Ενσωματωμένα Συστήματα Πραγματικού Χρόνου

Τελική Εργασία

Σύνδεσμος κώδικα: Github

Όνομα : Αλέξανδρος Δελητζάς
AEM : 8448
e-mail : adelitzas@ece.auth.gr

15 Φεβρουαρίου 2020

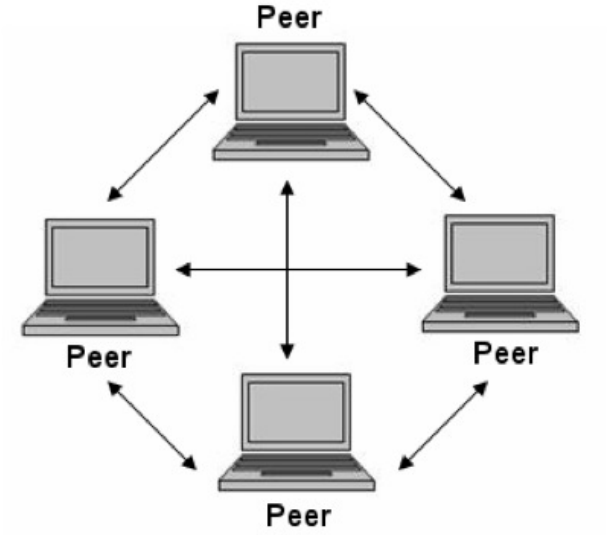
1 Εισαγωγή

Στα πλαίσια της εργασίας αναπτύχθηκε ένα σύστημα ανταλλαγής μηνυμάτων μεταξύ των κόμβων ενός ad hoc δικτύου. Οι κόμβοι του δικτύου είναι ενσωματωμένα συστήματα πραγματικού χρόνου. Κάθε συσκευή ανά τυχαία χρονικά διαστήματα παράγει μηνύματα για κάποιο τελικό παραλήπτη. Ο στόχος είναι τα μηνύματα να διαμοιραστούν μεταξύ των συσκευών έως ότου φτάσουν στον τελικό παραλήπτη.
Αρχικά, θα περιγραφεί η λειτουργία και η δομή του συστήματος που υλοποιήθηκε. Έπειτα, θα αναλυθούν τα αποτελέσματα από το πείραμα που έγινε για να διαπιστωθεί η ορθή λειτουργία της υλοποίησης καθώς και για να μελετηθεί στατιστικά.

2 Περιγραφή υλοποίησης

2.1 Αρχιτεκτονική δικτύου

Η αρχιτεκτονική του δικτύου είναι Peer-to-Peer (P2P). Κάθε κόμβος του δικτύου μπορεί να στείλει αίτημα σε κάποιο άλλο κόμβο για να ανταλλάξουν πληροφορίες καθώς επίσης και να λάβει αίτημα (Σχήμα 1). Με άλλα λόγια, οι κόμβοι του δικτύου μπορούν να λειτουργήσουν και ως πελάτες (clients) αλλά και ως σαν εξυπηρετητές (servers).



Σχήμα 1: Σχηματική απεικόνιση ενός δικτύου Peer-to-Peer

Παρόλο που στην αρχιτεκτονική P2P η έννοια του client και του server δεν υπάρχει, θα θεωρήσουμε καταχρηστικά ότι ο κάθε κόμβος μπορεί να βρεθεί σε 2 καταστάσεις:

- *client mode*: σε αυτή τη λειτουργία ο κόμβος στέλνει αίτημα επικοινωνίας στους άλλους κόμβους του δικτύου.
- *server mode*: σε αυτή τη λειτουργία ο κόμβος περιμένει να δεχτεί αίτημα επικοινωνίας από κάποιον άλλο κόμβο.

2.2 Περιγραφή λειτουργίας

Κάθε συσκευή του δικτύου χαρακτηρίζεται από ένα ID που είναι ένας 4ψήφιος αριθμός. Αν υποθέσουμε ότι $ID = XXYY$, τότε η διεύθυνση IP της στο δίκτυο θα είναι $10.0.XX.YY$.

Σε κάθε συσκευή δίνεται μια λίστα με τα IDs των συσκευών που βρίσκονται στο δίκτυο. Έτσι, κάθε συσκευή γνωρίζει τον αριθμό των υπόλοιπων συσκευών καθώς και τις διευθύνσεις που θα απευθυνθεί για ανταλλαγή μηνυμάτων. Για την επικοινωνία των συσκευών γίνεται χρήση TCP sockets.
Όταν μια συσκευή βρίσκεται σε server mode, τότε αναμένει αιτήματα επικοινωνίας από τις υπόλοιπες συσκευές του δικτύου. Για λόγους ασφαλείας, μόλις λάβει ένα νέο αίτημα από κάποια συσκευή, ελέγχει αν αυτή βρίσκεται στη λίστα με τις διαθέσιμες συσκευές του δικτύου, προκειμένου να συνεχίσει την επικοινωνία. Όταν βρίσκεται σε client mode, στέλνει αυτή αιτήματα στις συσκευές του δικτύου. Έτσι, για να επικοινωνήσουν δύο συσκευές θα πρέπει η μια να βρίσκεται σε client mode και η άλλη σε server mode.
Κάθε συσκευή διατηρεί μια λίστα με μηνύματα που είτε έλαβε είτε δημιούργησε η ίδια. Εκτός από το μήνυμα, αποθηκεύει επίσης σε ποιες συσκευές το έχει ήδη στείλει για να μη γίνουν άσκοπες αποστολές. Αν το μήνυμα το έχει λάβει από κάποια άλλη συσκευή, τότε αποθηκεύει ότι αυτή η συσκευή διαθέτει ήδη το συγκεκριμένο μήνυμα για να μην της το ξαναστείλει σε μελλοντική επικοινωνία. Επίσης, αν μια συσκευή λάβει ένα μήνυμα που έχει ως τελικό παραλήπτη το ID της, τότε δεν συνεχίζει να στέλνει αυτό το μήνυμα στις υπόλοιπες συσκευές καθώς έφτασε στον προορισμό του.
Όταν δυο συσκευές έρθουν σε επικοινωνία, τότε ανταλλάσσουν μηνύματα με την εξής διαδικασία:

- Βήμα 1: Γίνεται χειραψία (handshake) μεταξύ των δύο συσκευών. Για να γίνει η χειραψία με επιτυχία, η συσκευή που βρίσκεται σε client mode στέλνει το μήνυμα START, η συσκευή που βρίσκεται σε server mode το λαμβάνει και απαντάει με το μήνυμα ACK. Ο σκοπός του βήματος αυτού είναι ο συγχρονισμός των συσκευών και η αποφυγή conflicts (περισσότερες λεπτομέρειες στην Παράγραφο 2.4)

- Βήμα 2: Η συσκευή που βρίσκεται σε client mode στέλνει ένα ένα όσα μηνύματα δεν έχει ξαναστείλει στο παρελθόν στη συσκευή που βρίσκεται σε server mode, η δεύτερη τα λαμβάνει και αποθηκεύει στη λίστα όσα δεν υπάρχουν ήδη.

- Βήμα 3: Η συσκευή που βρίσκεται σε server mode στέλνει ένα ένα όσα μηνύματα δεν έχει ξαναστείλει στο παρελθόν στη συσκευή που βρίσκεται σε client mode, η δεύτερη τα λαμβάνει και αποθηκεύει στη λίστα όσα δεν υπάρχουν ήδη.

Σε περίπτωση που η χειραψία (Βήμα 1) μεταξύ των συσκευών αποτύχει, τότε δεν ανταλλάσσονται μηνύματα. Επίσης, κάθε φορά που εμφανίζεται ένα νέο μήνυμα στη λίστα των μηνυμάτων μιας συσκευής, σηκώνεται μια σημαία (`has_new_messages`). Ο αριθμός των σημαιών αυτών είναι ίσος με τον αριθμό των συσκευών στο δίκτυο. Αν, για παράδειγμα εμφανιστεί ένα νέο μήνυμα για τη συσκευή Z, τότε θα ενεργοποιηθεί η σημαία που αντιστοιχεί στη συσκευή Z. Έτσι, όταν μια συσκευή βρίσκεται σε client mode θα επιδιώξει να στείλει αιτήματα επικοινωνίας μόνο στις συσκευές που έχουν ενεργοποιημένη σημαία. Αυτή η ιδέα έχει ως σκοπό τη μείωση των άσκοπων συνδέσεων μεταξύ των συσκευών.

2.3 Χρήση νημάτων

Για την παραλληλοποίηση της υλοποίησης έγιναν οι εξής παρατηρήσεις:

- Η παραγωγή των μηνύματων πρέπει να γίνεται ανεξάρτητα από τις υπόλοιπες λειτουργίες

- Μια συσκευή μπορεί να τρέχει ταυτόχρονα τις λειτουργίες client και server

Σύμφωνα με αυτές τις παρατηρήσεις, δημιουργήθηκαν τα εξής νήματα:

- Master thread: Αυτό είναι το κύριο νήμα του προγράμματος, το οποίο δημιουργεί τα υπόλοιπα. Επίσης, μετράει το χρόνο που περνάει κι όταν ο χρόνος εκτέλεσης του πειράματος παρέλθει, τότε στέλνει ένα αίτημα ακύρωσης (cancellation request) στα υπόλοιπα νήματα.

- Message generator thread: Το νήμα αυτό είναι υπεύθυνο για την παραγωγή μηνυμάτων. Συγκεκριμένα, παράγει μηνύματα προς ένα τυχαίο παραλήπτη και ανά τυχαίο χρονικό διάστημα 1-5 λεπτών.

- Client mode thread: Αναλαμβάνει τη λειτουργία του client mode. Στέλνει αιτήματα στις συσκευές του δικτύου με σκοπό την ανταλλαγή μηνυμάτων (περισσότερες πληροφορίες αναφέρθηκαν στην Παράγραφο 2.2).

- Server mode thread: Αναλαμβάνει τη λειτουργία του server mode. Δέχεται αιτήματα από τις συσκευές του δικτύου με σκοπό την ανταλλαγή μηνυμάτων (περισσότερες πληροφορίες αναφέρθηκαν στην Παράγραφο 2.2).

Όταν ο χρόνος του πειράματος παρέλθει, το master thread αναλαμβάνει να στείλει αίτημα ακύρωσης στα υπόλοιπα νήματα. Για να μην τερματιστεί η ανταλλαγή μηνυμάτων μεταξύ των συσκευών βεβιασμένα, ακολουθείται μια συγκεκριμένη διαδικασία. Συγκεκριμένα, το master thread στέλνει πρώτα αίτημα ακύρωσης στο message generator thread. Έτσι, εξασφαλίζουμε ότι δε θα εμφανιστούν νέα μηνύματα στο δίκτυο. Έπειτα, δίνεται ένα διάστημα αναμονής για να διεκπεραιθούν όλες οι εναπομείνουσες ανταλλαγές μηνυμάτων στο δίκτυο. Το διάστημα αυτό στην υλοποίηση μας έχει οριστεί ως $5N\,sec$, όπου N ο αριθμός των συσκευών στο δίκτυο.

