



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

**Σύστημα παρακολούθησης δικτυακής καθυστέρησης
μεταφοράς δεδομένων σε περιβάλλοντα υπολογισμού ακμής**

Πτυχιακή εργασία

Μαριάννα Στεφάνοβα

Αθήνα, 2022



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT OF INFORMATICS AND TELEMATICS

**Network latency monitoring in edge computing
environments**

Graduate thesis

Mariana Stefanova

Athens, 2022



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

**ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ**

Τριμελής Εξεταστική Επιτροπή

Τσερπές Κωνσταντίνος (Επιβλέπων)

Αναπληρωτής Καθηγητής, Τμήμα Πληροφορική και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Κουσιουρής Γεώργιος

Επίκουρος Καθηγητής, Τμήμα Πληροφορική και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Καμαλάκης Θωμάς

Καθηγητής, Τμήμα Πληροφορική και Τηλεματικής, Χαροκόπειο
Πανεπιστήμιο

Η Μαριάννα Στεφάνοβα

δηλώνω υπεύθυνα ότι:

- 1)** Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλλει τα πνευματικά δικαιώματα τρίτων.
- 2)** Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.
- 3)** Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να αποδώσω τις ευχαριστίες μου, κυρίως, στον επιβλέποντα καθηγητή της πτυχιακής αυτής εργασίας, κύριο Τσερπέ Κωνσταντίνο, για την πολύτιμη βοήθεια και καθοδήγηση που μου προσέφερε, συμβάλλοντας ενεργά στην ολοκλήρωσή της.

Ακόμη, θα ήθελα να ευχαριστήρω την οικογένειά μου, που με βοήθησε με κάθε δυνατό τρόπο, σε όλη τη διάρκεια των σπουδών μου, και ειδικά κατά τη διάρκεια εκπόνησης της πτυχιακής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους συμφοιτητές και φίλους μου για την υποστήρηξή τους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη	9
Abstract	10
Κατάλογος Εικόνων	11
Συντομογραφίες	12
1. Εισαγωγή	14
1.1 Κίνητρο.....	14
1.2 Σκοπός.....	15
1.3 Χάρτης κειμένου.....	16
2. Βασικές έννοιες και Τρέχουσα κατάσταση	17
2.1 Διαδίκτυο των Πραγμάτων.....	17
2.1.1 Ορισμός.....	17
2.1.2 Ιστορία.....	17
2.1.3 Λειτουργία.....	18
2.2 Υπολογιστικό νέφος.....	19
2.2.1 Ορισμός.....	19
2.2.2 Ιστορία.....	20
2.2.3 Αρχιτεκτονική υπολογιστικού νέφους.....	20
2.2.4 Τύποι υπολογιστικού νέφους.....	22
2.2.5 Υπηρεσίες υπολογιστικού νέφους.....	23
2.2.6 Πλεονεκτήματα.....	23
2.2.7 Μειονεκτήματα.....	24
2.3 Καθυστέρηση μεταφοράς δεδομένων.....	25
2.3.1 Ορισμός.....	25
2.3.2 Αιτίες καθυστέρησης.....	25
2.3.3 Μέτρηση καθυστέρησης.....	26
2.4 Περιβάλλον υπολογισμού ακμής.....	30

2.4.1 Ορισμός.....	30
2.4.2 Αρχιτεκτονική.....	32
2.4.3 Λειτουργία.....	32
2.4.4 Πλεονεκτήματα.....	33
2.4.5 Μειονεκτήματα.....	34
2.4.6 Χρήση.....	35
2.5 Τεχνολογίες ανάπτυξης συστήματος.....	36
2.5.1 Πρωτόκολλο HTTP.....	36
2.5.2 WebSockets.....	38
2.5.3 Redis.....	39
2.5.4 Android.....	39
3. Υλοποίηση.....	41
3.1 Αρχιτεκτονική.....	41
3.2 Gateway.....	42
3.3 Echoserver.....	42
3.4 Probe.....	43
3.4.1 MainActivity.java.....	43
3.4.2 PingingService.java.....	44
3.4.3 ServiceReceiver.java.....	44
3.5 Κώδικας.....	45
4. Αξιολόγηση.....	46
4.1 Ping των Echoservers.....	46
4.2 Μεταβολή του latency.....	47
4.3 Αντικατάσταση κόμβου ακμής.....	48
5. Συμπεράσματα.....	50
6. Βιβλιογραφία.....	52

Περίληψη

Στις ημέρες μας, ολοένα και περισσότερες συσκευές καθημερινής χρήσης συνδέονται μεταξύ τους, με σκοπό την ανταλλαγή δεδομένων, δημιουργώντας το Διαδίκτυο των Πραγμάτων (Internet of Things). Τα Πράγματα αυτά μπορεί να έχουν αισθητήρες, λογισμικό ή συνδυασμό των δύο αυτών στοιχείων. Τα Πράγματα αυτά έχουν ως στόχο τη συλλογή δεδομένων από το περιβάλλον τους με ελάχιστη ανθρώπινη παρέμβαση. Η αποθήκευση και επεξεργασία αυτών των δεδομένων γίνεται μέσω του Υπολογιστικού Νέφους (Cloud Computing). Ο αριθμός των Πραγμάτων αυξάνεται με πολύ γρήγορους ρυθμούς. Αυτό σημαίνει πως η μεταφορά των δεδομένων στο Υπολογιστικό Νέφος μπορεί να έχει μεγάλες καθυστερήσεις. Υπάρχουν, όμως, περιπτώσεις στις οποίες απαιτείται οι υπολογισμοί να γίνουν γρήγορα, και οι καθυστερήσεις αυτές μπορεί να αποτελέσουν αποτρεπτικό παράγοντα για τη χρήση του Υπολογιστικού Νέφους. Για το λόγο αυτό, εξετάζονται άλλα υπολογιστικά μοντέλα, όπως το Περιβάλλον Υπολογισμού Ακμής (Edge Computing). Σε αυτό το μοντέλο, η επεξεργασία και η αποθήκευση των δεδομένων γίνεται πιο κοντά στις πηγές των δεδομένων, στην ακμή του δικτύου. Για αυτές τις περιπτώσεις, στις οποίες η ταχύτητα είναι ζητούμενο, θα ήταν χρήσιμη η ύπαρξη ενός συστήματος που να μετρά το χρόνο που χρειάζεται για την μεταφορά των δεδομένων από το Πράγμα ως την κάθε Ακμή. Έτσι, θα μπορεί, κάθε φορά, να επιλέγεται από το Πράγμα, η Ακμή με την οποία συμβαίνει η μικρότερη καθυστέρηση. Έτσι, θα μπορεί να επιτυγχάνεται ο μικρότερος δυνατός χρόνος, κάθε φορά που πρέπει να γίνει ένα υπολογισμός. Αυτός είναι και ο στόχος της παρούσας εργασίας, η υλοποίηση ενός συστήματος παρακολούθησης της δικτυακής καθυστέρησης της μεταφοράς δεδομένων σε περιβάλλοντα υπολογισμού ακμής.

Λέξεις κλειδιά: Διαδίκτυο των Πραγμάτων, Υπολογιστικό Νέφος, Περιβάλλον Υπολογισμού Ακμής, Καθυστέρηση

Abstract

Nowadays, more and more devices of everyday use are connected to each other and their purpose is to exchange data, thus creating the Internet of Things. These Things could have sensors, software or a combination of these two components. These Things serve the purpose of collecting data from their environment with minimal human intervention. Cloud Computing is used for storing and processing this data. The number of Things is increasing at a very fast pace. This means that data transfer to the Cloud may result in big delays. However, there are cases that demand processing to happen very fast and these delays are a preventive factor for the usage of Cloud Computing. For this reason, other computing paradigms are being examined, such as Edge Computing. In this paradigm, data storing and processing happens closer to the source of the data, at the edge of the network. For these cases in which speed is required, it would be useful to have a system to measure the time that is needed for data transfer from the Thing to each Edge. In this way, every time, the Thing will choose the Edge that had the smallest latency. The smallest possible time will be achieved every time that a process needs to be done. This is the purpose of the present thesis, the implementation of a network latency monitoring system in an edge computing environment.

Keywords: Internet of Things, Cloud Computing, Edge Computing, Latency

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Πώς λειτουργεί το IoT	18
Εικόνα 2: Η αρχιτεκτονική του υπολογιστικού νέφους.....	21
Εικόνα 3: Η λειτουργία του ICMP.....	27
Εικόνα 4: Τα αποτελέσματα του ping.....	28
Εικόνα 5: Η λειτουργία του traceroute.....	29
Εικόνα 6: Τα αποτελέσματα του traceroute.....	30
Εικόνα 7: Πώς λειτουργεί το Edge Computing.....	31
Εικόνα 8: Πώς λειτουργεί το πρωτόκολλο HTTP.....	37
Εικόνα 9: Η λειτουργία των WebSockets.....	38
Εικόνα 10: Η αρχιτεκτονική του συστήματος.....	41
Εικόνα 11: Οι Echoservers.....	46
Εικόνα 12: Το Probe.....	47
Εικόνα 13: Τα περιεχόμενα του Probe.....	47
Εικόνα 14: Η δεύτερη μέτρηση.....	48
Εικόνα 15: Η τέταρτη μέτρηση.....	49

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

IoT	Internet of Things
RFID	Radio Frequency Identification
IP	Internet Protocol
IPv6	Internet Protocol version 6
Wi-Fi	Wireless Fidelity
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
ms	milliseconds
ICMP	Internet Control Message Protocol
RTT	Round Trip Time
ttl	time to live
TCP	Transmission Control Protocol
HTTP	HyperText Transfer Protocol
SDK	Software Development Kit
IDE	Integrated Development Enviroment

1. ΕΙΣΑΓΩΓΗ

1.1 Κίνητρο

Η τεχνολογία αναπτύσσεται και εξελίσσεται με γρήγορους ρυθμούς και η χρήση της είναι πλέον δεδομένη στην καθημερινότητα κάθε ανθρώπου. Ολοένα και περισσότερες συσκευές αποκτούν τη δυνατότητα να συνδέονται στο διαδίκτυο και να μοιράζονται πληροφορίες μεταξύ τους. Οι συσκευές αυτές γίνονται μέρος ενός δικτύου, του Διαδικτύου των Αντικειμένων (Internet of Things). Η εργασία, που καλούνται να εκτελέσουν οι συσκευές αυτές, είναι να συλλέξουν δεδομένα από το περιβάλλον τους, τα οποία στη συνέχεια χρησιμοποιούνται για την εκτέλεση διάφορων υπολογισμών. Συνήθως, τα δεδομένα αυτά μεταφέρονται στο υπολογιστικό νέφος, όπου γίνεται η αποθήκευση και επεξεργασία τους. Στη συνέχεια, τα αποτελέσματα της επεξεργασίας αποστέλλονται από το υπολογιστικό νέφος, ώστε αυτές να συνεχίσουν την εκτέλεση των εργασιών του. Όμως, οι συσκευές αυτές συνεχώς αυξάνονται σε αριθμό. Αυτό έχει ως αποτέλεσμα η μεταφορά των δεδομένων από και προς το υπολογιστικό νέφος να εισάγει μεγάλες καθυστερήσεις. Οι καθυστερήσεις αυτές μπορεί να αποβούν καταστροφικές, όταν πρόκειται για εφαρμογές στις οποίες είναι απαραίτητη η απόκριση σε πραγματικό χρόνο.

Προκειμένου να μειωθεί η καθυστέρηση, που προκύπτει από τη μεταφορά των δεδομένων από και προς το υπολογιστικό νέφος, μία καλή ιδέα είναι να χρησιμοποιηθεί κάποιο άλλο υπολογιστικό μοντέλο. Το Περιβάλλον Υπολογισμού Ακμής (Edge Computing) φαίνεται να είναι κατάλληλο για αυτές τις περιπτώσεις. Ο υπολογισμός και η αποθήκευση των

δεδομένων δεν γίνεται σε κάποιο κεντρικό υπολογιστικό σύστημα, αλλά συμβαίνει στην ακμή του δικτύου, κοντά στη φυσική τοποθεσία από όπου συλλέχθηκαν τα δεδομένα, στα κατά τόπους Κέντρα Δεδομένων Ακμής (Edge Datacenters) που υπάρχουν. Η κάθε συσκευή αποστέλλει τα δεδομένα στον Κόμβο Ακμής που βρίσκεται πιο κοντά της.

