Εργασία στο TinyOS

Βασική σελίδα με περιγραφή του TinyOS:
http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials
Οδηγίες εγκατάστασης του TinyOS:
http://tinyos.stanford.edu/tinyos-wiki/index.php/Installing_TinyOS

ΒΗΜΑ 1: ΕΓΚΑΤΑΣΤΑΣΗ

Στο http://pan.softnet.tuc.gr/other/TinyOS.vdi θα βρείτε ένα virtual machine (για 64bit λειτουργικά) όπου είναι εγκατεστημένο το TinyOS και ο βοηθητικός κώδικας. Εγώ το τρέχω από VirtualBox. Σε αυτή την περίπτωση δε χρειάζεται να κάνετε τίποτα άλλο στο βήμα αυτό.

Αν δε θέλετε να κατεβάσετε το virtual machine, στα Χρήσιμα Έγγραφα έχω τις εντολές που ακολούθησα για την εγκατάσταση το 2017. Ο χρήστης μου λέγεται tinyos (για αυτό πολλά paths περιέχουν το /home/tinyos).

Αν ακολουθήσετε τις εντολές και στη μεταγλώττιση ή την εκτέλεση του προγράμματος λαμβάνετε μήνυμα ότι δε βρίσκει το Python.h ....
Γράψτε:
locate Python.h
Σημειώστε το folder στο οποίο βρίσκεται το Python.h
Εμένα πχ βρίσκεται στο:
/usr/include/python2.7/Python.h
Ανοίξτε το αρχείο $HOME/local/src/tinyos-2x/support/make/extras/sim.extra
Στις πρώτες γραμμές προσθέστε την εντολή
CFLAGS += -I/usr/include/python2.7
Στην παραπάνω εντολή βάζετε δίπλα στο -I
το path του φακέλου που περιέχει το Python.h.
ΠΡΟΣΟΨΗ, σε εσάς μπορεί να είναι σε άλλο μονοπάτι το Python.h

ΠΡΟΣΟΧΗ: Αν κολλήσετε σε κάποιο βήμα, ΜΗΝ προχωρήσετε στο επόμενο. Ανοίξτε ένα θέμα συζήτησης στο courses, στη σελίδα του μαθήματος. Σε περίπτωση που λάβετε κάποιο μήνυμα σφάλματος, κάντε τουλάχιστον μία αναζήτηση στο google με το συγκεκριμένο μήνυμα πριν στείλετε την ερώτησή σας στο courses.

ΒΗΜΑ 2: ΒΟΗΘΗΤΙΚΟΣ ΚΩΔΙΚΑΣ

Αποσυμπιέστε το συμπιεσμένο αρχείο SRTree.zip σε ένα φάκελο (πχ κάτω από το directory apps). **Αν ακολουθήσατε τις παραπάνω οδηγίες, ή κατεβάσατε το virtual machine, τότε θα το έχετε ήδη εγκαταστήσει στο φάκελο /home/tinyos/local/src/tinyos-2x/apps/tinyOS.**

Ο φάκελος που θα δημιουργηθεί έχει μέσα ένα αρχείο Makefile και ένα αρχείο README. Δοκιμάστε να κάνετε compile τον κώδικα με την εντολή:
make micaz sim
Τρέχετε το simulation με την εντολή:
python ./mySimulation.py

ΒΗΜΑ 3: ΚΑΤΑΝΟΗΣΗ ΒΟΗΘΗΤΙΚΟΥ ΚΩΔΙΚΑ

1. ΤΟΠΟΛΟΓΙΑ

Τα αρχεία topology.txt και topology2.txt περιέχουν πληροφορίες για τη συνδεσιμότητα 2 κόμβων σε simulation mode. Εξετάζοντας το αρχείο topology.txt, γειτονικά ζευγάρια κόμβων είναι τα:
0 με 1
1 με 4
1 με 7
1 με 2
4 με 5
Όσα ζευγάρια δεν αναφέρονται στο topology.txt δε θα επικοινωνούν μεταξύ τους.

