

ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ



ΑΞΙΟΛΟΓΙΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΜΠΙΣΤΟΣΥΝΗΣ ΣΕ  
ΠΛΗΡΟΦΟΡΙΟΚΕΝΤΡΙΚΑ ΔΙΚΤΥΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παπαδόπουλος Βασίλειος  
ΑΜ : 56629

Επιβλέπων Καθηγητής : Τσαουσίδης Βασίλειος

Ξάνθη, 2021

## Ευχαριστίες

---

Η παρούσα διπλωματική εργασία με τίτλο «Αξιολόγηση αλγορίθμων εμπιστοσύνης σε πληροφοριοκεντρικά δίκτυα» διεκπεραιώθηκε στα πλαίσια περάτωσης των προπτυχιακών μου σπουδών. Πριν την παρουσίαση της, νιώθω την υποχρέωση να εκφράσω τις ευχαριστίες μου σε ορισμένους ανθρώπους.

Πρωτίστως, θα ήθελα να ευχαριστήσω τον επιβλέποντα μου Καθηγητή κ. Βασίλειο Τσαουσίδη, για την εμπιστοσύνη που μου έδειξε αναθέτοντας μου το θέμα της παρούσας διπλωματικής εργασίας καθώς και για την πολύτιμη καθοδήγησή του καθ' όλη την διάρκεια της.

Επίσης, θα ήθελα να ευχαριστήσω την υποψήφια διδάκτορα κα Καπετανίδου Ιωάννα-Αγγελική για τη συνεχή της στήριξη και τις σημαντικές συμβουλές της καθ' όλη την διάρκεια εκπόνησης της εργασίας.

Επιπλέον, θα ήθελα να εκφράσω θερμές ευχαριστίες στον Καθηγητή κ. Ηλία Κοσματοπούλο και στον Αναπληρωτή Καθηγητή κ. Παύλο Εφραιμίδη, που αφιέρωσαν το χρόνο τους για την αξιολόγηση της διπλωματικής εργασίας μου.

Τέλος, οφείλω τις ευχαριστίες μου στην οικογένειά μου και τους στενούς μου φίλους που δεν έπαψαν να είναι στο πλευρό μου σε όλη την ακαδημαϊκή μου πορεία και χάρη στην δική τους ηθική στήριξη έχω φτάσει μέχρι εδώ.

Παπαδόπουλος Βασίλειος

## Περιεχόμενα

|                                                              |        |
|--------------------------------------------------------------|--------|
| Ευχαριστίες .....                                            | - 2 -  |
| Λίστα Εικόνων .....                                          | - 5 -  |
| Περίληψη .....                                               | - 7 -  |
| Κεφάλαιο 1ο : Εισαγωγή .....                                 | - 9 -  |
| 1.1 Θέμα .....                                               | - 9 -  |
| 1.2 Οργάνωση Εργασίας .....                                  | - 10 - |
| Κεφάλαιο 2ο : Θεωρητικό Υπόβαθρο και Σχετικές Εργασίες ..... | - 11 - |
| 2.1 Η Αρχιτεκτονική Named Data Networking .....              | - 11 - |
| 2.1.1 NDN Caching .....                                      | - 14 - |
| 2.1.2 Reputation-based trust in NDN .....                    | - 15 - |
| 2.2 Dynamic Adaptive Streaming over NDN.....                 | - 18 - |
| 2.3 Αδυναμία στο DAS over NDN.....                           | - 20 - |
| 2.3.1 Προϋποθέσεις Επίθεσης .....                            | - 21 - |
| 2.3.2 Επίθεση Ταλάντωσης του Bitrate .....                   | - 21 - |
| 2.4 Reputation-based trust Scheme Design.....                | - 22 - |
| Κεφάλαιο 3ο : Εργαλεία .....                                 | - 25 - |
| 3.1 Προσομοιωτής δικτύων ns-3 .....                          | - 25 - |
| 3.2 Προσομοιωτής δικτύων ndnSIM.....                         | - 27 - |
| 3.3 Amust framework (Adaptive Multimedia Streaming) .....    | - 28 - |
| Κεφάλαιο 4ο : Μεθοδολογία .....                              | - 29 - |
| 4.1 Μοντέλο 1 .....                                          | - 30 - |
| 4.2 Μοντέλο 2 .....                                          | - 32 - |
| 4.3 Μοντέλο 3 .....                                          | - 34 - |
| 4.4 Γενικές Παράμετροι – Simulation Setup .....              | - 35 - |
| Κεφάλαιο 5ο : Υλοποίηση .....                                | - 37 - |
| 5.1 Πείραμα 1 .....                                          | - 38 - |
| 5.2 Πείραμα 2 .....                                          | - 39 - |
| 5.3 Πείραμα 3 .....                                          | - 39 - |
| 5.4 Πείραμα 4 .....                                          | - 40 - |
| 5.5 Πείραμα 5 .....                                          | - 40 - |
| Κεφάλαιο 6ο : Αποτελέσματα.....                              | - 42 - |
| 6.1 Πείραμα 1 .....                                          | - 42 - |
| 6.2 Πείραμα 2 .....                                          | - 51 - |
| 6.3 Πείραμα 3 .....                                          | - 58 - |

|                              |        |
|------------------------------|--------|
| 6.4 Πείραμα 4.....           | - 65 - |
| 6.5 Πείραμα 5.....           | - 74 - |
| Κεφάλαιο 7ο : Συζήτηση ..... | - 82 - |
| 7.1 Γενικά Συμπεράσματα..... | - 82 - |
| 7.2 Μελλοντική Εργασία ..... | - 82 - |
| Παράρτημα.....               | - 83 - |
| Βιβλιογραφία .....           | - 92 - |

## Λίστα Εικόνων

|                                                                                         |      |
|-----------------------------------------------------------------------------------------|------|
| Εικόνα 1 : NDN μοντέλο προώθησης [1] .....                                              | 13 - |
| Εικόνα 2 : NDN Interest Forwarding .....                                                | 15 - |
| Εικόνα 3 : NDN Data Caching/Forwarding .....                                            | 15 - |
| Εικόνα 4 : Dynamic adaptive multimedia streaming over ICN [29] .....                    | 19 - |
| Εικόνα 5 Number of switches (Μοντέλο 1, Πείραμα 1) .....                                | 43 - |
| Εικόνα 6 Average switch magnitude (Μοντέλο 1, Πείραμα 1) .....                          | 43 - |
| Εικόνα 7 Number of switches (Μοντέλο 2, Πείραμα 1) .....                                | 44 - |
| Εικόνα 8 Average switch magnitude (Μοντέλο 2, Πείραμα 1) .....                          | 45 - |
| Εικόνα 9 Number of switches (Μοντέλο 2, Πείραμα 1, threshold = 6) .....                 | 46 - |
| Εικόνα 10 Average switch magnitude (Μοντέλο 2, Πείραμα 1, threshold = 6) .....          | 46 - |
| Εικόνα 11 Number of switches (Μοντέλο 2, Πείραμα 1, threshold = 7) .....                | 47 - |
| Εικόνα 12 Average switch magnitude (Μοντέλο 2, Πείραμα 1, threshold = 7) .....          | 48 - |
| Εικόνα 13 Number of switches (Μοντέλο 3, Πείραμα 1) .....                               | 49 - |
| Εικόνα 14 Average switch magnitude (Μοντέλο 3, Πείραμα 1) .....                         | 49 - |
| Εικόνα 15 Number of switches (Σύγκριση Μοντέλων, Πείραμα 1) .....                       | 50 - |
| Εικόνα 16 Average switch magnitude (Σύγκριση Μοντέλων, Πείραμα 1) .....                 | 51 - |
| Εικόνα 17 Number of switches (Μοντέλο 1, Πείραμα 2) .....                               | 52 - |
| Εικόνα 18 Average switch magnitude (Μοντέλο 1, Πείραμα 2) .....                         | 53 - |
| Εικόνα 19 Number of switches (Μοντέλο 2, Πείραμα 2) .....                               | 54 - |
| Εικόνα 20 Average switch magnitude (Μοντέλο 2, Πείραμα 2) .....                         | 54 - |
| Εικόνα 21 Number of switches (Μοντέλο 3, Πείραμα 2) .....                               | 55 - |
| Εικόνα 22 Average switch magnitude (Μοντέλο 3, Πείραμα 2) .....                         | 56 - |
| Εικόνα 23 Number of switches (Μοντέλο 1, Πείραμα 3) .....                               | 59 - |
| Εικόνα 24 Average switch magnitude (Μοντέλο 1, Πείραμα 3) .....                         | 59 - |
| Εικόνα 25 Number of switches (Μοντέλο 2, Πείραμα 3) .....                               | 60 - |
| Εικόνα 26 Average switch magnitude (Μοντέλο 2, Πείραμα 3) .....                         | 60 - |
| Εικόνα 27 Number of switches (Μοντέλο 3, Πείραμα 3) .....                               | 61 - |
| Εικόνα 28 Average switch magnitude (Μοντέλο 3, Πείραμα 3) .....                         | 61 - |
| Εικόνα 29 Number of switches (Σύγκριση Μοντέλων, Πείραμα 3) .....                       | 63 - |
| Εικόνα 30 Average switch magnitude (Σύγκριση Μοντέλων, Πείραμα 3) .....                 | 63 - |
| Εικόνα 31 Number of switches (Σύγκριση Μοντέλων, Πείραμα 3) .....                       | 64 - |
| Εικόνα 32 Average switch magnitude (Σύγκριση Μοντέλων, Πείραμα 3) .....                 | 64 - |
| Εικόνα 33 Number of switches (Μοντέλο 1, Πείραμα 4) .....                               | 66 - |
| Εικόνα 34 Average switch magnitude (Μοντέλο 1, Πείραμα 4) .....                         | 66 - |
| Εικόνα 35 Number of switches (Μοντέλο 2, Πείραμα 4) .....                               | 67 - |
| Εικόνα 36 Average switch magnitude (Μοντέλο 2, Πείραμα 4) .....                         | 68 - |
| Εικόνα 37 Number of switches (Μοντέλο 3, Πείραμα 4) .....                               | 69 - |
| Εικόνα 38 Average switch magnitude (Μοντέλο 3, Πείραμα 4) .....                         | 70 - |
| Εικόνα 39 Number of switches (Σύγκριση μοντέλων, Πείραμα 4, onoff=1/3) .....            | 71 - |
| Εικόνα 40 Average switch magnitude (Σύγκριση μοντέλων, Πείραμα 4, onoff=1/3) ...        | 71 - |
| Εικόνα 41 Number of switches (Σύγκριση μοντέλων, Πείραμα 4, onoff=1/4) .....            | 72 - |
| Εικόνα 42 Average switch magnitude (Σύγκριση μοντέλων, Πείραμα 4, onoff=1/4) ...        | 73 - |
| Εικόνα 43 Number of switches (Μοντέλο 1, Πείραμα 5, threshold=5, onoff=1/2) -           | 74 - |
| Εικόνα 44 Average switch magnitude (Μοντέλο 1, Πείραμα 5, threshold=5, onoff=1/2) ..... | 75 - |
| Εικόνα 45 Number of switches (Μοντέλο 1, Πείραμα 5, threshold=7, onoff=1/3) -           | 76 - |

|                                                                                         |        |
|-----------------------------------------------------------------------------------------|--------|
| Εικόνα 46 Average switch magnitude (Μοντέλο 1, Πείραμα 5, threshold=7, onoff=1/3) ..... | - 76 - |
| Εικόνα 47 Number of switches (Μοντέλο 2, Πείραμα 5, threshold=5, onoff=1/2) -           | 77 -   |
| Εικόνα 48 Average switch magnitude (Μοντέλο 2, Πείραμα 5, threshold=5, onoff=1/2) ..... | - 77 - |
| Εικόνα 49 Number of switches (Μοντέλο 2, Πείραμα 5, threshold=7, onoff=1/3) -           | 78 -   |
| Εικόνα 50 Average switch magnitude (Μοντέλο 2, Πείραμα 5, threshold=7, onoff=1/3) ..... | - 78 - |
| Εικόνα 51 Number of switches (Μοντέλο 3, Πείραμα 5, threshold=5, onoff=1/2) -           | 79 -   |
| Εικόνα 52 Average switch magnitude (Μοντέλο 3, Πείραμα 5, threshold=5, onoff=1/2) ..... | - 79 - |
| Εικόνα 53 Number of switches (Μοντέλο 3, Πείραμα 5, threshold=7, onoff=1/3) -           | 80 -   |
| Εικόνα 54 Average switch magnitude (Μοντέλο 3, Πείραμα 5, threshold=5, onoff=1/2) ..... | - 81 - |

## Περίληψη

---

Η υφιστάμενη Host-Centric αρχιτεκτονική δυσκολεύεται να ανταποκριθεί στις προκλήσεις του σημερινού υπερφορτωμένου διαδικτύου. Για τον λόγο αυτό αναπτύσσονται διαφορετικά πρότυπα περισσότερο κατάλληλα για την αντιμετώπιση του προβλήματος. Η ICN (Information-Centric Networking) αναδύεται ως κατάλληλη πρόταση στις λειτουργικές απαιτήσεις του διαδικτύου. Ο σχεδιασμός στην Information-Centric αρχιτεκτονική βασίζεται στην ίδια την πληροφορία σε αντίθεση με την Host-Centric που το βασικό συστατικό της είναι οι IP διευθύνσεις. Η NDN συγκεντρώνει τις περισσότερες προσδοκίες από τα υπόλοιπα παραδείγματα ICN. Στην NDN κάθε πακέτο δεδομένων έχει διαφορετικό όνομα με βάση το οποίο γίνεται η δρομολόγηση του και είναι ανεξάρτητο από την τοποθεσία του χρήστη και της πηγής. Λόγω των μειωμένων απαιτήσεων εύρους ζώνης που διευκολύνει την αποτελεσματική κατανομή περιεχομένου, η NDN κρίνεται κατάλληλη για την υποστήριξη υπηρεσιών όπως η ροή πολυμέσων χρησιμοποιώντας πρότυπα όπως η προσαρμοστική ροή (DAS).

Αυτή η ριζική μεταρρύθμιση οδηγεί σε σημαντικές διαφορές στις λειτουργίες παράδοσης δεδομένων και μας φέρνει αντιμέτωπους με καινούργιες προκλήσεις. Οι περισσότερες λειτουργίες ασφάλειας στην NDN βασίζονται σε μεθόδους πιστοποίησης και λαμβάνουν χώρα στην πλευρά των καταναλωτών. Οι επιθέσεις ταλάντωσης του bitrate (Boas) όμως -όπως και άλλες επιθέσεις- δεν περιορίζονται από αυτή τη στρατηγική και απαιτούν διαφορετική προσέγγιση. Με την ταλάντωση του bitrate ο επιτιθέμενος προσδοκά την συνεχή αλλαγή της ποιότητας του περιεχομένου πολυμέσων -μέσω της λειτουργίας DAS- και τελικά στην χαμηλή ποιότητα της εμπειρίας του χρήστη.

Στην παρούσα διπλωματική εργασία εξετάζουμε τρεις μεθόδους άμυνας, που βασίζονται στην φήμη (reputation-based) σε ένα συγκεκριμένο σενάριο (βίντεο αναφοράς). Βασιστήκαμε σε ένα ήδη ανεπτυγμένο μοντέλο και αναπτύξαμε δύο καινούργια προσαρμόζοντας διαφορετικούς μηχανισμούς βαθμολόγησης στο αρχικό μοντέλο άμυνας και τέλος τα εξετάσαμε στο αρχικό σενάριο. Τα τρία μοντέλα που βασίζονται στη φήμη δεν διαφέρουν στον τρόπο ανίχνευσης του επιτιθέμενου αλλά στην διαχείριση των στοιχείων που προκύπτουν από την ανίχνευση και τελικά στην

κρίση ενός χρήστη ως κακόβουλου ή καλόβουλου. Υλοποιήθηκαν προσομοιώσεις με την βοήθεια του Amust-ndnSIM και τα αποτελέσματα αξιολογούνται ως προς δύο μετρήσεις: τον αριθμό των ταλαντώσεων και την ένταση αυτών.

## Κεφάλαιο 1ο : Εισαγωγή

---

### 1.1 Θέμα

Στην παρούσα διπλωματική εργασία θα ασχοληθούμε με ένα θεμελιώδες χαρακτηριστικό της NDN αρχιτεκτονικής. Πρόκειται για την In-network αποθήκευση δεδομένων στους δρομολογητές του δικτύου. Αυτό το χαρακτηριστικό αν και συμβάλλει στην μείωση των συνολικών απαιτήσεων καθυστέρησης και εύρους ζώνης δημιουργεί νέα θέματα ασφάλειας [4]. Στις περιπτώσεις των πολυμέσων περιεχομένου συγκεκριμένα λαμβάνουν χώρα οι επιθέσεις ταλάντωσης του bitrate (Boas) οι οποίες εκμεταλλεύονται την In-network αποθήκευση και σε συνδυασμό με το πρωτόκολλο DAS σε NDN αναγκάζουν τους χρήστες σε περιττές ταλαντώσεις bitrate υποβαθμίζοντας την ποιότητα εμπειρίας τους.

Ο μηχανισμός άμυνας που εξετάζεται εδώ εκμεταλλεύεται ένα χαρακτηριστικό που προκύπτει και πάλι από την NDN αρχιτεκτονική. Η δυνατότητα να αξιολογείται το ίδιο το περιεχόμενο και έμμεσα ο εκάστοτε χρήστης που ζητάει αυτό το περιεχόμενο. Στον προτεινόμενο μηχανισμό φήμης ο κάθε δρομολογητής εκχωρεί τιμές φήμης σε κάθε περιεχόμενο με σκοπό την απαγόρευση της προσωρινής αποθήκευσης περιεχομένου με χαμηλή τιμή φήμης. Το περιεχόμενο επαναξιολογείται με την λήψη κάθε πακέτου. Η τιμή φήμης του ενημερώνεται και τελικά ο δρομολογητής αποφασίζει σε τακτά χρονικά διαστήματα αν θα αποθηκεύσει ή όχι περιεχόμενο. Η τιμή που εκχωρείται όταν γίνεται αντιληπτή μία κακόβουλη συμπεριφορά ή μία θετική συμπεριφορά μπορεί να κρίνει την αποτελεσματικότητα του μηχανισμού. Η παρούσα διπλωματική εξετάζει την διαχείριση αυτής της πληροφορίας με τρεις διαφορετικούς τρόπους σε ένα συγκεκριμένο σενάριο.

Μελετάμε τρία μοντέλα με βάση τα παραπάνω και από τα πειράματα που εκτελούμε, στόχος είναι να βελτιώσουμε τα αρχικά μοντέλα, να βρούμε το πιο αποτελεσματικό και να κατανοήσουμε σε βάθος τους μηχανισμούς άμυνας και επίθεσης σχολιάζοντας τα αποτελέσματα.

## 1.2 Οργάνωση Εργασίας

Στο δεύτερο κεφάλαιο παρουσιάζονται εργασίες που έχουν γίνει σχετικά με την αρχιτεκτονική NDN, τις επιθέσεις ταλάντωσης του bitrate και το προτεινόμενο μοντέλο άμυνας βασιζόμενο στην φήμη.

Στο τρίτο κεφάλαιο γίνεται μια σύντομη περιγραφή των εργαλείων που χρησιμοποιήθηκαν στην εργασία. Πιο συγκεκριμένα, παρουσιάζονται ο προσομοιωτής δικτύων ns-3, ο προσομοιωτής δικτύων ndnSIM και ο προσομοιωτής Amust-ndnSIM.

Στο τέταρτο κεφάλαιο περιγράφεται η μεθοδολογία που ακολουθήσαμε. Παρουσιάζονται τα τρία διαφορετικά μοντέλα reputation-based που θα χρησιμοποιηθούν στις προσομοιώσεις και το μοντέλο αναφοράς με βάση το οποίο υλοποιούνται.

Στο πέμπτο κεφάλαιο αναλύονται οι πέντε διαφορετικές συνθήκες κατά τις οποίες εκτελούνται οι προσομοιώσεις.

Στο έκτο κεφάλαιο παρουσιάζονται τα αποτελέσματα των προσομοιώσεων και αξιολογούνται τα μοντέλα ως προς την συνθήκη της εκάστοτε προσομοίωσης.

Στο έβδομο και τελευταίο κεφάλαιο συμπεριλαμβάνονται γενικές παρατηρήσεις και συμπεράσματα καθώς και οι στόχοι που θέτουμε για μελλοντική επέκταση της εργασίας μας.

Τέλος, έχουμε προσθέσει ένα παράρτημα όπου παρατίθεται το σημαντικότερο τμήμα του κώδικά μας.

## Κεφάλαιο 2ο : Θεωρητικό Υπόβαθρο και Σχετικές Εργασίες

---

Σε αυτό το κεφάλαιο θα ασχοληθούμε με το θεωρητικό υπόβαθρο το οποίο είναι απαραίτητο για την εργασία μας. Η έρευνα και η μελέτη εργασιών που σχετίζονται με το αντικείμενο της εργασίας είναι σημαντική για την κατανόηση των εννοιών του Information-Centric Networking, του Named Data Networking και του Dynamic Adaptive Streaming πάνω στην NDN. Μέσα από την κατανόηση αυτών των εννοιών θα μπορούμε να κατανοήσουμε βαθύτερα την επίθεση ταλαντώσεων του bitrate αλλά και τον τρόπο με τον οποίο την αντιμετωπίζουμε.

### 2.1 Η Αρχιτεκτονική Named Data Networking

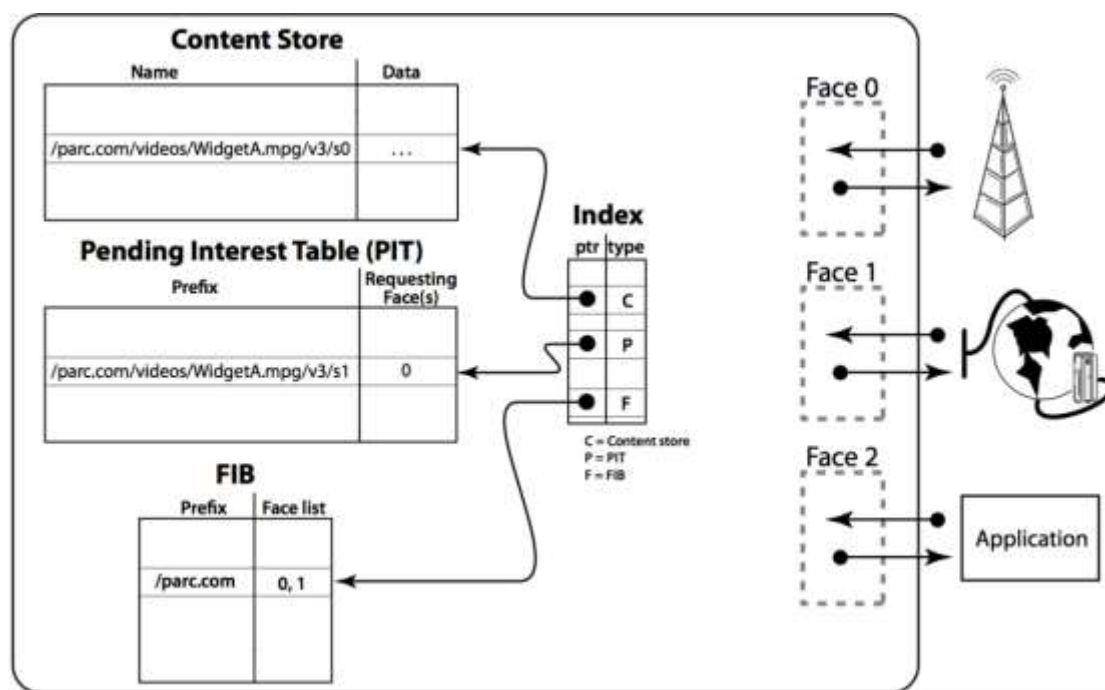
Όταν αναπτύχθηκαν οι βασικές ιδέες του διαδικτύου η τηλεφωνία ήταν το μοναδικό φαινόμενο επιτυχημένης παγκόσμιας επικοινωνίας. Τότε δημιουργήθηκε το πρωτόκολλο TCP/IP για να λύσει το σχεδιαστικό πρόβλημα επικοινωνίας μεταξύ δύο σημείων. Σε αυτόν τον παραδοσιακό τρόπο λειτουργίας, ο οποίος υπερέβη κάθε προσδοκία και εφαρμόζεται μέχρι σήμερα, ο πυρήνας της σχεδίασης είναι τα δύο σημεία της επικοινωνίας και πιο συγκεκριμένα οι διευθύνσεις IP οι οποίες αποτυπώνουν ψηφιακά τα σημεία αυτά. Το πρωτόκολλο TCP/IP ανήκει στο Host-Centric Networking [1].

Το ICN (Information-Centric Networking) αποτελεί μια ριζοσπαστική πρόταση για τον σχεδιασμό των μελλοντικών δικτύων υπολογιστών. Ο σχεδιασμός σε αυτά τα δίκτυα βασίζεται στην ίδια την πληροφορία σε αντίθεση με το παραδοσιακό πρότυπο Host-Centric Networking και προέκυψε ως αρχιτεκτονική λύση που ανταποκρίνεται αποτελεσματικότερα στις ανάγκες του σύγχρονου υπερφορτωμένου διαδικτύου. Οι ανάγκες αυτές γίνονται εύκολα αντιληπτές αν σκεφτεί κανείς πόσο εύκολο έγινε για τον οποιοδήποτε να ανακαλύψει, να καταναλώσει και να δημιουργήσει περιεχόμενο. Σήμερα παράγονται και διανέμονται exabytes νέων δεδομένων κάθε χρόνο [1]. Η ICN καθιστά τα ίδια τα δεδομένα κεντρικό στοιχείο στην αρχιτεκτονική του διαδικτύου. Η NDN (Named Data Networking) αρχιτεκτονική αποτελεί το πιο διαδεδομένο παράδειγμα του ICN και αποτελεί την απικρατέστερη επιλογή για την σχεδίαση του μελλοντικού διαδικτύου [1].

Στην NDN λοιπόν, κάθε πακέτο δεδομένων έχει μοναδικό όνομα το οποίο το διαφοροποιεί από τα άλλα πακέτα όπως ακριβώς διαφοροποιεί σήμερα η IP το κάθε σημείο αποστολής ή παραλαβής πακέτων. Αρχικά, πρέπει να σημειώσουμε ότι τους κόμβους που ζητούν πακέτα δεδομένων τους ονομάζουμε καταναλωτές (consumers) και αυτούς που δημιουργούν πακέτα δεδομένων τους ονομάζουμε παραγωγούς (producers) ή παρόχους. Ένας καταναλωτής εκδίδει πακέτα αίτησης για να ζητήσει το επιθυμητό περιεχόμενο, τα οποία ονομάζουμε Interest packets. Έτσι, τα πακέτα χωρίζονται σε αυτά που στέλνουν οι καταναλωτές “Interest packets” και σε αυτά που δέχονται οι καταναλωτές, τα οποία ονομάζονται “data packets”. Σε αντίθεση με το Host-Centric Networking στην NDN αρχιτεκτονική όταν ο καταναλωτής επιθυμεί να λάβει ένα περιεχόμενο στέλνει ένα Interest Packet το οποίο προσδιορίζει απευθείας το όνομα του περιεχομένου που τον ενδιαφέρει και όχι τη θέση του παρόχου περιεχομένου. Στη συνέχεια, σε κάθε δρομολογητή (router) που διασχίζει το Interest Packet λαμβάνουν χώρα οι θεμελιακές λειτουργίες της NDN που βασίζονται στις δομές FIB, PIT και CS. Ο κάθε κόμβος με την έλευση ενός νέου Interest αναζητεί την σχετική καταχώρηση στις δομές CS, PIT και FIB με την σειρά που αναφέρονται. Αν αποτύχει στον εντοπισμό της σχετικής καταχώρησης στην δομή CS τότε και μόνο εκτελείται αναζήτηση στην δομή PIT. Αν η αναζήτηση στην δομή PIT αποτύχει τότε και μόνο τότε ελέγχει ο δρομολογητής τη δομή FIB. Στη βάση δεδομένων FIB (Forwarding Information Base) η οποία συμπληρώνεται από το πρωτόκολλο δρομολόγησης, υπάρχουν πληροφορίες σχετικές με την προώθηση του Interest σε επόμενο router. Εκεί αναζητεί ο router το όνομα που του υποδεικνύει το Interest και στη συνέχεια το προωθεί με βάση τις πληροφορίες βρίσκει στην FIB. Στη βάση δεδομένων PIT (Pending Interest Table) αποθηκεύονται όλα τα Interest που έχουν προωθηθεί από τον router και αναμένουν να ικανοποιηθούν με την επιστροφή του ζητούμενου πακέτου δεδομένων. Εκεί αποθηκεύεται και η διασύνδεση από την οποία έρχεται το Interest έτσι ώστε όταν έρθει το αναμενόμενο πακέτο δεδομένων ο router να το προωθήσει προς την αντίστοιχη διασύνδεση. Όταν έρθει ένα Interest το οποίο ζητά ένα πακέτο δεδομένων και στη PIT υπάρχει η καταχώρηση με το συγκεκριμένο όνομα τότε δεν δημιουργείται ξεχωριστή καταχώρηση αλλά προστίθεται στην παλιά η διασύνδεση του νέου Interest έτσι, κάθε καταχώρηση μπορεί να περιέχει ένα όνομα και ένα σύνολο από διασυνδέσεις (Interest Aggregation). Επιπλέον, αυτό το Interest δεν προωθείται αφού έχει ήδη ζητηθεί από τον router το πακέτο δεδομένων που

επιθυμεί. Μόλις έρθει το αναμενόμενο πακέτο δεδομένων ο router βρίσκει την αντίστοιχη καταχώρηση στη PIT, προωθεί τα δεδομένα σε όλες τις διασυνδέσεις και διαγράφει την καταχώρηση αυτή από την βάση. Παράλληλα, αποθηκεύει προσωρινά τα δεδομένα στην βάση Content Store για να ικανοποιήσει μελλοντικά Interest (Caching). Παρατηρούμε ότι με αυτόν το τρόπο μπορεί ένα Interest να ικανοποιηθεί από έναν router πριν φτάσει στον παραγωγό, μία πολύ σημαντική λειτουργία με την οποία θα ασχοληθούμε εκτενέστερα στη συνέχεια.

Στην παρακάτω εικόνα φαίνονται σχηματικά οι λειτουργίες και οι δομές που μόλις αναλύσαμε.



Εικόνα 1 : NDN μοντέλο προώθησης [1]

Ένα πακέτο δεδομένων στη NDN αρχιτεκτονική έχει νόημα ανεξάρτητο από το σημείο από το οποίο προέρχεται και από αυτό που μπορεί να προωθηθεί έτσι, μπορεί να αποθηκευτεί προσωρινά μέσα στο δίκτυο για να ικανοποιήσει μελλοντικά Interest. Επιπλέον, τα πακέτα δεδομένων και τα πακέτα Interest στην NDN αρχιτεκτονική δεν μεταφέρουν πληροφορίες σχετικές με τον παραγωγό και τον καταναλωτή, αυτό είναι πολύ σημαντικό για την ασφάλεια της επικοινωνίας (π.χ. είναι δύσκολη η στοχοποίηση ενός συγκεκριμένου υπολογιστή). Η ασφάλεια στην NDN δεν είναι μια λειτουργία που πρέπει να αναπτύξουμε αλλά είναι ενσωματωμένη στα ίδια τα πακέτα που μεταφέρονται [1].

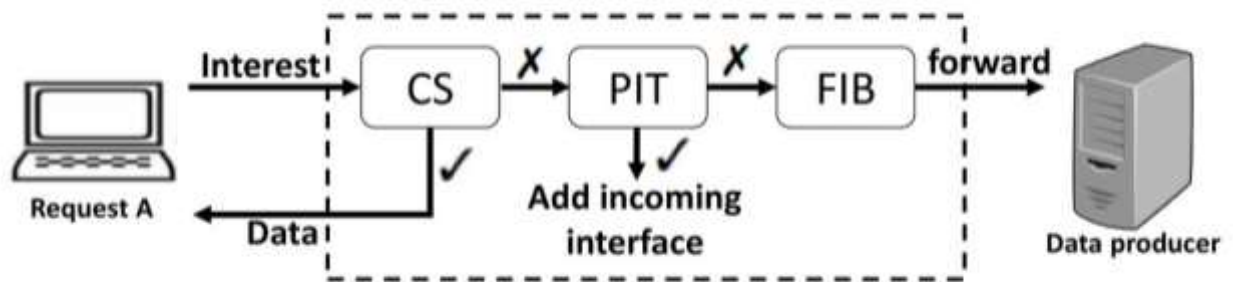
Η δρομολόγηση και η προώθηση στην NDN λύνει τέσσερα πολύ σημαντικά προβλήματα της host-centric δικτύωσης, αυτά είναι:

1. Η εξάντληση του χώρου διευθύνσεων (ο χώρος των ονομάτων είναι απεριόριστος).
2. Η διέλευση NAT αφού στην NDN ένας κεντρικός υπολογιστής δεν χρειάζεται να δημοσιεύσει την διεύθυνση του.
3. Η κινητικότητα δεδομένου ότι τα ονόματα των δεδομένων παραμένουν ίδια για κάθε νέα θέση του πελάτη ενώ η διεύθυνση IP στο host-centric θα πρέπει να αλλάζει και να διακόπτει ίσως την επικοινωνία.
4. Η διαχείριση διευθύνσεων (Address assignment and management), δεν απαιτείται πλέον στα τοπικά δίκτυα, γεγονός που είναι πολύ σημαντικό για τα ενσωματωμένα δίκτυα αισθητήρων (embedded sensor networks).