Ακόμη, πολλές από τις σύγχρονες συσκευές έχουν δυνατότητα μετακίνησης, δηλαδή δεν έχουν μία σταθερή τοποθεσία. Στην πράξη, αυτό σημαίνει πως η κάθε συσκευή πιθανόν να μη γνωρίζει ποιο είναι το ποιο κοντινό της Κέντρο Δεδομένων Ακμής. Για το λόγο αυτό, θα ήταν χρήσιμη η ύπαρξη ενός συστήματος που να μπορεί να κατευθύνει την κάθε συσκευή στον ποιο κοντινό για αυτήν Κόμβο Ακμής την εκάστοτε χρονική στιγμή.

1.2 Σκοπός

Σκοπός αυτής της πτυχιακής εργασίας είναι η υλοποίηση ενός συστήματος που αποτελείται από 3 συστατικά στοιχεία, το Gateway, τους Echoservers και το Probe. Το Gateway έχει το ρόλο ενός κεντρικού συστήματος αποθήκευσης δεδομένων και αποθηκεύει τις IP διευθύνσεις όλων των Echoservers. Οι Echoservers αντιπροσωπεύουν τους Κόμβους Ακμής. Το Probe είναι μία android εφαρμογή, της οποίας η δουλειά είναι ανά διαστήματα να κάνει ring τους Echoservers και να καταγράφει το χρόνο που χρειάστηκαν για να απαντήσουν. Είναι προφανές πως ο Echoserver με τον μικρότερο χρόνο, θα είναι και ο πιο κοντινός. Ακόμη, το σύστημα αυτό θα είναι χρήσιμο για να παρατηρήσουμε εάν αυτός ο χρόνος αλλάζει από τη μία μέτρηση στην άλλη. Τέλος, θα μας δίνει τη δυνατότητα να δούμε εάν έχουμε αλλαγές στον κοντινότερο Echoserver στην κάθε μέτρηση.

1.3 Χάρτης Κειμένου

Στο δεύτερο κεφάλαιο παρουσιάζονται κάποιες βασικές έννοιες που είναι απαραίτητες για την κατανόηση της παρούσας εργασίας. Στο τρίτο κεφάλαιο γίνεται παρουσίαση της δομής του συστήματος και περιγράφεται ο τρόπος με τον οποίο υλοποιήθηκαν τα επιμέρους στοιχεία. Στο τέταρτο κεφάλαιο γίνεται μία αξιολόγηση του συστήματος σύμφωνα με τους αρχικούς στόχους. Στο πέμπτο κεφάλαιο αναφέρονται παρατηρήσεις και συμπεράσματα σχετικά με το σύστημα.

2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΚΑΙ ΤΡΕΧΟΥΣΑ ΚΑΤΑΣΤΑΣΗ

2.1 Διαδίκτυο των Πραγμάτων

2.1.1 Ορισμός

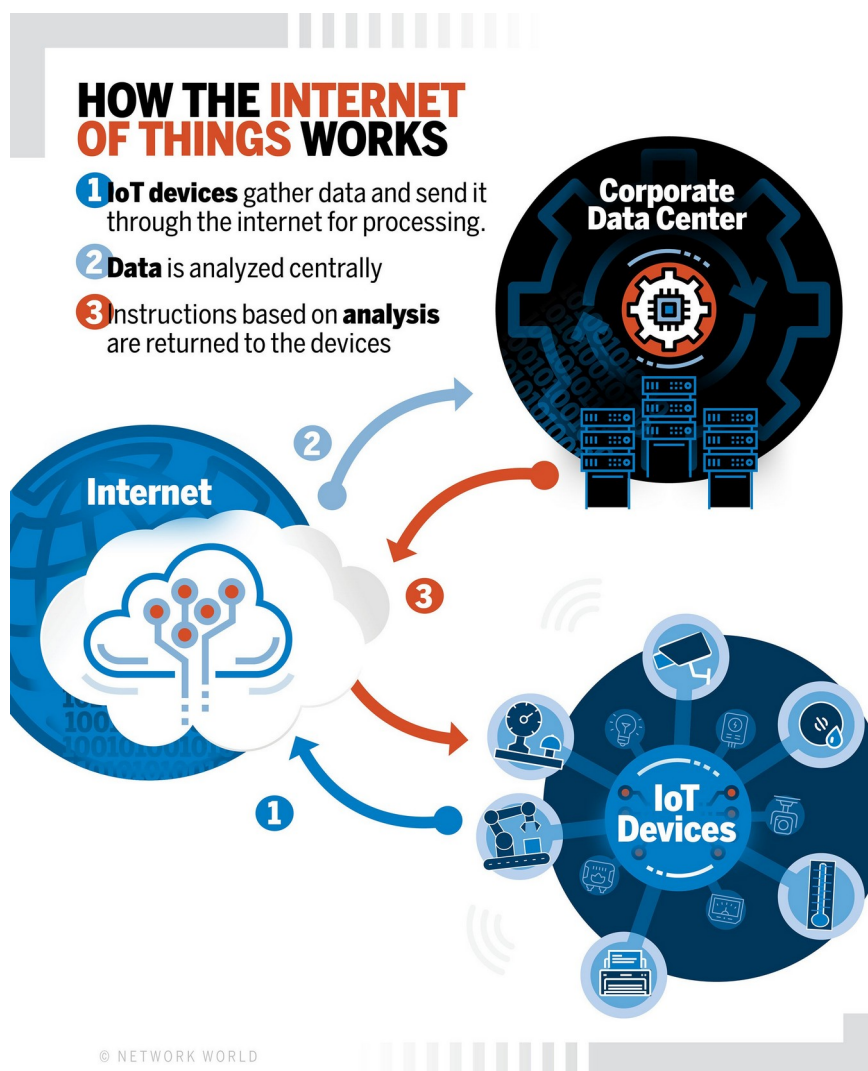
Όπως αναφέρει ο Ranger S. [1], το Διαδίκτυο των Πραγμάτων (Internet of Things) ορίζεται ως ένα δίκτυο στο οποίο συνδέονται πλήθος “έξυπνων” συσκευών, που διαθέτουν αισθητήρες, λογισμικό, υπολογιστική ισχύ, με σκοπό την ανταλλαγή δεδομένων μεταξύ τους. “Έξυπνες” συσκευές θεωρούνται όσες συσκευές δεν είναι αναμενόμενο να έχουν σύνδεση στο διαδίκτυο (ή γενικά σε κάποιο δίκτυο) και μπορούν να επικοινωνούν χωρίς να είναι απαραίτητη η ανθρώπινη παρέμβαση[1].

2.1.2 Ιστορία

Η ιδέα των αντικειμένων που έχουν αισθητήρες και κάποια βασική νοημοσύνη υπήρχε ακόμα από τη δεκαετία το 1980, προχωρούσε ωστόσο με αργούς ρυθμούς. Η υπαρκτή, τότε, τεχνολογία δεν μπορούσε να υποστηρίξει αυτό το εγχείρημα. Τα ηλεκτρονικά τσιπ ήταν πολύ ογκώδη και δεν υπήρχε τρόπος, ώστε τα αντικείμενα να επικοινωνούν αποδοτικά μεταξύ τους. Όμως, σύντομα άρχισαν να χρησιμοποιούνται οι ετικέτες RFID (RFID tags) για αυτό το σκοπό, επιλύοντας το παραπάνω πρόβλημα [1]. Οι ετικέτες RFID είναι τσιπ χαμηλής ισχύος που χρησιμοποιούν ραδιοσυχνότητες για να επικοινωνήσουν ασύρματα με άλλες συσκευές [2]. Ακόμη, η υιοθέτηση χρήσης του IPv6 (Internet Protocol version 6) μπορεί να παρέχει μία IP

διεύθυνση για όλες τις υπαρκτές συσκευές, των οποίων ο αριθμός αυξάνεται συνεχώς. Ο όρος “Διαδίκτυο των Πραγμάτων” χρησιμοποιήθηκε για πρώτη φορά από τον Kevin Ashton το 1999 σε μία παρουσίαση που έκανε ο ίδιο και έχει επικρατήσει μέχρι και σήμερα.

2.1.3 Λειτουργία



Εικόνα 1: Πώς λειτουργεί το IoT

(Πηγή: <https://www.networkworld.com/article/3207535/what-is-iot-the-internet-of-things-explained.html&>)

Τα βασικά στοιχεία που συνθέτουν το IoT είναι οι συσκευές που συλλέγουν δεδομένα. Παραδείγματα τέτοιων συσκευών είναι τα smartwatches, τα ψυγεία, ακόμη και τα κλιματιστικά που πλέον έχουν δυνατότητα σύνδεσης στο διαδίκτυο. Σύμφωνα με τον Fruhlinger J. [3], τα δεδομένα μεταδίδονται από τις συσκευές σε κάποιο κεντρικό σύστημα αποθήκευσης. Τα δεδομένα μεταδίδονται μέσα από το διαδίκτυο ή μέσα από άλλα δίκτυα σε κάποιο κέντρο δεδομένων ή υπολογιστικό νέφος που έχει υπολογιστή ισχύ και δυνατότητες αποθήκευσης [3]. Μετά την επεξεργασία των δεδομένων, επιστρέφονται οδηγίες πίσω στις συσκευές, ώστε να συνεχίσουν τη λειτουργία τους. Όσον αφορά την επεξεργασία των δεδομένων υπάρχουν 2 περιπτώσεις [4]:

- Η επεξεργασία μπορεί να γίνει σε μεταγενέστερο χρόνο. Τα δεδομένα συλλέγονται από τις διάφορες συσκευές και επεξεργάζονται με σκοπό να προκύψουν χρήσιμα συμπεράσματα και να δημιουργηθούν μοντέλα πρόβλεψης, με βάση τα οποία θα ληφθούν μελλοντικές αποφάσεις.
- Η επεξεργασία πρέπει να γίνει σε πραγματικό χρόνο. Σε αυτήν την περίπτωση, η ταχύτητα και ο μικρός χρόνος απόκρισης είναι πολύ σημαντικά, όμως η επικοινωνία με το κέντρο δεδομένων ή το υπολογιστικό νέφος μπορεί να εισάγει μεγάλες καθυστερήσεις.
-

2.2 Υπολογιστικό Νέφος

2.2.1 Ορισμός

Σύμφωνα με τον Ranger S. [5], το υπολογιστικό νέφος (Cloud computing) είναι η παροχή υπολογιστικών πόρων, απομακρυσμένων από το χρήστη, μέσω διαδικτύου. Οι πόροι αυτοί μπορεί να προσφέρουν στο χρήστη

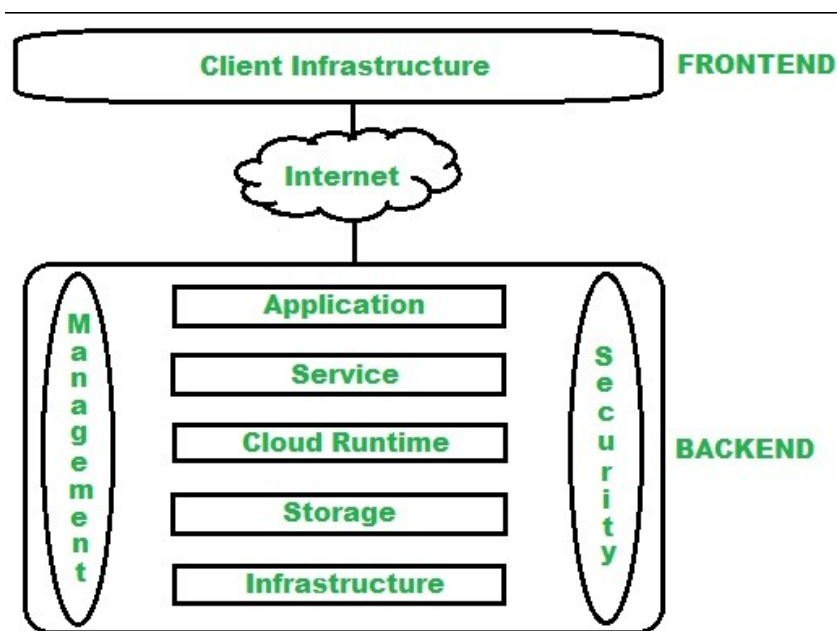
αποθήκευση, βάσεις δεδομένων, λογισμικό, εξυπηρέτες και άλλα. Κατά αυτό τον τρόπο, ο χρήστης δεν χρειάζεται να έχει στην κατοχή του όλον αυτόν τον εξοπλισμό και την υποδομή, αλλά πληρώνει μόνο για αυτό που χρησιμοποιεί και όταν το χρησιμοποιεί.

