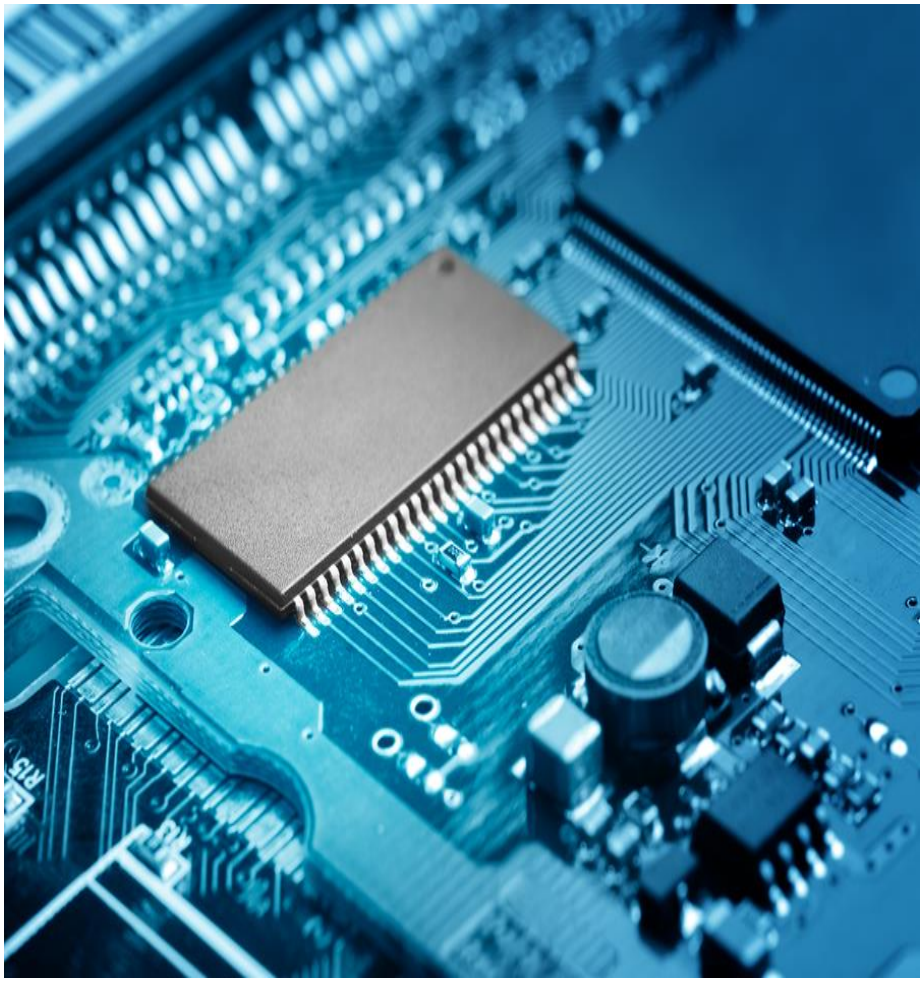
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ



ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ

Περιεχόμενο

Ο σκοπός της συγκεκριμένης εργασίας είναι η δημιουργία μιας εφαρμογής για την ιχνηλάτηση του Covid

Χριστιανού Γεωργία
ΑΕΜ: 8419

gkchristi@ece.auth.gr

Περιεχόμενα

Εισαγωγή

Ο στόχος της εργασίας ήταν να δημιουργηθεί ένα πρόγραμμα που να λειτουργεί ως μια εφαρμογή ιχνηλάτησης κρουσμάτων για τον covid. Πρακτικά χρησιμοποιούμε μια fifo ουρά και κρατάμε σε αυτή τις mac διευθύνσεις των συσκευών με τις οποίες έχουμε έρθει σε επαφή για ένα ορισμένο χρονικό διάστημα. Σε περίπτωση που είτε εμείς είτε μια στενή μας επαφή βρεθεί θετική στον covid πρέπει να επικοινωνήσουμε με όλες τις συσκευές που έχουμε αποθηκεύσει ώστε να τις ενημερώσουμε για την πιθανότητα νόσησης μας. Στην συνέχεια αναλύονται οι θεματικές ενότητες στις οποίες έχει χωριστεί ο κώδικας και θα αναφερθούν ορισμένες δυσκολίες καθώς και μελλοντικές βελτιστοποιήσεις.

Ο κώδικας της εργασίας μπορεί να βρεθεί στον εξής σύνδεσμο:
https://github.com/geochristi/embedded.systems

Ρυθμίσεις

Δημιουργήθηκε ένα ad-hoc δίκτυο δηλαδή ένα δίκτυο που ο κάθε χρήστης έχει τις ίδιες δυνατότητες (Peer to peer). Η κάθε συσκευή που χρησιμοποιείται πρέπει να είναι ρυθμισμένη σε ένα δίκτυο με συγκεκριμένο όνομα και κωδικό ώστε το μόνο που να χρειάζεται για τη σύνδεση με κάποια άλλη στο ίδιο δίκτυο να είναι η IP της. Για την υλοποίηση και τον έλεγχο χρησιμοποιήθηκε ένας υπολογιστής (linux mint 20.3 σε επεξεργαστή Intel Core i5-8300H με 4 πυρήνες (8 logical processors) και συχνότητα 2.3GHz), με τοπική IP 10.0.0.7, και δύο rasberry pi 0 με IP 10.0.84.19 και 10.0.84.8 αντίστοιχα.

