Εργαστήριο Πληροφορίας και Δικτύων ΤΗΛ418
Διδάσκων: Ιωάννης Πευκιανάκης

Τελική αναφορά Project WiFi Doctor

Νεαμονιτάκη Ζωγραφούλα Ιωάννα
Καράλης Αστερινός
Νιαροπέτρος Εμμανουήλ

Δευτέρα 26 Μαίου 2025

1 Introduction

Στο πλαίσιο αυτής της εργασίας, σχεδιάστηκε και υλοποιήθηκε μια εφαρμογή SpeedTest βασισμένη σε αρχιτεκτονική client-server με χρήση Unix TCP sockets σε C, η οποία μετρά την απόδοση δικτύου (throughput) μεταξύ AP και server.
Η εφαρμογή επιτρέπει τη μέτρηση του ρυθμού λήψης δεδομένων από τον server σε χρονικά παράθυρα των 2 δευτερολέπτων για συνολική διάρκεια 30 δευτερολέπτων. Επιπλέον, στο τέλος της κάθε μέτρησης υπολογίζεται το μέσο throughput.
Παράλληλα, με τη χρήση του Wi-Fi Doctor, έγινε ανάλυση των πακέτων 802.11 με την βοήθεια μίας τρίτης συσκευής σε monitor mode. Συγκεκριμένα, μελετήθηκαν μεταβλητές όπως το RSSI, το rate gap και η θεωρητική ταχύτητα μετάδοσης (data rate), με σκοπό να εξηγηθούν τυχόν διαφορές ή μειώσεις στην πραγματική απόδοση της εφαρμογής.

2 Design

2.1 Server

Για το Speed Test ζητήθηκε μία υλοποιήση της αρχιτεκτονικής client-server χρησιμοποιοώντας TCP Unix Sockets σε γλώσσα προγραμματισμού C. Αρχικά υλοποιήθηκε ο Server σε γλώσσα C. Το πρώτο πράγμα που γίνεται σε έναν server είναι να δημιουργηθεί το socket, το οποίο ρυθμίζεται να χρησιμοποιεί δέχεται IPv4 (`AF_INET`) και TCP (`SOCK_STREAM`) πρωτόκολλα. Στην συνέχεια το socket πρέπει να δεσμευτεί (bind) σε μια μία συγκεκριμένη IP διεύθυνση και θύρα. Αν τρέχουμε το demo τοπικά, τότε χρησιμοποίειται η τοπική IP (localhost), αλλιώς χρησιμοποιείται η IP του υπολογιστή στον οποίο εκτελείται ο server. Στην περίπτωση μας επιλέξαμε την θύρα 5555. Αφού πάρουμε τις απαραίτητες πληροφορίες για IP και port, κάνουμε bind το socket. Έπειτα ο server πρέπει να ακούει στο port που έγινε bind, με την εντολή listen για εισερχόμενες συνδέσεις. Όταν ο server δεχτεί μια καινούργια σύνδεση με την εντολή accept τότε το system call θα μπλοκαριστεί μέχρι να έρθει ένας άλλος client. Αφού συνδεθεί, δημιουργείται νέο socket για την επικοινωνία με τον συγκεκριμένο client. Την ώρα της εξυπηρέτησης πραγματοποιείται το Speed Test. Ο client στέλνει συνεχώς δεδομένα στον server, τόσα ώστε το socket να είναι γεμάτο συνέχεια. Ανά δύο δευτερόλεπτα υπολογίζεται το throughput από τον server, με βάση τον τύπο:

$$\text{throughput} = \frac{\text{bytes} \times 8.0}{\text{seconds} \times 1\,000\,000.0}$$

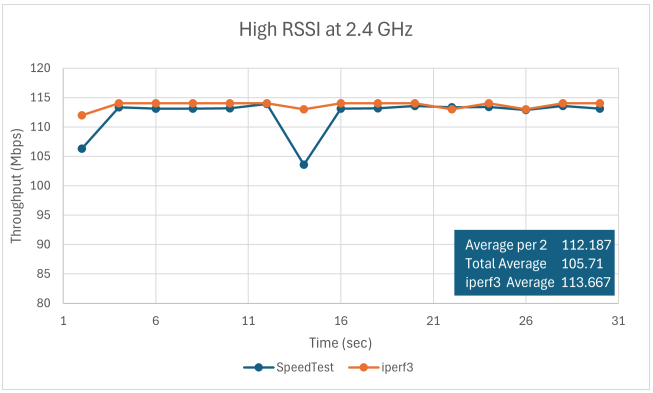
Κάθε πείραμα Speed Test διαρκεί 30 δευτερόλεπτα, στα οποία κάθε 2 δευτερόλεπτα εκτυπώνονται στο terminal με πόση ταχύτητα ήρθαν τα δεδομένα από τον client. Στην εφαρμογή μας ο client στέλνει στον server. Όταν τελειώσει η σύνδεση με τον client κλείνει ο server το socket του client και παραμένει ανοιχτός για να ακούσει επόμενες συνδέσεις.