2.2.2 Ιστορία

Όπως αναφέρει ο Foote D. K. [6], η ιδέα της ενοικίασης ενός υπολογιστή υπήρχε ακόμη από τη δεκαετία του 1960, όταν οι ηλεκτρονικοί υπολογιστές δεν ήταν οικονομικά προσιτοί. Τη δεκαετία του 1990, διάφορες εταιρείες άρχισαν να προσφέρουν προς ενοικίαση εικονικά, ιδιωτικά δίκτυα, τα οποία αποτέλεσαν τις βάσεις για το σύγχρονο Cloud Computing. Το 2002, η Amazon δημιούργησε τα Amazon Web Services που έδινε τη δυνατότητα ενοίκιασης υποδομών με πιο χαμηλό κόστος. Στη συνέχεια, ακολούθησαν και άλλες εταιρίες με αποτέλεσμα το Cloud Computing να επικρατεί μέχρι και σήμερα.

2.2.3 Αρχιτεκτονική υπολογιστικού νέφους

Σύμφωνα με τον Satyabrata J. [7], η αρχιτεκτονική του υπολογιστικού νέφους αποτελείται από 2 βασικά μέρη, το Frontend και το Backend. Τα 2 αυτά μέρη επικοινωνούν μεταξύ τους μέσω του διαδικτύου ή κάποιου άλλου δικτύου:



Εικόνα 2: Η αρχιτεκτονική του υπολογιστικού νέφους

(Πηγή: <https://www.geeksforgeeks.org/architecture-of-cloud-computing/>)

1. Το Frontend αναφέρεται σε αυτό το κομμάτι του Cloud Computing που έχει να κάνει με το χρήστη. Περιλαμβάνει τη διεπαφή χρήστη και τις εφαρμογές που χρησιμοποιούνται από αυτόν για την πρόσβασή του στο υπολογιστικό νέφος. Για παράδειγμα, ο χρήστης μπορεί να χρησιμοποιεί κάποιον web browser για την πρόσβασή του. Η Υποδομή Χρήστη (Client Infrastructure) είναι το μοναδικό συστατικό στοιχείο του Frontend και παρέχει στο χρήστη μία Γραφική Διεπαφή (Graphical User Interface), ώστε αυτός να αλληλεπιδρά με το υπολογιστικό νέφος.
2. Το Backend είναι στην ουσία το υπολογιστικό νέφος. Περιλαμβάνει και διαχειρίζεται όλους τους πόρους, όπως αποθήκευση, εικονικά μηχανήματα, εφαρμογές και άλλα. Το Backend αποτελείται από τα παρακάτω επίπεδα:
 - Εφαρμογής (Application): Είναι το επίπεδο στο οποίο έχει πρόσβαση ο χρήστης και παρέχει υπηρεσίες σύμφωνα με τις απαιτήσεις του.

- Υπηρεσίας (Service): Ορίζει σε ποιον από τους 3 τύπους υπηρεσίας έχει πρόσβαση ο χρήστης, SaaS, PaaS και IaaS.
- Εκτέλεσης (Runtime cloud): Εδώ βρίσκεται το εικονικό περιβάλλον του εικονικού μηχανήματος.
- Αποθήκευσης (Storage): Παρέχει ευέλικτη και επεκτάσιμη αποθήκευση και διαχείριση των αποθηκευμένων δεδομένων.
- Υποδομής (Infrastructure): Αναφέρεται στο υλικό και λογισμικό που περιλαμβάνει το υπολογιστικό νέφος.
- Διαχείρισης (Management): Πρόκειται για τη διαχείριση των συστατικών στοιχείων του Backend, όπως υπηρεσίες, υποδομή, αποθήκευση και άλλα.
- Ασφάλειας (Security): Εδώ υλοποιούνται διάφοροι μηχανισμοί ασφαλείας, ώστε οι πόροι, τα δεδομένα και η υποδομή να είναι ασφαλή.
- Διαδικτύου (Internet)

2.2.4 Τύποι υπολογιστικού νέφους

Υπάρχουν 4 διαφορετικοί τύποι υπολογιστικού νέφους, ώστε ο καθένας να επιλέξει τον τύπο που είναι καταλληλότερος για αυτόν [8]:

- Public clouds: Είναι ο κλασσικός τύπος υπολογιστικού νέφους, όπου οι πόροι του υπολογιστικού νέφους διαμοιράζονται σε πολλούς χρήστες. Υπάρχει αφθονία πόρων, ώστε να μπορούν να δωθούν σε όποιον χρήστη ζητήσει παραπάνω. Το βασικό πλεονέκτημα αυτού του τύπου είναι η εύκολη και γρήγορη επεκτασιμότητα.
- Private clouds: Υπολογιστικά περιβάλλοντα που παρέχονται για αποκλειστική χρήση από κάποιον οργανισμό. Χρησιμοποιείται κυρίως από εταιρείες για να δημιουργήσουν τα δικά τους κέντρα

δεδομένων. Ένα επιπλέον πλεονέκτημα που προσφέρει είναι η ασφάλεια των δεδομένων των χρηστών.

- Hybrid clouds: Είναι μία μίξη μεταξύ public και private clouds. Είναι αυτό που χρησιμοποιείται περισσότερο στην πράξη.
- Multiclouds: Παρέχονται υπηρεσίες υπολογιστικού νέφους από περισσότερους από έναν παρόχους. Αυτό συμβαίνει κυρίως για να μπορέσει να χρησιμοποιήσει ο καθένας τα καλύτερα στοιχεία που προσφέρει ο κάθε πάροχος.

2.2.5 Υπηρεσίες υπολογιστικού νέφους

Οι υπηρεσίες υπολογιστικού νέφους παρέχονται στους χρήστε μέσα από το διαδίκτυο και χωρίζονται σε 3 βασικές κατηγορίες [8]:

1. Infrastructure as a Service (IaaS): Ουσιαστικά, ο χρήστης νοικιάζει την υποδομή, φυσικούς ή εικονικούς server, αποθηκευτικό χώρο και διακομιστές. Αυτή είναι μία καλή επιλογή για όσους θέλουν να χτίσουν τις εφαρμογές τους από την αρχή, ενώ ο πάροχος ασχολείται με το κομμάτι του υλικού.
2. Platform as a Service (PaaS): Μαζί με την υποδομή, ο χρήστης έχει πρόσβαση και σε λογισμικό και άλλα προγραμματιστικά εργαλεία, ώστε να δημιουργήσει την εφαρμογή του.
3. Software as a Service (SaaS): Η διάθεση μία εφαρμογής ως υπηρεσία, που είναι και η πιο συνηθής περίπτωση. Με αυτό τον τρόπο, ο πάροχος αναλαμβάνει την όλη συντήρηση της εφαρμογής και οι χρήστες δεν χρειάζεται να εγκαταστήσουν τις εφαρμογές τοπικά στα μηχανήματά τους.

2.2.6 Πλεονεκτήματα

Όπως μας λέει ο Peterson R. [9], η χρήση του υπολογιστικού νέφους έχει επικρατήσει, καθώς έχει σημαντικά πλεονεκτήματα. Το κυριότερο είναι το χαμηλό κόστος. Ο χρήστης δεν χρειάζεται να διαθέσει χρήματα για την αγορά και συντήρηση εξοπλισμού, καθώς αυτό γίνεται εξ' ολοκλήρου από τον πάροχο του υπολογιστικού νέφους. Ένα ακόμη πλεονέκτημα είναι η αξιοπιστία. Τα δεδομένα των χρηστών είναι αποθηκευμένα σε πολλούς servers, σε πολλές διαφορετικές τοποθεσίες. Σκοπός αυτού είναι να μην χαθούν δεδομένα χρηστών, σε περίπτωση κάποις δυσλειτουργίας. Άλλο ένα πλεονέκτημα είναι η επεκτασιμότητα. Το υπολογιστικό νέφος προσφέρει σχεδόν απεριόριστη χωριτικότητα, γεγονός που σημαίνει πως ο χρήστης μπορεί να επεκτείνει την εφαρμογή του ή το χώρο αποθήκευσής του όταν χρειαστεί.

2.2.7 Μειονεκτήματα

Ωστόσο, το υπολογιστικό νέφος έχει και κάποια μειονεκτήματα. Σύμφωνα με τον Tolsma A. [10], ένα από αυτά είναι η προστασία των προσωπικών δεδομένων των χρηστών. Το υπολογιστικό νέφος διαμοιράζει τους πόρους τους στους διάφορους χρήστες. Αυτό πρακτικά σημαίνει πως κάποια ευαίσθητα προσωπικά δεδομένα μπορεί να είναι προσβάσιμα από τρίτους, πράγμα που μπορεί να οδηγήσει στη διαρροή τους. Σημαντικό ρόλο παίζει και η γεωγραφική θέση. Η φυσική τοποθεσία αποθήκευσης είναι καθοριστικός παράγοντας για να προσδιορίσουμε ποιοι νόμοι ισχύουν όσον αφορά τα προσωπικά δεδομένα. Με το υπολογιστικό νέφος, όμως, ο τόπος δεν είναι πάντα ξεκάθαρος. Επομένως, είναι δύσκολο να βρούμε ποιοι νόμοι πρέπει να εφαρμοστούν.

Το κυριότερο, όμως, μειονέκτημα, σύμφωνα με τον Fruhlinger J. [3], είναι η καθυστέρηση στη μεταφορά των δεδομένων. Υπάρχουν εφαρμογές, οι οποίες είναι ευαίσθητες στο χρόνο. Η επεξεργασία των δεδομένων που συλλέγουν πρέπει να γίνει το γρηγορότερο δυνατό. Αυτό ίσως να μην μπορεί να επιτευχθεί. Ο κύριος παράγοντας, που συμβάλει στην αποτυχία αυτού του στόχου, είναι η τοποθεσία. Σε κάποιες περιπτώσεις, τα δεδομένα χρειάζεται να κάνουν μεγάλο “ταξίδι” μέχρι να φτάσουν στον προορισμό τους, γεγονός που τα καθιστά ανώφελα και μπορεί να αποβεί καταστροφικό για την εκάστοτε εφαρμογή.

2.3 Καθυστέρηση μεταφοράς δεδομένων

2.3.1 Ορισμός

Ως καθυστέρηση (latency) ορίζεται ο χρόνος μεταξύ ενός αιτήματος του χρήστη και της απόκρισης από τον πάροχο της υπηρεσίας, όπως αναφέρει ο Gibb [12]. Με άλλα λόγια, η καθυστέρηση είναι ο χρόνος που χρειάζονται τα δεδομένα για να φτάσουν στον προορισμό τους. Ο χρόνος καθυστέρησης μετράται σε milliseconds (ms).

2.3.2 Αιτίες καθυστέρησης

Μία σημαντική αιτία που οδηγεί στην καθυστέρηση είναι ο πολύ μεγάλος και συνεχώς αυξανόμενος όγκος των δεδομένων. Οι συσκευές που συνδέονται στο διαδίκτυο και συλλέγουν δεδομένα γίνονται όλο και περισσότερες, με αποτέλεσμα να αυξάνονται αντιστοίχως και τα δεδομένα. Όταν έχουμε περισσότερα δεδομένα, είναι λογικό να χρειάζεται και περισσότερος χρόνος για τη μεταφορά τους.

