ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ, ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Αποδοτική Ανίχνευση Δικτυακών Απειλών σε Κρυπτογραφημένη Κίνηση με Πιθανοτικές Δομές Δεδομένων και Αλγορίθμους Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Ψαλλίδας

Επιβλέπων : Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ, ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αποδοτική Ανίχνευση Δικτυακών Απειλών σε Κρυπτογραφημένη Κίνηση με Πιθανοτικές Δομές Δεδομένων και Αλγορίθμους Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Ψαλλίδας

Επιβλέπων : Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 4η Φεβρουαρίου 2025.

.....	.....	.....
Συμεών Παπαβασιλείου	Βασίλειος Μάγκλαρης	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Ομότιμος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2025

.....
Χρήστος Ψαλλίδας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η παρούσα εργασία έχει σκοπό την αξιολόγηση της χρήσης πιθανοτικών δομών δεδομένων στην ανίχνευση κακόβουλης κρυπτογραφημένης δικτυακής κίνησης σε πραγματικό χρόνο. Για την επίτευξη του στόχου δημιουργήθηκαν δύο προγραμματιστικές υλοποιήσεις, μία που αξιοποιεί ορισμένες πιθανοτικές δομές και μία αμιγώς ντετερμινιστική. Χρησιμοποιήθηκε dataset που περιλαμβάνει καταγραφές με ποικιλία σχετικής κίνησης, κακόβουλης και καλόβουλης. Από τα κρυπτογραφημένα πακέτα εξάγονται ορισμένα χαρακτηριστικά (μήκος πακέτου, χρόνος, κατεύθυνση), τα οποία ομαδοποιούνται ανά ροή πακέτων, και υπολογίζονται με βάση αυτά συγκεκριμένες στατιστικές ιδιότητες. Στην πιθανοτική υλοποίηση αξιοποιούνται για την αποθήκευση των δεδομένων οι κατάλληλες πιθανοτικές δομές, συγκεκριμένα T-Digest, Top-K και Count Min Sketch. Οι δομές αυτές είναι κατάλληλες για αποθήκευση δεδομένων ροής, απαιτούν σταθερή μνήμη ανεξάρτητη από τον όγκο δεδομένων, και οι χρόνοι εισαγωγής ενός στοιχείου και εξαγωγή μίας μέτρησης από αυτές γίνονται σε σταθερό χρόνο. Κάθε δομή προσφέρει κάποιο από τα απαιτούμενα στατιστικά. Στην ντετερμινιστική υλοποίηση τα χαρακτηριστικά που εξάγονται από τα πακέτα αποθηκεύονται αυτούσια σε λίστες, και τα στατιστικά εξάγονται από αυτές με τη χρήση κλασσικών μαθηματικών συναρτήσεων. Τα στατιστικά που προκύπτουν για κάθε ροή πακέτων οδηγούνται στο στάδιο πρόβλεψης, όπου κάθε ροή κρίνεται είτε κακόβουλη είτε καλόβουλη. Η πρόβλεψη γίνεται από μοντέλα μηχανικής μάθησης, που έχουν εκπαιδευτεί εκ των προτέρων χρησιμοποιώντας ένα υποσύνολο των ροών πακέτων από τις καταγραφές του dataset. Η αξιολόγηση γίνεται με σύγκριση των δύο υλοποιήσεων ως προς τους τομείς ακρίβειας προβλέψεων, χρησιμοποίησης μνήμης και ρυθμού επεξεργασίας.

Λέξεις-κλειδιά:

Πιθανοτικές δομές, κρυπτογραφημένη κίνηση, ασφάλεια δικτύων, DNS, DNS tunneling, DoH, HTTPS, ταξινόμηση, ακρίβεια, μοντέλα μηχανικής μάθησης, μνήμη, απόδοση

Abstract

This thesis aims to evaluate the use of probabilistic data structures for detecting malicious encrypted network traffic in real time. To achieve this goal, two implementations were developed: one utilizing specific probabilistic structures and another purely deterministic. A dataset containing logs of both malicious and benign network traffic was used. From the encrypted packets, certain features (packet length, time, direction) are extracted, then grouped by packet flow, and specific statistical properties are computed based on them. In the probabilistic implementation, appropriate probabilistic data structures are used for data storage, specifically T-Digest, Top-K, and Count Min Sketch. These structures are well-suited for streaming data, require fixed memory independent of data volume, and allow for constant-time insertion of elements and retrieval of measurements. Each structure provides some of the required statistics. In the deterministic implementation, the extracted packet features are stored directly in lists, and statistics are derived from them using classical mathematical functions. The statistics generated for each packet flow are then passed to a prediction stage, where each flow is classified as either malicious or benign. The prediction is performed by machine learning models pre-trained on a subset of packet flows from the dataset logs. Evaluation is carried out by comparing the two implementations in terms of prediction accuracy, memory usage, and processing rate.

Keywords:

Probabilistic data structures, encrypted traffic, cybersecurity, DNS, DNS tunneling, DoH, HTTPS, classification, accuracy, machine learning models, memory, efficiency

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω τους γονείς μου και όλη μου την οικογένεια για την στήριξη που μου παρείχαν καθ' όλη την διάρκεια των σπουδών μου.

Θερμές ευχαριστίες αρμόζουν επίσης στον καθηγητή κ. Βασίλειο Μάγκλαρη, που με καλωσόρισε στο εργαστήριο NETMODE ώστε να εκπονήσω την εργασία.

Τέλος, ευχαριστώ τον διδάκτορα Νίκο Κωστόπουλο για την ουσιαστική στήριξη που μου παρείχε σε όλα τα στάδια της διπλωματικής, βοηθώντας καθοριστικά στην ολοκλήρωση του έργου.

Περιεχόμενα

Περιεχόμενα σχημάτων

Κεφάλαιο 1

Εισαγωγή

Στην εποχή της πληροφορίας, η ποσότητα δεδομένων που ανταλλάσσεται στο διαδίκτυο αυξάνεται συνεχώς, με πρωτόγνωρους ρυθμούς. Δισεκατομμύρια άνθρωποι ανά τον κόσμο αξιοποιούν το δίκτυο στέλνοντας και λαμβάνοντας υλικό κάθε είδους. Το εμπόριο, η ψυχαγωγία, η ενημέρωση και η επικοινωνία είναι λίγες μόνο από τις πτυχές της ζωής μας που έχουν απλοποιηθεί σε μεγάλο βαθμό χάρις στον παγκόσμιο ιστό, ο οποίος λειτουργεί ως μία εκτενής, ελεύθερη σε όλους, κοινότητα.

Όμως όταν η ελευθερία παρέχεται χωρίς περιορισμούς, ελλοχεύει ο κίνδυνος να γίνει αντικείμενο κατάχρησης. Ορισμένοι χρήστες της κοινότητας αξιοποιούν τα εγγενή χαρακτηριστικά της όπως η ανωνυμία και η ευκολία χρήσης της για να εξυπηρετήσουν κακόβουλους σκοπούς. Παραδείγματα τέτοιων σκοπών είναι η κλοπή ευαίσθητων προσωπικών δεδομένων, το παράνομο εμπόριο, ή ακόμα και τρομοκρατία. Η παγκοσμιοποιημένη φύση του διαδικτύου καθιστά επίσης πολύ δύσκολη την παρακολούθηση αυτών των δραστηριοτήτων και την ανάληψη ευθυνών, κάτι που καθιστά την πρόληψη των επιθέσεων ακόμα πιο απαιτητική.

Ως εκ τούτου, προβάλλει ως αδήριτη ανάγκη η εγγύηση της ασφάλειας στον κυβερνοχώρο. Η προστασία των δεδομένων και των επικοινωνιών είναι κρίσιμη, όχι μόνο για την αποτροπή κακόβουλων επιθέσεων, αλλά και για την εμπιστοσύνη που απαιτείται για τη συνεχιζόμενη ανάπτυξη του διαδικτύου ως εργαλείου για τη διαχείριση της καθημερινότητας. Η εξασφάλιση της ιδιωτικότητας και της ακεραιότητας των δεδομένων καθίσταται πιο σημαντική από ποτέ, καθώς οι απειλές εξελίσσονται συνεχώς και οι επιθέσεις γίνονται όλο και πιο σύνθετες.

Μία από τις μεθόδους που χρησιμοποιούνται ευρύτατα για την προστασία των προσωπικών δεδομένων είναι η κρυπτογραφία. Είναι δυνατή η εγκατάσταση σύνδεσης μεταξύ πελάτη και εξυπηρετητή με κατάλληλο τρόπο ώστε να είναι πρακτικά αδύνατη η αθέμιτη παρακολούθηση της επικοινωνίας από οποιονδήποτε τρίτο. Επομένως, οι χρήστες μπορούν να στέλνουν και να λαμβάνουν πληροφορία γνωρίζοντας ότι κανένας δεν μπορεί να την αποσπάσει ή να την χρησιμοποιήσει. Όμως, αυτή η τεχνική δεν αποτελεί πανάκεια. Με τον ίδιο ακριβώς τρόπο μπορεί ένας κακόβουλος χρήστης να αποκρύψει το περιεχόμενο των μηνυμάτων του, διεισδύοντας μέσα από ελέγχους που θα αποκάλυπταν τις πραγματικές του προθέσεις.

Μία από τις μεθόδους που χρησιμοποιούνται ευρύτατα για την προστασία των προσωπικών δεδομένων είναι η κρυπτογραφία. Με τη χρήση κρυπτογραφικών τεχνικών, επιτυγχάνεται η εγκατάσταση σύνδεσης μεταξύ πελάτη και εξυπηρετητή με κατάλληλο τρόπο, έτσι ώστε να είναι πρακτικά αδύνατη η αθέμιτη παρακολούθηση της επικοινωνίας από οποιονδήποτε τρίτο. Μέσω αυτής της διαδικασίας, οι χρήστες μπορούν να στέλνουν και να λαμβάνουν πληροφορίες γνωρίζοντας ότι κανένας δεν μπορεί να την αποσπάσει ή να την χρησιμοποιήσει. Αυτή η εγγύηση επιτρέπει στους χρήστες να αισθάνονται ασφαλείς όταν

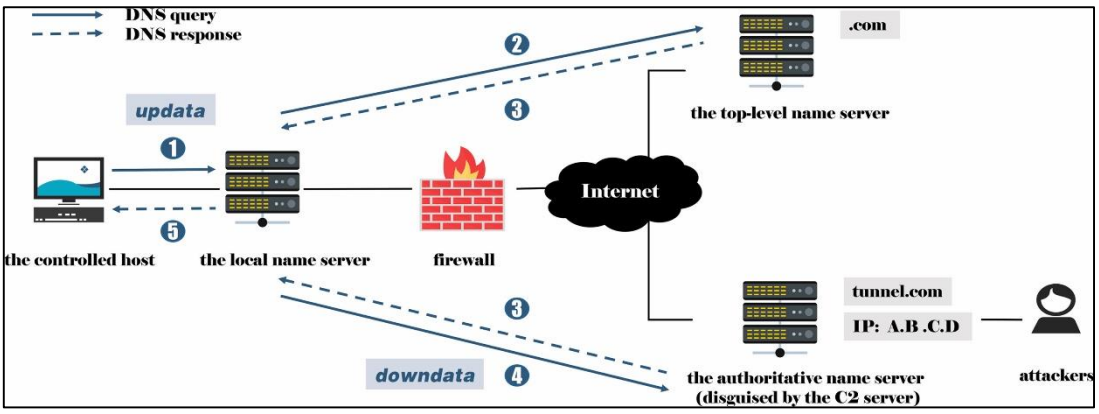
πραγματοποιούν συναλλαγές, ανταλλάσσουν προσωπικά δεδομένα ή συμμετέχουν σε άλλες δραστηριότητες στο διαδίκτυο.

Ωστόσο, αυτή η τεχνική δεν αποτελεί πανάκεια. Με τον ίδιο ακριβώς τρόπο που οι καλοπροαίρετοι χρήστες προστατεύουν τις επικοινωνίες τους, ένας κακόβουλος χρήστης μπορεί να αποκρύψει το περιεχόμενο των μηνυμάτων του, παρακάμπτοντας τους ελέγχους που θα αποκάλυπταν τις πραγματικές του προθέσεις. Οι τεχνικές κρυπτογράφησης, ενώ ισχυρές, δεν αποτελούν από μόνες τους μια πλήρη λύση στην καταπολέμηση της παράνομης δραστηριότητας. Η ανάγκη για ενίσχυση της ανίχνευσης και πρόληψης των επιθέσεων, η χρήση νέων τεχνολογιών και η ανάπτυξη μεθόδων ανίχνευσης κακόβουλης δραστηριότητας είναι πιο επιτακτική.

Επί παραδείγματι, τα τελευταία χρόνια έχει ξεκινήσει να καθιερώνεται το πρωτόκολλο DNS over HTTPS (DoH) [1]. Ενώ το παραδοσιακό DNS λειτουργεί χωρίς κρυπτογράφηση και οποιοσδήποτε ενδιάμεσος μπορεί να διαβάσει τα πεδία του πακέτου, το DoH αξιοποιεί την κρυπτογράφηση που παρέχει το HTTPS. Αυτό προσφέρει ιδιωτικότητα στους χρήστες, συμπεριλαμβανομένων και των κακόβουλων. Η κατηγοριοποίηση της επικοινωνίας σε αυτή την περίπτωση δυσχεραίνεται καθώς οι ενδιάμεσοι κόμβοι δεν μπορούν να διαβάσουν το περιεχόμενο των μηνυμάτων ώστε να αποφανθούν για το ποιόν του.

1.1 Ανάλυση του προβλήματος

Οι επιθέσεις υπό ανάλυση πρόκεινται για το DNS tunneling [2], το οποίο λειτουργεί ως εξής: ο κακόβουλος χρήστης ορίζει έναν εξυπηρετητή ως authoritative name server για έναν συγκεκριμένο τομέα. Οι ανυποψίαστοι χρήστες που μολύνθηκαν ακούσια από συγκεκριμένο λογισμικό γίνονται clients. Οι clients στέλνουν εν αγνοία τους διάφορα στοιχεία στον server, όπως passwords, ενθυλακώνοντας τα ως υποτιθέμενο subdomain στο όνομα της σελίδας του server. Έτσι αποστέλλεται ένα ερώτημα DNS (over HTTPS) στον τοπικό recursive name server, ο οποίος με τη σειρά του θα το στείλει σε έναν επόμενο, μέχρι να φτάσει στον authoritative server, ο οποίος μπορεί να απαντήσει με αντίστοιχο τρόπο, δημιουργώντας μία αμφίδρομη επικοινωνία. Το κακόβουλο DNS tunneling μπορεί να χρησιμοποιηθεί για απόσπαση δεδομένων από τον υπολογιστή του χρήστη ή για συντονισμό του υπολογιστή από ένα κεντρικό σημείο ελέγχου, με σκοπό την εκτέλεση λοιπών κακόβουλων ενεργειών. Αυτό είναι γνωστό ως Command & Control [3].



Σχήμα 1: Παράδειγμα DNS tunnelling[4]

Με απλούστερα λόγια, το tunneling πρόκειται για τεχνολογία ενθυλάκωσης ενός πακέτου, σε κάποιο άλλο πρωτόκολλο [4]. Σαν ιδέα, η τεχνολογία αυτή έχει πολλά κοινά με την τεχνικής της στεγανογραφίας [5], όπως αναφέρεται στο [6]. Ειδικότερα, όπως στην στεγανογραφία η πληροφορία κρύβεται στο φέρον μέσο, παραδείγματος χάριν μία εικόνα, έτσι και στο tunneling η πληροφορία, δηλαδή τα πακέτα που πρέπει να μεταδοθούν, κρύβονται στο φέρον, που στην προκειμένη είναι πακέτα του πρωτοκόλλου DNS.

Επιφορτισμένος με την αναγνώριση της κακόβουλης επικοινωνίας είναι ο μηχανισμός άμυνας του δικτύου, Intrusion Detection System (IDS) που βρίσκεται πριν το firewall. Στην περίπτωση του plain text DNS, η αναγνώριση μπορεί να γίνει με μελέτη των πεδίων του μηνύματος, δηλαδή με Deep Packet Inspection (DPI) [7]. Όμως αυτή η τεχνική δεν μπορεί να εφαρμοστεί όταν τα δεδομένα του πακέτου δεν είναι διαθέσιμα, άρα όταν πρόκειται για κρυπτογραφημένο DNS, αυτό καθίσταται ανεπαρκές [8]. Συνεπώς η μόνη αξιοποιήσιμη πληροφορία για το IDS είναι οι στατιστικές ιδιότητες των πακέτων, οι οποίες πρέπει να αναλυθούν σε πραγματικό χρόνο.

1.2 Παρεμφερές έργο

Για την αναγνώριση της κακόβουλης κίνησης αναγκαία κρίνεται η μελέτη των στατιστικών ιδιοτήτων (features) των πακέτων που ανταλλάσσονται, όπως ο ρυθμός δεδομένων και οι χρόνοι απόκρισης. Ήδη υπάρχουν έρευνες [9], [10] προς αυτή την κατεύθυνση, που αναλύουν συλλεγμένα πακέτα και μέσω αλγορίθμων μηχανικής καταφέρνουν να τα κατηγοριοποιήσουν, πετυχαίνοντας μάλιστα πολύ υψηλά ποσοστά ακρίβειας. Εντούτοις, έχουν ορισμένους περιορισμούς και μειονεκτήματα, τα οποία μπορούν να βελτιωθούν.

Συγκεκριμένα η έρευνα [9] αξιοποιεί το ίδιο dataset που θα χρησιμοποιηθεί και στο παρόν, το οποίο θα αναλυθεί παρακάτω. Όμως συμπεριλαμβάνει στην ανάλυση, εκτός από στατιστικές ιδιότητες των πακέτων, διευθύνσεις IP και ports. Η πρακτική αυτή προκαλεί μεγάλο bias στα μοντέλα απόφασης, καθώς το dataset, ενώ εμπεριέχει μεγάλο πλήθος διαφορετικών ροών, δεν παύει να είναι εργαστηριακό. Ορισμένες διευθύνσεις ανακυκλώνονται σε πολλά flows, άρα μπορούν πολύ εύκολα να κατηγοριοποιηθούν, χωρίς να δοθεί μεγάλη σημασία στα υπόλοιπα features. Σε πραγματικές συνθήκες οι διευθύνσεις IP σαφώς και επιδέχονται κατηγοριοποίησης, καθώς μία σεσημασμένη κακόβουλη IP είναι πολύ πιθανό να συνεχίσει να παράγει κακόβουλα flows. Στα πλαίσια της εργασίας όμως, δεδομένων των περιορισμών του dataset, αυτό δεν αποτελεί αντικείμενο μελέτης. Προκειμένου να αποφευχθεί πλήρως αυτό το bias, δεν αναλύονται διευθύνσεις IP ή ports από τα μοντέλα μάθησης.

Και οι δύο προαναφερθείσες έρευνες όμως έχουν τα βασικό κοινό χαρακτηριστικό ότι έγιναν offline, δηλαδή εκπαίδευσαν και δοκίμασαν τα μοντέλα μάθησης με έτοιμα δεδομένα. Τα διάφορα flows είχαν ήδη αναλυθεί και είχαν προκύψει τα τελικά στατιστικά χαρακτηριστικά τους, πάνω στα οποία έγιναν οι προβλέψεις. Η πρακτική αυτή είναι χρήσιμη για την θεωρητική μελέτη του προβλήματος. Μολαταύτα, προκειμένου να διαπιστωθεί η αποδοτικότητα ενός μοντέλου που σκοπεύει να προλαμβάνει, όχι απλώς να αναλύει, τέτοιες επιθέσεις, αρμόζει η διεξαγωγή ενός online πειράματος, όπου τα πακέτα θα εξετάζονται σε πραγματικό χρόνο και οι αλγόριθμοι μάθησης θα αποφασίζουν επιτόπου αν μία ροή πακέτων (flow) είναι κακόβουλη ή όχι. Με άλλα λόγια, ένα flow θα υφίσταται μελέτη καθ' όλη την διάρκεια του, ώστε να μπορέσει να γίνει πρόβλεψη το συντομότερο δυνατόν. Κατ' αυτόν τον τρόπο, μία κακόβουλη ροή θα διακοπεί έγκαιρα μόλις εντοπιστεί.

Επιπροσθέτως, σε ένα πραγματικό δίκτυο συνυπάρχουν πολλά flows, από τα οποία πρέπει να εξάγεται ένας σημαντικός αριθμός features για να γίνει η πρέπουσα ταξινόμηση. Αυτό σημαίνει ότι η μνήμη του υπολογιστή όπου γίνεται η επεξεργασία μπορεί εύκολα να υπερχειλίσει, επιβάλλοντας ακριβές αναβαθμίσεις υλικού για την εύρυθμη λειτουργία του. Η λύση είναι να περιοριστεί ο αριθμός των δεδομένων που χρήζουν διατήρησης. Μία αποδοτική μέθοδος είναι η χρήση πιθανοτικών δομών δεδομένων, οι οποίες εισάγουν ένα μικρό σφάλμα στον υπολογισμό των στατιστικών αριθμών που χρειάζονται, μειώνοντας όμως σημαντικά την απαιτούμενη μνήμη.

Συνεπώς, η παρούσα εργασία εκπονείται για την αναγνώριση κακόβουλων flows κρυπτογραφημένου DNS tunneling σε πραγματικό χρόνο, χρησιμοποιώντας πιθανοτικές δομές δεδομένων για μείωση της απαιτούμενης μνήμης.

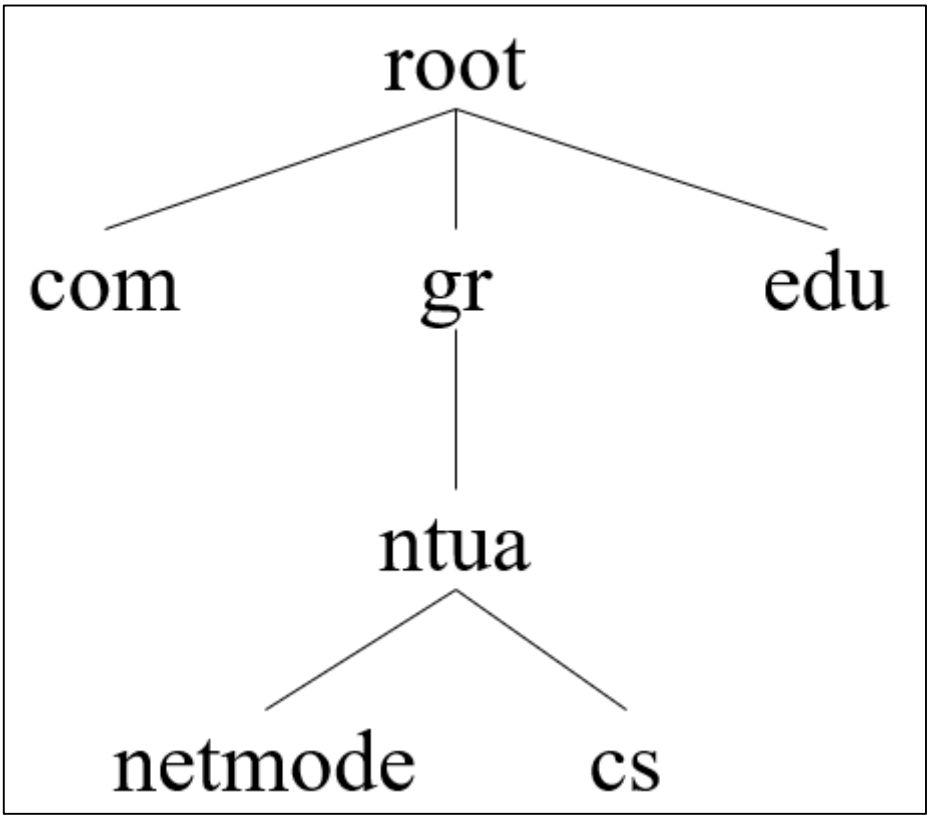
Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο παρόν κεφάλαιο παρουσιάζεται το πλαίσιο στο οποίο εντάσσεται η έρευνα, και αναλύονται τα εργαλεία και οι μηχανισμοί που θα χρησιμοποιηθούν. Ειδικότερα, θα εξηγηθεί το πρωτόκολλο DNS, σε απλή και κρυπτογραφημένη έκδοση και θα γίνει αναφορά σε μεθόδους παρακολούθησης δικτύων (network monitoring). Ακόμα, θα αναλυθεί η λειτουργία των πιθανοτικών δομών, οι οποίες αποτελούν τον ακρογωνιαίο λίθο της ιδέας της εργασίας. Τέλος, παρατίθενται οι στατιστικές ιδιότητες (features) που θα χρησιμοποιηθούν για την ανάλυση των flows ομαδοποιημένες σε τέσσερεις κατηγορίες, και γίνεται επεξήγηση τους.

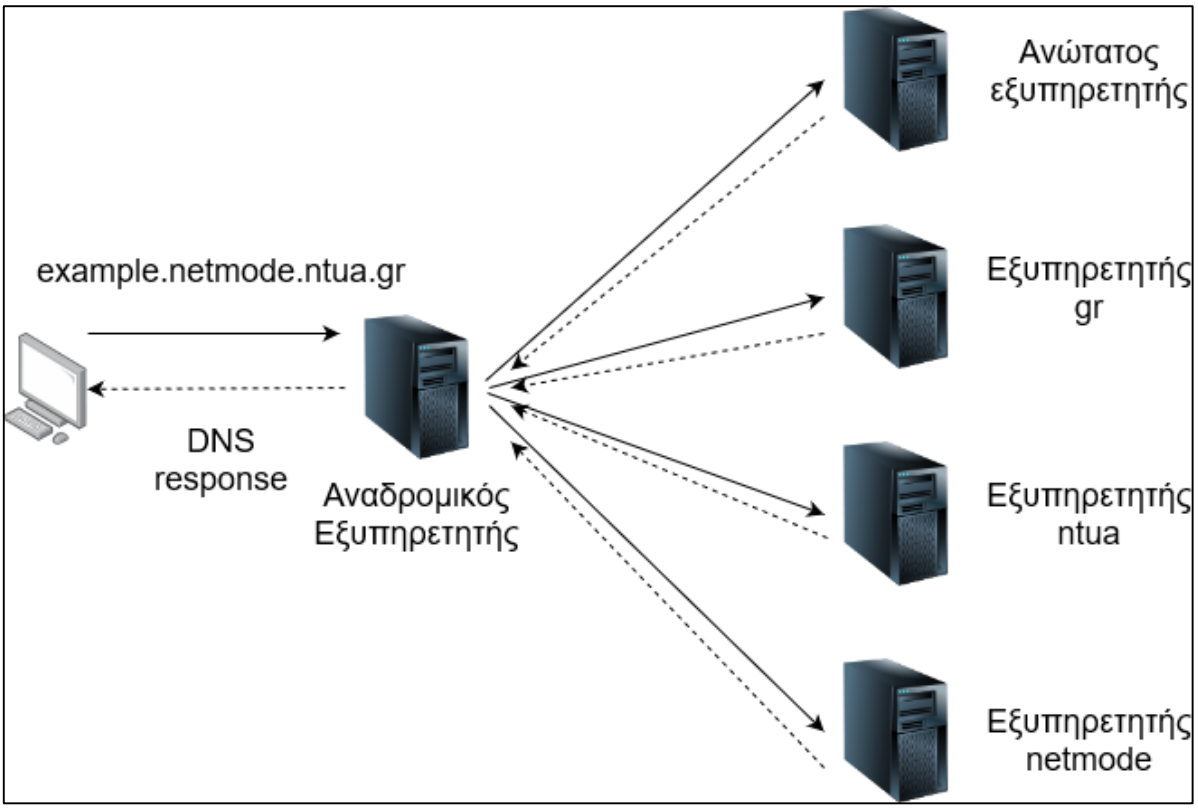
2.1 DNS

Το πρωτόκολλο Domain Name System (DNS) συγκαταλέγεται ανάμεσα στα βασικότερα συστατικά του διαδικτύου [11], [12], καθώς επιτρέπει την αντιστοίχιση φιλικών προς τον άνθρωπο ονομάτων σε αριθμητικές διευθύνσεις IP. Πρόκειται για μία διεσπαρμένη βάση δεδομένων, όπως αναφέρεται στο [13], με δενδρική δομή. Το δέντρο χωρίζεται σε ζώνες (zones), ξεκινώντας από την ρίζα. Για κάθε κόμβο του δέντρου υπάρχει τουλάχιστον ένας εξυπηρετητής (authoritative name server) που έχει πληροφορίες για τον αντίστοιχο τομέα, και αντιστοιχίσεις ονομάτων για όλους τους ιεραρχικά κατώτερους τομείς.



Σχήμα 2: Ιεραρχική δομή DNS

Με την βοήθεια του σχήματος μπορεί να γίνει αναπαράσταση της αναδρομικής (recursive) λειτουργίας του πρωτοκόλλου. Έστω ότι ο χρήστης επιθυμεί να προσπελάσει μία διεύθυνση που ανήκει στον χώρο netmode της εικόνας. Αρχικά δεν υπάρχει καμία διαθέσιμη πληροφορία, άρα θα ζητηθεί από κάποιον ανώτατο εξυπηρετητή (του οποίου η διεύθυνση IP είναι σταθερή και γνωστή) η διεύθυνση του υπεύθυνου για τη ζώνη gr. Μετά, θα ζητηθεί από τον εξυπηρετητή της ζώνης η διεύθυνση του υπεύθυνου για τη ζώνη ntua, και από αυτόν θα ζητηθεί ύστερα η πληροφορία για τον υπεύθυνο του netmode, ο οποίος θα δώσει και την πληροφορία που ζήτησε ο χρήστης. Με αυτό τον τρόπο οποιαδήποτε ερώτηση για κάποια διεύθυνση μπορεί να απαντηθεί αναδρομικά, εφόσον υπάρχει κάποιος υπεύθυνος για αυτή.