2.2 Client

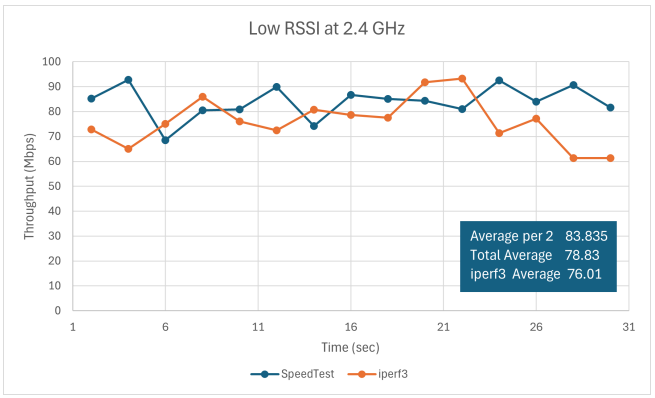
Ο client επίσης δημιουργεί ένα TCP socket, το οποίο μπορεί να συνδεθεί στον server μέσω της connect(). Η διεύθυνση IP για τοπικές δοκιμές είναι η localhost, αλλιώς χρησιμοποείται η διεύθυνση του server. Η θύρα που χρησιμοποιείται είναι η 5555 όπως και στον server. Μετά την επιτυχή σύνδεση με τον server, ο client ξεκινά να στέλνει συνεχώς δεδομένα προς τον server. Το buffer αποστολής έχει μέγεθος 64 KB, το οποίο γεμίζεται με χαρακτήρες 'A' πριν αποσταλεί. Η αρχική επιλογή αποστολής δεδομένων ήταν για τον client(write buffer) να είναι 16,384 Bytes και για τον server(read buffer) να είναι 131,072 Bytes. Οι επιλογές αυτές βασίστηκαν στις τιμές που επιστρέφουν οι εντολές `cat /proc/sys/net/ipv4/tcp_wmem` και `cat /proc/sys/net/ipv4/tcp_rmem`, οι οποίες παρέχουν αντίστοιχα το ελάχιστο, προεπιλεγμένο και μέγιστο μέγεθος των TCP buffers για αποστολή (write) και λήψη (read) δεδομένων σε συνδέσεις IPv4. Όμως στην συνέχεια με τα πειράματα διαπιστόθηκε πως το μέγεθος των 64 KBytes είναι αυτό που δίνει τα καλύτερα αποτελέσματα σε σχέση με το έτοιμο εργαλείο iperf3. Η αποστολή δεδομένων γίνεται συνέχεια για να έχουμε συνεχή ροή δεδομένων στον buffer. Το throughput υπολογίζεται και στην μεριά του client με τον ίδιο τρόπο όπως στον server, δηλαδή ανά δύο δευτερόλεπτα. Μόλις τελειώσει η περίοδος των 30 δευτερολέπτων υπολογίζεται το συνολικό throughput και κλείνει το socket.

3 Evaluation

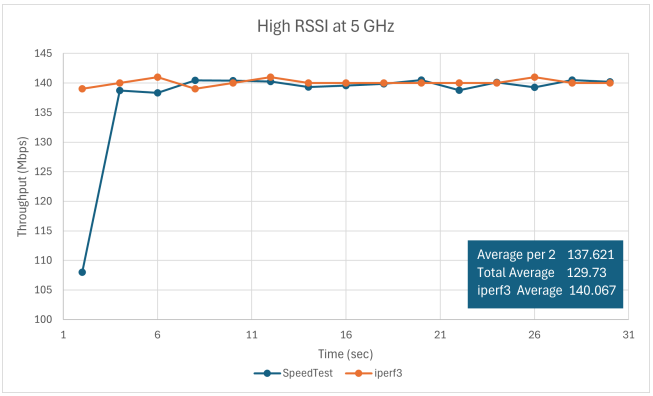
Αρχικά διεξάχθηκε το πείραμα με 3 διαφορετικές συσκευές. Ο client ήταν σταθερός υπολογιστής με IP: 192.168.1.3 με ethernet σύνδεση στο internet. Ο server ήταν φορητός υπολογιστής με IP: 192.168.1.6 και τον μετακινούσαμε ανάλογα με το πείραμα. Τέλος υπήρχε και μία τρίτη συσκευή η οποία ήταν το Wi-Fi Sniffer, ήταν σε monitor mode και έκανε sniff τα πακέτα μεταξύ του AP και του server. Μόλις τελειώνε κάθε πείραμα, γινόταν και πείραμα με το εργαλείο iperf3 με τις ίδιες ρυθμίσεις όπως του αρχικού πειράματος. Τα πακέτα που λάμβανε το sniffer στα 2.4 GHz ήταν 802.11n και στα 5 GHz ήταν 802.11ac. Έγιναν οι καταλλήλες αλλαγές μέσα από το web interface, έτσι ώστε στα 2.4 GHz να στέλνονται πακέτα μόνο στο κανάλι 1 και στα 5 GHz μόνο στο κανάλι 48. Επίσης και στα 2.4 GHz και στα 5 GHz ορίσαμε το bandwidth να είναι στα 20 MHz.
Σύγκριση του Speed Test μας με το εργαλείο iperf3:



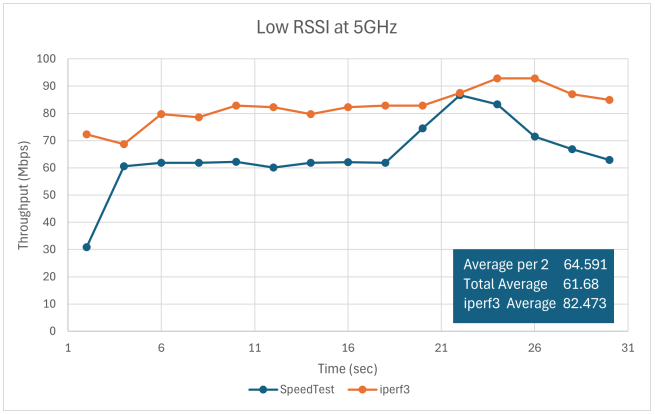
(a) Throughput SpeeTest vs iperf3, High RSSI at 2.4 GHz