Ένας από τους στόχους της εργασίας ήταν η υλοποίηση της με την χρήση pthread για να επιτύχουμε καλύτερο real-time behaviour. Επομένως δημιουργήθηκαν 6 βασικά threads, 5 από αυτά τρέχουν συνεχόμενα και καλούνται στην main συνάρτηση, και το 6ο καλείται μόνο σε περιπτώσεις που πρέπει να επιτευχθεί επικοινωνία.
Όσον αφορά την λειτουργία του προγράμματος στον υπολογιστή τα threads δημιουργούν μεγάλη διαφορά τόσο στη χρήση της cpu όσο και στον πραγματικό χρόνο λειτουργίας της εφαρμογής.
Τα rasberry pi 0 διαθέτουν μόνο ένα πυρήνα το καθένα άρα δεν μπορεί να υπάρξει πραγματικός παραλληλισμός αλλά ταυτοχρονισμός μέσω pipelining.
Τέλος αφού υπήρχαν μόνο αυτά τα διαθέσιμα μέσα, για να γίνουν κάποιες σωστές μετρήσεις το πρόγραμμα έτρεξε κατά κύριο λόγο με επικοινωνία μόνο για αυτές τις διευθύνσεις.

Ουρά queue

Η υλοποίηση της ουράς μας δόθηκε έτοιμη στην προηγούμενη εργασία. Αποτελείται από ένα struct που κρατάει ως στοιχεία του την κεφαλή και το τέλος της ουράς καθώς και ορισμένες μεταβλητές για έλεγχο σε περίπτωση που η ουρά είναι άδεια ή γεμάτη. Τελικά προστέθηκε το struct contact το οποίο αποτελείται από τα στοιχεία που χρειαζόμαστε για να συγκρίνουμε τις επαφές μας στο κυρίως πρόγραμμα. Επίσης μας δόθηκαν οι συναρτήσεις αρχικοποίησης και διαγραφής της ουράς καθώς και η εισαγωγή και διαγραφή στοιχείων.

Συνάρτηση server()

Η συγκεκριμένη συνάρτηση αρχικοποιεί το σύστημα για την λήψη μηνυμάτων. Ουσιαστικά ανοίγει καινούργιο socket και το αρχικοποιεί, περιμένοντας μηνύματα από άλλα συστήματα στο ίδιο δίκτυο. Έπειτα μέσω της συνάρτησης listen() επιτρέπουμε στο σύστημα να συνδεθεί με έως 3 συσκευές ταυτόχρονα.
Τελικά, η συνάρτηση μπαίνει σε ατέρμονη επανάληψη και σε περίπτωση επικοινωνίας δέχεται το μήνυμα μέσω της συνάρτησης accept(),το διαβάζει με τη συνάρτηση read() και στην πορεία το εκτυπώνει. Τελικά καλεί την συνάρτηση client

καθώς αφού λάβαμε μήνυμα κάποια από τις κοντινές μας επαφές είναι θετική στον covid και πρέπει να ενημερώσουμε και εμείς τις δικές μας κοντινές επαφές.

Συνάρτηση client()

Η συγκεκριμένη συνάρτηση αρχικά δημιουργεί ένα socket, λαμβάνει τις πληροφορίες τις πληροφορίες που χρειάζεται για την IP με την οποία επιθυμούμε να επικοινωνήσει και τέλος συνδέεται με αυτήν. Αφού συνδεθεί της στέλνει το μήνυμα ότι ο χρήστης της συσκευής αυτής είναι θετικός στον covid.

Συνάρτηση timer()

Η συνάρτηση αυτή εισάγει μια καθυστέρηση 10 δευτερολέπτων ανάμεσα στις αναζητήσεις των mac διευθύνσεων. Ουσιαστικά ανά 10 δευτερόλεπτα καλεί μια συνάρτηση για την εύρεση των διευθύνσεων.
Η συνάρτηση αυτή έχει επίσης τρεις μετρητές counter, counter2 και counter3 αντίστοιχα.
Ο πρώτος αυξάνεται και όταν φτάσει να μετρήσει 22 λεπτά στέλνει σήμα μέσω του wait_timer signal στην συνάρτηση find_close_contacts που περιμένει για να την ενεργοποιήσει.
Ο δεύτερος αντίστοιχα μετράει 14 μέρες ώστε να στείλει σήμα μέσω του delete_close_wait signal στην συνάρτηση delete_close_contacts.
Τελικά ο τρίτος μετράει 4 ώρες ώστε να στείλει σήμα μέσω του test_wait signal στη συνάρτηση test.

Συνάρτηση find_mac()

Η συνάρτηση αυτή καταρχάς επιλέγει με τυχαίο βαθμό αν έχει βρεθεί κάποια mac διεύθυνση σε απόσταση αρκετά κοντινή για να την ανακαλύψουμε και αν αποφασίσει ότι υπάρχει, επιλέγει μια τυχαία τιμή από τον πίνακα με τις Mac διευθύνσεις. Αυτός ο πίνακας έχει αρχικοποιηθεί στην αρχή του προγράμματος. Έπειτα αποθηκεύει στην ουρά addresses τον χρόνο της επαφής, την διεύθυνση mac και τέλος την θέση της στον αρχικό πίνακα Macs. Για τους υπολογισμούς με βάση τον αριθμό των συσκευών που είχαμε στη διάθεση μας σε περίπτωση που βρεθεί κάποια mac σε κοντινή απόσταση, θα είναι μια από τις 2 με τις οποίες μπορούμε πρακτικά να επικοινωνήσουμε.