Σχήμα 3: Αναδρομική λειτουργία DNS

Παράλληλα με την αναδρομή, υπάρχει και η μέθοδος του caching, η οποία αποτελεί βασικό μηχανισμό για τη βελτίωση της απόδοσης του DNS. Για την αποφυγή του υπερβολικού φορτίου στους ανώτερους εξυπηρετητές στην ιεραρχία, οι ενδιάμεσοι κόμβοι, γνωστοί ως recursive resolvers, δύνανται να αποθηκεύουν προσωρινά την αντιστοίχιση ονόματος με διεύθυνση IP για την οποία δεν είναι υπεύθυνοι. Αυτή η διαδικασία επιτρέπει τη μείωση του αριθμού των αιτήσεων που καταλήγουν στους ανώτερους εξυπηρετητές, εξοικονομώντας πόρους και βελτιώνοντας την αποδοτικότητα ολόκληρου του συστήματος.

2.2 DNS over HTTPS

Το DNS over HTTPS (DoH) [1] αποτελεί εξέλιξη του απλού DNS. Είναι μια τεχνολογία που κρυπτογραφεί τις αιτήσεις DNS μέσω του πρωτοκόλλου HTTPS. Εισήχθη για να βελτιώσει την ιδιωτικότητα και την ασφάλεια των χρηστών κατά την περιήγησή τους στο διαδίκτυο, εξασφαλίζοντας ότι οι αιτήσεις DNS δεν μπορούν να παρακολουθούνται ή να αλλοιωθούν από τρίτους. Σε αντίθεση με το παραδοσιακό DNS, όπου οι αιτήσεις στέλνονται απροστάτευτες, το DoH αξιοποιεί την προστασία που παρέχει το HTTPS, επιτυγχάνοντας πρακτικά τέλεια κρυπτογράφηση.

Το DoH λειτουργεί ενθυλακώνοντας τις αιτήσεις DNS σε αιτήματα HTTPS. Η λειτουργία του πρωτοκόλλου είναι πανομοιότυπη με το απλό DNS. Η διαφορά έγκειται στο γεγονός ότι το πακέτο κρυπτογραφείται και μεταφέρεται μέσω μιας ασφαλούς σύνδεσης HTTPS, χρησιμοποιώντας την πόρτα 443, αντί για UDP στην πόρτα 53. Το αποτέλεσμα είναι ότι το περιεχόμενο της αίτησης DNS αποκρύπτεται από τρίτους που παρακολουθούν τη σύνδεση, όπως ISP ή κακόβουλους εισβολείς, ενώ ταυτόχρονα αποκρύπτεται η φύση του πακέτου, καθώς δεν ξεχωρίζει από λοιπή HTTPS κίνηση.

Το κύριο πλεονέκτημα του DoH είναι η ενίσχυση της ιδιωτικότητας. Καθώς οι αιτήσεις και αποκρίσεις είναι κρυπτογραφημένες, είναι πιο δύσκολο (εώς αδύνατο) για κάποιον να κατασκοπεύσει ποιους ιστότοπους επισκέπτεται ένας χρήστης. Επίσης, το DoH μειώνει τον κίνδυνο επιθέσεων man-in-the-middle [14], όπου κακόβουλοι επιτιθέμενοι μπορούν να ανακατευθύνουν τις αιτήσεις DNS σε κακόβουλες IP διευθύνσεις. Επιπλέον, το DoH εξασφαλίζει μεγαλύτερη συμβατότητα με υπάρχοντες μηχανισμούς προστασίας που βασίζονται στο HTTPS.

Το DoH έχει υιοθετηθεί από μεγάλους browser, όπως ο Mozilla Firefox και ο Google Chrome, καθώς και από εταιρείες όπως η Cloudflare και η Google, που παρέχουν υπηρεσίες DNS με υποστήριξη DoH. Παράλληλα, οργανισμοί όπως το Internet Engineering Task Force (IETF) έχουν προωθήσει την τυποποίηση του DoH ως σύγχρονη λύση για την ιδιωτικότητα στο διαδίκτυο.

Παρόλο που το DoH προσφέρει ενισχυμένη ασφάλεια, υπάρχουν προκλήσεις και προβληματισμοί σχετικά με την χρήση του. Η κρυπτογράφηση αποτρέπει μεν την παρακολούθηση των αιτημάτων από ανεπιθύμητους τρίτους, προσφέρει δε μονοπώλιο της πληροφορίας αυτής στους εξυπηρετητές DoH. Έτσι δίνεται ένα ενδεχομένως αθέμιτο ανταγωνιστικό πλεονέκτημα στους εξυπηρετητές, καθώς έχουν αποκλειστικότητα στο να παρακολουθούν τις προτιμήσεις των χρηστών.

2.2.1 Network Monitoring

Η εργασία εντάσσεται στο ευρύτερο πλαίσιο του Network Monitoring, το οποίο αποτελεί έναν από τους θεμελιώδεις τομείς στη διαχείριση δικτύων, εστιάζοντας στη συνεχή παρακολούθηση της κίνησης και της λειτουργίας τους με στόχο τη διασφάλιση της απόδοσης, της αξιοπιστίας και της ασφάλειας. Πρόκειται για τη διαδικασία συλλογής, ανάλυσης και ερμηνείας δεδομένων που προέρχονται από την κίνηση των πακέτων, τα πρωτόκολλα και τις εφαρμογές που χρησιμοποιούνται, ώστε να εντοπίζονται προβλήματα ή ανωμαλίες.

Η αποτελεσματική επιτήρηση δικτύων είναι ζωτικής σημασίας σε σύγχρονα περιβάλλοντα, όπως:

- **Επιχειρηματικά Δίκτυα:** Διατήρηση της συνεχούς λειτουργίας κρίσιμων υπηρεσιών.
- **Δίκτυα Παρόχων Υπηρεσιών Διαδικτύου (ISPs):** Διαχείριση μεγάλων όγκων δεδομένων και την παροχή ποιοτικών υπηρεσιών.
- **Κυβερνοασφάλεια:** Εντοπισμός απειλών, όπως επιθέσεις DDoS, κακόβουλο DNS tunneling ή παραβιάσεις δεδομένων.

Οι κύριες λειτουργίες του network monitoring περιλαμβάνουν:

1. **Εντοπισμό Ανωμαλιών:** Η παρακολούθηση σε πραγματικό χρόνο επιτρέπει τον εντοπισμό απότομων αυξήσεων στην κίνηση, μη εξουσιοδοτημένων ενεργειών ή άλλων ύποπτων δραστηριοτήτων.
2. **Ανάλυση Απόδοσης:** Εξασφαλίζεται η βέλτιστη λειτουργία του δικτύου μέσω της μέτρησης βασικών δεικτών απόδοσης (KPIs), όπως ο ρυθμός μετάδοσης δεδομένων, οι καθυστερήσεις (latency) και οι απώλειες πακέτων.
3. **Διαχείριση Πόρων:** Εντοπίζονται σημεία συμφόρησης και βελτιστοποιείται η κατανομή των δικτυακών πόρων.

Το σύγχρονο network monitoring αντιμετωπίζει αυξημένες προκλήσεις, όπως:

- **Κρυπτογράφηση Κίνησης:** Πρωτόκολλα όπως το HTTPS και το DNS over HTTPS (DoH) καθιστούν δυσκολότερη την ανάλυση του περιεχομένου των πακέτων, απαιτώντας νέες τεχνικές ανάλυσης.
- **Μεγάλοι Όγκοι Δεδομένων:** Η αύξηση της διαδικτυακής κίνησης απαιτεί λύσεις που να είναι αποδοτικές σε επίπεδο μνήμης και επεξεργασίας.
- **Αυτοματισμός και Πραγματικός Χρόνος:** Οι οργανισμοί χρειάζονται εργαλεία που μπορούν να αναλύουν δεδομένα και να αντιδρούν άμεσα, σε πραγματικό χρόνο.

Η παρούσα εργασία αποσκοπεί στην αντιμετώπιση αυτών των προβλημάτων. Με την χρήση μοντέλων ML για την ανάλυση στατιστικών ιδιοτήτων, γίνεται προσπάθεια παράκαμψης του κρυπτογραφημένου περιεχομένου. Οι πιθανοτικές δομές δεδομένων συνεισφέρουν στην μείωση της απαιτούμενης μνήμης και προσφέρουν σταθερούς ρυθμούς επεξεργασίας. Επίσης, η υλοποίηση λειτουργεί με σύγχρονη και εξετάζει πακέτα σύγχρονα, λαμβάνοντας αποφάσεις σε πραγματικό χρόνο.

2.3 Πιθανοτικές Δομές

Οι πιθανοτικές δομές δεδομένων (probabilistic data structures) αποτελούν ένα καινοτόμο εργαλείο για την καταμέτρηση συμβάντων και τον υπολογισμό στατιστικών ιδιοτήτων από μεγάλες ροές δεδομένων, προσφέροντας αποδοτικότητα τόσο στον χώρο όσο και στον χρόνο [15]. Έχουν σχεδιαστεί για περιπτώσεις όπου η πλήρης αποθήκευση ή ακριβής επεξεργασία όλων των δεδομένων είναι ανέφικτη λόγω περιορισμένων πόρων, ή για περιπτώσεις όπου ένα μικρό σφάλμα στην εκτίμηση είναι ανεκτό. Οι δομές αυτές λειτουργούν με βάση την ιδέα ότι μπορούν να παρέχουν εκτιμήσεις στατιστικών ιδιοτήτων με ένα μικρό, ελεγχόμενο σφάλμα, αντί για ακριβή αποτελέσματα.

Ένα από τα βασικά χαρακτηριστικά τους είναι η χρήση σταθερού μεγέθους μνήμης, ανεξαρτήτως του αριθμού των δεδομένων που επεξεργάζονται. Αυτό τις καθιστά ιδιαίτερα χρήσιμες για εφαρμογές σε ροές δεδομένων (data streams), όπου οι απαιτήσεις σε μνήμη αυξάνονται συνεχώς με τις παραδοσιακές μεθόδους.

Οι εκτιμήσεις που παρέχονται από αυτές τις δομές βρίσκονται, με υψηλή πιθανότητα, εντός ενός αποδεκτού εύρους σφάλματος. Αυτό σημαίνει ότι, αντί να παρέχουν ακριβείς τιμές, επιστρέφουν αποτελέσματα που είναι σχεδόν εγγυημένα εντός προκαθορισμένων ορίων. Η πιθανότητα αυτή μπορεί να οριστεί και να παραμετροποιηθεί κατά τη φάση του σχεδιασμού ή της υλοποίησης, ανάλογα με τις απαιτήσεις της εφαρμογής.

Για παράδειγμα, στα Count-Min Sketches, οι εκτιμήσεις για τη συχνότητα εμφάνισης ενός στοιχείου εξαρτώνται από το πλήθος των hash functions και το πλάτος του πίνακα, με το σφάλμα να μειώνεται όσο αυξάνεται η μνήμη που χρησιμοποιείται.

Η ικανότητα ρύθμισης αυτού του σφάλματος προσφέρει μεγάλη ευελιξία. Σε εφαρμογές που δεν υπάρχουν μεγάλες απαιτήσεις ακρίβειας, μπορεί να επιτραπεί ένα σχετικά μεγάλο σφάλμα, με κέρδος πολύ μεγάλη εξοικονόμιση μνήμης. Αντίστροφα, όταν η ακρίβεια είναι μείζονος σημασίας, μπορεί να θυσιαστεί κάποια μνήμη, ώστε τα αποτελέσματα να είναι πιο κοντά στα πραγματικά.

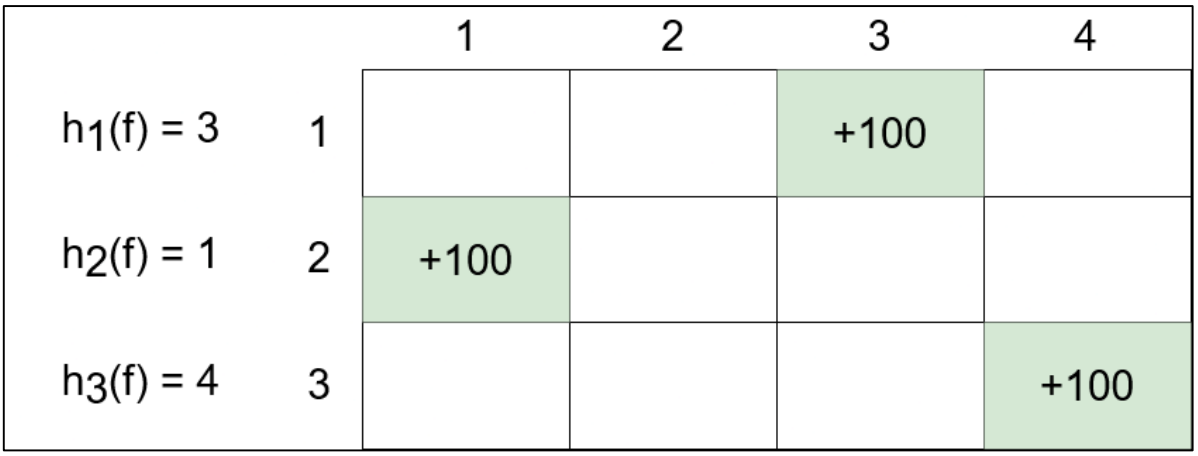
Επιπλέον, το εύρος σφάλματος είναι προκαθορισμένο και δεν εξαρτάται από τον αριθμό των δεδομένων που επεξεργάζεται η δομή, κάτι που αποτελεί σημαντικό πλεονέκτημα σε σενάρια που περιλαμβάνουν μεγάλες ροές δεδομένων (data streams). Αυτό καθιστά τις πιθανοτικές δομές ιδιαίτερα χρήσιμες σε περιβάλλοντα περιορισμένων πόρων, όπως τα κατανεμημένα συστήματα και τα δίκτυα, όπου η ανάγκη για ταχύτητα και χαμηλή κατανάλωση μνήμης είναι ζωτικής σημασίας.

Η δυνατότητα συντονισμού του σφάλματος ανάλογα με τις απαιτήσεις της εφαρμογής παρέχει ευελιξία στις πιθανοτικές δομές, καθιστώντας τις ένα ιδανικό εργαλείο για πλήθος πρακτικών εφαρμογών, από την παρακολούθηση δικτύων έως τη διαχείριση μεγάλων δεδομένων.

Η ανάλυση των χαρακτηριστικών και των εφαρμογών κάθε δομής θα παρουσιαστεί αναλυτικά στις επόμενες ενότητες.

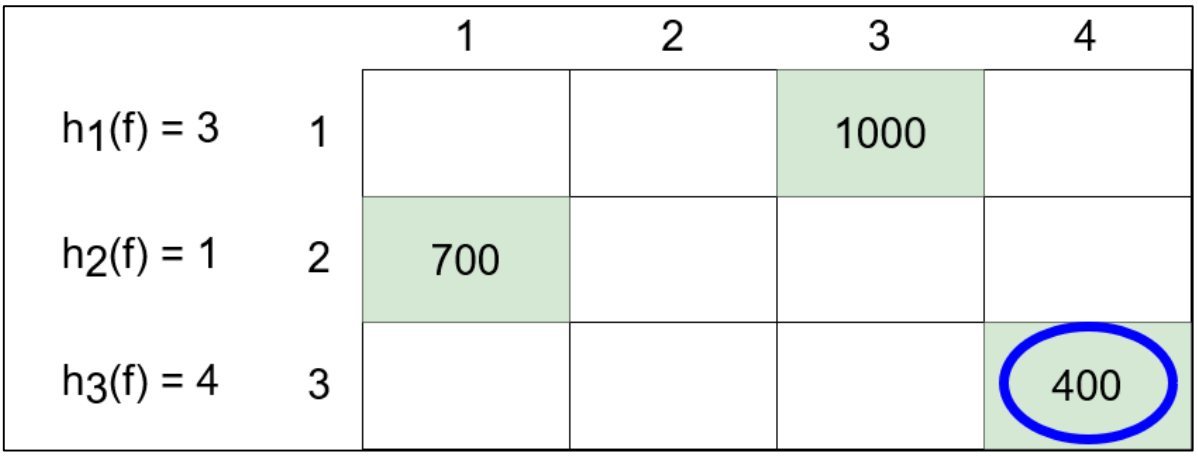
2.3.1 Count Min Sketch

Count Min Sketch (CMS): Το CMS [16] είναι μία δομή που μετράει την συχνότητα εμφάνισης διαφόρων συμβάντων. Για τις ανάγκες τις εργασίας δημιουργούνται δύο CMS, ένα για την καταμέτρηση bytes που εστάλησαν και ένα για bytes που ελήφθησαν. Η αρχή λειτουργίας είναι απλή και πρακτική. Το CMS είναι ένας δισδιάστατος πίνακας, τα στοιχεία του οποίου είναι ακέραιοι αριθμοί. Σε κάθε συμβάν αντιστοιχίζεται ένα μοναδικό αναγνωριστικό (στην προκειμένη, το flow) στο οποίο εφαρμόζονται τόσες συναρτήσεις κατακερματισμού (hash functions) όσες και οι σειρές του πίνακα. Το αποτέλεσμα κάθε hash είναι ένα κελί στην αντίστοιχη σειρά. Αυτά τα κελιά προσθέτουν στον αριθμό τους την μέτρηση του συμβάντος. Έτσι, κάθε φορά που γίνεται ένα συμβάν, ενημερώνονται ορισμένες θέσεις στον πίνακα.



Σχήμα 4: Εισαγωγή στοιχείου στη δομή CMS

Για την εξαγωγή της συχνότητας του συμβάντος, ο αλγόριθμος επιστρέφει την μικρότερη τιμή που βρέθηκε σε αυτές τις θέσεις (εξ' ου και το όνομα). Με αυτό τον τρόπο ελαχιστοποιείται η επίδραση των συγκρούσεων, δηλαδή κάποιο συμβάν να αύξησε θέση που αντιστοιχεί και σε κάποιο άλλο, και υπάρχει εγγύηση πως ο αλγόριθμος δεν θα επιστρέψει ποτέ εκτίμηση μικρότερη από την πραγματική, αλλά ίσως επιστρέψει μεγαλύτερη.



Σχήμα 5: Εξαγωγή μέτρησης από τη δομή CMS

2.3.2 T-Digest

T-Digest: Η δομή t-digest [17] χρησιμοποιείται για τον προσεγγιστικό υπολογισμό εκατοστημορίων (percentiles) σε μία ροή δεδομένων, δηλαδή πόσα συμβάντα της ροής είναι μικρότερα ή μεγαλύτερα από μία ορισμένη τιμή. Για τις ανάγκες της εργασίας, η δομή αξιοποιείται για την εύρεση των τριών διαμέσων που απαιτούνται για κάθε flow.

2.3.3 Top-K

Top-K: Ανάμεσα στα ζητούμενα features υπάρχουν ορισμένες επικρατούσες τιμές. Όπως προαναφέρθηκε, μόνο το επικρατούν μήκος πακέτου έχει νόημα στην ανάλυση. Για τον πιθανοτικό υπολογισμό του, χρησιμοποιείται η δομή Top-K [18]. Όπως υποδηλώνει το όνομα της, υπολογίζει προσεγγιστικά τα k συμβάντα που εμφανίζονται πιο συχνά σε μία ροή

δεδομένων, όπου το k είναι αυθαίρετα επιλεγμένος αριθμός. Εφόσον απαιτείται μόνο μία τιμή, στην εργασία είναι k = 1.

Η δομή κατά τη δημιουργία της δέχεται δύο παραμέτρους: eps, και εμπιστοσύνη. Το eps καθορίζει το πλάτος της δομής, δηλαδή πόσες στήλες θα έχει η εσωτερική αναπαράσταση. Η εμπιστοσύνη είναι ο βαθμός εγγύησης σωστού αποτελέσματος, και κυμαίνεται από 0.5 μέχρι 0.99. Όσο μεγαλύτερη, τόσο περισσότερο χώρο καταλαμβάνει η δομή.

2.4 Στατιστικές ιδιότητες

Τα features που χρησιμοποιούνται για την ταξινόμηση της κίνησης παρατίθενται στον παρακάτω πίνακα. Η επιλογή τους δεν είναι αυθαίρετη, αλλά έγινε με γνώμονα το dataset [2] όπως θα αναλυθεί αργότερα.

1	Bytes	Αριθμός flow bytes που εστάλησαν
2		Ρυθμός flow bytes που εστάλησαν
3		Αριθμός flow bytes που ελήφθησαν
4		Ρυθμός flow bytes που ελήφθησαν
5	Μήκος πακέτου	Μέσο μήκος πακέτου
6		Διάμεσος μήκος πακέτου
7		Επικρατούν μήκος πακέτου
8		Διακύμανση μήκους πακέτου
9		Τυπική απόκλιση μήκους πακέτου
10		Συντελεστής διακύμανσης μήκους πακέτου
11		Ασυμμετρία από διάμεσο μήκους πακέτου
12		Ασυμμετρία από επικρατούν μήκος πακέτου
13	Χρόνος	Μέσος χρόνος πακέτου
14		Διάμεσος χρόνου πακέτου
15		Επικρατών χρόνος πακέτου
16		Διακύμανση χρόνου πακέτου
17		Τυπική απόκλιση χρόνου πακέτου
18		Συντελεστής διακύμανσης χρόνου πακέτου
19		Ασυμμετρία από διάμεσο χρόνου πακέτου
20		Ασυμμετρία από επικρατούντα χρόνο πακέτου
21	Χρονική διαφορά αιτήματος με απόκριση	Μέση χρονική διαφορά αιτήματος/απόκρισης
22		Διάμεσος χρονικής διαφοράς αιτήματος/απόκρισης
23		Επικρατούσα χρονική διαφορά αιτήματος/απόκρισης
24		Διακύμανση χρονικής διαφοράς αιτήματος/απόκρισης
25		Τυπική απόκλιση χρονικής διαφοράς αιτήματος/απόκρισης
26		Συντελεστής διακύμανσης χρονικής διαφοράς αιτήματος/απόκρισης
27		Ασυμμετρία από διάμεσο χρονικής διαφοράς αιτήματος/απόκρισης
28		Ασυμμετρία από επικρατούσα χρονική διαφορά αιτήματος/απόκρισης

Πίνακας 1: Στατιστικές ιδιότητες (features) [2]

Για την αποφυγή σύγχυσης δίνονται εδώ οι ορισμοί των στατιστικών εννοιών που χρησιμοποιούνται. Με την λέξη *μέσος* εννοείται η μέση τιμή του αντίστοιχου μεγέθους.

Διάμεσος, διακύμανση και *τυπική απόκλιση* είναι έννοιες σαφώς ορισμένες στην βιβλιογραφία και δεν χρήζουν περαιτέρω ανάλυσης.

Επικρατούσα τιμή είναι η τιμή που εμφανίζεται συχνότερα στην εξεταζόμενη ροή δεδομένων (mode).
Ο *συντελεστής διακύμανσης* ορίζεται ως το πηλίκο τυπικής απόκλισης προς μέση τιμή.

$$Συντελεστής\ διακύμανσης = \frac{Τυική\ απόκλιση}{Μέση\ τιμή}$$

Ασσυμετρία από διάμεσο είναι το τριπλάσιο της διαφοράς της διαμέσου από την μέση τιμή, διά την τυπική απόκλιση.

$$Ασσυμετρία\ από\ διάμεσο = \frac{3 * (Μέση\ τιμή - Διάμεσος)}{Τυπική\ απόκλιση}$$

Τέλος, *ασσυμετρία από επικρατούσα* είναι η διαφορά της επικρατούσας τιμής από την μέση, διά την τυπική απόκλιση.

$$Ασσυμετρία\ από\ επικρατούσα = \frac{Μέση\ τιμή - Επικρατούσα}{Τυπική\ απόκλιση}$$

Παρακάτω ακολουθεί η ανάλυση των κατηγοριών features που μελετήθηκαν στο [2]. Στα πλαίσια της εργασίας τα πακέτα ανταλλάσσονται σε πραγματικό χρόνο, οπότε οι τιμές αλλάζουν δυναμικά. Ακολουθεί η επεξήγηση των κατηγοριών.

- Η πρώτη κατηγορία αφορά στην καταμέτρηση του αριθμού και του ρυθμού των bytes της ροής που εστάλησαν, δηλαδή πακέτα με κατεύθυνση από τον client προς τον server, και αντίστροφα για τα bytes που ελήφθησαν.
- Η κατηγορία που αφορά στο μήκος πακέτου εξάγει στατιστικά σχετικά με τα bytes που φέρουν τα πακέτα της ροής.
- Η έννοια του χρόνου πακέτου στο παρόν ορίζεται ως η χρονική απόσταση του τρέχοντος πακέτου από το πρώτο της ροής. Συνεπώς η κάθε καταγραφή χρόνων είναι μεγαλύτερη από τις προηγούμενες.
- Η τελευταία κατηγορία εξετάζει τον χρόνο ανάμεσα σε ένα πακέτο (αίτημα) από τον client και στο αντίστοιχο πακέτο (απόκριση) από τον server. Ο υπολογισμός μίας μέτρησης γίνεται κάθε φορά που εντοπίζεται πακέτο με πηγή τον server, δηλαδή ένα response. Καταγράφεται η χρονική διαφορά από το τελευταίο πακέτο με πηγή τον client. Η προσέγγιση αυτή είναι απλουστευτική, όμως η κρυπτογράφηση των πακέτων δεν επιτρέπει ακριβή ταυτοποίηση αιτημάτων και αποκρίσεων στο στρώμα εφαρμογής (application layer). Άλλωστε η ανάγκη σύγχρονης ανάλυσης των ροών επιβάλλει τους απλούς και γρήγορους υπολογισμούς.

Αυτά είναι όλα τα features που αναλύονται στο [2], και εξηγήθηκε ο μαθηματικός ορισμός τους. Ένα υποσύνολο αυτών αξιοποιείται στην εργασία, καθώς ορισμένα features θεωρήθηκαν περιττά ή αναξιόπιστα. Πιο συγκεκριμένα, στην εργασία δεν υπολογίζονται καθόλου η επικρατούσα τιμή χρόνου και διαφοράς αιτήματος με απόκριση, καθώς επίσης και οι αντίστοιχες ασυμμετρίες από την επικρατούσα. Αυτή η απόφαση έγινε καθώς η επικρατούσα τιμή σε δείγμα με δεκαδικούς αριθμούς δεν ακολουθεί τον κλασικό ορισμό. Αντιθέτως, ο υπολογισμός σε αυτή την περίπτωση εισάγει μεγάλη πολυπλοκότητα, η οποία δεν προσφέρει ουσιαστικά στους σκοπούς της εργασία. Στο ίδιο μήκος κύματος αρμόζει η παρατήρηση ότι από τα features που συζητήθηκαν απουσιάζουν χαρακτηριστικά όπως διευθύνσεις IP ή ports, καθώς στην παρούσα εργασία δεν διεξάγεται ανάλυση με βάση αυτά, όπως συζητήθηκε παραπάνω.

Όμως ακόμα και αγνοώντας τις διευθύνσεις, υπάρχει μεγάλο πλήθος features που πρέπει να αναλυθούν. Σε συνδυασμό με το γεγονός ότι σε ένα πραγματικό σύστημα ενδέχεται να υπάρχει πληθώρα από flows, καθένα από τα οποία μπορεί να περιλαμβάνει πολλά πακέτα, αρμόζει η χρήση δομών που θα υλοποιούν την καταμέτρηση οικονομικά από άποψη μνήμης.

Κεφάλαιο 3

Αρχή Λειτουργίας

Για την ανάλυση των πακέτων δημιουργήθηκε ένα πρόγραμμα σε Python, το οποίο συνδυάζει διάφορες τεχνικές για τη λήψη, επεξεργασία και ανάλυση των δεδομένων. Το πρόγραμμα χωρίζεται σε τρία βασικά στάδια, τα οποία εκτελούνται διαδοχικά και επιτρέπουν την πλήρη ανάλυση των δικτυακών πακέτων. Το πρώτο στάδιο επικεντρώνεται στην εκπαίδευση του μοντέλου μάθησης. Στη συνέχεια, το δεύτερο στάδιο αφορά την εξαγωγή από κάθε πακέτο των απαραίτητων δεδομένων που χρειάζονται για την επεξεργασία. Τέλος, το τρίτο στάδιο αφορά την εξαγωγή των features για κάθε flow, με βάση τα δεδομένα κάθε πακέτου, και ταξινόμηση των flows σε καλόβουλα ή κακόβουλα.

Επιπλέον, υπάρχουν παραλλαγές του προγράμματος που εκτελούν παρόμοιες λειτουργίες, αλλά με διαφορετικούς τρόπους ή με τροποποιημένες παραμέτρους. Αυτές οι παραλλαγές επιτρέπουν την εξερεύνηση διαφορετικών προσεγγίσεων για την ανάλυση των δεδομένων, παρέχοντας ευελιξία στην εφαρμογή της ανάλυσης σε διαφορετικά σενάρια ή δεδομένα. Κάθε παραλλαγή μπορεί να χρησιμοποιεί διαφορετικές μεθόδους εξαγωγής χαρακτηριστικών, άλλους αλγόριθμους μάθησης ή διαφοροποιημένες τεχνικές ανάλυσης, ώστε να προσαρμόζεται στις εκάστοτε ανάγκες και απαιτήσεις κάθε πειράματος. Η βασική διάκριση είναι η ντετερμινιστική και η πιθανοτική υλοποίηση, δηλαδή η εκτέλεση του προγράμματος χρησιμοποιώντας μόνο παραδοσιακές δομές, ή πιθανοτικές.

3.1 Dataset

Το Πανεπιστήμιο του New Brunswick παρήγε το 2020 ερευνητικά δεδομένα που αφορούν το υπό εξέταση πρόβλημα [2]. Χρησιμοποιώντας διάφορα λογισμικά δημιούργησαν στο εργαστήριο μεγάλο όγκο δικτυακής κίνησης. Ένα σημαντικό μέρος αυτής είναι κακόβουλο, συγκεκριμένα DNS tunneling. Ακόμα, δημιούργησαν καλόβουλη κίνηση, χρησιμοποιώντας την δυνατότητα για DNS over HTTPS που προσφέρουν αρκετοί δημοφιλείς browsers, επιλέγοντας διαφορετικούς κάθε φορά εξυπηρετητές DNS. Παράλληλα, κατέγραψαν απλή HTTPS κίνηση, που δεν είναι DNS. Εξαιτίας του μεγάλου φάσματος από διαφορετικά εργαλεία που χρησιμοποιήθηκαν, οι καταγραφές αυτές είναι μία αξιόλογη αναπαράσταση πραγματικών συνθηκών, παρά το γεγονός ότι κατασκευάστηκαν στα πλαίσια εργαστηριακής έρευνας.