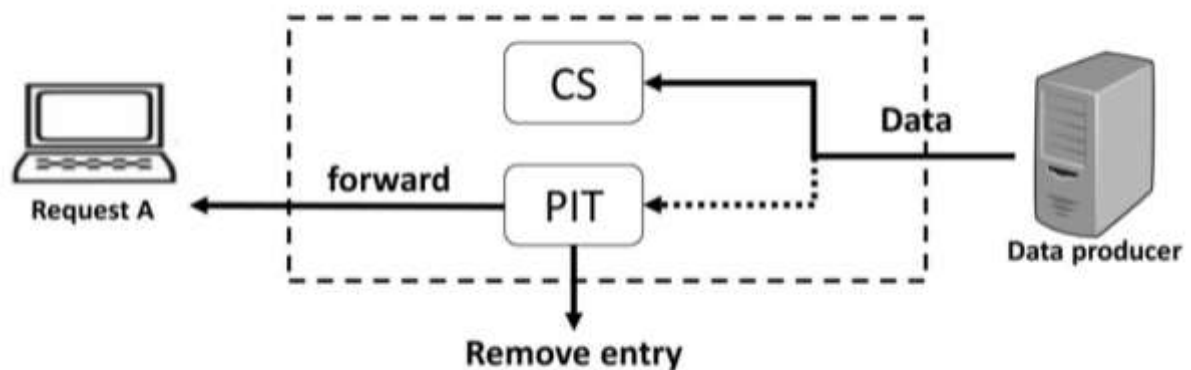
### **2.1.1 NDN Caching**

Η κρυφή μνήμη (cache) είναι μια λειτουργία των router που ανήκουν στην NDN αρχιτεκτονική με την οποία ο κάθε router μπορεί να αποθηκεύσει προσωρινά στο Content Store (CS) πακέτα δεδομένων για να ικανοποιήσει μελλοντικές αιτήσεις. Μόλις φτάσει ένα νέο Interest, ο router ψάχνει πρώτα στο Content Store για να εξετάσει αν μπορεί να το ικανοποιήσει ο ίδιος ή αν πρέπει να το προωθήσει. Το γεγονός ότι τα πακέτα δεδομένων προσδιορίζονται από το όνομα τους επιτρέπει την επαναχρησιμοποίηση τους πολλαπλές φορές σε αντίθεση με τις cache μνήμες των ID router όπου μετά την προώθηση τους από την cache χάνεται αυτή η δυνατότητα [1]. Με την προσθήκη ωστόσο περισσότερου περιεχομένου στις cache των router μετακινούμε και περισσότερες ευθύνες σε αυτούς για θέματα ασφάλειας [2].

Στις δύο παρακάτω εικόνες βλέπουμε τις διαδικασίες που περιγράψαμε παραπάνω· ότι περικλείεται στο διακεκομμένο περίγραμμα αφορά ενέργειες που λαμβάνουν χώρα μέσα σε κάθε τυπικό router της NDN.



Εικόνα 2 : NDN Interest Forwarding



Εικόνα 3 : NDN Data Caching/Forwarding

### 2.1.2 Reputation-based trust in NDN

Η ασφάλεια στην NDN αποτελείται από τρία θεμελιώδους σημασίας δομικά στοιχεία. Αυτά είναι τα Digital Keys, τα Certificates και οι Trust Policies. Με τα τρία αυτά συστατικά εξασφαλίζεται η αυθεντικότητα των δεδομένων, η ακεραιότητα και υπό προϋποθέσεις η εμπιστευτικότητα [3].

Τα περισσότερα ωστόσο προβλήματα ασφαλείας στην NDN προκύπτουν από το γεγονός ότι οι routers στην NDN πρέπει να υποστηρίζουν δομές όπως η βάση PIT και το Content Store. Από αυτές τις δύο δομές που υπάρχουν σε κάθε router το

Content Store είναι αυτό που ελέγχεται πρώτο με την έλευση κάθε νέου Interest. Εφόσον δεν βρεθεί εκεί το ζητούμενο περιεχόμενο ο router ελέγχει την άλλη δομή στην οποία αν υπάρχει καταχώρηση που να αναφέρεται στο ίδιο περιεχόμενο απλά προσθέτει την διασύνδεση του νέου Interest σε αυτήν. Στην περίπτωση που δεν υπάρχει τέτοια καταχώρηση δημιουργεί μια νέα και μόνο σε αυτή την περίπτωση το Interest προωθείται από τον router με πληροφορίες προώθησης που θα βρει στην βάση FIB (Forwarding Information Base). Οι δομές CS και PIT όμως πέρα από την αποδοτικότητα που προσφέρουν στο σύστημα το καθιστούν εν τέλει ευαίσθητο σε επιθέσεις DoS [3]. Σε αυτές τις επιθέσεις ο επιτιθέμενος προσπαθεί να εξαπατήσει τον router είτε στοχεύοντας στον πίνακα PIT είτε στην δομή CS. Στις επιθέσεις που στοχεύουν το CS ονομάζονται cache-related attacks και τέτοιες είναι η cache pollution, η cache poisoning και η cache snooping.

Ένας τρόπος προστασίας των δρομολογητών θα μπορούσε να είναι η επαλήθευση της εγκυρότητας της υπογραφής του παραγωγού, η οποία εμπεριέχεται στο εκάστοτε πακέτο δεδομένων. Ωστόσο, η επαλήθευση υπογραφής είναι μία βαριά κρυπτογραφική λειτουργία τόσο από τη μεριά του υπολογιστικού κόστους αλλά και από αυτή της προστιθέμενης πολυπλοκότητας στο δίκτυο. Οπότε, κρίνεται απαγορευτική να συμβαίνει σε κάθε router του δικτύου [4,5], ειδικά σε σύγκριση με την παραγωγή υπογραφής. Η επίθεση cache poisoning εκμεταλλεύεται ακριβώς αυτή την έλλειψη ενός συστήματος επαλήθευσης υπογραφής εντός δικτύου, για να εισάγει μη έγκυρο περιεχόμενο στα CS του δικτύου. Για να μετριάσουν αυτές τις επιθέσεις οι routers αξιοποιούν εναλλακτικές, λιγότερο χρονοβόρες και κοστοβόρες μεθόδους αντιμετώπισης. Μία τέτοια βασική μέθοδος είναι τα συστήματα εμπιστοσύνης που βασίζονται στην φήμη (reputation-based trust systems) [3].

Πρόκειται για ένα μηχανισμό στον οποίο μια κατανεμημένη κοινότητα αξιολογεί το περιεχόμενο και με βάση αυτή την αξιολόγηση αποφασίζει ο router αν θα αποθηκεύσει περιεχόμενο ή όχι [3]. Με λίγα λόγια, η διαχείριση εμπιστοσύνης σε αυτή την προσέγγιση γίνεται με βάση την φήμη (reputation based) που έχει κάθε πακέτο δεδομένων. Μία προσέγγιση της reputation-based βασίζεται σε ένα προαιρετικό πεδίο μέσα στο Interest packet το οποίο ονομάζεται Exclude [6]. Αυτό συμπληρώνεται από τον καταναλωτή και παρέχει πληροφορίες στον router για τα περιεχόμενα που θεωρεί κακόβουλα. Αυτές οι πληροφορίες ονομάζονται πληροφορίες αποκλεισμού και η τεχνική αυτή είναι εμπνευσμένη από την παραδοσιακή επαλήθευση υπογραφών. Πιο συγκεκριμένα, αν ένας καταναλωτής λάβει κακόβουλο

περιεχόμενο μπορεί σε επόμενο Interest, μέσω αυτού του πεδίου, να δηλώσει στους routers ότι θέλει να αποκλείσει το συγκεκριμένο περιεχόμενο. Αυτή η τεχνική είναι ωστόσο επιρρεπής σε κάποιες επιθέσεις. Αν πολλοί επιτιθέμενοι στοχεύσουν ένα έγκυρο περιεχόμενο μπορούν να το αποκλείσουν ή αν ζητήσουν ένα κακόβουλο περιεχόμενο μπορούν να το ανεβάσουν κατάταξη στην cache των routers. Ένας μηχανισμός [7] που θα εκχωρεί τιμές φήμης πέρα από το ίδιο το περιεχόμενο και σε κάθε πάροχο περιεχομένου θα είναι πιο ενισχυμένος σε σύγκριση με τον παραπάνω. Δηλαδή ένας πάροχος θα κρίνεται έμπιστος με βάση την αξία περιεχομένου που παρείχε στο δίκτυο. Εδώ πρέπει να σημειώσουμε ότι τα περιεχόμενα έχουν trust values ενώ οι πάροχοι περιεχομένου έχουν credibility values. Ωστόσο, ο μηχανισμός με το πεδίο exclusion δεν είναι πλέον διαθέσιμος στην NDN v.3.

Μία γενική ανάλυση διαχείρισης εμπιστοσύνης με βάση την φήμη μπορεί να περικλείει και αξιολογήσεις για κάθε router για κάθε παραγωγό και για κάθε καταναλωτή. Έτσι, η διαχείριση εμπιστοσύνης με βάση την φήμη μπορεί να χρησιμοποιηθεί και για τον περιορισμό επιθέσεων πλημμύρας (flooding attacks) . Ο μηχανισμός ICRP που βασίζεται στην διαχείριση εμπιστοσύνης εκχωρεί μία τιμή φήμης (reputation value) σε κάθε καταναλωτή [8]. Αυτές οι τιμές αντιπροσωπεύουν τον βαθμό μετάδοσης των Interests που ζητούν περιεχόμενο που υπάρχει ήδη. Στη συνέχεια αυτές οι τιμές φήμης συγκρίνονται με ένα κατώφλι (threshold) και έτσι κάθε router κρίνει αν ένας καταναλωτής είναι κακόβουλος ή έχει καλές προθέσεις. Επιπλέον, η ICRP καταγράφει ανύπαρκτα ονόματα περιεχομένου δημιουργώντας μια μαύρη λίστα από αυτά έτσι ώστε να περιορίσει περαιτέρω την ροή κακόβουλων Interests στο δίκτυο.

Ένας τρόπος επιλογής του περιεχομένου που θα αποθηκευτεί ή αλλιώς της συμπεριφοράς του router απέναντι σε κακόβουλους χρήστες, εφόσον υπάρχει μια reputation value είναι η αποθήκευση με πιθανότητα ίση με την τιμή reputation value. Ένας δεύτερος τρόπος επιλογής είναι η αποθήκευση μόνο των περιεχομένων εκείνων που η reputation value ξεπερνάει ένα κατώφλι (threshold). Η εκτέλεση της επιλεκτικής αποθήκευσης λοιπόν μπορεί να γίνει με αυτούς τους δύο τρόπους ή με τον συνδυασμό αυτών [9].

Οπότε, η Reputation-based trust στην NDN βασίζεται στις παρακάτω τιμές φήμης ανάλογα με τον μηχανισμό που χρησιμοποιεί κάθε φορά:

- Trust values για αξιολόγηση περιεχομένου
- Credibility values για αξιολόγηση της αξιοπιστίας των βαθμολογητών
- Reputation values για αξιολόγηση καταναλωτών ή δρομολογητών

## 2.2 Dynamic Adaptive Streaming over NDN

Οι υπηρεσίες streaming (δηλαδή περιεχόμενο βίντεο ή ήχου) καλύπτουν το 50% των πακέτων που κινούνται στο Internet [10]. Αυτό οδήγησε τους παρόχους περιεχομένου να αναζητούν τρόπους βελτιστοποίησης του bandwidth. Ένας πολύ αποτελεσματικός τρόπος που χρησιμοποιούν πάροχοι περιεχομένου όπως το Netflix και το Youtube είναι η χρήση του πρωτοκόλλου DASH (Dynamic Adaptive Streaming over HTTP), το οποίο υποστηρίζει δυναμική προσαρμογή του streaming στις συνθήκες του δικτύου [11]. Στο DASH ο στόχος είναι να κατεβάσει ο χρήστης το βίντεο υψηλότερης δυνατής ανάλυσης (ποιότητας) με βάση το διαθέσιμο bandwidth. Δεδομένης της αποτελεσματικότητας της ICN στην μείωση των απαιτήσεων του bandwidth αλλά και των νέων θεμάτων ασφαλείας που προκύπτουν από αυτή, η ερευνητική κοινότητα έχει ερευνήσει σε βάθος την εφαρμογή του DASH στην ICN [12]-[15]. Πολλά αποτελέσματα ερευνών κρίνουν αποτελεσματική την χρήση DASH με in-network caching εντός δικτύου ICN και αναδεικνύουν τα πλεονεκτήματα της [16]-[20]. Έτσι, το DAS έχει γίνει η πιο ευρέως χρησιμοποιούμενη μέθοδος.

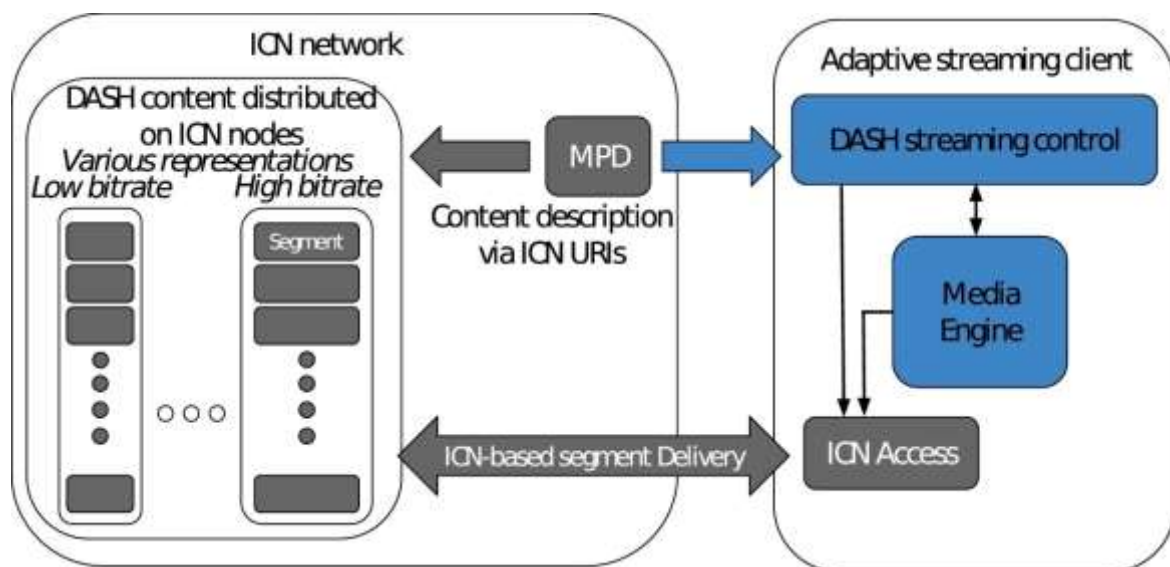
Πιο συγκεκριμένα, στο DASH τα πολυμέσα media content (για παράδειγμα ένα βίντεο) κωδικοποιείται σε διάφορες εκδοχές (representations) οι οποίες διαφέρουν μεταξύ τους ως προς το bitrate (ρυθμός μετάδοσης), την ανάλυση του βίντεο, την ποιότητα και λοιπά. Πέρα από αυτές τις εκδοχές κάθε πολυμέσο χωρίζεται σε επιμέρους τμήματα (segments) ορισμένου χρόνου (π.χ. διάρκειας του ενός δευτερολέπτου). Έτσι προκύπτει ότι κάθε τμήμα έχει και τις αντίστοιχες εκδόσεις του. Ο καταναλωτής ζητάει μεμονωμένα τμήματα χρησιμοποιώντας αιτήματα HTTP GET [16]. Ακόμα, ο καταναλωτής προσαρμόζεται στις διαφορετικές συνθήκες που μπορεί να προκύψουν στο δίκτυο και μπορεί να ζητήσει διαδοχικά τμήματα με διαφορετική ανάλυση. Αυτή η προσαρμογή από τον καταναλωτή επιτυγχάνεται με δύο τρόπους:

1. Rate Based (RB)
2. Buffer Based (BB)

Στην πρώτη περίπτωση (RB), χρησιμοποιούνται εκτιμήσεις του διαθέσιμου δικτύου για να επιλέξει ο καταναλωτής το επόμενο τμήμα με την κατάλληλη κωδικοποίηση, όπως στην περίπτωση PANDA(Probe and Adapt) [21]. Ενώ στη δεύτερη περίπτωση (BB), χρησιμοποιούνται πληροφορίες από τον buffer και ανάλογα με τον διαθέσιμο χώρο που εκτιμάται επιλέγεται τελικά πάλι το κατάλληλο τμήμα. Μία τέτοια προσέγγιση είναι η BOLA(Buffer Occupancy based Lyapunov Algorithm) [22],[23]. Οι δύο αυτές στρατηγικές μπορούν να χρησιμοποιηθούν και σε συνδυασμό σύμφωνα με τους Akhshabi et al, [24].

Το MPD (Multimedia Presentation Description) περιγράφει τις πληροφορίες των τμημάτων του media content δηλαδή το bitrate, την ανάλυση, τον χρόνο και άλλα χαρακτηριστικά. Στην DAS σε HTTP υπάρχει και το URL ενώ στην NDN αυτό το χαρακτηριστικό αντικαθίσταται από το όνομα URI (Uniform Resource Identifier) [12].

Στην παρακάτω φωτογραφία φαίνεται σχηματικά η προτεινόμενη υλοποίηση της DASH πάνω σε ICN. Τα στοιχεία της εικόνας που έχουν μπλε χρώμα αφορούν την DASH ενώ αυτά με σκούρο γκρι σχετίζονται με την ICN.



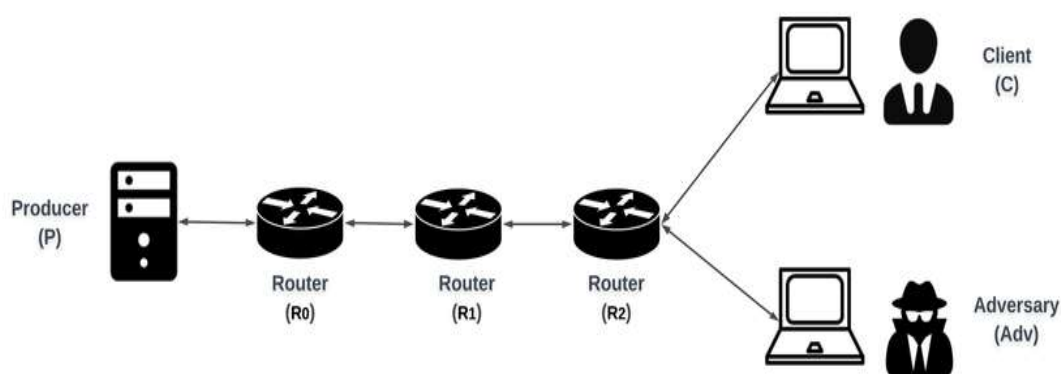
Εικόνα 5 : Dynamic adaptive multimedia streaming over ICN [29]

Συνοψίζοντας, στο DAS στην NDN κάθε πολυμέσο χωρίζεται σε τμήματα και κάθε επιμέρους τμήμα είναι διαθέσιμο σε διάφορες εκδόσεις. Αν το βίντεο αυτό είναι διαθέσιμο σε τρεις διαφορετικές ποιότητες τότε κάθε τμήμα θα χωρίζεται και σε τρεις εκδόσεις.

## 2.3 Αδυναμία στο DAS over NDN

Για γίνει πιο κατανοητή η αδυναμία και ο τρόπος με τον οποίο ο επιτιθέμενος την εκμεταλλεύεται κρίνεται απαραίτητο να απεικονιστεί ένα σενάριο τοπολογίας δικτύου. Αυτό φαίνεται στο σχήμα 5 (θα την χρησιμοποιήσουμε και στη συνέχεια αυτή τη τοπολογία). Ο πάροχος (P) περιεχομένου παράγει τα δεδομένα πολυμέσων (S). Το S χωρίζεται σε N τμήματα ίσου μεγέθους και κάθε τμήμα χωρίζεται επιμέρους σε διάφορες κωδικοποιημένες εκδόσεις που η κάθε μία αναπαριστά μία ορισμένη τιμή  $bitrate(b)$ . Δηλαδή, αν  $s(n)$  ένα τυχαίο τμήμα του βίντεο τότε  $s(n+1)$  το αμέσως επόμενο κομμάτι του βίντεο. Παρομοίως αν  $s(n,b)$  μία αναπαράσταση (ανάλυση) ενός συγκεκριμένου κομματιού (του n-οστού) τότε  $s(n,b+1)$  η αμέσως επόμενη καλύτερη διαθέσιμη ανάλυση του ίδιου κομματιού. Ο καταναλωτής (C) ζητάει τα τμήματα σε ένα από τα διαθέσιμα  $b(bitrates \text{ του } S)$ .

Το σενάριο λειτουργεί με DAS σε ICN οπότε και όλοι οι εμπλεκόμενοι, δηλαδή οι routers, ο πάροχος (P), ο επιτιθέμενος(Adv) και ο καταναλωτής(C) είναι συμβατοί με τον μηχανισμό. Οι P και C χρησιμοποιούν DAS πάνω σε ICN όπως για παράδειγμα [19] και [16] .



Εικόνα 5: Τοπολογία [9]

### 2.3.1 Προϋποθέσεις Επίθεσης

Στο σενάριο μας υπάρχουν μερικές βασικές προϋποθέσεις οι οποίες πρέπει να καλύπτονται έτσι ώστε η επίθεση να θεωρείται επιτυχημένη.

1. Ο αντίπαλος (Adv) γνωρίζει το περιεχόμενο (S) που ζήτησε ο client (C) εκ των προτέρων. Υπάρχουν αρκετές τεχνικές με τις οποίες ο Adv μπορεί να το εντοπίσει για παράδειγμα, θα μπορούσε να αξιοποιήσει επιθέσεις χρονισμού και να ανακαλύψει το MPD αρχείο [25]. Επίσης, ειδικά για περίπτωση identical wireless links, μέσω άλλης τεχνικής παρακολούθησης μπορεί να προβλέψει την τοποθεσία και την θέση του C αναλύοντας τις διαδικτυακές του δραστηριότητες [26], [27].
2. Ο Adv πρέπει να εντοπίσει τους routers που βρίσκονται πλησιέστερα στον C. Αυτό πραγματοποιείται με την τεχνική (geo-locating) που παρουσιάζεται στο [28]. Επιπλέον, πρέπει να χρησιμοποιήσει αυτούς τους routers για την επιτυχία της επίθεσης.
3. Θεωρούμε ότι το περιεχόμενο S είναι διαθέσιμο στο κοινό και ο Adv μπορεί να έχει πρόσβαση μέσω ISPs (Internet Service Providers).

### 2.3.2 Επίθεση Ταλάντωσης του Bitrate

Σε αυτό το κεφάλαιο περιγράφουμε την επίθεση ταλάντωσης BOA (Bitrate Oscillation Attack) στόχος της οποίας είναι να υποβαθμίσει την εμπειρία του καταναλωτή C κατά τη διάρκεια ενός περιεχομένου πολυμέσων.

Για να το πετύχει αυτό ο Adv εφόσον γνωρίζει το MPD αρχείο το οποίο περιλαμβάνει όλα τα κομμάτια (segment) του βίντεο με όλες τις διαθέσιμες κωδικοποιήσεις/αναλύσεις, ζητάει μη διαδοχικά τμήματα του S [19]. Εκτός αυτού, ζητάει όλες τις διαθέσιμες κωδικοποιήσεις για κάθε τέτοιο τμήμα. Αυτό συμβαίνει για να είναι σίγουρος ότι το τμήμα που θα ζητήσει μετά από λίγο ο καταναλωτής (C) θα υπάρχει στην cache του router. Τα μη διαδοχικά αυτά τμήματα έχουν ένα κενό ( $\gamma$ ) μεταξύ τους το οποίο το ορίζει ο Adv έτσι ώστε κάθε επόμενο interest που εκδίδει να αφορά το τμήμα  $s(n+\gamma)$ , παρακάμπτοντας έτσι έναν αριθμό  $\gamma$  τμημάτων. Αυτό το τμήμα θα αποθηκεύεται στην cache του router και όταν το ζητήσει το θύμα θα

υπάρχει εκεί με αποτέλεσμα να ικανοποιηθεί από τον router. Συνολικά το θύμα θα ζητήσει όλα τα τμήματα του S με την σειρά ή αλλιώς θα ζητάει διαδοχικά, μετά το  $s(n)$  θα ζητήσει το  $s(n+1)$  για να δει όλο το βίντεο. Έτσι, εφόσον έχει ενεργήσει πριν ο Adv το θύμα ενώ θα καταναλώνει το περιεχόμενο (πχ θα βλέπει το βίντεο ) είτε θα βρίσκει κάποιο τμήμα στην cache του router (κάποιο από τα  $s(n+\gamma)$ ) και θα ικανοποιείται από εκεί, είτε θα πηγαίνει πιο μακριά το Interest του για να ικανοποιηθεί, πιθανώς και ως την πηγή (P).

Με βάση το παραπάνω φαινόμενο όταν ο C καταναλώσει ένα από τα  $s(n+\gamma)$  θα ερμηνεύσει εξαιρετικά γρήγορους χρόνους παράδοσης με αποτέλεσμα να αξιολογήσει ότι το διαθέσιμο εύρος ζώνης (bandwidth) είναι αρκετό για να ζητήσει την επόμενη φορά ένα τμήμα με καλύτερη ανάλυση ( ίσως με την υψηλότερη ανάλυση). Τα επόμενα όμως τμήματα μετά το  $s(n+\gamma)$  δεν υπάρχουν στην cache του router με αποτέλεσμα να ικανοποιηθούν από τον P και να έχουν εξαιρετικά μεγάλο χρόνο παράδοσης. Επομένως, ο C θα αξιολογήσει ότι πρέπει να ζητήσει το επόμενο πακέτο με πολύ χαμηλότερη ανάλυση. Όλη αυτή η διαδικασία αναγκάζει τον μηχανισμό DAS να εναλλάσσεται δυναμικά μεταξύ υψηλών και χαμηλών αναπαραστάσεων που εν τέλη προκαλεί ταλαντώσεις στην ποιότητα του περιεχομένου που καταναλώνει ο χρήστης. Αυτές ταλαντώσεις είναι που υποβαθμίζουν την τελική εμπειρία του χρήστη.

## 2.4 Reputation-based trust Scheme Design

Σε αυτό το υπο-κεφάλαιο θα αναλύσουμε τον προτεινόμενο μηχανισμό φήμης (reputation-based trust) που χρησιμοποιούμε για τον μετριάσμό των επιθέσεων ταλάντωσης (BOA). Αυτός ο στόχος επιτυγχάνεται μέσω του ελέγχου και του περιορισμού της προσωρινής αποθήκευσης από τους routers. Προσθέτοντας ένα σύστημα ελέγχου και επιλεκτικής αποθήκευσης σε κάθε router. Πριν προχωρήσουμε με περισσότερες λεπτομέρειες στον σχεδιασμό άμυνας πρέπει να σημειώσουμε ότι όταν αναφερόμαστε στο περιεχόμενο ή στο αντικείμενο περιεχομένου εννοούμε ολόκληρο το βίντεο (δηλαδή το multimedia content) και όταν αναφερόμαστε στα segments εννοούμε τα τμήματα στα οποία είναι χωρισμένο το βίντεο και στο σύνολό τους αποτελούν το βίντεο. Κάθε segment συνιστά ένα πακέτο δεδομένων και κάθε interest αναφέρεται σε ένα συγκεκριμένο segment (πακέτο δεδομένων). Πιο συγκεκριμένα, οι routers υπολογίζουν και εκχωρούν μια τιμή φήμης σε κάθε

αντικείμενο περιεχομένου με βάση το αν τα κομμάτια (segments) του ζητούνται από τον χρήστη διαδοχικά δηλαδή με τη σειρά και χωρίς «κενά». Με βάση αυτή την τιμή φήμης αποφασίζουν αν θα αποθηκεύσουν στην cache τους τα τμήματα του βίντεο. Χαρακτηρίζουν δηλαδή ολόκληρο το βίντεο το οποίο με τη σειρά του χαρακτηρίζει έμμεσα τον χρήστη που το ζητάει. Αργότερα, όταν ένας χρήστης ζητήσει το ίδιο περιεχόμενο αλλά με διαδοχικά interest που αναφέρονται σε διαδοχικά τμήματα του βίντεο θα χαρακτηριστεί από τον router ως καλόβουλος χρήστης εφόσον το περιεχόμενο που ζητάει αποθηκεύεται στην cache.

Η διαδικασία της αξιολόγησης ξεκινάει με την παραλαβή του MPD αρχείου του εκάστοτε βίντεο από τον router. Το MPD αρχείο όπως περιγράψαμε και παραπάνω περιέχει όλες τις εκδόσεις/αναπαραστάσεις όλων των τμημάτων ενός multimedia περιεχομένου και είναι το πρώτο πακέτο δεδομένων που ζητάει ο χρήστης έτσι ώστε να επιλέγει στη συνέχεια από αυτό το κατάλληλο τμήμα στην κατάλληλη - ανάλογα με την διαθεσιμότητα του δικτύου του- έκδοση. Κατά την άφιξη του MPD αρχείου στον router ο οποίος στην συνέχεια θα το προωθήσει στον χρήστη, πραγματοποιείται ένας έλεγχος για να διαπιστωθεί αν είναι η πρώτη φορά που λαμβάνεται το συγκεκριμένο περιεχόμενο στο οποίο αναφέρεται το MPD. Αν είναι όντως ο router του εκχωρεί μια προεπιλεγμένη τιμή φήμης την οποία θα ονομάζουμε reputation value. Γίνεται έτσι αντιληπτό ότι οι routers κρατούν αρχείο με τα περιεχόμενα και τις αντίστοιχες reputation values. Στη συνέχεια και μετά την λήψη του MPD ο καταναλωτής ζητάει πακέτα δεδομένων που αναφέρονται σε τμήματα του βίντεο. Κάθε τέτοιο πακέτο που περνάει από τον router ελέγχεται για το αν πρόκειται για διαδοχικό ή μη διαδοχικό τμήμα συγκρίνοντας τους αριθμούς τμήματος του πακέτου με το αμέσως προηγούμενο πακέτο που πέρασε από τον router και αναφερόταν στο ίδιο περιεχόμενο. Ο router αξιολογεί αρνητικά αν διαπιστώσει ότι υπάρχει κενό και θετικά στην αντίθετη περίπτωση. Η τιμή φήμης όμως ενημερώνεται μετά από ένα ορισμένο χρονικό διάστημα το οποίο συνήθως περιέχει παραπάνω από μία αξιολογήσεις του router. Η ενημερωμένη τιμή φήμης υπολογίζεται με βάση των συγκεντρωμένων αυτών αξιολογήσεων αλλά και της προηγούμενης τιμής φήμης (που ήταν καταχωρημένη για το συγκεκριμένο περιεχόμενο). Όσο λοιπόν ένας χρήστης καταναλώνει το περιεχόμενο η τιμή φήμης του συνεχώς αλλάζει, είτε βελτιώνεται είτε επιδεινώνεται. Ο router τελικά αποφασίζει να αποθηκεύσει προσωρινά τα επόμενα τμήματα αυτού του περιεχομένου βασιζόμενος στις παραγόμενες αυτές τιμές που ονομάζουμε reputation values.

Τέλος επισημαίνεται ότι η προσέγγιση που προτείνεται αφορά routers που υπολογίζουν και αποθηκεύουν ατομικές τιμές χωρίς να επικοινωνούν με άλλους routers του δικτύου. Έτσι, το σύστημα ονομάζεται αποκεντρωμένο και δεν απαιτείται ο συντονισμός μεταξύ των routers. Αυτό συμβαίνει επειδή ένα πρωτόκολλο ανταλλαγής που απαιτείται σε ένα κεντρικό πλαίσιο θα πρόσθετε επιπλέον ανεπιθύμητη επιβάρυνση [3].

## Κεφάλαιο 3ο : Εργαλεία

---

Σε αυτό το κεφάλαιο θα παρουσιαστούν εν συντομία τα βασικά εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας. Ο προσομοιωτής που μας έδωσε τη δυνατότητα να υλοποιήσουμε ένα δίκτυο NDN αρχιτεκτονικής ονομάζεται ndnSIM, είναι προσομοιωτής ανοιχτού κώδικα και βασίζεται στο framework προσομοίωσης ns-3.

### 3.1 Προσομοιωτής δικτύων ns-3

Ο προσομοιωτής δικτύων ns-3 στοχεύει κυρίως στην έρευνα και εκπαιδευτική χρήση, είναι ένας προσομοιωτής ανοιχτού κώδικα διαθέσιμος στο κοινό για έρευνα και ανάπτυξη και αποτελεί την βελτιωμένη έκδοση του προσομοιωτή ns-2. Ο ns-3 διατηρείται από μια παγκόσμια ομάδα εθελοντών συντηρητών και αναπτύσσεται συνεχώς για να πληροί τις ανάγκες της ερευνητικής κοινότητας στην σύγχρονη προσομοίωση δικτύων. Επιπλέον, καλύπτει ολοκληρωμένα την προσομοίωση από την σχεδίαση μέχρι τη συλλογή και ανάλυση των δεδομένων [31]. Είναι γραμμένος σε C++ και επιτρέπει στον χρήστη να εκτελεί άμεσες εντολές σε C++ ενώ δέχεται και Python scripts. Σε σύγκριση με τον ns-2 που χρησιμοποιούσε Otel στον ns-3 υπάρχουν περισσότερα προνόμια στους ερευνητές όπως η ύπαρξη έξυπνων δεικτών, προτύπων, τυχαίων μεταβλητών επανακλήσεων κλάσεων και αντικειμένων [30]. Επιπρόσθετα, χάρη στην C++ διευκολύνεται κατά πολύ το debugging ενώ επιτρέπεται και η ενσωμάτωση κώδικα σε προγραμματιστικά περιβάλλοντα που είναι βασισμένα σε Python με το Python scripting.

Για την κατανόηση της λειτουργίας της προσομοίωσης στον ns-3 πρέπει να αναλύσουμε τα ακόλουθα στοιχεία στα οποία βασίζεται η προσομοίωση. Όπως συμβαίνει και σε άλλα εργαλεία προσομοίωσης δικτύου έτσι και ο ns-3 διαθέτει μοντέλα για όλα αυτά τα διαφορετικά στοιχεία.

1. Κόμβος (Node): η βασική οντότητα που χαρακτηρίζει κυρίως απλούς υπολογιστές και routers, αλλά σε ορισμένες περιπτώσεις μπορεί να αναφέρεται σε hubs ή switches. Προγραμματίζεται από τον χρήστη.
2. Κανάλι (Channel): μια οντότητα που συνδέει δύο κόμβους (π.χ. point-to-point συνδέσεις με οπτικές ίνες, wi-fi κλπ). Αντιπροσωπεύουν το μέσο που

χρησιμοποιείται για την αποστολή και παραλαβή πληροφοριών μεταξύ των συσκευών του δικτύου.

