Εργασία στο TinyOS

Βασική σελίδα με περιγραφή
http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Tutorials

ΒΗΜΑ 1: ΕΓΚΑΤΑΣΤΑΣΗ

Στο https://tucgr-my.sharepoint.com/:u:/g/personal/adeligiannakis_tuc_gr/Eakc0geez6pBkQCgBSvoEFQBR59krWagLVRhLOxs63rNuA?e=miwvTs

θα βρείτε ένα virtual machine (για Ubuntu 64bit λειτουργικά) όπου είναι εγκατεστημένο το TinyOS και ο βοηθητικός κώδικας. Εγώ το τρέχω από VirtualBox.

Ο χρήστης μου λέγεται tinyos και ο κωδικός είναι ο tinyosRULES.

ΠΡΟΣΟΧΗ: Σε καμία περίπτωση μην κάνετε **αυτόματες** ενημερώσεις λογισμικού στο virtual machine, καθώς αν κατεβάσετε νέες εκδόσεις του μεταγλωττιστή θα χρειαστείτε νέα εγκατάσταση (που δε θα σας κάνω ατομικά εγώ) του TinyOS. Εγώ έχω απενεργοποιήσει τις αυτόματες ενημερώσεις στο vdi που σας δίνω – μην τις ενεργοποιήσετε. Μπορείτε αν θέλετε να εγκαταστήσετε χειροκίνητα κάποιες ενημερώσεις.

Μπορείτε από τα settings να αλλάξετε (αν θέλετε) τις ρυθμίσεις της ανάλυσης της οθόνης (Display), σύμφωνα με αυτές του υπολογιστή σας.

Σημείωση: Η συγκεκριμένη έκδοση του Ubuntu έχει ως editors προγράμματα όπως τα gedit, emacs, nano, pico, vi, αλλά όχι άλλα όπως το sublime. Μπορείτε χειροκίνητα να εγκαταστήσετε κάποιο editor της επιλογής σας.

ΒΗΜΑ 2: ΒΟΗΘΗΤΙΚΟΣ ΚΩΔΙΚΑΣ

Ο βοηθητικός κώδικας βρίσκεται **στο φάκελο /home/tinyos/local/src/tinyos-2x/apps/tinyOS.**

Ο φάκελος 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.

Προσέξτε ότι οι γραμμές κώδικα που δημιουργούν το μοντέλο θορύβου απαιτούν αρκετή μνήμη για το VM σας. Αν σας σκάει το πρόγραμμα σε αυτό το σημείο, δοκιμάστε με λιγότερους (πχ 36) κόμβους.

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 τωναντίστοιχων 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.

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 συναθροιστικές συναρτήσεις **σύμφωνα με το TAG**.
- Στο δεύτερο κομμάτι θα πρέπει να προσομοιώσετε τη λειτουργία του MicroPulse.

Πιο συγκεκριμένα, καλείστε να κάνετε τα ακόλουθα.

ΒΟΗΘΗΤΙΚΟ ΠΡΟΓΡΑΜΜΑ (Προθεσμία: 1/12/2024)

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) 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 και στη στήλη j%D.
- Θεωρώντας ότι οι οριζόντιες και κάθετες αποστάσεις των κόμβων στο grid είναι ίσες με1, 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: 1/12/2024)

Ο σταθμός βάσης επιλέγει να εκτελέσει ερωτήματα συνάθροισης. Πιο συγκεκριμένα:

- Η συναθροιστική συνάρτηση που θα πρέπει να υπολογίζεται ~~ανά ομάδα~~ θα είναι είτε η MAX (μέγιστη τιμή), είτε η AVG (μέσος όρος).
- Ο σταθμός βάσης, πριν στείλει το ερώτημα στο δίκτυο θα επιλέγει τυχαία έναν ακέραιο αριθμό RANDOM_NUM από το 1 έως και το 2 (δηλαδή, 1 ή 2) και βάσει αυτού του αριθμού θα επιλέγεται η συναθροιστική συνάρτηση.

Η λειτουργία να τερματίζεται μετά από **600** δευτερόλεπτα. Ο σταθμός βάσης (κόμβος 0) θα πρέπει να τυπώνει το τελικό αποτέλεσμα σε κάθε εποχή, η οποία διαρκεί **40** δευτερόλεπτα.

Κάθε αισθητήρας θα πρέπει στην πρώτη εποχή να παράξει μία τυχαία ακέραια τιμή στο διάστημα [1..50] ως μέτρησή του. Σε κάθε επόμενη εποχή, η μέτρηση ενός αισθητήρα θα επιλέγεται τυχαία, χωρίς όμως να απέχει παραπάνω από ±30% από την προηγούμενη μέτρησή του, και πάντα στο διάστημα [1..50]. Ο κάθε αισθητήρας θα πρέπει να προωθεί στον πατέρα του κατάλληλη πληροφορία ώστε να μπορεί να υπολογιστεί τελικά η ζητούμενη συναθροιστική συνάρτηση σε όλο το δίκτυο.

Η λειτουργία του κώδικά σας ΠΡΕΠΕΙ να γίνεται **ΣΥΜΦΩΝΑ ΜΕ ΤΟ TAG**, τροποποιώντας κατάλληλα πράγματα που υπάρχουν στο βοηθητικό κώδικα που σας δίνω, όταν κάτι δε συμβαδίζει με τη λογική του TAG, ιδίως κατά τη διαδικασία του routing.