Εκτός από τις καταγραφές που υπάρχουν σε μορφή αρχείων pcap, το dataset περιλαμβάνει και αρχεία csv με τα features του κάθε flow, όπως προέκυψαν από ανάλυση του Πανεπιστημίου, και την κατηγορία του (δηλαδή κακόβουλο, καλόβουλο, μη DoH). Τα features, ή αλλιώς στατιστικές ιδιότητες, είναι χαρακτηριστικά των flows που σχετίζονται κυρίως με χρονικές διαφορές ανάμεσα στα πακέτα του εκάστοτε flow και το μήκος σε bytes των πακέτων.

Όλα τα πειράματα και η ανάλυση που ακολουθεί βασίστηκε στα δεδομένα που υπάρχουν στο dataset.

3.2 Εξαγωγή δεδομένων

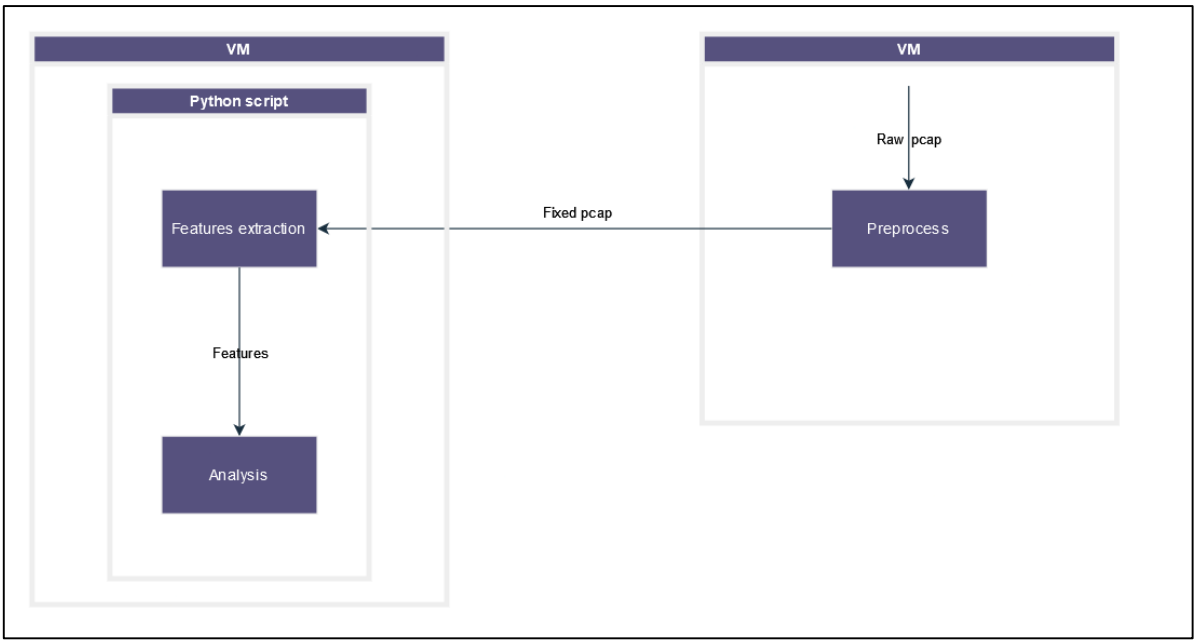
Η εκτέλεση των πειραμάτων χρησιμοποιεί network monitoring για την συλλογή των απαραίτητων δεδομένων από τα πακέτα, ανά flow. Παρόμοια με δημοφιλή πρωτόκολλα όπως το NetFlow [19] και το sFlow [20], χρησιμοποιούνται μεταδεδομένα (metadata) σχετικά με τα πακέτα, και αγνοείται το περιεχόμενο τους. Άλλωστε όπως έχει συζητηθεί σε προηγούμενη ενότητα, δεν είναι δυνατή η ανάγνωση του περιεχομένου κρυπτογραφημένης κίνησης. Η ειδοποιός διαφορά των πρωτοκόλλων αυτών με την παρούσα εργασία είναι η χρήση πιθανοτικών δομών για την αποθήκευση και διαχείριση των μεταδεδομένων. Επίσης, σε αντίθεση με το sFlow δεν διεξάγεται δειγματοληψία στα πακέτα, ώστε οι προβλέψεις να πετυχαίνουν μέγιστη ακρίβεια.

Αναφορικά με την εξαγωγή των δεδομένων από τα πακέτα, υπάρχουν δύο μέθοδοι, η σύγχρονη και η προσομοίωση της σύγχρονης. Και οι δύο μέθοδοι εξετάζουν κάθε πακέτο σειριακά. Χρησιμοποιείται η λέξη δεδομένα αντί για features γιατί πρόκειται για δύο ξεχωριστές έννοιες. Τα δεδομένα που εξάγονται από τα πακέτα είναι τα εξής:

- Ταυτότητα flow όπου ανήκει το πακέτο, δηλαδή ένα string της μορφής clientIP_clientPort_serverIP_443. Η πόρτα του server είναι πάντα η 443 εφόσον πρόκειται για HTTPS κίνηση.
- Μήκος πακέτου.
- Ένδειξη για το αν το πακέτο αποτελεί request (δηλαδή από τον client προς τον server) ή response (από τον server προς τον client). Πρόκειται για μία απλή Boolean μεταβλητή.
- Timestamp, ο χρόνος που ελήφθη το πακέτο. Πρόκειται για τον χρόνο που αναγράφεται στο αρχείο pcap, όχι για τον χρόνο που έλαβε το πακέτο το πρόγραμμα.

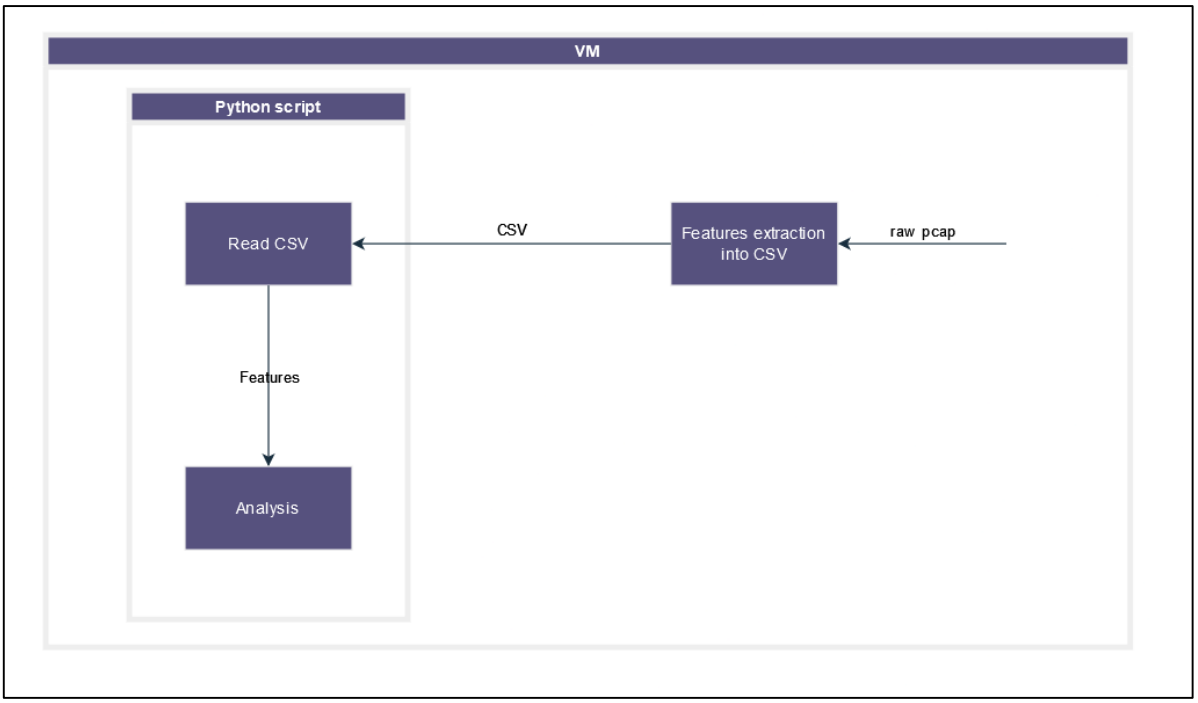
Τα features είναι οι στατιστικές ιδιότητες που προκύπτουν από αυτά τα δεδομένα, και αφορούν ένα flow, δηλαδή μία ροή πακέτων.

Η πρώτη μέθοδος είναι πλήρως σύγχρονη. Εκτελεί sniffing πακέτων που αναπαράγονται με τα κατάλληλα προγράμματα, και εξάγονται από αυτά τα απαιτούμενα features. Για αυτή την λειτουργία αξιοποιήθηκαν δύο Virtual Machines (VMs), ένα για την εκτέλεση του προγράμματος και ένα για την έγχυση πακέτων, μέσω του λογισμικού tcpreplay. Ο υπολογιστής με τον κώδικα θεωρείται στο πείραμα ως ένας κόμβος ανάμεσα στους DNS servers και στους clients. Ο άλλος υπολογιστής αναπαράγει τις καταγραφές pcap που προέρχονται στο σύνολο τους από το dataset. Οι καταγραφές υφίστανται την κατάλληλη επεξεργασία στο στρώμα Ethernet ώστε η MAC προορισμού να είναι του πρώτου υπολογιστή ώστε τα πακέτα να καταλήγουν εκεί που πρέπει. Αυτή η αλλαγή δεν επηρεάζει καθόλου τα features που απαιτούνται για την ανάλυση.



Σχήμα 6: Πρώτος τρόπος λειτουργίας (σύγχρονη αναπαραγωγή πακέτων)

Η δεύτερη λειτουργία πραγματοποιεί επίσης ανάλυση πακέτο προς πακέτο, όμως διαβάζει τα features από ένα αρχείο CSV. Από τα αρχεία pcap εξάγονται εκ των προτέρων τα απαιτούμενα χαρακτηριστικά (διευθύνσεις IP και ports για την ταυτοποίηση κάθε flow, μήκη πακέτων και timestamps για τον υπολογισμό των διαφόρων μεγεθών) και αποθηκεύονται σε μορφή CSV. Το τελικό αρχείο έχει μία σειρά για κάθε πακέτο της αρχικής καταγραφής. Σημειώνεται πως αυτά τα CSV είναι εντελώς διαφορετικά από τα CSV που περιλαμβάνονται στο dataset. Τα τελευταία προέκυψαν από συνολική ανάλυση κάθε flow και δεν μπορούν να χρησιμοποιηθούν για σύγχρονη ανάλυση. Τα CSV που περιλαμβάνονταν στο dataset δεν χρησιμοποιήθηκαν στην εργασία. Το πρόγραμμα Python διαβάζει το αρχείο που δημιουργήθηκε σειρά προς σειρά, προσομοιώνοντας πλήρως την ανάλυση σε πραγματικό χρόνο.



Σχήμα 7: Δεύτερος τρόπος λειτουργίας (ανάγνωση από CSV)

Η δεύτερη λειτουργία υπάρχει για εξοικονόμηση χρόνου, και για διασφάλιση ότι δεν θα χαθούν πακέτα στην ανάλυση. Οι υπολογιστικοί πόροι που διατέθηκαν για την εργασία δεν είναι απεριόριστοι. Αποτέλεσμα είναι πως κατά την αναπαραγωγή πακέτων υπάρχει πιθανότητα κάποια να χαθούν, όταν υπάρχει μεγάλη ριπή (burst) στην καταγραφή. Αντίστροφα, αν μειωνόταν από πριν ο ρυθμός με τον οποίο αναπαράγονται τα πακέτα ώστε να εξαλειφθούν πλήρως οι απώλειες, ο χρόνος για την εκτέλεση της ανάλυσης θα αυξανόταν σε απαγορευτικά επίπεδα, ιδίως εάν ληφθεί υπόψιν ο αριθμός των πειραμάτων που τελικά διεξήχθησαν.

Συμπεραίνοντας, ο δεύτερος τρόπος λειτουργίας είναι πολύ πιο αποδοτικός στην πειραματική διαδικασία χωρίς να επηρεάζει στο ελάχιστο τις τελικές τιμές των εξαγόμενων features, άρα είναι αυτός που προτιμάται για την επίτευξη των σκοπών της εργασίας, αναφορικά με την μνήμη και την ακρίβεια. Όμως, επειδή το ποσοστό απολεσθέντων πακέτων είναι σημαντικός δείκτης αποδοτικότητας ενός δικτυακού αλγορίθμου, διεξάγεται ανάλυση και σε αυτόν τον τομέα, σε ξεχωριστό κεφάλαιο, αξιοποιώντας τον πρώτο τρόπο λειτουργίας. Παράλληλα επισημαίνεται πως το κομμάτι της ανάλυσης των πακέτων είναι το ίδιο ασχέτως του τρόπου εξαγωγής των ιδιοτήτων.

3.3 Μοντέλο μάθησης

Βασική συνιστώσα του προγράμματος είναι ο δυαδικός αλγόριθμος μάθησης (binary classifier) που κατηγοριοποιεί κάθε flow σε καλόβουλο ή κακόβουλο. Στο [2] προτείνεται αρχικά η ταξινόμηση της κίνησης σε DoH και μη DoH, και στη συνέχεια η ανάλυση της DoH κίνησης. Ωστόσο, αυτή η προσέγγιση δεν επιλέχθηκε, καθώς η διαδικασία κατά την οποία ένα flow πρέπει πρώτα να χαρακτηριστεί ως DoH και ακολούθως να αξιολογηθεί για την κακόβουλη φύση του, προσθέτει σημαντική καθυστέρηση στην τελική απόφαση. Αυτό συμβαίνει όχι μόνο λόγω της ανάγκης για διαδοχική ταξινόμηση, αλλά και λόγω του αυξημένου υπολογιστικού φορτίου που επιφέρει. Συνεπώς, το μοντέλο εκπαιδεύεται με δύο

κατηγορίες: κακόβουλη κίνηση και λοιπή, με την τελευταία να περιλαμβάνει τόσο καλόβουλα DoH flows όσο και μη DoH κίνηση.

3.3.1 Training testing split

Το training dataset προέκυψε από την εξαγωγή features από ένα μικρό δείγμα flows από κάθε κατηγορία κίνησης. Η διαδικασία παραγωγής του είναι η εξής: επιλέχθηκαν τυχαία ορισμένα flows από κάθε κατηγορία. Ύστερα πέρασαν από το στάδιο εξαγωγής features, και τα αποτελέσματα συγκεντρώθηκαν σε ένα αρχείο CSV, μαζί με την κατηγορία κάθε flow (κακόβουλο ή μη). Έκτοτε χρησιμοποιείται αυτό το training dataset για τα μοντέλα.

Το testing dataset είναι διακεκριμένο από το training, και έχουν μεγάλη διαφορά μεγέθους. Ειδικότερα, το training περιλαμβάνει δεδομένα που εξήχθησαν από 210 flows, ενώ για testing χρησιμοποιούνται περίπου 60000. Δηλαδή 0.3% του χρησιμοποιημένου dataset είναι για εκπαίδευση και το υπόλοιπο 99.7% για δοκιμές. Παρά την σχετικά ασυνήθιστη διαφορά στα μεγέθη, οι προβλέψεις των αλγορίθμων κυμαίνονται σε ικανοποιητικές τιμές, με ποσοστό ακρίβειας περίπου 80% στη γενική περίπτωση. Αν αφιερωνόταν μεγαλύτερο ποσοστό για εκπαίδευση οι προβλέψεις των μοντέλων θα ήταν σχεδόν τέλειες, όπως φαίνεται από τα αποτελέσματα της έρευνας [9], εξαιτίας των αναπόφευκτων ομοιοτήτων που εμφανίζονται στο εργαστηριακό dataset. Συνεπώς η σύγκριση των διαφορετικών υλοποιήσεων θα ήταν δυσχερής, καθώς οι ελάχιστες λάθος προβλέψεις δεν θα μπορούσαν να προσφέρουν ισχυρές ενδείξεις για την αποδοτικότητα ή μη κάποιας δομής. Άλλωστε, ο σκοπός της εργασίας δεν είναι η μεγιστοποίηση της ακρίβειας, αλλά ο έλεγχος της αποδοτικότητας στην περίπτωση χρήσης πιθανοτικών δομών.

3.3.2 Αλγόριθμοι μάθησης

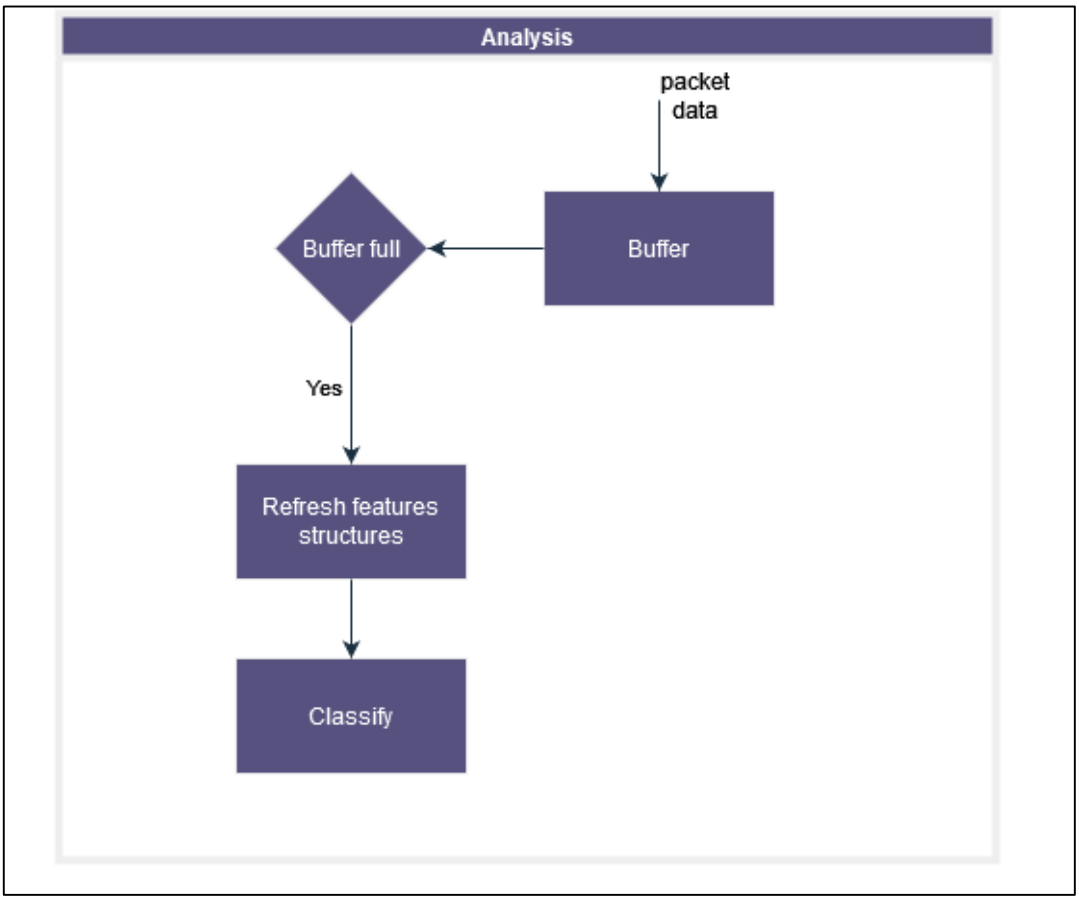
Όλοι οι αλγόριθμοι που χρησιμοποιήθηκαν είναι supervised machine learning. Αυτό σημαίνει ότι το μοντέλο εκπαιδεύεται χρησιμοποιώντας ένα σύνολο δεδομένων που περιέχει παραδείγματα με γνωστά χαρακτηριστικά και τους αντίστοιχους επιθυμητούς χαρακτηρισμούς (labels). Στην προκειμένη, το label είναι είτε κακόβουλο ή καλόβουλο.

Δοκιμάστηκε ένα εύρος από αλγορίθμους μάθησης. Δεδομένου ότι ο κώδικας είναι σε Python, αξιοποιήθηκε η βιβλιοθήκη sklearn [21], από την οποία χρησιμοποιήθηκαν ορισμένοι αλγόριθμοι. Αναλυτικά, έγιναν πειράματα χρησιμοποιώντας Support Vector Classification (SVC) [22], Gaussian Naïve Bayes [23], Random Forests [24] και Histogram-based Gradient Boosting Classification Tree [25]. Από αυτούς οι δύο τελευταίοι είχαν τα υψηλότερα ποσοστά ακρίβειας, και με βάση αυτούς έγινε το μεγαλύτερο μέρος της ανάλυσης. Εντούτοις, η παρούσα έρευνα δεν έχει σκοπό την εύρεση των βέλτιστων αλγορίθμων μάθησης, συνεπώς δεν γίνεται ενδελεχής ανάλυση της λειτουργίας τους.

3.4 Ανάλυση

Αφού γίνει η εκπαίδευση του μοντέλου, ξεκινάει η διαδικασία εξαγωγής δεδομένων από κάθε πακέτο, τα οποία περνάνε στο στάδιο της ανάλυσης. Υπάρχει ένα buffer που συλλέγει τα δεδομένα. Μόλις το buffer δεχθεί ορισμένο αριθμό πακέτων, αδειάζει ό,τι έχει σε μία συνάρτηση που καλείται να ανανεώσει τις τιμές των διαφόρων στατιστικών ιδιοτήτων των σχετικών flows με βάση τις τελευταίες πληροφορίες. Έπειτα, τα flows που δέχθηκαν αλλαγή περνάνε από το μοντέλο για πρόβλεψη. Η λογική που αναφέρθηκε ακολουθεί το μοντέλο

Map Reduce [26], καθώς τα πακέτα δέχονται ξεχωριστή επεξεργασία, αλλά στο τέλος περνάνε τα συνολικά flows από το μοντέλο.



Σχήμα 8: Ανάλυση δεδομένων

3.4.1 Υπολογισμός στατιστικών ιδιοτήτων και εισαγωγή στις δομές

Εφόσον η δομή buffer έχει γεμίσει, τα δεδομένα περνάνε στο στάδιο της ανανέωσης των στατιστικών ιδιοτήτων. Σε ορισμένες από αυτές συνεισφέρουν οι πιθανοτικές δομές, οπότε για να ελεγχθεί η αποδοτικότητα τους έγιναν δύο υλοποιήσεις του προγράμματος, μία πιθανοτική και μία πλήρως ντετερμινιστική. Στον παρακάτω πίνακα περιγράφεται συνοπτικά ο τρόπος διαχείρισης των ιδιοτήτων.

Στατιστική ιδιότητα	**Ντετερμινιστική υλοποίηση**	**Πιθανοτική υλοποίηση**
Αριθμός flow bytes που εστάλησαν	Μία μεταβλητή ανά flow. Προσαυξάνεται κάθε φορά με το μήκος του τρέχοντος πακέτου που εστάλη.	Μία πιθανοτική δομή για όλα τα flows.
Ρυθμός flow bytes που εστάλησαν	Αξιοποιείται ο αριθμός bytes που εστάλησαν και γίνεται διαίρεση με τον χρόνο.	Αξιοποιείται ο αριθμός bytes που εστάλησαν και γίνεται διαίρεση με τον χρόνο.
Αριθμός flow bytes που ελήφθησαν	Παρόμοια με αριθμό bytes που εστάλησαν.	Παρόμοια με αριθμό bytes που εστάλησαν.

Ρυθμός flow bytes που ελήφθησαν	Παρόμοια με ρυθμό bytes που εστάλησαν.	Παρόμοια με ρυθμό bytes που εστάλησαν.
Μέσο μήκος πακέτου	Υπολογισμός με την τεχνική του κινούμενου μέσου.	
Διάμεσος μήκος πακέτου	Ένας πίνακας ανά flow για τα μήκη. Σε αυτόν εφαρμόζεται η συνάρτηση διαμέσου.	Μία πιθανοτική δομή ανά flow (TDigest [17]) στην οποία εισάγονται τα μήκη.
Επικρατούν μήκος πακέτου	Αξιοποιείται ο προηγούμενος πίνακας, στον οποίο εφαρμόζεται η συνάρτηση mode.	Μία πιθανοτική δομή ανά flow (Top K [18]) στην οποία εισάγονται τα μήκη.
Διακύμανση μήκους πακέτου	Υπολογισμός με την τεχνική της κινούμενης διακύμανσης.	
Τυπική απόκλιση μήκους πακέτου	Υπολογίζεται από τα παραπάνω.	
Συντελεστής διακύμανσης μήκους πακέτου	Υπολογίζεται από τα παραπάνω.	
Ασυμμετρία από διάμεσο μήκους πακέτου	Υπολογίζεται από τα παραπάνω.	Στον υπολογισμό χρησιμοποιείται η διάμεσος, άρα αξιοποιείται πιθανοτική δομή.
Ασυμμετρία από επικρατούν μήκος πακέτου	Υπολογίζεται από τα παραπάνω.	Στον υπολογισμό χρησιμοποιείται η επικρατούσα τιμή, άρα αξιοποιείται πιθανοτική δομή.
Μέσος χρόνος πακέτου	Υπολογισμός με την τεχνική του κινούμενου μέσου.	
Διάμεσος χρόνου πακέτου	Ένας πίνακας ανά flow για τους χρόνους. Σε αυτόν εφαρμόζεται η συνάρτηση διαμέσου.	Μία πιθανοτική δομή ανά flow (TDigest [17]) στην οποία εισάγονται οι χρόνοι.
Επικρατών χρόνος πακέτου	Δεν ορίζεται.	
Διακύμανση χρόνου πακέτου	Υπολογισμός με την τεχνική της κινούμενης διακύμανσης.	
Τυπική απόκλιση χρόνου πακέτου	Υπολογίζεται από τα παραπάνω.	
Συντελεστής διακύμανσης χρόνου πακέτου	Υπολογίζεται από τα παραπάνω.	
Ασυμμετρία από διάμεσο χρόνου πακέτου	Υπολογίζεται από τα παραπάνω.	Στον υπολογισμό χρησιμοποιείται η διάμεσος, άρα αξιοποιείται πιθανοτική δομή.
Ασυμμετρία από επικρατούντα χρόνο πακέτου	Δεν ορίζεται.	
Μέση χρονική διαφορά αιτήματος/απόκρισης	Υπολογισμός με την τεχνική του κινούμενου μέσου.	

Διάμεσος χρονικής διαφοράς αιτήματος/απόκρισης	Ένας πίνακας ανά flow για τις διαφορές. Σε αυτόν εφαρμόζεται η συνάρτηση διαμέσου.	Μία πιθανοτική δομή ανά flow (TDigest [17]) στην οποία εισάγονται οι διαφορές.
Επικρατούσα χρονική διαφορά αιτήματος/απόκρισης	Δεν ορίζεται.	
Διακύμανση χρονικής διαφοράς αιτήματος/απόκρισης	Υπολογισμός με την τεχνική της κινούμενης διακύμανσης.	
Τυπική απόκλιση χρονικής διαφοράς αιτήματος/απόκρισης	Υπολογίζεται από τα παραπάνω.	
Συντελεστής διακύμανσης χρονικής διαφοράς αιτήματος/απόκρισης	Υπολογίζεται από τα παραπάνω.	
Ασυμμετρία από διάμεσο χρονικής διαφοράς αιτήματος/απόκρισης	Υπολογίζεται από τα παραπάνω.	Στον υπολογισμό χρησιμοποιείται η διάμεσος, άρα αξιοποιείται πιθανοτική δομή.
Ασυμμετρία από επικρατούσα χρονική διαφορά αιτήματος/απόκρισης	Δεν ορίζεται.	

Πίνακας 2: Τρόπος υπολογισμού features

Η τεχνική του κινούμενου μέσου και της κινούμενης διακύμανσης είναι κομψές μαθηματικές εξισώσεις που επιτρέπουν την εύρεση των μεγεθών αυτών σε πραγματικό χρόνο από μία ροή δεδομένων, χωρίς να χρειάζεται μνήμη. Η μέση τιμή ενός μεγέθους x σε n δείγματα είναι:

$$avg_n = avg_{n-1} + \frac{(x_n - avg_{n-1})}{n}$$

Όπου avg_n είναι η μέση τιμή της ροής x συμπεριλαμβανομένου του πιο πρόσφατου δείγματος x_n. Η σχέση αυτή είναι αναδρομική και για την εύρεση της μέσης τιμής απαιτεί μόνο γνώση της παλιάς μέσης τιμής, τον συνολικό αριθμό δειγμάτων και φυσικά το τρέχον δείγμα.

Για την διακύμανση ο τύπος είναι ο εξής:

$$var_n = \frac{n-1}{n} * var_{n-1} + (x_n - avg_n) * \frac{x_n - avg_{n-1}}{n}$$

Οι μεταβλητές που εμπλέκονται έχουν παρόμοια λογική με τον τύπο για τη μέση τιμή.

Οι υπόλοιπες ιδιότητες δεν υπολογίζονται απευθείας, αλλά εισάγονται στις κατάλληλες ή λίστες στην ντετερμινιστική περίπτωση. Ο υπολογισμός τους γίνεται κατ' απαίτηση, όπως θα εξηγηθεί στην επόμενη ενότητα.

3.4.2 Εξαγωγή στατιστικών ιδιοτήτων και πρόβλεψη

Όπως φαίνεται στο σχήμα, μετά την ανανέωση των features και τις εισαγωγές στις δομές, ακολουθεί το στάδιο της πρόβλεψης. Αναλυτικότερα, κάθε flow που δέχθηκε νέα πακέτα περνάει τα features του από το μοντέλο μάθησης, το οποίο ταξινομεί το flow ως κακόβουλο ή καλόβουλο.