Άλλος ένας λόγος είναι το εύρος ζώνης (bandwidth) του δικτύου [11]. Το εύρος ζώνης είναι η μέγιστη ποσότητα δεδομένων που μπορούν να μεταφέρονται από το δίκτυο κάθε στιγμή. Είναι σαφές πως, αν ο όγκος των δεδομένων ξεπερνά το εύρος ζώνης, θα έχουμε επιπλέον καθυστέρηση.

Τέλος, ο σημαντικότερος παράγοντας που επηρεάζει η καθυστέρηση είναι η απόσταση [12]. Όσο πιο μακριά βρίσκεται η συσκευή, που θέλει να μεταφέρει δεδομένα, από το κέντρο δεδομένων, για το οποίο αυτά προορίζονται, τόσο περισσότερος χρόνος θα χρειαστεί.

Συνεπώς, ο καλύτερος τρόπος, σύμφωνα με τον Gillis [11], για να μειώσουμε την καθυστέρηση είναι να μειώσουμε την απόσταση. Αυτό μπορούμε να το πετύχουμε μεταφέροντας τα κέντρα δεδομένων, ή ένα μέρος αυτών, πιο κοντά στο χρήστη, στην ακμή του δικτύου.

2.3.3 Μέτρηση καθυστέρησης

Τα εργαλεία, που χρησιμοποιούνται για την μέτρηση της δικτυακής καθυστέρησης, βασίζονται στη λειτουργία του πρωτοκόλλου ICMP (Internet Control Message Protocol) [11]. Σύμφωνα με τον Lutkevich [14], το πρωτόκολλο ICMP χρησιμοποιείται για τη διάγνωση σφαλμάτων σε ένα δίκτυο. Η βασική του δουλειά είναι να προσδιορίσει εάν τα δεδομένα που αποστέλλονται φτάνουν έγκαιρα στον προορισμό τους. Η λειτουργία του ICMP είναι απλή. Ο υπολογιστής του χρήστη στέλνει ένα ICMP echo request και λαμβάνει από τον server ένα ICMP echo reply.

ICMP echo request and reply



ICONS: LUKPEDCLUB/ADOBE STOCK, ENISGORELKIN/ADOBE STOCK
©2021 TECHTARGET. ALL RIGHTS RESERVED

Εικόνα 3: Η λειτουργία του ICMP

(Πηγή: <https://www.techtarget.com/searchnetworking/definition/ICMP>)

Ο Notermans [13] μας δίνει μία περιγραφή των βασικών εργαλείων. Ένα από τα βασικά εργαλεία για τη μέτρηση της δικτυακής καθυστέρησης είναι η εντολή ping. Το ping αξιοποιεί το πρωτόκολλο ICMP, καθώς στέλνει ένα ICMP echo request και περιμένει να λάβει ένα ICMP echo reply. Αυτό που κάνει το ping είναι η μέτρηση του χρόνου μετάδοσης και επιστροφής (round trip time, RTT). RTT είναι ο χρόνος από τη στιγμή που γίνεται το ICMP echo request έως τη στιγμή που θα ληφθεί το ICMP echo reply. Η εντολή ping πραγματοποιείται μέσα από τον terminal του υπολογιστή μας. Η βασική σύνταξή της είναι η εξής:

`ping όνομα ή IP του server`

```
mariana@io:~$ ping -c 5 google.com
PING google.com (172.217.17.206) 56(84) bytes of data.
64 bytes from sof02s22-in-f206.1e100.net (172.217.17.206): icmp_seq=1 ttl=112 time=35.7 ms
64 bytes from sof02s22-in-f206.1e100.net (172.217.17.206): icmp_seq=2 ttl=112 time=35.7 ms
64 bytes from sof02s22-in-f206.1e100.net (172.217.17.206): icmp_seq=3 ttl=112 time=35.8 ms
64 bytes from sof02s22-in-f206.1e100.net (172.217.17.206): icmp_seq=4 ttl=112 time=36.6 ms
64 bytes from sof02s22-in-f206.1e100.net (172.217.17.206): icmp_seq=5 ttl=112 time=36.1 ms

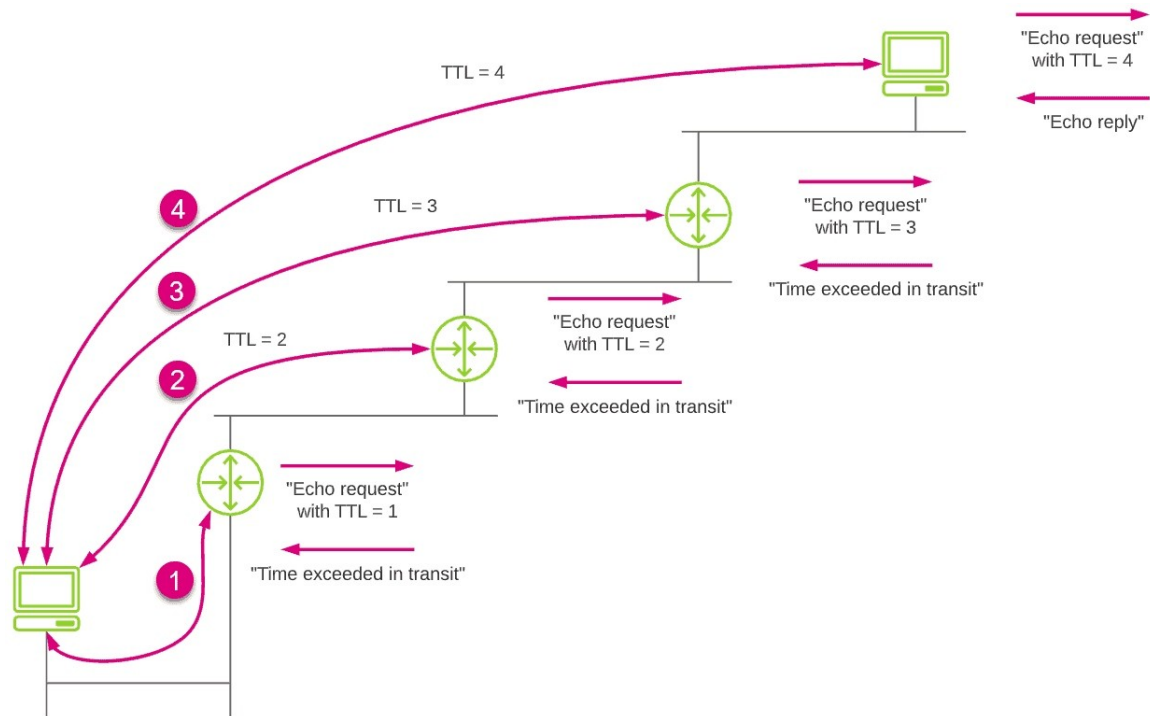
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 35.656/35.957/36.563/0.341 ms
```

Εικόνα 4: Τα αποτελέσματα του ping

Στο παραπάνω παράδειγμα έχει προστεθεί η επιπλέον παράμετρος -c 5, που σημαίνει ότι η εντολή ping θα εκτελεστεί 5 φορές, δηλαδή θα αποσταλούν 5 ICMP πακέτα. Για κάθε μία εκτέλεση, έχουμε τα πεδία icmp_seq, ttl και time. Το πεδίο icmp_seq υποδικνεύει ποιο είναι κατά σειρά το πακέτο που στάλθηκε. Το πεδίο ttl (time to live) είναι ένας μετρητής, που μειώνεται κατά ένα κάθε φορά που το πακέτο περνά από κάποιον δρομολογητή. Όταν ο αριθμός αυτός φτάσει στο 0, το πακέτο σταματά να μεταδίδεται. Το πεδίο time αναφέρεται στο RTT. Στο τέλος, βλέπουμε κάποια στατιστικά στοιχεία. Αναφέρεται πόσα πακέτα παραδόθηκαν και πόσα παραλήφθηκαν, το ποσοστό απώλειας πακέτων και τον συνολικό χρόνο που χρειάστηκε για την εκτέλεση όλων των ping. Στην τελευταία γραμμή, δίνονται κάποιες τιμές για το RTT, συγκεκριμένα το μικρότερο, το μέσο και το μεγαλύτερο RTT, καθώς και η τυπική απόκλισή του.

Μία άλλη εναλλακτική για τη μέτρηση της καθυστέρησης, είναι το traceroute. Η εντολή traceroute αξιοποιεί το πεδίο ttl, προκειμένου να ανακαλύψει όλους τους ενδιάμεσους κόμβους από τον χρήστη έως τον προορισμό. Με αυτόν τον τρόπο μετρά την καθυστέρηση έως τον κάθε ένα κόμβο ξεχωριστά. Το traceroute θέτει την αρχική τιμή του ttl σε ένα και την αυξάνει κατά ένα κάθε φορά. Έτσι, κάθε φορά που το traceroute στέλνει ένα ICMP echo request, λαμβάνει ένα ICMP error message αρχικά από τον

πρώτο κόμβο, στη συνέχεια από τον δεύτερο και ούτω καθεξής έως ότου φτάσει στον τελευταίο κόμβο, από τον οποίο θα λάβει ένα ICMP echo reply.



Εικόνα 5: Η λειτουργία του traceroute

(Πηγή: <https://www.kadiska.com/measure-network-latency/>)

Η σύνταξη του traceroute είναι απλή:

traceroute όνομα ή IP του server

Ας δούμε ξανά το παραπάνω παράδειγμα, αυτή τη φορά με τη χρήση του traceroute:

```

mariana@io:~$ traceroute google.com
traceroute to google.com (172.217.17.206), 30 hops max, 60 byte packets
 1  vodafone.station (192.168.2.1)  3.539 ms  3.518 ms  3.489 ms
 2  loopback2004.med01.dsl.hol.gr (62.38.0.170)  12.743 ms  12.736 ms  12.715 ms
 3  ppp062038028233.dsl.hol.gr (62.38.28.233)  12.664 ms  12.640 ms  17.007 ms
 4  62.38.96.137 (62.38.96.137)  25.627 ms  25.661 ms  25.637 ms
 5  62.38.96.150 (62.38.96.150)  25.587 ms  25.559 ms  25.533 ms
 6  62.74.30.194 (62.74.30.194)  25.442 ms  21.198 ms  23.141 ms
 7  ae3-100-ucr.ata.cw.net (195.89.103.69)  23.136 ms  22.811 ms  22.761 ms
 8  ae25-xcr1.sof.cw.net (195.2.16.13)  34.116 ms  34.150 ms  34.130 ms
 9  72.14.218.54 (72.14.218.54)  35.811 ms  209.85.168.146 (209.85.168.146)  37.797 ms  72.14.208.246 (72.14.208.246)  35.919 ms
10  * * *
11  209.85.142.64 (209.85.142.64)  37.555 ms  108.170.238.170 (108.170.238.170)  39.733 ms  209.85.142.54 (209.85.142.54)  39.703 ms
12  209.85.143.115 (209.85.143.115)  37.561 ms  209.85.246.71 (209.85.246.71)  37.482 ms  37.439 ms
13  sof02s22-in-f14.1e100.net (172.217.17.206)  37.313 ms  37.334 ms  37.331 ms

```

Εικόνα 6: Τα αποτελέσματα του traceroute

Τα αποτελέσματα του traceroute εμφανίζονται σε 5 στήλες. Η πρώτη στήλη μας δείχνει την τιμή του πεδίου ttl. Η δεύτερη στήλη αναγράφει το όνομα του σταθμού ή την IP διεύθυνση του συγκεκριμένου σταθμού ή και τα δύο. Στην πράξη, η εντολή traceroute στέλνει 3 ICMP echo requests κάθε φορά. Στις 3 τελευταίες στήλες, έχουμε το RTT για κάθε ένα από αυτά τα requests. Ακόμη, σε κάποιες από τις γραμμές, μπορεί να δούμε αστερίσκους αντί για τα παραπάνω στοιχεία. Αυτό σημαίνει πως ο συγκεκριμένος σταθμός δεν έδωσε απάντηση εντός του αποδεκτού χρόνου. Αξίζει να παρατηρήσουμε πως όσο αυξάνεται ο αριθμός των κόμβων, αυξάνεται και το RTT. Συμπεραίνουμε, λοιπόν, πως για να μειώσουμε το χρόνο μεταφοράς των δεδομένων, είναι απαραίτητο να μετακινηθούν τα κέντρα δεδομένων πιο κοντά στους τελικούς χρήστες, στην ακμή του δικτύου.