Ιδανικά: το πρόγραμμα, αντί να λαμβάνει μια τυχαία τιμή από τον πίνακα των mac, χρησιμοποιεί τον κώδικα που ήδη υπάρχει στην συνάρτηση (σε σχόλιο) και αποθηκεύει τις mac διευθύνσεις των συσκευών που βρίσκονται στο ίδιο δίκτυο.

Συνάρτηση find_close_contacts()

Η συνάρτηση αυτή αρχικά περιμένει ένα σήμα από το wait_timer signal. Δηλαδή περιμένει την συνάρτηση timer να την ενημερώσει πότε θα είναι η ώρα της να ενεργοποιηθεί. Για την εύρεση κοντινών επαφών η εκφώνηση της άσκησης ζητάει να έχουν περάσει 4-20 λεπτά με επαναλαμβανόμενη επαφή για κάθε mac διεύθυνση. Επομένως η συνάρτηση καταρχάς ελέγχει ότι η ουρά με τις διευθύνσεις δεν είναι άδεια. Έπειτα ελέγχει μήπως το πρώτο στοιχείο της βρίσκεται στην ουρά για παραπάνω από 41 λεπτά και το διαγράφει (η συνάρτηση θα είχε χρόνο να το βρει ακόμα και στις οριακές του τιμές αν υπήρχε ξανά). Έπειτα ξεκινάει να ψάχνει μέσα στην ουρά για mac διευθύνσεις που επαναλαμβάνονται. Η έρευνα γίνεται με τον συγκρίνουμε το πρώτο στοιχείο με το τελευταίο και έπειτα το δεύτερο με το τελευταίο και ούτε καθεξής. Αν το πρώτο στοιχείο στην ουρά έχει λιγότερο από 4 λεπτά που βρίσκεται στην ουρά κατευθείαν η έρευνα σταματάει. Η αναζήτηση γίνεται ως εξής:

Σε περίπτωση που βρεθεί η ίδια mac στην ουρά δύο φορές το πρόγραμμα συγκρίνει το timestamp τους, δηλαδή το πόση ώρα έχει περάσει από την πρώτη φορά που έχει σώσει την mac αυτή με την τελευταία.

1. Σε περίπτωση που αυτό το timestamp είναι μεγαλύτερο των 20 λεπτών που ζητάει η εκφώνηση το πρώτο στοιχείο της επαφής αυτής διαγράφεται από την ουρά.
2. Σε περίπτωση που το timestamp είναι μεγαλύτερο από 4 λεπτά και μικρότερο από 20 το πρώτο στοιχείο της επαφής αυτής προστίθεται στην δεύτερη ουρά με την συνάρτηση save_close_contact. Επίσης αποθηκεύεται και η mac της διεύθυνση στην μεταβλητή history έτσι ώστε να μην υπάρξουν διπλοεγγραφές των επαφών στο συγκεκριμένο κομμάτι εύρεσης τους. Έπειτα διαγράφουμε το πρώτο στοιχείο της επαφής από την αρχική ουρά καθώς δεν θα το χρησιμοποιήσουμε περαιτέρω.
3. Στην τελευταία περίπτωση που το timestamp είναι μικρότερο από 4 λεπτά αποχωρούμε από την διαδικασία εύρεσης κοντινών επαφών και περιμένουμε ξανά να μας καλέσει η συνάρτηση timer για να ξαναρχίσουμε.

Σε περίπτωση που δεν βρεθεί η ίδια mac στην ουρά μετακινούμαστε πιο αριστερά από την ουρά μέχρι είτε να βρεθεί ίδια mac είτε να εξαντλήσουμε τους πιθανούς συνδυασμούς της ουράς.

Συνάρτηση save_close_contacts()

Η συνάρτηση αυτή αποθηκεύει τις επαφές που θεωρήθηκαν κοντινές σε μια δεύτερη ουρά την close_contacts με χρήση της συνάρτησης closeContactAdd που είναι παρόμοια με την queueAdd.

Συνάρτηση delete_close_contacts()

Η συνάρτηση αυτή διαγράφει τις κοντινές επαφές οι οποίες βρίσκονται στην ουρά για παραπάνω από 14 μέρες. Καταρχάς η συνάρτηση περιμένει σήμα από την συνάρτηση

timer() που σημαίνει ότι έχουν περάσει 14 μέρες από τη στιγμή που επαναλήφθηκε η συγκεκριμένη διαδικασία. Η συνάρτηση ελέγχει κάθε μια από τις κοντινές επαφές ξεκινώντας από την πρώτη στην ουρά και σε περίπτωση που το timestamp τους είναι μεγαλύτερο από 14 μέρες τις διαγράφει. Σε περίπτωση που φτάσουμε σε επαφή που το timestamp της είναι μικρότερη από 14 μέρες η συνάρτηση τερματίζει μέχρι να την ξαναξυπνήσουμε, αφού δεν υπάρχει περίπτωση να υπάρχει μετέπειτα στην ουρά επαφή με μεγαλύτερο timestamp.