Κεφάλαιο 4

Πειράματα

Έχοντας γνώση της της λογικής της εργασίας μπορούμε πλέον να προβούμε στην πειραματική διαδικασία.

4.1 Κατηγοριοποίηση του dataset

Όπως έχει ήδη αναφερθεί, από το dataset [2] αξιοποιήθηκαν οι δικτυακές καταγραφές, σε μορφή pcap. Κάθε καταγραφή είναι μία συλλογή από παρόμοια flows (όλα έχουν το ίδιο label), που έχουν παραχθεί με συγκεκριμένο τρόπο. Στην εργασία χρησιμοποιήθηκαν καταγραφές από τις εξής κατηγορίες:

- Malicious (κακόβουλες): Πρόκειται για καταγραφές που περιέχουν πλήρως flows DNS tunneling. Με τη σειρά τους χωρίζονται σε δύο υποκατηγορίες, τις καταγραφές που προέκυψαν από dns2tcp, και αυτές που προέκυψαν από iodine. Και τα δύο είναι πακέτα λογισμικού που χρησιμοποιούνται για DNS tunneling.
- Benign (καλόβουλες): Καταγραφές που περιέχουν καλόβουλα flows, είναι αποτέλεσμα DNS ερωτημάτων που συνέβησαν όταν οι χρήστες θέλησαν να προσπελάσουν μία ιστοσελίδα από έναν browser.
- Non DoH: Πρόκειται για τυχαία HTTPS κίνηση που σχετίζεται με την επικοινωνία του χρήστη με διάφορες ιστοσελίδες. Δεν είναι DoH, αλλά όπως έχει αναφερθεί θεωρείται ίδια κατηγορία με την καλόβουλη κίνηση. Στο dataset παρήχθη με την χρήση διαφόρων browsers. Στην εργασία χρησιμοποιήθηκε ένα υποσύνολο των καταγραφών, που προέκυψαν από την χρήση δύο δημοφιλών browsers, Google Chrome και Mozilla Firefox.

Για την παραγωγή καλόβουλης και μη DoH κίνησης στα πλαίσια του εργαστηρίου, αναπαράχθηκαν συνθήκες ενός τυπικού χρήστη του διαδικτύου, ο οποίος επιχειρεί να προσπελάσει διάφορες ιστοσελίδες μέσω ενός browser. Κάνοντας το αυτό προκαλεί αιτήματα DNS (over HTTPS, καθώς ο browser έχει την αντίστοιχη ρύθμιση) και φυσικά απλά μηνύματα HTTPS, για την επικοινωνία του με την εκάστοτε ιστοσελίδα. Ωστόσο, η παραγωγή tunneling κίνησης είναι μία διαδικασία που απαιτεί εξειδικευμένο λογισμικό, και χρήζει περαιτέρω ανάλυσης.

4.1.1 Παραγωγή κακόβουλης κίνησης

Τα κακόβουλα flows που μελετώνται στο παρόν δημιουργήθηκαν στην έρευνα [2] χρησιμοποιώντας τα εξής εργαλεία λογισμικού:

- dns2tcp: Το dns2tcp είναι λογισμικό που επιτρέπει την ενθυλάκωση κίνησης TCP σε ερωτήματα DNS. Λειτουργεί με βάση την client-server αρχιτεκτονική [27].
- iodine: Το iodine είναι επίσης tunneling εφαρμογή, επιτρέποντας ενθυλάκωση IPv4 κίνησης σε πακέτα DNS [28].

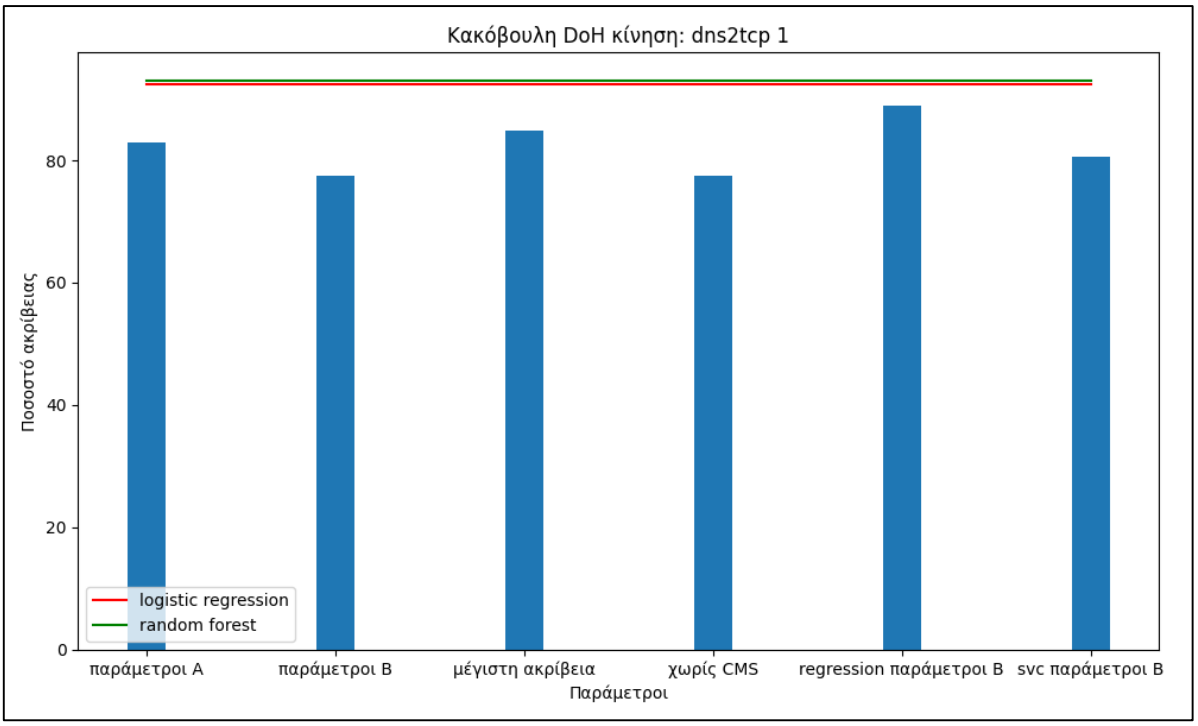
4.2 Διαγράμματα ακρίβειας προβλέψεων

Η πιθανοτική έκδοση του προγράμματος δοκιμάστηκε με ποικιλία παραμέτρων σε σύγκριση με την ντετερμινιστική. Κάθε πείραμα αφορά μία καταγραφή από το dataset, η οποία λειτουργεί ως input στο πρόγραμμα που εξετάζεται. Έτσι, για κάθε καταγραφή προκύπτει ένα διάγραμμα με την ακρίβεια που πέτυχε κάθε πιθανοτική έκδοση. Στο ίδιο διάγραμμα υπερτίθενται τα αποτελέσματα από τον ντετερμινιστικό αλγόριθμο, με όσα μοντέλα αυτός δοκιμάστηκε. Συγκεκριμένα οι μπάρες αφορούν τις πιθανοτικές υλοποιήσεις ενώ οι ευθείες γραμμές τις ντετερμινιστικές. Συνεπώς γίνεται να συγκριθούν οι πιθανοτικές εκτελέσεις με διαφορετικές παραμέτρους μεταξύ τους, ενώ ταυτόχρονα φαίνεται η σχέση με τα ντετερμινιστικά αποτελέσματα.

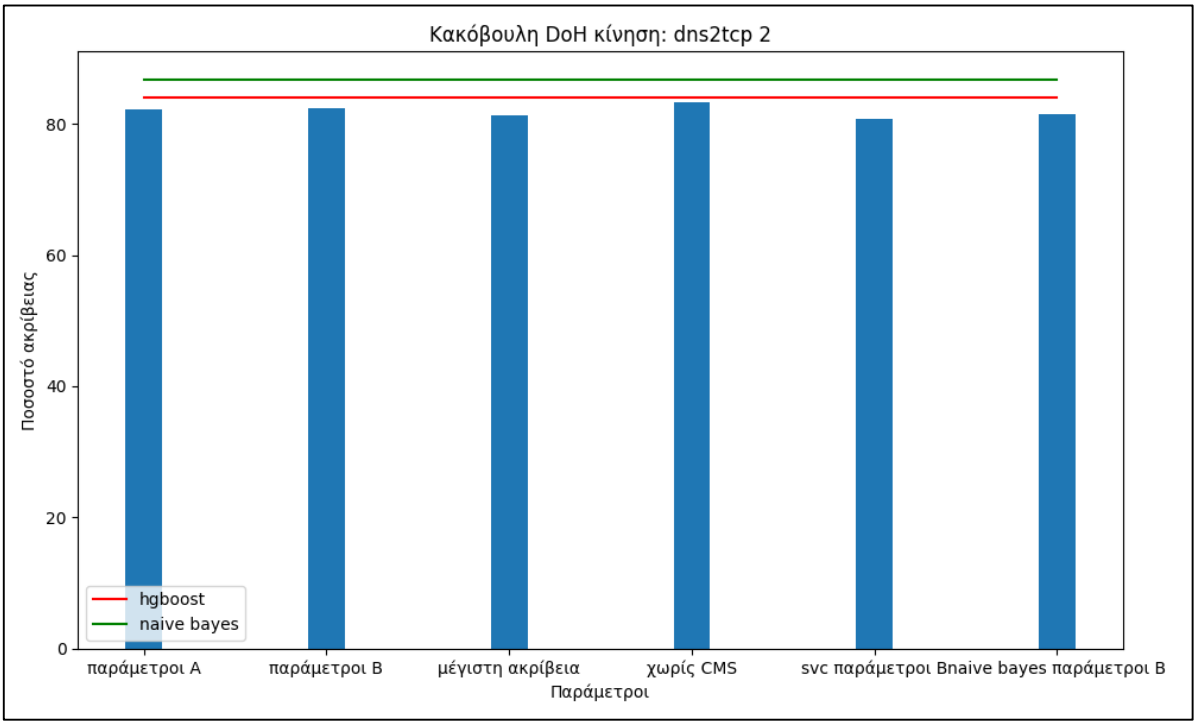
Τα διαγράμματα που ακολουθούν περιλαμβάνουν μόνο ένα δείγμα των πειραμάτων που διεξήχθησαν. Επιλέχθηκαν τα πιο αντιπροσωπευτικά από κάθε κατηγορία. Ύστερα από το καθένα θα γίνεται συνοπτικός σχολιασμός ορισμένων χρήσιμων χαρακτηριστικών και στο τέλος θα γίνει συνολική σύνοψη. Περιγράφονται εδώ οι διάφορες παράμετροι:

Στον άξονα x η σημείωση κάτω από κάθε μπάρα δηλώνει τον αλγόριθμο ML και την κατηγορία παραμέτρων που χρησιμοποιήθηκαν στο πείραμα. Όπου δεν αναγράφεται αλγόριθμος ML εννοείται ο Histogram-based Gradient Boosting Classification Tree (hgboost). Πρόκειται για το μοντέλο με την μεγαλύτερη αξιοπιστία στα περισσότερα πειράματα. Σχετικά με τις κατηγορίες παραμέτρων, υπάρχουν οι εξής:

- Παράμετροι Α: Συνδυασμός παραμέτρων μέτριας ακρίβειας. Η δομές T-Digest έχουν παράμετρο (1, 20) και η Top-K (0.1, 0.5).
- Παράμετροι Β: Συνδυασμός που βελτιστοποιεί τη σχέση ακρίβειας ως προς απαιτούμενη μνήμη. Οι δομές δοκιμάστηκαν σε μεμονωμένο περιβάλλον και πειραματικά διαπιστώθηκε ότι δίνουν τις καλύτερες προσεγγίσεις για τα μεγέθη τους, παραμένοντας σε εύλογα ποσά μνήμης, η T-Digest με κενή κλήση, δηλαδή όταν δημιουργείται με τις default ρυθμίσεις της, και η Top-K με (0.01, 0.5).
- Μέγιστη ακρίβεια: Εδώ και οι δύο δομές δημιουργήθηκαν με μοναδικό γνώμονα την βελτιστοποίηση της ακρίβειας. Αυτές οι παράμετροι είναι για την T-Digest (0.000001, 20000000) και για την Top-K (0.01, 0.99).
- Χωρίς CMS: Πρόκειται για τις παραμέτρους Β, αλλά η καταμέτρηση των bytes που εστάλησαν και ελήφθησαν γίνεται ντετερμινιστικά, δηλαδή υπάρχουν δύο ακέραιοι για κάθε flow που κάνουν την καταμέτρηση. Ο λόγος που διεξήχθη αυτό το πείραμα αναλύεται διεξοδικά στο κεφάλαιο της μνήμης.

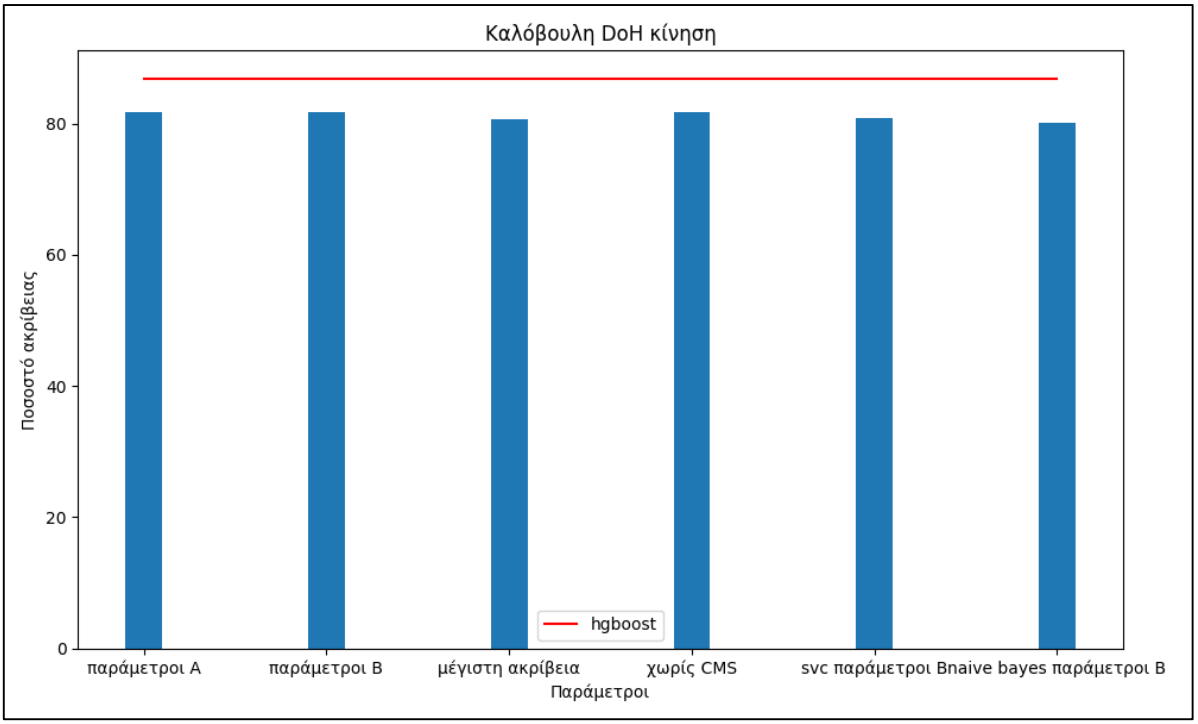


Σε αυτό το πείραμα input είναι η πρώτη dns2tcp καταγραφή, η οποία περιλαμβάνει μεγάλο πλήθος flows, με (ως επί το πλείστον) μικρό αριθμό πακέτων. Αξιοσημείωτη είναι η απόδοση του αλγορίθμου μάθησης logistic regression, τόσο στην ντετερμινιστική όσο και στην πιθανοτική υλοποίηση. Επίσης οι παράμετροι μέγιστης ακρίβειας πετυχαίνουν αξιόλογα αποτελέσματα. Οι υπόλοιπες παράμετροι κυμαίνονται περίπου στα ίδια επίπεδα. Ο random forest έχει σχεδόν την ίδια ακρίβεια με τον logistic regression στην ντετερμινιστική περίπτωση.



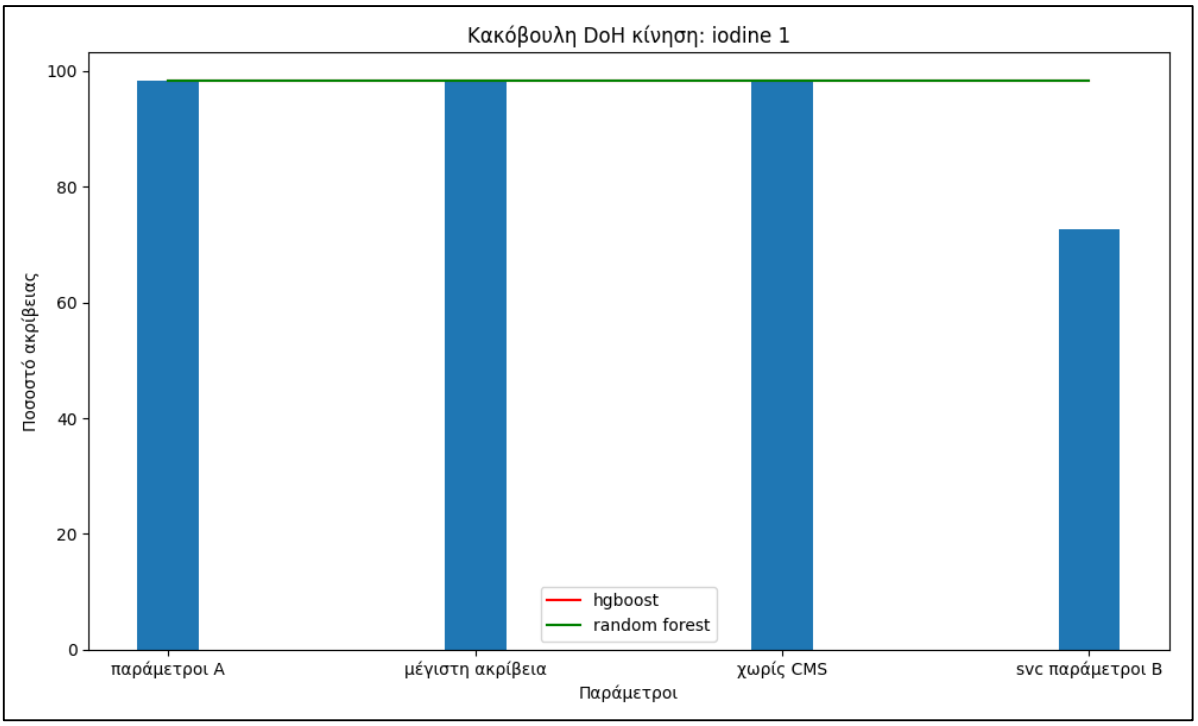
Για ακόμα μία φορά κάποιο μοντέλο, στην προκειμένη ο Naïve Bayes classifier, καταφέρνει να ξεπεράσει τον hgboost στην ντετερμινιστική υλοποίηση. Ωστόσο, στην πιθανοτική όλες οι παράμετροι έχουν πολύ παρόμοια ποσοστά ακρίβειας, ανεξάρτητα από το μοντέλο. Η υλοποίηση χωρίς CMS είναι η βέλτιστη μεταξύ των πιθανοτικών, χωρίς όμως σημαντική διαφορά.

Αξίζει να σημειωθεί η μείωση της απόστασης ανάμεσα στις πιθανοτικές υλοποιήσεις με την ντετερμινιστική σε σχέση με την προηγούμενη καταγραφή, παρά την παρόμοια φύση των πακέτων τους. Η ειδοποιός διαφορά των δύο είναι στο μέγεθος των flows. Εδώ τα flows έχουν αρκετά μεγαλύτερο αριθμό πακέτων κατά μέσο όρο.



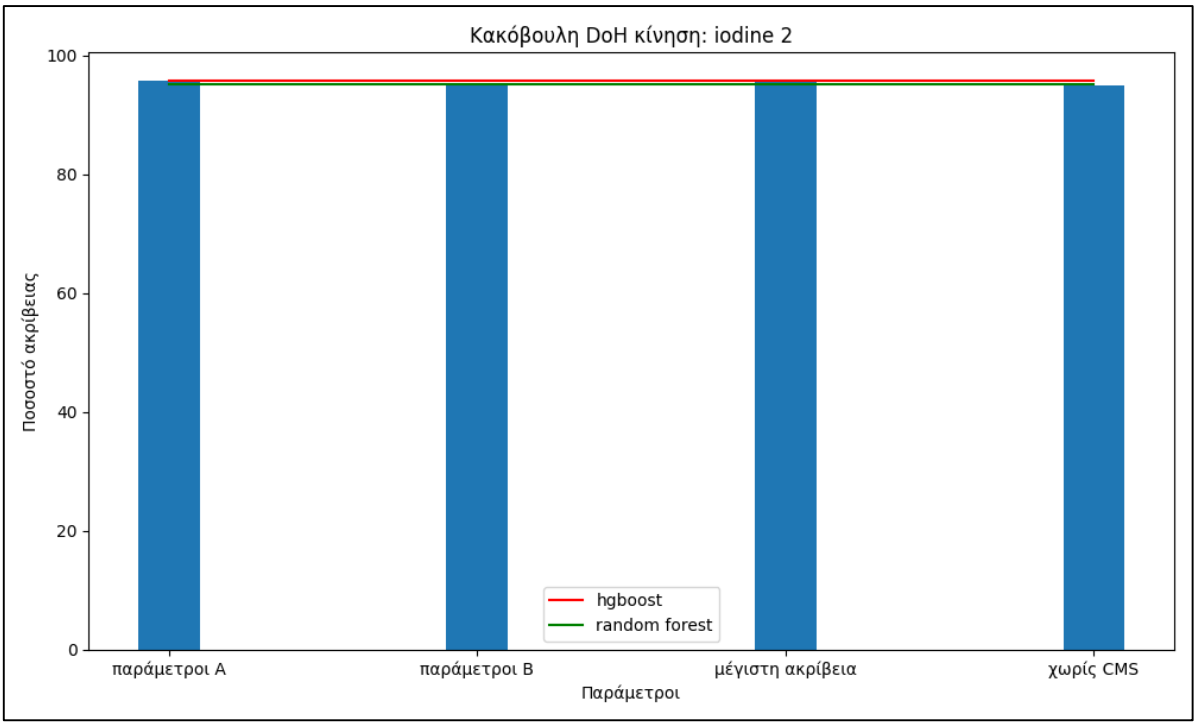
Σχήμα 9: Διάγραμμα ακρίβειας καλόβουλης DoH κίνησης

Σχετικά με την καλόβουλη DoH κίνηση, παρατηρείται πολύ μεγάλη ομοιομορφία στις πιθανοτικές υλοποιήσεις, όπου δεν διαδραματίζουν μεγάλο ρόλο οι επιλογές παραμέτρων και μοντέλων. Το μέγεθος των flows είναι στη συντριπτική πλειοψηφία πολύ μικρό, με μέσο αριθμό πακέτων 71. Αν εξαιρεθούν τα 5% μικρότερα και μεγαλύτερα flows, ο μέσος αριθμός πακέτων πέφτει στα 37.

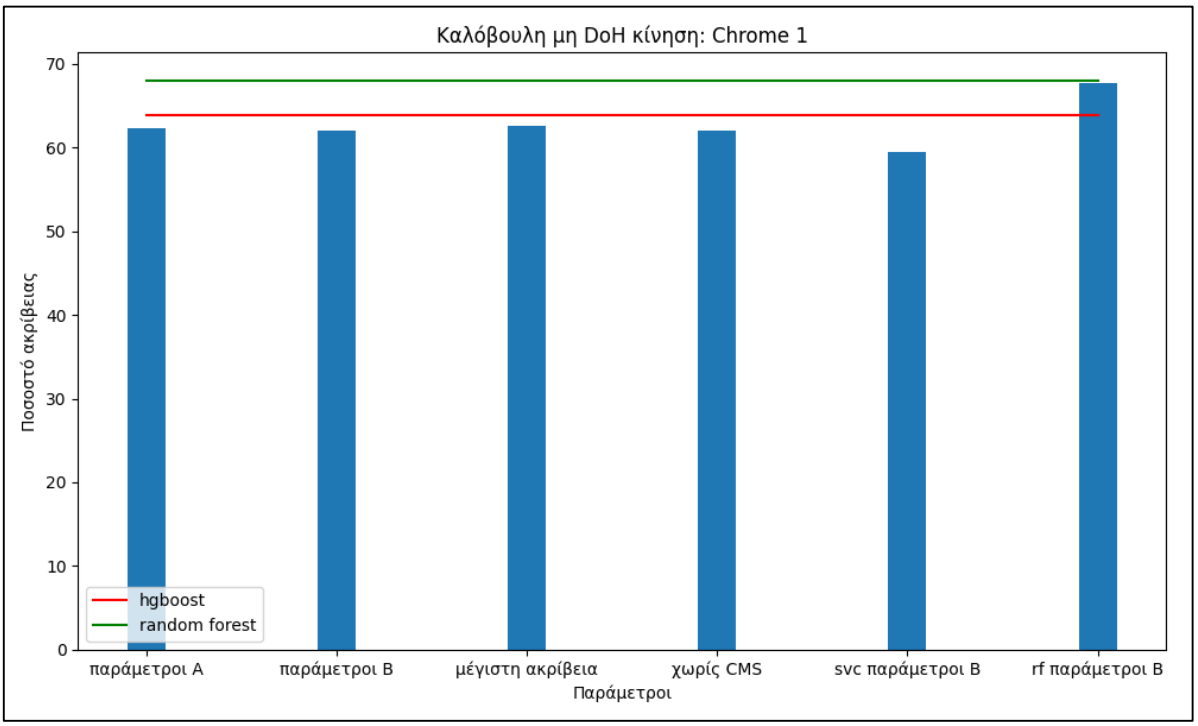


Σχήμα 10: Διάγραμμα ακρίβειας iodine 1

Οι iodine καταγραφές περιλαμβάνουν flows με πολύ μεγάλο αριθμό πακέτων. Η μέση τιμή πακέτων είναι 8000, και ακόμα και οι trimmed means, αν και μειώνονται, εξακολουθούν να είναι στις τάξεις των χιλιάδων. Επίσης, λόγω της εργαστηριακής φύσης των flows, έχουν αρκετά κοινά χαρακτηριστικά. Έτσι, η σύγκλιση των features γίνεται με μεγάλη επιτυχία, επιτρέποντας στα μοντέλα να κάνουν σχεδόν πλήρως σωστές προβλέψεις. Έχουν χρησιμοποιηθεί δύο ντετερμινιστικοί αλγόριθμοι που έχουν ακριβώς ίδια ακρίβεια, όπως και οι πιθανοτικοί. Μοναδική εξαίρεση είναι το μοντέλο Support Vector Machine, που κάνει σαφώς χειρότερες προβλέψεις. Σκόπιμα εμφανίζεται στο διάγραμμα, για να γίνει αντιληπτή η αδυναμία ορισμένων μοντέλων ML στα πειράματα.



Η δεύτερη iodine καταγραφή εμφανίζει πολλές ομοιότητες με την πρώτη, άρα τα παρόμοια αποτελέσματα δεν χρειάζονται ξεχωριστή ανάλυση.



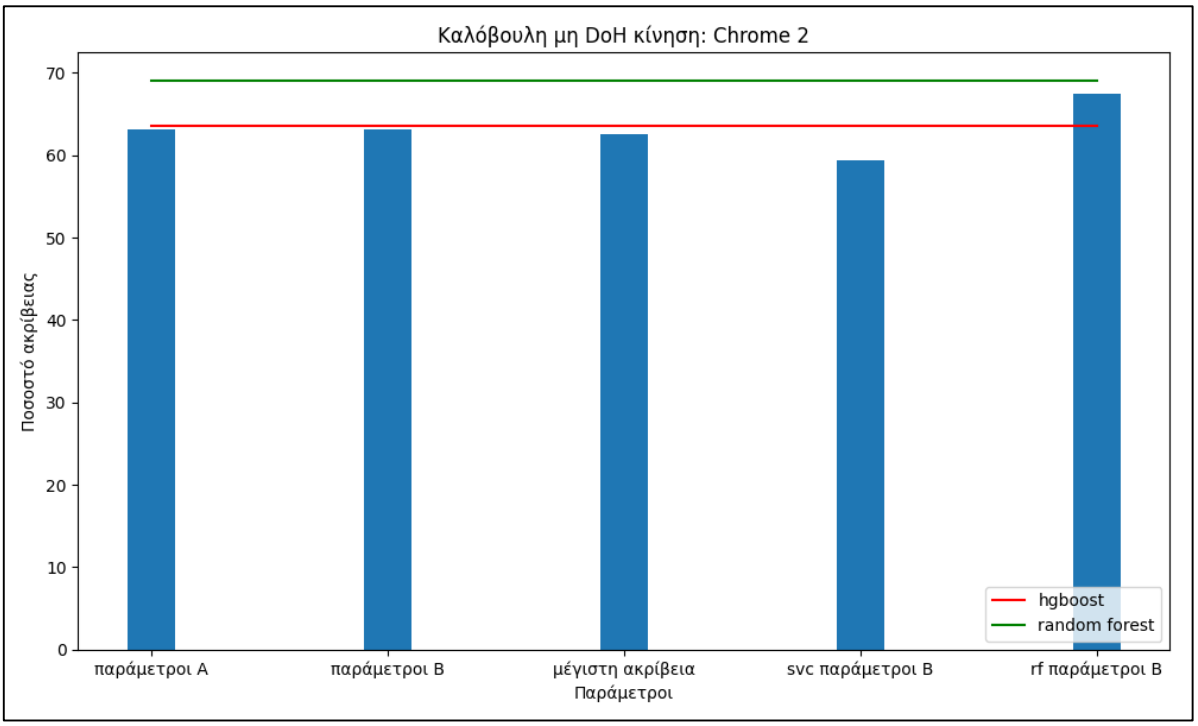
Στο πρώτο διάγραμμα της μη DoH κίνησης μπορούν να γίνουν ορισμένες πρωτοφανείς παρατηρήσεις.