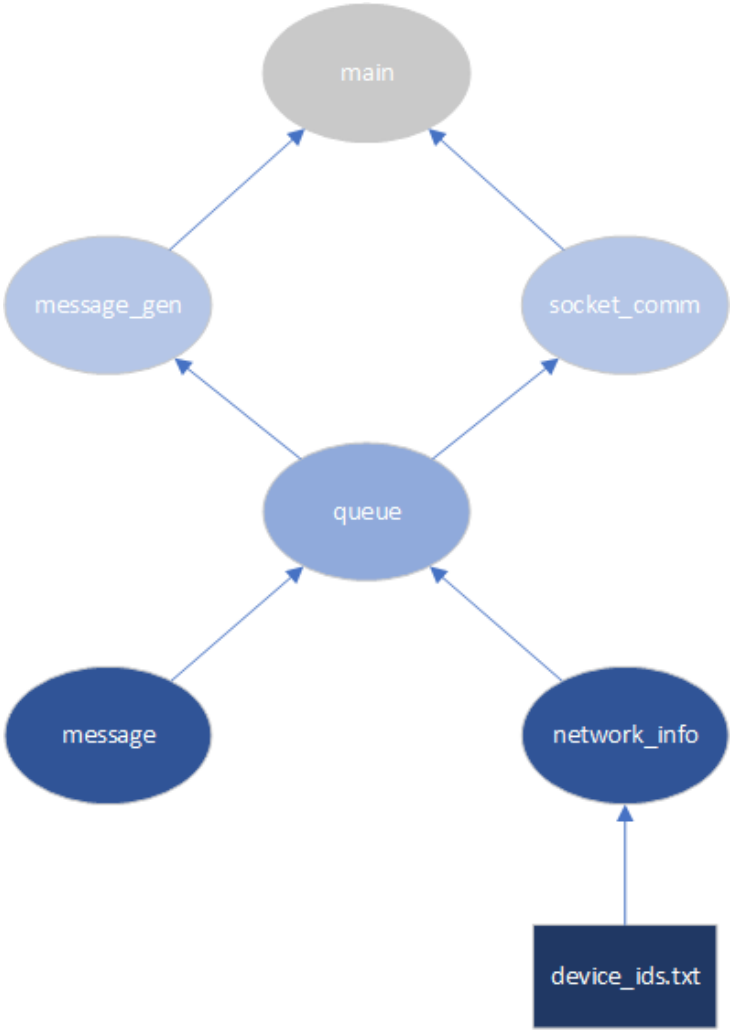
2.4 Αποφυγή conflicts

Έστω ότι δυο συσκευές έχουν συνδεθεί για να ανταλλάξουν μηνύματα και ότι η συσκευή X βρίσκεται σε client mode, ενώ η Y σε server mode. Αν κατά τη διάρκεια της ανταλλαγής, το client mode thread της Y στείλει αίτημα επικοινωνίας στη συσκευή X και το server mode thread της X το δεχτεί, τότε θα προκύψει conflict. Αυτό θα συμβεί γιατί οι δύο συσκευές θα προχωρήσουν σε ανταλλαγή μηνυμάτων, ενώ η ήδη υπάρχουσα είναι σε εξέλιξη. Για να αποφευχθεί αυτό, κάθε φορά που προκύπτει μια νέα σύνδεση με μια συσκευή, ενεργοποιείται μια σημαία (`active_connections`). Για κάθε συσκευή του δικτύου, υπάρχει η αντίστοιχη σημαία. Έτσι, πριν προχωρήσει μια συσκευή σε ανταλλαγή μηνυμάτων ελέγχει πρώτα αν υπάρχει ήδη ενεργή σύνδεση με τη συζευγμένη συσκευή. Επίσης, αν βρίσκεται σε client mode, πριν να σταλεί αίτημα σύνδεσης σε κάποια άλλη συσκευή, γίνεται πρώτα έλεγχος αν υπάρχει ήδη ενεργή σύνδεση με αυτή. Η σημαία αυτή απενεργοποιείται με την περάτωση της ανταλλαγής μηνυμάτων μεταξύ δυο συσκευών.
Ωστόσο, μετά από αυτή την αλλαγή παρατηρήθηκε και μια ακόμη περίπτωση conflict. Έστω ότι το client mode thread της συσκευής X στέλνει αίτημα σύνδεσης στη συσκευή Y. Με πολύ μικρή χρονική διαφορά, το client mode thread της συσκευής Y στέλνει αίτημα σύνδεσης στη συσκευή X. Αν αυτή η χρονική διαφορά είναι πολύ μικρή, έτσι ώστε τα server mode threads των συσκευών να μην έχουν προλάβει να ενημερωθούν για την ύπαρξη ενεργής σύνδεσης, υπάρχει το ενδεχόμενο να δημιουργηθούν δύο ταυτόχρονες συνδέσεις. Το κύριο πρόβλημα, το οποίο διαφοροποιεί αυτή την περίπτωση conflict από την προηγούμενη, είναι ότι οι δύο συσκευές θεωρούν ως ενεργή διαφορετική σύνδεση και συγκεκριμένα αυτή που η καθεμιά τους έχει το ρόλο του client. Για το λόγο αυτό εισάγαμε το βήμα της χειραψίας (handshake) που περιγράψαμε στην Παράγραφο 2.2. Στην ουσία πρόκειται για μια ερωτοαπάντηση για να εξασφαλιστεί ότι στην τρέχουσα σύνδεση η μια συσκευή έχει το ρόλο του client και η άλλη το ρόλο του server. Οι συσκευές συνεχίζουν σε ανταλλαγή μηνυμάτων, μόνο στην