Συνάρτηση test()

Η συνάρτηση αυτή δημιουργήθηκε για να δίνει αποτελέσματα για την εξέταση covid του χρήστη. Καταρχάς περιμένει 4 ώρες μέχρι δηλαδή να του έρθει σήμα από τη συνάρτηση timer() και έπειτα διενεργεί το τεστ. Αν τα αποτελέσματα του τεστ είναι αρνητικά τερματίζει μέχρι να ξανακαλεστεί. Αντίθετα αν τα αποτελέσματα του τεστ είναι θετικά πρέπει να ενημερώσει τις κοντινές επαφές οπότε καλεί την συνάρτηση save_server(). Η διενέργεια του test για τους σκοπούς της άσκησης έχει πιθανότητα 75% να είναι αρνητικό και 25% να είναι θετικό.

Συνάρτηση save_server()

Η συνάρτηση αυτή καλείται όταν είτε το δικό μας τεστ covid έχει βγει θετικό, είτε όταν έχουμε έρθει σε επαφή με επιβεβαιωμένο κρούσμα. Η συνάρτηση δημιουργεί ένα thread που καλεί την συνάρτηση client η οποία θα επικοινωνήσει με τις κοντινές επαφές για να τις ενημερώσει για την κατάσταση όπως έχουμε εξηγήσει παραπάνω. Επιλέγουμε να στείλουμε μόνο μια φορά σε κάθε συσκευή, ακόμα κι αν υπάρχουν διπλοεγγραφές στην ουρά μας.

Αρχεια για μετρήσεις

Δημιουργούμε 4 binary αρχεία.

1. Το πρώτο “getmessages.bin” κρατάει αρχείο το πόσα μηνύματα έχω λάβει.
2. Το δεύτερο “send_communication.bin” κρατάει αρχείο το πόσες επιτυχημένες και αποτυχημένες προσπάθειες είχα για τις επικοινωνίες που έγιναν όσο έτρεχε το πρόγραμμα.
3. Το τρίτο και πιο σημαντικό αρχείο “cpu_vs_real.bin” αποτελείται από δύο στήλες, το χρονικό timestamp της cpu και του πραγματικού χρόνου, από τη στιγμή που ξεκινάει το πρόγραμμα μέχρι και αφού επικοινωνήσω με τις κοντινές μου επαφές, κάνοντας πρακτικά ένα ολοκληρωτικό πέρασμα του προγράμματος. Η επόμενη μέτρηση ξεκινάει από το τέλος της προηγούμενης. Είναι φανερό ότι με την χρήση των threads και των signals ο χρόνος χρήσης της cpu είναι ιδιαίτερα χαμηλότερος από τον πραγματικό χρόνο που πέρασε.

4. Τέλος το αρχείο “times.bin” αποτελείται επίσης από δύο στήλες με το timestamp της cpu και του πραγματικού χρόνου από την στιγμή της έναρξης της επικοινωνίας των δύο συσκευών μέχρι και το τέλος της. Εδώ μπορούμε εύκολα να παρατηρήσουμε ότι οι δύο χρόνοι δεν διαφέρουν ιδιαιτέρως.