3. Πρωτόκολλα επικοινωνιών (communication protocols), τα οποία μοντελοποιούν την εφαρμογή του πρωτοκόλλου. Αυτά τα αντικείμενα πρωτοκόλλου συνήθως ομαδοποιούνται σε μία στοίβα πρωτοκόλλου όπου κάθε στρώμα στην στοίβα εκτελεί μία ορισμένη λειτουργία στα πακέτα δικτύου και στη συνέχεια το πακέτο περνάει σε ένα άλλο στρώμα για πρόσθετη επεξεργασία.
4. Κεφαλίδες πρωτοκόλλου (protocol headers), πρόκειται για υποσύνολα των πληροφοριών που βρίσκονται στα πακέτα δικτύου και έχουν συγκεκριμένη μορφή.
5. Συσκευές δικτύου (Net Devices): αντιπροσωπεύουν τόσο το προσομοιωμένο υλικό όσο και το λογισμικό που επιτρέπει σε έναν κόμβο να επικοινωνεί μέσω καναλιών με άλλους κόμβους (όπως μια απλή κάρτα διεπαφής δικτύου Ethernet ή μια πιο σύνθετη ασύρματη συσκευή).
6. Πακέτα δικτύου (Network packets). Αποτελούν την πλέον θεμελιώδη μονάδα ανταλλαγής πληροφοριών στα δίκτυα υπολογιστών. Ένα πακέτο δεδομένων περιέχει εκτός από την πληροφορία που θέλει να μεταφέρει (πχ ένα κομμάτι βίντεο που ζητάει ο χρήστης) και άλλου είδους πληροφορίες στις κεφαλίδες πρωτοκόλλου. Αυτές περιγράφουν τις πληροφορίες που απαιτούνται από την εκάστοτε εφαρμογή πρωτοκόλλου για την μεταφορά μέσα στο δίκτυο [32], [33].

Ο χρήστης κατά την υλοποίηση της προσομοίωσης γράφει ένα πρόγραμμα σε C++ το οποίο βασίζεται στα τέσσερα βασικά βήματα που ορίζουμε παρακάτω:

1. Δημιουργία της τοπολογίας δικτύου. Αυτή περιλαμβάνει τη δημιουργία των κόμβων, των συσκευών και των καναλιών.
2. Δημιουργία της ζήτησης δεδομένων στο δίκτυο. Πιο συγκεκριμένα, προσομοιώνονται εφαρμογές δικτύου οι οποίες εγκαθίστανται στους κόμβους και ορίζουν την αλληλεπίδραση τους με το δίκτυο.
3. Εκτέλεση της προσομοίωσης. Αυτό συμβαίνει όταν η προσομοίωση φτάσει στον κύριο βρόγχο συμβάντος. Η προσομοίωση τελειώνει όταν εκτελεστεί και το τελευταίο συμβάν και η λίστα συμβάντων μείνει κενή ή όταν έρθει το

πέρασ του χρόνου που ορίζεται από τον χρήστη ως χρόνος τερματισμού της προσομοίωσης.

4. Ανάλυση αποτελεσμάτων. Οι πληροφορίες που συλλέγονται από τα trace files (τα οποία παράγονται από την εκτέλεση της προσομοίωσης) χρησιμοποιούνται στην στατιστική ανάλυση και στην εξαγωγή συμπερασμάτων. Το tracing system είναι μια πολύ σημαντική υποδομή που στόχο έχει την καταγραφή δεδομένων κατά τη διάρκεια της προσομοίωσης. Μπορεί να γίνει καταγραφή δεδομένων σε οποιοδήποτε στοιχείο της προσομοίωσης και να δημιουργηθούν παραγόμενα αρχεία σε διάφορες μορφές ανάλογα με τις πληροφορίες που θεωρεί σημαντικές ο χρήστης [33].

### **3.2 Προσομοιωτής δικτύων ndnSIM**

Για τις ανάγκες προσομοίωσης δικτύων NDN αναπτύχθηκε ο προσομοιωτής που ονομάζουμε ndnSIM. Οι προδιαγραφές που τέθηκαν ήταν να παρέχει διαλειτουργικότητα ανάμεσα στις προσομοιώσεις και την προτυποποίηση, να επιτρέπει την αμφίδρομη πορεία μεταξύ πειραμάτων και συμπερασμάτων και να εξασφαλίζει αξιόπιστες προσομοιώσεις [26].

Ο ndnSIM είναι ένας προσομοιωτής δικτύου ανοικτού κώδικα που βασίζεται στον προσομοιωτή ns-3. Αυτό σημαίνει πως ο ndnSIM χρησιμοποιεί ελεύθερα και έχει ως θεμέλια τα στοιχεία του ns-3, δηλαδή όλα τα τμήματα, τα μοντέλα και τους ενσωματωμένους κώδικες. Επομένως, ακολουθώντας τη λογική του ns-3, διαθέτει τα μέσα για να δημιουργεί και να διαμορφώνει τοπολογίες δικτύων, να προσομοιώνει την κίνηση ανάμεσα στους κόμβους του δικτύου και διάφορα πρωτοκόλλα σε επίπεδο συνδέσεων, να επιλέγει την εικονική υλοποίηση για την παρακολούθηση της προσομοίωσης καθώς και να καταγράφει τα δεδομένα των προσομοιώσεων

Επίσης, χρησιμοποιεί τον Named Forwarding Daemon (NFD) για την προσομοίωση των βασικών δομών και των αντίστοιχων λειτουργιών του NDN, καθώς και σχετικά πρωτόκολλα. Τέλος, ο ndnSIM παρέχει εφαρμογές που προσομοιώνουν τους καταναλωτές και τους παραγωγούς.

### 3.3 Amust framework (Adaptive Multimedia Streaming)

Ο προσομοιωτής που χρησιμοποιούμε και στον οποίο λαμβάνουν χώρα τα πειράματά μας για να αξιολογήσουμε τα παραπάνω μοντέλα είναι ο AMuSt-ndnSIM [34], ο οποίος αποτελεί μια τροποποιημένη έκδοση του ndnSIM [32] στην οποία προσαρμόστηκε το λογισμικό Libdash [41] (βιβλιοθήκη ανοιχτού κώδικα του προτύπου MPEG-DASH). Στην δικιά μας εργασία το χρησιμοποιούμε επειδή προσομοιώνει το DAS σε δίκτυα NDN επεκτείνοντας τον ndnSIM. Ακόμα, παρέχει αντίστοιχες εφαρμογές multimedia consumer και file server.

## Κεφάλαιο 4ο : Μεθοδολογία

---

Για τον έλεγχο του αμυντικού μηχανισμού σε επιθέσεις ταλάντωσης σε δίκτυα NDN και της βέλτιστης εκδοχής του, εκτελέστηκαν πειράματα προσομοίωσης με διαφορετικούς τρόπους ανίχνευσης του επιτιθέμενου. Πιο συγκεκριμένα, η διαδικασία ενημέρωσης της μεταβλητής “reputation value”, η οποία καθορίζει την κρίση του μηχανισμού ως προς την επικινδυνότητα του περιεχομένου και τελικά την απόφαση για την αποθήκευση του στην μνήμη cache του router. Όπως αναλύθηκε στο 3ο κεφάλαιο, η υλοποίηση γίνεται στο περιβάλλον του προσομοιωτή δικτύων ndnSIM, τα αποτελέσματα καταγράφονται σε trace files και η επεξεργασία τους γίνεται σε περιβάλλον excel του office.

Ο μηχανισμός άμυνας λαμβάνει χώρα σε κάθε router του δικτύου και για την διαδικασία ενημέρωσης της τιμής φήμης μελετάμε τρία διαφορετικά μοντέλα και ελέγχουμε πιο μετριάζει αποτελεσματικότερα την επίθεση ταλάντωσης που προσομοιώνουμε. Ωστόσο, αξίζει να προσδιορίσουμε εδώ το πλαίσιο πάνω στο οποίο εφαρμόζουμε το κάθε μοντέλο. Κάθε φορά που φτάνει ένα πακέτο δεδομένων στον router ελέγχεται και βαθμολογείται είτε θετικά είτε αρνητικά η συμπεριφορά του. Η reputation value όμως δεν ενημερώνεται μετά από κάθε τέτοια αξιολόγηση. Η τιμή φήμης (reputation value) ενημερώνεται περιοδικά με προκαθορισμένη περίοδο η οποία αντιστοιχεί σε χρονικό διάστημα κατά το οποίο λαμβάνουν χώρα παραπάνω από μία αξιολογήσεις. Οι αξιολογήσεις αυτές συγκεντρώνονται και σε κάθε τέτοια περίοδο ενημερώνουν την τιμή φήμης η οποία συγκρίνεται με ένα κατώφλι (threshold). Αν η ενημερωμένη τιμή φήμης ξεπερνάει το κατώφλι ( $\text{reputation value} > \text{threshold}$ ) ο router αποφασίζει να αποθηκεύσει προσωρινά το επόμενο πακέτο που θα φτάσει. Μέχρι την επόμενη ενημέρωση ο router δεν αλλάζει πολιτική και αποθηκεύει το κάθε πακέτο. Σε αντίθετη περίπτωση όπου η τιμή φήμης είναι μικρότερη από το κατώφλι ( $\text{reputation value} < \text{threshold}$ ) ο router δεν αποθηκεύει τα εισερχόμενα πακέτα δεδομένων μέχρι την επόμενη ενημέρωση.

## 4.1 Μοντέλο 1

Στο πρώτο μοντέλο για την διαδικασία ενημέρωσης της τιμής φήμης (reputation value update) υιοθετήσαμε το πρωτόκολλο που περιγράφεται στην [28], μαζί με όποιες απαραίτητες προσαρμογές παρουσιάζονται στην [9] έτσι ώστε να καταστεί κατάλληλη η διαδικασία ενημέρωσης για τον μετριάσμο επιθέσεων BOA. Πιο συγκεκριμένα, σε αντίθεση με το [28] αναθέτουμε τις τιμές φήμης στο ίδιο το περιεχόμενο αντί για να τις αναθέτουμε στους routers με αυτόν το τρόπο καθιστούμε τον μηχανισμό άτρωτο στις επιθέσεις Sybil. Επιπλέον, τρεις τροποποιημένες μετρήσεις αντικαθιστούν τις μετρήσεις που παρουσιάζονται στο [28] και αυτές είναι το skipping ratio το rating και το reputation value. Ας δούμε όμως αναλυτικά τον ρόλο της καθεμίας.

**Skipping ratio:** Ορίζουμε τον λόγο των συνολικών κενών που έχουμε ανιχνεύσει μέχρι στιγμής προς τον αριθμό του τελευταίου τμήματος (segment) που έχει λάβει ο router. Η μεταβλητή skipping ratio μπορεί να πάρει τιμές από 0 έως 0,5 και εκφράζει την συχνότητα των κενών που εντοπίστηκαν. Αν ο επιτιθέμενος δεν στείλει τρία διαδοχικά τμήματα του βίντεο σημαίνει ότι εντοπίσαμε ένα κενό. Ο αριθμός των κενών είναι προφανώς ανεξάρτητος από το μέγεθος των κενών. Ο παρονομαστής μας επιτρέπει τελικά να εκφράσουμε (στο skipping ratio) την συχνότητα των κενών στο βίντεο. Με αυτό το τρόπο αποφεύγουμε τυχόν συμπεριφορές απώλειας της σύνδεσης στο δίκτυο ενός μη κακόβουλου χρήστη να βαθμολογηθούν αρνητικά. Αυτό συμβαίνει γιατί ένα μόνο κενό κατά τη διάρκεια ενός βίντεο δεν θα είναι ικανό να δώσει μεγάλη τιμή στο skipping ratio. Αν η απώλεια της σύνδεσης στο δίκτυο συμβεί στην αρχή του βίντεο η τιμή του skipping ratio θα μειώνεται όσο καταναλώνεται το περιεχόμενο.

Ο παρονομαστής εκφράζει ουσιαστικά τον συνολικό αριθμό τμημάτων που θα έπρεπε να είχαν ληφθεί αν δεν υπήρχαν κενά.

$$SR(C_i) = \frac{G(C_i)}{L(C_i)}$$

Όπου  $C_i$ : δηλώνει συγκεκριμένο αντικείμενο περιεχομένου(δλδ ολόκληρο το βίντεο)

$G(C_i)$ : ο αριθμός των κενών που εντοπίστηκαν μέχρι στιγμής

$L(C_i)$ : ο αριθμός του τελευταίου τμήματος που ελήφθη ή αλλιώς τα τμήματα που θα έπρεπε να έχουν ληφθεί υπό κανονικές συνθήκες.

**Rating:** Όταν ο router λαμβάνει ένα νέο πακέτο δεδομένων υπάρχουν δύο σενάρια. Όταν ο αριθμός τμήματος του λαμβανόμενου πακέτου είναι ο αναμενόμενος, δηλαδή κατά ένα μεγαλύτερος του αμέσως προηγούμενου έχουμε το θετικό σενάριο. Όταν ο αριθμός τμήματος δεν είναι διαδοχικός, είναι δηλαδή μεγαλύτερος από τον αναμενόμενο έχουμε το αρνητικό σενάριο που σημαίνει ότι κάποια τμήματα έχουν παραλειφθεί.

Όπως και στην [28] οι routers βαθμολογούν αυστηρά κάθε αρνητικό συμβάν συγκριτικά με ένα θετικό, ακολουθούν κατά κανόνα την αρχή γραμμική αύξηση, εκθετική μείωση.

$$R(C_i) = \begin{cases} 1, & \text{αν τα τμήματα είναι διαδοχικά} \\ -\theta e^a(1 - SR(C_i)), & \text{αν εντοπιστεί κενό} \end{cases}$$

Όπου  $\theta$ : παράγοντας ποινής που αντιπροσωπεύει την ισχύ της τιμωρίας

$a$ : παράγοντας προσαρμογής

Όπως παρατηρείτε, το **Skipping ratio** περιλαμβάνεται στο **Rating** όταν έχουμε ένα αρνητικό συμβάν ή αλλιώς όταν ανιχνεύσουμε μία κακόβουλη συμπεριφορά. Τελικά, το υπολογισμένο **Rating** χρησιμοποιείται στον υπολογισμό της ενημερωμένης τιμής φήμης (**reputation value**), αυξάνοντας την αν δεν έχουν εντοπιστεί κενά και μειώνοντας την διαφορετικά.

**Reputation value:** Η ενημερωμένη τιμή φήμης αντιπροσωπεύει τελικά την κρίση του μηχανισμού άμυνας για την αξιοπιστία ενός συγκεκριμένου αντικειμένου περιεχομένου. Πιο αναλυτικά, μία χαμηλή τιμή φήμης αντιπροσωπεύει μία κακόβουλη συμπεριφορά ενώ μία υψηλή τιμή φήμης σημαίνει ότι ο χρήστης ζητάει διαδοχικά τμήματα και δεν έχει σκοπό να παραπλανήσει τον router.

$$RV(t) = \delta RV(t-1) + \sum_{k=1}^n R(C_i)$$

Όπου  $n$ : ο αριθμός των τμημάτων που ελήφθησαν στο χρονικό διάστημα  $[t-1, t]$ .

Στον χρόνο  $t-1$  πραγματοποιήθηκε η προηγούμενη ενημέρωση της τιμής φήμης.

$\delta$ : αντιπροσωπεύει το ειδικό βάρος που δίνουμε σε παρελθούσες μετρήσεις. Όσο μεγαλύτερη η τιμή του  $\delta$  τόσο λιγότερη μεταβολή του reputation value σε κάθε ενημέρωση και όσο μικρότερη η τιμή του  $\delta$  τόσο μικρότερη επιρροή της παλιάς συμπεριφοράς και τόσο μεγαλύτερη μεταβολή σε κάθε ενημέρωση.

Τελικά ο κάθε router αποφασίζει αν θα αποθηκεύσει προσωρινά το εκάστοτε πακέτο βασισμένος στην τρέχουσα τιμή του RV.

## 4.2 Μοντέλο 2

Σε αυτή τη προσέγγιση υιοθετήσαμε τον μηχανισμό ενημέρωσης της τιμής φήμης που παρουσιάζεται στην [37], εξαιρώντας τις τιμές φήμης που αναθέτει σε κάθε router.

Η [35] προτείνει μία τροποποιημένη μορφή της μεθόδου Bayesian που παρουσιάζεται στις [35],[36]. Η τελική τιμή φήμης προκύπτει από δύο επιμέρους μετρήσεις. Στην πρώτη από αυτές ακολουθούμε την τροποποιημένη μορφή της μεθόδου Bayesian, που για κάθε θετική συμπεριφορά βαθμολογούμε με  $s=0$  και για κάθε αρνητική με  $s=1$ . Η μέθοδος Bayesian δίνει το ίδιο βάρος σε κάθε παρατήρηση ανεξάρτητα από το πότε ανιχνεύσαμε αυτή την παρατήρηση χρονικά. Για να ξεφύγουμε από αυτό και να δώσουμε λιγότερο βάρος σε παρατηρήσεις που έγιναν στο παρελθόν προσθέτουμε έναν κινούμενο σταθμισμένο μέσο όρο  $u$  όπως αναπτύχθηκε στην [37]. Τελικά, οι μεταβλητές που ενημερώνουμε σε αυτή τη πρώτη μέτρηση είναι οι  $a$  και  $b$ .

$$a = u * a + s$$

$$b = u * b + (1 - s)$$

Όπου  $u = 1 - 1/m$

Η τιμή του  $u$  καθορίζει την σημασία που δίνει ο μηχανισμός σε παλαιότερες συμπεριφορές, πιο συγκεκριμένα αν το  $m$  ήταν ακέραιος, τότε μετά από  $m$  παρατηρήσεις θα είχαμε την ίδια ουρά με τα αποτελέσματα μετά από πολλές παρατηρήσεις. Έτσι, μπορούμε να επιλέξουμε τον αριθμό των αμέσως προηγούμενων παρατηρήσεων για τις οποίες πιστεύουμε ότι έχει νόημα να περιλάβουμε στην τιμή

φήμης. Δηλαδή, αν θέλουμε να λαμβάνουμε υπόψη μέχρι 15 παρατηρήσεις πίσω θα επιλέξουμε  $m = 15$  οπότε και  $u = 0,93$ .

Μπορούμε ακόμα να πούμε ότι το  $\alpha$  δηλώνει πόσο γενικά κακός είναι ο χρήστης και το  $\beta$  πόσο γενικά καλός είναι. Προκύπτει από τον τύπο ότι σε μία θετική συμπεριφορά ( $s=0$ ) το  $\alpha$  θα μειωθεί, αφού  $\alpha = u \cdot \alpha + 0$ , και το  $\beta$  θα αυξηθεί αντίστοιχα.

Τελικά υπολογίζω το reputation value με τα  $\alpha$  και  $\beta$  ως εξής:

$$reputation\ value = E(Beta(a, b))$$

Όπου  $E()$ : η αναμενόμενη τιμή

$Beta()$ : η συνάρτηση βήτα

Αποδεικνύεται όμως ότι η αναμενόμενη τιμή της συνάρτησης βήτα δύο τιμών  $\alpha, \beta > 0$  είναι:

$$E(Beta(a, b)) = \frac{\alpha}{\alpha + \beta}$$

Έτσι έχουμε τελικά :

$$reputation\ value = \frac{\alpha}{\alpha + \beta}$$

Έτσι το reputation value θα παίρνει τιμές από 0 έως 1. Οι τιμές φήμης κοντά στο 1 αντιστοιχούν σε χειρότερες συμπεριφορές. Ένας καταναλωτής με καλή συμπεριφορά που ζητάει διαδοχικά τμήματα θα έχει πάντα reputation value = 0. Επειδή όμως στα υπόλοιπα μοντέλα ισχύει το αντίστροφο (δηλαδή για μεγαλύτερο reputation value έχουμε κακόβουλη συμπεριφορά) και θέλουμε να κρατήσουμε μια ομοιομορφία, θα αντιστρέψουμε αυτή την συσχέτιση έτσι ώστε και εδώ να ισχύει το ίδιο. Αυτό το πετυχαίνουμε με την εντολή:

$$reputation\ value = 1 - reputation\ value$$

### 4.3 Μοντέλο 3

Η τρίτη διαφορετική ενημέρωση της τιμής φήμης που θα μελετήσουμε στηρίζεται στην [40] η οποία παρουσιάζει μια τροποποιημένη εκδοχή των [38], [39]. Αυτό το μοντέλο είναι σχεδιασμένο ειδικά για επιθέσεις on/off καθώς εισάγει έναν παράγοντα  $f$  ο οποίος δίνει ειδικό βάρος στην παλιά συμπεριφορά του χρήστη ανάλογα με το πώς τον έχει βαθμολογήσει μέχρι στιγμής σύμφωνα με την [38]. Δηλαδή, αν η τιμή φήμης είναι τέτοια ώστε το αμυντικό σύστημα να χαρακτηρίζει τον χρήστη κακόβουλο τότε το  $f$  παίρνει τέτοια τιμή ώστε να διαμορφώνει το αμυντικό μας σύστημα μία πιο καχύποπτη πολιτική απέναντι στον συγκεκριμένο χρήστη. Αυτό το πετυχαίνει με το να μεταφέρουμε μεγαλύτερο βάρος στην προηγούμενη κακόβουλη συμπεριφορά του. Το  $f$  λοιπόν, είναι ένας παράγοντας που αλλάζει τιμή δίνοντας περισσότερο βάρος σε παλαιότερη ή νεότερη συμπεριφορά ανάλογα με την μέχρι στιγμής συμπεριφορά του χρήστη. Έτσι τελικά θα είναι δύσκολο ένας χρήστης με κακόβουλη συμπεριφορά να μας παραπλανήσει με μερικά διαδοχικά πακέτα. Αν τώρα η συμπεριφορά του είναι θετική τότε θα δίνεται μεγαλύτερο βάρος σε κάθε νέα αξιολόγηση και με αυτό το τρόπο κάθε αρνητική συμπεριφορά θα είναι ικανή να ρίξει την τιμή φήμης σε χαμηλά επίπεδα τέτοια ώστε να χαρακτηριστεί ο χρήστης κακόβουλος και να μην επιτρέπεται η προσωρινή αποθήκευση περιεχομένου από τον router.

Με κάθε ενημέρωση της τιμής φήμης ενημερώνεται και παράγοντας  $f$  σύμφωνα με την παρακάτω σχέση:

$$f = \begin{cases} 1 - e & , \quad \text{αν reputation value} \leq \text{threshold} \\ e & , \quad \text{αν reputation value} > \text{threshold} \end{cases}$$

Όπου  $e$  παίρνει τιμές από 0 έως 0,5.

Σε κάθε νέο πακέτο που έρχεται στον router ενημερώνονται δύο τιμές:

$$F(N_c(t)) = F(N_c(t-1)) * f + N_c(t) - N_c(t-1)$$

$$F(N_p(t)) = F(N_p(t-1)) * f + N_p(t) - N_p(t-1)$$

Όπου  $N_c(t)$ : ο συνολικός αριθμός πακέτων που έλαβε ο router μέχρι τη χρονική στιγμή  $t$

$N_p(t)$ : ο συνολικός αριθμός πακέτων που δεν έλαβε ο router και θα έπρεπε να είχε λάβει. Με άλλα λόγια πακέτα που παραλήφθηκαν.

Τελικά η τιμή φήμης υπολογίζεται από τη σχέση :

$$reputation\ value = \frac{F(N_c(t))}{F(N_c(t)) + F(N_p(t))}$$

Η τελευταία σχέση εκφράζει το ποσοστό των πακέτων που έλαβε ο router προς τον συνολικό αριθμό των πακέτων που θα έπρεπε να είχε λάβει αν ζητούσε όλα τα πακέτα διαδοχικά, χωρίς κενά. Επιπλέον, δίνεται ένα βάρος όπως εξηγήσαμε παραπάνω σε παρελθούσες συμπεριφορές με τον παράγοντα  $f$  (forgetting factor).

Η τιμή φήμης όπως αναφέραμε και παραπάνω σε κάθε ενημέρωση της συγκρίνεται με το κατώφλι και ενημερώνει και την τιμή του  $f$  όπως

#### 4.4 Γενικές Παράμετροι – Simulation Setup

Το σενάριο τοπολογίας φαίνεται στην εικόνα 5 και όπως περιγράψαμε και παραπάνω αποτελείται από τρεις routers, έναν πάροχο(P), έναν επιτιθέμενο(Adv) και έναν καταναλωτή(C). Το multimedia content που επιλέξαμε σε αυτή την εργασία είναι το Big Buck Bunny clip from the DASH/AVC Dataset το οποίο είναι κωδικοποιημένο σε AVC [42].Ο καταναλωτής (C) για να επιλέξει το τμήμα με την κατάλληλη κωδικοποίηση χρησιμοποιεί εκτιμήσεις του διαθέσιμου δικτύου RB. Όπως περιγράψαμε και στην ενότητα 2.2 υπολογίζοντας το throughput. Οι routers χρησιμοποιούν τη στρατηγική προώθησης του BestRoute. Αυτό σημαίνει ότι προωθούν τα πακέτα Interest με το χαμηλότερο κόστος δρομολόγησης. Ακόμη οι routers χρησιμοποιούν LRU για την ενημέρωση των καταχωρήσεων της δομής Content Store.

Για να αυξήσουμε την δυσκολία ανίχνευσης του επιτιθέμενου χρησιμοποιούμε μια υβριδική συμπεριφορά του επιτιθέμενου κατά την οποία θα συμπεριφέρεται για ένα ορισμένο χρονικό διάστημα φυσιολογικά και για ένα αντίστοιχο διάστημα κακόβουλα. Ορίζουμε αυτό το διάστημα σε 250 sec. Η θετική συμπεριφορά θα διαδέχεται την αρνητική (για το σύστημά μας) συμπεριφορά και αντίστροφα. Κατά την διάρκεια της κακόβουλης συμπεριφοράς θα ζητάει μη διαδοχικά τμήματα με κενό  $\gamma$ . Το  $\gamma$  έχει οριστεί ίσο με 5. Δηλαδή, κατά την διάρκεια αυτής της συμπεριφοράς θα ζητάει το τμήμα  $s(n)$  και στη συνέχεια το τμήμα  $s(n+\gamma)$  (σε όλες τις διαθέσιμες κωδικοποιήσεις) παρακάμπτοντας 5 τμήματα. Αυτές οι επιθέσεις ονομάζονται on-off attacks και είναι πιο ρεαλιστικές. Στον παρακάτω πίνακα περιλαμβάνονται όλες οι ρυθμίσεις παραμέτρων που χρησιμοποιήσαμε στις προσομοιώσεις μας.

TABLE I  
SIMULATION CONFIGURATIONS

| Parameter                                   | Value         |
|---------------------------------------------|---------------|
| Link Bandwidth                              | 10 Mbps       |
| Point-to-point link delay                   | 2 ms          |
| Link between Producer and edge router delay | 350 ms        |
| Content Store size                          | 0 (unlimited) |
| Cache replacement policy                    | LRU           |
| Forwarding Strategy                         | BestRoute     |
| Video codec                                 | AVC           |
| No. of video segments                       | 299           |
| No. of available representations            | 20            |
| Segment duration                            | 2 s           |
| Adaptation Logic                            | RB            |
| Start up delay                              | 0.1 s         |
| Max. buffered seconds                       | 30            |
| Consecutive gap $\gamma$                    | 5             |

Πίνακας1 Simulation Configurations

## Κεφάλαιο 5ο : Υλοποίηση

---

Η υλοποίηση έγινε υπό τις συνθήκες πέντε διαφορετικών πειραμάτων. Τα στοιχεία που συλλέγονται μέσω trace files καταφέρνουμε να τα απεικονίσουμε μέσω του excel της Microsoft. Επιλέγουμε να εξάγουμε τα στοιχεία από έναν συγκεκριμένο κόμβο (τον κόμβο 5) καθώς με γνώμονα την τοπολογία της εικόνας 5 θα έχουμε τα ίδια αποτελέσματα σε κάθε κόμβο, οπότε κρίνεται περιττή η παρουσίαση στοιχείων από άλλους κόμβους. Με βάση τις πληροφορίες που υπάρχουν στα πακέτα περιεχομένου μπορούμε να εξάγουμε σημαντικά στοιχεία για την αξιολόγηση του μηχανισμού. Το ποιο σημαντικό στοιχείο που επιλέγουμε να εμφανίσουμε στα trace files είναι ο αριθμός που αντιστοιχεί στην αναπαράσταση του βίντεο(με τον όρο αναπαράσταση εννοούμε την ανάλυση/ποιότητα του βίντεο). Τελικά, με την βοήθεια εντολών ελέγχου του excel εξάγουμε δύο μετρικές πάνω στις οποίες θα στηριχθούν τα συμπεράσματα αυτής της εργασίας.

- **Αριθμός αλλαγών (Number of switches):** Αν το επόμενο πακέτο έχει αριθμό αναπαράστασης διαφορετικό από το προηγούμενο τότε ο **Number of switches** αυξάνεται κατά ένα.
- **Η μέση τιμή της “έντασης” των αλλαγών (Average switch magnitude) :** Αν το επόμενο πακέτο έχει αριθμό αναπαράστασης διαφορετικό από το προηγούμενο τότε υπολογίζεται η νέα μέση τιμή της έντασης των αλλαγών. Ο τρόπος υπολογισμού της έγινε στο περιβάλλον του excel σύμφωνα με τον παρακάτω τύπο:

$$M_{n+1} = \text{IF}(R_n = R_{n+1}, M_n, M_n * ((N_{n+1} - 1)/N_{n+1}) + \text{ABS}(R_{n+1} - R_n)/R_{n+1})$$

Όπου  $M_{n+1}$  : Η νέα τιμή του Average switch magnitude

$M_n$  : Η προηγούμενη τιμή του Average switch magnitude

$R_n$  : Η αναπαράσταση του τμήματος n

$N_{n+1}$  : Οι αλλαγές που έχουν συμβεί μέχρι εκείνη τη στιγμή

Οι μετρικές που χρησιμοποιούμε για την αξιολόγηση των αποτελεσμάτων μας στηρίχθηκαν στις εργασίες [43], [44].

Κάθε πείραμα λέμε ότι αποτελείται από ένα σύνολο σεναρίων, κάθε σενάριο αποτελεί μία ολοκληρωμένη προσομοίωση. Η έναρξη μίας προσομοίωσης/σεναρίου αποτελείται από τις εξής ενέργειες:

1. Ο επιτιθέμενος ζητάει το mpd και στη συνέχεια ζητάει τα πακέτα του αντικειμένου περιεχομένου με έναν προκαθορισμένο λόγο on/off.
2. Ο επιτιθέμενος τελειώνει τις αιτήσεις προς τον router. Ο καταναλωτής ζητάει το mpd και στη συνέχεια ζητάει/καταναλώνει με διαδοχικά πακέτα όλο το περιεχόμενο.

Πριν ξεκινήσει ο καταναλωτής τις αιτήσεις στην cache του router θα πρέπει να υπάρχουν κάποια πακέτα αν η επίθεση ήταν επιτυχημένη. Αξίζει να σημειωθεί ακόμη ότι ο επιτιθέμενος σε όλα τα πειράματα ξεκινάει με θετική συμπεριφορά. Με αυτόν το τρόπο δυσκολεύουμε περισσότερο το έργο του μηχανισμού άμυνας.

## 5.1 Πείραμα 1

### *Μεταβλητές τιμές παραμέτρων*

Στο πρώτο πείραμα θα εξετάσουμε κάθε μοντέλο ξεχωριστά για ένα σύνολο διαφορετικών τιμών των επιμέρους παραμέτρων του. Με αυτόν το τρόπο θα βρούμε τις βέλτιστες τιμές των παραμέτρων αυτών για τον αμυντικό μηχανισμό κάθε μοντέλου. Τελικά, συγκρίνουμε τις βέλτιστες αυτές εκδοχές μεταξύ τους.

Αρχικοποιούμε:

- threshold = 5
- on/off =  $\frac{1}{2}$  (ο λόγος on/off εκφράζει τον χρόνο που ο επιτιθέμενος ζητάει μη διαδοχικά πακέτα (malicious behavior) προς τον χρόνο που ζητάει διαδοχικά πακέτα (normal behavior)).
- Update time = 100 sec (κάθε 100 sec έχουμε ενημέρωση της reputation value και πραγματοποιείται η σύγκριση με το κατώφλι).

Οι παραπάνω τιμές ισχύουν για κάθε προσομοίωση που λαμβάνει χώρα σε αυτό το πείραμα. Όσον αφορά τα μοντέλα μας, θα εξεταστούν στις παρακάτω παραμέτρους το καθένα:

**Μοντέλο 1 :** Μεταβάλλουμε την τιμή του  $\delta$  στον τύπο για τον υπολογισμό του reputation ( $\text{reputation} = \delta * \text{reputation} + \text{rating}$ ).

**Μοντέλο 2 :** Μεταβάλλουμε την τιμή του  $u$  στον τύπο για τον υπολογισμό των  $a$  και  $b$ .

$$a = u * a + \text{rating}$$

$$b = u * b + (1 - \text{rating})$$

**Μοντέλο 3 :** Μεταβάλλουμε τις τιμές του  $e$  από 0,1 έως 0,4. Πιο συγκεκριμένα το  $e$  καθορίζει την τιμή του  $f$ .

$$\text{if reputation} \leq \text{threshold}, f = 1 - e$$

$$\text{if reputation} > \text{threshold}, f = e$$

## 5.2 Πείραμα 2

### *Proactive vs Reactive*

Έχοντας καταλήξει στις καλύτερες τιμές των παραμέτρων  $\delta$ ,  $u$  και  $e$ , για τα μοντέλα 1,2 και 3 αντίστοιχα θα κάνουμε χρήση μόνο αυτών των βέλτιστων τιμών στη συνέχεια των πειραμάτων μας. Στο πείραμα 2 θα ελέγξουμε πώς συμπεριφέρονται τα μοντέλα άμυνας σε δύο διαφορετικές περιπτώσεις.

1. Αρχικοποιούμε την τιμή του reputation έτσι ώστε  $\text{reputation} < \text{threshold}$  (proactive)
2. Αρχικοποιούμε την τιμή του reputation έτσι ώστε  $\text{reputation} > \text{threshold}$  (reactive)

Στην πρώτη περίπτωση τα μοντέλα άμυνας είναι προκατειλημμένα αρνητικά και δεν αφήνουν τον router να κάνει caching από την αρχή. Ενώ στην δεύτερη είναι προκατειλημμένα θετικά και αφήνουν τον router να αποθηκεύει στην cache από το πρώτο κιόλας εισερχόμενο πακέτο.

Στο πείραμα αυτό διατηρήσαμε το  $\text{threshold} = 5$ , τον  $\text{onoff} = \frac{1}{2}$  και το  $\text{update} = 100 \text{ sec}$ .

## 5.3 Πείραμα 3

### *Μεταβλητή τιμή κατοφλίου*

Στο πείραμα αυτό εξετάζουμε τα μοντέλα άμυνας για τρεις διαφορετικές τιμές στο threshold του reputation. Όσο μεγαλύτερη είναι η τιμή του threshold τόσο

αυστηρότερος είναι ο μηχανισμός άμυνας. Ωστόσο, στόχος είναι να συγκρίνουμε και πάλι τα μοντέλα μεταξύ τους αλλά αυτή τη φορά υπό το πρίσμα διαφορετικών τιμών κατωφλίου.

