ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΔΙΚΤΥΑ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ-ΠΡΩΤΟ ΠΑΡΑΔΟΤΕΟ 2018-2019

KEEP
CALM
IT WORKS
ON
MY MACHINE

ΟΜΑΔΑ 4

Ον/μο: Σωκράτης Μπέης **ΑΜ:** 1115201300113

Ον/μο: Συμέλα-Φωτεινή Κομίνη **ΑΜ:** 1115201400072

Ον/μο: Ευάγγελος Τζιρώνης **ΑΜ:** 1115201400199

Ον/μο: Ιωάννης Μανωλάκης **ΑΜ:** 1115201500088

EDGE SERVER

Στον **edge server** τρέχουν τέσσερα threads:

- Ένα **main** thread: **Edge.java**
- Τρία threads:
  - **MamaThread.java**, διαχειρίζεται τα μηνύματα που στέλνονται από τα android terminals στον MQTT broker,
  - **AndroidThread.java**, στέλνει τα μηνύματα στα android terminals και
  - **BackhaulThread.java**, στέλνει στον Backhaul server τα δεδομένα που θα εισάγει στη βάση.

Τα τρία τελευταία threads ανταλλάσουν δεδομένα με τη χρήση 2 **ConcurrentLinkedQueue<QueueBufferNode>** buffers. Η παραπάνω επιλογή έγινε γιατί τα ConcurrentLinkedQueue<> είναι thread-safe και διαχειρίζονται των συγχρονισμό των read και writes στην κοινή μνήμη, εσωτερικά. Και οι δύο buffers έχουν τύπου **QueueBufferNode** δεδομένα.

Το **QueueBufferNode class** αποτελείται από τα παρακάτω πεδία:

- **private int mobile, m1, m2:** id terminal που έστειλε το μήνυμα, μήνυμα για criticality level(0 ή 1), μήνυμα για criticality level(0 ή 2), αντίστοιχα.
- **private String DB_message:** Το μήνυμα που θα στείλει το BackhaulThread στον Backhaul server.
- **private MqttClient mqttClient:** Ο mqttclient στον οποίο κάνει subscribe το MamaThread και publish το AndroidThread.

MamaThread.java

Κάθε φορά που λαμβάνει ένα μήνυμα από το android διαχωρίζει το μήνυμα στα επιμέρους στοιχεία και καλεί τις συναρτήσεις **CreateTestCSV** και **knn_classifier.**

- **CreateTestCSV()**

  Δημιουργεί ένα csv αρχείο για το συγκεκριμένο test csv, το μετατρέπει σε ένα αντικείμενο Experiment και το επιστρέφει, όπως και στον Backhaul.

- **knn_classifier(Experiment[] training_set,**

  Experiment test,

  String[] labels,

  int K)

  Παίρνει ως ορίσματα το training set από το πρώτο παραδοτέο, το δημιουργημένο test set από την CreateTestCSV() κάθε φορά, έναν πίνακα με 2 strings για τα labels και το K που δίνεται παραπάνω.

Εκτελεί τα βήματα της παραλλαγής του knn, σύμφωνα με την παρουσίαση, ταξινομεί τις αποστάσεις με αύξουσα σειρά ώστε να πάρουμε τις k μικρότερες για το σετ I, ελέγχοντας παράλληλα και για διπλότυπες τιμές. Έπειτα, υπολογίζει τα βάρη και επιστρέφει το **label της κλάσης** στην οποία ταξινομήθηκε το σετ (**“Eyes_opened”** / **“Eyes_closed”**).

Στο τέλος της προσωμοίωσης, εκτυπώνονται **στατιστικά για την απόδοση του knn**.

Ενδεικτικά για 30 μηνύματα:

Για **K=3** → απόδοση = **50%**

Για **K=5** → απόδοση = **73.33%**

Για **K=7** → απόδοση = **83.33%**

Για **K=9** → απόδοση = **80%**

Για **K=11** → απόδοση = **66.6%**

Μόλις λάβουμε το αποτέλεσμα του knn υπολογίζουμε το **criticality level** βάσει του παραπάνω και την απόσταση μεταξύ των κινητών, σύμφωνα με την εκφώνηση.

Για κάθε criticality level που προκύπτει, προσθέτει στο τέλος των 2 buffer(qa,qb), ένα νέο QueueBufferNode.

AndroidThread.java

Κάθε φορά που εισάγεται ένα νέο στοιχείο στο buffer, αφαιρεί το πρώτο στοιχείο, εξετάζει το criticality level του μηνύματος και αναλόγως στέλνει το κατάλληλο μήνυμα είτε στο ένα είτε και στα δύο τερματικά.

BackhaulThread.java

Κάθε φορά που εισάγεται ένα νέο στοιχείο στο buffer, αφαιρεί το πρώτο στοιχείο και στέλνει το κατάλληλο μήνυμα στον Backhaul server.

BACKHAUL SERVER

DatabaseThread.java

Διαβάζονται τα μηνύματα από τον buffer, διαχωρίζονται τα πεδία με βάση το κόμμα και μεταβιβάζονται στην βάση με ένα query INSERT INTO.

ΣΥΝΘΗΚΗ ΤΕΡΜΑΤΙΣΜΟΥ

Ο edge server τερματίζει μετά από έναν αριθμό μηνυμάτων(πχ 30) και ενημερώνει όλα τα threads του να τερματίσουν και τον Backhaul server ότι αποσυνδέεται.