Επεξεργασία και Διαχείριση Δεδομένων σε Δίκτυα Αισθητήρων

1η εργαστηριακή αναφορά (TAG)

Κωνσταντίνος Πεσλής ΑΜ:2014030074
Σοφοκλής Φιλάρετος Γαβριηλίδης ΑΜ:2014030062

Βοηθητικό Πρόγραμμα:
Η υλοποίηση του βοηθητικού προγράμματος, για την δημιουργία του topology.txt, έγινε στο αρχείο myTopology.py σε γλώσσα python .

Πρόγραμμα 1:
Α) Τα κομμάτια του κώδικα που αφαιρέσαμε εμείς ήταν τα εξής:

- Όλα τα μηνύματα που εμφανιζόταν με την εντολή printf.
- Όλα τα interfaces που χρησιμοποιήθηκαν για serial port.
- Όλα τα interfaces που χρησιμοποιήθηκαν για τα leds.
- Όλο τον κωδικά ο οποίος βρισκόταν στη receiveRoutingTask(), όπου έβρισκε "καλύτερο" πατέρα στον εκάστοτε κόμβο.
- Και, πάλι στη receiveRoutingTask() , τον κώδικα που ειδοποιούσε κάθε κόμβος τον πατέρα του ότι έλαβε κάποιο μήνυμα.

Κώδικα που παραποιήσαμε:

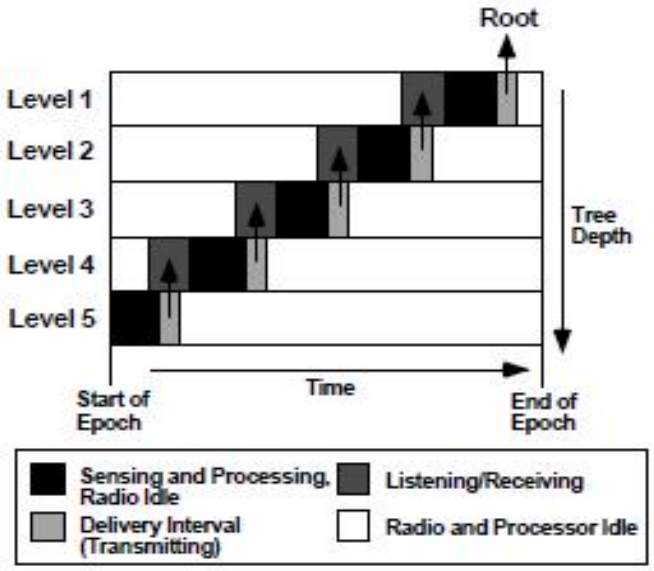
- Στη RoutingMsgTimer.fired() αν ο κόμβος ήταν ο TOS_NODE_ID==0 καλούνταν ξανά η RoutingMsgTimer.fired(). Εμείς βάλαμε να καλείται η δικία μας συνάρτηση, NotifyParentMsgTask, στην οποία ουσιαστικά εκτυπώνουμε τα AVERAGE και VARIANCE , ή ετοιμάζουμε την πληροφορία για να την στείλουμε στον πατέρα.

Β)
Αρχικά να αναφέρουμε ότι δημιουργήσαμε έναν πίνακα στον οποίο κρατάμε την συνολική πληροφορία όλων των παιδιών του εκάστοτε κόμβου.Δηλαδή το sum όλων των παιδιών (για το average) , το squares(για το variance),
το children(για το πόσα παιδιά έχει αυτός ο κόμβος) , και το senderID με το οποίο ουσιαστικά ξεχωρίζουμε τα ID των παιδιών του κόμβου.

receiveRoutingTask(): Σε αυτή την συνάρτηση ουσιαστικά οι κόμβοι ακούνε broadcast μηνύματα , και το 1ο μήνυμα που λάβει θα το δεχτεί ώς τον πατέρα του.Δημιουργείται ουσιαστικά το δέντρο με την σχέση πατέρα-παιδιού που θέλουμε.