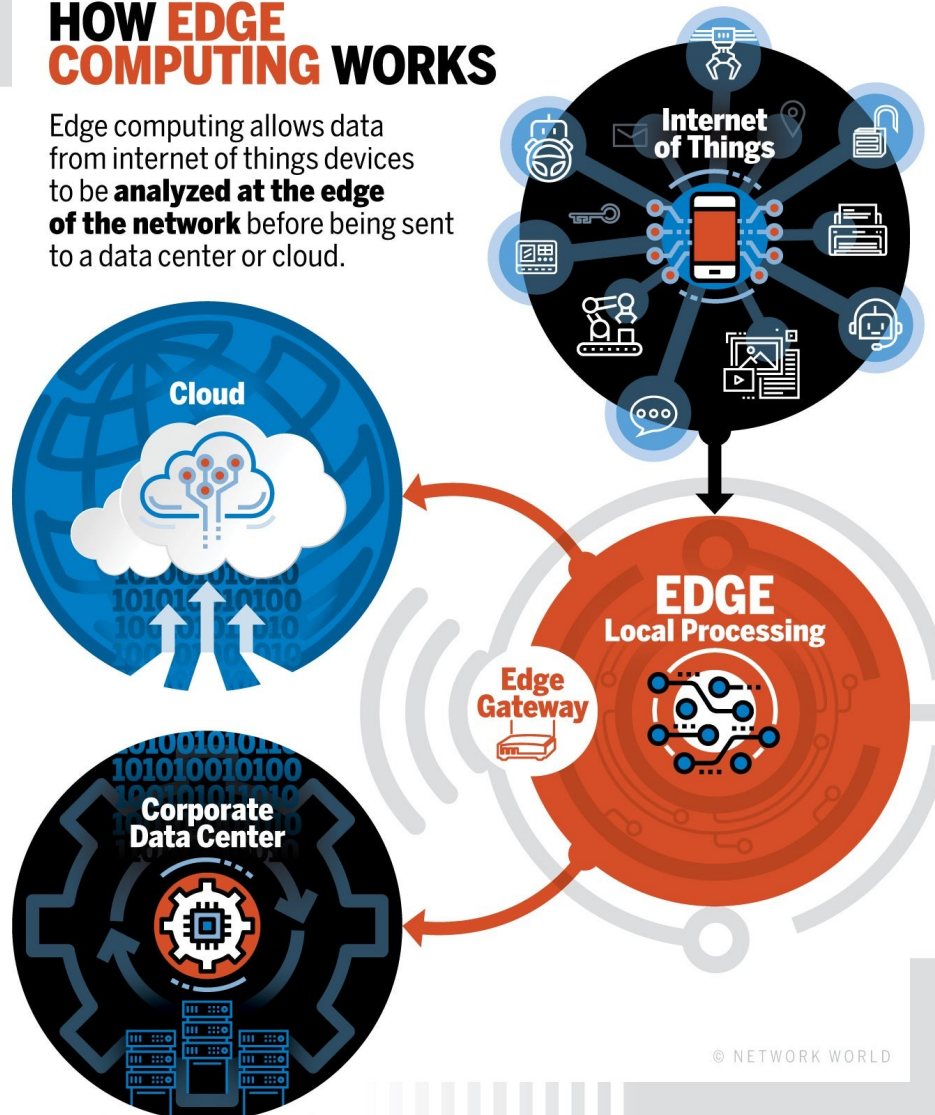
2.4 Περιβάλλον Υπολογισμού Ακμής

2.4.1 Ορισμός

Σύστημα παρακολούθησης δικτυακής καθυστέρησης μεταφοράς δεδομένων σε περιβάλλοντα υπολογισμού ακμής / Μαριάννα Στεφάνοβα

HOW EDGE COMPUTING WORKS

Edge computing allows data from internet of things devices to be **analyzed at the edge of the network** before being sent to a data center or cloud.



Εικόνα 7: Πώς λειτουργεί το Edge Computing

(Πηγή: <https://www.networkworld.com/article/3207535/what-is-iot-the-internet-of-things-explained.html&>)

Ο Bigelow [15] μας αναφέρει πως το Περιβάλλον Υπολογισμού Ακμής (Edge Computing) είναι μία κατανεμημένη υπολογιστική αρχιτεκτονική,

όπου η αποθήκευση και επεξεργασία των δεδομένων πραγματοποιείται κοντά στην πηγή από όπου συλλέχθηκαν. Αντί τα δεδομένα να αποσταλλούν σε κάποιο κέντρο δεδομένων, η επεξεργασία τους γίνεται στην ακμή του δικτύου. Το Edge Computing δεν αντικαθιστά το Cloud Computing, αλλά το συμπληρώνει [17]. Η ακμή αναλαμβάνει δεδομένα, των οποίων η επεξεργασία πρέπει να γίνει γρήγορα, και μπορεί στη συνέχεια να τα στείλει στο υπολογιστικό νέφος για περαιτέρω ανάλυση και αποθήκευση.

2.4.2 Αρχιτεκτονική

Σύμφωνα με τον Velimirovic [17], ένα περιβάλλον υπολογισμού ακμής αποτελείται από τα παρακάτω στοιχεία:

- Συσκευές ακμής (edge devices): Είναι αυτές που συλλέγουν δεδομένα. Οι συσκευές αυτές μπορεί να είναι κινητά τηλέφωνα, κάμερες ασφαλείας, αυτοοδηγούμενα αυτοκίνητα, αισθητήρες μέτρησης της θερμοκρασίας και πολλές ακόμη.
- Κόμβοι ακμής (edge nodes): Εκεί γίνεται η επεξεργασία και αποθήκευση των δεδομένων.
- Server ακμής (edge server): Εκτελεί εργασίες και κοινόχρηστες υπηρεσίες. Βρίσκεται σε κάποιο κτίριο κοντά στις συσκευές ακμής.
- Πύλη ακμής (edge gateway): Ένας server που εκτελεί διάφορες δικτυακές λειτουργίες.
- Υπολογιστικό νέφος: Εκεί καταλήγουν τα δεδομένα που χρειάζονται περαιτέρω επεξεργασία ή μακροχρόνια αποθήκευση.

2.4.3 Λειτουργία

Όπως αναφέρουν οι Gold et al [16], η λειτουργία του edge computing έχει να κάνει κυρίως με την τοποθεσία. Οι συσκευές ακμής συλλέγουν δεδομένα και τα μεταδίνουν σε κάποιον κόμβο ακμής, που βρίσκεται κοντά στη γεωγραφική τους θέση. Ο κόμβος ακμής επεξεργάζεται και αποθηκεύει τα δεδομένα που έλαβε. Στη συνέχεια, επιστρέφει τα αποτελέσματα της επεξεργασίας πίσω στην συσκευή ακμής, ώστε αυτή να συνεχίσει τη λειτουργία της. Αν θεωρηθεί απαραίτητο, μεταφέρει τα δεδομένα στο υπολογιστικό νέφος ή σε κάποιο κέντρο δεδομένων, όπου μπορεί να γίνει κάποια επιπλέον επεξεργασία ή τα δεδομένα να αποθηκευτούν για μεγάλο χρονικό διάστημα.

2.4.4 Πλεονεκτήματα

Ο Velimirovic [17] κάνει μία αναφορά στα κυριότερα πλεονεκτήματα του υπολογισμού ακμής. Το βασικό πρόβλημα, που επιλύεται με τη χρήση του edge computing, είναι η καθυστέρηση μεταφοράς των δεδομένων. Υπάρχουν πολλοί κόμβοι ακμής, διασκορπισμένοι σε διάφορα σημεία, ώστε να βρίσκονται όσο πιο κοντά γίνονται στην πηγή, όπου παράγονται τα δεδομένα. Εφόσον οι κόμβοι ακμής βρίσκονται πιο κοντά στις συσκευές, ο χρόνος που απαιτείται για τη μεταφορά των δεδομένων είναι πολύ μικρότερος. Τα δεδομένα δε χρειάζεται να περάσουν από πολλούς, ενδιάμεσους σταθμούς, για να φτάσουν στο υπολογιστικό νέφος ή σε κάποιο απομακρυσμένο κέντρο δεδομένων. Κατά αυτόν τον τρόπο, γίνεται γρηγορότερα και η επεξεργασία των δεδομένων, πράγμα που είναι ιδιαίτερα χρήσιμο για εφαρμογές ευαίσθητες στην καθυστέρηση.

Άλλο ένα σημαντικό πλεονέκτημα είναι η αυτονομία [15]. Αν υπάρξει κάποιο δικτυακό πρόβλημα σε κάποιον μακρινό κόμβο, αυτό δε θα επηρεάσει

τη λειτουργία των συσκευών και κόμβων ακμής. Ακόμη, το εύρος ζώνης προς κάποιο κέντρο δεδομένων μπορεί να είναι περιορισμένο, ιδιαίτερα αν τα δεδομένα είναι πολλά σε όγκο. Όμως, με το edge computing, ο αριθμός των δεδομένων, που πρέπει να αποσταλούν στο υπολογιστικό νέφος ή σε κάποιο κέντρο δεδομένων, μειώνεται σημαντικά, καθώς γίνεται επεξεργασία τους στους κόμβους ακμής.

Ακόμη ένα πλεονέκτημα που προσφέρει το edge computing είναι η προστασία των προσωπικών δεδομένων [17]. Με τη χρήση του υπολογιστικού νέφους, τα δεδομένα μπορεί να αποθηκεύονται σε κάποια άλλη χώρα, η οποία πολύ πιθανόν να έχει κάποια διαφορετική νομοθεσία σχετικά με την προστασία των προσωπικών δεδομένων. Όμως, με το edge computing, η επεξεργασία των δεδομένων αυτών δεν απομακρύνεται πολύ από τον τόπο παραγωγής τους. Έτσι, δίνεται η δυνατότητα, τα ευαίσθητα προσωπικά δεδομένα να προστατευτούν, πριν αποσταλούν σε κάποιο κέντρο δεδομένων που είναι υπό άλλη δικαιοδοσία. Σε κάποιες περιπτώσεις, ίσως να μη χρειαστεί καθόλου η αποστολή τους.

2.4.5 Μειονεκτήματα

Οστόσο, το edge computing έχει και κάποια μειονεκτήματα [17]. Ένα από τα μειονεκτήματα του edge computing είναι η ασφάλεια. Προσθέτοντας περισσότερα στοιχεία σε ένα δίκτυο, προστίθενται και περισσότερα σημεία στα οποία μπορεί να γίνει επίθεση. Είναι δύσκολο να στηθεί επαρκής ασφάλεια, καθώς κάθε συσκευή ακμής έχει διαφορετικές δυνατότητες.

Ένα ακόμη αρνητικό στοιχείο είναι το κόστος της υποδομής. Το απαραίτητο υλικό και λογισμικό για την υλοποίηση της υποδομής ενός περιβάλλοντος υπολογισμού ακμής μπορεί να έχει αρκετά υψηλό κόστος, το οποίο ανεβαίνει ακόμα περισσότερο όταν η υπολοποίηση πρέπει να γίνει σε πολλές και διαφορετικές τοποθεσίες.

Άλλη μία ανησυχία που προκύπτει είναι η απώλεια δεδομένων. Εφόσον ο κόμβος ακμής κάνει την επεξεργασία των δεδομένων του, μπορεί να αποφασίσει πως αυτά τα δεδομένα είναι πλέον άχρηστα και να μην τα προωθήσει στο υπολογιστικό νέφος για αποθήκευση.

