pp. 45ů55, October 2001.

[32] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing", ACM Mobicom 2000.

[33] W. Liao and J. Sheu and Y. Tseng, "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks", In Telecommunication Systems, Volume 18, pages 37ů60, 2001.

[34] L. Popa, A. Rostamizadeh, R. M. Karp, C. H. Papadimitriou and I. Stoica, "Balancing traffic load in wireless networks with curveball routing." MobiHoc, 2007.

[35] S. Burresi, C. Canali, M. E. Renda, P. Santi, "MESHCHORD: A Location-Aware, Cross-Layer Specialization of Chord for Wireless Mesh Networks", PerCom, 2008.