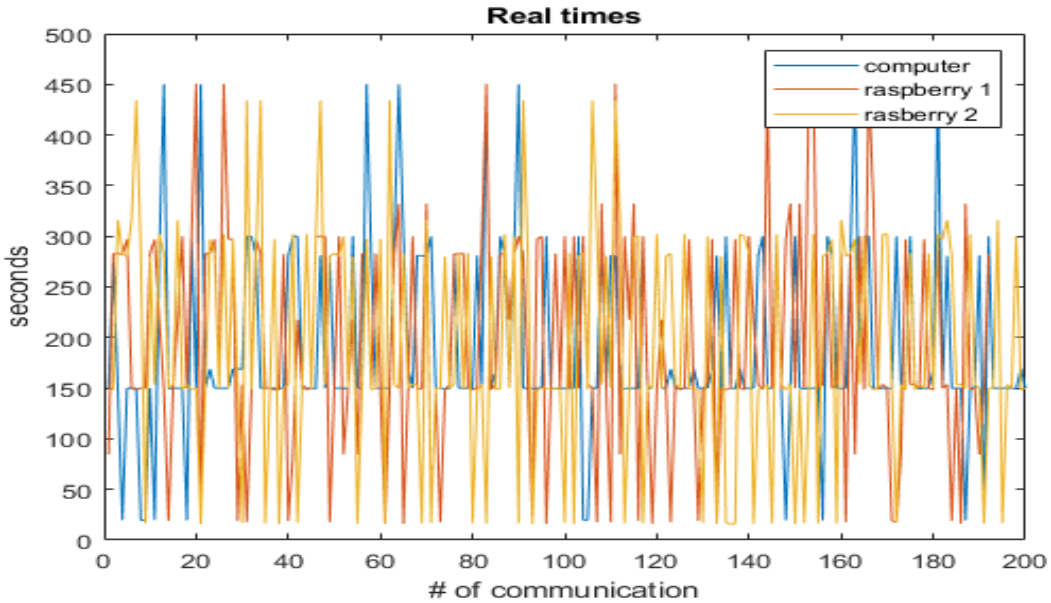
Παραμετροποιήσεις

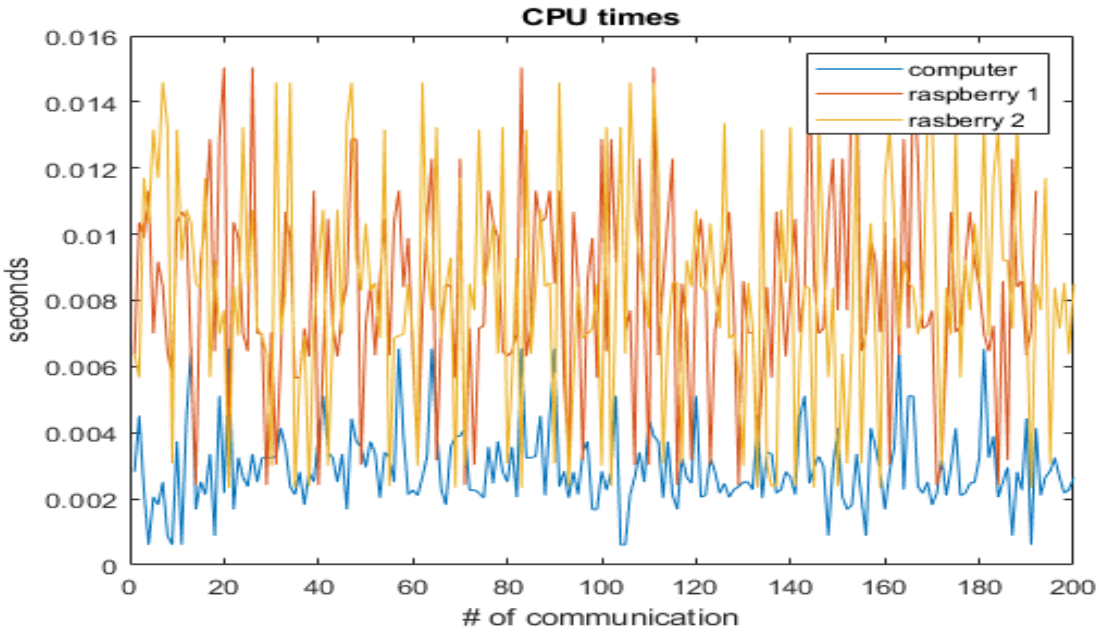
Η εργασία ζητούσε το πρόγραμμα να τρέχει για 30 μέρες αλλά εμείς επιταχύναμε την διαδικασία κατά 100 φορές. Επομένως οι παράμετροι με τις οποίες έγιναν οι έλεγχοι είναι:

- αντί για 20+ λεπτά για τον έλεγχο των κοντινών επαφών 12.6 δευτερόλεπτα
- αντί για 14 μέρες διαγραφής των κοντινών επαφών 12096 δευτερόλεπτα
- αντί για 4 ώρες ανά τεστ covid 144 δευτερόλεπτα