Σας ζητείται να υλοποιήσετε τη βελτιωμένη λογική του TAG, όπως την περιγράψαμε στο μάθημα, με μικρά παράθυρα αποστολής και λήψης (π.χ., 50-100 ms) για κάθε κόμβο. Προσέξτε ότι ανάλογα με την τιμή του RANDOM_NUM, αλλάζει ο όγκος της πληροφορίας που θα χρειαστεί να μεταδίδει ο

κάθε κόμβος. Θυμηθείτε ότι το TAG στέλνει το routing μήνυμα μόνο 1 φορά.

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 την 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. Ίσως σας διευκολύνει να δείτε τη χρήση των συναρτήσεων

```
sim_time_t    sim_time()          // Επιστρέφει το χρόνο σε αριθμό από ticks
sim_time_t    sim_ticks_per_sec() // Επιστρέφει τον αριθμό από ticks σε ένα εικονικό sec
```

Επίσης, θα πρέπει ο χρόνος στον οποίο θα λειτουργεί ένας κόμβος να ελαχιστοποιηθεί, καθώς ζητούμενο είναι οι κόμβοι να κοιμούνται για όσο το δυνατόν μεγαλύτερο διάστημα.

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, 16, ανάλογα με το πώς δημιουργήσατε την τοπολογία σας), ή το μέγιστο αριθμό κόμβων στο δίκτυο (πχ, 100).

ΠΡΟΓΡΑΜΜΑ 2 (Προθεσμία: 15/12/2023)

Σας ζητείται να υλοποιήσετε τη λειτουργία του MicroPulse. Πιο συγκεκριμένα:

- Κάθε κόμβος τραβάει έναν τυχαίο ακέραιο αριθμό στο διάστημα [20, 60] που αναπαριστά το φόρτο μετάδοσης δεδομένων προς τον πατέρα του.
- Κάπου μέσα στην $5^{η}$ εποχή οι κόμβοι υπολογίζουν (χρησιμοποιώντας την παραπάνω πληροφορία) το κρίσιμο μονοπάτι και μετά προγραμματίζουν τα παράθυρα χρόνου στα οποία θα στέλνουν δεδομένα σύμφωνα με το MicroPulse. Θυμηθείτε ότι η διαδικασία αυτή αποτελείται από 2 φάσεις: στην $1^{η}$ φάση υπολογίζεται το κρίσιμο μονοπάτι από τα φύλλα προς τη ρίζα, ενώ η $2^{η}$ φάση εκτελείται από τη ρίζα προς τα φύλλα. Η διαδικασία αυτή πρέπει να ολοκληρωθεί εντός της $5^{ης}$ εποχής και οι χρονισμοί που θα αποφασίσουν οι κόμβοι να εφαρμοστούν από την $6^{η}$ εποχή και μετά, μέχρι το τέλος του προγράμματος.
- Για να σας βοηθήσω, προσέξτε ότι ο υπολογισμός του κρίσιμου μονοπατιού μοιάζει με την MAX συνάρτηση, ενώ η μετάδοση της πληροφορίας (από τη ρίζα προς τα φύλλα) σχετικά με τα παράθυρα αποστολής μοιάζει με τη διαδικασία του routing.
- Μπόνους έως 20% του βαθμού του $2^{ου}$ προγράμματος θα έχουν όσοι (προαιρετικά) κάνουν κάτι έξυπνο για να αντιμετωπίσουν μηνύματα που πιθανώς θα χαθούν κατά τη διαδικασία υπολογισμού του κρίσιμου μονοπατιού και χρονισμού των κόμβων (κατά οποιαδήποτε από τις 2 φάσεις υπολογισμού του).

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 δελειτουργούσαν σύμφωνα με τις αρχές του TAG; Αν ναι, ποια;
   b. Σημαντικά κομμάτια κώδικα (με screenshots) για κάθε ερώτημα και επεξήγηση.
   c. Εξήγηση για τα περιεχόμενα κάθε μηνύματος που στέλνετε – σε τι σας χρησιμεύει το κάθε πεδίο;
   d. Schreenshots από εκτελέσεις του προγράμματος. 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. Η δήλωση αυτή πρέπει να είναι απολύτως ακριβής. Σε περίπτωση αναληθών δηλώσεων θα υπάρχει βαθμολογική ποινή σε όλα τα μέλη της ομάδας.

Η αναφορά βαθμολογείται και είναι σημαντικό να είναι αναλυτική και να περιγράφει την πληροφορία που αναφέρθηκε παραπάνω. Ελλιπείς ή ανύπαρκτες αναφορές θα οδηγήσουν σε σημαντική απώλεια βαθμών, άσχετα με το πόσο καλά τρέχει ο κώδικας.
Αναφορές που θα λένε ότι ο κώδικας τρέχει σωστά, ενώ προφανώς αυτό δε συμβαίνει όταν τον τρέξω, θα έχουν σημαντική απώλεια βαθμών.

Επισημαίνεται ότι στη βαθμολογία σας θα μετρήσει ΣΗΜΑΝΤΙΚΑ και το αν έχετε ελαχιστοποιήσει τη μεταδιδόμενη πληροφορία (αριθμό μηνυμάτων και αριθμό bytes σε κάθε μήνυμα), καθώς και το χρόνο που που ακούει/λαμβάνει δεδομένα ένας κόμβος. 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 του πρότζεκτ σας «Ε, ναι, θα μπορούσα να μην το κάνω αυτό» δε μετράει – θα πρέπει να έχετε κάνει τις βελτιστοποιήσεις σας στον κώδικα.

Σημείωση: Στο 2ο παραδοτέο πρέπει να συμπεριλάβετε και το βοηθητικό πρόγραμμα του 1ου παραδοτέου.