Διατηρούμε και εδώ τις εξής αρχικοποιήσεις:

- threshold = 5
- on/off =  $\frac{1}{2}$
- Update time = 100 sec

## 5.4 Πείραμα 4

*Μεταβλητός λόγος on/off*

Σε αυτό το πείραμα εξετάζουμε την συμπεριφορά του κάθε μοντέλου όταν μεταβάλλουμε τον λόγο on/off του attacker. Όσο αυξάνουμε τον παρονομαστή στον λόγο on/off τόσο δυσκολότερο είναι να εντοπίσουμε τον επιτιθέμενο. Όμως, για μεγαλύτερο παρονομαστή ο επιτιθέμενος θα δημιουργεί λιγότερα σημεία ταλάντωσης. Ένας επιτιθέμενος που θα καταφέρει να παραπλανήσει τον router έτσι ώστε να αποθηκεύει τα πακέτα που προορίζονται για αυτόν αλλά θα δημιουργήσει δυο ή τρία σημεία ταλάντωσης σε όλο το βίντεο δεν κρίνεται και πολύ επιτυχημένος. Με βάση αυτή τη σκέψη και την διάρκεια του δικού μας βίντεο ορίζουμε και παίρνουμε αποτελέσματα για λόγο on/off:  $\frac{1}{2}$ ,  $\frac{1}{3}$ ,  $\frac{1}{4}$ .

## 5.5 Πείραμα 5

*Μεταβλητές τιμές ενημέρωσης της φήμης*

Στο πείραμα αυτό εξετάζουμε τα μοντέλα μας για διαφορετικούς χρόνους update. Πιο συγκεκριμένα, για 50, 100, 150 και 200 sec. Επίσης, εξετάζουμε και την περίπτωση της ενημέρωσης με κάθε πακέτο δεδομένων που λαμβάνει ο router, την οποία ονομάζουμε real-time ενημέρωση. Γίνεται αντιληπτό ότι όσο συχνότερη η ενημέρωση τόσο μεγαλύτερη πολυπλοκότητα προσθέτουμε στο σύστημα μας εμείς όμως εδώ εξετάζουμε τα αποτελέσματα καθαρά από την σκοπιά της αποτελεσματικότητας σε επίπεδο περιορισμού της επίθεσης.

Επιλέχθηκε να ελεγχθούν τα μοντέλα για  $\text{threshold} = 5$  με λόγο on/off =  $\frac{1}{2}$  και για  $\text{threshold} = 7$  με on/off =  $\frac{1}{3}$ .

## Κεφάλαιο 6ο : Αποτελέσματα

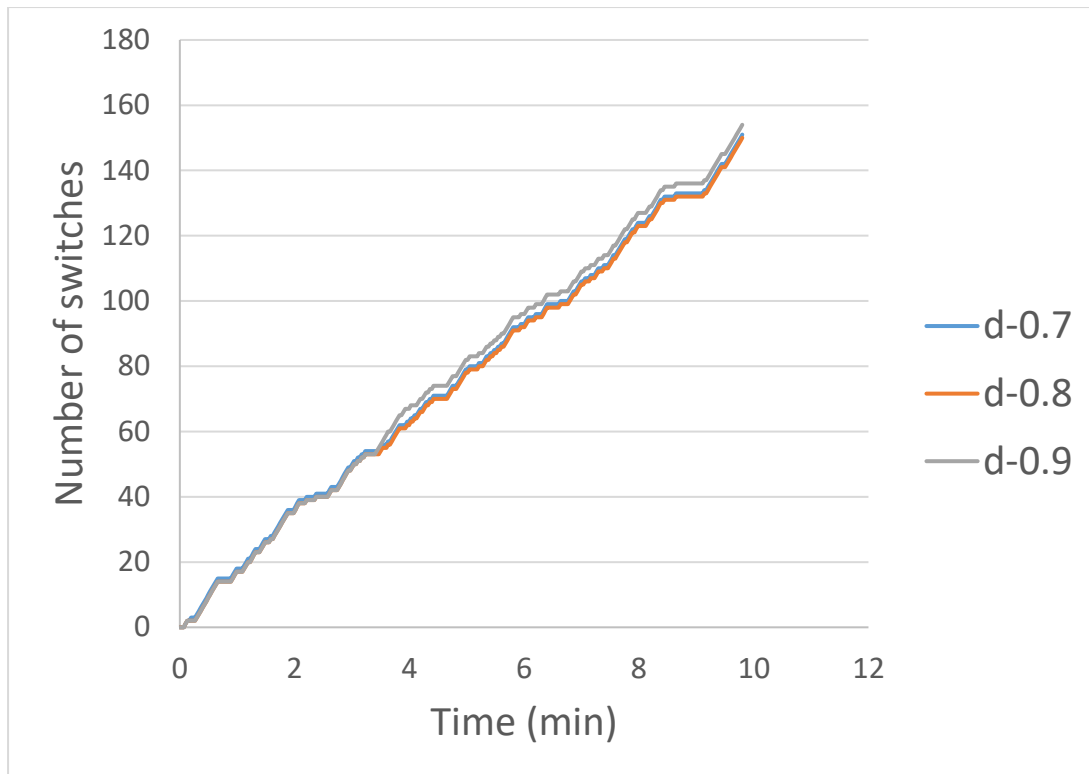
---

Σ' αυτή την ενότητα παραθέτουμε τα συγκριτικά αποτελέσματα που παίρνουμε από τις διάφορες προσομοιώσεις. Εξάγουμε τα στοιχεία που μας ενδιαφέρουν μέσω των trace files και στην συνέχεια τα επεξεργαζόμαστε στο περιβάλλον του excel της Microsoft. Σχεδιάζοντας τα γραφικά διαγράμματα από τα οποία μπορούμε να καταλήξουμε σε συμπεράσματα σχετικά με την απόδοση κάθε περίπτωσης. Πιο συγκεκριμένα, σε κάθε ένα από τα παρακάτω πειράματα, εξετάστηκαν τα τρία μοντέλα αρχικά μεμονωμένα και στη συνέχεια συγκρίθηκαν οι βέλτιστες εκδοχές αυτών. Τα μοντέλα αξιολογήθηκαν ως προς δυο παραμέτρους, τον αριθμό των αλλαγών (Number of switches) και την μέση ένταση των αλλαγών (Average switch magnitude). Ο αριθμός και η μέση ένταση των αλλαγών όπως κατανέμονται κατά τη διάρκεια του βίντεο, παρουσιάζονται στα παρακάτω διαγράμματα με σκοπό την αποτελεσματικότερη αξιολόγηση των μοντέλων.

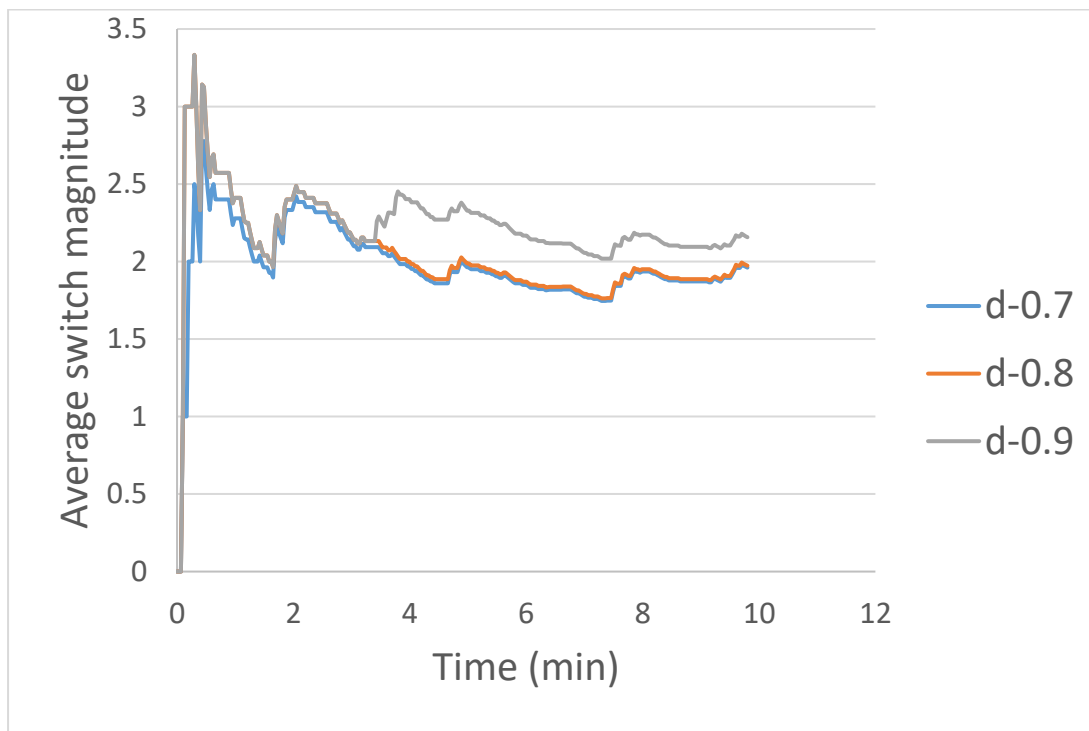
### 6.1 Πείραμα 1

#### Μοντέλο 1

Στα παρακάτω διαγράμματα παρουσιάζονται οι αριθμοί των αλλαγών για διαφορετικές τιμές της παραμέτρου  $d$  και το μέγεθος των αλλαγών για το Μοντέλο 1. Συγκεκριμένα ελέγχουμε τη παράμετρο  $d$  για τις τιμές 0.7, 0.8 και 0.9.



Εικόνα 6. Number of switches (Μοντέλο 1, Πείραμα 1)



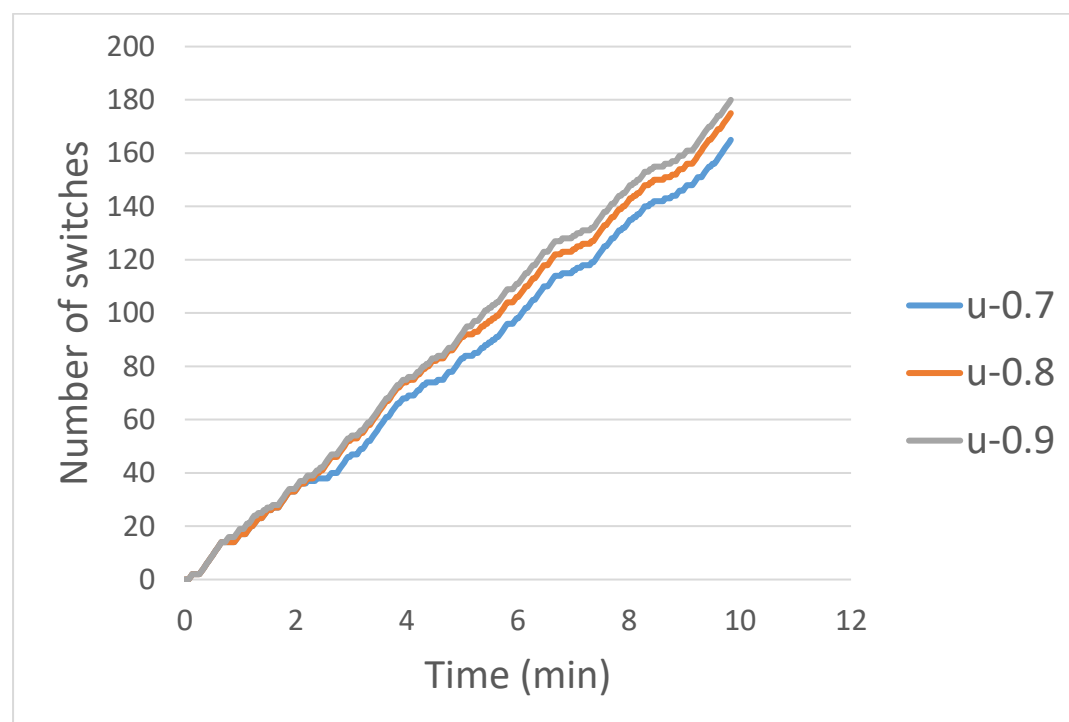
Εικόνα 7. Average switch magnitude (Μοντέλο 1, Πείραμα 1)

Από τα διάγραμμα διακρίνουμε ότι η καμπύλη για την παράμετρο  $d=0.8$  και η καμπύλη για την παράμετρο  $d=0.7$  παρουσιάζουν τα καλύτερα αποτελέσματα καθώς έχουν τον μικρότερο αριθμό αλλαγών και την μικρότερη μέση ένταση ανά αλλαγή.

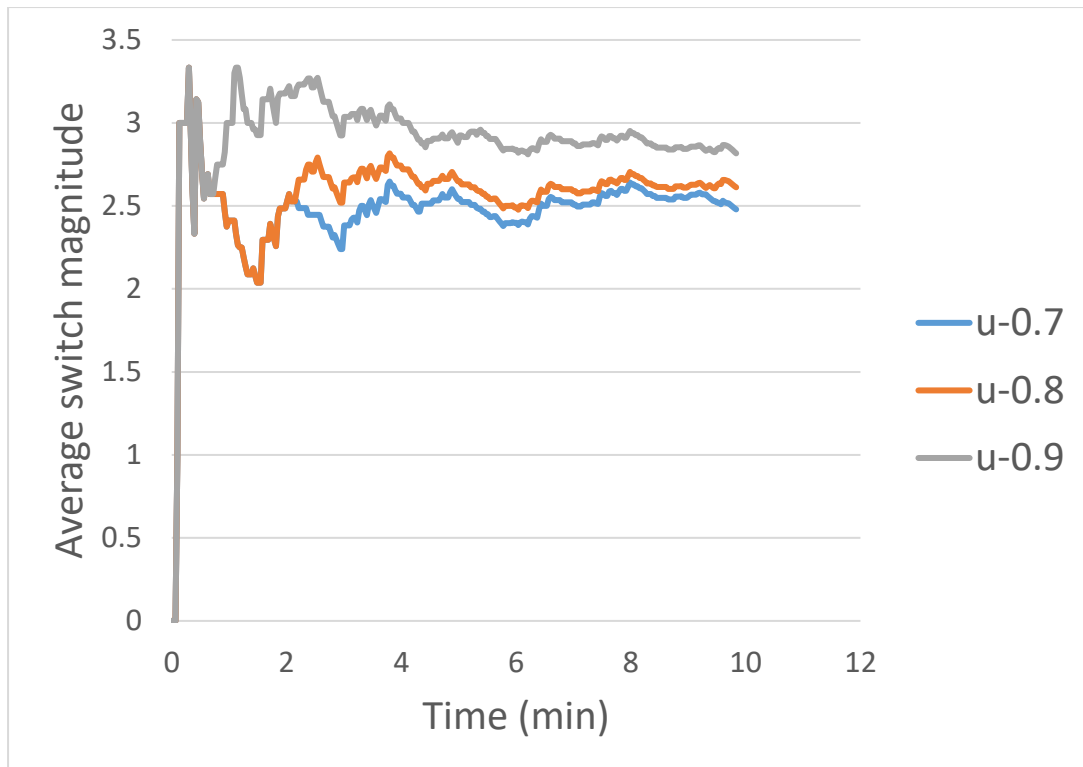
Κάτι τέτοιο μάλιστα θα ήταν αναμενόμενο λόγω της αυστηρότητας που προκύπτει από την αύξηση της παραμέτρου. Η ενημερωμένη τιμή του reputation θα είναι πάντα μεγαλύτερη όσο μεγαλύτερο είναι το  $\delta$  (αφού  $\text{reputation} = \delta * \text{reputation} + \text{rating}$ ). Τέλος, η καμπύλη για την παράμετρο  $d=0.9$  παρουσιάζει τα λιγότερο καλά αποτελέσματα καθώς εμφανίζει τον μεγαλύτερο αριθμό αλλαγών αλλά και αλλαγές μεγαλύτερης έντασης.

## Μοντέλο 2

Στα παρακάτω διαγράμματα παρουσιάζονται οι αριθμοί των αλλαγών για διαφορετικές τιμές της παραμέτρου  $u$  και το μέγεθος των αλλαγών για το Μοντέλο 2. Συγκεκριμένα παρουσιάζονται οι αλλαγές για τη παράμετρο  $u$  με τιμές 0.7, 0.8 και 0.9.



Εικόνα 8 Number of switches (Μοντέλο 2, Πείραμα 1)



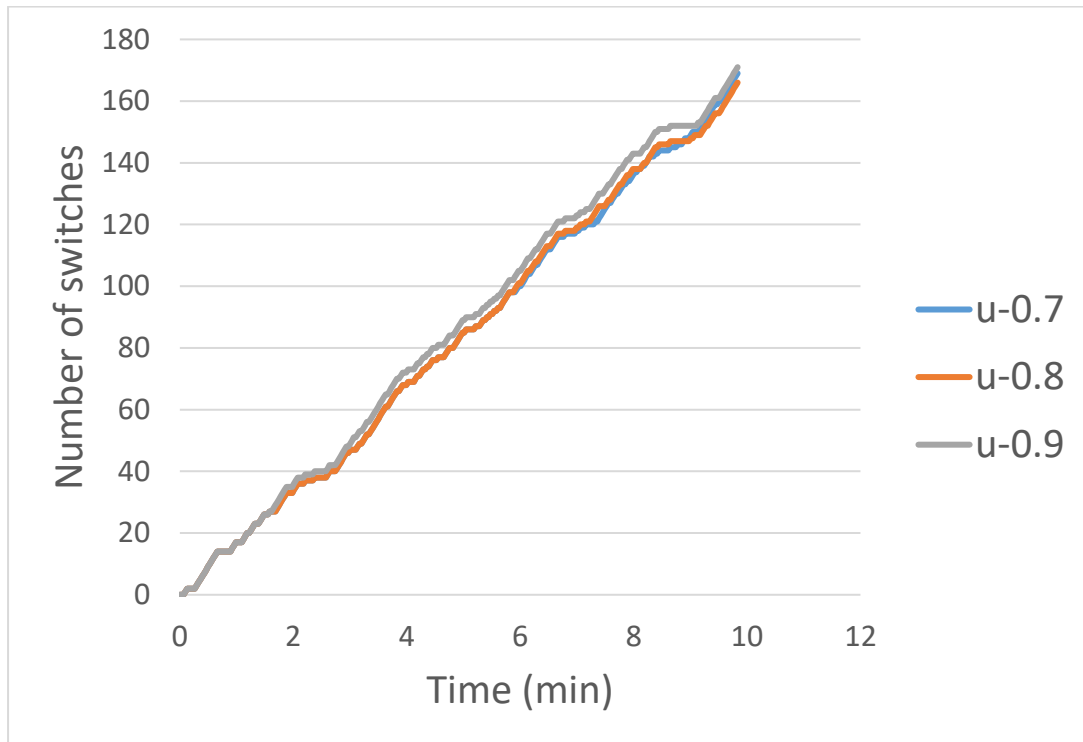
Εικόνα 9. Average switch magnitude (Μοντέλο 2, Πείραμα 1)

Από τα διάγραμμα διακρίνουμε ότι η καμπύλη για την παράμετρο  $u=0.7$  παρουσιάζει τα καλύτερα αποτελέσματα και στις δύο μετρικές. Ενώ η καμπύλη 0,9 παρουσιάζει τα χειρότερα αποτελέσματα

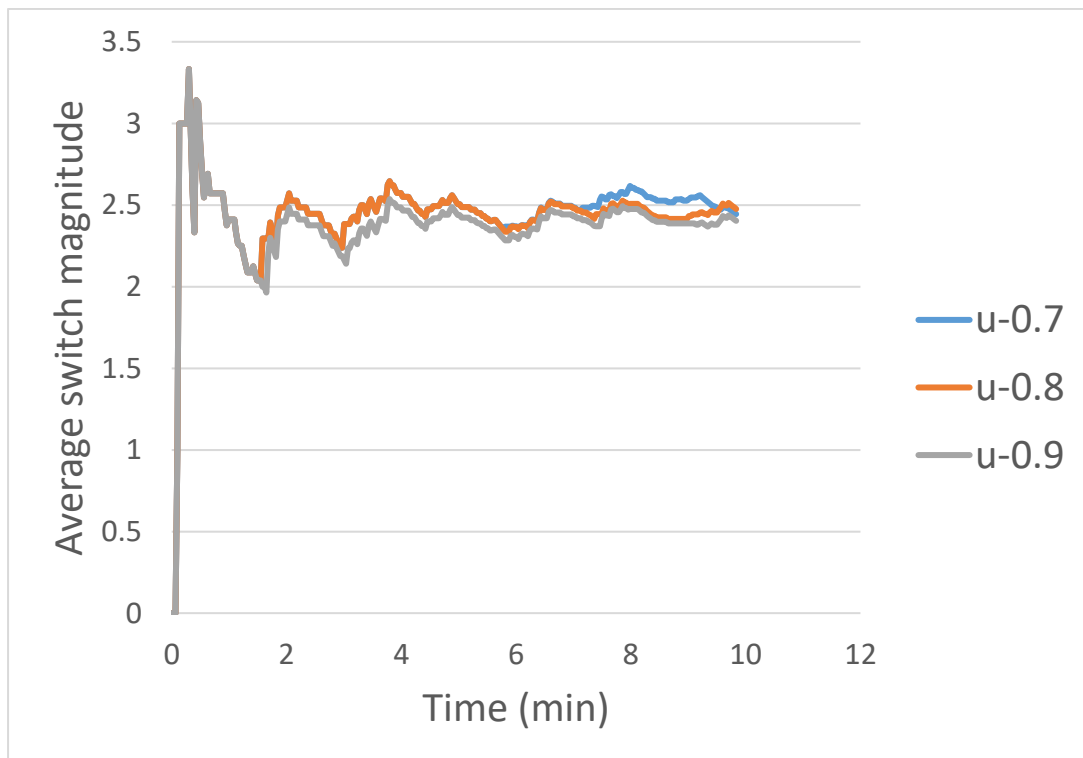
Ωστόσο, για να κατανοήσουμε σε βάθος τη λειτουργία και το αντίκτυπο της παραμέτρου  $u$ , εξετάστηκαν άλλες δυο διαφορετικές τιμές threshold. Συγκεκριμένα, εκτελέστηκαν τα πειράματα με threshold 6 και 7 σε αντιπαραβολή με την αρχική τιμή 5. Αρχικά, υπενθυμίζουμε ότι ο ρόλος του  $u$  είναι να δίνει το επιθυμητό βάρος στην παλιά συμπεριφορά του client. Το  $u$  παίρνει τιμές από 0 έως 1. Όσο μεγαλύτερο  $u$  τόσο περισσότερη σημασία δίνουμε στην συμπεριφορά του client μέχρι τώρα και τόσο λιγότερη στην συμπεριφορά του αυτή τη στιγμή δηλαδή την συμπεριφορά που είχε στο τελευταίο rating.

Θα περιμέναμε λοιπόν, να έχουμε καλύτερα αποτελέσματα για μεγαλύτερες τιμές του  $u$ . Αυτό όμως γίνεται κυρίως διακριτό όταν έχουμε μεγαλύτερες τιμές threshold. Γι' αυτό και στο πείραμα με threshold 7 το μοντέλο άμυνας ανταποκρίνεται καλύτερα για  $u = 0,9$ . Ενώ για threshold 6 έχουμε ίδια αποτελέσματα και για τις τρεις τιμές του  $u$ .

Στα παρακάτω διαγράμματα παρουσιάζονται οι αριθμοί των αλλαγών και το μέγεθος των αλλαγών για διαφορετικές τιμές της παραμέτρου  $u$  για το Μοντέλο 2 και τιμή  $\text{threshold} = 6$ .

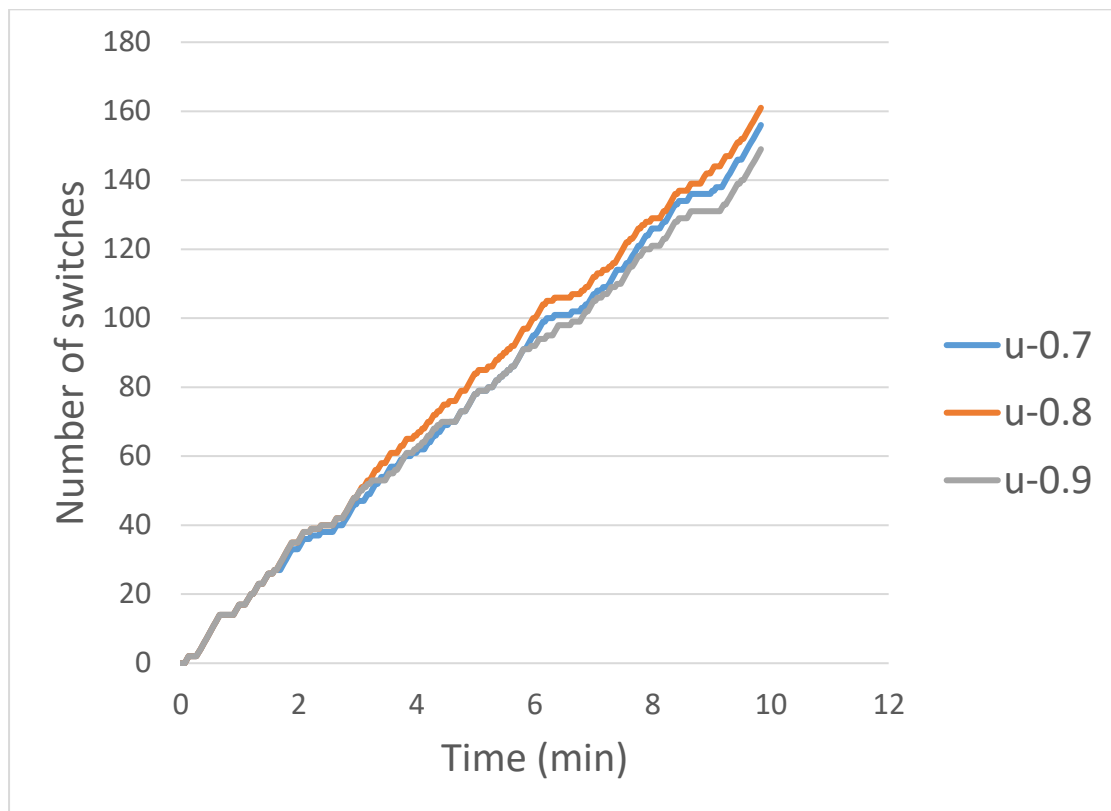


Εικόνα 10. Number of switches (Μοντέλο 2, Πείραμα 1, threshold = 6)

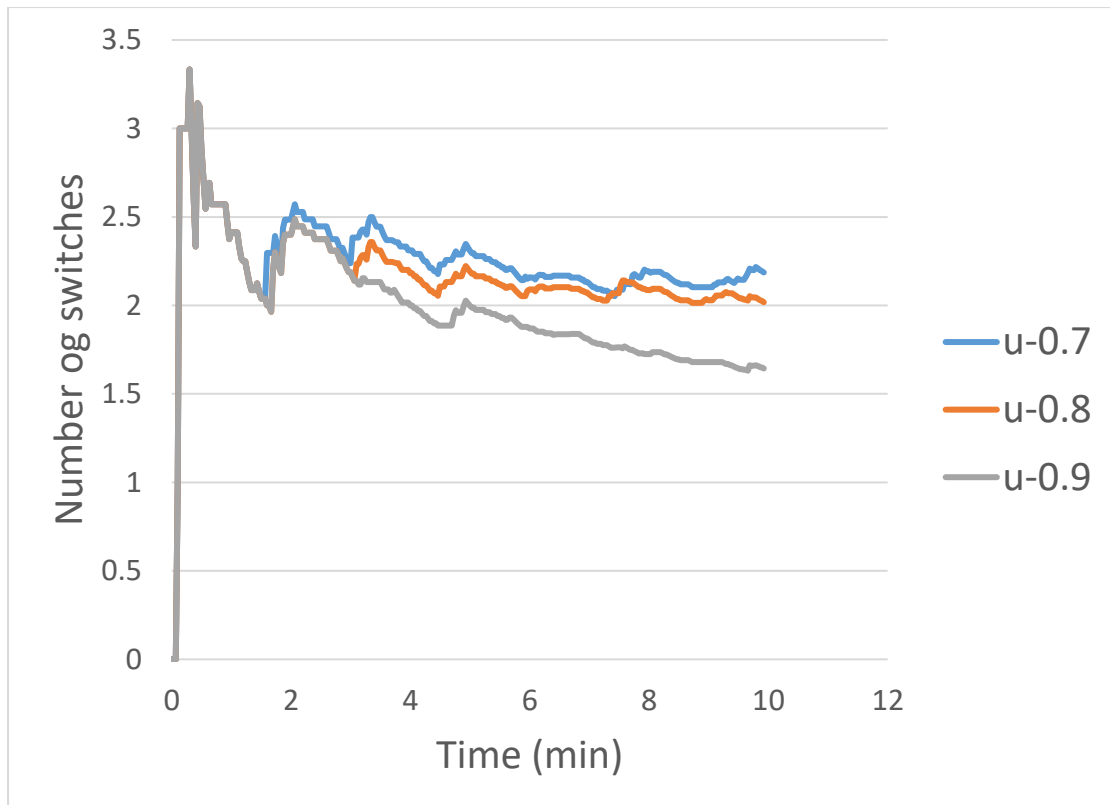


Εικόνα 11. Average switch magnitude (Μοντέλο 2, Πείραμα 1, threshold = 6)

Στα παρακάτω διαγράμματα παρουσιάζονται οι αριθμοί των αλλαγών και το μέγεθος των αλλαγών για διαφορετικές τιμές της παραμέτρου  $u$  για το οντέλο 2 και τιμή  $\text{threshold} = 7$ .



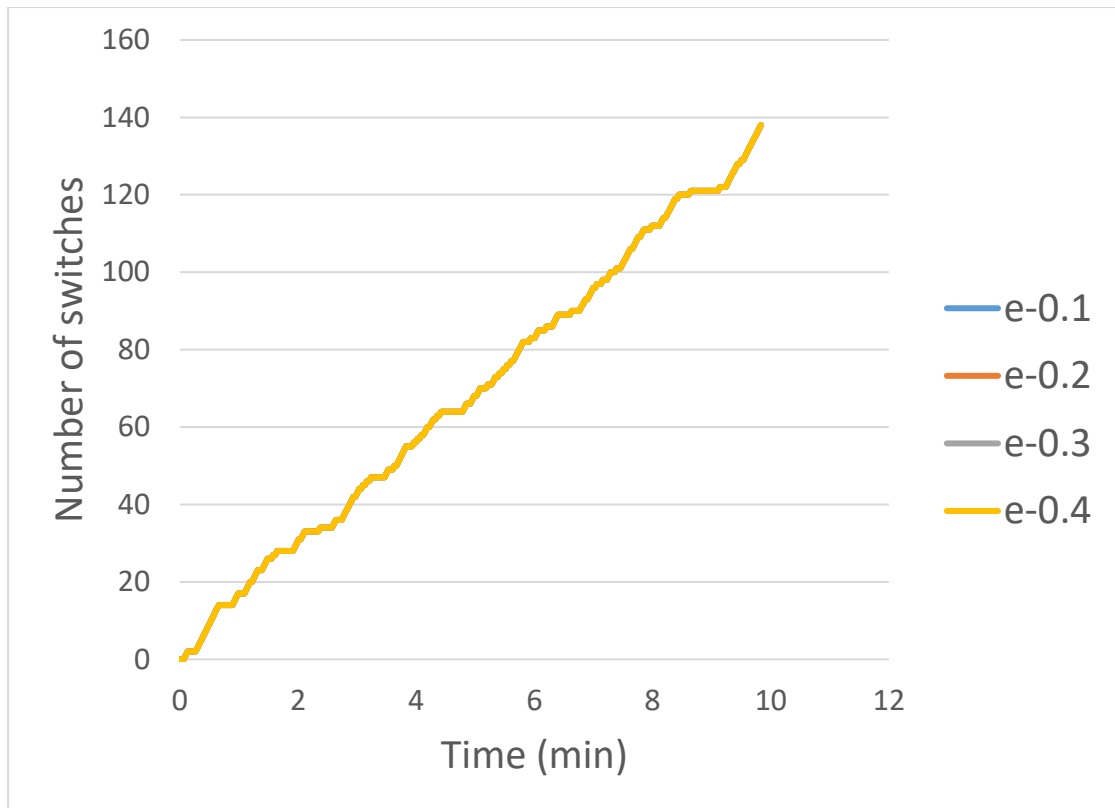
Εικόνα 12. Number of switches (Μοντέλο 2, Πείραμα 1,  $\text{threshold} = 7$ )



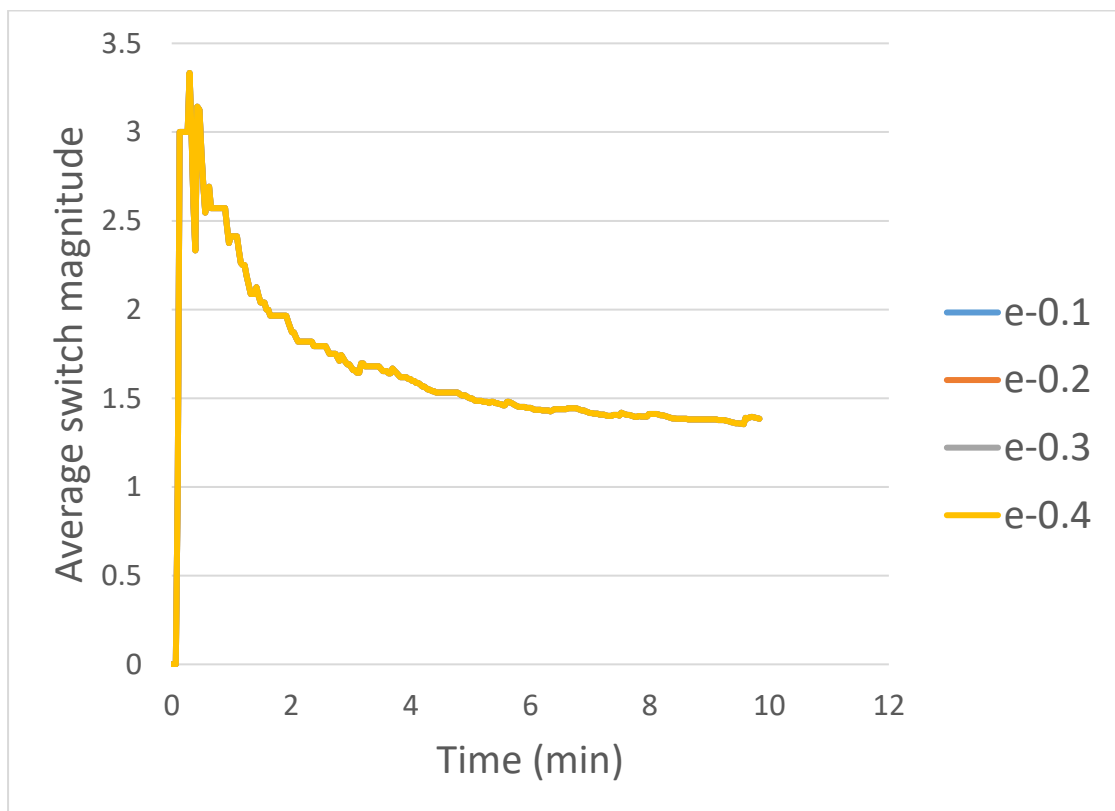
Εικόνα 13. Average switch magnitude (Μοντέλο 2, Πείραμα 1, threshold = 7)

### Μοντέλο 3

Στα παρακάτω διαγράμματα παρουσιάζονται οι αριθμοί των αλλαγών για διαφορετικές τιμές της παραμέτρου  $u$  και το μέγεθος των αλλαγών για το Μοντέλο 3. Συγκεκριμένα παρουσιάζονται οι αλλαγές για τη παράμετρο  $u$  και για τιμές  $\epsilon$  0.1, 0.2, 0.3 και 0.4.