2. Αρχείο του Simulation

Τα αρχικά στάδια της εργασίας σας θα γίνουν σε simulation mode. Συνεπώς, είναι σημαντικό να κατανοήσετε (σε γενικές γραμμές) το αρχείο **mySimulation.py**
το οποίο τρέχει την προσομοίωση.

Οι γραμμές:
```
from TOSSIM import *
import sys ,os
import random
t=Tossim([])
```
δημιουργούν ένα αντικείμενο του Tossim.

Οι γραμμές:
```
for i in range(0,10):
        m=t.getNode(i)
        m.bootAtTime(10*t.ticksPerSecond() + i)
```
εκκινεί τους κόμβους (από 0..9) σε λίγο διαφορετικές στιγμές.

Οι εντολές
```
f=sys.stdout
t.addChannel("Boot",f)
t.addChannel("RoutingMsg",f)
t.addChannel("NotifyParentMsg",f)
t.addChannel("Radio",f)
#t.addChannel("Serial",f)
t.addChannel("SRTreeC",f)
#t.addChannel("PacketQueueC",f)
```
Καθορίζουν ποια dbg μηνύματα θα εκτυπώνονται στην οθόνη. Αν πχ το πρώτο όρισμα ενός dbg μηνύματος είναι Boot, το αντίστοιχο μήνυμα θα εκτυπωθεί, αφού έχει προστεθεί το κανάλι Boot. Το σύμβολο '#' είναι το σύμβολο για σχόλιο γραμμής.

3. Ουρά πακέτων

Τα αρχεία PacketQueue.nc και PacketQueueC.nc παρέχουν ένα interface και ένα module (αντίστοιχα) για μία ουρά πακέτων. Αυτές οι ουρές χρησιμοποιούνται συχνά στο βοηθητικό κώδικα που σας παρέχεται.

4. Κύριο module: SRTreeC.nc και configuration: SRTreeAppC.nc

Το κεντρικό αρχείο είναι το SRTreeAppC.nc, όπου γίνεται η καλωδίωση (wiring) όλων των components. Θα πρέπει να ξεχωρίσετε:

a) το component MainC (για την εκκίνηση – θα υλοποιήσουμε το event Booted στο SRTreeC).
b) το component LedsC για το χειρισμό των Leds του αισθητήρα
c) για αποστολή/λήψη μηνυμάτων στον ασύρματο, τα components ActiveMessageC, AMSenderC (2: ένα για routing μηνύματα, και ένα για μηνύματα προς τον πατέρα), και AMReceiverC (2: ένα για routing μηνύματα, και ένα για μηνύματα προς τον πατέρα).
d) για αποστολή/λήψη μηνυμάτων στο serial port, τα components SerialActiveMessageC, SerialAMSenderC, και SerialAMReceiverC.
e) Components για τις ουρές μηνυμάτων σχετικά με την αποστολή/λήψη των αντίστοιχων 2 τύπων μηνυμάτων.
f) Διάφορους μετρητές (Timers) με ακρίβεια ms.

Αναζητήστε πως ο κόμβος 0 ξεκινάει ένα μετρητή για να στείλει το πρώτο routing μήνυμα. Παρατηρούμε ότι όταν χτυπάει ο μετρητής, κάνει στο τέλος post ένα task για την πραγματική αποστολή του μηνύματος.

Αντίστοιχα, στο αντίστοιχο task (receiveRoutingTask) λήψης του μηνύματος, ο κάθε κόμβος που δεν είχε πατέρα θέτει το επίπεδό του και τον πατέρα του, και στη συνέχεια στέλνει ένα μήνυμα NotifyParent στον πατέρα του, προτού προωθήσει το μήνυμα. Το συγκεκριμένο task είναι αρκετά σύνθετο, καθώς περιέχει και επιλογές για την περίπτωση που ενημερωθούμε για έναν καλύτερο πατέρα από αυτόν που έχουμε αρχικά επιλέξει.