(b) Throughput over time, Low RSSI at 2.4 GHz



(c) Throughput over time, High RSSI at 5 GHz



(d) Throughput over time, Low RSSI at 5 GHz

Παρατηρούμε πως στις 2 περιπτώσεις όπου ο server είναι σταθερός και κοντά στο router (High RSSI) σε 2.4 και 5 GHz το εργαλείο iperf3 έχει αρκετά παρόμοιο throughput με το δικό μας app. Εκτός από μερικές τιμές οι οποίες φαίνονται να είναι outliers, στα σενάρια του μεγάλου RSSI, τα throughput φαίνονται να κυμαίνονται στις ίδιες τιμές. Για τις average τιμές παρατηρήσαμε πως το iperf3 υπολογίζει το average throughput προσθέτοντας το average ανά 2 δευτερόλεπτα κι διαιρώντας το με τον συνολικό χρόνο, δηλαδή 30 δευτερόλεπτα. Για αυτό υπολογίζουμε κι εμείς το ίδιο average, που το ονομάζουμε average per 2, το οποίο φαίνεται να είναι αρκετά κοντά στον υπολογισμό του μέσου throughput του iperf3. Το δικό μας συνολικό average (total average) είναι τα δεδομένα που έχει κάνει receive ο client σε πάροδο 30 δευτερολέπτων, το οποίο φαίνεται να είναιι πιο κοντά στο average του iperf3 για χαμηλά RSSI.
Για τα σενάρια στα οποία ήμασταν μακριά από το router φαίνεται πως οι τιμές σε σύγκριση με το iperf3 διαφέρουν, ιδιαίτερα στα 5 Ghz. Επίσης παρατηρούμε πως ενώ στα 5 GHz, κοντά στο router έχουμε υψηλότερο throughput (140 Mpbs) από το 2.4 GHz (112 Mpbs), μακριά από το router στα 5 GHz έχουμε χαμηλότερο throughput (64 Mbps) από ότι στα 2.4 GHz (80 Mbps). Αυτό συμβαίνει διότι στα 5 GHz η εξασθένηση σήματος είναι μικρή όταν ήμαστε κοντά στον router, αλλά όταν αρχίζουμε να απομακρυνόμαστε εξασθενεί πιο γρήγορα το σήμα με την απόσταση και επηρεάζεται πιο έντονα από τοίχους και εμπόδια. Αντίθετα το 2.4 GHz έχει καλύτερη διάδοση μέσα από τοίχους και εμβέλεια, οπότε διατηρείται καλύτερο σήμα όταν απομακρυνόμαστε.

4 Wi-Fi Doctor Analysis

Στο παρόν στάδιο αξιοποιήθηκε το εργαλείο που είχε αναπτυχθεί στο προηγούμενο project, το οποίο τροποποιήθηκε κατάλληλα ώστε να υποστηρίζει την εκτύπωση των απαιτούμενων μετρικών όπως αυτές ζητούνταν στην εκφώνηση.

- Throughput (as estimated by the Wi-Fi Doctor).
- Data Rate.
- Frame Loss.
- RSSI.
- Rate Gap.

4.1 Throughput-Estimation

Ο αρχικός θεωρητικός τύπος για το throughput, με βάση το project του wifi doctor ήταν: $Throughput = Data\_Rate * (1 - Frame\_Loss\_Rate)$. Για να γίνει η εκτίμηση του throughput πιο ρεαλιστική και πιο κοντά στις συνθήκες ενός πραγματικού περιβάλλοντος ασύρματης επικοινωνίας, στον αρχικό τύπο προστέθηκαν τρεις επιπλέον παράγοντες: το channel utilization (με τη μορφή του RSSID), οι overheads των headers (μέσω του payload efficiency), και το rate gap (δηλαδή η απόκλιση από τον αναμενόμενο ρυθμό μετάδοσης λόγω κακών συνθηκών ή παρεμβολών).

Ο τελικός εμπλουτισμένος τύπος γίνεται:

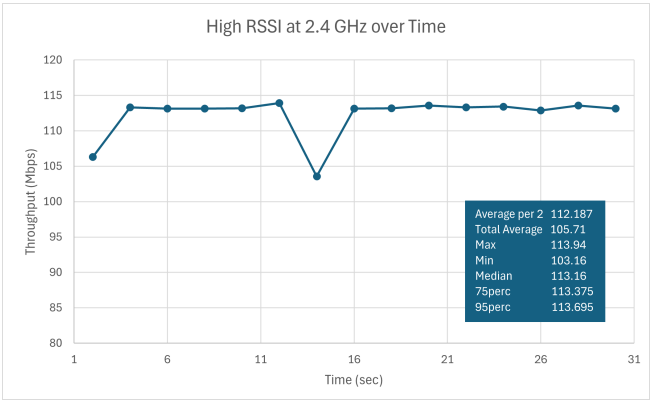
$$Throughput = Data\_Rate * (1 - Frame\_Loss\_Rate) * Rate\_Gap\_Penalty * Utilization\_Penalty * Payload\_Efficiency$$