περίπτωση που η χειραψία γίνει επιτυχώς.
Σε κάθε επανάληψη το client mode thread των συσκευών προσπαθεί να συνδεθεί με τις υπόλοιπες συσκευές του δικτύου. Σε περίπτωση που η χειραψία με κάποια αποτύχει, οι δύο συσκευές θα προσπαθήσουν να ξανασυνδεθούν μετά από κάποιο διάστημα. Αν το χρονικό διάστημα αυτό, είναι σχεδόν το ίδιο για τις δύο συσκευές, υπάρχει περίπτωση η χειραψία να αποτύχει ξανά για μια έως και περισσότερες φορές. Για να το αποφύγουμε αυτό, σε κάθε επανάληψη του client mode thread εισάγουμε μια τυχαία καθυστέρηση της τάξεως $0.5-1\,sec$ για να μειώσουμε την πιθανότητα συγχρονισμού των αποπειρών για σύνδεση.

2.5 Υλοποίηση σε κώδικα

Για την υλοποίηση του συστήματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού C. Το σχεδιάγραμμα της δομής του φαίνεται στο Σχήμα 2.



Σχήμα 2: Σχεδιάγραμμα της δομής του συστήματος

Στα αρχεία $message.\{h, c\}$ βρίσκονται οι συναρτήσεις για την αναπαράσταση των μηνυμάτων και κάποιες βοηθητικές συναρτήσεις για τη διαχείριση τους (πχ. σύγκριση αν δυο μηνύματα είναι ίδια). Για την αναπαράσταση ενός μηνύματος χρησιμοποιείται το $struct\,message$. Στα αρχεία $network\_info.\{h, c\}$ υπάρχουν οι συναρτήσεις που παρέχουν όλες τις απαραίτητες πληροφορίες σχετικά με τις συσκευές του δικτύου. Μετά την έναρξη του προγράμματος δημιουργείται ένα $struct\,devices\_in\_network\_info$ που περιέχει πληροφορίες, όπως το συνολικό αριθμό των συσκευών του δικτύου, τα ID τους και τις διευθύνσεις IP. Σε αυτή τη

διαδικασία χρησιμοποιείται και το αρχείο *device_ids.txt*, το οποίο περιέχει όλα τα ID των συσκευών του δικτύου.
Στα αρχεία $queue.\{h, c\}$ υπάρχει η υλοποίηση της λίστας μηνυμάτων. Το μέγιστο μήκος της έχει ρυθμιστεί στα 2000 μηνύματα, αλλά μπορεί να τροποποιηθεί από την παράμετρο $QUEUE\_SIZE$. Στην ουσία πρόκειται για μια αντιστραμμένη ουρά που τα στοιχεία εισέρχονται από μπροστά και σε περίπτωση που γεμίσει διαγράφεται το πρώτο στοιχείο από πίσω. Για την υλοποίηση της έχει χρησιμοποιηθεί συνδεδεμένη λίστα (linked list). Η λίστα μηνυμάτων αντιπροσωπεύεται από το *struct queue* και το κάθε στοιχείο της από το *struct queue_item*.
Η γεννήτρια μηνύματων, την οποία διαχειρίζεται το message generator thread, έχει υλοποιηθεί στα αρχεία $message\_gen.\{h, c\}$. Επίσης, στα $socket\_comm.\{h, c\}$ βρίσκονται όλες οι απαραίτητες συναρτήσεις για την επικοινωνία μέσω sockets, οι οποίες χρησιμοποιούνται από τα client mode και server mode threads. Το *main.c* περιέχει το κυρίως πρόγραμμα, το οποίο δημιουργεί τα νήματα που περιεγράφηκαν. Επίσης, μεταξύ των νημάτων ρυθμίσαμε το message generator thread ως υπεύθυνο για τη διαχείριση του σήματος $SIGALRM$, καθώς το χρησιμοποιεί για την παραγωγή μηνυμάτων.

3 Πείραμα

3.1 Setup

Για το πείραμα χρησιμοποιήθηκε ο παρακάτω εξοπλισμός:

- 2 Raspberry Pi Zero W
- 1 PC Dell με λειτουργικό σύστημα Ubuntu 16.04
- 1 WiFi dongle με Ralink 5370 Chipset[1]

Λόγω μη εύρεσης περισσότερων Raspberry Pi συσκευών, το PC χρησιμοποιήθηκε ως τρίτη συσκευή. Το WiFi dongle αξιοποιήθηκε για την σύνδεση του PC στο ad hoc δίκτυο. Αρχικά, το PC αδυνατούσε να συνδεθεί με τη δική του κάρτα δικτύου αλλά η χρήση του WiFi dongle έλυσε το πρόβλημα.
Στον Πίνακα 1 φαίνονται οι κόμβοι του δικτύου του πειράματος. Εκτός από το $ID = 8448$ που αντιστοιχεί στο AEM του συγγραφέα, η επιλογή των υπόλοιπων IDs είναι **τυχαία**.