Ο κώδικας ΔΕΝ είναι βέλτιστος, και περιέχει περιττά κομμάτια, ή/και περιττή πληροφορία στα μηνύματα, ή/και λειτουργία μη σύμφωνη με το TAG/TiNA. Θα χρειαστεί πέραν από προσθήκες και να προβείτε σε διορθώσεις/βελτιώσεις του. Αυτό αποτελεί κομμάτι του βαθμού σας στο 1^{ο} τμήμα της εργασίας.

ΕΡΓΑΣΙΑ

Η εργασία αποτελείται από κομμάτια, τα οποία βασίζονται το ένα στο άλλο:

- Ένα βοηθητικό πρόγραμμα που θα δημιουργεί τις τοπολογίες στις οποίες θα ελέγξετε το πρόγραμμά σας.
- Το πρώτο κομμάτι θα σας μάθει πώς να χρησιμοποιείτε μετρητή (ρολόι) και να στέλνετε/λαμβάνετε μηνύματα, υλοποιώντας μία συναθροιστική συνάρτηση **σύμφωνα με το TAG**.
- Στο δεύτερο κομμάτι θα πρέπει να υλοποιήσετε τη λειτουργία του LEACH. Θα χρειαστείτε το πρώτο κομμάτι για την επικοινωνία των clusterheads (ηγετών) με το σταθμό βάσης.
- Υπάρχει και ένα επιπλέον κομμάτι για τους μεταπτυχιακούς φοιτητές μόνο.

Πιο συγκεκριμένα, καλείστε να κάνετε τα ακόλουθα.

ΒΟΗΘΗΤΙΚΟ ΠΡΟΓΡΑΜΜΑ (Προθεσμία: 3/12)

Για να μπορέσετε να ελέγξετε τη λειτουργία των προγραμμάτων σας, απαιτείται η δημιουργία αρχείων τοπολογίας, που θα περιλαμβάνουν πολλούς αισθητήρες, με «αυτόματο» τρόπο. Στο βοηθητικό πρόγραμμα αυτό καλείστε να δημιουργήσετε (σε όποια γλώσσα προγραμματισμού θέλετε) ένα πρόγραμμα που να:

- Παίρνει ως παραμέτρους 1 ακέραιο (θα αναφερόμαστε σε αυτόν ως διάμετρος D – **θεωρήστε ότι η μέγιστη τιμή του D είναι το 8**) και 1 αριθμό κινητής υποδιαστολής (θα αναφερόμαστε σε αυτόν ως εμβέλεια).
- Θα δημιουργεί D*D κόμβους, με αναγνωριστικά από 0 έως D^2-1, τοποθετημένους σε ένα grid μεγέθους D*D. Ο κόμβος j θα ανήκει στη γραμμή j/D (ακέραια διαίρεση) και στη στήλη j%D.
- Θεωρώντας ότι οι οριζόντιες και κάθετες αποστάσεις των κόμβων στο grid είναι ίσες με 1, είναι εύκολο για έναν οποιοδήποτε κόμβο να βρείτε όλους τους κόμβους που βρίσκονται σε απόσταση μικρότερη ή ίση με την εμβέλειά του (δεύτερη παράμετρος του προγράμματος). Πχ, αν η εμβέλεια είναι 1.5, τότε ένας κεντρικός κόμβος έχει 8 γείτονες (σε σχηματισμό αστεριού γύρω από αυτόν, δηλαδή πάνω/κάτω/αριστερά, δεξιά και στις διαγωνίους.
- Αν για κάθε κόμβο βρείτε τους γείτονές του, τότε μπορείτε αυτή την πληροφορία να τη χρησιμοποιήσετε για να δημιουργήσετε ένα αρχείο τοπολογίας, με παρόμοια μορφή με αυτή που σας δίνεται στο αρχείο topology.txt.