Εικόνα 6. Number of switches (Μοντέλο 3, Πείραμα 1)

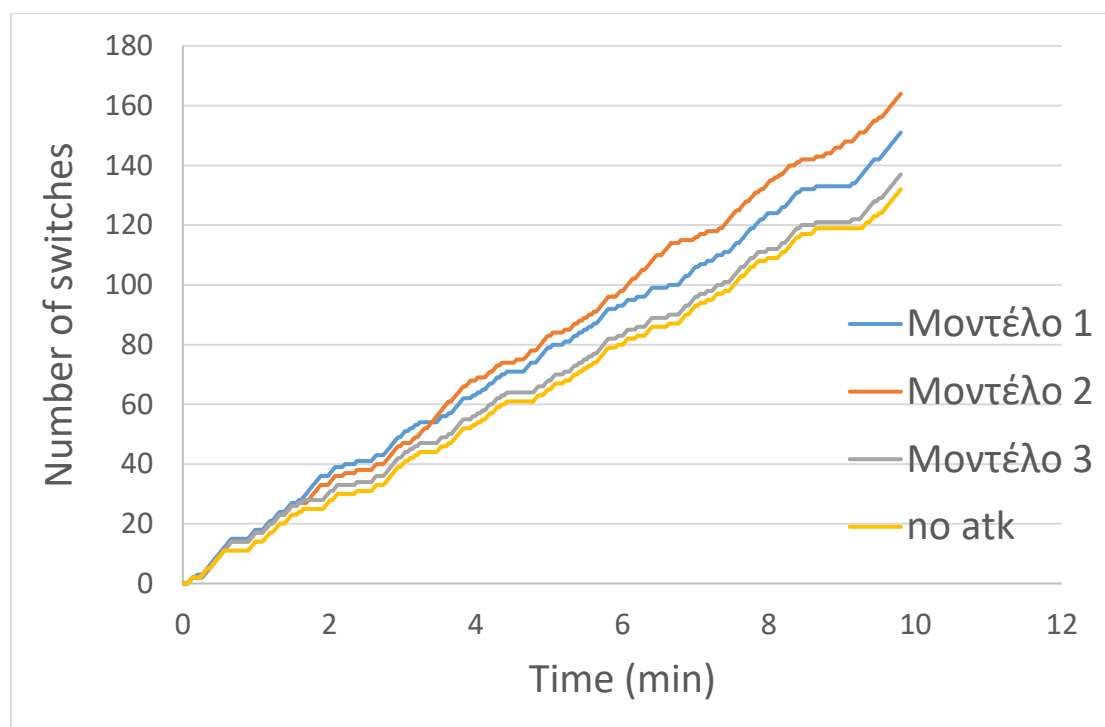


Εικόνα 15. Average switch magnitude (Μοντέλο 3, Πείραμα 1)

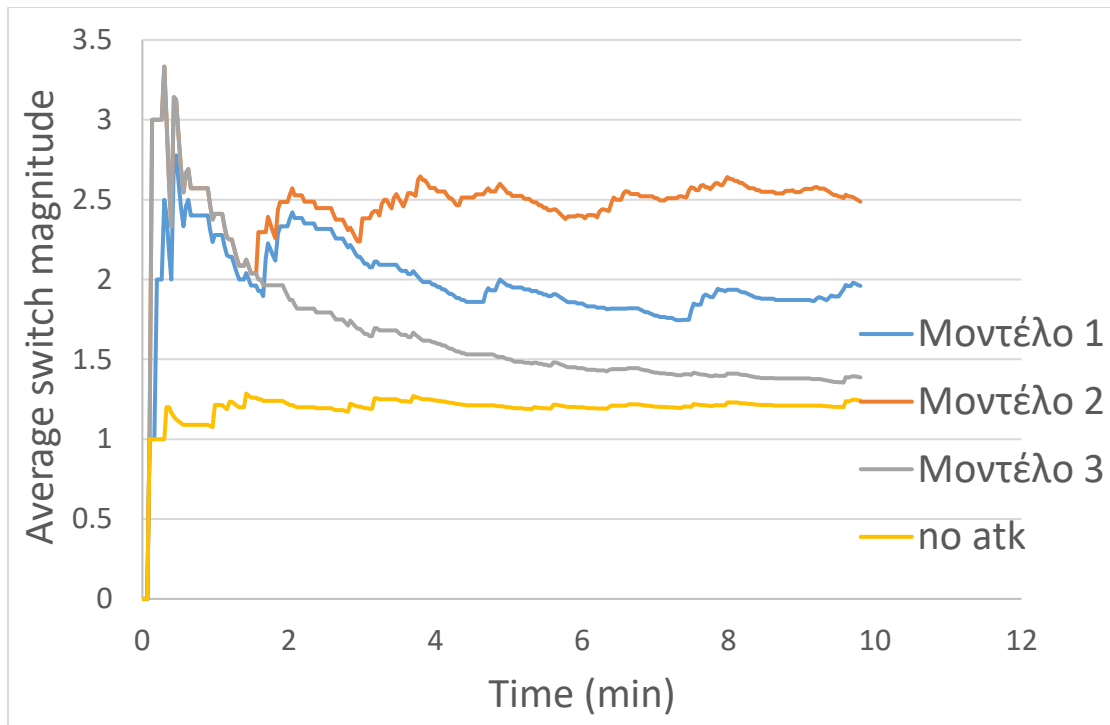
Από τα διαγράμματα διακρίνουμε ότι οι καμπύλες για τις τέσσερις διαφορετικές τιμές του  $\epsilon$  παρουσιάζουν το ίδιο καλά αποτελέσματα.

### Σύγκριση Μοντέλων

Στα παρακάτω διαγράμματα παρουσιάζεται η σύγκριση των τριών μοντέλων λαμβάνοντας υπόψη τις βέλτιστες παραμέτρους για το καθένα. Η σύγκριση γίνεται ως προς τον αριθμό των αλλαγών (Number of switches) και το μέσο μέγεθος των αλλαγών (Average switch magnitude). Στο σημείο αυτό προσθέτουμε και μία προσομοίωση που λαμβάνει χώρα χωρίς επιτιθέμενο.



Εικόνα 16. Number of switch (Σύγκριση Μοντέλων, Πείραμα1)



Εικόνα 17. Average switch magnitude (Σύγκριση Μοντέλων, Πείραμα1)

Από αυτή τη σύγκριση προκύπτει ότι για το Πείραμα 1 την μεγαλύτερη αποδοτικότητα συνολικά παρουσιάζει το Μοντέλο 3 επειδή όπως φαίνεται και από τις γραφικές καταφέρει να μην ξεγελαστεί ξανά από τον επιτιθέμενο. Αυτό γίνεται αντιληπτό και στις δύο μετρικές όμως στην δεύτερη φαίνεται πιο καθαρά ότι κοντά στα 2 λεπτά της διάρκειας του βίντεο η μέση ένταση των αλλαγών στα 2 πρώτα μοντέλα αυξάνεται απότομα ενώ στο μοντέλο 3 μειώνεται σταθερά. Σε εκείνο ακριβώς το σημείο ο δρομολογητής των μοντέλων 1 και 2 έχει αποθηκεύσει κακόβουλο περιεχόμενο με κενά. Τέλος το μοντέλο 2 παρουσιάζει τα χειρότερα αποτελέσματα.

## 6.2 Πείραμα 2

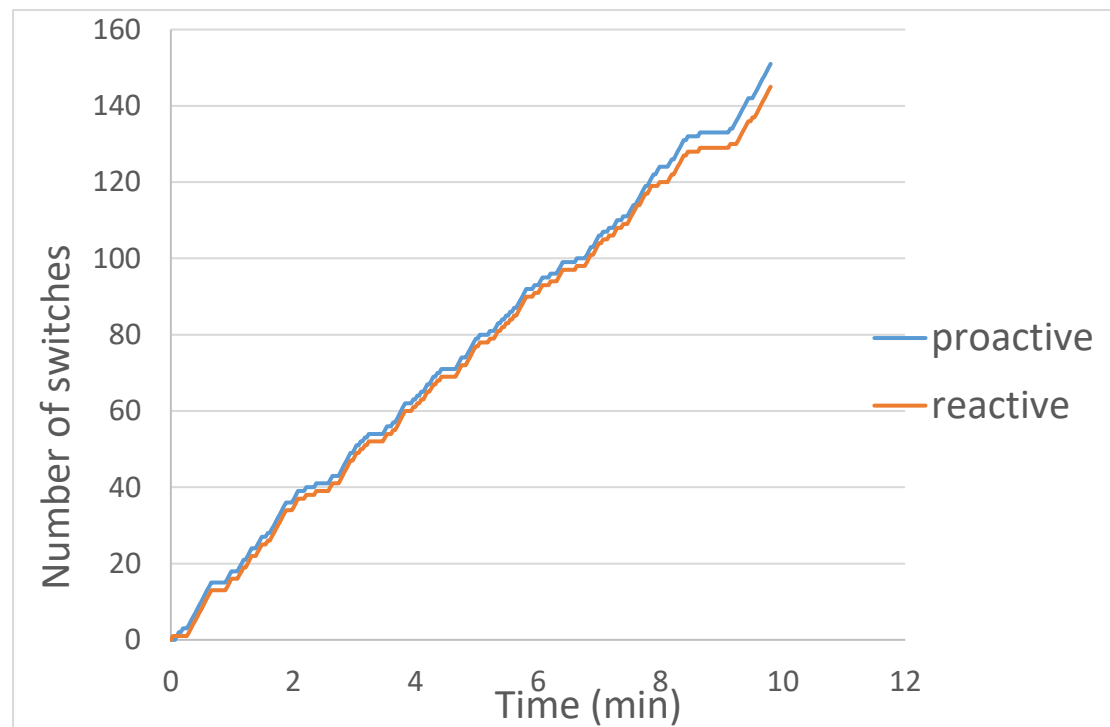
Στο πείραμα 2 επιλέγουμε τις καλύτερες εκδοχές των μοντέλων 1,2 και 3. Εξετάζουμε το κάθε μοντέλο υπό το πρίσμα δύο διαφορετικών καταστάσεων.

- i. Αρχικοποιούμε την τιμή του reputation έτσι ώστε  $\text{reputation} < \text{threshold}$  (proactive)
- ii. Αρχικοποιούμε την τιμή του reputation έτσι ώστε  $\text{reputation} \geq \text{threshold}$  (reactive)

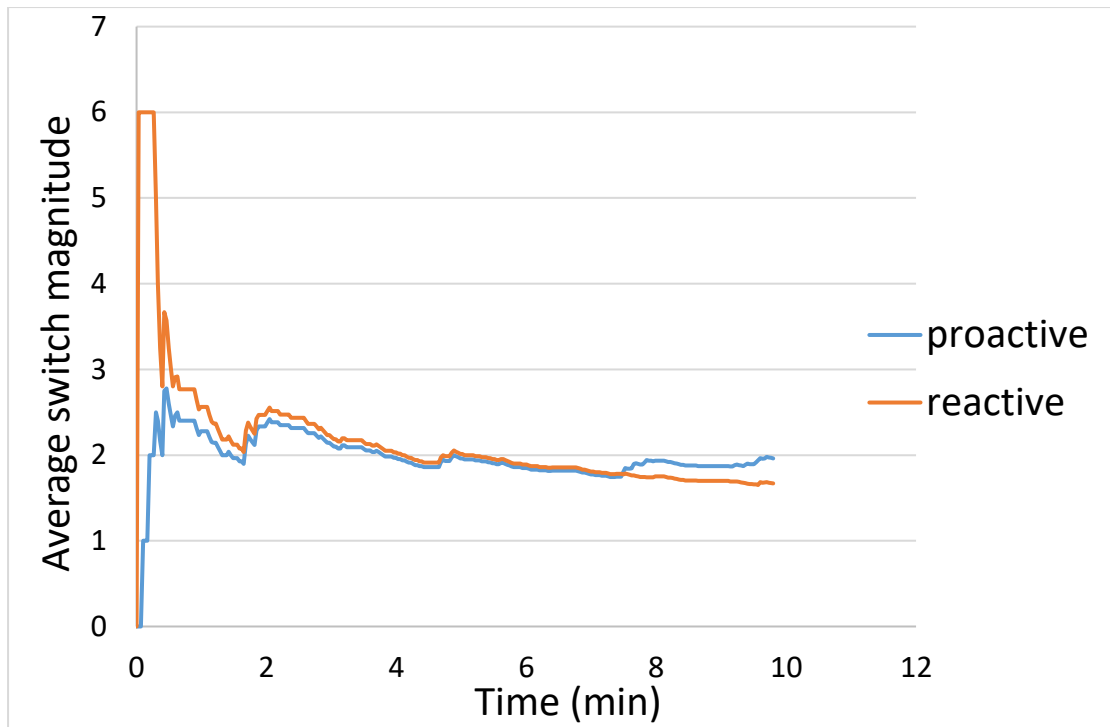
Με άλλα λόγια στην πρώτη περίπτωση είμαστε προκατειλημμένοι αρνητικά απέναντι σε κάθε χρήστη και δεν αφήνουμε τον router να κάνει caching από την αρχή. Ο χρήστης θα πρέπει να κερδίσει την εμπιστοσύνη μας, ονομάζουμε αυτή τη περίπτωση proactive. Στη δεύτερη περίπτωση είμαστε προκατειλημμένοι θετικά και επιτρέπουμε στον router να αποθηκεύει προσωρινά περιεχόμενο από την αρχή (από τα πρώτα πακέτα). Ονομάζουμε αυτή την περίπτωση reactive.

### Μοντέλο 1

Στα δύο παρακάτω διαγράμματα παρουσιάζονται ο αριθμός των αλλαγών και η μέση ένταση των αλλαγών ως προς την χρονική διάρκεια του βίντεο κατά την εκτέλεση της προσομοίωσης με τον αμυντικό μηχανισμό του μοντέλου 1.



Εικόνα 18. Number of switches (Μοντέλο 1, Πείραμα 2)

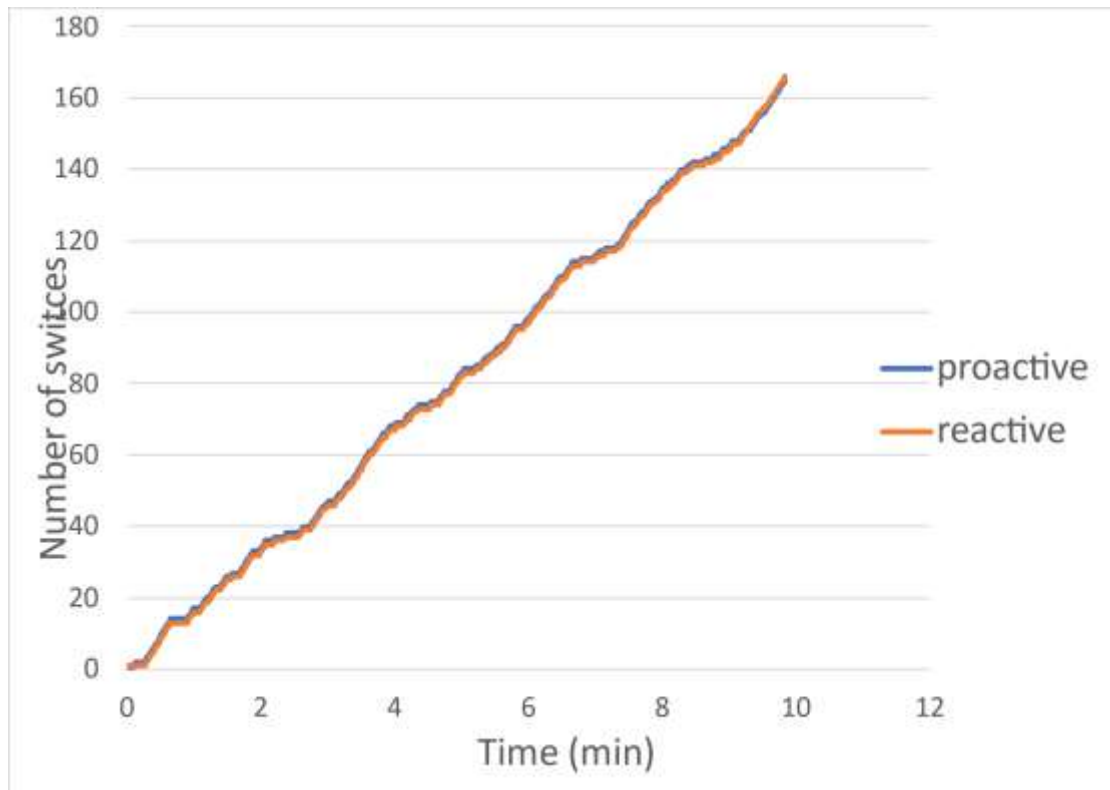


Εικόνα 19. Average switch magnitude (Μοντέλο 1, Πείραμα 2)

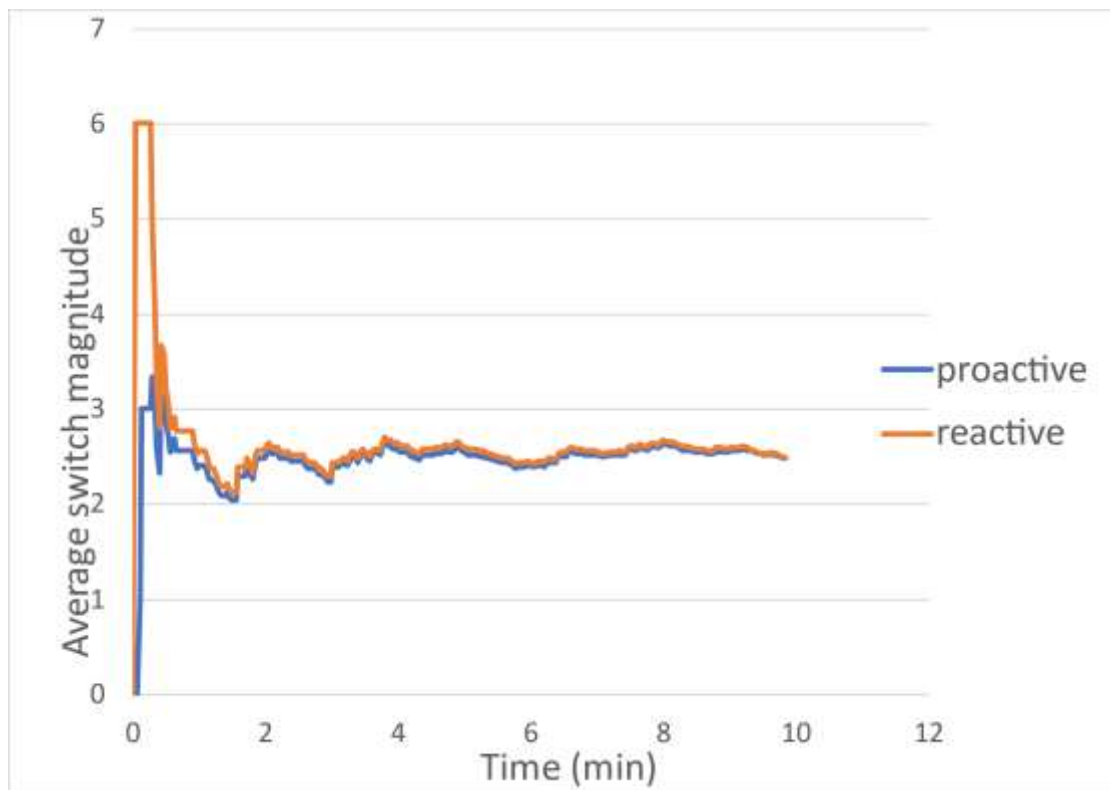
Συνολικά στο μοντέλο 1 παρατηρούμε ότι η reactive μέθοδος παρουσιάζει πιο θετικά αποτελέσματα, χωρίς όμως σημαντική διαφορά.

### Μοντέλο 2

Στα δύο παρακάτω διαγράμματα παρουσιάζονται ο αριθμός των αλλαγών και η μέση ένταση των αλλαγών ως προς την χρονική διάρκεια του βίντεο κατά την εκτέλεση της προσομοίωσης με τον αμυντικό μηχανισμό του μοντέλου 2.



Εικόνα 20. Number of switches (Μοντέλο 2, Πείραμα 2)

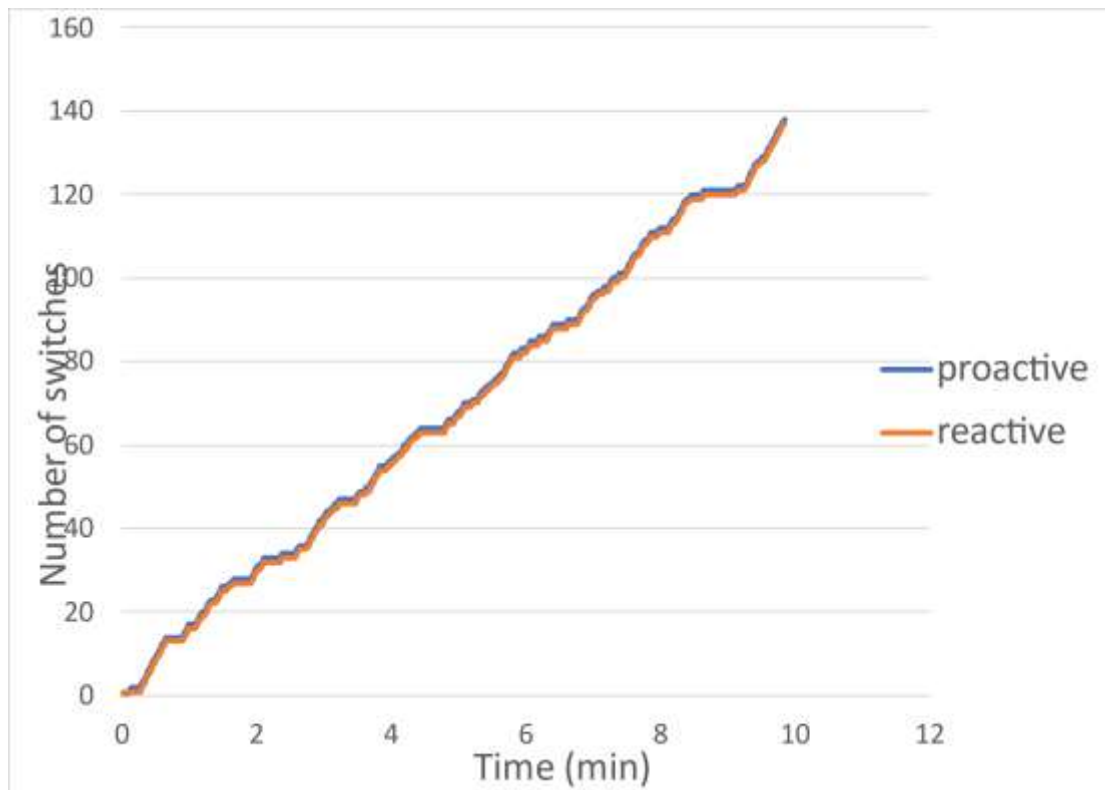


Εικόνα 21. Average switch magnitude (Μοντέλο 2, Πείραμα 2)

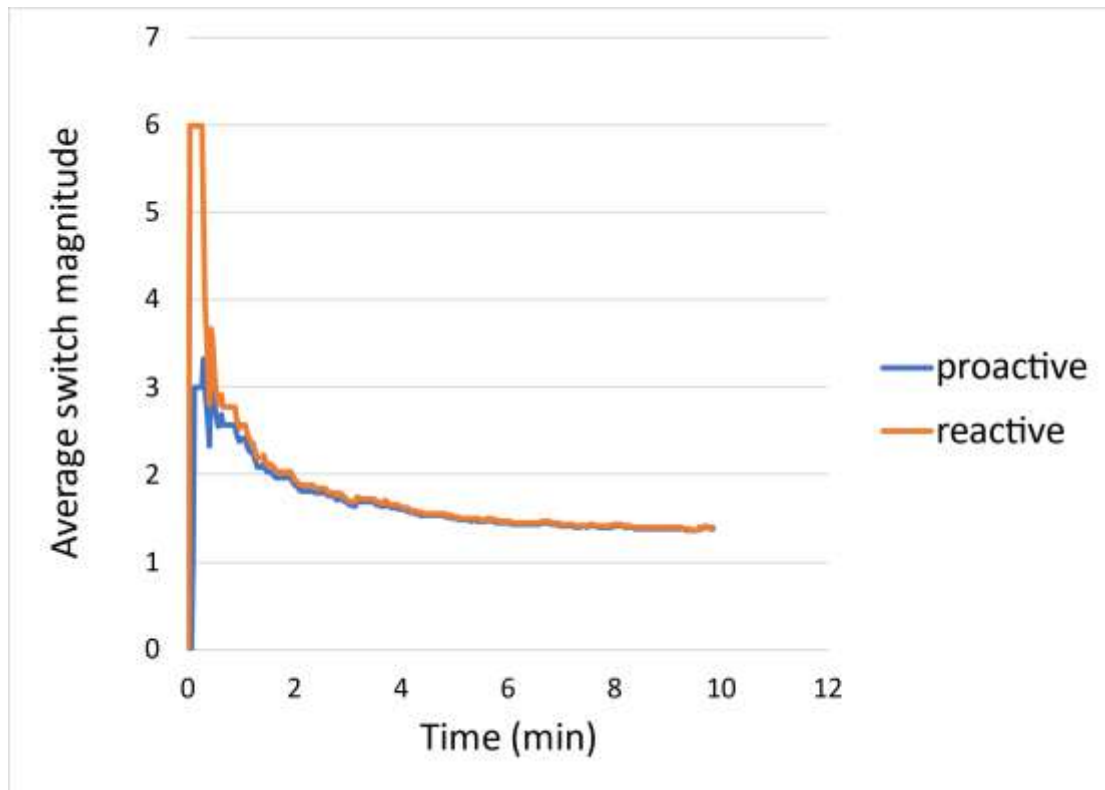
Συνολικά στο μοντέλο 2 παρατηρούμε ότι η proactive μέθοδος παρουσιάζει πιο θετικά αποτελέσματα, στην αρχή του βίντεο. Ωστόσο μετά τα πρώτα 40 sec οι δύο προσεγγίσεις συγκλίνουν σε ίδιο αριθμό αλλαγών καθώς και ίδιας έντασης.

### Μοντέλο 3

Στα δύο παρακάτω διαγράμματα παρουσιάζονται ο αριθμός των αλλαγών και η μέση ένταση των αλλαγών ως προς την χρονική διάρκεια του βίντεο κατά την εκτέλεση της προσομοίωσης με τον αμυντικό μηχανισμό του μοντέλου 3.



Εικόνα 7 Number of switches (Μοντέλο 3, Πείραμα 2)



Εικόνα 23. Average switch magnitude (Μοντέλο 3, Πείραμα 2)

Συνολικά στο μοντέλο 3 παρατηρούμε ότι η proactive μέθοδος παρουσιάζει πιο θετικά αποτελέσματα στην αρχή του βίντεο. Ωστόσο και εδώ μετά τα πρώτα 40 sec οι δύο μέθοδοι δεν παρουσιάζουν διαφορές στα διαγράμματα.

Τα αποτελέσματα από τις δύο αυτές προσεγγίσεις δεν διαφέρουν πολύ στα δικά μας πειράματα. Αυτό μας κάνει να συμπεράνουμε ότι η τιμή του reputation ακολουθεί μια συγκεκριμένη πορεία που επηρεάζεται κυρίως από τη συμπεριφορά του client και όχι τόσο από τις αρχικές τιμές. Ωστόσο, στην αρχή του βίντεο παρατηρούμε κάποιες αλλαγές όπως ήταν αναμενόμενο. Στην εργασία αυτή έχουμε δημιουργήσει έναν attacker ο οποίος ξεκινά με κανονική συμπεριφορά για να μας παραπλανήσει και ως εκ τούτου να είναι δυσκολότερο το έργο μας. Στην αρχή του βίντεο λοιπόν και ποιο συγκεκριμένα στα πρώτα 100 sec, μέχρι και να γίνει η πρώτη ενημέρωση από στοιχεία του δικτύου, ο router μας παίρνει αποφάσεις για την προσωρινή αποθήκευση με βάση την αρχικοποιημένη τιμή που του ορίσαμε. Σε αυτά ακριβώς τα 100 sec εντοπίζουμε τις όποιες διαφορές στις 2 μεθόδους.

Στην μέθοδο που ονομάζουμε reactive ο router αποφασίζει να αποθηκεύσει περιεχόμενο (γεγονός που είναι εσφαλμένο ειδικά για attacker που ξεκινάει με μη-κανονική συμπεριφορά) στα πρώτα 100 sec και μετά με βάση τη συμπεριφορά του

attacker να ρίξει ή να ανεβάσει την αρχική τιμή που του δώσαμε. Ας εξετάσουμε πιο αναλυτικά τις καταστάσεις σε αυτό το πείραμα:

### Reactive

Στην περίπτωση όπου ο attacker ξεκινάει με κανονική συμπεριφορά για να μας παραπλανήσει ο router αποθηκεύει από αυτόν περιεχόμενο το οποίο στη συνέχεια μεταδίδει κατά τη διάρκεια των πρώτων 100seconds στον καλόβουλο χρήστη. Πράγμα που κάποιες φορές μπορεί να ευνοήσει την εμπειρία (όπως παρατηρούμε στα μοντέλα 2 και 3). Πιο αναλυτικά θα γίνει μία και μόνο αλλαγή στα representation των segment για τα επόμενα πρώτα 100sec. Μεταξύ του πρώτου και δεύτερου πακέτου. Εφόσον μετά την παραλαβή του πρώτου πακέτου ο καλόβουλος χρήστης ερμηνεύσει καλές επιδόσεις δικτύου θα ζητήσει το δεύτερο πακέτο να έχει το μέγιστο representation. Αυτό το πακέτο υπάρχει στον router όπως και μερικά ακόμα διαδοχικά. Μέχρι τη στιγμή όπου ο επιτιθέμενος άλλαξε συμπεριφορά από καλόβουλη σε κακόβουλη. Δηλαδή σε περίπτωση που δεν αποθηκεύαμε από τον attacker και η μετάδοση γινόταν χωρίς την παρεμβολή του θα είχαμε συνήθως περισσότερες αλλά ομαλότερες αλλαγές στα representations. Στο πείραμα μας φαίνεται αυτό ξεκάθαρα και στα 3 μοντέλα.

Όπως είπαμε και παραπάνω ο attacker ξεκινάει με καλόβουλη συμπεριφορά στην περίπτωση όμως που ο attacker ξεκινάει με κακόβουλη συμπεριφορά ο router αποθηκεύει περιεχόμενο με «κενά» ενδιάμεσα στα ληφθέντα πακέτα πράγμα που οδηγεί σε αυξημένες ταλαντώσεις και την κακή εμπειρία του χρήστη.

### Proactive

Στην περίπτωση όπου ο attacker ξεκινάει με κενό ο router δεν αποθηκεύει πράγμα που είναι σωστό καθώς αποφεύγει τις ταλαντώσεις και την κακή εμπειρία του χρήστη.

Στην περίπτωση όπου ο attacker ξεκινάει με κανονική συμπεριφορά για να μας παραπλανήσει ο router πάλι δεν αποθηκεύει από αυτόν περιεχόμενο.

### **Συνολικά :**

Το συμπέρασμα μας είναι ότι ένας μηχανισμός άμυνας με proactive συμπεριφορά θα λειτουργούσε καλύτερα απέναντι σε οποιονδήποτε attacker (είτε αρχίζει με κανονική είτε με μη-κανονική συμπεριφορά, πράγμα που δεν μπορούμε να

το προβλέψουμε) καθώς στην περίπτωση μας (ξεκινάει με κανονική συμπεριφορά) οι διαφορές ανάμεσα στον proactive και reactive είναι αμελητέες.

Ένα ακόμα χρήσιμο συμπέρασμα από αυτό το πείραμα είναι ότι η τιμή του reputation μετά τα πρώτα updates γίνεται ίδια. Οπότε όσο μεγαλύτερο το περιεχόμενο που μεταδίδουμε (όσα περισσότερα πακέτα δεδομένων μεταδίδουμε) τόσο λιγότερο αντιληπτή γίνεται η διαφορά.

Με αυτή την επιλογή όμως δεν αφήνουμε και έναν καλό χρήστη να αποθηκεύσει περιεχόμενο για τα πρώτα sec μέχρι να έρθει η ενημέρωση και να αποδοθεί δικαιοσύνη. Λύση σε αυτό το πρόβλημα μπορούμε να δώσουμε με πιο συχνές ενημερώσεις ή με ενημερώσεις real-time (με κάθε νέο πακέτο).

Μια ενδιαμέση λύση ίσως ήταν να ξεκινάει με real-time (για τα πρώτα πακέτα μόνο) και στη συνέχεια να συνεχίζει με updates ανά κάποια sec, για λόγους πολυπλοκότητας.