Τα ποσοστά ακρίβειας είναι αισθητά μικρότερα σε σχέση με τις υπόλοιπες καταγραφές. Αυτό οφείλεται στο γεγονός ότι πρόκειται για μη DoH κίνηση, δηλαδή τα πακέτα δεν εμφανίζουν χαρακτηριστικά που αντιστοιχούν σε συμπεριφορά DNS επικοινωνίας. Στις έρευνες που

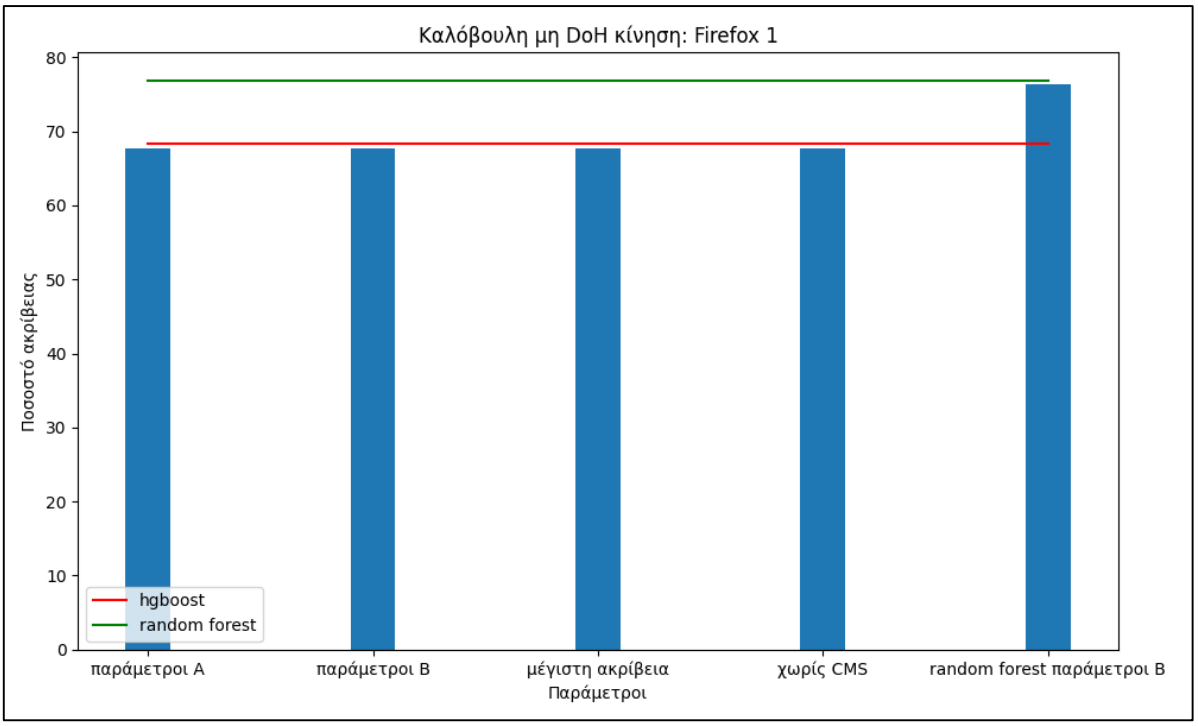
αναφέρθηκαν, χρησιμοποιείται διπλή ταξινόμηση, ώστε πρώτα να αφαιρεθούν αυτά τα *προβληματικά* flows. Όμως σύμφωνα με τις παραδοχές και τους περιορισμούς της εργασίας, απορρίφθηκε η διπλή ταξινόμηση. Η ακρίβεια εξακολουθεί να κυμαίνεται σε τιμές σημαντικά μεγαλύτερες του 50%, άρα τα αποτελέσματα δεν είναι σε καμία περίπτωση τυχαία. Άλλωστε έχει χρησιμοποιηθεί δείγμα μη DoH κίνησης στην εκπαίδευση.

Επίσης, ο ML αλγόριθμος random forest πετυχαίνει σημαντικά καλύτερα ποσοστά από τον hgboost. Μάλιστα η πιθανοτική υλοποίηση του random forest ξεπερνά ακόμα και την ντετερμινιστική του hgboost, κάτι που αναδεικνύει την αποδοτικότητα της σε αυτή την κατηγορία flows. Αντίθετα, το μοντέλο svc υπολειτουργεί. Οι υπόλοιπες υλοποιήσεις δεν εμφανίζουν κάποια ιδιαίτερη συμπεριφορά.

Σημειώνεται πως το μέσο μέγεθος των flows είναι 190 πακέτα, ενώ αν εξαιρεθούν τα 10% μικρότερα και μεγαλύτερα flows η μέση τιμή γίνεται 55.

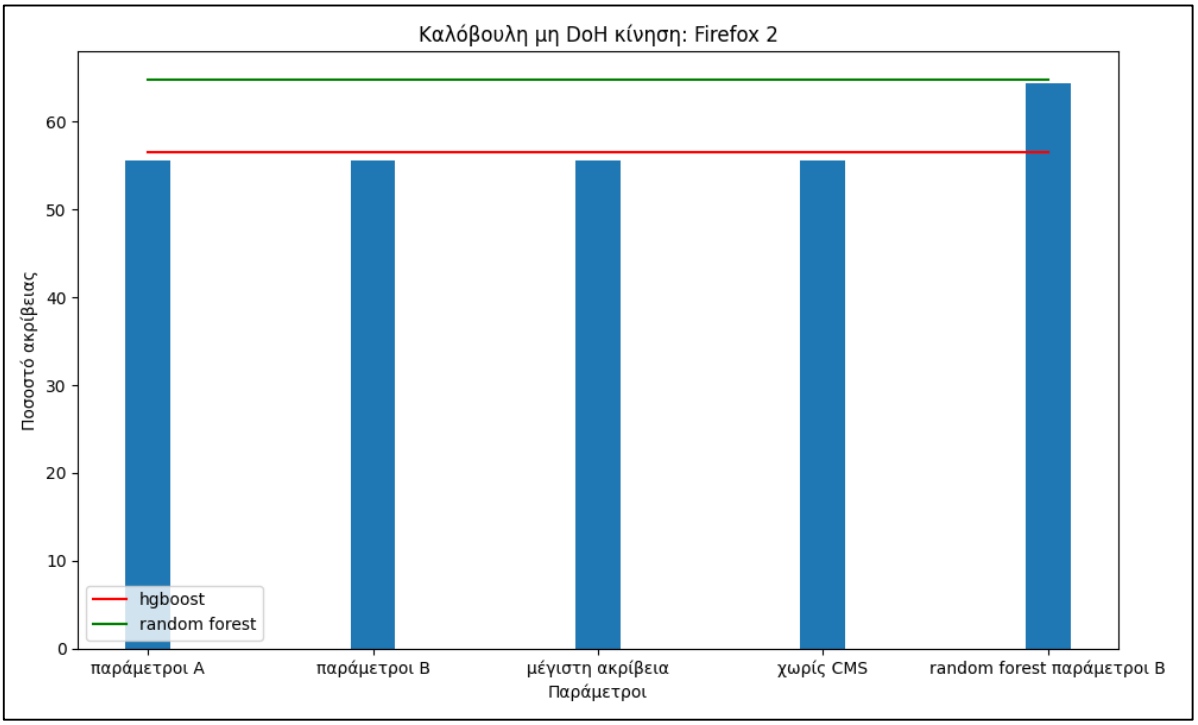


Τα αποτελέσματα για την δεύτερη καταγραφή που αφορά στο Google Chrome είναι σχετικά παρόμοια. Οι πιθανοτικές υλοποιήσεις είναι πολύ κοντά μεταξύ τους αλλά και με την αντίστοιχη ντετερμινιστική.



Το πρώτο διάγραμμα για το Mozilla Firefox παρουσιάζει κοινά χαρακτηριστικά με τα προηγούμενα, αν και πρόκειται για διαφορετικό browser. Ο random forest πετυχαίνει πολύ καλύτερα ποσοστά από τον hgboost σε όλες τις υλοποιήσεις, ενώ οι πιθανοτικές υλοποιήσεις του hgboost είναι πολύ κοντά μεταξύ τους. Η διαφορά ανάμεσα στις ντετερμινιστικές και πιθανοτικές είναι πολύ μικρή.

Τα μεγέθη των flows όμως διαφέρουν από αυτά του Chrome, με το μέσο flow να περιέχει 87 πακέτα, και εξαιρουμένων των 10% μικρότερων και μεγαλύτερων, 33 πακέτα.



Η δεύτερη Firefox καταγραφεί συνάδει πλήρως με τα αποτελέσματα της πρώτης.

4.3 Μνήμη

Σύμφωνα με την οπτική αναπαράσταση των αποτελεσμάτων, φαίνεται πως, σε γενικές γραμμές, η παραμετροποίηση των πιθανοτικών δομών δεν επηρεάζει δραματικά την απόδοση τους, η οποία είναι λίγο χειρότερη από την ντετερμινιστική υλοποίηση. Το ερώτημα που τίθεται είναι αν το κόστος ακρίβειας αντισταθμίζεται από μείωση στην απαιτούμενη μνήμη. Προκειμένου να απαντηθεί, θα συγκριθούν τα μεγέθη των διαφορετικών δομών που χρησιμοποιούνται κάθε φορά. Όλα τα μεγέθη που αναφέρονται θα είναι σε bytes, και θα αφορούν την τελική έκδοση της κάθε δομής, δηλαδή αφού έχουν εισαχθεί όλα τα δεδομένα.

Στην Python ο υπολογισμός της μνήμης των δομών δεδομένων γίνεται ως εξής:

- Ακέραιοι (integers): Οι απλοί ακέραιοι αριθμοί θεωρούνται αντικείμενα, άρα η μνήμη που απαιτείται για αυτούς περιέχει κάποιο overhead, ώστε να περιγραφούν οι ιδιότητες του αντικειμένου, πέραν της τιμής του. Ένας ακέραιος λοιπόν χρησιμοποιεί 28 bytes. Σημειώνεται ότι οι πολύ μεγάλοι αριθμοί χρειάζονται περισσότερη μνήμη, όμως στα πλαίσια της εργασίας δεν θα προκύψουν τέτοιοι, άρα κάθε ακέραιος μπορεί να θεωρηθεί ασφαλώς ότι καταλαμβάνει 28 bytes.
- Δεκαδικοί (floats): Οι δεκαδικοί αριθμοί θεωρούνται επίσης πλήρη αντικείμενα, άρα έχουν και αυτοί κάποιο overhead. Όμως, αντίθετα από τους ακέραιους το μέγεθος τους είναι προκαθορισμένο. Άρα από την αναπαράσταση τους ως αντικείμενα λείπει το πεδίο που δηλώνει το μέγεθος, αφού είναι αχρείαστο. Έτσι καταλήγουν να χρησιμοποιούν συνολικά 24 bytes, μέγεθος μικρότερο από το αντίστοιχο των ακεραίων.
- Λίστες (lists): Στην Python, οι λίστες έχουν δυναμικό μέγεθος. Αυτό σημαίνει πως όταν προστεθούν δεδομένα που θα έκαναν τη λίστα να υπερχειλίσει, αυτή επεκτείνεται ώστε να μπορέσει να δεχτεί ακόμα περισσότερα δεδομένα. Αυτό είναι ένας παράγοντας που καθιστά πολύ δύσκολο τον υπολογισμό της μνήμης με θεωρητικούς υπολογισμούς.

Έχοντας το παραπάνω συμπέρασμα, και με δεδομένο ότι η βασική δομή που χρησιμοποιείται στην ντετερμινιστική υλοποίηση είναι οι λίστες, η ανάλυση της μνήμης θα βασιστεί στο module pympler [29] της Python, και συγκεκριμένα στην συνάρτηση asizeof.

Επίσης, με σκοπό την διατήρηση απλότητας στις αριθμητικές πράξεις και αποφυγή περιττής ανάλυσης, υπολογίζονται μόνο μεγέθη που είναι διαφορετικά στις δύο υλοποιήσεις. Για παράδειγμα, δεν υπολογίζονται οι μέσες τιμές, καθώς η μνήμη τους είναι ίδια ανεξαρτήτως υλοποίησης.

4.3.1 Πιθανοτική υλοποίηση

Πρώτα θα αναλυθεί η πιθανοτική περίπτωση χρήσης, καθώς οι δομές διατηρούν σχεδόν σταθερό μέγεθος που εξαρτάται μόνο από τις παραμέτρους τους, όχι από την ποσότητα των δεδομένων που αποθηκεύουν. Βέβαια, αρμόζει να αναφερθεί ότι η δυναμικότητα της Python στον τομέα της μνήμης επηρεάζει και τις πιθανοτικές δομές, οι οποίες εσωτερικά χρησιμοποιούν απλούστερες δομές της γλώσσας. Οπότε ενδέχεται να υπάρχουν μικρές διαφορές στα μεγέθη, ανάλογα με το σύστημα στο οποίο γίνονται τα πειράματα, ή ακόμα και σε διαδοχικές εκτελέσεις στο ίδιο σύστημα.

Ανεξάρτητα από τις παραμέτρους, οι δομές που απαιτεί ο πιθανοτικός αλγόριθμος είναι:

- Δύο Count Min Sketches (CMS), ένα για την μέτρηση των bytes που εστάλησαν και ένα για τα bytes που ελήφθησαν.
- Τρεις δομές T-Digest (TD) ανά flow, για τον υπολογισμό των τριών απαιτούμενων διαμέσων (μήκους πακέτου, χρόνου, χρονικής διαφοράς αιτήματος απόκρισης). Σύμφωνα με τα αποτελέσματα των πειραμάτων, όλες οι TDigest δομές έχουν ίδιο, αμετάβλητο μέγεθος.
- Μία Top-K ανά flow για τον υπολογισμό του επικρατούντος μήκους πακέτου.

Επισημαίνεται ότι οι T-Digest και η Top-K είναι ανά flow, δηλαδή κάθε flow έχει τις δικές του δομές, ενώ τα CMS είναι δύο συνολικά, και τα αξιοποιούν όλα τα flows. Άρα η συνολική μνήμη που απαιτεί ένα flow είναι:

$$Μνήμη = 2 * \frac{CMS}{n} + 3 * TD + TopK$$

Ο όρος $\frac{1}{n}$ υποδηλώνει ότι η μνήμη που καταλαμβάνουν τα δύο CMS αφορά όλα τα (έστω n) flows. Όσα περισσότερα flows, τόσο περισσότερο μοιράζεται το κόστος του CMS. Ωστόσο, επειδή ο αριθμός των flows ποικίλει ανάλογα με το πείραμα, ο όρος που αφορά στα CMS δεν θα ληφθεί υπόψιν στους προσεχείς υπολογισμούς, και θα αναλυθεί αργότερα. Ακολουθεί η ανάλυση κάθε δομής με βάση τις παραμέτρους.

Επειδή το μέγεθος της T-Digest είναι συνάρτηση μόνο των παραμέτρων της, συμφέρει η περιεκτική απεικόνιση της μνήμης σε έναν πίνακα:

Παράμετροι	**Μνήμη**
Κενές παράμετροι ()	648
(1, 20)	656
(0.000001, 20000000)	808

Table 2: Μνήμη T-Digest συναρτήσει των παραμέτρων

Οι προεπιλεγμένες ρυθμίσεις της δομής είναι βέλτιστες από την άποψη χρησιμοποίησης μνήμης. Όσο χειραγωγούνται οι παράμετροι προς αύξηση της ακρίβειας παρατηρείται όπως αναμένεται αύξηση στην απαιτούμενη μνήμη, η οποία όμως δεν είναι ιδιαίτερα μεγάλη.

Η Top-K δεν εμφανίζει την ίδια σταθερότητα, αλλά εναλλάσσεται ανάλογα με τον αριθμό των πακέτων του flow σε συγκεκριμένες τιμές. Αυτές οι εναλλαγές δεν εξαρτώνται πλήρως από τον αριθμό πακέτων, οπότε είναι ως ένα βαθμό απρόβλεπτες. Αυτές οι παρατηρήσεις θα φανούν καλύτερα στα παρακάτω διαγράμματα.

Παράμετροι	**Τιμές μνήμης**
eps = 0.10, εμπιστοσύνη = 0.5	2576, 2608, 2640, 2760, 2864, 2896, 2928, 3016
eps = 0.01, εμπιστοσύνη = 0.5	3296, 3328, 3360, 3392, 3584, 3616, 3648
eps = 0.00001, εμπιστοσύνη = 0.99	~5.5 MB

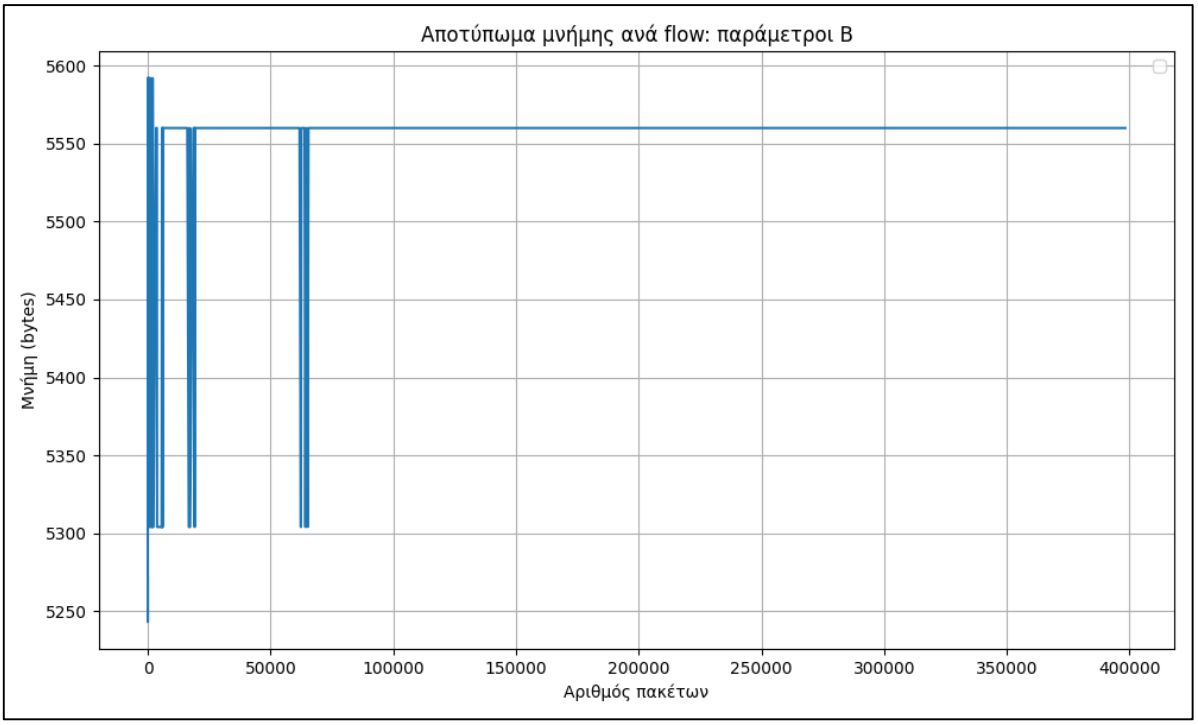
Πίνακας 1: Πιθανές τιμές μνήμης Top-K με βάση τις παραμέτρους

Σε αντίθεση με την T-Digest, εδώ η μνήμη έχει μεγάλη ευαισθησία στις παραμέτρους. Ειδικά για την μέγιστη ακρίβεια που μπορεί να επιτευχθεί, το μέγεθος της δομής εκτινάσσεται, καθώς αυξάνεται πάνω από χίλιες φορές σε σχέση με τις προηγούμενες παραμέτρους.

Στα πειράματα που έγιναν χρησιμοποιήθηκαν ορισμένοι συνδυασμοί των παραμέτρων T-Digest και Top-K που καταγράφηκαν. Οι συνδυασμοί σχολιάζονται στις επόμενες παραγράφους.

Παράμετροι B

Οι παράμετροι αυτοί σημαίνουν ότι οι δομές T-Digest δημιουργούνται με τις προεπιλεγμένες τους ρυθμίσεις (κενή κλήση του constructor), και η Top-K καλείται με eps 0.01 και εμπιστοσύνη 0.5, που είναι η ελάχιστη τιμή. Σε αυτή την περίπτωση οι δομές T-Digest έχουν όλες μέγεθος 648 bytes, ενώ η Top-K εμφανίζει ποικίλες τιμές, που εξαρτώνται κυρίως από τον αριθμό των πακέτων στο εκάστοτε flow. Εντούτοις, η διακύμανση είναι πολύ μικρή. Ενδεικτικά, σε πειράματα με εκατοντάδες flows που είχαν από μονοψήφιο αριθμό πακέτων μέχρι δεκάδες χιλιάδες, οι διάφορες Top-K χρησιμοποιούσαν μία από τις παρακάτω τιμές μνήμης: 3296, 3328, 3360, 3392, 3584, 3616, 3648. Η μέγιστη απόκλιση είναι κάτω από 400 bytes. Παραστατικά:

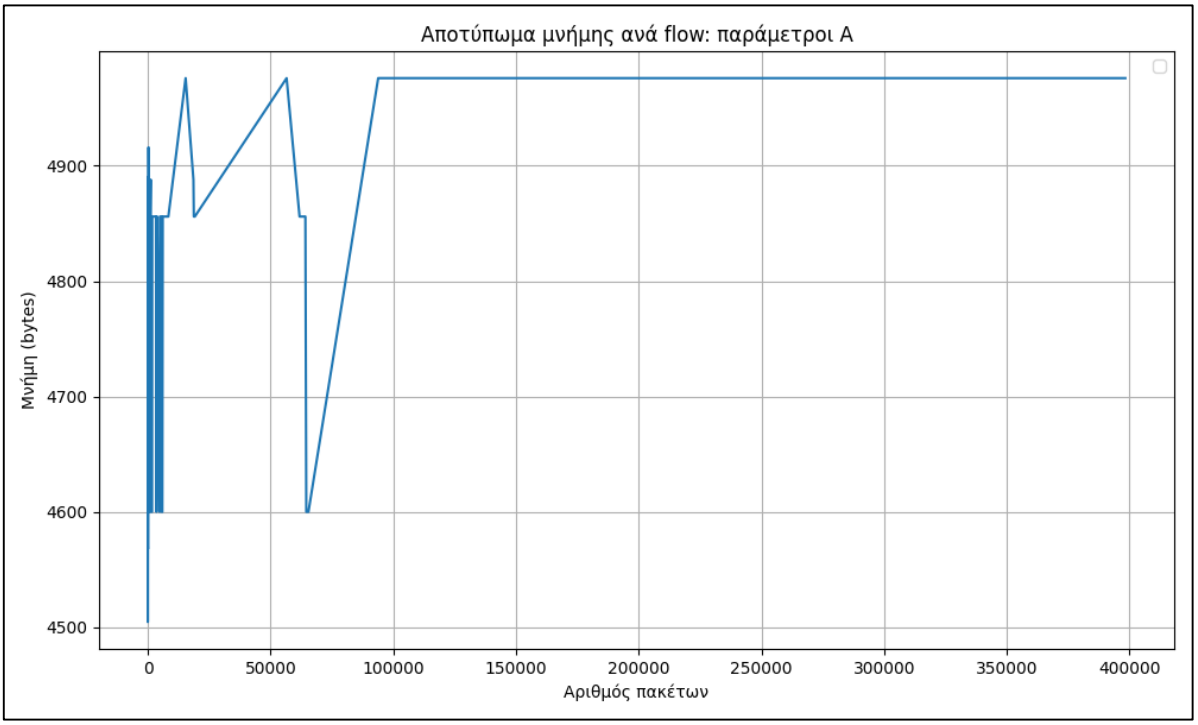


Σχήμα 11: Μνήμη πιθανοτικών δομών (ανά flow) παράμετροι B

Το διάγραμμα συνάδει πλήρως με την ανάλυση που διεξήχθη. Η συνολική μνήμη (ανά flow) παραμένει σταθερή, ανεξάρτητα από τον αριθμό πακέτων. Έχει γίνει προοικονομία για τις διακυμάνσεις στο μέγεθος της Top-K, οι οποίες φαίνονται λόγω του πάχους των γραμμών, που δηλώνουν απότομες αυξομειώσεις. Όσο αυξάνεται ο αριθμός των πακέτων, τα δείγματα από flows ήταν συγκριτικά λιγότερα, οπότε τυχαίνει να μην υπάρχουν διακυμάνσεις. Αξιοσημείωτο είναι για κάποιους πολύ μικρούς αριθμούς πακέτων η Top-K σημειώνει μεγαλύτερο αποτύπωμα στη μνήμη ακόμα και από flows με εκατοντάδες χιλιάδες πακέτα. Αυτή η συμπεριφορά οφείλεται στην ορισμένη τυχαιότητα που διέπει τη δομή καθώς και στον τρόπο με τον οποίο αποδίδει μνήμη η Python. Υπενθυμίζεται ότι η T-Digest δεν μεταβάλλει το μέγεθος της, γι’ αυτό αναλύεται μόνο η συμπεριφορά της Top-K.

Παράμετροι A

Σε αυτή την περίπτωση η T-Digest καλείται με παραμέτρους (1,20). Η αλλαγή αυξάνει ελαφρώς την ακρίβεια της, όπως και το αποτύπωμα της στη μνήμη. Κάθε T-Digest χρειάζεται πλέον 656 bytes. Η Top-K έχει παραμέτρους eps 0.1, δέκα φορές μεγαλύτερο από πριν, και ίδια εμπιστοσύνη, 0.5. Αυτό μείωσε αισθητά το μέσο μέγεθος της δομής, η οποία πλέον καταλαμβάνει 2576, 2608, 2640, 2760, 2864, 2896, 2928, 3016 bytes. Όπως και προηγουμένως, τα flows με τα περισσότερα πακέτα συνεπάγονται (εν γένει) μεγαλύτερο μέγεθος της Top-K, αλλά στα μικρότερα flows η διακύμανση είναι σχετικά απρόβλεπτη.

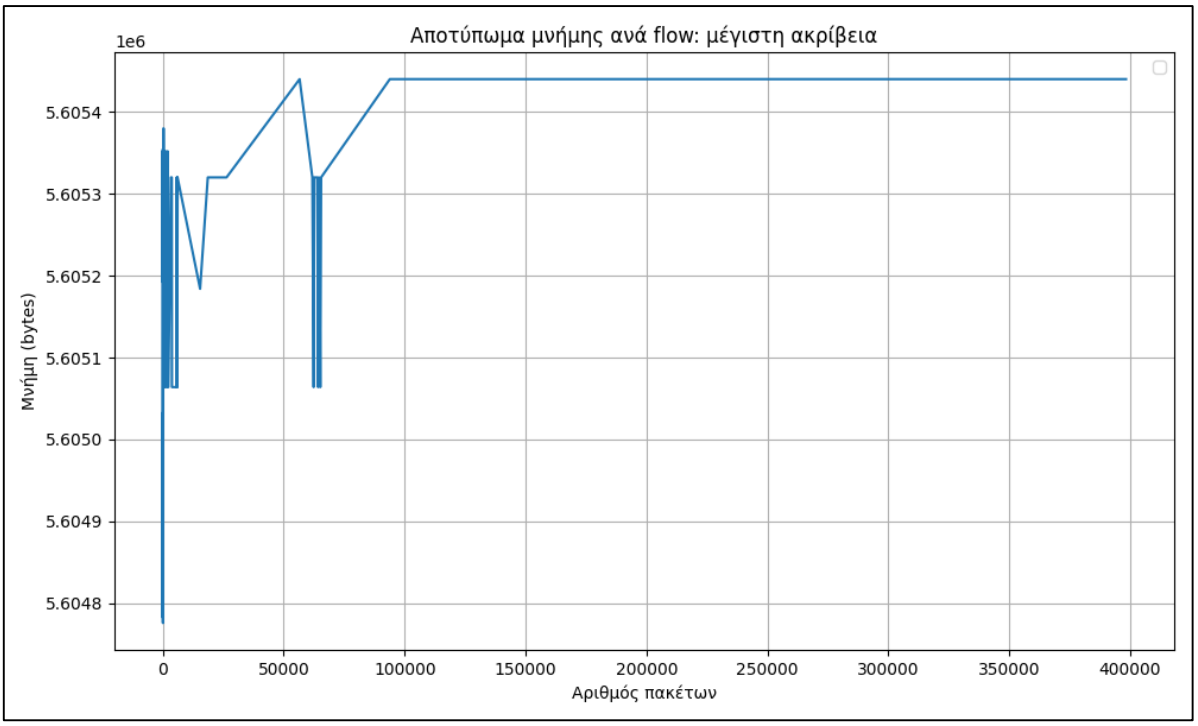


Σχήμα 12: Μνήμη πιθανοτικών δομών (ανά flow) παράμετροι A

Το σχήμα είναι, όπως αναμενόταν, παρόμοιο με το προηγούμενο. Αξιοσημείωτη διαφορά είναι ότι σε αυτή την περίπτωση η Top-K αποκτά το μέγιστο μέγεθος στους μεγάλους αριθμούς πακέτων. Επίσης η κλίμακα μνήμης είναι μικρότερη από προηγουμένως, αποτέλεσμα της χαλάρωσης των παραμέτρων της Top-K.

Μέγιστη ακρίβεια

Σε αυτό το ζευγάρι παραμέτρων και οι δύο δομές λειτουργούν με τη μέγιστη δυνατή απόδοση. Φυσικά, αυτό συνεπάγεται και μέγιστη κατάχρηση μνήμης.