Συσκευή	ID	Διεύθυνση IP
Raspberry Pi	8448	10.0.84.48
PC	8449	10.0.84.49
Raspberry Pi	8450	10.0.84.50

Πίνακας 1: Οι κόμβοι του δικτύου του πειράματος

Η μεταγλώττιση των προγραμμάτων για κάθε συσκευή έγινε με χρήση cross-compiler και έπειτα στάλθηκαν στις συσκευές μέσω SCP. Πριν την έναρξη του πειράματος, έγινε προσπάθεια να συγχρονιστούν οι συσκευές στην ίδια μέρα και ώρα. Εκτελώντας την ίδια εντολή σε όλους τους κόμβους του δικτύου μέσω broadcasting καταφέραμε να πετύχουμε ακρίβεια

[1] http://www.grobotronics.com/wifi-dongle-for-raspberry-pi-nano-usb.html

της τάξης των $10^{-2}\,sec$. Με τον ίδιο τρόπο τρέξαμε την εντολή εκκίνησης του προγράμματος σε όλους τους κόμβους του δικτύου. Η διάρκεια του πειράματος ήταν $7200\,sec = 2\,h$.

3.2 Καταγραφή στατιστικών

Κατά τη διάρκεια του πειράματος έγινε καταγραφή κάποιων στατιστικών για να γίνει έλεγχος ορθότητας της υλοποίησης καθώς και για να μελετηθεί στατιστικά. Συγκεκριμένα, κάθε συσκευή κατέγραψε:

- το τελικό περιεχόμενο της λίστας μηνυμάτων της
- τα μηνύματα που παρήγαγε καθώς και τα χρονικά διαστήματα παραγωγής
- πληροφορίες για το κάθε session με άλλη συσκευή, συγκεκριμένα:
  - τη χρονική στιγμή που ξεκίνησε το session
  - με ποια συσκευή συνδέθηκε
  - ποια μηνύματα έστειλε
  - ποια μηνύματα έλαβε κι αν υπήρχαν ήδη στη λίστα της εκείνη τη στιγμή
  - τη χρονική διάρκεια του session

Μέσα στο φάκελο **logs/** του project είναι διαθέσιμα τα στατιστικά που καταγράφηκαν για κάθε συσκευή. Στο αρχείο **queue_<ID>.txt** βρίσκεται η τελική μορφή της λίστας μηνυμάτων, στο **msg_creation_timestamps_<ID>.txt** οι πληροφορίες για τα παραχθέντα μηνύματα και στο **sessions_<ID>.json** το ιστορικό των sessions.

3.3 Έλεγχος ορθότητας

Η ορθότητα της υλοποίησης ελέγχθηκε με 4 διαφορετικούς τρόπους, οι οποίοι παρουσιάζονται στη συνέχεια. Οι έλεγχοι έγιναν πάνω στα στατιστικά που συλλέχθηκαν με χρήση των Python scripts που βρίσκονται στο φάκελο **data_statistics/** του project. Η υλοποίηση ανταποκρίθηκε θετικά σε όλους τους ελέγχους ορθότητας.

Πρώτος Έλεγχος:
Έγινε έλεγχος αν όλα τα μηνύματα έφτασαν στον τελικό τους προορισμό. Για να το κάνουμε αυτό, είδαμε αν τα παραχθέντα μηνύματα κάθε συσκευής βρίσκονται στη λίστα μηνυμάτων του τελικού τους παραλήπτη.

Δεύτερος Έλεγχος:
Έγινε έλεγχος της τελικής λίστας μηνυμάτων κάθε συσκευής για να βεβαιωθούμε ότι δεν περιέχουν διπλότυπα μηνύματα.

Τρίτος Έλεγχος:
Έγινε έλεγχος αν το ιστορικό των sessions κάθε συσκευής συνάδει με των υπολοίπων. Πιο συγκεκριμένα, έστω δυο συσκευές του δικτύου X και Y. Για κάθε session μεταξύ των δύο συσκευών, τα μηνύματα που έστειλε η συσκευή X θα πρέπει να συμπίπτουν με τα μηνύματα που έλαβε η Y και το αντίστροφο.

Τέταρτος Έλεγχος:
Έγινε έλεγχος της στατιστικής συμπεριφοράς του συστήματος, ο οποίος παρουσιάζεται αναλυτικά στην επόμενη παράγραφο.

3.4 Παρουσίαση στατιστικών

Στον Πίνακα 2 βλέπουμε κάποια στατιστικά σχετικά με τις ανταλλαγές μηνυμάτων κατά τη διάρκεια των sessions.

	ID Συσκευής		
	8448	**8449**	**8450**
Συνολικός αριθμός sessions	243	242	241
Συνολικά μηνύματα στη λίστα	124	124	124
Μηνύματα που παρήγαγε	42	40	42
Μηνύματα που έστειλε	117	126	129
Μηνύματα που έλαβε	131	122	119
Μηνύματα που έλαβε και είχε ήδη στη λίστα	49	38	37
Μέσος αριθμός απεσταλμένων μηνυμάτων / session	0.48	0.52	0.54
Μέσος αριθμός ληφθέντων μηνυμάτων / session	0.54	0.50	0.49

Πίνακας 2: Στατιστικά των μηνύματων που ανταλλάχθηκαν για κάθε συσκευή

Όπως βλέπουμε στην τελική λίστα μηνυμάτων κάθε συσκευής βρίσκεται ο ίδιος αριθμός μηνυμάτων (124). Ο αριθμός αυτός ισούται με το άθροισμα των μηνυμάτων που παράχθηκαν συνολικά από όλες τις συσκευές, το οποίο δείχνει τη σωστή λειτουργία του συστήματος. Ένας ακόμα έλεγχος ορθότητας που μπορεί να γίνει είναι ο εξής:

$$\begin{aligned}\text{Συνολικά μηνύματα στη λίστα} = &\ (\text{Μηνύματα που παρήγαγε}) \\ &+ (\text{Μηνύματα που έλαβε}) \\ &- (\text{Μηνύματα που έλαβε και είχε ήδη στη λίστα})\end{aligned}$$

Αυτή η εξίσωση ικανοποιείται και για τις 3 συσκευές. Επίσης, οι αριθμοί των απεσταλμένων/ληφθέντων μηνυμάτων είναι κοντά μεταξύ τους για τις διάφορες συσκευές, το οποίο δείχνει την ισοτιμία των κόμβων στο δίκτυο. Αυτό φαίνεται και στον Πίνακα 3, όπου οι αριθμοί των sessions μεταξύ των συσκευών έχουν ελάχιστη διαφορά.