### 6.3 Πείραμα 3

Στο πείραμα αυτό εξετάζουμε τα μοντέλα άμυνας για τρεις διαφορετικές τιμές στο threshold του reputation (5 6 και 7). Η τιμή του threshold καθορίζει την ανοχή του εκάστοτε αμυντικού μηχανισμού στην κακόβουλη συμπεριφορά. Αν θέσουμε threshold = 5 επιτρέπουμε 50% κακόβουλη συμπεριφορά, ενώ αν ορίσουμε threshold = 7 επιτρέπουμε 30% αρνητική συμπεριφορά. Η τελική απόφαση του router για προσωρινή αποθήκευση πακέτων καθορίζεται από την παρακάτω έλεγχο ως εξής:

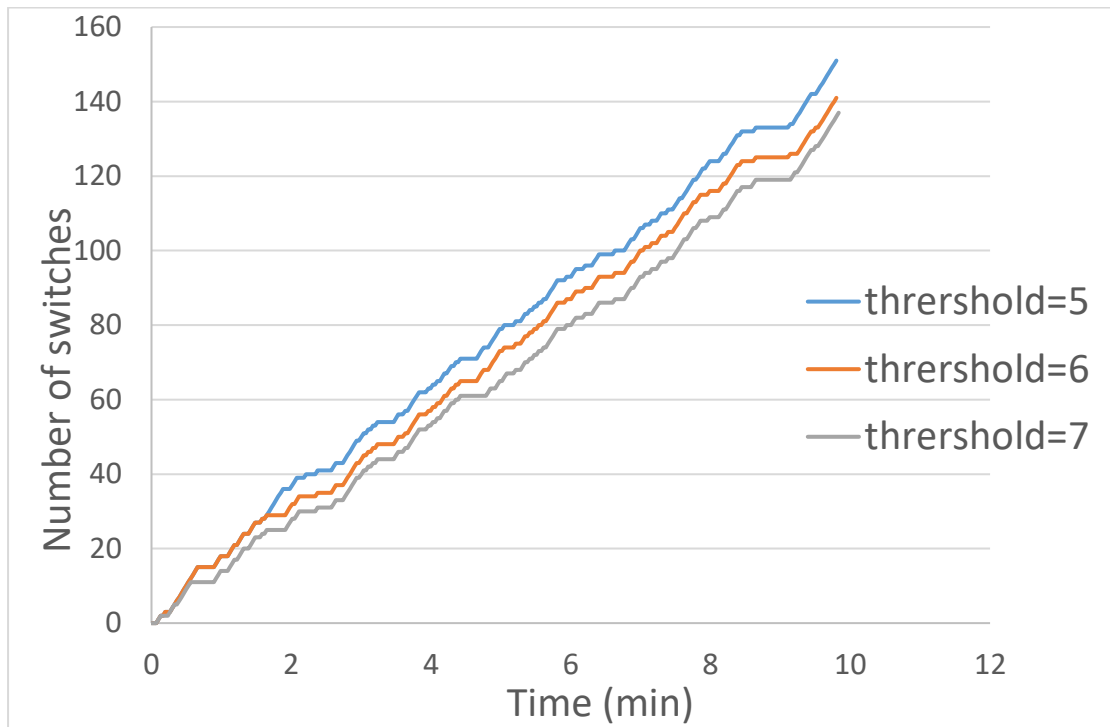
if reputation value  $\leq$  threshold , αποθήκευσε το πακέτο

if reputation value  $\leq$  threshold , μην αποθηκεύεις

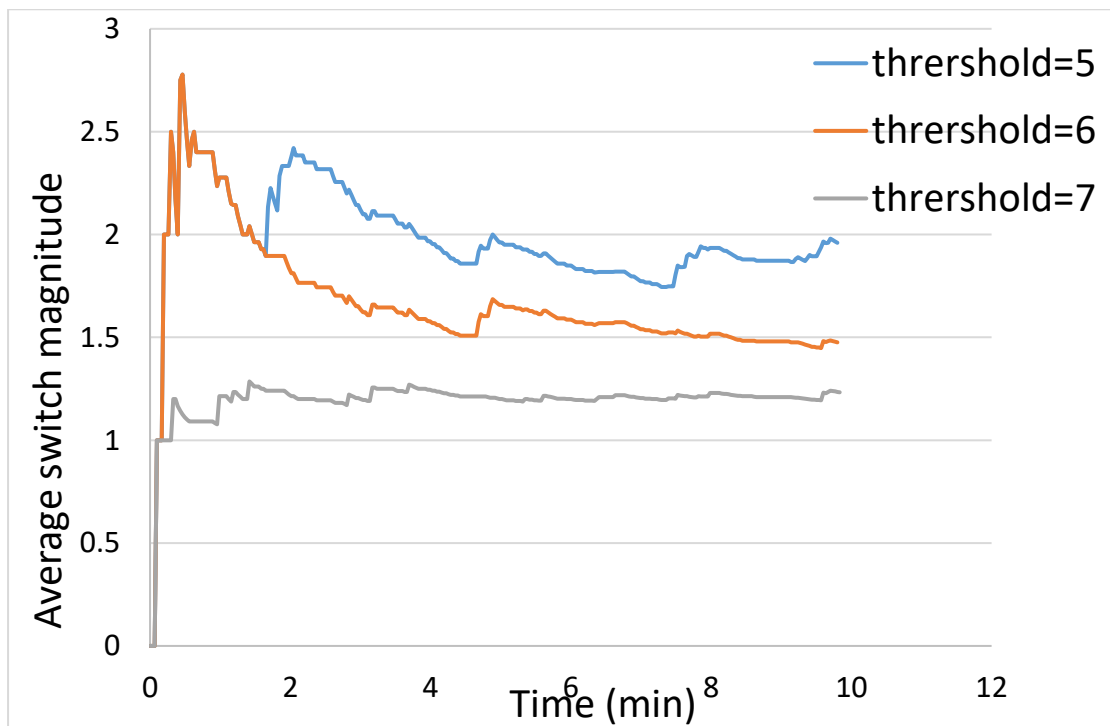
Ο έλεγχος αυτός πραγματοποιείται με την έλευση κάθε πακέτου ανεξάρτητα από την στιγμή που ανανεώνεται η τιμή του reputation.

Παρακάτω παρουσιάζουμε τις συγκρίσεις των διαφορετικών τιμών του threshold στα τρία μοντέλα ξεχωριστά.

### Μοντέλο 1

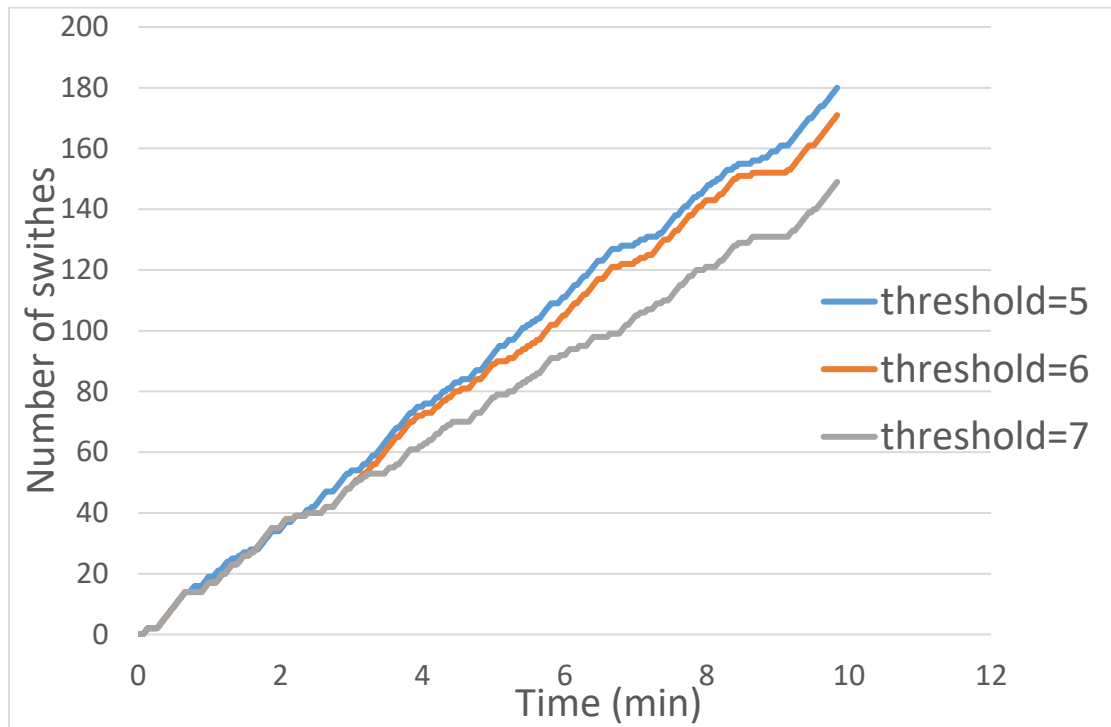


Εικόνα 24 Number of switches(Μοντέλο 1, Πείραμα 3)

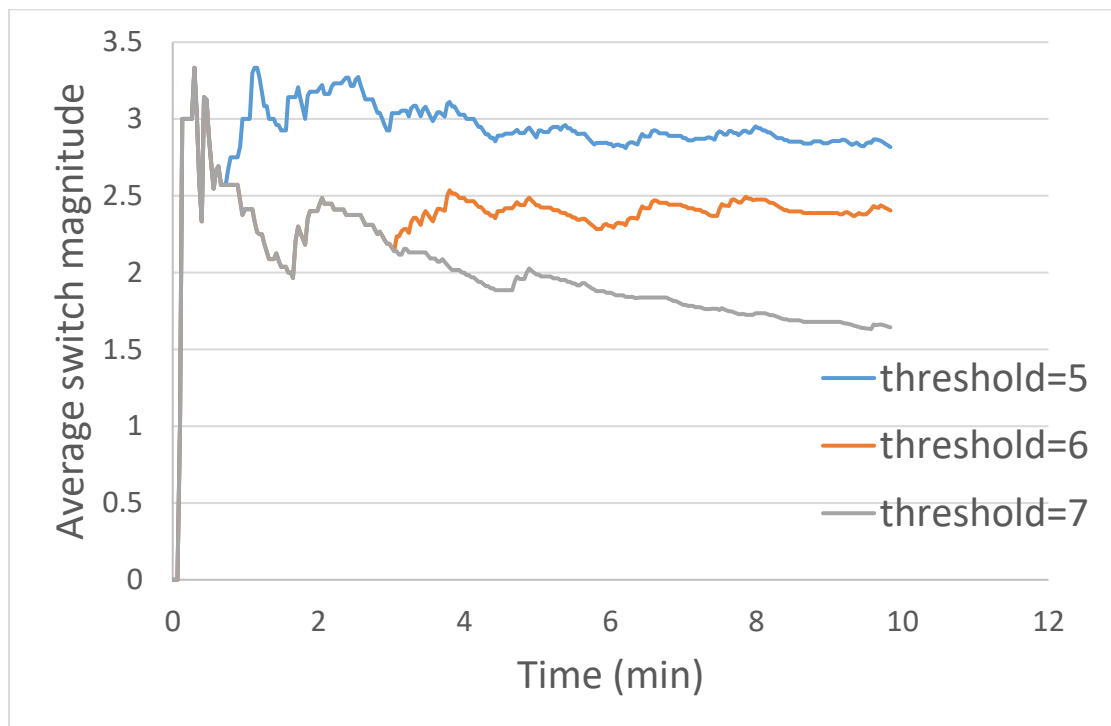


Εικόνα 25. Average switch magnitude (Μοντέλο 1, Πείραμα 3)

## Μοντέλο 2

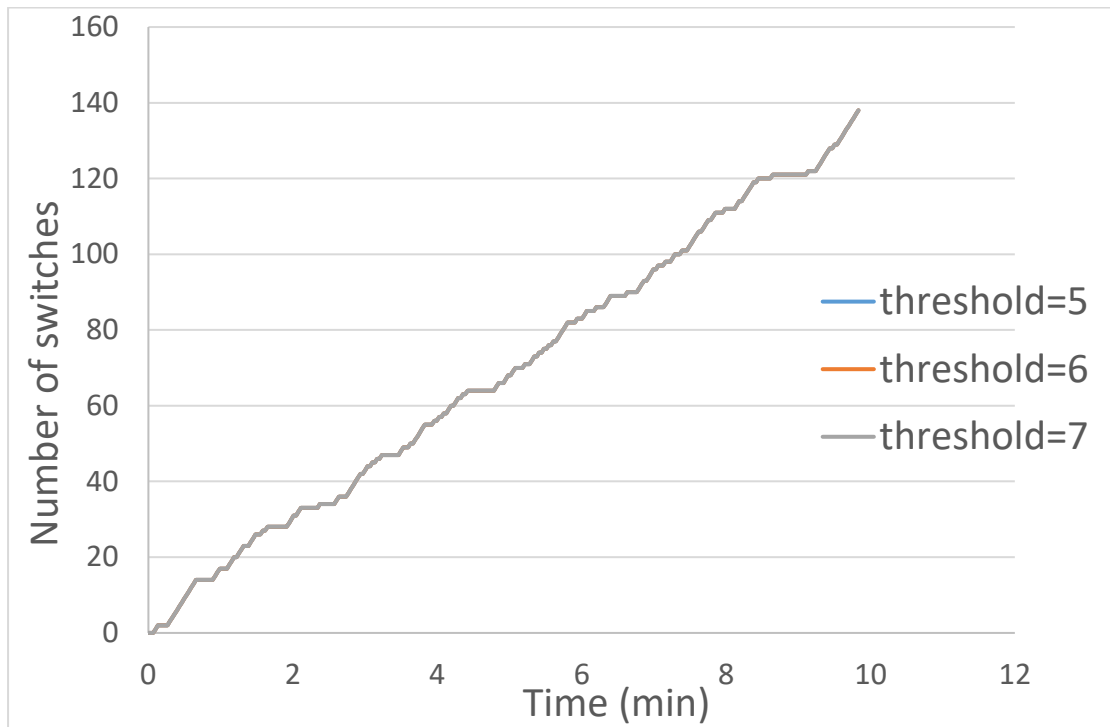


Εικόνα 26. Number of swithes(Μοντέλο 2, Πείραμα 3)

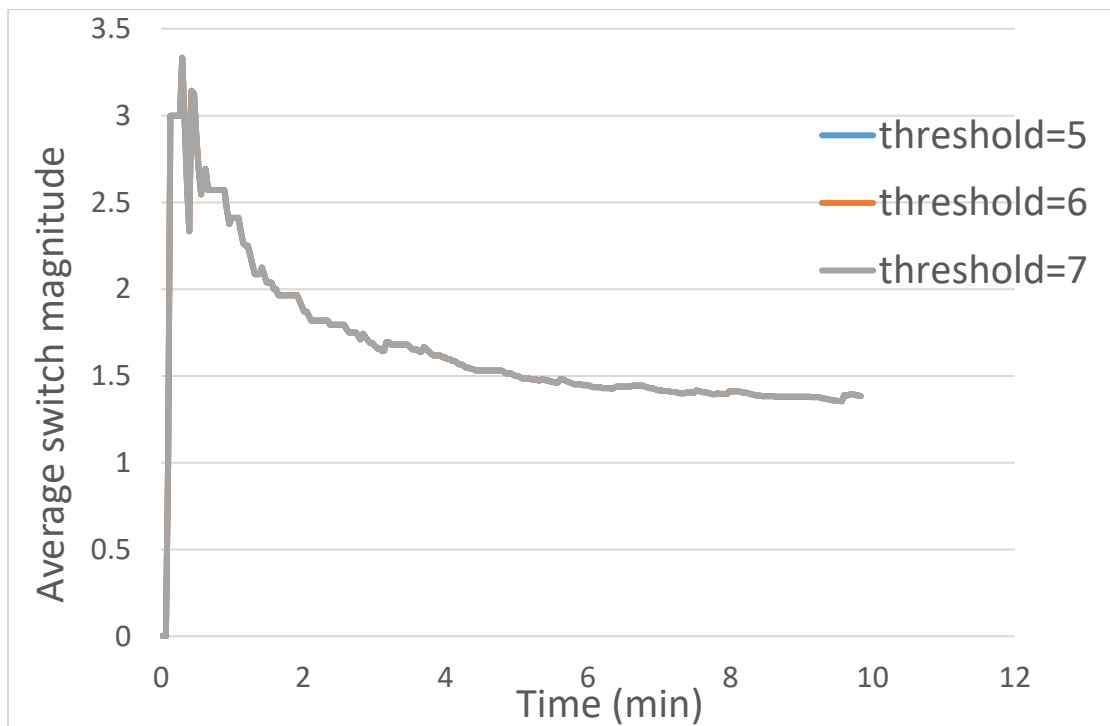


Εικόνα 27. Average switch magnitude(Μοντέλο 2, Πείραμα3)

### Μοντέλο 3



Εικόνα 28. Number of switches (Μοντέλο 3, Πείραμα 3)

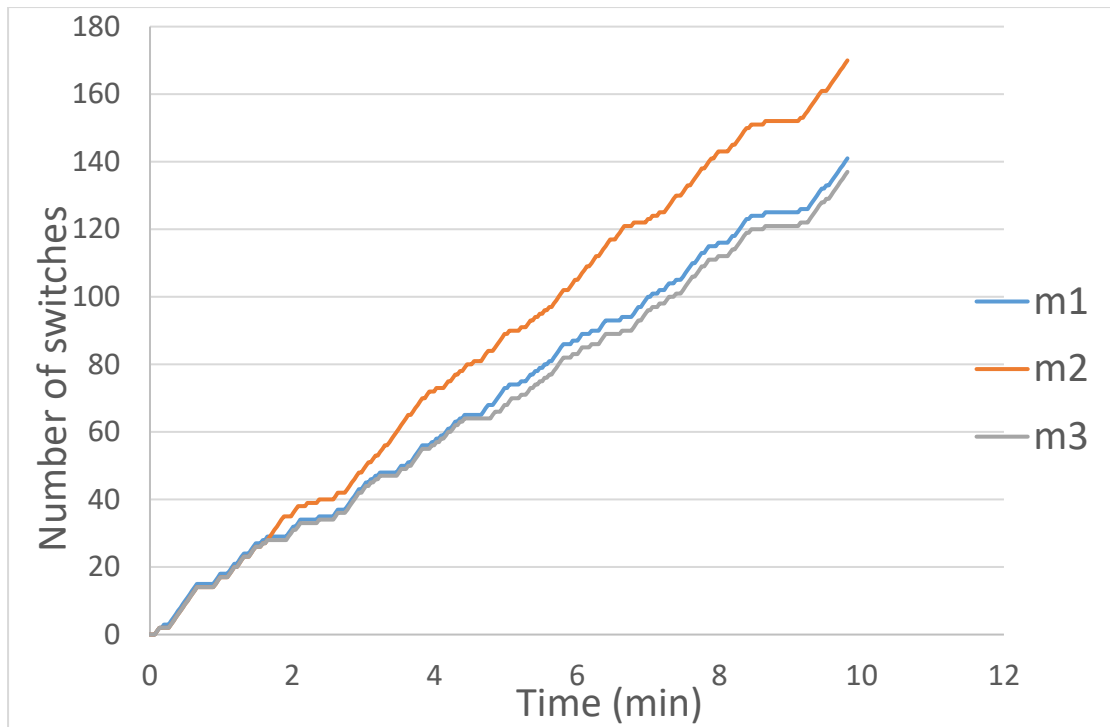


Εικόνα 29. Average switch magnitude (Μοντέλο 3, Πείραμα 3)

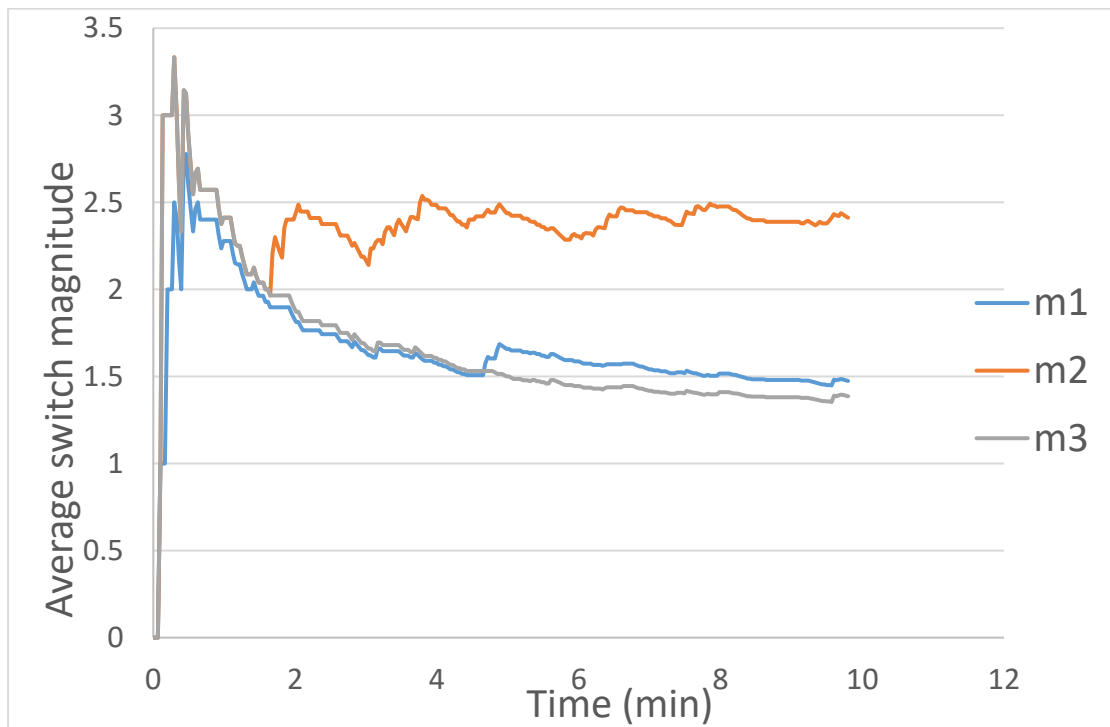
Όπως περιμέναμε τα αποτελέσματα για  $\text{threshold} = 7$  μας φέρουν καλύτερα αποτελέσματα από  $\text{threshold} = 6$  που μας φέρουν καλύτερα αποτελέσματα από  $\text{threshold} = 5$ . Αυτό είναι αναμενόμενο καθώς κάνουμε αυστηρότερο τον μηχανισμό με την αύξηση του  $\text{threshold}$ . Στο μοντέλο 3 ωστόσο δεν παρατηρούμε καμία διαφορά για τις τρεις αυτές τιμές, αυτό οφείλεται στο γεγονός ότι η τιμή του  $\text{reputation}$  και στις τρεις περιπτώσεις ξεπερνάει το 5 όσες φορές ξεπερνάει και το 6 και το 7. Στην προσομοίωση μας ο επιτιθέμενος καταφέρνει να ξεγελάσει αυτό το μοντέλο και να πετύχει υψηλή τιμή  $\text{reputation}$  μία φορά και μάλιστα η τιμή του  $\text{reputation}$  γίνεται μεγαλύτερη από 7. Όλες τις επόμενες φορές που προσπαθεί να ξεγελάσει τον router με καλόβουλη συμπεριφορά η τιμή του  $\text{reputation}$  δεν φτάνει σε επίπεδα μεγαλύτερα του 5. Αν έφτανε σε επίπεδο μεγαλύτερο του 5 αλλά μικρότερο του 7 τότε στο διάγραμμά μας θα εμφανιζόντουσαν οι διαφορές ανάμεσα στις δύο προσεγγίσεις. Πράγμα που συμβαίνει στα άλλα δύο μοντέλα και το βλέπουμε να αποτυπώνεται στα διαγράμματα.

Παρακάτω θα συγκρίνουμε τα μοντέλα μεταξύ τους για διαφορετικές τιμές του  $\text{threshold}$ . Η σύγκριση των μοντέλων για  $\text{threshold} = 5$  παρουσιάστηκε παραπάνω στο πείραμα 1, εικόνες 15 και 16.

#### Σύγκριση Μοντέλων ( $\text{threshold}=6$ )



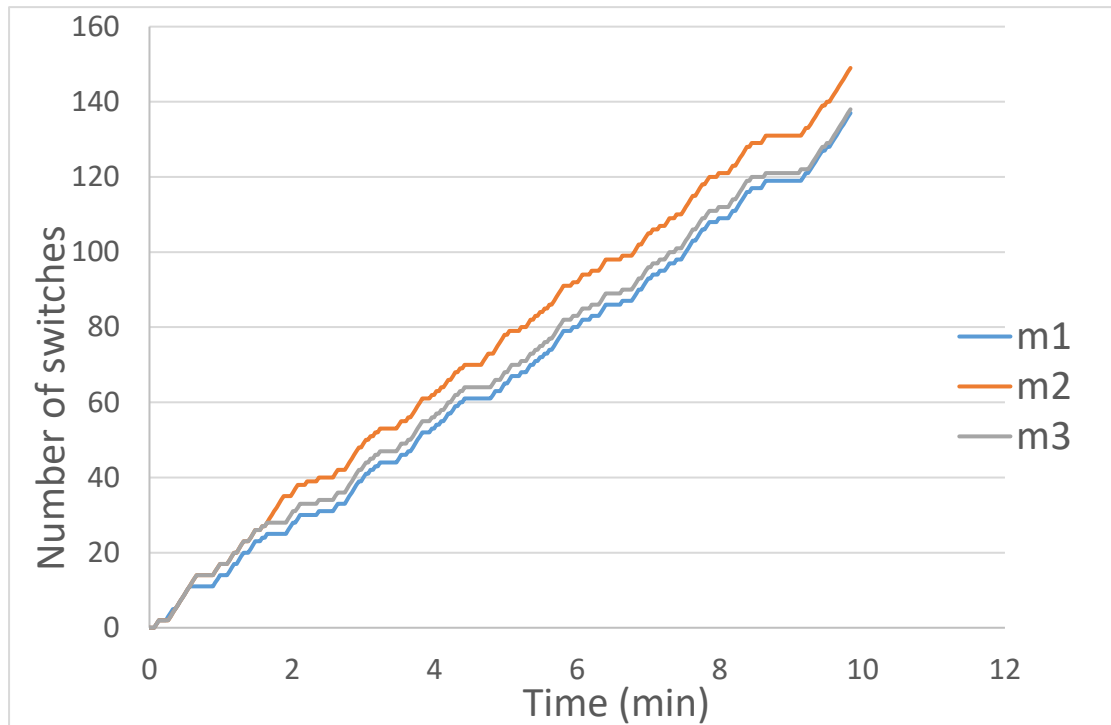
Εικόνα 30. Number of switches (Σύγκριση Μοντέλων, Πείραμα 3)



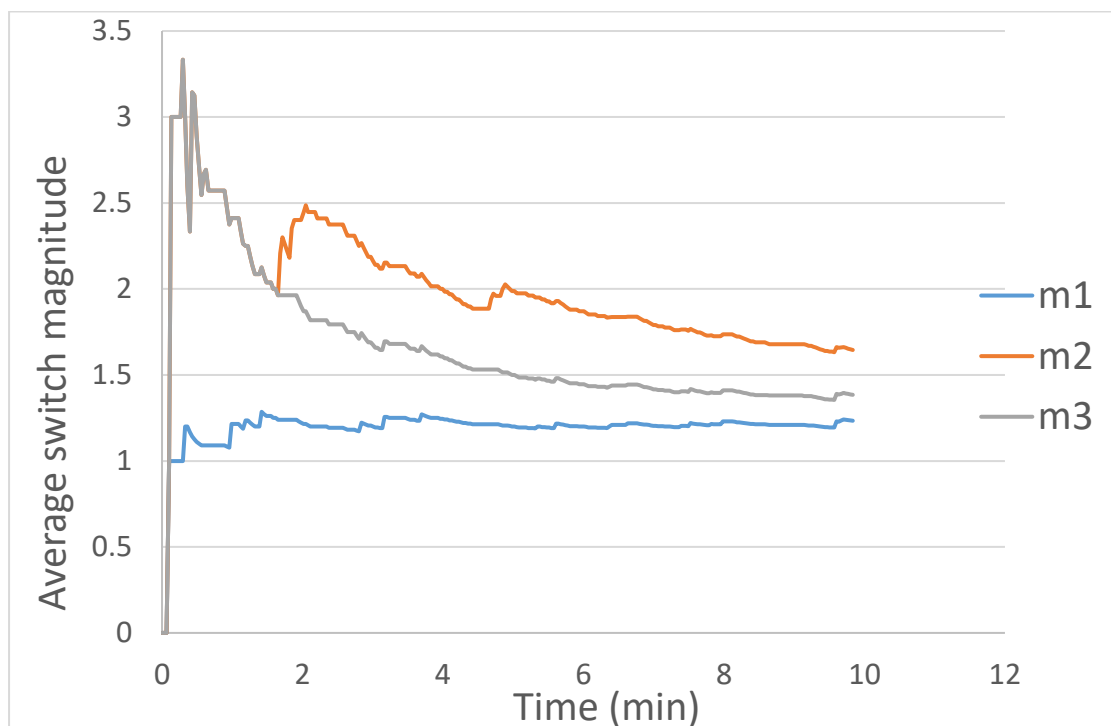
Εικόνα 31. Average switch magnitude (Σύγκριση Μοντέλων, Πείραμα 3)

Παρατηρούμε ότι το μοντέλο 3 και 1 έχουν πολύ καλύτερα αποτελέσματα συγκριτικά με το μοντέλο 2, ως προς τον αριθμό των αλλαγών και την ένταση αυτών. Ακόμη σημειώνουμε ότι το μοντέλο 3 για threshold = 6 παρουσιάζει ελαφρώς καλύτερα αποτελέσματα από το μοντέλο 1.

### Σύγκριση Μοντέλων (threshold=7)



Εικόνα 32. Number of switches (Σύγκριση Μοντέλων, Πείραμα 3)



Εικόνα 33. Average switch magnitude (Σύγκριση Μοντέλων, Πείραμα 3)

Τέλος για reputation = 0.7 παρατηρούμε ότι και πάλι τα αποτελέσματα των μοντέλων 1 και 3 είναι κατά πολύ καλύτερα από αυτά του μοντέλου 2. Επίσης το

μοντέλο 1 φαίνεται να τα πηγαίνει ελαφρώς καλύτερα και από το μοντέλο 3 για αυτή την τιμή του reputation.

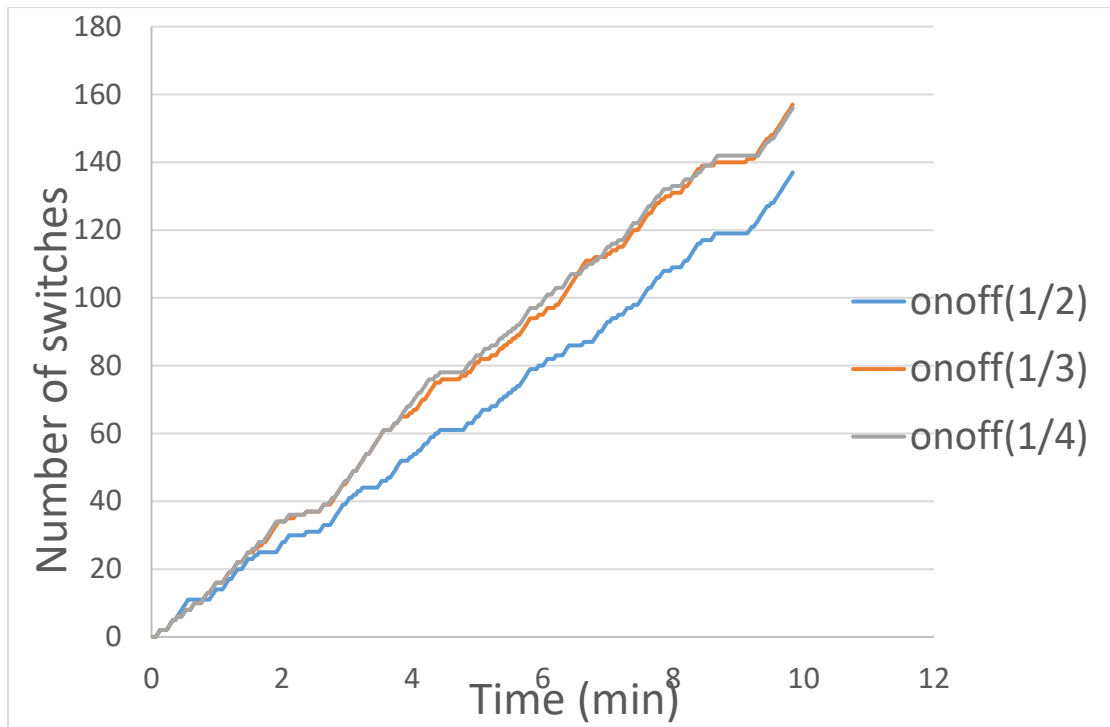
## 6.4 Πείραμα 4

Επειδή στο προηγούμενο πείραμα που εκτελέστηκε καταλήξαμε στο συμπέρασμα ότι τα μοντέλα παρουσιάζουν καλύτερα αποτελέσματα για  $\text{threshold} = 7$  (εκτός από το μοντέλο 3 το οποίο παρουσιάζει τα ίδια ακριβώς αποτελέσματα) η συνέχεια των πειραμάτων θα πραγματοποιηθεί με τιμή  $\text{threshold} = 7$ .

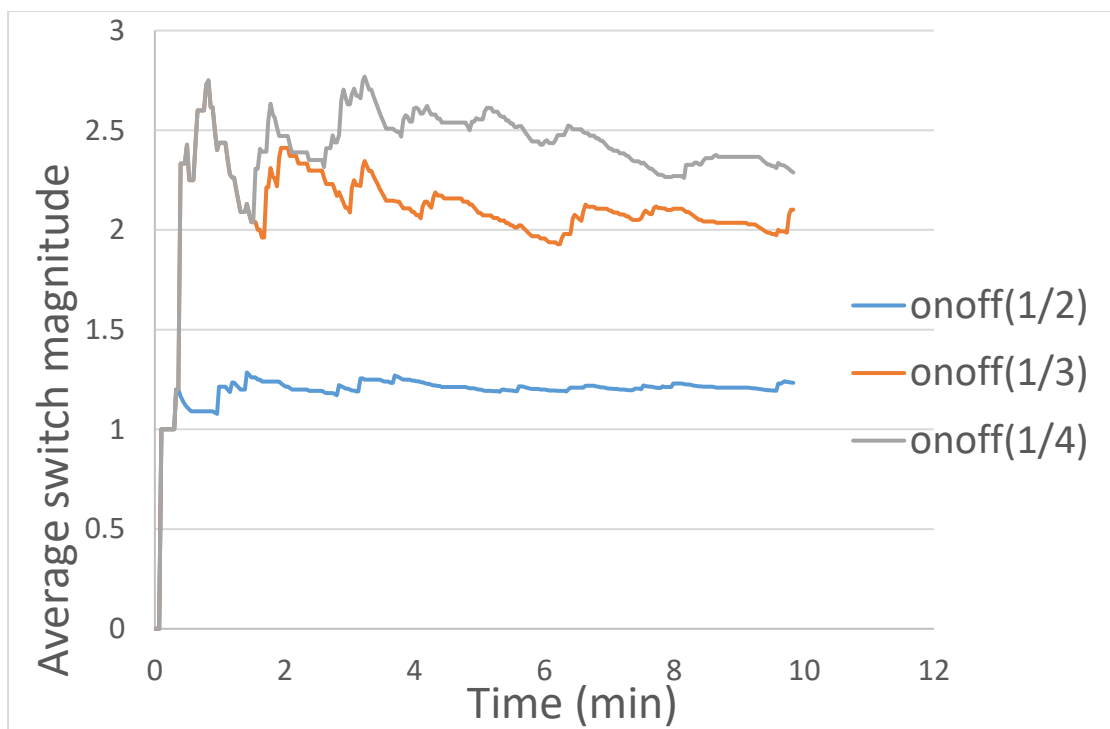
Σε αυτό το πείραμα εξετάζουμε την συμπεριφορά του κάθε μοντέλου όταν μεταβάλλουμε τον λόγο on/off του attacker. Αλλά και μεταξύ τους πιο συμπεριφέρεται αποδοτικότερα. Ορίζουμε και παίρνουμε αποτελέσματα για λόγο on/off:  $\frac{1}{2}$ ,  $\frac{1}{3}$ ,  $\frac{1}{4}$

Ο λόγος on/off όπως περιγράψαμε εκφράζει την χρονική διάρκεια που ο επιτιθέμενος συμπεριφέρεται θετικά (δηλαδή ζητάει διαδοχικά τμήματα του βίντεο) ως προς την χρονική διάρκεια που συμπεριφέρεται αρνητικά (δηλαδή ζητάει μη διαδοχικά τμήματα).