2.4.6 Χρήση

Ένα περιβάλλον υπολογισμού ακμής είναι ιδιαίτερα χρήσιμο όταν δεν είναι δυνατόν να μεταφέρουμε όλα τα δεδομένα μας σε κάποιο κέντρο δεδομένων [3]. Αυτό μπορεί να συμβαίνει γιατί τα δεδομένα είναι ευαίσθητα στο χρόνο και η επεξεργασία τους πρέπει να γίνει σε μικρό χρονικό διάστημα, ειδικά η επεξεργασία τους δεν έχει κανένα νόημα. Άλλος ένας λόγος είναι ο πολύ μεγάλος όγκος των δεδομένων, ο οποίος θα οδηγήσει σε ακόμα μεγαλύτερη καθυστέρηση της μεταφοράς τους. Κάποια παραδείγματα, όπου το edge computing έχει ουσιαστική σημασία, είναι τα παρακάτω [17]:

- Υγεία: Ο αριθμός δεδομένων που συλλέγονται, μέσω διάφορων συσκευών, αισθητήρων και άλλου ιατρικού εξοπλισμού, για τον κάθε ασθενή, έχει αυξηθεί δραματικά. Με τη βοήθεια του edge computing, τα δεδομένα αυτά μπορούν να μεταφερθούν πιο γρήγορα και να αναγνωστούν οποιεσδήποτε ανωμαλίες, προκειμένου το ιατρικό προσωπικό να μην χάσει πολύτιμο χρόνο.

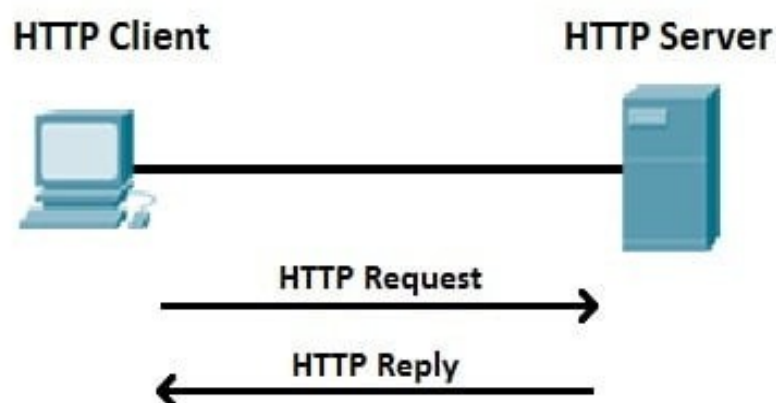
- **Μετακίνηση:** Ένα αυτοκίνητο μπορεί να είναι μία συσκευή ακμής που μαζεύει πολλά δεδομένα από διάφορους αισθητήρες. Σκοπός είναι να λάβει μία απάντηση σε πραγματικό χρόνο σχετικά με καταστάσεις που συμβαίνουν στο δρόμο. Αυτό είναι ιδιαίτερα χρήσιμο στην ανάπτυξη των αυτοοδηγούμενων αυτοκινήτων.
- **Βιντεοεπιτήρηση:** Η μεταφορά δεδομένων σε μορφή βίντεο σε κάποιο κέντρο δεδομένων είναι αργή. Το edge computing κάνει πιο γρήγορη αυτή τη διαδικασία, πραγματοποιώντας ανάλυση του βίντεο και υποδεικνύοντας γεγονότα μεγάλης σημασίας, π.χ. το ξέσπασμα μίας φωτιάς.
- **Εμπόριο [15]:** Τα εμπορικά καταστήματα επίσης παράγουν τεράστιο όγκο δεδομένων, κάνοντας παρακολούθηση του αποθέματος και των πωλήσεων. Ένα υπολογιστικό περιβάλλον ακμής μπορεί να αναλύσει αυτά τα δεδομένα και να εξάγει χρήσιμα συμπεράσματα, για παράδειγμα ποια προϊόντα κάνουν μεγαλύτερες πωλήσεις και πρέπει να υπάρχει μεγαλύτερο απόθεμα από αυτά.

2.5 Τεχνολογίες ανάπτυξης συστήματος

2.5.1 Πρωτόκολλο HTTP

Ο Asati [19] τον τρόπο λειτουργίας του πρωτοκόλλου HTTP. Το πρωτόκολλο HTTP είναι ένα πρωτόκολλο επικοινωνίας μονής κατεύθυνσης, όπου ο client στέλνει ένα αίτημα και στη συνέχεια ο server στέλνει μία απάντηση. Είναι ένα stateless πρωτόκολλο, που σημαίνει ότι η σύνδεση τερματίζεται μόλις ο client λάβει την απάντηση από τον server. Για να γίνει μία HTTP σύνδεση, πρώτα στέλνει ο client ένα HTTP request στον server. Έστερα, ο server στέλνει ένα HTTP response στον client με τα δεδομένα που ζήτησε και η σύνδεση κλείνει. Κάθε φορά που ο client χρειάζεται κάτι από

τον server, πρέπει να στείλει εκ νέου ένα HTTP request, ώστε να λάβει το αντίστοιχο HTTP response. Η σύνδεση που εγκαθιδρύεται μεταξύ τους είναι TCP και εκγυάται την παράδοση των πακέτων. Το πρωτόκολλο HTTP χρησιμοποιείται κυρίως όταν δεν είναι αναγκαία η μόνιμη σύνδεση μεταξύ client και server.



Εικόνα 8: Πώς λειτουργεί το πρωτόκολλο HTTP

(Πηγή: <https://www.websiterating.com/el/web-hosting/glossary/what-is-http/>)

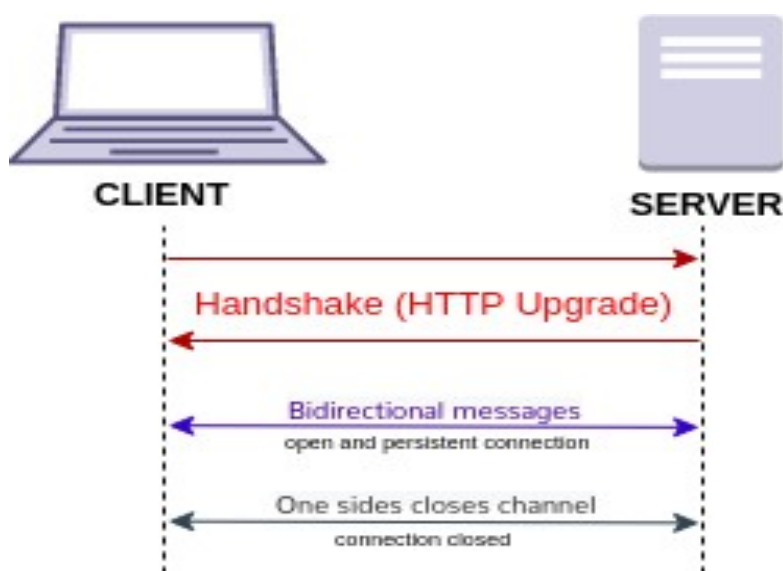
Το πρωτόκολλο HTTP παρέχει κάποιες μεθόδους, που συνοδεύουν τα requests και υποδεικνύουν την πράξη που πρέπει να κάνει ο server. Κάποιες από τις κυριότερες μεθόδους είναι οι εξής [23]:

- GET: Χρησιμοποιείται για την αποστολή δεδομένων προς τον client.
- POST: Χρησιμοποιείται ώστε να στείλει ο client δεδομένα στον server.
- PUT: Η μέθοδος αυτή αντικαθιστά δεδομένα που υπάρχουν ήδη στον server με αυτά που του στέλνει ο client

- DELETE: Διαγράφει από τον server τα δεδομένα που ορίζει ο client.

2.5.2 WebSockets

Με τον όρο WebSockets αναφερόμαστε σε ένα πρωτόκολλο επικοινωνίας, το οποίο παράχει πλήρη αμφίδρομη επικοινωνία μέσω μίας TCP σύνδεσης [18]. Είναι ένα stateful πρωτόκολλο, που σημαίνει πως η σύνδεση μεταξύ ενός client και ενός server παραμένει ανοιχτή έως ότου κάποιος από τους δύο να την κλείσει. Για να εγκαθιδρυθεί μία σύνδεση, αρχικά ο client στέλνει ένα αίτημα στον server, με τη βοήθεια ενός HTTP request, μέσα από το οποίο ζητά μία WebSocket σύνδεση [19]. Ο server απαντά με ένα HTTP response, πως δέχεται να αναβαθμίσουν τη σύνδεσή τους σε WebSocket. Στη συνέχεια, ο client και ο server μπορούν να ανταλλάσσουν μηνύματα μέχρι ένας από τους δύο να τερματίσει τη σύνδεση. Ένας server μπορεί να έχει πολλές WebSocket συνδέσεις με πολλούς clients ταυτόχρονα. Οι WebSocket συνδέσεις είναι ιδιαίτερα χρήσιμες όταν θέλουμε η επικοινωνία να γίνεται σε πραγματικό χρόνο.



Εικόνα 9: Η λειτουργία των WebSockets
Σύστημα παρακολούθησης δικτυακής καθυστέρησης μεταφοράς δεδομένων σε περιβάλλοντα υπολογισμού ακμής / Μαριάννα Στεφάνοβα

(Πηγή: <https://en.wikipedia.org/wiki/WebSocket>)

2.5.3 Redis

Το Redis είναι ένας χώρος αποθήκευσης δεδομένων στη μνήμη, που χρησιμοποιείται ως βάση δεδομένων, με τη μορφή “κλειδί-τιμή”. Συχνά, αποκαλείται και server δομών δεδομένων, γιατί οι τύποι δεδομένων που προσφέρει είναι αυτοί που χρησιμοποιούνται και στον προγραμματισμό, όπως strings, hashes, λίστες και άλλα. Λόγω της ευκολίας στη χρήση του, είναι κατάλληλο για γρήγορη ανάπτυξη εφαρμογών. Το Redis είναι ιδιαίτερα χρήσιμο όταν γίνεται πολύ συχνά προσπέλαση των ίδιων δεδομένων και συμβάλλει σημαντικά στη μείωση της καθυστέρησης. Η διαχείριση του redis γίνεται μέσα από το redis-cli, το οποίο εκτελείται με τη βοήθεια της γραμμής εντολών.

2.5.4 Android

Το Android είναι ένα λειτουργικό σύστημα, ανοιχτού κώδικα βασισμένο στο Linux, που έχει σχεδιαστεί για κινητές συσκευές, όπως smartphone ή tablets [21]. Το Android είναι το πιο διαδεδομένο λειτουργικό σύστημα για κινητές συσκευές, με συνεχώς αυξανόμενο αριθμό συσκευών που το χρησιμοποιούν. Με το Android, δίνεται στους προγραμματιστές η δυνατότητα να κατασκευάζουν εφαρμογές, γνωρίζοντας πως οι εφαρμογές τους θα μπορούν να τρέχουν σε διάφορες συσκευές με λειτουργικό σύστημα Android. Οι Android εφαρμογές συνύθως γράφονται σε γλώσσα Java, αν και είναι πλέον αρκετά διαδεδομένη και η Kotlin. Για τη δημιουργία των

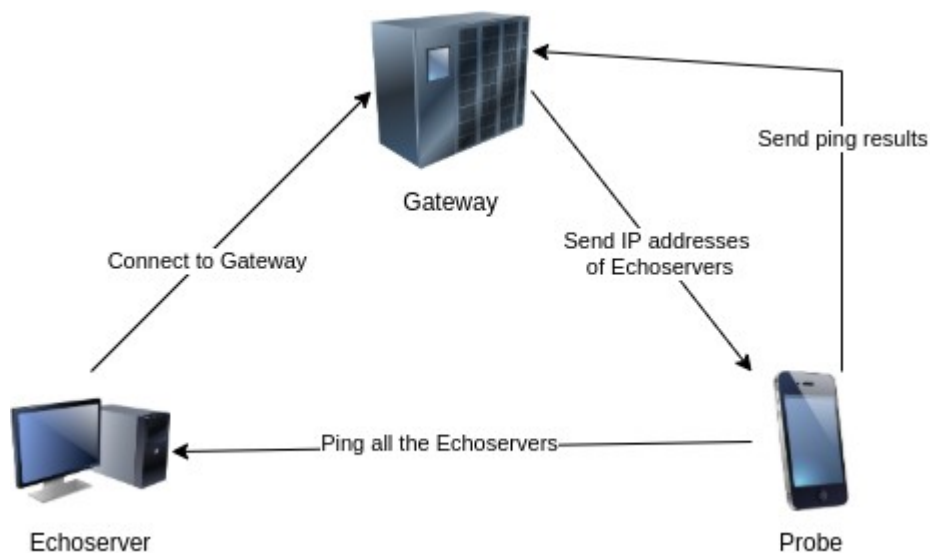
εφαρμογών είναι απαραίτητη η χρήση του Android SDK (Software Development Tool). Η υλοποίηση των Android εφαρμογών γίνεται με τη βοήθεια κάποιου ολοκληρωμένου περιβάλλοντος ανάπτυξης (Integrated Development Enviroment, IDE). Το κύριο IDE της Google είναι το Android Studio, το οποίο σχεδιάστηκε αποκλειστικά για προγραμματισμό Android [22].