ID Συσκευής	8448	8449	8450
8448	-	122	121
8449	122	-	120
8450	121	120	-
Σύνολο	243	242	241

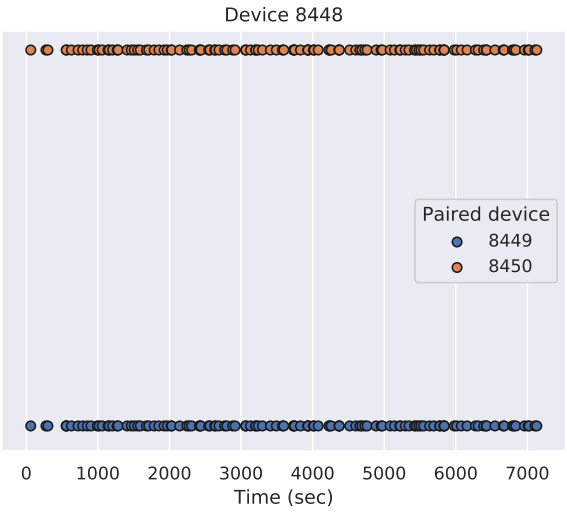
Πίνακας 3: Αριθμός των sessions μεταξύ των συσκευών

Αυτή η συμπεριφορά είναι επιθυμητή και επιβεβαιώνει τη λογική της υλοποίησης. Όταν μια συσκευή παράγει ένα μήνυμα, το client mode thread της θα το στείλει στις άλλες δυο συσκευές δημιουργώντας δύο sessions το ένα μετά το άλλο. Από την άλλη, όταν λάβει ένα

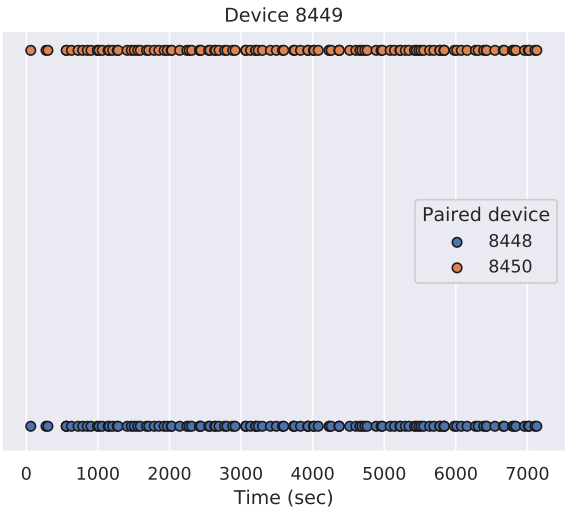
μήνυμα μέσω του server mode thread της, θα ακολουθήσει ένα ακόμα session για να το στείλει στην άλλη συσκευή. Βέβαια, υπάρχει και περίπτωση να μη συνεχίσει τη μετάδοση, αν είναι ο τελικός παραλήπτης του μηνύματος. Και σε αυτή την περίπτωση, όμως, θα επικοινωνήσει μαζί της η άλλη συσκευή μόλις ενημερωθεί για το μήνυμα. Δεδομένου ότι η ίδια πιθανότητα αποστολής και λήψης ισχύει για όλες τις συσκευές αφού είναι ισότιμοι κόμβοι, είναι αναμενόμενο ο αριθμός των sessions μεταξύ των συσκευών να παρουσιάζει μικρές αποκλίσεις.

Ενδιαφέρον παρουσιάζουν και οι μέσοι αριθμοί απεσταλμένων και ληφθέντων μηνυμάτων ανά session (≈ 0.50) του Πίνακα 2. Συγκεκριμένα, μαρτυρούν ότι στη μέση περίπτωση θα σταλεί ή θα ληφθεί ένα μήνυμα ανά 2 sessions, το οποίο ταιριάζει με την εξήγηση που δώσαμε προηγουμένως.

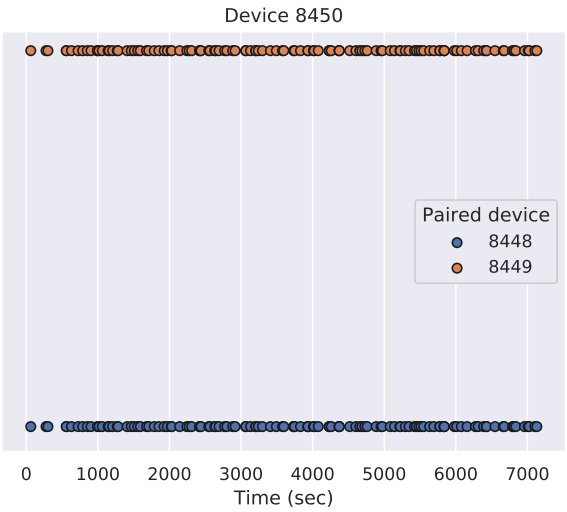
Επομένως, όταν μια συσκευή πραγματοποιεί session με κάποια άλλη συσκευή, στην πλειοψηφία των περιπτώσεων αναμένουμε να ακολουθήσει ένα ακόμα session και με την τρίτη. Στο Σχήμα 3 φαίνονται οι χρονικές στιγμές που ξεκίνησε κάποιο session κατά τη διάρκεια του πειράματος για κάθε συσκευή. Σε κάθε σχεδιάγραμμα, βλέπουμε πως οι χρονικές στιγμές έναρξης των sessions με τη μια συσκευή ακολουθούν της άλλης (οι μπλε κουκκίδες ακολουθούν τις πορτοκαλί και το αντίστροφο), επιβεβαιώνοντας τις παραπάνω παρατηρήσεις.