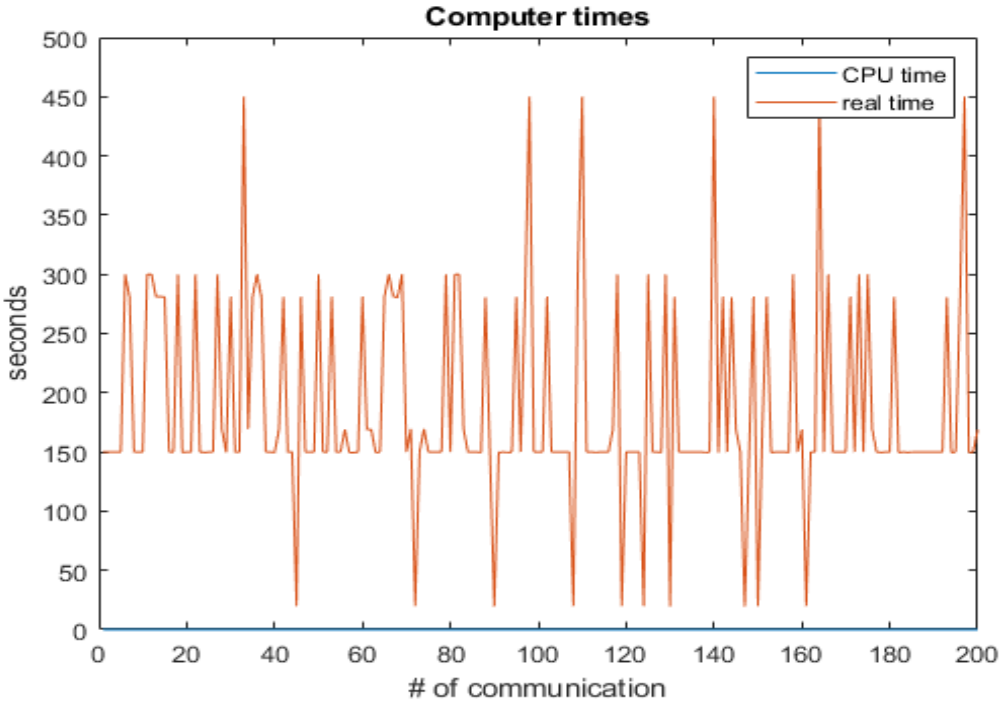
Στατιστικά

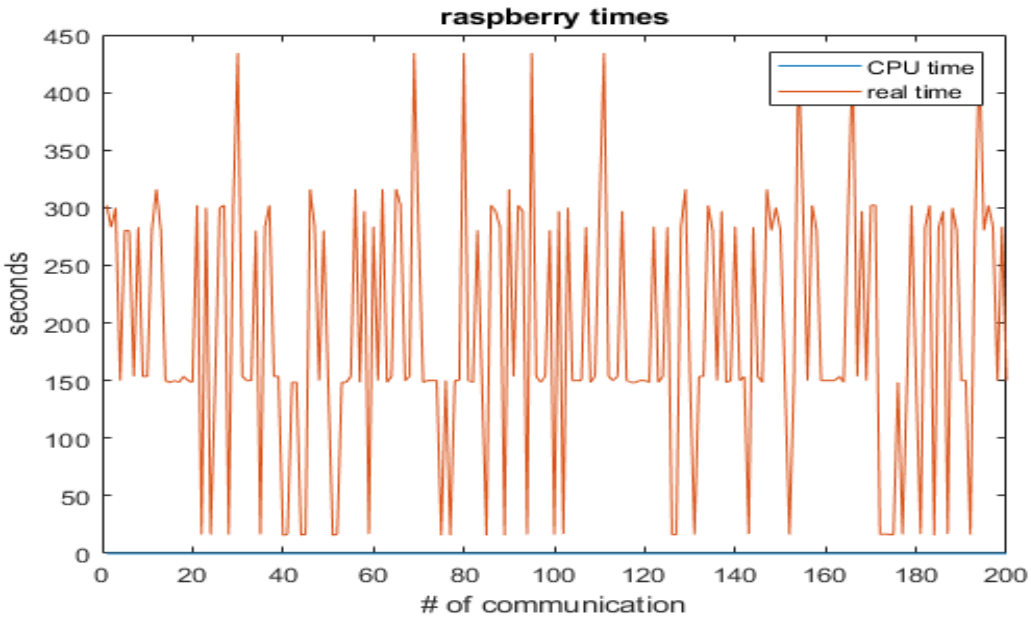
Στα επόμενα διαγράμματα παρουσιάζονται οι χρόνοι που μετρήθηκαν από την έναρξη του προγράμματος (ή από το τέλος της προηγούμενης μέτρησης) μέχρι και μια επιτυχημένη ή αποτυχημένη επικοινωνία.
Είναι φανερό ότι οι χρόνοι παραμένουν σχεδόν σταθεροί και στα δύο διαγράμματα. Είναι ενδιαφέρον να παρατηρήσουμε την μεγάλη διαφορά στους πραγματικούς χρόνους (150-300sec) σε σχέση με τους χρόνους χρήσης της cpu (0.02-0.012sec). Επίσης ενδιαφέρον έχει η φανερή, αν και μικρή σε τιμή, διαφορά στους χρόνους της χρήσης cpu από τον υπολογιστή σε σύγκριση με τα raspberry pi. Τελικά, ο μικρός χρόνος χρήσης της cpu είναι ιδιαίτερα χρήσιμος σε ένα ενσωματωμένο σύστημα που θέλουμε να λειτουργεί σε πραγματικό χρόνο.