3. ΥΛΟΠΟΙΗΣΗ

3.1 Αρχιτεκτονική

Όπως αναφέρθηκε και στο κεφάλαιο 1, το σύστημα αυτό αποτελείται από 3 δομικά στοιχεία:

1. Gateway: Έχει το ρόλο ενός απομακρυσμένου κέντρου δεδομένων, όπου αποθηκεύονται οι IP διευθύνσεις των Echoservers και των Probes, με τη βοήθεια μίας βάσης Redis.
2. Echoservers: Αναπαριστούν τους κόμβους ακμής και επιτρέπουν στα Probes να τους κάνουν ping.
3. Probes: Είναι οι συσκευές ακμής, συγκεκριμένα κινητές συσκευές, οι οποίες κάνουν ping τους Echoservers, μέσω μίας android εφαρμογής και στέλνουν στον Gateway τα αποτελέσματα.



Εικόνα 10: Η αρχιτεκτονική του συστήματος
Σύστημα παρακολούθησης δικτυακής λειτουργικότητας μεταφορών σε περιβάλλοντα υπολογισμού ακμής / Μαριάννα Στεφάνοβα

(Το παραπάνω σχήμα δημιουργήθηκε με τη βοήθεια του draw.io available at <https://app.diagrams.net/>)

3.2 Gateway

Ο Gateway εκτελεί ορισμένες λειτουργίες. Αρχικά, συνδέεται με τους Echoservers μέσω Websockets και διατηρεί μία συνεδρία για κάθε έναν από αυτούς. Συλλέγει όλες τις IP διευθύνσεις τους και τις αποθηκεύει σε μία βάση δεδομένων Redis, μαζί με την τρέχουσα κατάστασή τους, ότι είναι δηλαδή συνδεδεμένοι (connected). Η αποθήκευση των δεδομένων γίνεται με τη χρήση της δομής hash. Εάν κάποιος από τους EchoServer αποσυνδεθεί, διαγράφεται πλήρως από το hash.

Οι υπόλοιπες λειτουργίες του Gateway αφορούν την επικοινωνία του με το Probe. Ο Gateway επικοινωνεί με το Probe με τη βοήθεια ενός HTTP request με τη μέθοδο GET, ώστε να του αποστείλει μία λίστα με τις IP διευθύνσεις όλων των Echoservers, που είναι συνδεδεμένοι εκείνη τη στιγμή. Στη συνέχεια, επικοινωνεί άλλη μία φορά με το Probe, αυτή τη φορά με τη μέθοδο POST, ώστε το Probe να του στείλει τη λίστα με τους χρόνους που χρειάστηκαν για κάθε ring. Ο Gateway αποθηκεύει τη λίστα αυτή σε άλλο ένα hash, το οποίο έχει ως key την IP διεύθυνση του συγκεκριμένου Probe. Στην περίπτωση πολλών Probe, δημιουργείται ένα ξεχωριστό hash για το κάθε Probe. Ο Gateway έχει γραφτεί σε γλώσσα προγραμματισμού Python.

3.3 Echoserver

Η δουλειά του Echoserver είναι να συνδέεται με τον Gateway, όποτε αυτό είναι εφικτό, για παράδειγμα όταν γίνεται εκκίνησή του ή όταν αποκατασταθεί η σύνδεση στο διαδίκτυο. Ο Echoserver συνδέεται με τον Gateway μέσω Websockets. Η σύνδεσή τους τερματίζεται μόνο εάν κάποιος από τους 2 σταματήσει να λειτουργεί. Ακόμη, επιτρέπει στα probes να τον κάνουν ring. Για να γίνει αυτό, δεν χρειάζεται η χρήση κάποιας τεχνολογίας, καθώς η χρήση του ring επιτρέπεται εξ' ορισμού από όλα τα λειτουργικά συστήματα. Το ring γίνεται με τη χρήση του πρωτοκόλλου ICMP, δηλαδή με την αποστολή ενός ICMP echo request. Για αυτό το λόγο, το συγκεκριμένο δομικό στοιχείο ονομάστηκε Echoserver. Η υλοποίησή του έγινε σε γλώσσα προγραμματισμού Python.

3.4 Probe

Το Probe είναι ουσιαστικά μία android εφαρμογή. Η υλοποίησή της έγινε με τη χρήση του Android Studio σε γλώσσα προγραμματισμού Java. Το Probe εκτελεί περιοδικά 3 λειτουργίες:

1. Στέλνει ένα HTTP request ώστε να πάρει τη λίστα με τις IP διευθύνσεις των Echoservers.
2. Κάνει ένα ring σε κάθε μία από αυτές τις IP διευθύνσεις και καταγράφει το χρόνο που χρειάστηκε, δηλαδή το RTT.
3. Στέλνει άλλο ένα HTTP request στον Gateway, ώστε να του αποστείλλει τη λίστα με τους χρόνους RTT που κατέγραψε.

Αυτές οι λειτουργίες γίνονται μέσα από 3 Java κλάσεις.

3.4.1 MainActivity.java

Η MainActivity είναι αυτή που φορτώνει πρώτη όταν εκκινείται η εφαρμογή. Αρχικά, ελέγχεται αν η συσκευή είναι συνδεδεμένη στο διαδίκτυο. Εάν δεν είναι, εμφανίζεται ένα μήνυμα που προτρέπει το χρήστη να συνδεθεί στο Internet. Εάν είναι συνδεδεμένη, τότε η οθόνη δείχνει την IP διεύθυνση της συσκευής. Έπειτα, ξεκινά την εκτέλεση ενός Service. Το Service εκτελείται μέσα σε ένα νέο thread, με σκοπό η εφαρμογή να λειτουργεί ομαλά. Εάν η εκτέλεση του Service γινόταν στο κύριο thread, τότε η εφαρμογή δεν θα ήταν ανταποκρίσιμη έως ότου τελειώσουν οι εργασίες του Service.

3.4.2 PingingService.java

Ένα Service είναι ένα στοιχείο που εκτελεί κάποια διεργασία στο παρασκήνιο, χωρίς ο χρήστης να το αντιλαμβάνεται. Το Service συνεχίζει την εκτέλεσή του, ακόμη και αν ο χρήστης ασχολείται με κάποια άλλη εφαρμογή. Το PingingService είναι υπεύθυνο για την περιοδικότητα της εφαρμογής. Αυτό που κάνει είναι να ξεκινά έναν Alarm Manager ανά τακτά χρονικά διαστήματα, ο οποίος με τη σειρά του πυροδοτεί ένα Broadcast Receiver. Ο Alarm Manager μας βοηθάει, ώστε να προγραμματίσουμε την εφαρμογή μας να εκτελέσει μία εργασία σε προκαθορισμένο χρόνο. Στη συγκεκριμένη εφαρμογή, ο Alarm Manager εκτελείται μία φορά μόλις ο χρήστης ανοίξει την εφαρμογή και στη συνέχεια εκτελείται κάθε 10 λεπτά.

3.4.3 ServiceReceiver.java

Ένα Broadcast είναι ένα γεγονός του συστήματος. Στην περίπτωσή μας, το γεγονός αυτό είναι η εκκίνηση του Alarm Manager. Ο Broadcast Receiver είναι ένας τρόπος για να ειδοποιηθεί η εφαρμογή μας για το

γεγονός που συνέβη. Κάθε φορά που συμβαίνει αυτό το γεγονός, ο Broadcast Receiver ξεκινά μία σειρά από άλλα γεγονότα. Αρχικά, στέλνει ένα HTTP request, με τη μέθοδο GET, στον Gateway, με σκοπό να λάβει τις IP διευθύνσεις των Echoservers. Ύστερα, γίνεται ring όλων των Echoservers με τη σειρά. Το ring γίνεται με τη βοήθεια της κλάσης Process. Η κλάση Process παρέχει τη δυνατότητα εκτέλεσης εντολών του λειτουργικού συστήματος, στην προκειμένη το ring. Στη συνέχεια, το RTT απομονώνεται από τα υπόλοιπα ring statistics και αποθηκεύεται σε ένα Hash map, δίπλα από την IP διεύθυνση του αντιστοιχού Echoserver. Τέλος, το Probe κάνει άλλο ένα HTTP request στον Gateway, αυτή τη φορά με την μέθοδο POST, και του στέλνει το Hash map με τους χρόνους των ring. Πριν γίνει το HTTP request, πραγματοποιείται και ένας έλεγχος για να διαπιστωθεί αν το Hash map περιλαμβάνει στοιχεία ή είναι άδειο. Αυτό γίνεται για να καλύψουμε και την περίπτωση της μη ύπαρξης Echoservers, οπότε δεν υπάρχει και λόγος για να γίνει το HTTP request.

3.5 Κώδικας

Ο κώδικας της εφαρμογής μπορεί να βρεθεί στον εξής σύνδεσμο:

<https://github.com/MarianaNameless/NLMSystem>

4. ΑΞΙΟΛΟΓΗΣΗ

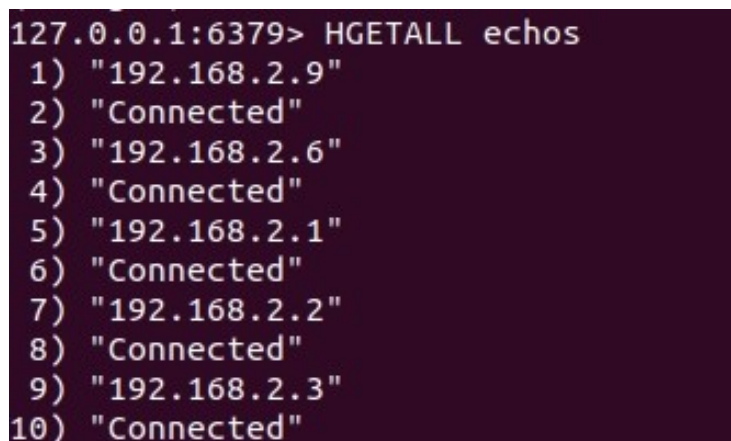
Οι αρχικοί στόχοι που τέθηκαν ως προς την υλοποίηση του συγκεκριμένου συστήματος είναι οι 3 παρακάτω:

1. Το Probe να μπορεί να κάνει ping τους Echoservers.
2. Να παρατηρήσουμε εάν γίνεται μεταβολή στην τιμή του latency από τη μία μέτρηση στην άλλη.
3. Να ανιχνεύσουμε εάν υπήρξε αντικατάσταση του κοντινότερου κόμβου ακμής.

Ας εξετάσουμε τον κάθε στόχο ξεχωριστά.