(α′) Συσκευή 8448



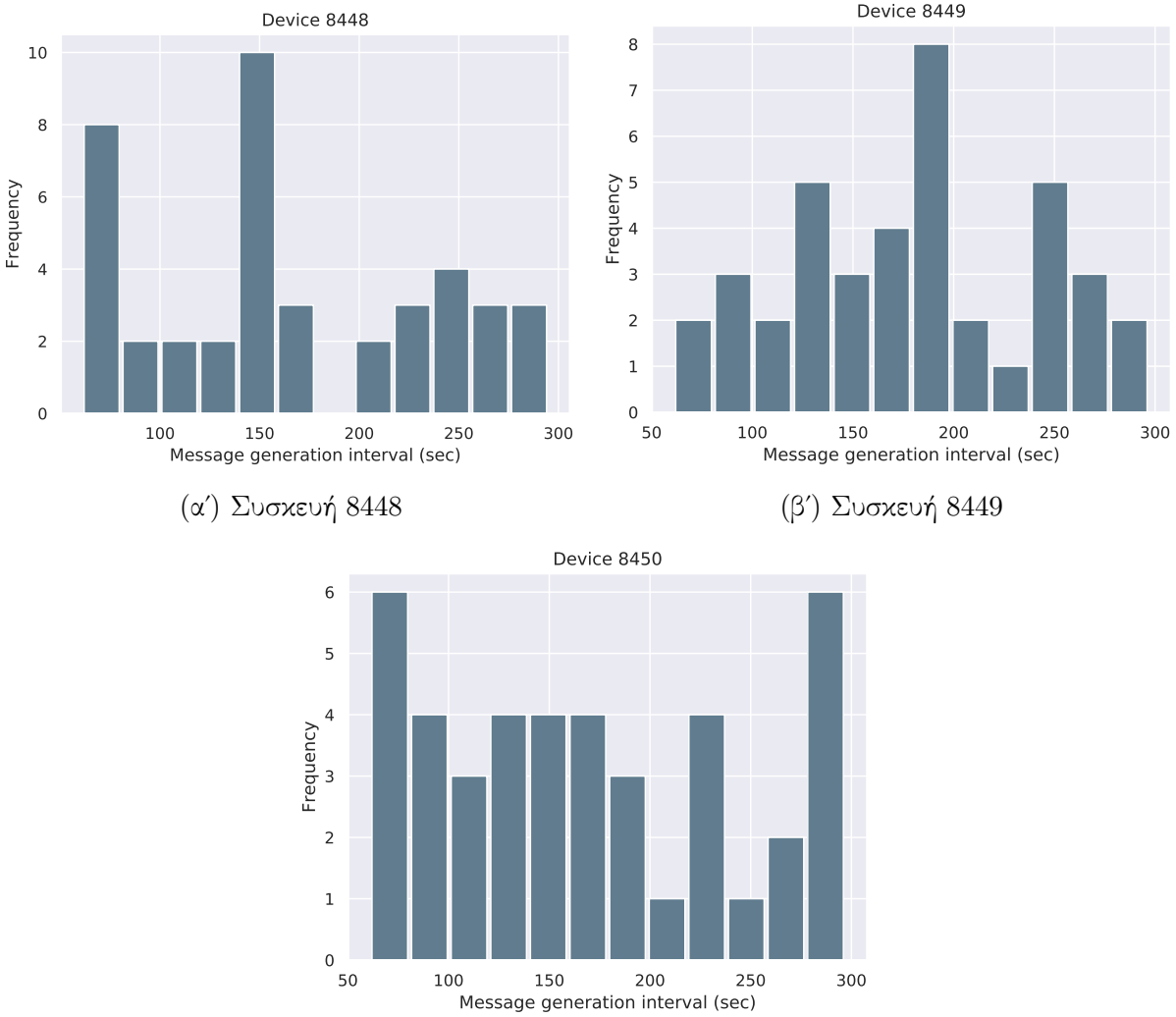
(β′) Συσκευή 8449



(γ′) Συσκευή 8450

Σχήμα 3: Χρονικές στιγμές έναρξης των sessions για κάθε συσκευή

Στη συνέχεια θα εξετάσουμε τη λειτουργία της παραγωγής μηνυμάτων για κάθε συσκευή. Στο Σχήμα 4 φαίνεται η κατανομή των χρονικών διαστημάτων παραγωγής μηνύματος για κάθε συσκευή. Όπως παρατηρούμε, τα χρονικά αυτά διαστήματα ανήκουν στο εύρος $[60, 300]\, sec$ ή $[1, 5]\, min$, όπως είναι το ζητούμενο.