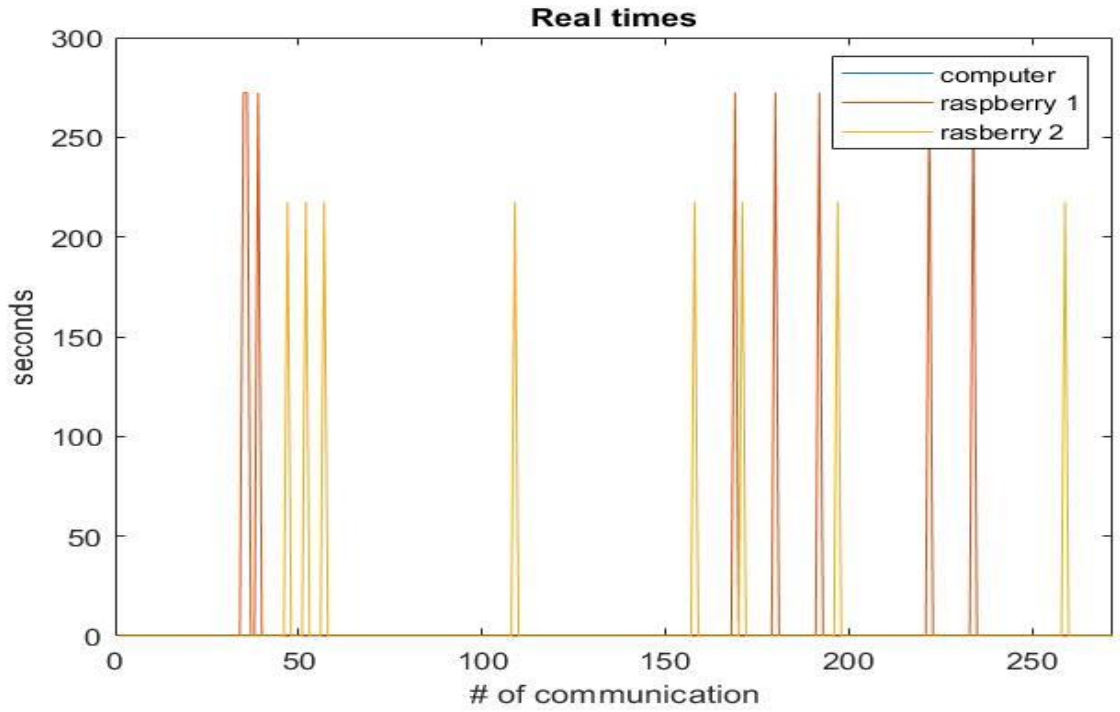


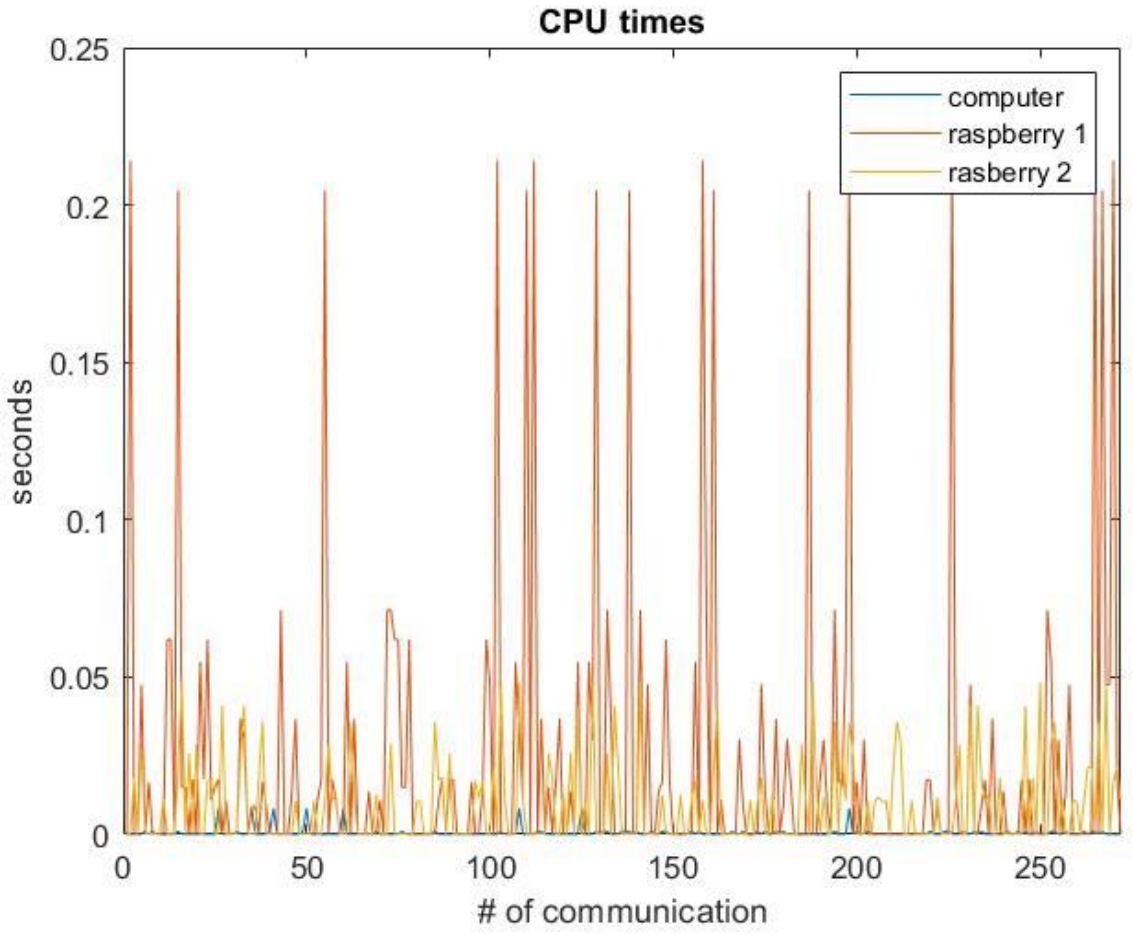
Παρακάτω παρουσιάζονται τα διαγράμματα αυτή τη φορά με σύγκριση του πραγματικού χρόνου με το χρόνο της cpu στον υπολογιστή και σε ένα raspberry pi. Στα συγκεκριμένα διαγράμματα φαίνεται επίσης ότι ο υπολογιστής καταφέρνει να κρατήσει πιο σταθερό χρόνο ανάμεσα στις επικοινωνίες του.