Επίσης εδώ υλοποιήσαμε το κομβικό σημείο στο οποίο ξυπνά κάθε κόμβος για να επεξεργαστεί τα μηνύματα που θα λάβει από τα παιδιά του και να στείλει στον πατέρα του.
Η σχέση με την οποία ξυπνούσαμε κάθε κόμβο είναι η:
routingTime + (maxNodes - curdepth)*TIMER_FAST_PERIOD - (listeningNotifyMsgTime*(curdepth+1)) + rand()60%.
Ο χρόνος αυτός είναι για την χειρότερη πιθανή κατάσταση όπου το δέντρο είναι εκφυλισμένο,δηλαδή, κάθε επίπεδο αποτελείται και από έναν κόμβο.
Οπότε, όταν το τελειώσει το routing, κάθε κόμβος θα πρέπει να ξυπνήσει όταν:

1. Όλοι οι από κάτω κόμβοι έχουν επεξεργαστεί τα δεδομένα τους **(maxNodes - curdepth)*TIMER_FAST_PERIOD**
2. Για να μην χάσει το μήνυμα που του στέλνει το παιδί του από το προηγούμενο επίπεδο, **-(listeningNotifyMsgTime*(curdepth+1))**
3. Συν μια τυχαιότητα για να μην ξυπνήσουν όλοι οι κόμβοι του ίδιου επιπέδου την ίδια χρονική στιγμή και υπάρξει σύγκρουση δεδομένων.(Αν και στο παράδειγμα που αναφέραμε δεν χρειάζεται αφού κάθε επίπεδο αποτελείται από έναν κόμβο.)



<u>NotifyParentMsgTask():</u>
Αυτή είναι η κύρια συνάρτηση μας στην οποία εμφανίζαμε τα επιθυμητά αποτελέσματα και επεξεργαζόμασταν τα δεδομένα μετά το Routing.

- Πιο συγκεκριμένα, αν ο κόμβος ήταν ο TOS_NODE_ID=0 εμφανίζαμε το ROUND,AVERAGE,VARIANCE.

AVERAGE: Κάθε κόμβος κρατάει το sum(value),squares(value*value) και τον αριθμό παιδιών που έχουν συγκεντρώσει τα παιδιά του.
Δηλαδή κάθε κόμβος αποθηκεύει την πληροφορία που υπάρχει για όλα τα επίπεδα που υπάρχουν πιο κάτω απο αυτόν.
Με αυτόν τον τρόπο όταν φτάσουμε στον TOS_NODE_ID=0 θα ξέρουμε τα παιδιά του τα οποία θα περιέχουν ουσιαστικά όλη την πληροφορία από τα κάτω επίπεδα.Επομένως , το average θα ισούται με το sum/children, όπου, sum το άθροισμα όλων των sum που έχουν συγκεντρώσει τα παιδιά του, και children τα συνολικά παιδιά(κόμβοι) που υπαρχουν.
VARIANCE: Από την στιγμή που το variance = $E[x^2] - E^2[x]$, ακολουθούμε ακριβώς την ίδια λογική με το average, απλώς κρατάμε το τετράγωνο των αθροισμάτων των τιμών για να το αφαιρέσουμε από το τετράγωνο της μέσης τιμής.
Τέλος μηδενίζουμε τις τιμές του κόμβου ώστε σε επόμενη εποχή να πάρει άλλες τιμές.

- Αν τώρα είμαστε στην περίπτωση όπου είναι οποιοσδήποτε άλλος κόμβος εκτός του 0, χρειάζεται απλά να ετοιμάσουμε ένα μήνυμα για να στείλουμε στον πατέρα του κόμβου.