### Μοντέλο 1



Εικόνα 34. Number of switches (Μοντέλο 1, Πείραμα 4)

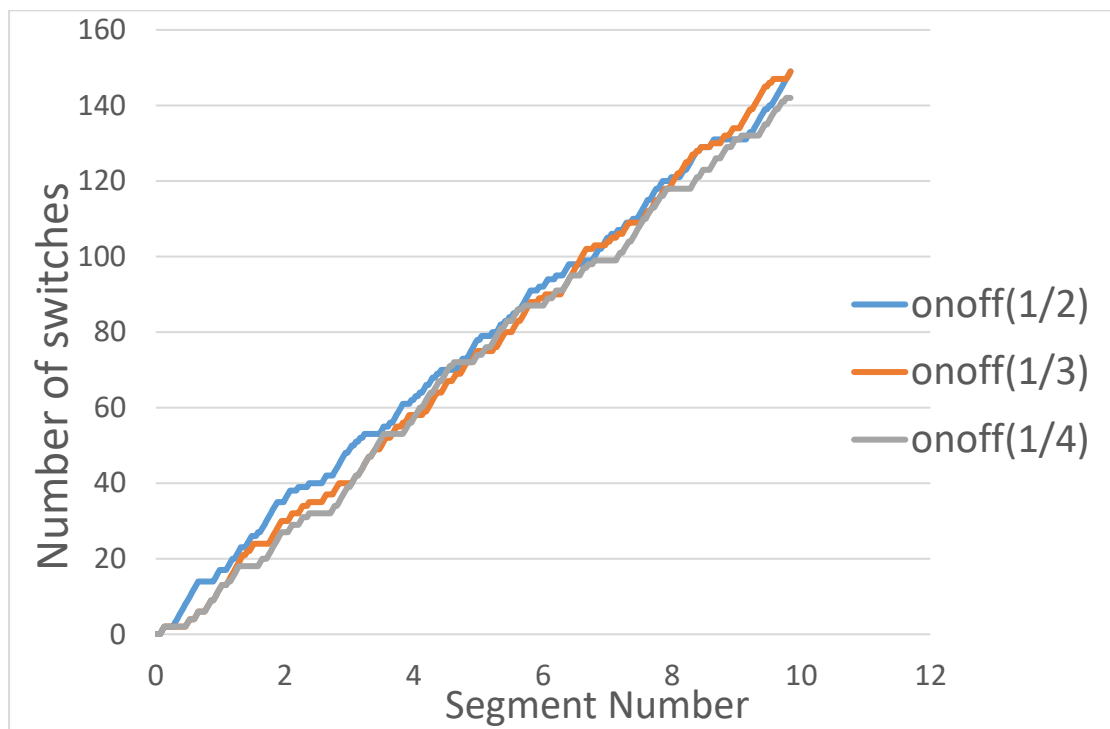


Εικόνα 35. Average switch magnitude (Μοντέλο 1, Πείραμα 4)

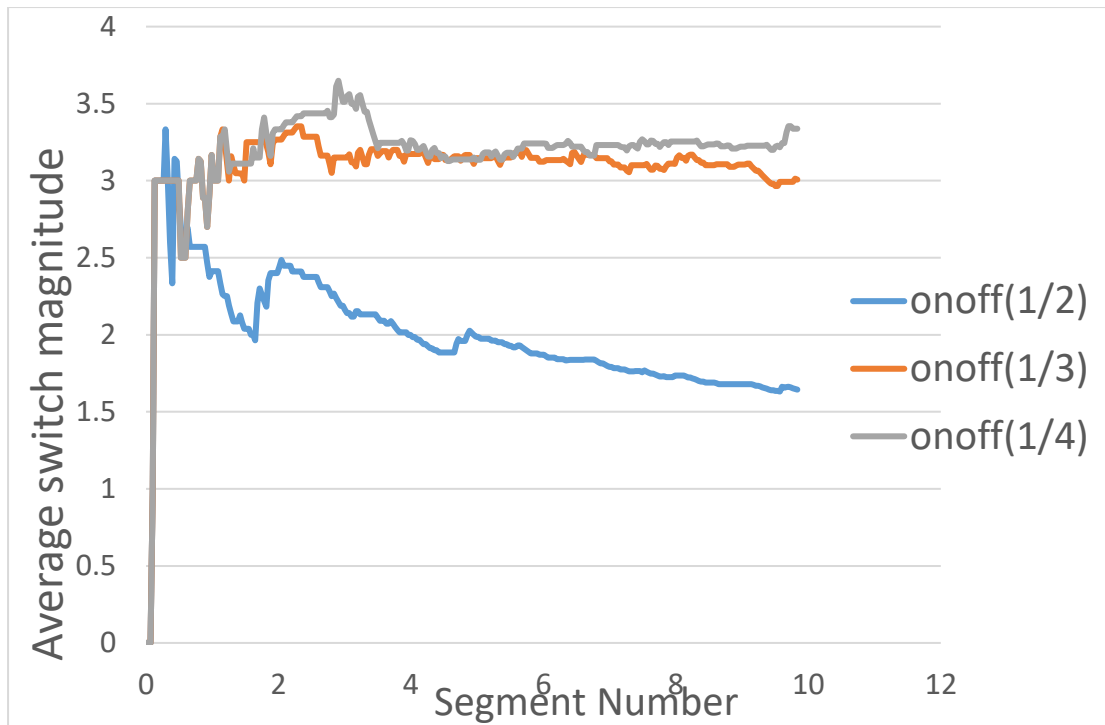
Παρατηρούμε ότι ο επιτιθέμενος καταφέρνει να πετύχει καλύτερα αποτελέσματα για λόγο 1/3 και 1/4. Παραπλανά το μοντέλο μας και καταφέρνει να βαθμολογηθεί με μεγαλύτερο reputation και να αποθηκεύσει μη διαδοχικά πακέτα. Έτσι έχουμε αρκετά χειρότερη εμπειρία χρήστη. Για λόγο on/off = 1/4 παρατηρούμε

μεγαλύτερης έντασης αλλαγές που θα γίνονται αντιληπτές από τον χρήστη και θα προκαλούν χειρότερη εμπειρία.

## Μοντέλο 2



Εικόνα 36. Number of switches (Μοντέλο 2, Πείραμα 4)



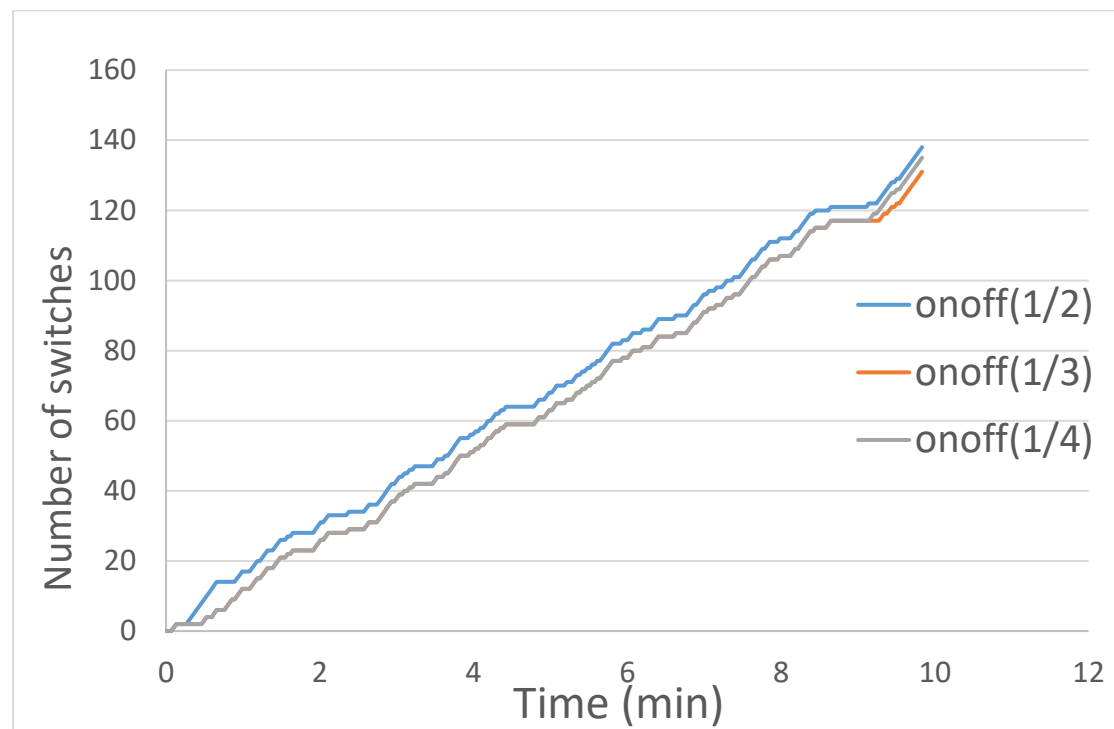
Εικόνα 37. Average switch magnitude (Μοντέλο 2, Πείραμα 4)

Το μοντέλο άμυνας τα καταφέρνει εξίσου καλά και στις τρεις εκδοχές επίθεσης όσον αφορά τον αριθμό των αλλαγών. Τελικά όμως φαίνεται ότι ο επιτιθέμενος με λόγο on/off  $\frac{1}{4}$  και  $\frac{1}{3}$  καταφέρνει και εδώ να κάνει μεγαλύτερη ζημιά στην τελική εμπειρία, καθώς ενώ προκαλεί τον ίδιο αριθμό αλλαγών πρόκειται για αλλαγές πολύ μεγαλύτερης έντασης.

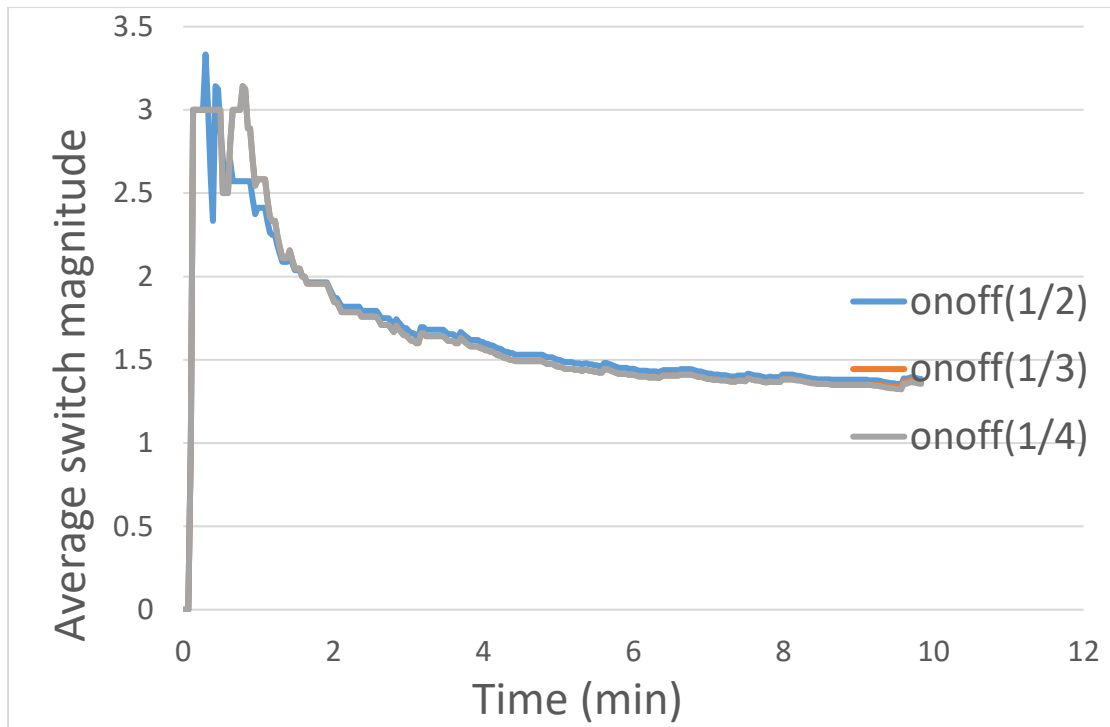
Παρατηρούμε ακόμα ότι για λόγο επίθεσης on/off  $\frac{1}{3}$  έχουμε 67% θετική συμπεριφορά (και 33% αρνητική συμπεριφορά) οπότε η τιμή του reputation συγκλίνει στην τιμή 6.7 ενώ για λόγο on/off  $\frac{1}{4}$  που έχουμε 75% θετική συμπεριφορά η τιμή του reputation συγκλίνει στο 7.5 . Γιατί όμως δεν βλέπουμε περισσότερες αλλαγές; Η εξήγηση βρίσκεται στο γεγονός ότι για μεγαλύτερο λόγο on/off έχουμε περισσότερη διάρκεια καλής συμπεριφοράς άρα και περισσότερο καλό περιεχόμενο(χωρίς κενά) στην cache του router. Μπορεί δηλαδή να επιτρέπουμε όντως περισσότερες φορές την αποθήκευση από τον router και την μετάδοση μετά αυτού του περιεχομένου, αλλά τις περισσότερες φορές αυτό θα είναι συνεχόμενα πακέτα. Όταν δεν είναι συνεχόμενα θα συμβαίνουν μεγάλες αλλαγές. Έτσι δικαιολογούνται και οι διαφορετικές εντάσεις, παρότι έχουμε τον ίδιο αριθμό αλλαγών.

Ο λόγος  $1/3$  και  $1/4$  δεν φαίνεται να παρουσιάζουν σημαντικές διαφορές μεταξύ τους, σχετικά με τα αποτελέσματα που φέρουν στην τελική εμπειρία.

### Μοντέλο 3



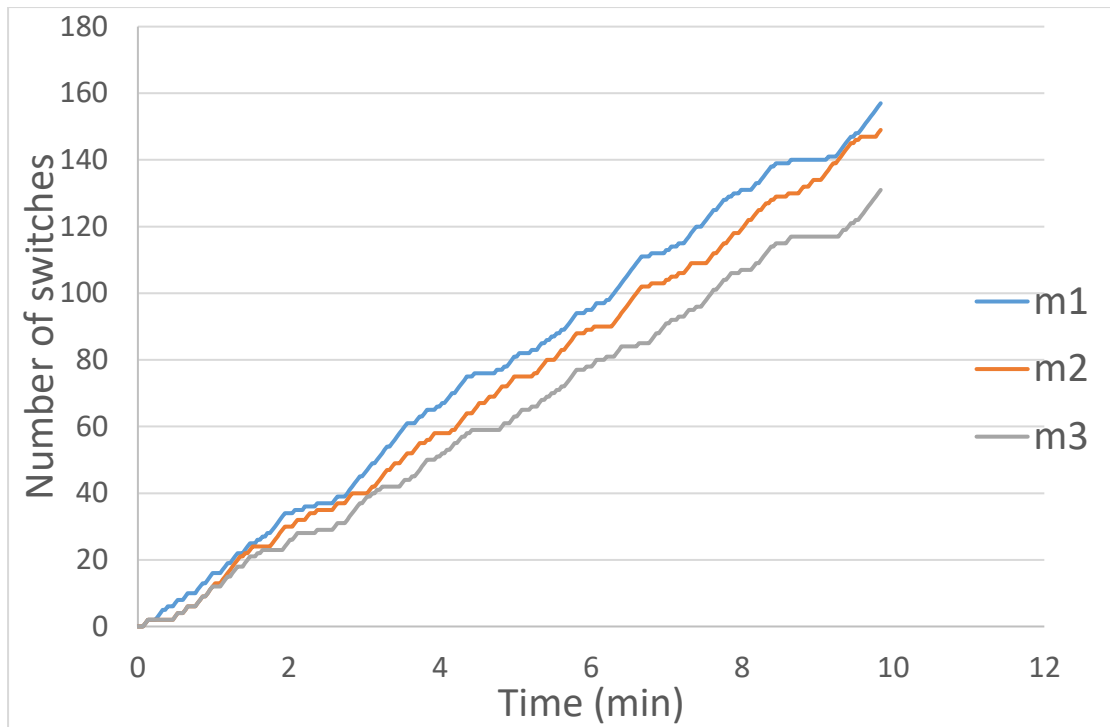
Εικόνα 38. Number of switches (Μοντέλο 3, Πείραμα 4)



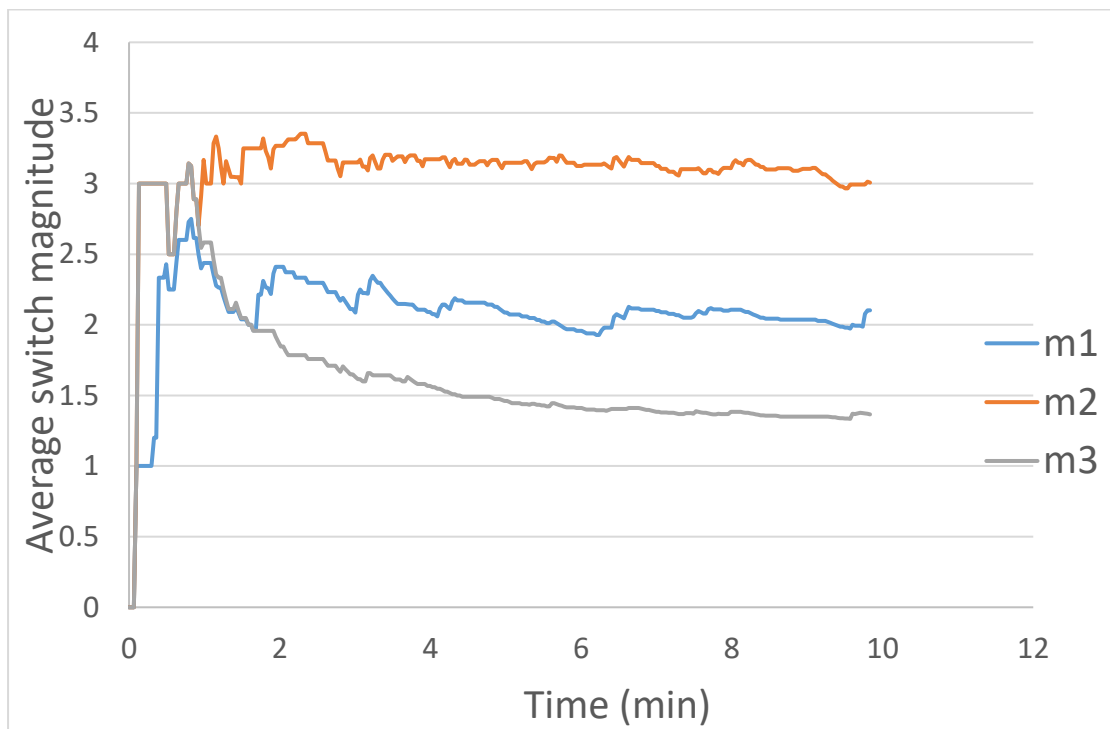
Εικόνα 39. Average switch magnitude (Μοντέλο 3, Πείραμα 4)

Εδώ το μοντέλο άμυνας ανταποκρίνεται εξαιρετικά και τα πάει εξίσου καλά και στις τρεις προσπάθειες του επιτιθέμενου. Παρατηρούμε διαφορές αλλά ανάξιες σχολιασμού. Η επιθέσεις με λόγο 1/2 και 1/3 καταφέρνουν να περάσουν μόνο μια φορά στην αρχή το threshold του reputation και να κάνουν αποθηκεύσουν περιεχόμενο που θα δημιουργήσει κακή εμπειρία στην μετέπειτα κατανάλωση του από το καλόβουλο χρήστη. Ενώ η επίθεση με λόγο 1/4 καταφέρνει 2 μία στην αρχή και μία στο τέλος του βίντεο.

#### Σύγκριση Μοντέλων (on/off= 1/3 )



Εικόνα 40. Number of switches (Σύγκριση μοντέλων, Πείραμα 4, onoff=1/3)

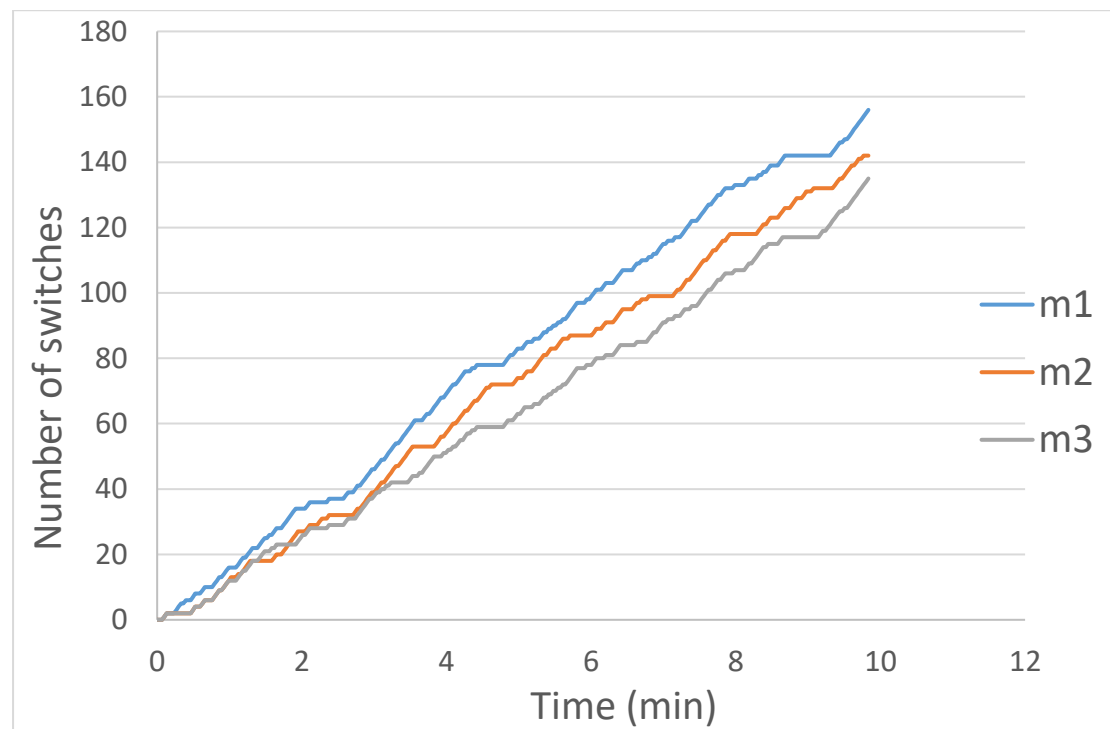


Εικόνα 41. Average switch magnitude (Σύγκριση μοντέλων, Πείραμα 4, onoff=1/3)

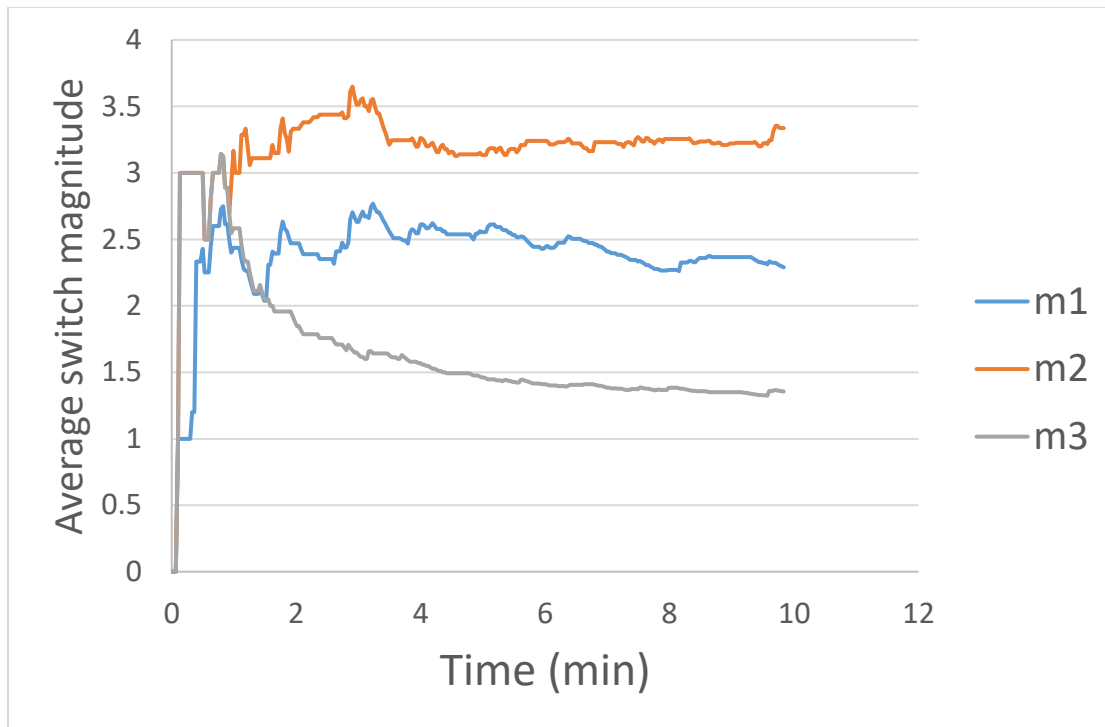
Στη σύγκριση των μοντέλων μεταξύ τους παρατηρούμε διαφορετικές συμπεριφορές για τα 3 μοντέλα. Το μοντέλο 3 συμπεριφέρεται καλύτερα από τα άλλα δύο, πετυχαίνοντας χαμηλότερες τιμές και στις δύο μετρικές. Οπότε με μικρότερο

αριθμό αλλαγών και μικρότερης έντασης χαρακτηρίζεται ως το πιο επιτυχημένο για λόγο επιτιθέμενου  $on/off = 1/3$ . Στο μοντέλο 1 συμβαίνουν ελαφρώς περισσότερες αλλαγές από το μοντέλο 2 αλλά η ένταση αυτών είναι εμφανώς μικρότερη. Οπότε κρίνεται ελαφρώς πιο πετυχημένο από το μοντέλο 2.

#### Σύγκριση Μοντέλων ( $on/off = 1/4$ )



Εικόνα 42. Number of switches (Σύγκριση μοντέλων, Πείραμα 4,  $on/off=1/4$ )



Εικόνα 43. Average switch magnitude (Σύγκριση μοντέλων, Πείραμα 4, on/off=1/4)

Όταν ο επιτιθέμενος έχει λόγο on/off =  $\frac{1}{4}$  παρατηρούμε και πάλι ότι το μοντέλο άμυνας με τις καλύτερες επιδόσεις είναι το 3. Με λιγότερες αλλαγές κατά τη διάρκεια και χαμηλότερης μέσης έντασης ανά αλλαγή. Όσο αφορά τα άλλα δύο μοντέλα στο μοντέλο 2 παρατηρούμε λιγότερες αλλαγές αλλά μεγαλύτερης έντασης.

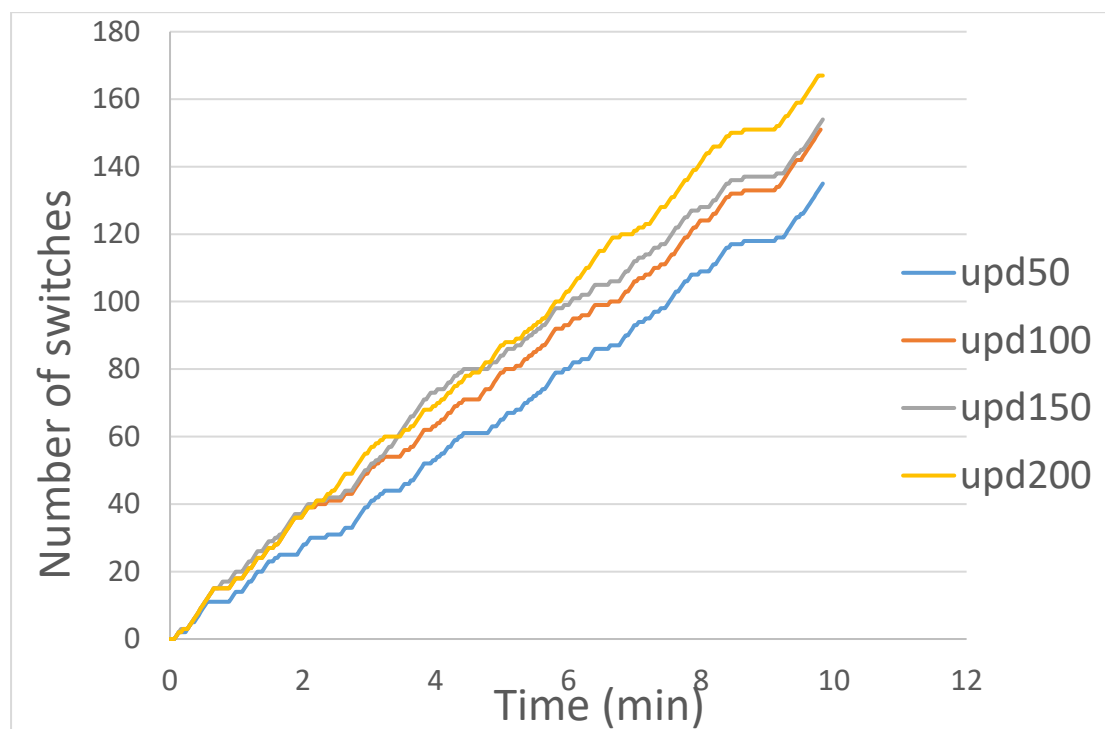
Συνολικά το μοντέλο 3 είναι πιο ανθεκτικό σε επιθέσεις on/off λόγω της παραμέτρου  $f$ . Λόγω της οποίας ο router γίνεται αρνητικά προκατειλημμένος αν η τιμή του reputation είναι μικρότερη από το threshold (πιθανός attacker) και δίνει λιγότερο βάρος στην νέα βαθμολογία και μεγαλύτερο βάρος στην παλιά (στην έως τώρα βαθμολογία). Κάνοντας έτσι δυσκολότερο να ανέβει ξανά η τιμή του reputation πάνω από το threshold. Όταν όμως το reputation είναι μεγαλύτερο του threshold δίνει μεγαλύτερο βάρος στην νέα βαθμολογία και μικρότερο στην παλιά. Έτσι ώστε αν πάει να τον παραπλανήσει ο attacker με καλή συμπεριφορά να μπορέσει να τον εντοπίσει ξανά, άμεσα. Μπορούμε ακόμα να πούμε ότι το μοντέλο 1 συμπεριφέρεται καλύτερα από το μοντέλο 2 για λόγο on/off =  $\frac{1}{2}$  ενώ στις προσομοιώσεις με on/off  $\frac{1}{3}$  και  $\frac{1}{4}$  το μοντέλο 1 παρουσιάζει περισσότερες αλλαγές αλλά μικρότερης έντασης. Αυτό σημαίνει ότι η τιμή του reputation στο μοντέλο 1 ξεπερνάει περισσότερες φορές την τιμή του threshold και γίνεται χρήση της προσωρινής αποθήκευσης πιο συχνά συγκριτικά με το μοντέλο 2.

## 6.5 Πείραμα 5

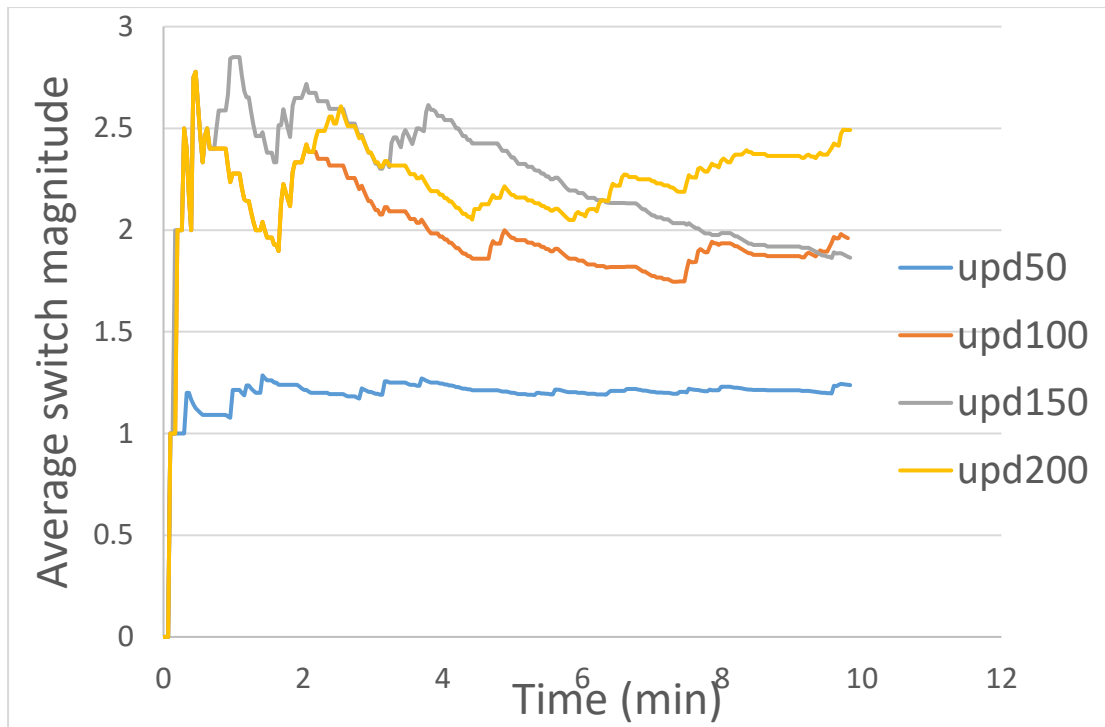
Στο πείραμα αυτό εξετάζουμε τα μοντέλα μας για διαφορετικούς χρόνους update. Πιο συγκεκριμένα για 50, 100, 150 και 200 sec. Επίσης εξετάζουμε και την περίπτωση της ενημέρωσης με κάθε πακέτο δεδομένων που λαμβάνει ο router, την ονομάζουμε real-time ενημέρωση. Γίνεται αντιληπτό ότι όσο συχνότερη η ενημέρωση τόσο μεγαλύτερη πολυπλοκότητα προσθέτουμε στο σύστημα μας εμείς όμως εδώ εξετάζουμε τα αποτελέσματα καθαρά από την σκοπιά της αποτελεσματικότητας σε επίπεδο περιορισμού της επίθεσης.