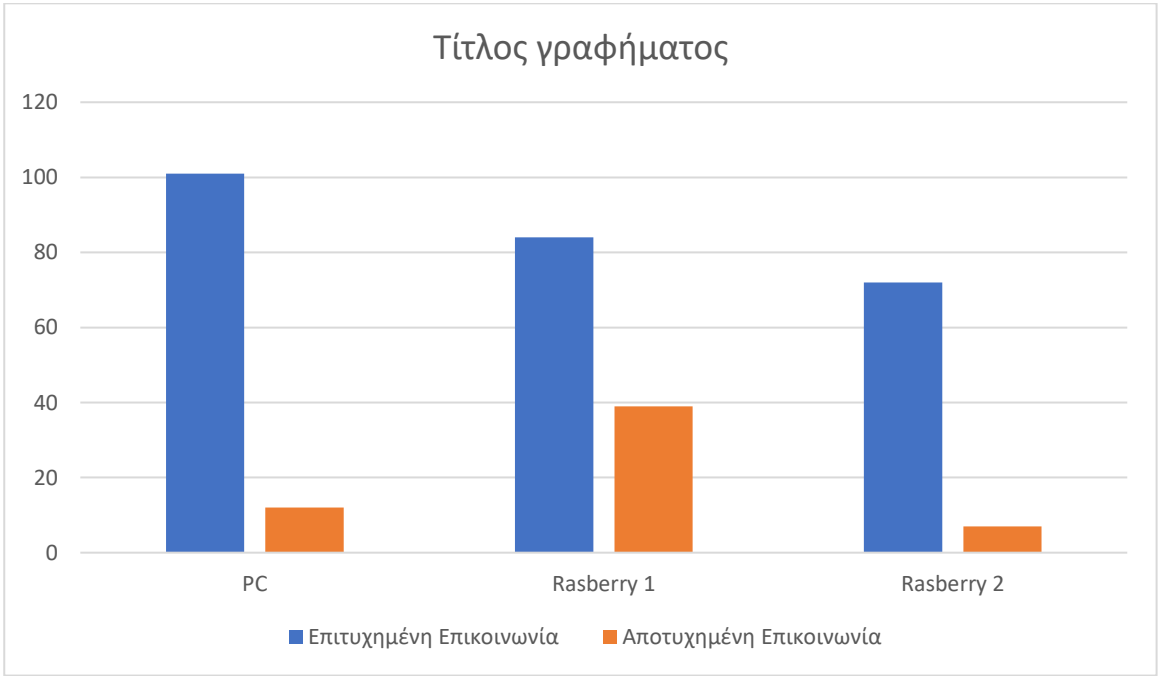


Στο επόμενο διάγραμμα παρουσιάζονται οι μετρήσεις χρόνου από τη στιγμή προσπάθειας επικοινωνίας μέχρι και την επιτυχία ή αποτυχία της. Στα συγκεκριμένα διαγράμματα δεν συμπεριλήφθηκε το σφάλμα του υπολογιστή σε μια συχνότητα του 10% της επικοινωνίας. Σε αυτή την περίπτωση βλέπαμε ότι ο υπολογιστής προσπαθούσε να επικοινωνήσει για σχεδόν την τριπλάσια ώρα από τα raspberry pi 0. Επίσης είναι σημαντικό να παρατηρήσουμε ότι στους χρόνους της cpu οι χρόνοι είναι πολύ χαμηλοί για όλες τις συσκευές, ακόμα και σε περίπτωση σφάλματος.





Τελικά στα παρακάτω διαγράμματα φαίνονται πόσα μηνύματα προσπάθησε να στείλει ο υπολογιστής και το raspberry pi αντίστοιχα, και πόσα από αυτά ήταν επιτυχημένα, δηλαδή ο άλλος τα έλαβε. Οι μετρήσεις αυτές βγήκαν τρέχοντας το πρόγραμμα για 2 ώρες.



Βλέπουμε το πλήθος των επικοινωνιών που είχε η κάθε συσκευή και πόσες από αυτές ήταν επιτυχημένες ή όχι.
Έτσι από τον υπολογιστή φαίνεται ότι το 10% των μηνυμάτων που προσπάθησε να στείλει δεν μπορέσανε να φύγουνε. Αντίστοιχα για το ένα raspberry pi η τιμή αυτή είναι σχεδόν 50% ενώ για την άλλη είναι στο 5%. Αυτό συμβαίνει γιατί η μια συσκευή δεν κάνει καλή επαφή με το καλώδιο τροφοδοσίας και έσβηνε. Επίσης το πλήθος των μηνυμάτων που στάλθηκαν είναι τυχαίο καθώς εξαρτάται από τις συναρτήσεις find_mac και find_close_contact.

Μελλοντικές βελτιστοποιήσεις:

- Στην συνάρτηση find_close_contact θα μπορούσε η εύρεση επαφών να γίνεται με παράλληλο search της ουράς.
- Περισσότερα mac addresses καθώς τώρα που είναι λίγα υπάρχει μεγαλύτερη πιθανότητα κάποιο από αυτά να βρεθεί πολλές φορές και να ξεπεράσει τα 20 λεπτά επαφής οπότε να διαγραφεί.
- Έλεγχος για το κάθε μήνυμα που στέλνω ότι παραδίδεται ολόκληρο.
- Προσπάθεια επαναποστολής μηνύματος αν αυτό δεν επιτευχθεί
- Η χρήση πραγματικών mac διευθύνσεων δεν ενδείκνυται για επικοινωνία καθώς προσδιορίζουν την φυσική υπόσταση της συσκευής. Βρέθηκε τρόπος με χρήση raw sockets να επικοινωνήσω με γνωστή συσκευή μέσω της διεύθυνσης mac καθώς και να βρεθούν οι mac διευθύνσεις του δικτύου αλλά δεν πρόλαβα να τα συνδυάσω για να δουλέψουν στο πρόγραμμα, οπότε το πρόγραμμα επικοινωνεί μέσω IP.