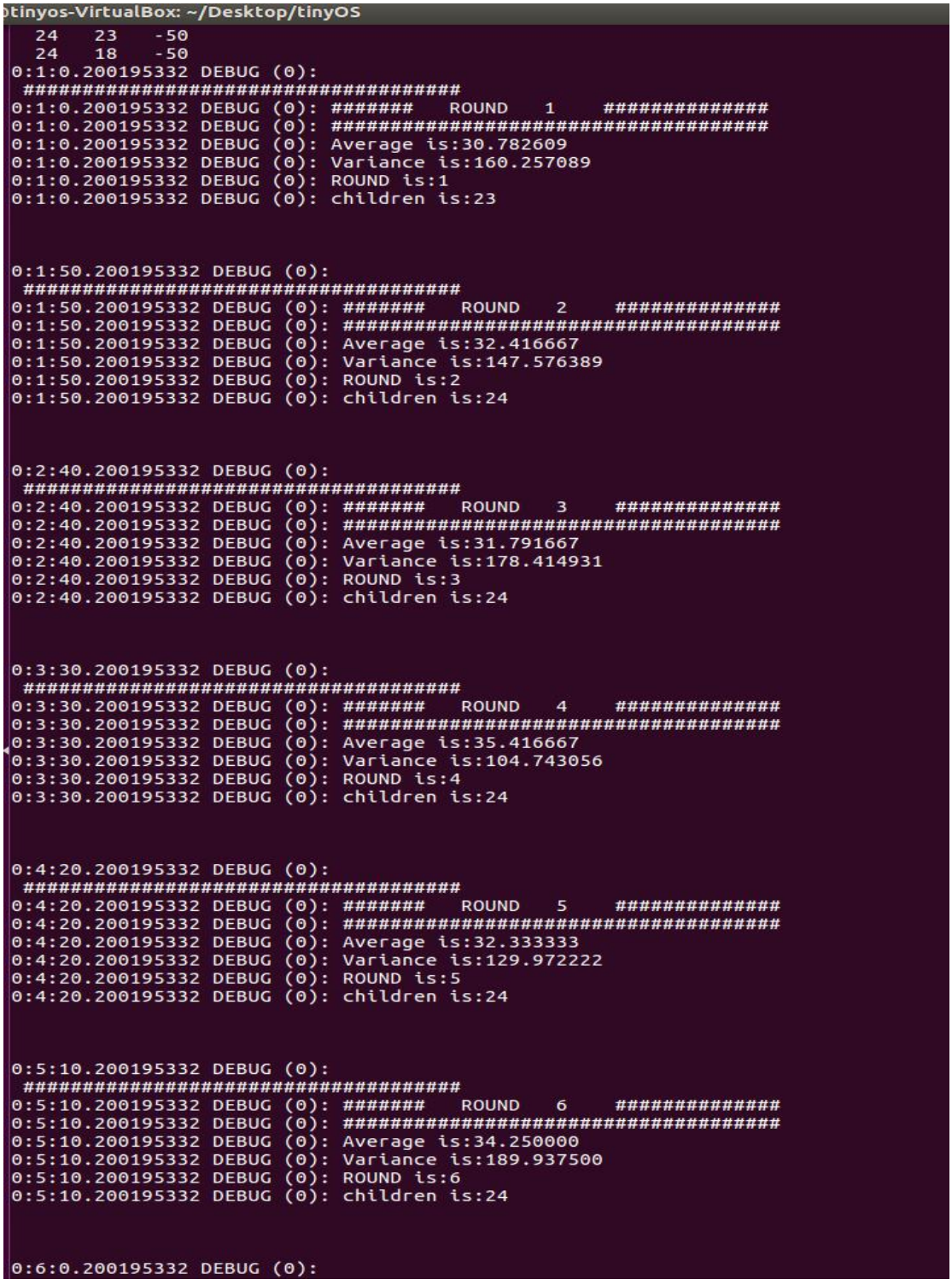
Παίρνουμε τα sum,squares,children όπως αναφέραμε παραπάνω και τα βάζουμε σε μια NotifyPacket ουρά.
Να επισημανθεί εδώ ότι αφού το βάλουμε στην ουρά, κρατάμε και τις τιμές του κόμβου ώστε αν χαθεί μήνυμα σε επόμενο round να στείλουμε τις πιο τελευταία ανανεωμένες τιμές του.

receiveNotifyTask(): Σε αυτή την συνάρτηση ουσιαστικά οι κόμβοι "ακούν" την πληροφορία που τους στέλνουν τα παιδιά τους.
Αν ο εκάστοτε κόμβος έχει ήδη καταχωρίσει τον κόμβο που ακούει ώς παιδί του , ανανεώνει τις τιμές του, αλλιώς, πρώτα, τον προσθέτει στην λίστα των παιδιών του.
Επιπροσθέτως, εδώ είναι που κρατάμε και τις παλιές τιμές των παιδιών του στην περίπτωση που χαθεί μήνυμα.

Γ)Αρχικά για την κατανόηση του κώδικα χρησιμοποιούσαμε ολα τα dbg μηνύματα τόσο στις συναρτήσεις για το Routing που υπήρχαν , οσο και στις συναρτήσεις για την μεταφορά των δεδομένων. Αφού κατανοήσαμε το υπάρχον πρόγραμμα και υλοποιήσαμε το υπόλοιπο ,αφήσαμε dbg μηνύματα μόνο για τα αποτελέσματα που

ζητάει η εκφώνηση ,όπως φαίνεται στο παρακάτω παράδειγμα τρεξίματος του προγραμματος

Δ)



Το παραπάνω παράδειγμα αφορά το τρέξιμο του προγράμματος για 25 κόμβους με το αρχείο τοπολογίας που παράχθηκε από το βοηθητικό μας πρόγραμμα

Ε)Ολόκληρη η εργασία εκπονήθηκε ομαδικώς.