Τα τρία αυτά κριτήρια αναλύονται παρακάτω:

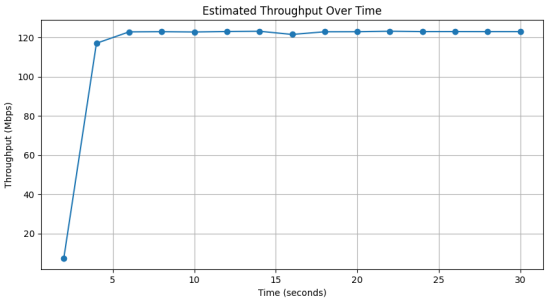
- `Utilization Penalty (RSSID)`: Το RSSID εκτιμάει τη συμφόρηση του καναλιού. Όσο υψηλότερη η τιμή, τόσο περισσότερες συσκευές χρησιμοποιούν το κανάλι. Εφόσον το RSSID αυξάνεται, το διαθέσιμο bandwidth για τη συσκευή μειώνεται. Έτσι, εισάγεται ο πρωτος συντελεστής μείωσης στον τύπο του throughput:
  $Utilization\_Penalty = 1 - min(RSSID, 1.0)$
- `Payload Efficiency`: Ο συνολικός ρυθμός μετάδοσης δεν μεταφράζεται πλήρως σε ωφέλιμα δεδομένα, καθώς τα πακέτα περιέχουν headers (radiotap, wlan headers κ.ά.). Ο λόγος του payload προς το συνολικό frame length υπολογίζει την απόδοση σε επίπεδο χρήσιμων δεδομένων. Αν, για παράδειγμα, τα headers είναι μεγάλα και το payload μικρό, η αποτελεσματικότητα μειώνεται.
  $Payload\_Efficiency = Average\_Frame\_Length / Average\_Payload$
- `Rate Gap Penalty`: Το rate gap δείχνει κατά πόσο το πραγματικό MCS index αποκλίνει από το αναμενόμενο, με βάση το σήμα (RSSI) και τον αριθμό spatial streams. Αν η συσκευή μεταδίδει με μικρότερο ρυθμό από αυτόν που θα μπορούσε, τότε αυτό θεωρείται υπό-απόδοση. Η ποινή αυτή υλοποιείται ως συνάρτηση:
  $Penalty = max(0.3, 1 - 0.07 * Rate\_Gap)$
  Η χρήση του 0.07 ως συντελεστής προήλθε εμπειρικά ώστε να επιβάλλεται αυστηρή ποινή αφού ασχολούμαστε με real-time εφαρμογές.

Με την προσθήκη αυτών των συντελεστών, η εκτίμηση του throughput γίνεται πιο ρεαλιστική, καθώς λαμβάνει υπόψη παρεμβολές, υπο-απόδοση λόγω MCS και overhead από headers. Οι τιμές του υπολογισμένου throughput είναι συνήθως λίγο χαμηλότερες από τα data rates — ειδικά όταν δεν υπάρχει frame loss. Ωστόσο, όπως φαίνεται και στα αποτελέσματα που ακολουθούν, σε ορισμένες περιπτώσεις το τελικό throughput παραμένει υψηλό σε σχέση με το πραγματικό, γεγονός που ίσως δείχνει πως οι ποινές θα μπορούσαν να είναι λίγο πιο αυστηρές.
Αναλύοντας τα `.pcapng` αρχεία, όπως αυτά προέκυψαν από τα πέντε διαφορετικά σενάριαφ με την χρήση του Wi-Fi Doctor, προκύπτουν οι παρακάτω γραφικές:

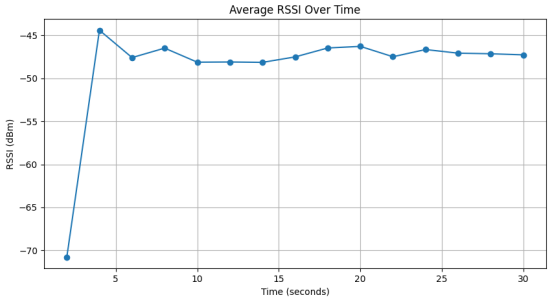
4.2 High RSSI at 2.4 GHz



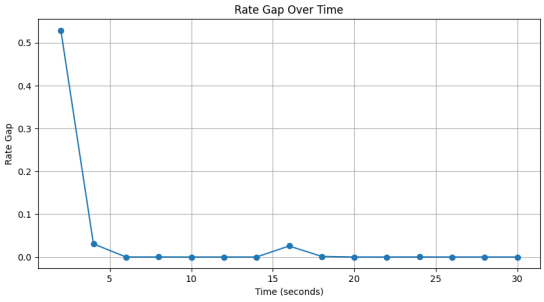
(a) Throughput SpeedTest



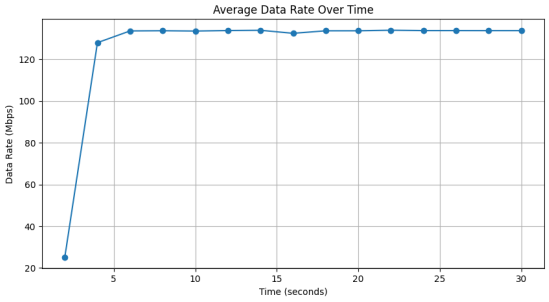
(b) Throughput over time



(c) RSSI over time



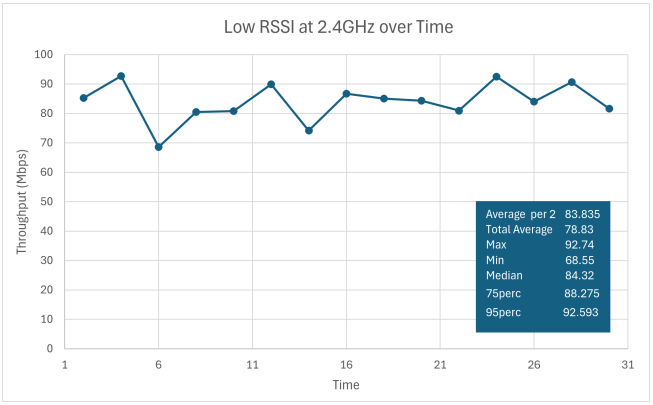
(d) Rate gap over time



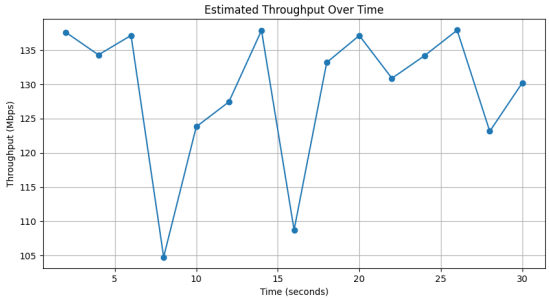
(e) Data Rate over time

Παρατηρούμε πως το throughput είναι σταθερό στα 112 Mbps στο πραγραματικό πείραμα, (με ένα outlier που πέφτει στα 103 Mpbs). Η σταθερότητα είναι λογική καθώς ο server ήταν δίπλα στο router χωρίς καθόλου να μετακινείται. Από την ανάλυση του Wi-Fi Doctor βλέπουμε πως το througphut μένει σταθερό στα 120 Mbps, το οποίο είναι λογικό να είναι μεγαλύτερο στο Wi-Fi Doctor καθώς είναι θεωρητική η ανάλυση. Βλέπουμε επίσης πως το rate gap δεν αλλάζει και παραμένει σταθερό στο 0, καθώς και το RSSI μένει σταθερό κατά την διάρκεια του πειράματος.