ΠΡΟΓΡΑΜΜΑ 1 (Προθεσμία: 3/12)

Κάθε αισθητήρας θα πρέπει κάθε 60 δευτερόλεπτα να παράγει μία τυχαία τιμή ως μέτρησή του και να προωθεί στον πατέρα του κατάλληλη πληροφορία ώστε να μπορεί να υπολογιστεί τελικά ο μέσος όρος και η μέγιστη τιμή των μετρήσεων σε όλο το δίκτυο. Η λειτουργία ΠΡΕΠΕΙ να γίνεται **ΣΥΜΦΩΝΑ ΜΕ ΤΟ TAG**. Συνεπώς, υλοποιούμε τις συναρτήσεις AVG και MAX, **οι οποίες υπολογίζονται ταυτόχρονα**. Η λειτουργία να τερματίζεται μετά από 900 δευτερόλεπτα. Ο σταθμός βάσης (κόμβος 0) θα πρέπει να τυπώνει το τελικό αποτέλεσμα σε κάθε εποχή.

Η τιμή κάθε κόμβου με αναγνωριστικό Κ σε κάθε εποχή θα είναι ένας τυχαίος ακέραιος αριθμός στο διάστημα [0..50].

Ουσιαστικά, ο κάθε κόμβος είναι καλό να γνωρίζει σε κάποια μεταβλητή τις αντίστοιχες **τελευταίες τιμές που έχει λάβει από τα παιδιά του** και στη συνέχεια να υπολογίζει την πληροφορία που πρέπει να μεταδώσει για το υποδέντρο του.

Το συγκεκριμένο πρόγραμμα ΔΕΝ απαιτεί συγχρονισμό των κόμβων με βάση το TAG και δεν απαιτεί να ανοιγοκλείνετε τον ασύρματο. Θα πρέπει να προσπαθήσετε (μπορεί να μην τη καταφέρετε τέλεια, αλλά έστω μερικώς θα πρέπει να γίνει) να πετύχετε όμως τη λογική του TAG (πρώτα μεταδίδουν τα παιδιά, μετά οι γονείς) χρησιμοποιώντας το επίπεδο του κάθε κόμβου στο δέντρο. Σκεφτείτε ότι η λειτουργία που θα κάνει ο κάθε κόμβος θα είναι επαναλαμβανόμενη και θα εξαρτάται από ένα μετρητή. Αν το σκεφτείτε καλά, πρέπει να ρυθμίσετε πότε θα χτυπήσει πρώτη φορά ο μετρητής αυτός.

Το πρόγραμμά σας θα πρέπει να δουλεύει για οποιοδήποτε αρχείο topology.txt. Η μοναδική υπόθεση που μπορείτε να κάνετε (αν χρειαστεί) είναι για το μέγιστο αριθμό

παιδιών κάθε κόμβου (πχ, 32, ανάλογα με το πώς δημιουργήσατε την τοπολογία σας) και ο μέγιστος αριθμός κόμβων (πχ, 64).

ΠΡΟΓΡΑΜΜΑ 2 (Προθεσμία: 14/12)

Το 2^{ο} κομμάτι της εργασίας περιέχει 2 επεκτάσεις του 1^{ου} προγράμματος. Ο κόμβος 0 θα πρέπει να επιλέγει **τυχαία** ποιο από τα ερωτήματα 2.1 ή 2.2 θα εκτελείται κάθε φορά (πριν αποστείλει το ερώτημα).

2.1 Επέκταση του TAG σε περισσότερες συναρτήσεις

Σας ζητείται να επεκτείνετε τη λειτουργικότητα του TAG ώστε να μπορεί να υπολογίσει τις ακόλουθες συναθροιστικές συναρτήσεις: SUM, AVG, MAX, MIN, COUNT, VARIANCE (διασπορά).

Πιο συγκεκριμένα: Στην 1^{η} εποχή, πριν στείλει ο κόμβος 0 το ερώτημα στο δίκτυο, επιλέγει έναν τυχαίο αριθμό **NUM** από το 1 έως το 2. Στη συνέχεια επιλέγει **τυχαία** NUM από τις παραπάνω 6 συναθροιστικές συναρτήσεις και ζητάει από το δίκτυο **να τις υπολογίσει ταυτόχρονα**. Θα χρειαστεί να σκεφτείτε σοβαρά τη δομή που μπορεί να έχει το μήνυμα των δεδομένων και να έχετε δηλώσει πιθανότατα παραπάνω της μίας τέτοιες δομές.
Δώστε μεγάλη σημασία στο σχεδιασμό σε αυτό το κομμάτι – αν δεν το σκεφτείτε σωστά θα δημιουργήσετε πάρα πολλές δομές (γράφοντας αρκετό κώδικα μετά), ή θα έχετε πλεονασματική πληροφορία στις δομές σας (το οποίο για αισθητήρες ΔΕΝ είναι καλό).