4.1 Ping των Echoservers

Αρχικά, καλό είναι να εξετάσουμε αν υπάρχουν καταχωρημένοι Echoservers στη redis βάση. Βλέπουμε ότι έχει δημιουργηθεί ένα hash, που περιλαμβάνει τις IP διευθύνσεις από 5 Echoservers, συνοδευόμενες από το status τους, το οποίο είναι Connected, δηλαδή συνδεδεμένοι.



```
127.0.0.1:6379> HGETALL echos
1) "192.168.2.9"
2) "Connected"
3) "192.168.2.6"
4) "Connected"
5) "192.168.2.1"
6) "Connected"
7) "192.168.2.2"
8) "Connected"
9) "192.168.2.3"
10) "Connected"
```

Εικόνα 11: Οι Echoservers

Στην παρακάτω εικόνα, μπορούμε να δούμε ότι έχει δημιουργηθεί άλλο ένα hash, που έχει ως κλειδί την IP διεύθυνση του Probe.

```
127.0.0.1:6379> KEYS *  
1) "echos"  
2) "192.168.2.5"
```

Εικόνα 12: Το Probe

Η ύπαρξη και μόνο του hash σημαίνει ότι τα ring από το Probe προς τους Echoservers έχουν γίνει. Καλύτερο είναι, όμως, να δούμε και το περιεχόμενο του hash.

```
127.0.0.1:6379> HGETALL 192.168.2.5  
1) "192.168.2.9"  
2) "42.521"  
3) "192.168.2.6"  
4) "not responding"  
5) "192.168.2.2"  
6) "945.661"  
7) "192.168.2.1"  
8) "108.552"  
9) "192.168.2.3"  
10) "not responding"
```

Εικόνα 13: Τι περιέχει το hash

Βλέπουμε ότι για 3 από τους 5 Echoservers έχει καταγραφεί το latency σε milliseconds, ενώ για τους άλλους 2 έχει καταχωρηθεί ότι δεν αποκρίθηκαν στο ring, που σημαίνει ότι δεν ήταν διαθέσιμοι εκείνη τη στιγμή. Οστόσο, καταλήγουμε στο συμπέρασμα ότι τα ring πραγματοποιούνται κανονικά. Παρατηρούμε, πως τη δεδομένη χρονική στιγμή, ο Echoserver με το μικρότερο latency είναι ο πρώτος.

4.2 Μεταβολή του latency

Στην Εικόνα 13, βλέπουμε την πρώτη μέτρηση που έγινε από το Probe. Για να διαπιστώσουμε αν υπήρξε κάποια μεταβολή, ας δούμε και την επόμενη μέτρηση.

```
127.0.0.1:6379> HGETALL 192.168.2.5
1) "192.168.2.9"
2) "9.449"
3) "192.168.2.6"
4) "not responding"
5) "192.168.2.2"
6) "370.314"
7) "192.168.2.1"
8) "24.613"
9) "192.168.2.3"
10) "not responding"
```

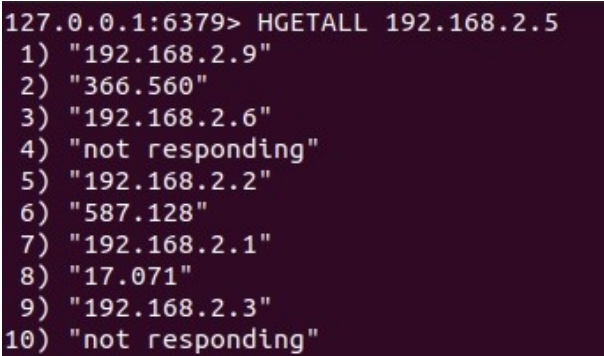
Εικόνα 14: Η δεύτερη μέτρηση

Όντως, μπορούμε να δούμε πως οι τιμές του latency έχουν μεταβληθεί και για τους 3 κόμβους ακμής, ενώ οι άλλοι 2 εξακολουθούν να μην είναι αναποκρίσιμοι. Ο πρώτος κόμβος είναι και πάλι αυτός με τη μικρότερη καθυστέρηση.

4.3 Αντικατάσταση κόμβου ακμής

Για να αντιληφθούμε την αντικατάσταση του κοντινότερου κόμβου ακμής, θα πρέπει σε κάποια επόμενη μέτρηση, κάποιος άλλος Echoserver να έχει τη μικρότερη καθυστέρηση. Όπως είδαμε παραπάνω, στην πρώτη και δεύτερη μέτρηση, ο Echoserver με τη μικρότερη καθυστέρηση ήταν ο

πρώτος. Το ίδιο συνέβη και στην τρίτη μέτρηση. Στην τέταρτη, όμως, μέτρηση αυτό άλλαξε.



```
127.0.0.1:6379> HGETALL 192.168.2.5
1) "192.168.2.9"
2) "366.560"
3) "192.168.2.6"
4) "not responding"
5) "192.168.2.2"
6) "587.128"
7) "192.168.2.1"
8) "17.071"
9) "192.168.2.3"
10) "not responding"
```

Εικόνα 15: Η τέταρτη μέτρηση

Στη Εικόνα 15 βλέπουμε ότι ο τέταρτος Echoserver έχει τη μικρότερη καθυστέρηση, συνεπώς ο κοντινότερος κόμβος ακμής έχει αλλάξει.

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Ο κόσμος γύρω μας αλλάζει καθημερινά λόγω της ταχύτατης ανάπτυξης της τεχνολογίας και των νέων δυνατοτήτων που μας προσφέρει. Ολοένα και περισσότερες συσκευές έχουν δυνατότητα σύνδεσης στο διαδίκτυο, καθώς συλλέγουν δεδομένα από το περιβάλλον τους, συνθέτοντας έτσι αυτό που σήμερα ονομάζουμε Διαδίκτυο των Πραγμάτων. Στόχος τους είναι να απλοποιήσουν εργασίες της καθημερινότητάς μας και να μας προσφέρουν την καλύτερη δυνατή εμπειρία. Όμως, οι συσκευές αυτές αυξάνονται μέρα με τη μέρα και τα δεδομένα καταλαμβάνουν πολύ μεγάλο όγκο. Υπάρχουν περιπτώσεις που η μεταφορά των δεδομένων στο υπολογιστικό νέφος ή σε κάποιο απομακρυσμένο κέντρο δεδομένων μπορεί να έχει καταστροφικές συνέπειες. Η μεταφορά τους έως εκεί μπορεί να πάρει αρκετά μεγάλο χρονικό διάστημα, γεγονός το οποίο σίγουρα θα προκαλέσει τη δυσαρέσκεια του χρήστη. Υπάρχουν και εφαρμογές στις οποίες αυτή η καθυστέρηση είναι και επικύνδινη. Για αυτό το λόγο, στρεφόμαστε πλέον σε ένα άλλο υπολογιστικό μοντέλο, το edge computing, κάνοντας την επεξεργασία των δεδομένων στην ακμή του δικτύου.

Για τους παραπάνω λόγους, δημιουργήθηκε ένα σύστημα το οποίο βοηθά στο να βρεθεί ο πιο κοντινός σε μία συσκευή κόμβος ακμής την εκάστοτε χρονική στιγμή, δίνοντας μας την καθυστέρηση από την συσκευή έως τον κάθε κόμβο ακμής. Επίσης, έχει τη δυνατότητα να ανιχνεύει μεταβολές στην καθυστέρηση των κόμβων, αλλά και εάν συμβαίνει αντικατάσταση του πιο κοντινού κομβού.

Η υλοποίηση αυτού του συστήματος σίγουρα επιδέχεται βελτιώσεις. Αυτή τη στιγμή, η android εφαρμογή δεν έχει κάποια διεπαφή χρήστη, παρά μόνο εκτελεί ένα background service. Θα μπορούσαν να προστεθούν κάποιες

λειτουργίες στην εφαρμογή, ώστε αυτή να γίνει πιο φιλική προς το χρήστη. Θα ήταν χρήσιμο να μπορεί ο χρήστης να βλέπει, μέσα από την εφαρμογή, το latency των Echoservers έπειτα από κάθε ring. Ακόμη, άλλη μία σημαντική λειτουργία, που θα μπορούσε να προστεθεί, είναι να μπορεί ο χρήστης να ορίσει πότε θέλει να γίνουν τα ring, ανεξάρτητα από το χρόνο που ορίζει ο Alarm Manager. Τέλος, καλό θα ήταν η εφαρμογή να υλοποιηθεί και για άλλα λειτουργικά συστήματα, όπως είναι το iOS.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Ranger S. (2020, February 03). What is the IoT? Everything you need to know about the Internet of Thing right now. *ZDNet*. Retrieved from <https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/>
- [2] (2021, February 12). What Are RFID Tags and How Are They Used?. *IDENTIV*. Retrieved from <https://www.identiv.com/community/2021/02/12/what-are-rfid-tags-and-how-are-they-used>
- [3] Fruhlinger J. (2020, May 13). What is IoT? The internet of things explained. *NETWORKWORLD*. Retrieved from <https://www.networkworld.com/article/3207535/what-is-iot-the-internet-of-things-explained.html&prev=search&pto=aue>
- [4] Gillis S. A. (2022, March). What is the internet of things (IoT)?. *TechTarget*. Retrieved from <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>
- [5] Ranger S. (2022, February 25). What is cloud computing? Everything you need to know about the cloud explained. *ZDNet*. Retrieved from <https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/>
- [6] Foote D. K. (2021, December 17). A Brief History of Cloud Computing. *Dataversity*. Retrieved from <https://www.dataversity.net/brief-history-cloud-computing/#>

- [7] Satyabrata J. (2022, May 23). Architecture of Cloud Computing. *GeeksForGeeks*. Retrieved from <https://www.geeksforgeeks.org/architecture-of-cloud-computing/>
- [8] (2018, March 15). Types of cloud computing. *Red Hat*. Retrieved from <https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud>
- [9] Peterson R. (2022, June 4). Advantages and Disadvantages of Cloud Computin. *Guru99*. Retrieved from <https://www.guru99.com/advantages-disadvantages-cloud-computing.html>
- [10] Tolsma A. GDPR and the impact on cloud computing. *Deloitte*. Retrieved from <https://www2.deloitte.com/nl/nl/pages/risk/articles/cyber-security-privacy-gdpr-update-the-impact-on-cloud-computing.html>
- [11] Gillis S. A. (2020, January). Latency. *TechTarget*. Retrieved from <https://www.techtarget.com/whatis/definition/latency?>
- [12] Gibb R. (2019, July 22). What is Latency?. *STACKPATH*. Retrieved from <https://blog.stackpath.com/latency/>
- [13] Notermans T. (2022, May 11). How to measure network latency: the 5 best tools. *Kadiska*. Retrieved from <https://www.kadiska.com/measure-network-latency/>
- [14] Lutkevich B. (2021, March). ICMP (Internet Control Message Protocol). *TechTarget*. Retrieved from <https://www.techtarget.com/searchnetworking/definition/ICMP>

[15] Bigelow J. S. (2021, December). What is edge computing? Everything you need to know. *TechTarget*. Retrieved from <https://www.techtarget.com/searchdatacenter/definition/edge-computing>

[16] Gold J., Shaw K. (2022, May 31). What is edge computing and why does it matter?. *Networkworld*. Retrieved from <https://www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html>

[17] Velimirovic A. (2021, April 14). What is Edge Computing? {Definition, Architecture & Use Cases}. *phoenixNAP*. Retrieved from <https://phoenixnap.com/blog/what-is-edge-computing>

[18] (2022, June 9). WebSocket. *Wikipedia*. Retrieved from <https://en.wikipedia.org/wiki/WebSocket>

[19] Asati A. (2022, February 21). What is web socket and how is it different from the HTTP?. *GeeksForGeeks*. Retrieved from <https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/>

[20] (2020, October 7). What is Redis™? An Overview. *Instaclustr*. Retrieved from <https://www.instaclustr.com/blog/what-is-redis/>

[21] Android – Overview. *Tutorialspoint*. Retrieved from https://www.tutorialspoint.com/android/android_overview.htm

[22] (2022, March 25). Android Studio. *Wikipedia*. Retrieved from https://el.wikipedia.org/wiki/Android_Studio

[23] MDN Contributors. (2022, May 13). HTTP request methods. *MDN web docs*. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>.