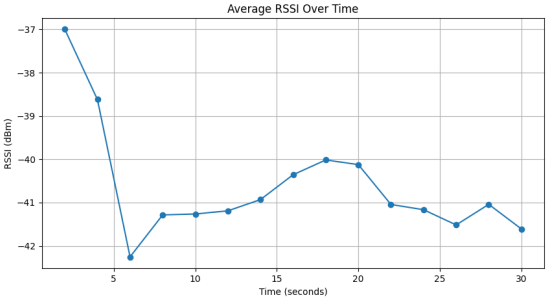
4.3 Low RSSI at 2.4 GHz



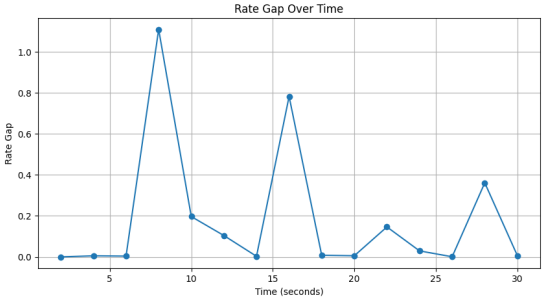
(a) Throughput SpeedTest



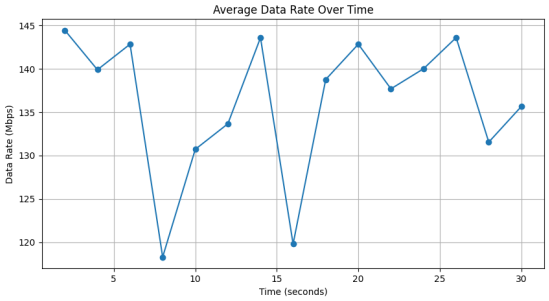
(b) Throughput over time



(c) RSSI over time



(d) Rate gap over time

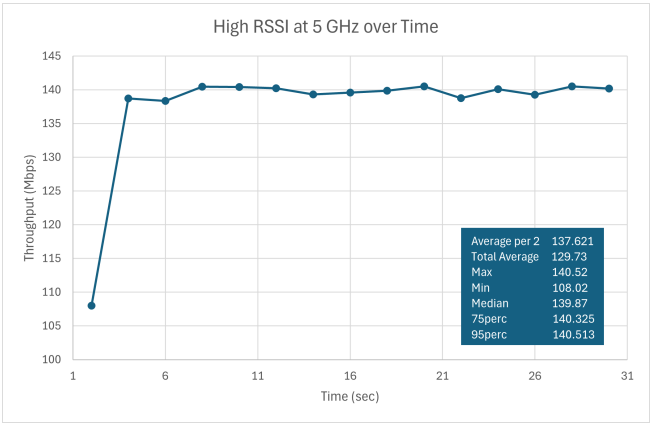


(e) Data Rate over time

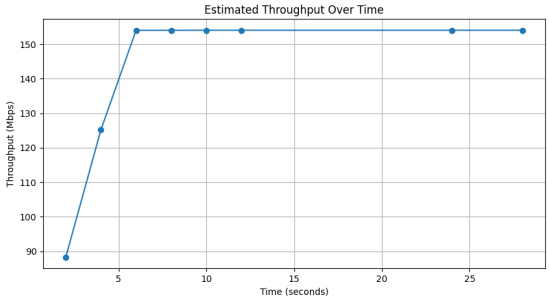
Παρατηρούμε πως το throughput μεταβάλλεται μεταξύ 92.74 Mbps και 68.55 Mbps στο πείραμα ενώ στο WiFi-Doctor οι θεωρητικές τιμές κυμαίνονται μεταξύ περίπου 134 Mpbs και 105 Mpbs. Και στις δύο γραφικές παταρηρούμε ότι το throughput έχει παρόμοια συμπεριφορά καθώς αυξάνεται και μειώνεται σχεδόν στα ίδια σημεία (Η γραφική του θεωρητικού throughput φαίνεται πιο απότομη λόγω της διαφορετικής κλίμακας). Συγκρίνοντας τις τιμές, βλέπουμε ότι υπάρχει σχετική διαφορά μεταξύ των πραγματικών τιμών και αυτών που υπολογίσαμε με τον τύπο μας περίπου κατά 30 Mbps πιο πάνω οι θεωριτικές τιμές. Αυτό το αποτέλεσμα πιθανώς οφείλεται στον θεωρητικό τύπο που χρησιμοποιήθηκε για τον υπολογισμό του throughput, και πιο συγκεκριμένα σε μη βέλτιστη ρύθμιση των παραμέτρων που επηρεάζουν την τελική του τιμή.

Ακόμη παρατηρούμε ότι όταν το `RSSI` μειώνεται, βλέπουμε ότι στις αντίστοιχες χρονικές στιγμές το Rate Gap αυξάνεται (μειώνεται το MCS Index), με αποτελέσμα το Data Rate επίσης να μειώνεται ώστε να μην έχουμε retries.