2.2 Υλοποίηση του TiNA

Σας ζητείται να υλοποιήσετε τη λειτουργικότητα του TiNA (δημοσίευση 6β), ώστε να μπορεί να υπολογίσει τις ακόλουθες συναθροιστικές συναρτήσεις: SUM, MAX, MIN, COUNT. Πιο συγκεκριμένα, πριν στείλει ο κόμβος 0 το ερώτημα στο δίκτυο, επιλέγει **τυχαία** μία από αυτές τις 4 συναρτήσεις και τη συμπεριλαμβάνει στο ερώτημα που στέλνει.
Σας ζητείται να χρησιμοποιήσετε τη βελτιστοποίηση που αναφέραμε στο μάθημα για την αποκοπή μηνυμάτων και στους ενδιάμεσους κόμβους του δέντρου.

ΤΙ ΘΑ ΠΑΡΑΔΩΣΕΤΕ (σε 2 φάσεις παράδοσης)

1. Κώδικα **με σχόλια**
2. Μία αναφορά **(1 σε κάθε φάση παράδοσης)** που θα περιγράφει **ΑΝΑΛΥΤΙΚΑ**
   a. Τι διορθώσατε στο αρχείο που σας δόθηκε και γιατί. Βρήκατε κομμάτια που δεν εκτελούνται ποτέ και τα αφαιρέσατε; Αν ναι, ποια; Βρήκατε κομμάτια που δε λειτουργούσαν σύμφωνα με τις αρχές του TAG; Αν ναι, ποια;
   b. Σημαντικά κομμάτια κώδικα για κάθε ερώτημα και επεξήγηση.
   c. Εξήγηση για τα περιεχόμενα κάθε μηνύματος που στέλνετε – σε τι σας χρησιμεύει το κάθε πεδίο;
   d. Παραδείγματα με πίνακες που θα έχουν συγκεκριμένο δέντρο, δεδομένα κόμβων και το τι υπολόγισε στη ρίζα του δέντρου το πρόγραμμά σας.
   e. Ποιος φοιτητής υλοποίησε ποιο κομμάτι.

ΣΗΜΕΙΩΣΗ: Η αναφορά αποτελεί κομμάτι του βαθμού και μία ελλιπής αναφορά που δεν περιέχει όλα τα παραπάνω θα οδηγήσει σε ποινή στο βαθμό.

<u>Επισημαίνεται ότι στη βαθμολογία σας θα μετρήσει ΣΗΜΑΝΤΙΚΑ και το αν έχετε ελαχιστοποιήσει τη μεταδιδόμενη πληροφορία (αριθμό μηνυμάτων και αριθμό bytes σε κάθε μήνυμα).</u> Αυτό πρέπει να γίνει στον κώδικα σας. Δεν αρκεί να τρέχει το πρόγραμμά σας, αλλά και να τρέχει σωστά και βέλτιστα. Το να πείτε στην προφορική εξέταση της εργασίας σας «Ε, ναι, θα μπορούσα να μην το κάνω αυτό» δε μετράει – θα πρέπει να έχετε κάνει τις βελτιστοποιήσεις σας στον κώδικα.

Βαθμολογία 1^{ου} παραδοτέου (βοηθητικό πρόγραμμα + Πρόγραμμα 1 + Αναφορά): 40%

Βαθμολογία 2^{ου} παραδοτέου (Πρόγραμμα 2 + Αναφορά): 60%. Προφανώς, σε αυτό το παραδοτέο πρέπει να συμπεριλάβετε και το βοηθητικό πρόγραμμα του 1^{ου} παραδοτέου.