Σχήμα 13: Μνήμη πιθανοτικών δομών (ανά flow) μέγιστη ακρίβεια

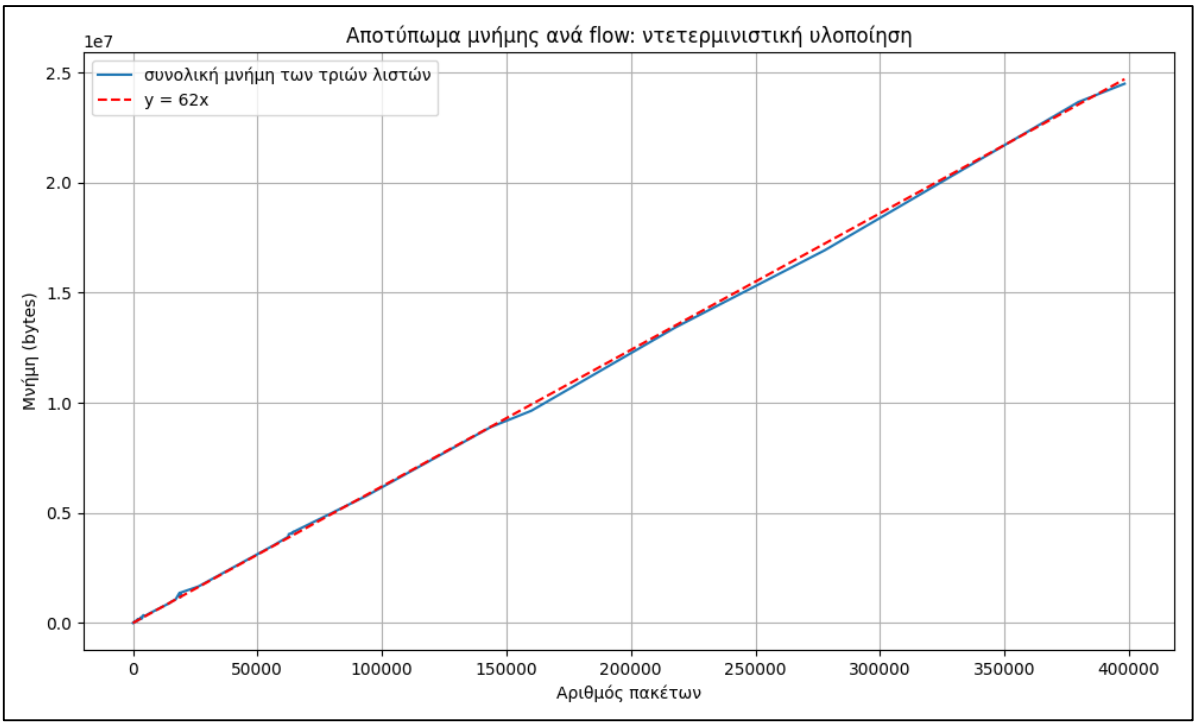
Πράγματι, ενώ σχηματικά το διάγραμμα μοιάζει με τα προηγούμενα, η αλλαγή στην κλίμακα είναι δραματική. Η μνήμη σε αυτή την περίπτωση δεν κυμαίνεται σε μερικές χιλιάδες bytes, αλλά σε μερικά εκατομμύρια.

4.3.2 Ντετερμινιστική υλοποίηση

Όπως εξηγήθηκε παραπάνω, αν και σε αυτή την υλοποίηση πρακτικά υπάρχουν μόνο λίστες αριθμών που πρέπει να αναλυθούν, η θεωρητική μελέτη ενδέχεται να οδηγήσει σε λανθασμένα συμπεράσματα. Συνεπώς εφαρμόστηκε παρόμοια ανάλυση με την πιθανοτική περίπτωση. Για ένα ευρύ φάσμα αριθμού πακέτων καταγράφηκαν τα τελικά μεγέθη των δομών που χρησιμοποιήθηκαν. Αναλυτικά, οι δομές που χρειάζεται κάθε flow είναι τρεις λίστες. Μία λίστα ακεραίων, για τα μήκη πακέτων, και δύο λίστες δεκαδικών (floats) για τους χρόνους. Σε αυτές εφαρμόζονται όλες οι απαραίτητες συναρτήσεις για την εύρεση των στατιστικών. Επίσης, χρησιμοποιούνται δύο μεταβλητές ανά flows, μία για να μετράει bytes που εστάλησαν και μία για bytes που ελήφθησαν. Ο τύπος που δίνει τη συνολική μνήμη για ένα flow είναι:

$$Μνήμη = list_{len} + list_{time} + list_{rr} + 2 * int$$

Όπου $list_{len}$ είναι το μέγεθος της λίστας των μηκών πακέτων, $list_{time}$ είναι το μέγεθος της λίστας των χρόνων και $list_{rr}$ είναι το μέγεθος της λίστας των χρονικών διαφορών αιτήματος απόκρισης. Με τον όρο int δηλώνεται το μέγεθος ενός ακεραίου, που είναι 28 bytes. Όπως και προηγουμένως, η μνήμη που απαιτείται για τα bytes θα μείνει στο περιθώριο της ανάλυσης, προσωρινά. Παρατίθενται τα διαγράμματα που αναπαριστούν την ντετερμινιστική μνήμη. Στο πρώτο, η συνολική μνήμη που χρειάζεται ένα flow για τις τρεις λίστες, ανάλογα με τον αριθμό των πακέτων. Στο δεύτερο, οι τρεις λίστες αναπαρίστανται χωριστά.

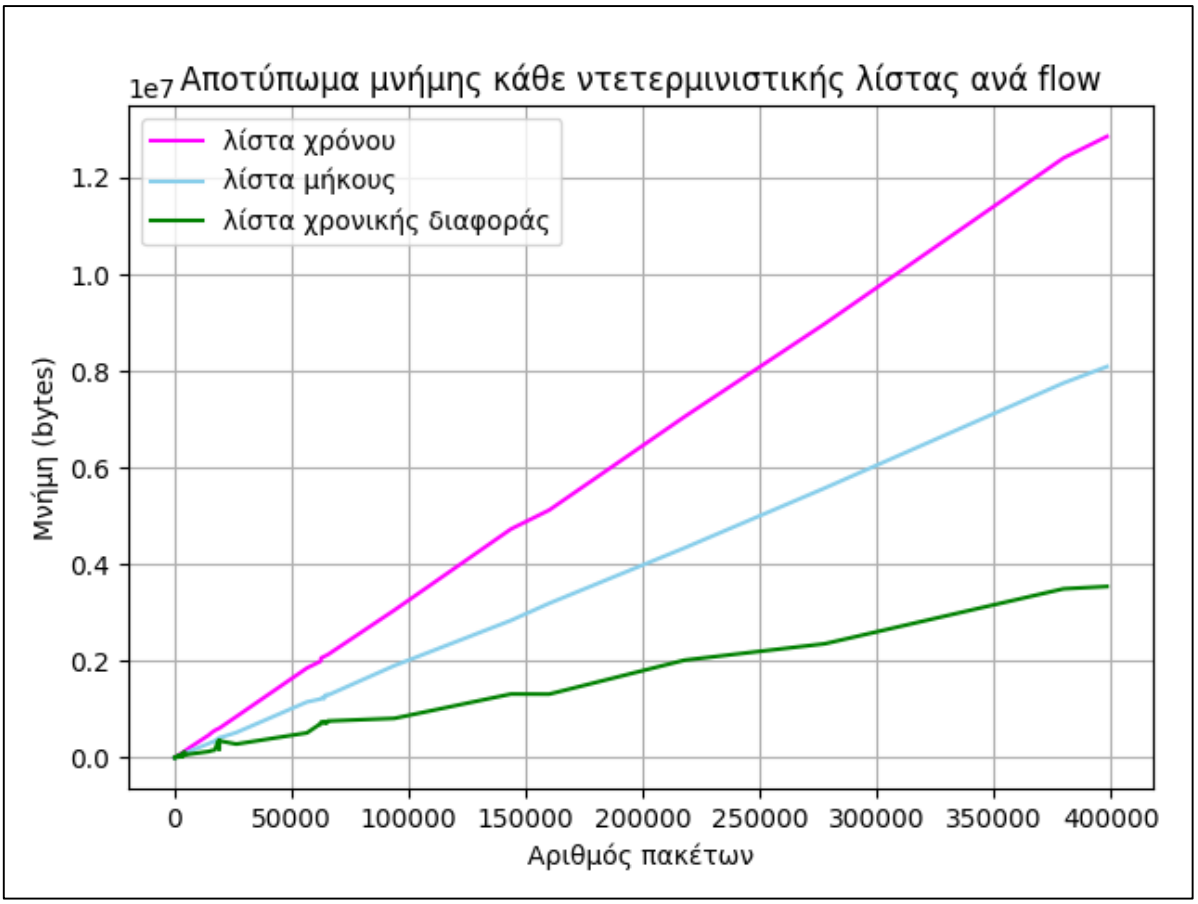


Σχήμα 14: Μνήμη ντετερμινιστικής υλοποίησης (ανά flow)

Στο από πάνω διάγραμμα αναπαρίσταται η μνήμη που χρειάζεται ο ντετερμινιστικός αλγόριθμος. Υπενθυμίζεται πως λείπουν οι μεταβλητές και οι δομές που είναι κοινές με τον πιθανοτικό αλγόριθμο, όπως επίσης και οι δύο ακέραιοι που εκτελούν την καταμέτρηση bytes. Το σημείο (x,y) στο σχήμα δηλώνει ότι χρειάζονται y bytes για ένα flow x πακέτων. Σε αντίθεση με την πιθανοτική υλοποίηση, η μνήμη είναι άμεση συνάρτηση του μεγέθους του flow.

Φαίνεται πως παρά τις όποιες δυναμικές τεχνικές χρησιμοποιεί η Python για memory allocation, η μνήμη αυξάνεται ανάλογα με τα πακέτα. Συγκεκριμένα, όπως δείχνει η διακεκομμένη ευθεία, απαιτούνται 62 bytes ανά πακέτο. Η αναλογική αύξηση είναι ένα αναμενόμενο και εύλογο συμπέρασμα, καθώς ο ντετερμινιστικός αλγόριθμος αποθηκεύει πληροφορία σχετικά με κάθε πακέτο, χωρίς συμπιέσεις και παραλείψεις.

Όμως, αυτή η παρατήρηση δεν είναι απόλυτα ακριβής. Η Python εισάγει ένα overhead στις λίστες, που αφορούν στην απεικόνιση της λίστας ως αντικείμενο στους εσωτερικούς μηχανισμούς της γλώσσας. Το overhead είναι εν γένει μία σταθερή ποσότητα, που προστίθεται στην σχέση αναλογίας. Για μεγάλους αριθμούς πακέτων το overhead είναι πρακτικά αμελητέο, οπότε η αναλογία φαίνεται να ισχύει. Όμως στα μικρότερα flows (που αποτελούν και πλειοψηφία) το overhead είναι μη αμελητέο, με αποτέλεσμα να χρειάζονται μέχρι και 90 bytes ανά πακέτο για την πλήρη αναπαράσταση των λιστών.

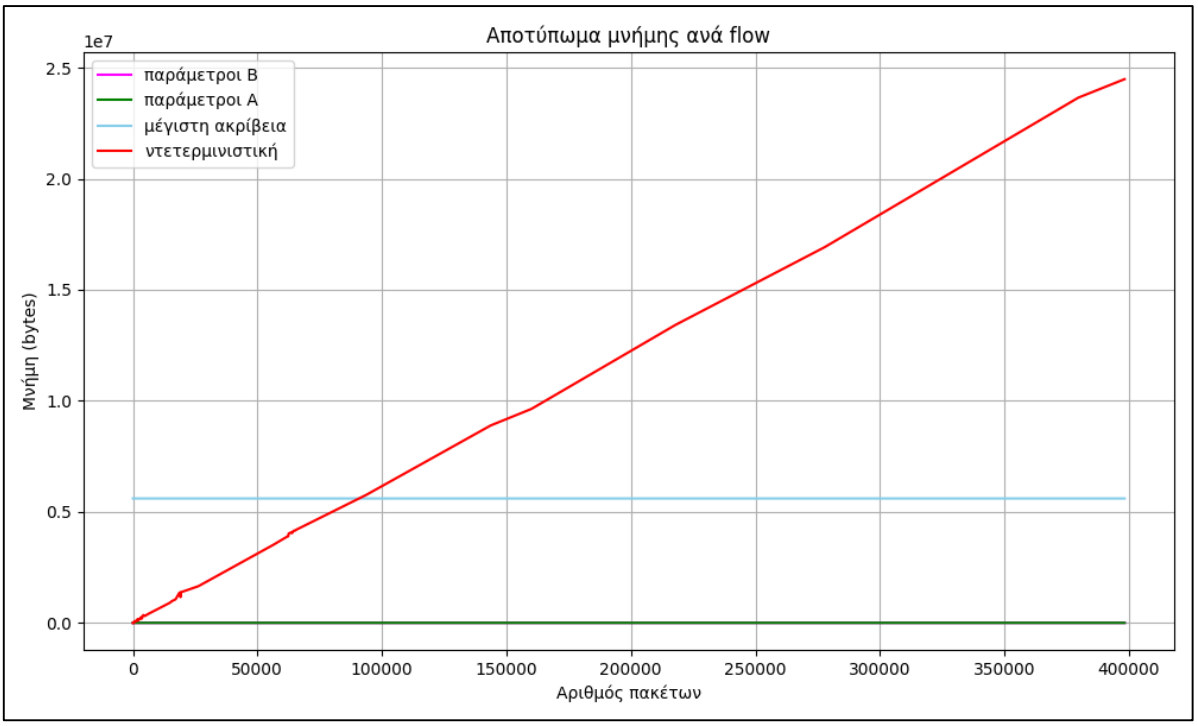


Σχήμα 15: Μνήμη που χρειάζεται κάθε λίστα (ανά flow)

Εδώ φαίνεται ο χώρος που απαιτεί κάθε μία από τις τρεις λίστες. Η λίστα με τα μήκη των πακέτων καταλαμβάνει τον λιγότερο χώρο, παρά το γεγονός ότι ένας ακέραιος είναι (συνήθως) μεγαλύτερος από έναν δεκαδικό. Οι ακέραιοι θεωρούνται ως αντικείμενα, και δεδομένου ότι τα μήκη πακέτων δεν παίρνουν απεριόριστες τιμές, αλλά κάποιες επαναλαμβάνονται, η Python εκτελεί κάποιες βελτιστοποιήσεις ως προς το τι αποθηκεύει. Οι άλλες δύο λίστες όμως αποτελούνται από δεκαδικούς, που πρακτικά είναι όλοι τους μοναδικοί. Η λίστα του χρόνου χρειάζεται περισσότερη μνήμη από τη λίστα των διαφορών. Αυτό είναι αναμενόμενο, αφού η λίστα χρόνων έχει ένα στοιχείο για κάθε πακέτο του flow (εκτός από το πρώτο) ενώ η λίστα διαφορών δέχεται κάποια τιμή μόνο όταν υπάρχει εναλλαγή αιτήματος και απόκρισης.

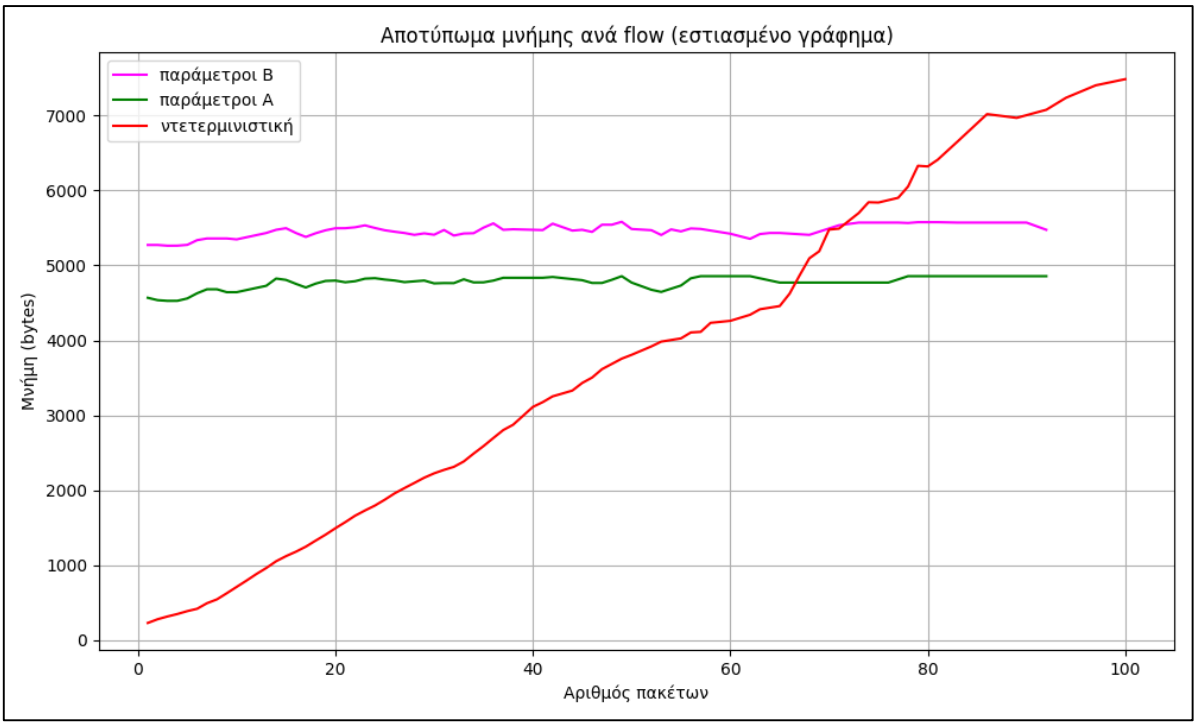
4.3.3 Σύγκριση

Τώρα που εξηγήθηκαν οι υπολογισμοί της μνήμης σε κάθε περίπτωση, αξίζει η παρουσίαση όλων των περιπτώσεων σε κοινό διάγραμμα ώστε να γίνουν πλήρως αντιληπτές οι διαφορές.



Σχήμα 16: Μνήμη που καταλαμβάνει κάθε υλοποίηση (ανά flow)

Από το κοινό διάγραμμα γίνεται εύκολα αντιληπτή η συντριπτική υπεροχή των πιθανοτικών δομών ως προς τη μνήμη. Όσο το μέγεθος των δεδομένων αυξάνεται, οι πιθανοτικές δομές καταφέρνουν να πετυχαίνουν πολύ καλές προσεγγίσεις των δεδομένων που ζητούνται, διατηρώντας το μέγεθος τους σταθερό. Αυτός είναι ο λόγος που είναι ιδιαίτερα δελεαστικές σε εφαρμογές big data [30]. Ακόμα και η περίπτωση που η Top-K χρησιμοποιείται με μέγιστη ακρίβεια χρησιμοποιεί λίγη μνήμη σε σχέση με την ντετερμινιστική υλοποίηση, για αρκούντως μεγάλο αριθμό πακέτων. Επειδή όμως χρήζει διερεύνησης το σημείο που οι πιο αποδοτικές υλοποιήσεις ξεκινάνε να χρειάζονται λιγότερη μνήμη από την ντετερμινιστική, παρατίθεται το ίδιο διάγραμμα εστιασμένο στην αρχή των αξόνων.

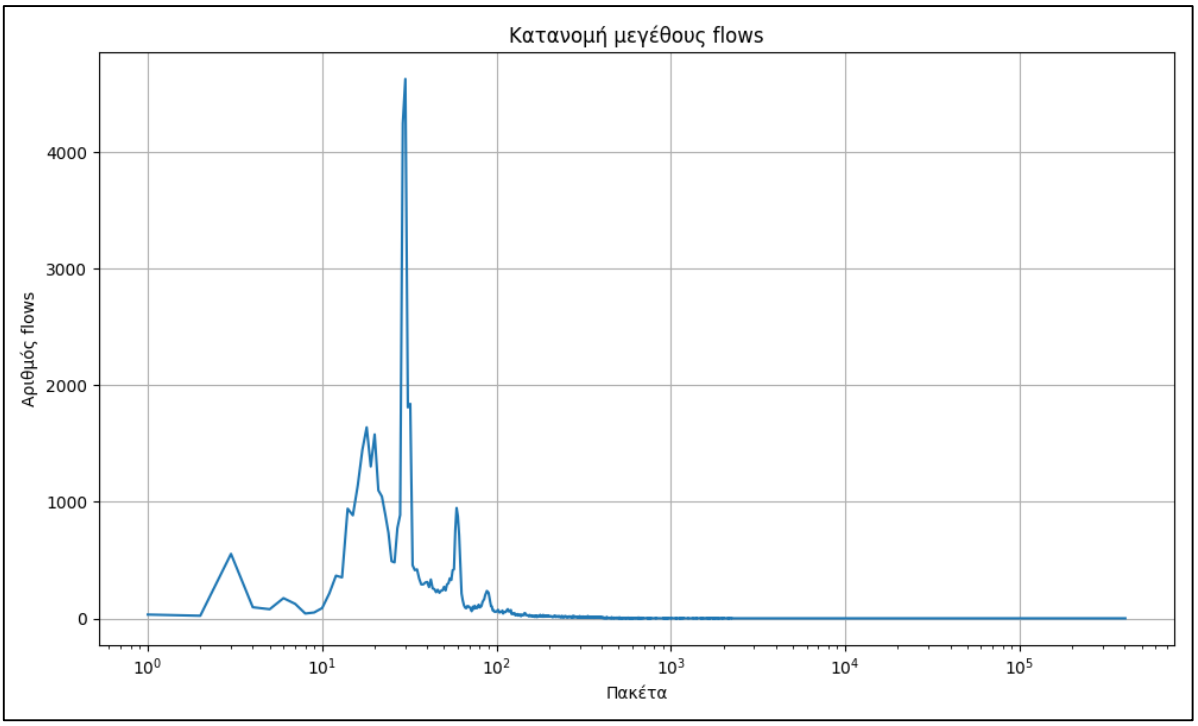


Σχήμα 17: Μνήμη κάθε υλοποίησης (ανά flow) σημεία τομής

Το διάγραμμα πέρασε από διαδικασία εξομάλυνσης ώστε να περιοριστούν οι απότομες διακυμάνσεις. Απουσιάζει η υλοποίηση με τη μέγιστη ακρίβεια της Top-K, καθώς το σημείο τομής της με την ντετερμινιστική είναι πολύ μακριά από τις άλλες δύο. Φαίνεται οπτικά πως για έναν αριθμό πακέτων και έπειτα συμφέρει η πιθανοτική υλοποίηση από πλευρά μνήμης. Συγκεκριμένα, έπειτα από ανάλυση των δεδομένων από τα οποία παρήχθη το διάγραμμα, διαπιστώθηκε ότι από 60 πακέτα και μετά οι παράμετροι A πετυχαίνουν μικρότερη μνήμη από την ντετερμινιστική υλοποίηση. Το αντίστοιχο σημείο για τις παραμέτρους B είναι 70 πακέτα. Και οι δύο αριθμοί αποτελούν προσέγγιση, καθώς σε τόσο χαμηλούς αριθμούς οι διακυμάνσεις (ακόμα και στην ντετερμινιστική υλοποίηση) είναι αρκετά έντονες. Περισσότερο ενδιαφέρουσα όμως είναι η αντίστροφη δήλωση: για flows με πολύ λίγα πακέτα, συμφέρει από άποψη μνήμης, και προφανώς από άποψη ακρίβειας, η ντετερμινιστική υλοποίηση. Ανακύπτει λοιπόν το ζήτημα αν αξίζει συνολικά η πιθανοτική υλοποίηση. Σαφώς στα μεγάλα flows η διαφορά στη μνήμη είναι ουσιώδης, όμως η πλειοψηφία των flows στο dataset που αναλύεται αλλά και γενικότερα στο πρωτόκολλο DNS είναι μικρές. Προκειμένου να δοθεί απάντηση στο θέμα διεξάγεται η ακόλουθη ανάλυση.

4.3.4 Κατανομή μεγέθους flows

Όλα τα flows που αναλύθηκαν στα πειράματα καταγράφηκαν και συνοψίστηκαν ανάλογα με τον αριθμό πακέτων, ώστε να βρεθεί η κατανομή μεγεθών. Το παρακάτω διάγραμμα απεικονίζει την εν λόγω κατανομή.



Σχήμα 18: Κατανομή μεγέθους flows, άξονας x σε λογαριθμική κλίμακα

Ο άξονας x είναι σε λογαριθμική κλίμακα για καλύτερη απεικόνιση των χαμηλών αριθμών. Εποπτικά φαίνεται πως τα περισσότερα flows έχουν μικρό αριθμό πακέτων. Η επικρατούσα τιμή είναι 30 πακέτα, και η διάμεσος είναι πολύ κοντά, στα 31. Ωστόσο αυτά τα μεγέθη τείνουν να αγνοούν τις οριακές τιμές. Πράγματι, η μέση τιμή είναι 450 πακέτα ανά flow. Αυτό το χαρακτηριστικό δεν φαίνεται στο διάγραμμα, καθώς υπάρχουν σχετικά λίγα flows με πολλά πακέτα, τα οποία είναι διεσπαρμένα. Όμως υπάρχουν και κάνουν αισθητή την παρουσία τους, αυξάνοντας την μέση τιμή.

Για να διαπιστωθεί αν συμφέρει ή όχι η χρήση των πιθανοτικών δομών στα παρόντα πειράματα με βάση πάντα το δεδομένο dataset, θα έπρεπε κανονικά να αθροιστεί η μνήμη που χρειάστηκαν οι υλοποιήσεις για κάθε flow. Τα χαρακτηριστικά που έχουν εντοπιστεί όμως μπορούν να συνεισφέρουν στην ανάλυση, προσφέροντας έναν απλούστερο τρόπο μελέτης. Οι πιθανοτικές υλοποιήσεις φάνηκε πως χρειάζονται (πρακτικά) σταθερή μνήμη, ανεξάρτητα από τον αριθμό πακέτων. Άρα κάθε flow χρειάζεται συγκεκριμένα bytes, τα οποία εξαρτώνται μόνο από την παραμετροποίηση. Η μνήμη της ντετερμινιστικής υλοποίησης από την άλλη εξαρτάται άμεσα από τον αριθμό πακέτων σε κάθε flow, αλλά η εξάρτηση είναι σχεδόν γραμμική. Ειδικότερα, ακολουθεί την ευθεία $y = 62x$ που σημαίνει ότι κάθε flow χρειάζεται μνήμη 62 bytes ανά πακέτο. Αυτή η δήλωση είναι κοντά στην αλήθεια, δεν παύει όμως να είναι μία προσέγγιση. Μάλιστα για flows με λίγα πακέτα (που είναι και η πλειοψηφία) το overhead που επιφέρουν οι λίστες της Python αυξάνουν αυτόν τον αριθμό. Για λόγους απλότητας όμως θα θεωρηθεί πως η προσέγγιση ισχύει, και τα αποτελέσματα θα κριθούν εκ των υστέρων.

Για τους υπολογισμούς θα χρησιμοποιηθούν οι εξής απλές σχέσεις. Ένα flow στην ντετερμινιστική υλοποίηση χρειάζεται 62 bytes ανά πακέτο, και το μέσο flow έχει 450 πακέτα. Άρα είναι:

$$Μέση\ ντετερμινιστική\ μνήμη\ (ανά\ flow) = 62 * 450 \rightarrow$$

$$Μέση\ ντετερμινιστική\ μνήμη\ (ανά\ flow) = 27900\ bytes$$

Η πιθανοτική υλοποίηση εμφανίζει μικρές και σχετικά απρόβλεπτες μεταβολές του μεγέθους της. Χάριν απλότητας θεωρείται για κάθε κατηγορία παραμέτρων ότι απαιτείται η μέγιστη φορά τιμή, ανάλογα με τις παραμέτρους. Ο υπολογισμός σε κάθε περίπτωση είναι τρεις φορές το αμετάβλητο μέγεθος της T-Digest (χρειάζονται τρεις ανά flow) και το μέγιστο μέγεθος που λαμβάνει η Top-K (χρειάζεται μία ανά flow).

$$standard\ parameters\ (ανά\ flow) = 3 * 656 + 3016 \rightarrow$$

$$standard\ parameters\ (ανά\ flow) = 4984\ bytes$$

$$new\ standard\ parameters\ (ανά\ flow) = 3 * 648 + 3648 \rightarrow$$

$$new\ standard\ parameters\ (ανά\ flow) = 5592\ bytes$$

$$\max accuracy\ \ (ανά\ flow) = 3 * 808 + 5603176 \rightarrow$$

$$\max accuracy\ \ (ανά\ flow) = 5605600\ bytes$$

Με εξαίρεση την περίπτωση μέγιστης ακρίβειας, η οποία κάνει τεράστια κατάχρηση, η πιθανοτική υλοποίηση με τις υπόλοιπες παραμέτρους είναι σημαντικά πιο οικονομική από την ντετερμινιστική. Επομένως, το σχετικά μικρό κόστος που αφορά στην ακρίβεια των προβλέψεων για κακόβουλα flows αντισταθμίζεται από την σημαντική εξοικονόμηση μνήμης. Μάλιστα όσο περισσότερα (και μεγαλύτερα) flows καλείται να επεξεργαστεί ένα σύστημα, τόσο περισσότερο αυξάνεται το κέρδος.

4.3.5 Δομές για bytes που εστάλησαν ή ελήφθησαν

Τώρα που ολοκληρώθηκε η ανάλυση των δομών που αφορούν στις κατηγορίες μηκών πακέτων, χρόνων και χρονικών διαφορών, πρέπει να γίνει συζήτηση για την κατηγορία που έμεινε στο περιθώριο, αυτή που αφορά στα bytes από και προς τον server. Έχει ιδιαίτερο ενδιαφέρον γιατί πλέον η σύγκριση της απόδοσης σχετικά με τη μνήμη εξαρτάται από τον αριθμό των flows, όχι από τον αριθμό πακέτων που περιέχονται σε ένα flow. Αυτό συμβαίνει γιατί στην πιθανοτική υλοποίηση αξιοποιούνται συνολικά, για όλα τα flows, δύο CMS, όπως έχει εξηγηθεί. Αυτή η δομή, όμοια με άλλες πιθανοτικές δομές, διατηρεί σταθερό μέγεθος ανεξάρτητα από τα δεδομένα που δέχεται. Άρα για κάθε πείραμα, άσχετα με το αν περιλαμβάνει πολλά ή λίγα flows με μεγάλο ή μικρό πλήθος πακέτων, η μνήμη που απαιτείται για αυτή την κατηγορία είναι σταθερή.