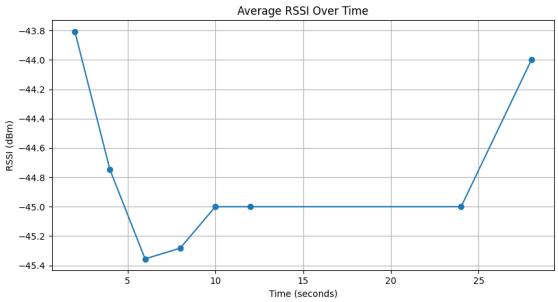
4.4 High RSSI at 5 GHz



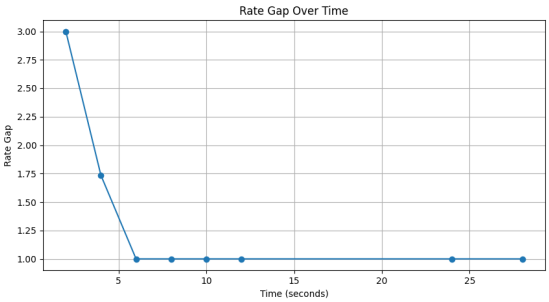
(a) Throughput SpeedTest



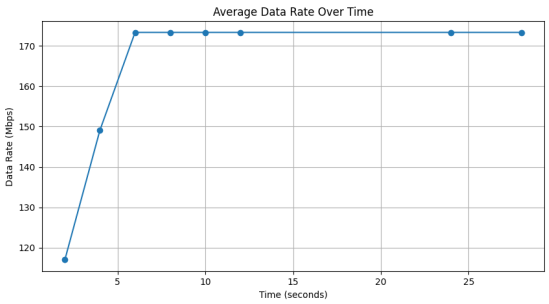
(b) Throughput over time



(c) RSSI over time



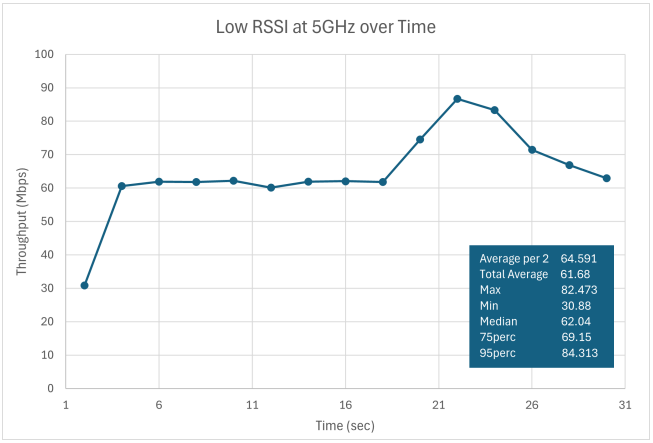
(d) Rate gap over time



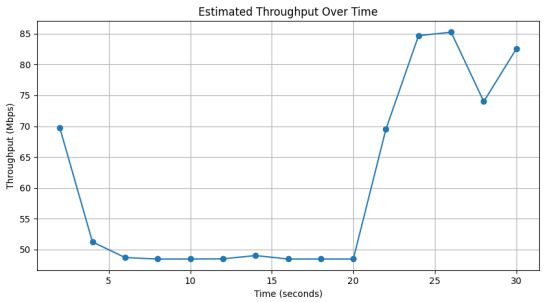
(e) Data Rate over time

Με βάση τα παραπάνω, παρατηρούμε πως και στην πραγματική μέτρηση μέσω Speedtest αλλά και στην εκτίμηση του throughput από τον WiFi Doctor, τα αποτελέσματα είναι αρκετά παρόμοια. Το throughput στο speedtest μένει σταθερό στα 140 mbps(με έναν outlier στα 107 Mbps) ενώ στο wifi doctor, παραμενει σταθερό στα 152 mbps. Παρατηρούμε επίσης πως το σήμα (RSSI) παραμένει σχετικά υψηλό και γι'αυτο το rate gap παραμένει σταθερο κοντά στο 1. Παρ' όλα αυτά, βλέπουμε πως το data rate φαίνεται στην πραγματικότητα αρκετά πιο υψηλό (170 mbps), το οποίο σημαίνει πως ο τύπος του throughput που χρησιμοποιήσαμε κατάφερε να φέρει σε πιο ρεαλιστικά επίπεδα τις τιμές του.

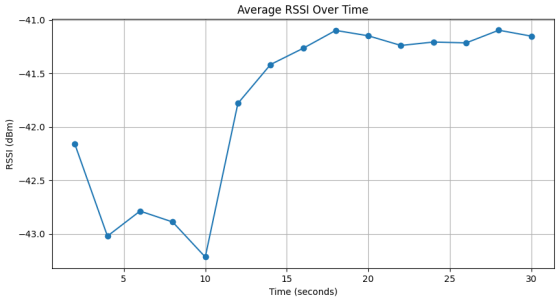
4.5 Low RSSI at 5 GHz



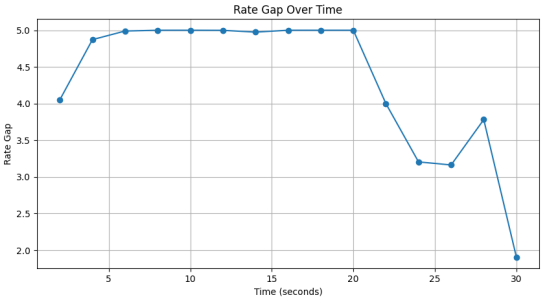
(a) Throughput SpeedTest



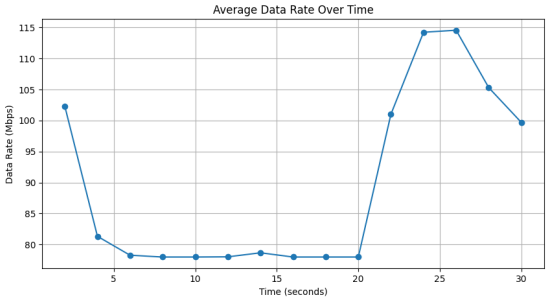
(b) Throughput over time



(c) RSSI over time



(d) Rate gap over time

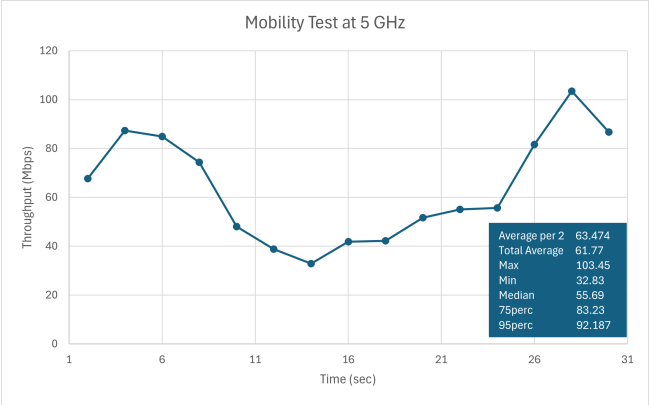


(e) Data Rate over time

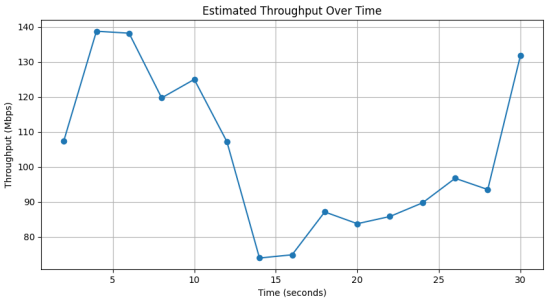
Παρατηρούμε πως το throughput μεταβάλλεται μεταξύ 82.47 Mbps και 30.88 Mbps στο πείραμα ενώ στο WiFi-Doctor οι θεωρητικές τιμές κυμαίνονται μεταξύ περίπου 85 Mpbs και 47 Mpbs. Και σε αυτό το σενάριο, στις δύο γραφικές παταρηρούμε ότι το throughput έχει παρόμοια συμπεριφορά. Η σχετική διαφορά μεταξύ των πραγματικών τιμών και αυτών που υπολογίσαμε με τον τύπο μας περίπου 20 Mbps πιο πάνω οι θεωρητικές τιμές. Και εδώ το αποτέλεσμα πιθανώς οφείλεται στον θεωρητικό τύπο που χρησιμοποιήθηκε για τον υπολογισμό του throughput.