(α′) Συσκευή 8448

(β′) Συσκευή 8449

(γ′) Συσκευή 8450

Σχήμα 4: Η κατανομή των χρονικών διαστημάτων παραγωγής μηνυμάτων για κάθε συσκευή

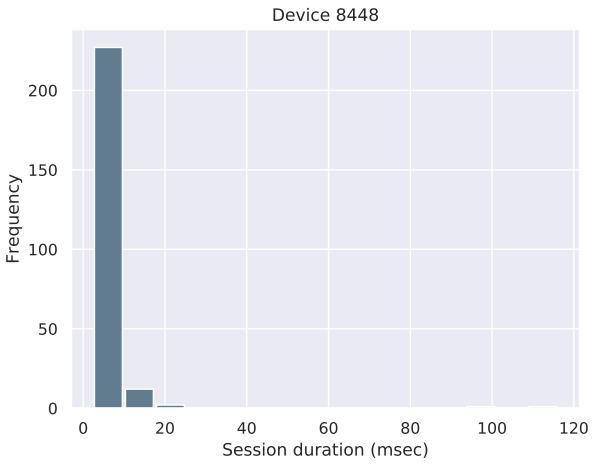
Στον Πίνακα 4 βλέπουμε τα στατιστικά της παραγωγής μηνυμάτων με βάση τις κατανομές.

ID Συσκευής	Μέση τιμή	Τυπική απόκλιση	Ελάχιστο	Μέγιστο
8448	165.69	72.18	61.00	295.00
8449	178.28	61.84	61.00	297.00
8450	169.17	74.74	61.00	297.00

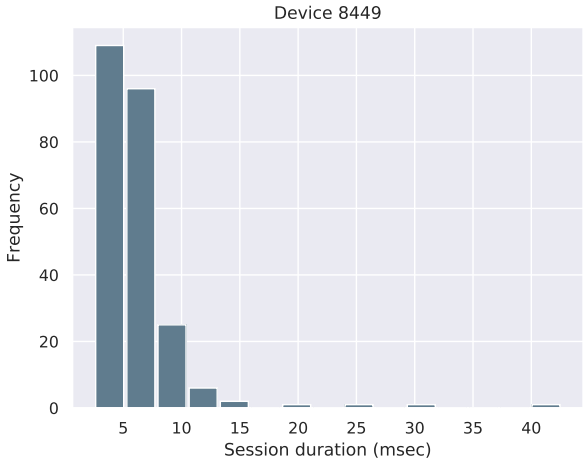
Πίνακας 4: Στατιστικά της παραγωγής μηνυμάτων για κάθε συσκευή (εκφρασμένα σε sec)

Ιδιαίτερη σημασία, επίσης, παρουσιάζει η διάρκεια των sessions, η οποία είναι επιθυμητό να βρίσκεται σε χαμηλά επίπεδα. Στο Σχήμα 5 φαίνεται η κατανομή της διάρκειας των sessions για κάθε συσκευή. Όπως γίνεται εύκολα αντιληπτό, στη συντριπτική πλειοψηφία τα sessions διαρκούν λιγότερο από $10\, msec$ για όλες τις συσκευές. Αυτό φαίνεται και στον Πίνακα 5,

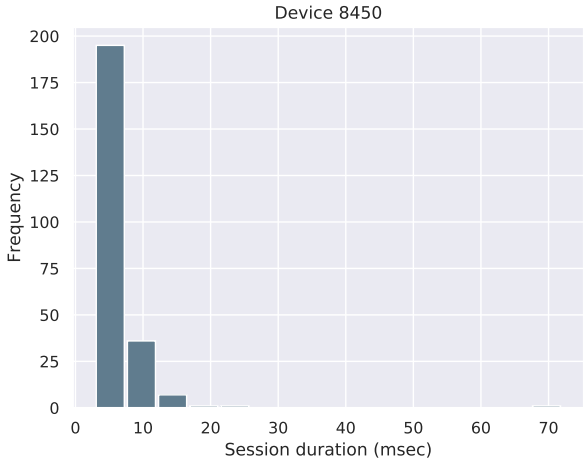
όπου είναι υπολογισμένα κάποια στατιστικά με βάση τις κατανομές. Η μέση τιμή και η τυπική απόκλιση λαμβάνουν ικανοποιητικά μικρές τιμές. Βέβαια, υπάρχουν και κάποιες περιπτώσεις που η διάρκεια των sessions θα ανέβει, αλλά αυτές είναι ελάχιστες. Οι μέγιστες τιμές για κάθε συσκευή βρίσκονται σε λογικά πλαίσια. Την πιο σταθερή συμπεριφορά παρουσιάζει η συσκευή με $ID = 8449$. Αυτό δεν θα πρέπει να μας προκαλεί εντύπωση καθώς για αυτόν τον κόμβο του δικτύου χρησιμοποιήθηκε υπολογιστής με περισσότερες δυνατότητες από ένα Raspberry Pi, όπως αναφέραμε στην Παράγραφο 3.1.



(α′) Συσκευή 8448



(β′) Συσκευή 8449



(γ′) Συσκευή 8450

Σχήμα 5: Η κατανομή της χρονικής διάρκειας των sessions για κάθε συσκευή

ID Συσκευής	Μέση τιμή	Τυπική απόκλιση	Ελάχιστο	Μέγιστο
8448	6.45	9.61	2.37	116.16
8449	5.75	3.90	2.50	42.59
8450	6.20	5.17	2.85	72.01

Πίνακας 5: Στατιστικά της διάρκειας των sessions για κάθε συσκευή (εκφρασμένα σε $msec$)