Αντίθετα, στην ντετερμινιστική περίπτωση πρόκειται απλώς για δύο αριθμούς ανά flow. Ο αριθμός των πακέτων είναι αδιάφορος. Αν ένα flow έχει πολλά πακέτα, προφανώς τα bytes των requests και των responses θα είναι εξίσου πολλά, όμως αυτό δεν επηρεάζει το μέγεθος των δύο ακεραίων: παραμένουν 28 bytes έκαστος. Άρα η μνήμη που απαιτείται είναι 2*28=56 bytes ανά flow.

Οπότε η σύγκριση εδώ έχει να κάνει με τον αριθμό flows. Τα CMS που χρησιμοποιήθηκαν στην εργασία είχαν μήκος 3000 και 10 hash functions, και χρειάζονται 124352 bytes έκαστο για την αναπαράσταση τους. Οι παράμετροι επιλέχθηκαν μετά από πειραματική διαδικασία ώστε η δομές να επιστρέφουν αξιόπιστες εκτιμήσεις με μεγάλη πιθανότητα, ακόμα και σε περιπτώσεις με εκατοντάδες χιλιάδες flows. Ένας ακέραιος στην Python χρειάζεται 28 bytes. Έστω n ο αριθμός flows για τον οποίο οι δύο υλοποιήσεις απαιτούν ίση μνήμη:

$$124352 * 2 = n * 28 * 2 \rightarrow$$

$$124352 = 28n \rightarrow$$

$$n \approx 4441$$

Άρα αν σε ένα πείραμα υπάρχουν πάνω από 4441 ενεργά flows, η πιθανοτική υλοποίηση με τα CMS πιο αποδοτική, και αντίστροφα. Στο παρόν dataset κάποια πειράματα είχαν περισσότερα, και κάποια λιγότερα από 4441 flows. Σχετικά με την ακρίβεια, όπως φάνηκε στα σχετικά διαγράμματα η πιθανοτική υλοποίηση αλλά με ντετερμινιστικό υπολογισμό των bytes είχε ελάχιστα καλύτερη απόδοση από την πλήρως πιθανοτική. Με άλλα λόγια τα CMS ανταπεξήλθαν με επιτυχία στην αποστολή τους. Επίσης αξίζει να σημειωθεί πως όσο περισσότερα flows υπάρχουν εξοικονομείται μεν μνήμη, όμως αυξάνεται και η ανακρίβεια της δομής. Το μόνο σίγουρο είναι ότι σε περιπτώσεις που πρέπει να αναλυθούν λίγα flows, αξίζει η ντετερμινιστική υλοποίηση. Αυτή η παρατήρηση, σε συνδυασμό με την απλότητα και την αξιοπιστία που προσφέρει η χρήση δύο ακεραίων, αντί για μία πιθανοτική δομή, ίσως κάνουν την πλάστιγγα να γείρει υπέρ της ντετερμινιστικής υλοποίησης. Πάντως, η σύγκριση δεν ανέδειξε ξεκάθαρο και αδιαμφισβήτητο νικητή για κάθε περίπτωση.

4.3.6 Συμπεράσματα

Με βάση τα προηγούμενα, προκύπτει αβίαστα το συμπέρασμα πως οι πιθανοτικές δομές επιτυγχάνουν τον σκοπό τους, που είναι η εξοικονόμηση μνήμης. Η σταθερότητα που προσφέρουν, ανεξάρτητα από τον όγκο δεδομένων που καλούνται να διαχειριστούν, είναι το χαρακτηριστικό που τις καθιστά τόσο χρήσιμες. Αντίθετα, η αναλογική αύξηση της μνήμης στην ντετερμινιστική υλοποίηση οδηγεί σε μεγάλη σπατάλη μνήμης για ένα μέσο flow, ενώ για τα μεγαλύτερα flows το πρόβλημα δυσχεραίνεται ακόμα περισσότερο. Στα πολύ μικρά flows όμως η κατάσταση αντιστρέφεται, και η ντετερμινιστική υλοποίηση έχει καλύτερη επίδοση.

Αναλυτικότερα, οι τρεις δομές T-Digest είχαν απόλυτα σταθερό μέγεθος, που εξαρτάται μόνο από τις παραμέτρους αρχικοποίησης. Καθ' όλη την διάρκεια εκτέλεσης οποιουδήποτε πειράματος, η μνήμη τους διατηρούνταν αμετάβλητη, όσοι αριθμοί κι αν εισάγονταν. Μάλιστα είχαν και πολύ μικρό αποτύπωμα, καταλαμβάνοντας μόνο μερικές εκατοντάδες bytes η κάθε μία.

Από την άλλη, η δομή Top-K εμφάνισε μικρές διακυμάνσεις στο μέγεθος της, ακόμα και για ίδιες παραμέτρους. Η αιτία είναι η τυχαιότητα που υπάρχει στην εσωτερική της αναπαράσταση. Εντούτοις, οι διακυμάνσεις δεν επηρέασαν σημαντικά το αποτέλεσμα, καθώς ακόμα και με τις χειρότερες επιδόσεις της η πιθανοτική μνήμη παραμένει σε ικανοποιητικά μικρά επίπεδα. Επίσης, η δομή έχει σημαντικά μεγαλύτερο αποτύπωμα από την T-Digest, στις τάξεις των χιλιάδων bytes.

Σχετικά με την κατηγορία καταμέτρησης bytes, εξακολουθεί να ισχύει η ιδέα της σταθερότητας της πιθανοτικής υλοποίησης έναντι της αναλογικής ντετερμινιστικής, με μία σημαντική διαφορά: το σημείο τομής της απόδοσης των υλοποιήσεων δεν σχετίζεται πλέον με τον αριθμό πακέτων, αλλά με τον αριθμό flows. Συγκεκριμένα οι δομές CMS καταλαμβάνουν σταθερή μνήμη ανεξάρτητα από τα flows, ενώ οι δύο ντετερμινιστικοί ακέραιοι δεν επηρεάζονται από το μέγεθος των flows, μόνο από τον αριθμό τους. Άρα όσο περισσότερα flows καλείται να διαχειριστεί ένα σύστημα, τόσο αυξάνεται το κέρδος των CMS. Η διαφορά όμως είναι πολύ μικροί, καθώς δύο ακέραιοι ανά flow επιβάλλουν μεν περισσότερη χρήση μνήμης, είναι δε πολύ μικρή η επιβάρυνση.

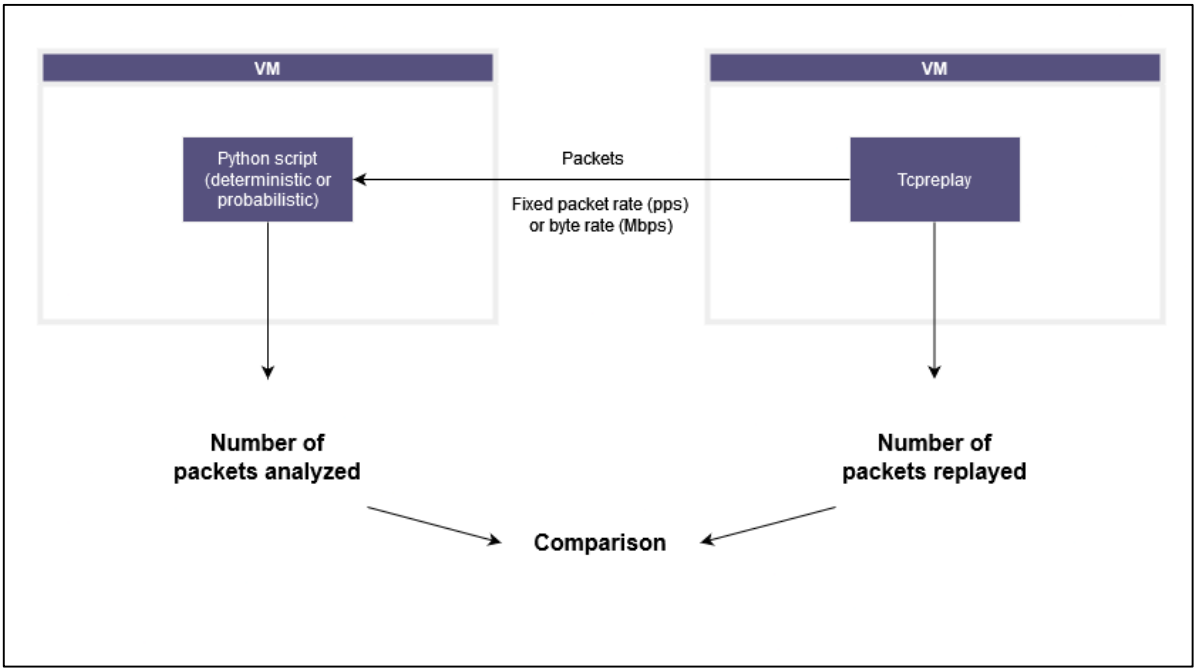
Συνεπώς, οι T-Digest και η Top-K καταφέρνουν σημαντική μείωση στις ανάγκες μνήμης του συστήματος. Οι δομές CMS, επίσης πετυχαίνουν μείωση στην γενική περίπτωση, όμως όχι τόσο ουσιώδη, και η επιλογή τους ή η χρήση των ντετερμινιστικών ακεραίων δεν επηρεάζει σημαντικά την συνολική κατάσταση. Πάντως οι T-Digest και η Top-K από μόνες τους αρκούν για να αποδείξουν το κέρδος της πιθανοτικής υλοποίησης.

4.4 Benchmark

Εκτός από την μνήμη όμως, υπάρχει άλλος ένας σημαντικός υπολογιστικός πόρος που χρήζει μελέτης απόδοσης. Πρόκειται φυσικά για τον επεξεργαστή, ή αλλιώς για την CPU. Η σύγκριση της πιθανοτικής με την ντετερμινιστική υλοποίηση σε αυτόν τον τομέα γίνεται με την εξέταση απόδοσης για δεδομένη επεξεργαστική ισχύ, σε πραγματική ροή πακέτων. Όλη η προηγούμενη ανάλυση έγινε με τον δεύτερο τρόπο λειτουργίας, δηλαδή με ανάγνωση των απαιτούμενων δεδομένων των πακέτων από ένα αρχείο CSV, και έπειτα γινόταν η επεξεργασία τους. Έτσι υπήρχε εγγύηση ότι όλα τα πακέτα θα εξετάζονταν χωρίς απώλειες, ενώ ταυτόχρονα γινόταν ορθή προσομοίωση μίας πραγματικής ροής. Όμως, σε ένα πραγματικό σύστημα υπάρχει η απαίτηση η επεξεργασία των πακέτων να γίνεται πιο γρήγορα από όσο συρρέουν τα πακέτα. Αν δεν ισχύει αυτό, αναπόφευκτα κάποια πακέτα θα χαθούν, εκτός αν υπάρχουν ουρές που αποθηκεύουν προσωρινά έναν αριθμό πακέτων. Αυτός ο μηχανισμός είναι καλός για να ελαττώσει τις απώλειες που μπορεί να προκύψουν από στιγμές εκρηκτικής κίνησης (spikes) όμως είναι ανεπαρκής εάν στη γενική περίπτωση ο ρυθμός άφιξης πακέτων είναι μεγαλύτερος από τον ρυθμό επεξεργασίας τους.

Αναφορικά με την εργασία, απώλεια πακέτων συνεπάγεται ελάττωση ακρίβειας των στατιστικών ιδιοτήτων του εκάστοτε flow, που οδηγεί τα μοντέλα ML σε λανθασμένες προβλέψεις. Μία προφανής λύση στο πρόβλημα είναι η αύξηση της επεξεργαστικής ισχύος ώστε να εξαλειφθεί πλήρως η απώλεια πακέτων ακόμα και σε υψηλούς ρυθμούς πακέτων (packet rates). Αυτή η απλή ιδέα θα αποτελούσε πανάκεια, ωστόσο οι υπολογιστικοί πόροι κοστίζουν, και σκοπός είναι η οικονομική χρήση τους. Συνεπώς χρειάζεται μελέτη των υλοποιήσεων ως προς την απώλεια πακέτων, για δεδομένη επεξεργαστική ισχύ.

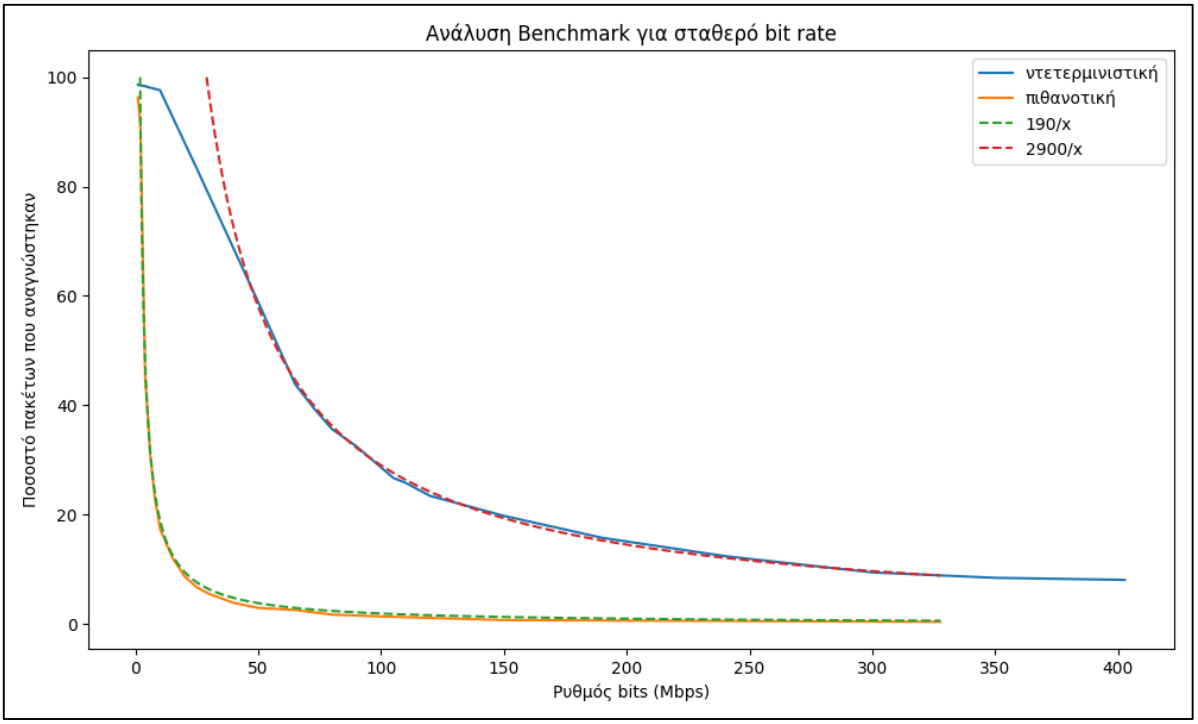
Τα πειράματα που έγιναν έχουν την εξής δομή. Από ένα VM αναπαράγονται καταγραφές από το dataset, και κάθε φορά προκαθορίζεται είτε ο ρυθμός πακέτων είτε ο ρυθμός bytes, δηλαδή οι καταγραφές δεν αναπαράγονται αυτούσιες. Σκοπός του πειράματος δεν είναι η μέτρηση της ακρίβειας, αλλά της ευρωστίας (robustness) του συστήματος. Όλες οι λειτουργίες του προγράμματος (δημιουργία δομών, εισαγωγή εξαγωγή features, προβλέψεις) εκτελούνται κανονικά, όμως τα αποτελέσματα της ταξινόμησης είναι πρακτικά τυχαία και συνεπώς ανούσια. Το φιξάρισμα bit rate και packet rate γίνεται για να εξεταστεί η συμπεριφορά των προγραμμάτων συναρτήσει και των δύο μεταβλητών. Στην παρακάτω εικόνα φαίνεται η γενική λειτουργία του πειράματος.



Σχήμα 19: Τρόπος λειτουργίας πειράματος benchmark

4.4.1 Σύγκριση

Το πείραμα επαναλήφθηκε πολλές φορές, με διαφορετικά bit rates. Σημειώνεται πως για την πιθανοτική υλοποίηση χρησιμοποιήθηκαν με παραμέτρους B. Παρατίθεται το διάγραμμα που απεικονίζει την απόδοση των δύο υλοποιήσεων καθώς αυξάνεται ο ρυθμός bytes.

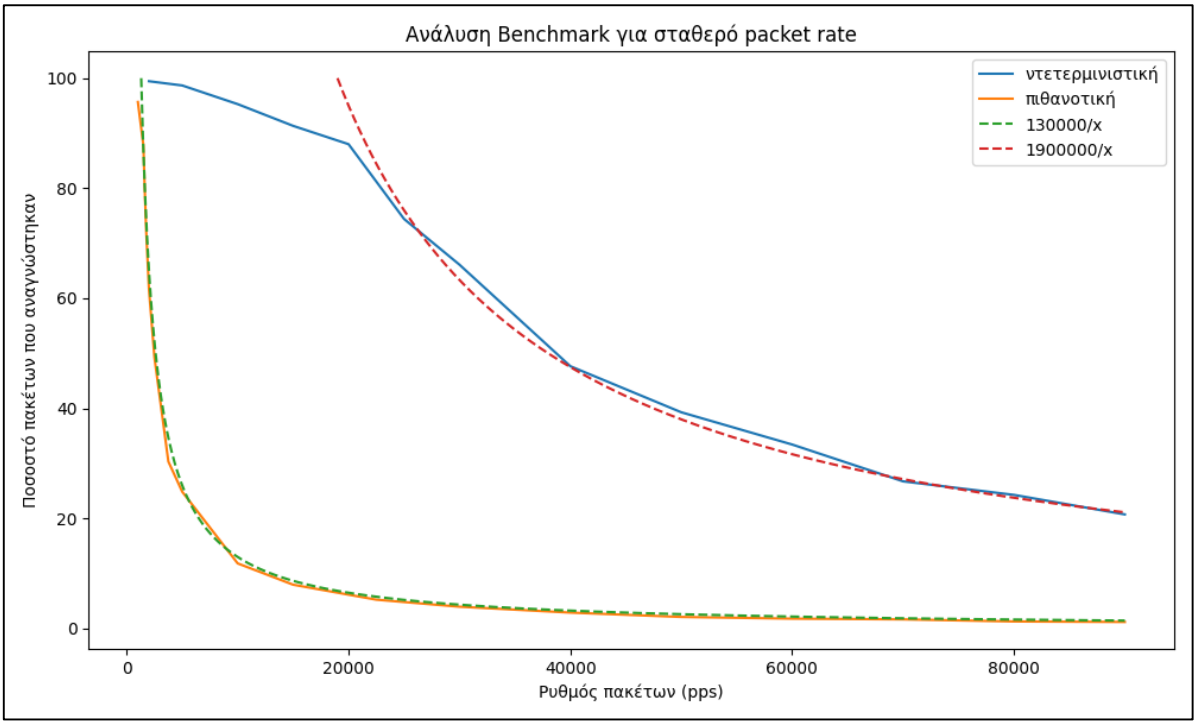


Σχήμα 20: Ανάλυση benchmark για σταθερό bit rate

Ο άξονας x δείχνει τον ρυθμό bytes, και ο άξονας y τον αριθμό πακέτων που έγιναν δεκτά από το πρόγραμμα, ως ποσοστό των συνολικών πακέτων που αναπαράχθηκαν. Γίνεται εύκολα αντιληπτό ότι η ντετερμινιστική υλοποίηση είναι πολύ καλύτερη. Φαίνεται στο διάγραμμα πως για πολύ μικρούς ρυθμούς διατηρεί την επίδοση της πρακτικά στο 100%. Ισχύει το ίδιο και για την πιθανοτική περίπτωση, για τόσο μικρούς ρυθμούς που δεν φαίνονται στο διάγραμμα. Όμως όταν ο ρυθμός άφιξης πακέτων ξεπερνά τον ρυθμό επεξεργασίας, η απόδοση μειώνεται σταδιακά καθώς το πρόγραμμα δεν προλαβαίνει να επεξεργαστεί κάποια πακέτα, και αφήνει αναγκαστικά κάποια να χαθούν. Από αυτό το σημείο οι υλοποιήσεις ακολουθούν συναρτήσεις της μορφής α/x όπως φαίνεται από τις βοηθητικές αναπαραστάσεις. Τα διαγράμματα δημιουργήθηκαν φυσικά με βάση τα αποτελέσματα των πειραμάτων, όχι από αυθαίρετους μαθηματικούς τύπους. Μολαταύτα, η σύγκλιση με τις συναρτήσεις είναι ένα λογικό και αναμενόμενο αποτέλεσμα, καθώς η απόδοση του συστήματος είναι αντιστρόφως ανάλογη του ρυθμού άφιξης.

Οι μαθηματικοί τύποι απλοποιούν την περεταίρω σύγκριση, καθώς εξ' ορισμού ο αριθμητής των συναρτήσεων δηλώνει τον ρυθμό επεξεργασίας του συστήματος και ο παρονομαστής τον ρυθμό άφιξης. Επειδή η απόδοση μετριέται σε ποσοστό και όχι σε απόλυτα μεγέθη, πρέπει ο αριθμητής να διαιρεθεί με 100. Συνεπώς προκύπτει άμεσα ότι η ντετερμινιστική υλοποίηση επεξεργάζεται 29 megabits ανά δευτερόλεπτο, ενώ η πιθανοτική μόλις 1.9. Δηλαδή η ντετερμινιστική υλοποίηση είναι 15 φορές ταχύτερη. Για την ακρίβεια, έχει 15 φορές μεγαλύτερο ρυθμό επεξεργασίας.

Ακολουθεί αντίστοιχο διάγραμμα, με σταθερό ρυθμό πακέτων, όχι bytes.



Σχήμα 21: Ανάλυση benchmark για σταθερό packet rate

Το διάγραμμα είναι σχηματικά παρόμοιο με το προηγούμενο, όπως αναμένεται, αφού οι δύο ρυθμοί είναι πρακτικά ανάλογοι. Θεωρητικά υπάρχουν μικρές διακυμάνσεις λόγω της ποικιλίας στα μεγέθη πακέτων, όμως είναι σχετικά ασήμαντες αφού τα περισσότερα πακέτα έχουν σχετικά κοντινά μήκη μεταξύ τους, με λίγες εξαιρέσεις, όπως έχει συζητηθεί. Όμοια

με την προηγούμενη ανάλυση, μπορεί να εξαχθεί ο ρυθμός επεξεργασίας πακέτων των προγραμμάτων. Η πιθανοτική υλοποίηση επεξεργάζεται 1300 πακέτα ανά δευτερόλεπτο, ενώ η ντετερμινιστική 19000. Ο λόγος των δύο αριθμών συνάδει με την προηγούμενη διαπίστωση, καθώς επαληθεύεται η δήλωση ότι η ντετερμινιστική έχει 15 φορές μεγαλύτερο ρυθμό επεξεργασίας.

Τα αποτελέσματα που προέκυψαν αφορούν τις προγραμματιστικές υλοποιήσεις της εργασίας, και δεν αναπαριστούν πλήρως τη σχέση των πιθανοτικών και ντετερμινιστικών δομών. Η Python είναι γνωστή για την ευκολία χρήσης της και την υψηλού επιπέδου υποστήριξη απλών δομών δεδομένων, όπως οι λίστες, οι οποίες είναι βελτιστοποιημένες σε χαμηλό επίπεδο για ταχύτητα και αποδοτικότητα, εκμεταλλευόμενες τη στενή ενσωμάτωσή τους με τον διερμηνέα της Python. Αντίθετα, οι πιθανοτικές δομές είναι ορισμένες από τον χρήστη (user defined) και συνδυάζουν διάφορες απλές δομές της γλώσσας για να πετύχουν τον σκοπό τους. Συνεπώς χάνουν τα πλεονεκτήματα των δομών χαμηλού επιπέδου, και είναι πολύ πιο αργές από τις θεωρητικές προδιαγραφές τους.

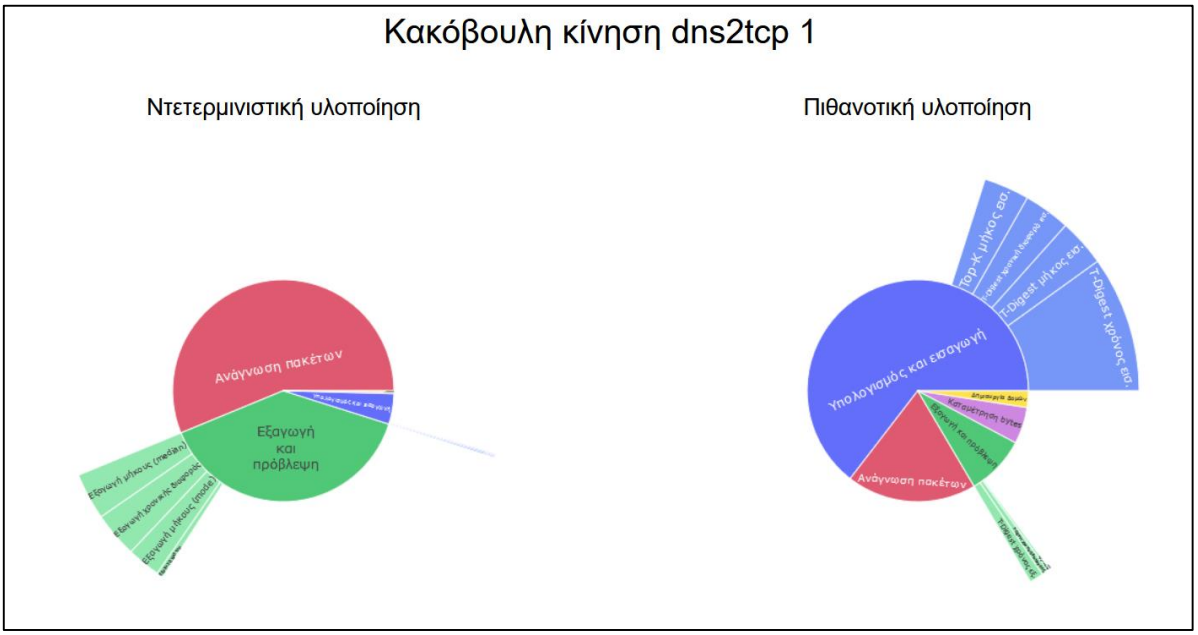
4.4.2 Ανάλυση μοιράσματος χρόνου

Παρά την τελευταία διαπίστωση, αξίζει να εξεταστεί σε βάθος η κατανομή του απαιτούμενου χρόνου για τις διάφορες λειτουργίες του προγράμματος, ώστε να διευκρινιστεί η απόδοση σε καθεμία από αυτές. Πιο συγκεκριμένα, μετρήθηκαν τα ποσοστά χρόνου επί της συνολικής διάρκειας του κάθε πειράματος που χρειάστηκε για την δημιουργία των διαφόρων δομών, την εισαγωγή και εξαγωγή δεδομένων από αυτές καθώς επίσης και τον υπολογισμό τους. Επισημαίνεται πως οι μετρήσεις έγιναν με τα προγράμματα που διαβάζουν τις πληροφορίες των πακέτων από αρχεία CSV (δεύτερος τρόπος λειτουργίας) και όχι με σύγχρονη αναπαραγωγή πακέτων (πρώτος τρόπος). Η απόφαση αυτή δεν επηρεάζει τους χρόνους που έχουν ενδιαφέρον στην ανάλυση, μειώνει όμως τον χρόνο που χρειάζεται για την ανάγνωση των πακέτων, καθιστώντας την εκτέλεση των πειραμάτων σημαντικά ταχύτερη. Παρακάτω εξηγείται συνοπτικά ο διαχωρισμός του προγράμματος σε διακριτά τμήματα.

- Main: Το βασικό μέρος. Εκτελεί σειριακή ανάγνωση των πακέτων από το csv, εξάγει τα απαραίτητα δεδομένα και τα προωθεί στις επόμενες φάσεις επεξεργασίας.
- General stuff: Προκαταρκτικό στάδιο για περεταίρω επεξεργασία. Η βασική (και πιο χρονοβόρα) λειτουργία του είναι η δημιουργία των διαφόρων δομών που απαιτούνται για κάθε καινούριο flow.
- Byte count: Αφορά στην καταμέτρηση των bytes που εστάλησαν ή ελήφθησαν σε κάθε flow.
- Statistics: Εδώ όλοι οι υπόλοιποι ενδιαφέροντες αριθμοί (μήκος πακέτου, χρόνος, χρονική διαφορά αιτήματος απόκρισης) εισάγονται στις κατάλληλες δομές και υπολογίζονται επίσης οι απλούστερες στατιστικές ιδιότητες όπως η διακύμανση. Επειδή αυτή η κατηγορία από μόνη της είναι πολύ γενική και αόριστη, αναλύεται σε περισσότερα επί μέρους στάδια, ώστε να μπορέσει να γίνει ορθότερα η σύγκριση των διαφορετικών υλοποιήσεων.
    - Η πρώτη υποκατηγορία αφορά στην εισαγωγή των αριθμών στις δομές. Συγκεκριμένα, μετριέται ο χρόνος που απαιτεί η εντολή εισαγωγής. Για παράδειγμα, μετριέται ο χρόνος που χρειάστηκε για να ενημερωθεί η δομή T-Digest με ένα νέο μήκος πακέτου, και αντίστοιχα στην ντετερμινιστική υλοποίηση η εισαγωγή του μήκους στη λίστα. Οι όμοιοι χρόνοι αθροίζονται στο τέλος του προγράμματος ώστε να υπολογιστεί το ποσοστό επί της συνολικής λειτουργίας. Στην πιθανοτική υλοποίηση οι εντολές που

χρονομετρούνται είναι οι εισαγωγές μήκους πακέτου, χρόνου και χρονικής διαφοράς στις αντίστοιχες T-Digest, και η εισαγωγή του μήκους πακέτου στην Top-K. Στην ντετερμινιστική μετριούνται οι εντολές εισαγωγής των τριών αριθμών στις λίστες τους.
    - Οι υπόλοιπες λειτουργίες του σταδίου είναι πανομοιότυπες στις δύο υλοποιήσεις, άρα δεν προκύπτει λόγος να γίνει περισσότερο ενδελεχής κατηγοριοποίηση.
- Classify: Το στάδιο αυτό εκτελεί την εξαγωγή των features που θα χρησιμοποιηθούν ως είσοδος στο μοντέλο μηχανικής μάθησης. Στο ίδιο στάδιο γίνεται επιπλέον η πρόβλεψη. Όμοια με πριν, η κατηγορία επιδέχεται επιπλέον διαχωρισμού.
    - Δυϊκώς αντίστροφα με το προηγούμενο στάδιο, εδώ χρονομετρούνται οι εντολές *εξαγωγής* των features από τις δομές τους, αντί της *εισαγωγής*. Στην πιθανοτική περίπτωση πρόκειται για τις εξαγωγές από τις τρεις T-Digest και την μία Top-K, σε πλήρη αντιστοιχία με τις εισαγωγές. Στην ντετερμινιστική όμως ενώ οι εισαγωγές ήταν τρεις, προστίθεται μία τέταρτη εξαγωγή, καθώς από τη λίστα με τα μήκη πακέτων πρέπει να εξαχθεί όχι μόνο η διάμεσος, αλλά και η επικρατούσα τιμή.
    - Οι υπόλοιπες λειτουργίες όπως η πρόβλεψη είναι ίδιες στις δύο υλοποιήσεις άρα δεν χρειάζονται περισσότερη ανάλυση.