Παρατηρούμε επίσης ότι στην αρχή και στο τέλος της προσομοίωσης, η ισχύς του σήματος (RSSI) αυξάνεται. Επομένως το Rate Gap μειώνεται με αποτέλεσμα το Data Rate (και άρα το throughput) να αυξάνεται. Τέλος μπορούμε να δούμε πως το throughput είναι μικρότερο σε Low RSSI στα 5 GHz από ότι στα στα 2.4 GHz, το οποίο επαληθεύεται και από το Wi-Fi Doctor. Οι λόγοι που συμβαίνει αυτό έχουν αναλυθεί παραπάνω στα πλαίσια του Evaluation.

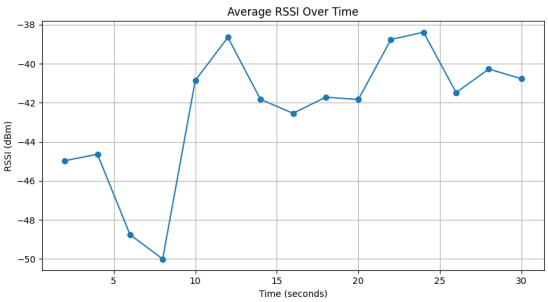
4.6 Mobility Scenario at 2.4GHz



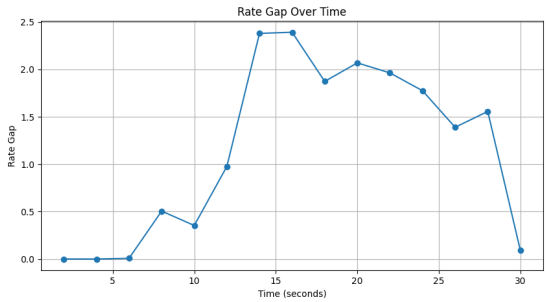
(a) Throughput SpeedTest



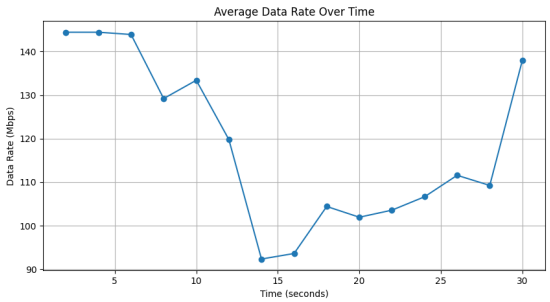
(b) Throughput over time



(c) RSSI over time



(d) Rate gap over time



(e) Data Rate over time

Σε αυτό το σενάριο, βλέπουμε στο πείραμα το throughput να αυξομειώνεται ανάμεσα στις τιμές 103 Mbps και 32 Mbps, το οποίο είναι λογικό, εφόσον όταν άρχισε το speed test και η καταγραφή πακέτων ξεκινήσαμε κοντά στο router και κάπου στα 15 δευτερόλεπτα (όπου βλέπουμε και την Min τιμή) περπατώντας βρισκόμασταν αρκετά μακριά από το AP. Μετά τα 15 δευτερόλεπτα ξανά περπατήσαμε δίπλα στο AP όπου φαίνεται πως το throughput έπεστεψε στην τιμή που ήταν αρχικά. Παρόμοια συμπεριφορά έχει και το throughput του Wi-Fi Doctor μόνο που η τιμή του είναι 30-40 Mbps πιο πάνω από το πραγματικό πείραμα. Έχουμε αναλύσει παραπάνω τους λόγους που συμβαίνει αυτό, αλλά το κομμάτι που είναι σημαντικό να εστιάσουμε είναι το rate gap και το RSSI τα οποία αυξομειώνονται σε παρόμοιες χρονικές στιγμές με το throughput. Όπως φαίνεται και στις γραφικές οι απόσταση επηρεάζει το RSSI με αποτέλεσμα όσο πιο μακριά από το AP ήμαστε τόσο μειώνεται το RSSI και το rate gap. Το RSSI φαίνεται να επηρεάζεται και από την κίνηση καθώς κάπου στα 15 δευτερόλεπτα που σταματήσαμε για λίγο να περπατάμε φαίνεται να σταθεροποιείται. Όμως την ώρα που απομακρυνόμασταν από το AP φαίνεται πως μειώθηκε το RSSI. Το παραπάνω πείραμα έγινε στα 2.4 GHz. Αν γινόταν στα 5 GHz πολύ πιθανότατα να είχαμε πιο εμαφνής αλλαγές στο RSSI λόγο της πιο απότομης εξασθένησης του σήματος. Το rate gap αλλάζει λόγο του MCS index εφόσον το router προσπαθεί να βρει ένα data rate που να μην έχει τόση ισχύ έτσι ώστε να μην υπάρχουν πολλά retries.

5 References

- Socket Buffer Size Linux:
  https://stackoverflow.com/questions/7865069/how-to-find-the-socket-buffer-size-of-linux
- Throughput Measuring: https://en.wikipedia.org/wiki/Measuring_network_throughput
- IEEE 802.11:https://en.wikipedia.org/wiki/IEEE_802.11
- MCS Table:https://www.watchguard.com/help/docs/help-center/en-US/Content/en-US/WGCloud/Devices/access_point/deployment_guide/Checklist_Charts/mcs_80211n_ac.pdf
- Pefkianakis et al. "Characterizing Home Wireless Performance: The Gateway View", IEEE INFOCOM 2015.
- TCP Server-Client implementation in C:
  https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
- Using the Berkeley Socket API in the C Programming Language https://www.youtube.com/watch?v=U28svzb1WUs