Έχοντας σαφώς ορισμένα τα επιμέρους τμήματα του προγράμματος προς χρονομέτρηση, και γνωρίζοντας από την προηγούμενη ανάλυση τις σημαντικές διαφορές στους ρυθμούς επεξεργασίας των δύο υλοποιήσεων, καθίσταται εφικτό το επόμενο βήμα, που είναι η αναγνώριση των χρονοβόρων λειτουργιών. Προς τον σκοπό αυτό παρατίθενται ορισμένα χαρακτηριστικά διαγράμματα, όπου φαίνεται ποιοτικά η διαφορά ανάμεσα σε κάθε στάδιο των υλοποιήσεων. Τα διαγράμματα αυτά δημιουργήθηκαν με την χρήση plotly [31], σε Python.

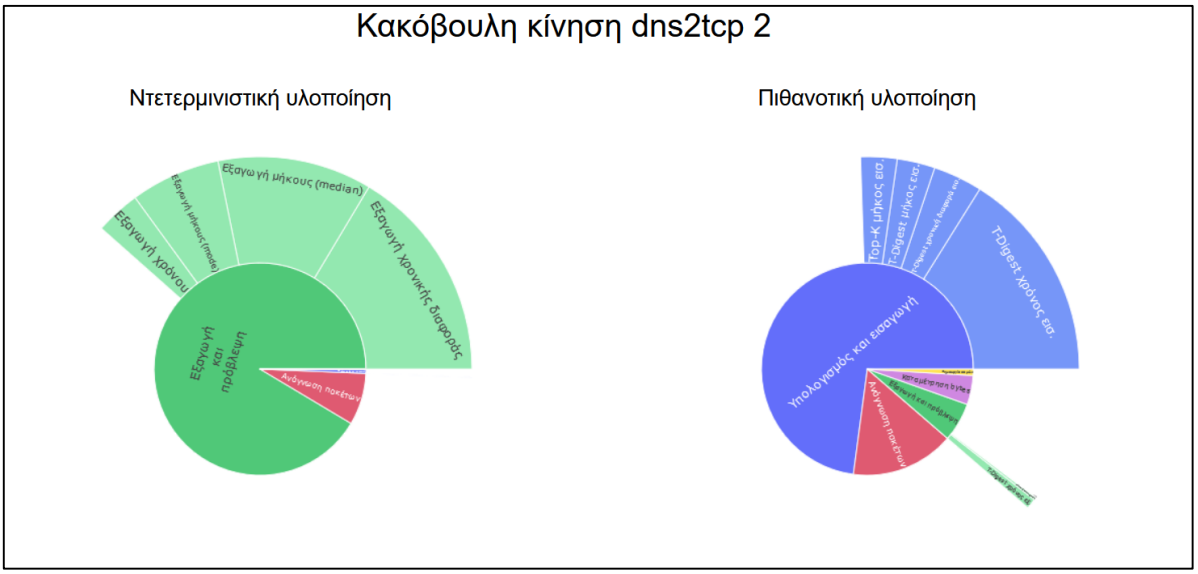


Σχήμα 22: Σύγκριση κατανομής χρόνου dns2tcp 1

Το πρώτο διάγραμμα εξετάζει την περίπτωση της πρώτης κακόβουλης καταγραφής, dns2tcp 1. Εφόσον πρόκειται για την πρώτη σύγκριση, μπορούν να γίνουν ορισμένες γενικές παρατηρήσεις. Και στις δύο υλοποιήσεις, ένα σημαντικό ποσοστό χρόνου αφιερώνεται στην ανάγνωση των πακέτων. Υπενθυμίζεται πως η ντετερμινιστική υλοποίηση είναι αρκετά ταχύτερη από την πιθανοτική. Άρα, είναι λογικό ότι η ανάγνωση πακέτων ευθύνεται για ένα μεγάλο ποσοστό του χρόνου, καθώς οι υπόλοιπες λειτουργίες είναι σύντομες. Αντιθέτως, στην πιθανοτική υλοποίηση η ανάγνωση πακέτων (που εκτελείται με ίδιο τρόπο και χρειάζεται πρακτικά ίδιο χρόνο) χρησιμοποιεί μικρότερο ποσοστό του χρόνου, αφού οι άλλες λειτουργίες είναι περισσότερο χρονοβόρες.

Αναλυτικότερα, τον περισσότερο χρόνο στην πιθανοτική περίπτωση χρειάζεται ο υπολογισμός και η εισαγωγή των features, ενώ στην ντετερμινιστική ο αντίστοιχος χρόνος είναι πολύ μικρότερος. Από το διάγραμμα φαίνεται πως οι εντολές εισαγωγής στις πιθανοτικές δομές κοστίζουν ιδιαίτερα, ενώ οι αντίστοιχες εισαγωγές στην ντετερμινιστική είναι σχεδόν αμελητέες. Συγκεκριμένα η εισαγωγή του χρόνου φαίνεται να είναι η λιγότερο αποδοτική, ενώ οι υπόλοιπες είναι παρόμοιες μεταξύ τους. Μία πειραματική εξήγηση είναι πως επειδή πρόκειται για δεκαδικές τιμές (float) η T-Digest απαιτεί πιο πολύπλοκους χειρισμούς για να κάνει σωστή διαχείριση, ενώ τα μήκη πακέτων διαχειρίζονται πιο εύκολα. Η χρονική διαφορά αιτήματος από απόκριση είναι επίσης float, όμως οι εισαγωγές αυτές γίνονται με μικρότερη συχνότητα, άρα χρειάζονται λιγότερο χρόνο συνολικά.

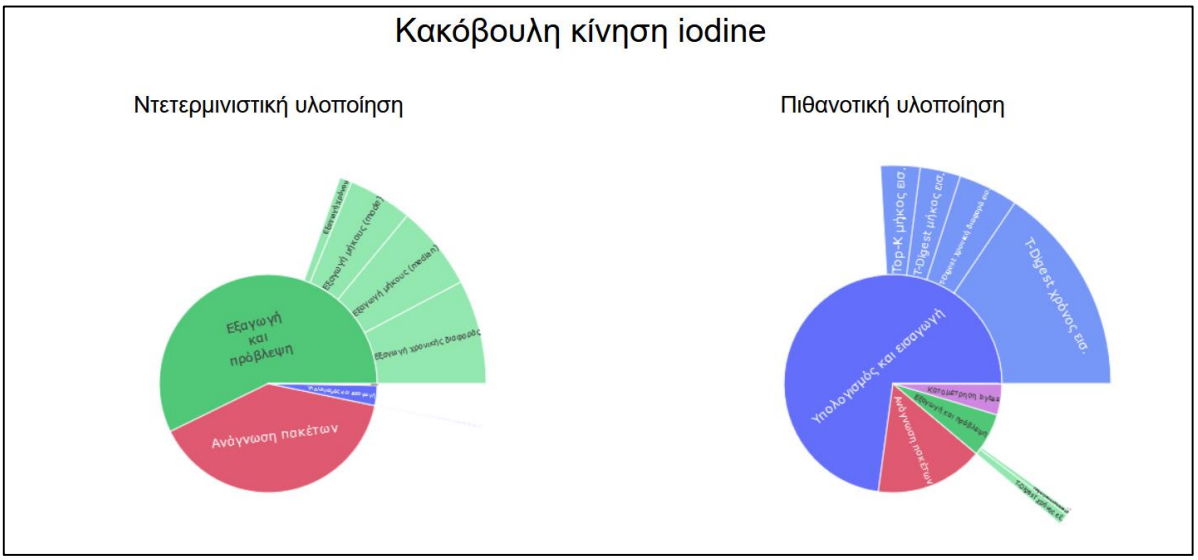
Το επόμενο σημαντικό τμήμα πρόκειται για την εξαγωγή των features και την πρόβλεψη. Εδώ παρατηρείται αντιστροφή της κατάστασης. Η ντετερμινιστική υλοποίηση κατανέμει πολύ χρόνο σε αυτό το στάδιο, ενώ η πιθανοτική ένα μικρό μέρος. Οι πιθανοτικές δομές έχουν σταθερό χρόνο εξαγωγής, ενώ οι στατιστικές συναρτήσεις που εφαρμόζονται στις λίστες χρειάζονται χρόνο για να εκτελέσουν πρώτα ταξινόμηση. Αυτό έχει κόστος, που αυξάνεται όσο μεγαλώνουν οι λίστες.



Σχήμα 23: Σύγκριση κατανομής χρόνου dns2tcp 2

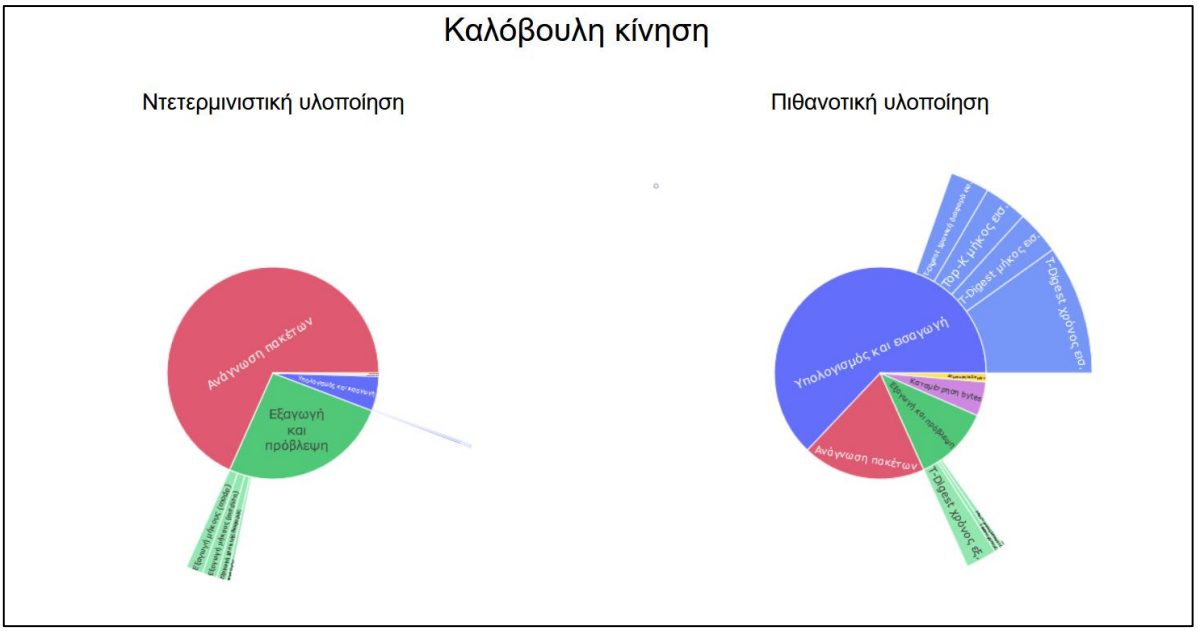
Η δεύτερη καταγραφή της dns2tcp κίνησης έχει λίγα, αλλά μεγάλα flows. Αυτό σημαίνει ότι η δημιουργία δομών δεν χρειάζεται πολύ χρόνο, αφού πρέπει να δημιουργηθούν λίγες. Ο μεγάλος αριθμός πακέτων συνεπάγεται μεγάλες λίστες, κάτι που δεν φαίνεται να επηρεάζει την πιθανοτική υλοποίηση. Αντιθέτως η ντετερμινιστική υλοποίηση δυσκολεύεται με τις

εξαγωγές. Όπως φαίνεται, χρειάζεται τόσο πολύ χρόνο για τις εξαγωγές των features που τα υπόλοιπα στάδια είναι σχεδόν αμελητέα.



Σχήμα 24: Σύγκριση κατανομής χρόνου iodine

Η iodine καταγραφή έχει επίσης σχετικά μεγάλα flows, αν και μικρότερα από την προηγούμενη. Επομένως, η ντετερμινιστική καταγραφή δυσκολεύεται πάλι με τις εξαγωγές, σε μικρότερο βαθμό. Η πιθανοτική υλοποίηση δεν επηρεάζεται.



Σχήμα 25: Σύγκριση κατανομής χρόνου καλόβουλης κίνησης

Η καλόβουλη κίνηση έχει πολλά και μικρά flows, άρα η ντετερμινιστική υλοποίηση έχει αποδοτική εξαγωγή features. Η πιθανοτική υλοποίηση εξακολουθεί να διατηρεί σταθερή τη συμπεριφορά της, με πολύ μεγάλους χρόνους για τις εισαγωγές.

4.4.3 Συμπεράσματα

Έχοντας ως κριτήριο το throughput και με βάση την σύγκριση που προηγήθηκε, η ετυμηγορία είναι σαφώς υπέρ της ντετερμινιστικής υλοποίησης. Αντίθετα με τον τομέα της μνήμης, όπου υπάρχει πληθώρα παραμέτρων που επηρεάζουν την συμφέρουσα επιλογή ανά περίπτωση, εδώ η απάντηση, με βάση πάντα τα πειράματα αυτά, είναι προφανής. Η πιθανοτική υλοποίηση χρειάζεται πολύ χρόνο για την εκτέλεση της, με αποτέλεσμα να χάνει πολλά πακέτα σε πραγματικά σενάρια.

Η επί μέρους ανάλυση της κατανομής ανέδειξε ιδιαίτερα προβληματική την εισαγωγή στις δομές, η οποία επιφέρει την μεγαλύτερη καθυστέρηση.

Κεφάλαιο 5

Συμπεράσματα και Μελλοντικό Έργο

Το παρόν κεφάλαιο περιλαμβάνει μια συνοπτική ανασκόπηση των κυριότερων θεμάτων που εξετάστηκαν στην εργασία, τονίζοντας τα βασικά ευρήματα και τις σημαντικές αναλύσεις που πραγματοποιήθηκαν. Παράλληλα, παρατίθενται τα τελικά συμπεράσματα, τα οποία αποτελούν την κορύφωση της ερευνητικής διαδικασίας και απαντούν στα ερωτήματα που τέθηκαν στην αρχή της μελέτης. Στο τέλος του κεφαλαίου, πραγματοποιείται μια ανασκόπηση των δυνατοτήτων για μελλοντικές επεκτάσεις, περιγράφοντας πιθανούς τομείς για περαιτέρω έρευνα και εφαρμογές, που μπορούν να ενισχύσουν και να επεκτείνουν τα αποτελέσματα της παρούσας εργασίας.

5.1 Ανασκόπηση

Η εργασία είχε σκοπό τον έλεγχο της απόδοσης και αξιοπιστίας ανίχνευσης κακόβουλης κρυπτογραφημένης DNS tunneling κίνησης με την χρήση πιθανοτικών δομών δεδομένων. Δημιουργήθηκαν δύο προγραμματιστικές υλοποιήσεις, μία πιθανοτική και μία ντετερμινιστική, ώστε να είναι δυνατή η σύγκριση των αποτελεσμάτων. Τα δεδομένα δικτυακής κίνησης πάνω στα οποία έγιναν τα πειράματα προήλθαν από το dataset [2].

Τα πειράματα έδειξαν πως η πιθανοτική υλοποίηση πετυχαίνει σωστή ταξινόμηση της κίνησης με ποσοστά επιτυχίας λίγο μικρότερα από τα αντίστοιχα της ντετερμινιστικής. Αυτό το μειονέκτημα όμως φάνηκε πως αντισταθμίζεται από σημαντική μείωση στην απαιτούμενη μνήμη.

5.2 Μελλοντικές επεκτάσεις

5.2.1 Επόμενα βήματα

Βασικό επόμενο βήμα για την εξέλιξη της εργασίας είναι ο πειραματισμός με πραγματικά δεδομένα, ώστε να διαπιστωθεί η αποτελεσματικότητα των υλοποιήσεων σε ρεαλιστικά σενάρια. Παρόλο που το υπάρχον dataset παρέχει μια ευρεία ποικιλία κίνησης, είναι κατασκευασμένο εργαστηριακά, γεγονός που περιορίζει τη δυνατότητα γενίκευσης των αποτελεσμάτων σε πραγματικές συνθήκες.

Η χρήση πραγματικών δεδομένων, τα οποία έχουν συλλεχθεί από πραγματικά δίκτυα, θα ενισχύσει την εγκυρότητα και τη χρησιμότητα της ανάλυσης, καθώς θα περιέχονται χαρακτηριστικά και αβεβαιότητες που δεν μπορούν να αποτυπωθούν πλήρως σε εργαστηριακές συνθήκες. Επιπλέον, στα πραγματικά δεδομένα, η κατηγοριοποίηση δεν χρειάζεται να περιοριστεί μόνο σε στατιστικές ιδιότητες αλλά μπορεί να βασιστεί και σε άλλα χαρακτηριστικά, όπως διευθύνσεις IP και ports, που έπρεπε να αγνοηθούν στην εργασία λόγω ανακύκλωσης.

Μάλιστα η ποικιλομορφία των δεδομένων θα επιφέρει προκλήσεις στα μοντέλα μάθησης, καθώς θα χρειάζονται πλέον ακόμα πιο ακριβείς μετρήσεις για να δίνουν σωστές προβλέψεις. Αυτό συνεπάγεται πως η ακρίβεια των πιθανοτικών δομών θα έχει μεγαλύτερη σημασία από πριν, άρα θα έχει μεγαλύτερο πρακτικό ενδιαφέρον το δίλλημα μνήμης – απόδοσης.

Το πέρασμα από την εργαστηριακή προσομοίωση στην ανάλυση πραγματικών δεδομένων συνιστά κρίσιμο βήμα για την προώθηση των σκοπών της έρευνας και τη βελτίωση των αποτελεσμάτων.

Επιπροσθέτως, ένα αξιοσημείωτο χαρακτηριστικό της εργασίας είναι η μελέτη κρυπτογραφημένης κίνησης. Δηλαδή, δεν έγινε εστιασμένη ανάλυση των χαρακτηριστικών του DNS tunneling, αλλά διαφόρων στατιστικών ιδιοτήτων των πακέτων. Αυτό επιτρέπει γενίκευση της μεθοδολογίας που ακολουθήθηκε και σε άλλες κατηγορίες προβλημάτων κρυπτογραφημένης δικτυακής κίνησης. Οι προγραμματιστικές υλοποιήσεις μπορούν πρακτικά να εφαρμοστούν αυτούσιες στα εν λόγω προβλήματα. Αν υπάρχουν datasets παρόμοια με αυτό που χρησιμοποιήθηκε εδώ, μπορεί να διεξαχθεί εκ νέου η ανάλυση της εργασίας για τον έλεγχο της ακρίβειας, επεκτείνοντας την δυνατότητα εφαρμογής της σε μεγάλο φάσμα κακόβουλης κίνησης.

Παράλληλα, η ανάλυση δύναται να επεκταθεί στον τομέα των μοντέλων μηχανικής μάθησης (ML). Στην παρούσα εργασία, τα μοντέλα χρησιμοποιήθηκαν αποκλειστικά ως εργαλεία, χωρίς να διερευνηθεί η συμπεριφορά τους ή η εσωτερική τους λειτουργία. Ωστόσο, η ανάλυση της σημασίας (feature importance) των χαρακτηριστικών που συμβάλλουν στις προβλέψεις μπορεί να προσφέρει πολύτιμες πληροφορίες. Αυτές με τη σειρά τους θα καθοδηγήσουν τη βελτιστοποίηση των χαρακτηριστικών που χρησιμοποιούνται, συμβάλλοντας στην ανάπτυξη υλοποιήσεων με ακόμα μεγαλύτερη ακρίβεια και αποδοτικότητα.

Επιπλέον, μια πιο λεπτομερής μελέτη της απόδοσης και των χαρακτηριστικών διαφόρων μοντέλων ML μπορεί να αναδείξει κάποιο συγκεκριμένο μοντέλο που εμφανίζει γενικά καλύτερη συμπεριφορά στην ταξινόμηση της κρυπτογραφημένης κίνησης. Τέτοια μοντέλα μπορεί να είναι πιο ανθεκτικά σε αβεβαιότητες ή σε δεδομένα με θόρυβο, ενώ παράλληλα να απαιτούν λιγότερους υπολογιστικούς πόρους, γεγονός που είναι κρίσιμο για την εφαρμογή τους σε πραγματικά δίκτυα.

Τέλος, η ανάλυση των μοντέλων ML μπορεί να αποκαλύψει περιοχές όπου η ακρίβεια μειώνεται ή παρουσιάζονται σφάλματα, οδηγώντας σε στοχευμένες βελτιώσεις. Η ενσωμάτωση αυτών των γνώσεων στη μεθοδολογία θα επιτρέψει τη δημιουργία ενός πιο αποδοτικού και ευέλικτου συστήματος, ικανού να προσαρμόζεται στις μεταβαλλόμενες συνθήκες και να ανταποκρίνεται σε διαφορετικούς τύπους κρυπτογραφημένης κίνησης.

5.2.2 Αντιμετώπιση προβλήματος

Τέλος, αξίζει να δοθεί έμφαση στο πρόβλημα που αντιμετωπίζει η πιθανοτική υλοποίηση, συγκεκριμένα στην ταχύτητα. Ο λόγος που παρατηρείται η μεγάλη καθυστέρηση είναι το προγραμματιστικό περιβάλλον. Η Python είναι μία σύγχρονη, ισχυρή γλώσσα όπου οι built-in δομές και συναρτήσεις τις λειτουργούν με τον βέλτιστο (optimal) τρόπο. Κάτι αντίστοιχο όμως δεν είναι σίγουρο για τις πιθανοτικές δομές, οι οποίες είναι εξωτερικά modules. Όμως, σε ένα περιβάλλον όπου οι σύνθετες πιθανοτικές δομές υλοποιούνται ως χαμηλού επιπέδου αντικείμενα και όχι ως συνδυασμός προϋπαρχουσών δομών, τα θετικά χαρακτηριστικά τους δύνανται να αναδειχθούν πλήρως. Συνεπώς, η δημιουργία ενός optimal περιβάλλοντος για την υλοποίηση των πιθανοτικών δομών ενδέχεται να αλλάξει πλήρως την κατάσταση, επιταχύνοντας δραστικά τους ρυθμούς επεξεργασίας.

Επίσης, όπως συζητήθηκε στις αντίστοιχες ενότητες, οι πιθανοτικές δομές παρέχουν στον χρήστη μεγάλη ευελιξία. Στα πλαίσια της ανάλυσης όμως, απαιτείται ένα πολύ μικρό υποσύνολο των δυνατοτήτων τους. Για παράδειγμα, η δομή T-Digest μπορεί να δώσει trimmed mean και percentiles σε οποιοδήποτε εύρος, αν και το μόνο που απαιτείται για την ανάλυση είναι μία διάμεσος. Αν οι δομές ανακατασκευάζονταν με απλούστερο τρόπο ώστε να προσφέρουν μόνο αυτό που απαιτείται, ίσως οδηγούσε σε αύξηση της ταχύτητας τους, χωρίς μείωση στην απόδοση τους.

Αντίστροφα, η ανάλυση θα μπορούσε να μεταβληθεί και να επεξεργάζεται περισσότερα features, τα οποία θα προκύπτουν από τις δομές. Η T-Digest θα μπορούσε να αξιοποιεί κάποια από τις δυνατότητες της οι οποίες μένουν αχρησιμοποίητες στα πλαίσια του παρόντος. Επί παραδείγματι, παράλληλα με το median μήκος πακέτου θα μπορούσε να αξιοποιείται για τις προβλέψεις και κάποια trimmed mean του μήκους. Αυτό προφανώς δεν θα αυξήσει την ταχύτητα της πιθανοτικής υλοποίησης, αλλά μπορεί με κατάλληλα επιλεγμένα features να βελτιώσει σημαντικά την ακρίβεια, αυξάνοντας ταυτόχρονα την αξιοποίηση της δομής. Η ντετερμινιστική υλοποίηση αντιθέτως μπορεί να δυσκολευθεί με τους επιπλέον σύνθετους υπολογισμούς και να έχει σημαντικές καθυστερήσεις, άρα η διαφορά στην ταχύτητα να μην είναι πλέον τόσο αξιόλογη.

Βιβλιογραφία

[1] P. E. Hoffman and P. McManus, “DNS Queries over HTTPS (DoH),” no. 8484. in Request for Comments. RFC Editor, Oct. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8484

[2] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari, “Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic,” in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2020, pp. 63–70. doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026.

[3] J. Gardiner, M. Cova, and S. Nagaraja, “Command & Control: Understanding, Denying and Detecting - A review of malware C2 techniques, detection and defences.” 2015. [Online]. Available: https://arxiv.org/abs/1408.1136

[4] Y. Wang, A. Zhou, S. Liao, R. Zheng, R. Hu, and L. Zhang, “A comprehensive survey on DNS tunnel detection,” *Comput. Netw.*, vol. 197, p. 108322, 2021, doi: https://doi.org/10.1016/j.comnet.2021.108322.

[5] N. Provos and P. Honeyman, “Hide and seek: an introduction to steganography,” *IEEE Secur. Priv.*, vol. 1, no. 3, pp. 32–44, 2003, doi: 10.1109/MSECP.2003.1203220.

[6] S. Zander, G. Armitage, and P. Branch, “A survey of covert channels and countermeasures in computer network protocols,” *IEEE Commun. Surv. Tutor.*, vol. 9, no. 3, pp. 44–57, 2007.

[7] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, “Deep Packet Inspection as a Service,” in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, in CoNEXT ’14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 271–282. doi: 10.1145/2674005.2674984.

[8] D. Naylor *et al.*, “The Cost of the ‘S’ in HTTPS,” in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, in CoNEXT ’14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 133–140. doi: 10.1145/2674005.2674991.

[9] Y. M. Banadaki, “Detecting Malicious DNS over HTTPS Traffic in Domain Name System using Machine Learning Classifiers,” *J. Comput. Sci. Appl.*, vol. 8, no. 2, pp. 46–55, 2020, doi: 10.12691/jcsa-8-2-2.

[10] S. K. Singh and P. K. Roy, “Detecting Malicious DNS over HTTPS Traffic Using Machine Learning,” in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 2020, pp. 1–6. doi: 10.1109/3ICT51146.2020.9312004.

[11] “Domain names - concepts and facilities,” no. 1034. in Request for Comments. RFC Editor, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1034

[12] “Domain names - implementation and specification,” no. 1035. in Request for Comments. RFC Editor, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1035

[13] C. Liu and P. Albitz, *DNS and Bind*. O’Reilly Media, Inc., 2006.

[14] V. Kampourakis, G. Kambourakis, E. Chatzoglou, and C. Zaroliagis, “Revisiting man-in-the-middle attacks against HTTPS,” *Netw. Secur.*, vol. 2022, no. 3, p. null, 2022, doi: 10.12968/S1353-4858(22)70028-1.

[15] A. Singh, S. Garg, R. Kaur, S. Batra, N. Kumar, and A. Y. Zomaya, “Probabilistic data structures for big data analytics: A comprehensive review,” *Knowl.-Based Syst.*, vol. 188, p. 104987, 2020, doi: https://doi.org/10.1016/j.knosys.2019.104987.
[16] G. Cormode and S. Muthukrishnan, “An improved data stream summary: the count-min sketch and its applications,” *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005, doi: https://doi.org/10.1016/j.jalgor.2003.12.001.
[17] T. Dunning, “The t-digest: Efficient estimates of distributions,” *Softw. Impacts*, vol. 7, p. 100049, 2021, doi: https://doi.org/10.1016/j.simpa.2020.100049.
[18] T. Yang *et al.*, “HeavyKeeper: An Accurate Algorithm for Finding Top- k Elephant Flows,” *IEEEACM Trans. Netw.*, vol. 27, no. 5, pp. 1845–1858, 2019, doi: 10.1109/TNET.2019.2933868.
[19] B. Claise, “Cisco Systems NetFlow Services Export Version 9,” no. 3954. in Request for Comments. RFC Editor, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3954
[20] S. Panchen, N. McKee, and P. Phaal, “InMon Corporation’s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks,” no. 3176. in Request for Comments. RFC Editor, Sep. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3176
[21] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[22] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[23] H. Zhang, “The optimality of naive Bayes,” *Aa*, vol. 1, no. 2, p. 3, 2004.
[24] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.
[25] M. Mayer, S. C. Bourassa, M. Hoesli, and D. Scognamiglio, “Machine Learning Applications to Land and Structure Valuation,” *J. Risk Financ. Manag.*, vol. 15, no. 5, 2022, doi: 10.3390/jrfm15050193.
[26] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Commun ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.
[27] R. Alenezi and S. A. Ludwig, “Classifying DNS Tunneling Tools For Malicious DoH Traffic,” in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021, pp. 1–9. doi: 10.1109/SSCI50451.2021.9660136.
[28] U. GUDEKLI and B. CIYLAN, “DNS Tunneling Effect on DNS Packet Sizes,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 8, no. 1, pp. 154–162, 2019.
[29] *pympler*. (2024). Python. [Online]. Available: https://github.com/pympler/pympler
[30] A. Gakhov, *Probabilistic data structures and algorithms for big data applications*. BoD–Books on Demand, 2022.
[31] P. T. Inc, “Collaborative data science.” [Online]. Available: https://plot.ly