Επιλέχθηκε να ελεγχθούν τα μοντέλα για  $\text{threshold} = 5$  με  $\text{on/off} = \frac{1}{2}$  και για  $\text{threshold} = 7$  με  $\text{on/off} = \frac{1}{3}$ , επειδή αυτές είναι δύο δύσκολες καταστάσεις. Επιλέχθηκαν για αυτό το σκοπό φυσικά οι καλύτερες εκδοχές του κάθε μοντέλου όπως και στα προηγούμενα πειράματα.

Μοντέλο 1 ( $\text{threshold} = 5$ ,  $\text{on/off} = 1/2$ )



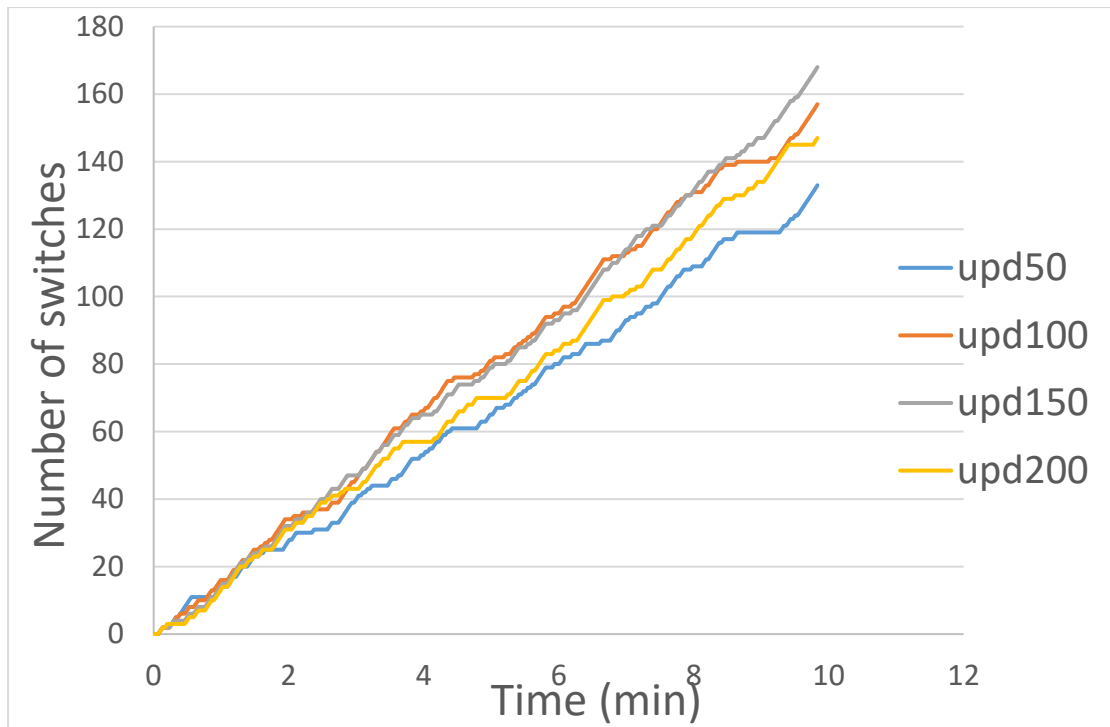
Εικόνα 44. Number of switches (Μοντέλο 1, Πείραμα 5,  $\text{threshold} = 5$ ,  $\text{on/off} = 1/2$ )



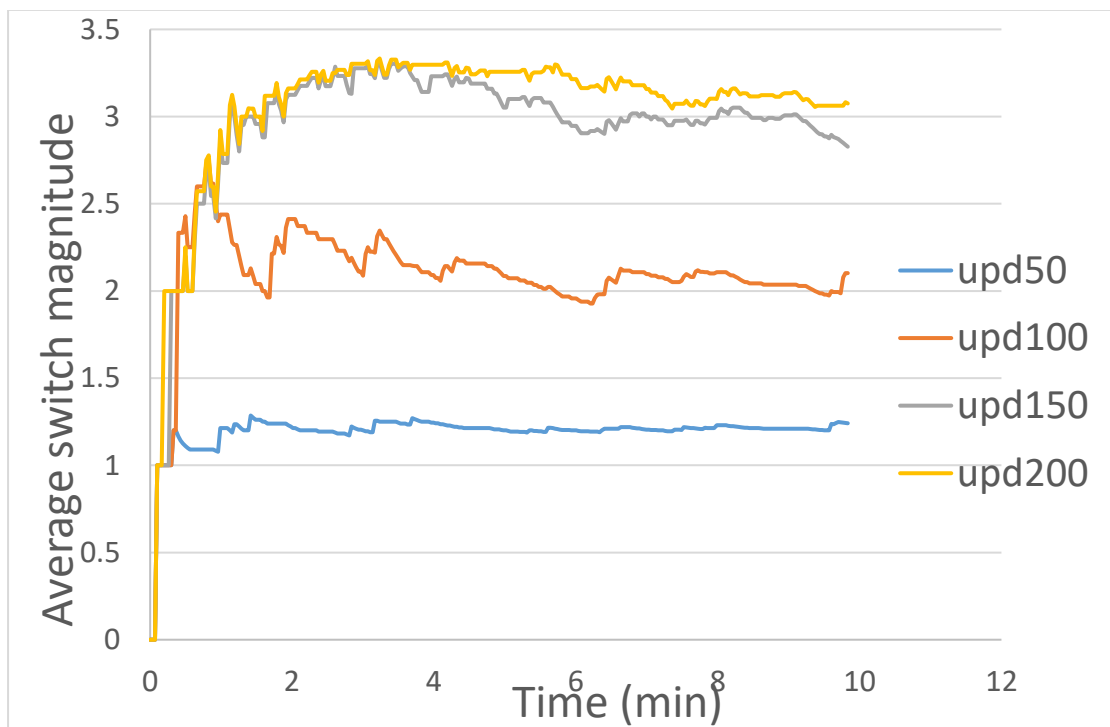
Εικόνα 45. Average switch magnitude (Μοντέλο 1, Πείραμα 5, threshold=5, on/off=1/2)

Στο μοντέλο 1 παρατηρούμε ότι η ανανέωση κάθε 50 sec προστατεύει τον χρήστη αποτελεσματικότερα ενώ η ανανέωση κάθε 200 sec παρέχει την μικρότερη ασφάλεια. Οι ανανεώσεις ανά 100 και 150 sec παρουσιάζουν κοντινά αποτελέσματα, με την ανανέωση ανά 100 sec να τα πηγαίνει καλύτερα στην μεγαλύτερη διάρκεια του βίντεο. Τελικά όμως αν ο χρήστης καταναλώσει ολόκληρο το βίντεο θα έχουν σχεδόν ίδιο αριθμό αλλαγών και ίδιας έντασης.

Μοντέλο 1 (threshold = 7 , on/off = 1/3)

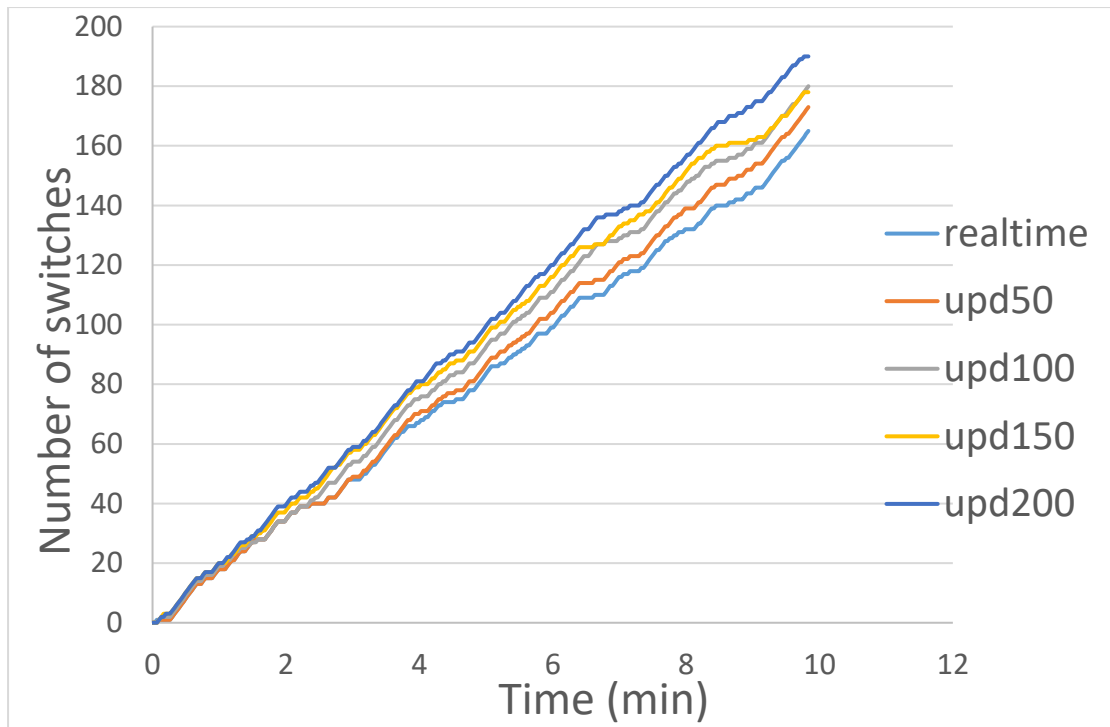


Εικόνα 46. Number of switches (Μοντέλο 1, Πείραμα 5, threshold=7, onoff=1/3)

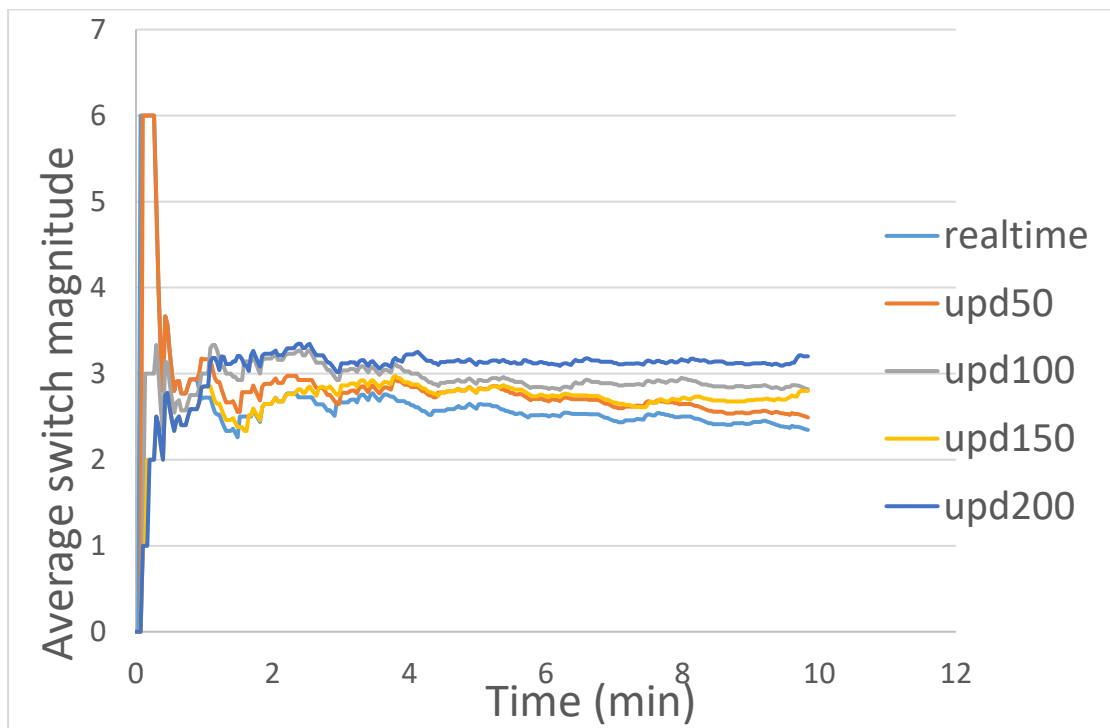


Εικόνα 47. Average switch magnitude (Μοντέλο 1, Πείραμα 5, threshold=7, onoff=1/3)

Μοντέλο 2 (threshold = 5 , on/off = 1/2)



Εικόνα 48. Number of switches (Μοντέλο 2, Πείραμα 5, threshold=5, onoff=1/2)

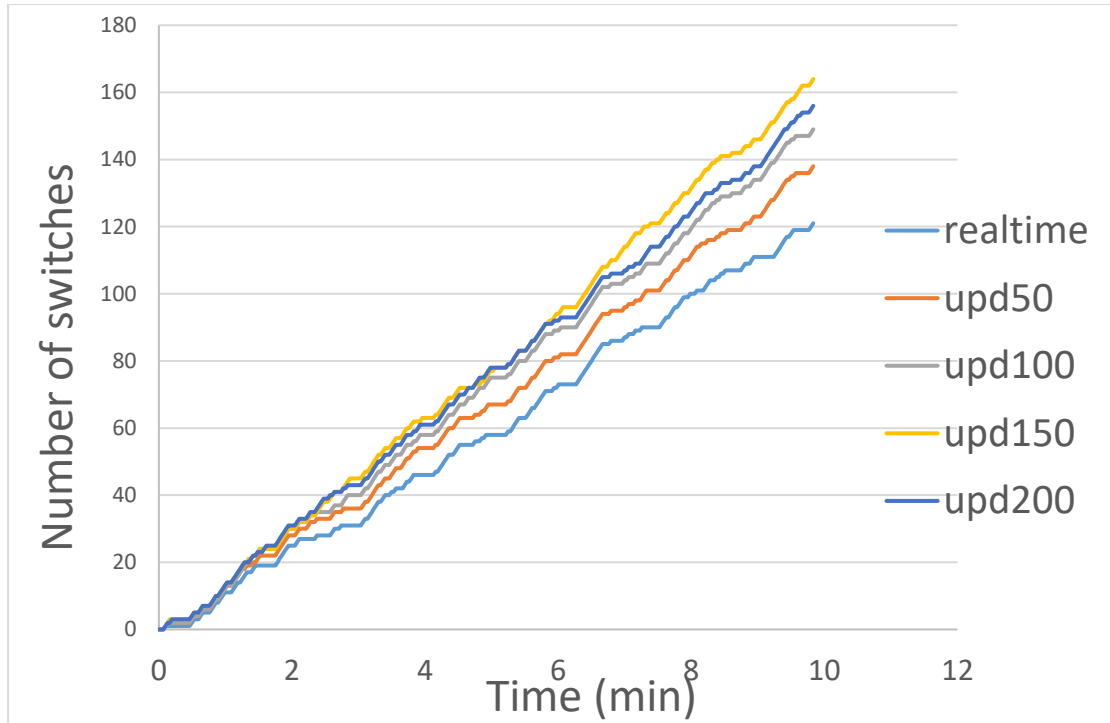


Εικόνα 49. Average switch magnitude (Μοντέλο 2, Πείραμα 5, threshold=5, onoff=1/2)

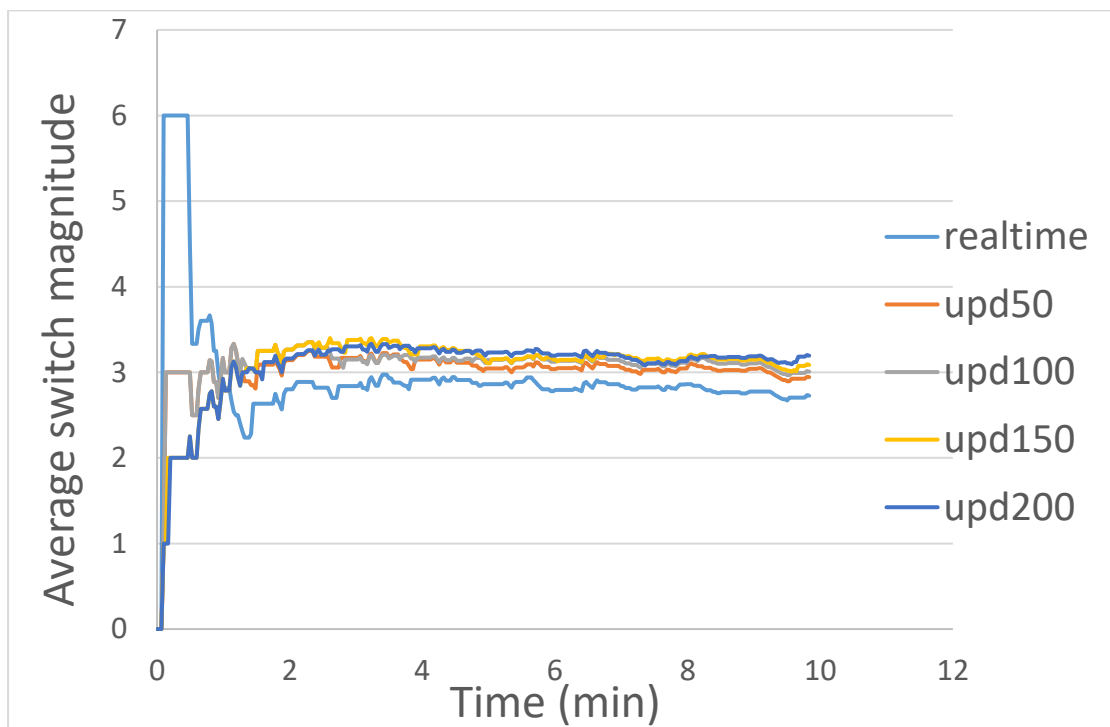
Στην προσομοίωση αυτή παρατηρούμε σημαντικά χειρότερα αποτελέσματα για ανανέωση κάθε 200sec. Επίσης ένας χρήστης που θα καταναλώσει ολόκληρο το βίντεο θα έχει ίδια εμπειρία όσον αφορά τις περιπτώσεις που ανανεώνουμε ανά 150 και 100sec. Οπότε αν είχαμε να επιλέξουμε ανάμεσα σε αυτές τις 2 τιμές θα

επιλέγαμε την ανανέωση ανά 150 για να μην επιβαρύνουμε το σύστημα μας με περιττό υπολογιστικό φορτίο.

Μοντέλο 2 (threshold = 7 , on/off = 1/3)



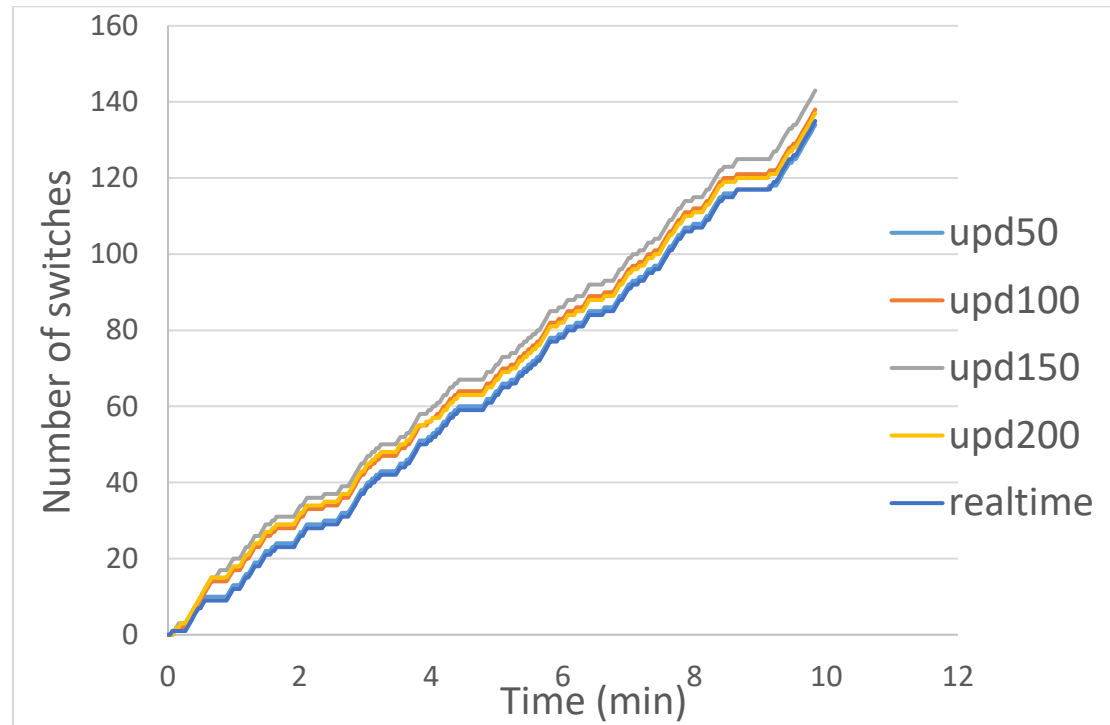
Εικόνα 50. Number of switches (Μοντέλο 2, Πείραμα 5, threshold=7, on/off=1/3)



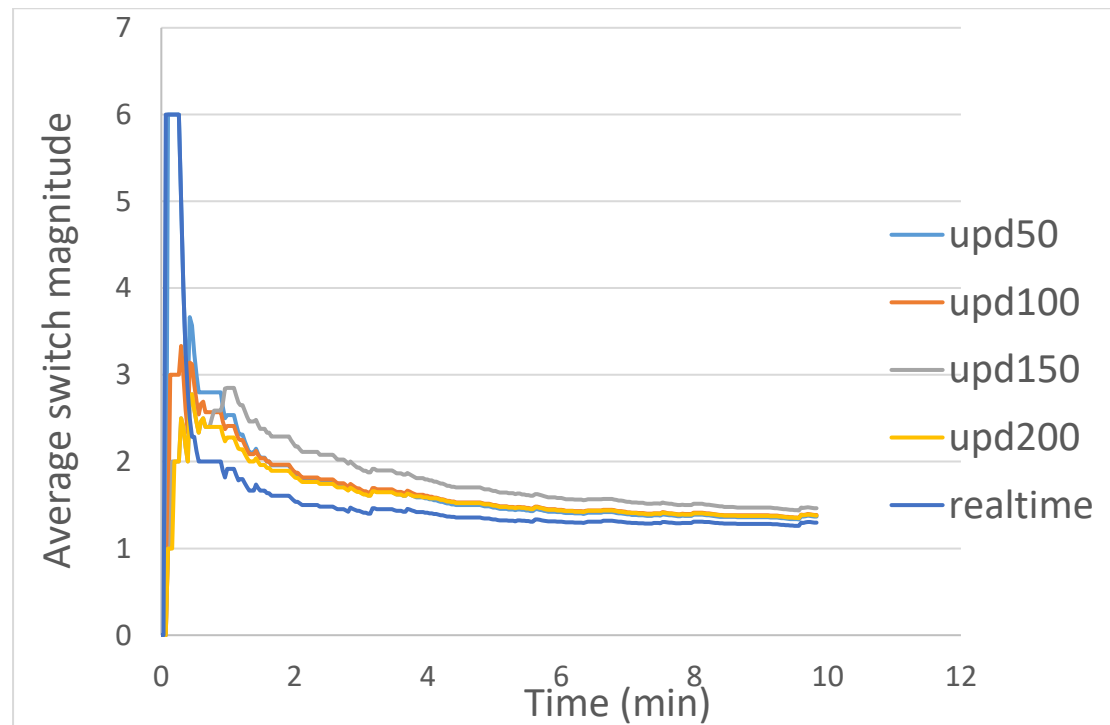
Εικόνα 51. Average switch magnitude (Μοντέλο 2, Πείραμα 5, threshold=7, on/off=1/3)

Στην προσομοίωση αυτή παρατηρούμε και πάλι σημαντικά καλύτερα αποτελέσματα για ανανέωση σε real-time.

Μοντέλο 3 (threshold = 5 , on/off = 1/2)



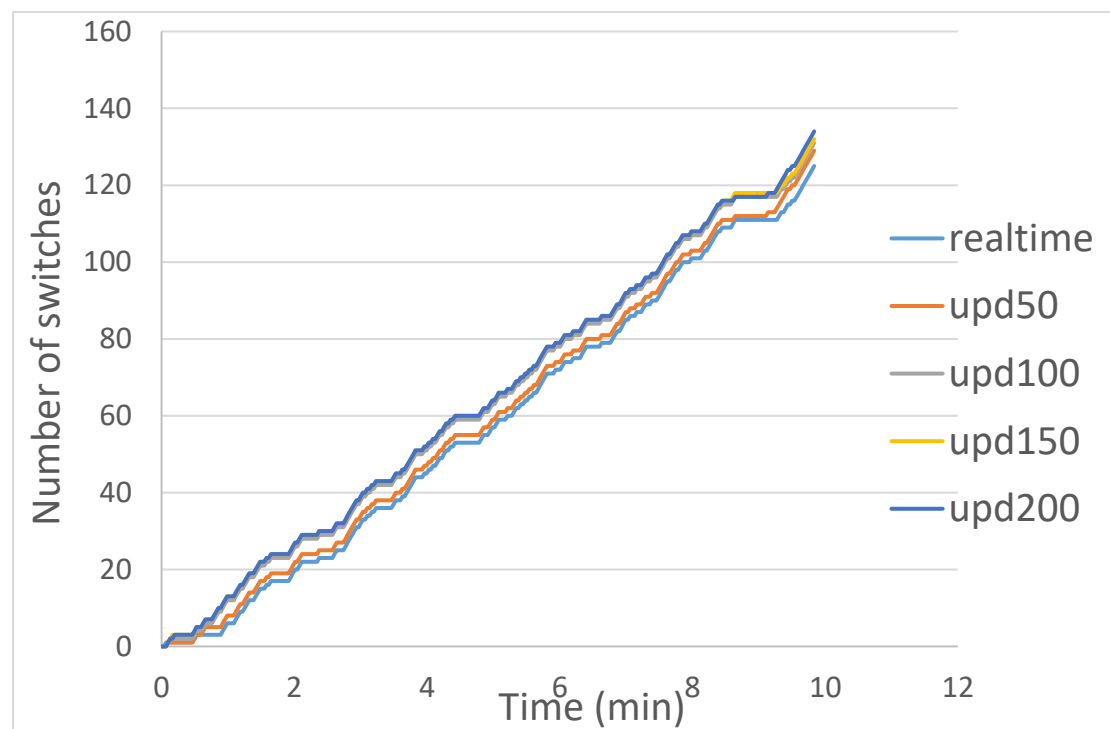
Εικόνα 52. Number of switches (Μοντέλο 3, Πείραμα 5, threshold=5, on/off=1/2)



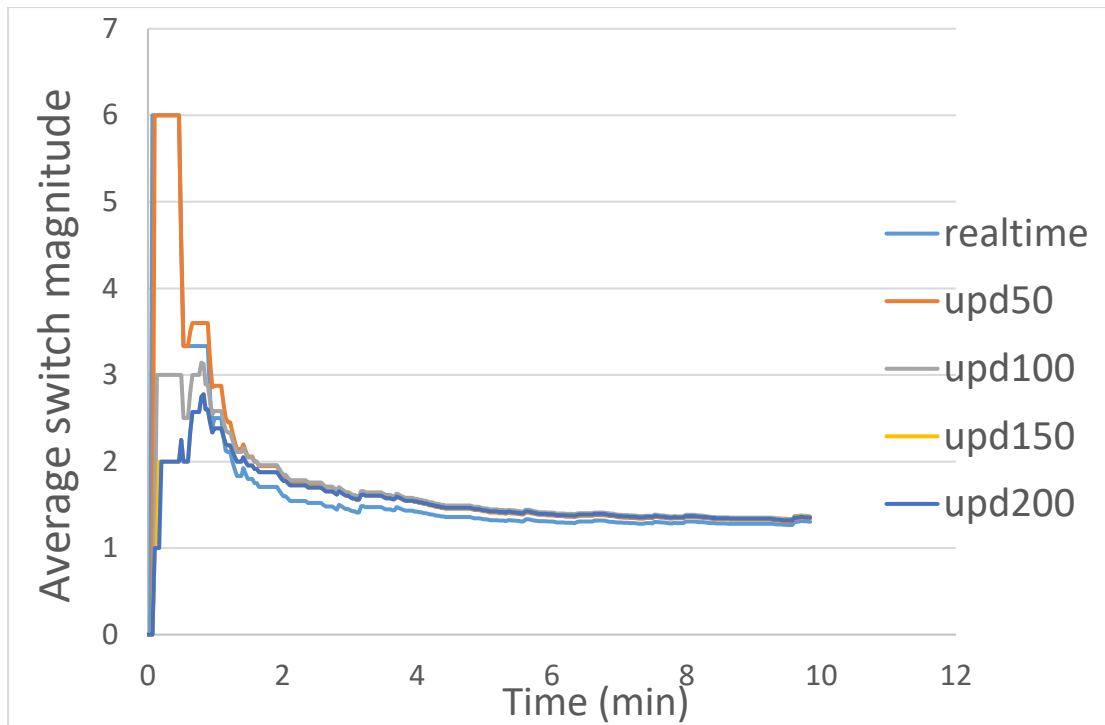
Εικόνα 53. Average switch magnitude (Μοντέλο 3, Πείραμα 5, threshold=5, on/off=1/2)

Στο μοντέλο 3 παρατηρούμε τις μικρότερες διαφορές μεταξύ των διαφορετικών τιμών του χρόνου ανανέωσης. Επειδή είναι το μοντέλο που ανταποκρίνεται με την μεγαλύτερη επιτυχία και καταφέρνει να εντοπίσει τον επιτιθέμενο και να τον περιορίσει αποτελεσματικά ακόμα και για μεγαλύτερους χρόνους ανανέωσης. Παρατηρούμε ότι η πορεία των γραφικών δεν παρουσιάζει διαφορές μετά το πρώτο λεπτό. Ο router επιτρέπει την προσωρινή αποθήκευση μόνο στην αρχή του βίντεο, στην συνέχεια δεν καταφέρνει ο επιτιθέμενος να τον παραπλανήσει ξανά και να βαθμολογηθεί με  $reputation > threshold$ . Αυτό συμβαίνει για όλες τις διαφορετικές τιμές ανανέωσης που μελετήσαμε σε αυτή την προσομοίωση.

Μοντέλο 3 (threshold = 7 , on/off = 1/3)



Εικόνα 54. Number of switches (Μοντέλο 3, Πείραμα 5, threshold=7, onoff=1/3)



Εικόνα 55. Average switch magnitude (Μοντέλο 3, Πείραμα 5, threshold=5, onoff=1/2)

Μπορούμε να σχολιάσουμε τα αποτελέσματα στο μοντέλο 3 συνολικά αφού οι δυο προσομοιώσεις παρουσιάζουν ίδιες γραφικές παραστάσεις.

Στο μοντέλο 3, λοιπόν, παρατηρούμε τις μικρότερες διαφορές μεταξύ των διαφορετικών τιμών του χρόνου ανανέωσης. Επειδή είναι το μοντέλο που ανταποκρίνεται με την μεγαλύτερη επιτυχία και καταφέρνει να εντοπίσει τον επιτιθέμενο και να τον περιορίσει αποτελεσματικά ακόμα και για μεγαλύτερους χρόνους ανανέωσης. Παρατηρούμε ότι η πορεία των γραφικών δεν παρουσιάζει διαφορές μετά το πρώτο λεπτό. Ο router επιτρέπει την προσωρινή αποθήκευση μόνο στην αρχή του βίντεο, στην συνέχεια δεν καταφέρνει ο επιτιθέμενος να τον παραπλανήσει ξανά και να βαθμολογηθεί με  $reputation > threshold$ . Αυτό συμβαίνει για όλες τις διαφορετικές τιμές ανανέωσης που μελετήσαμε σε αυτή την προσομοίωση.

## Κεφάλαιο 7ο : Συζήτηση

---

Σ' αυτό το κεφάλαιο σημειώνουμε τις γενικές παρατηρήσεις που προκύπτουν με την περάτωση της εργασίας. Επίσης, θέτουμε μελλοντικούς στόχους για τη διεύρυνση και τη βελτιστοποίηση της υλοποίησης μας.

### 7.1 Γενικά Συμπεράσματα

Ξεκινήσαμε τις προσομοιώσεις με σκοπό αρχικά να βελτιώσουμε τα μοντέλα άμυνας. Στη συνέχεια, τα συγκρίναμε στην βάση διαφορετικών συνθηκών κάθε φορά. Ολοκληρώνοντας την εργασία λοιπόν έχουμε έναν ικανοποιητικό σύνολο πειραμάτων για να χαρακτηρίσουμε το μοντέλο 3 ως το πλέον επιτυχημένο μοντέλο των πειραμάτων μας το οποίο ανταπεξήλθε εξαιρετικά σε πλήθος διαφορετικών καταστάσεων στη βάση ενός περιεχομένου αναφοράς.

### 7.2 Μελλοντική Εργασία

Θα μπορούσαν να πραγματοποιηθούν τα ίδια πειράματα και σε διαφορετικό βίντεο ή και στο ίδιο περιεχόμενο αλλά με διαφορετικό μέγεθος τμήματος. Το συγκεκριμένο περιεχόμενο διατίθεται και σε τμήματα του ενός sec. Στην εργασία αυτή όπως αναφέραμε χρησιμοποιήσαμε τμήματα των 2 sec οπότε μια τέτοια προοπτική θα σήμαινε ότι τα πακέτα δεδομένων που αποτελούν το σύνολο του περιεχομένου θα διπλασιαζόντουσαν. Θα ήταν ενδιαφέρον να δούμε πως θα συμπεριφερθούν τα μοντέλα άμυνας σε αυτή τη περίπτωση.

## Παράρτημα

---

Η αξία της εργασίας μας έγκειται κυρίως στην υλοποίηση της εφαρμογής. Επομένως, θεωρήσαμε χρήσιμο να παραθέσουμε εδώ ενδεικτικά ένα τμήμα του κώδικά μας, αυτό που παρουσιάζει το μεγαλύτερο ενδιαφέρον.

*ndn-router1.cpp*

```
1 #include "ndn-router.hpp"
2
3 #include "ns3/ptr.h"
4 #include "ns3/log.h"
5 #include "ns3/simulator.h"
6 #include "ns3/packet.h"
7 #include "ns3/callback.h"
8 #include "ns3/string.h"
9 #include "ns3/boolean.h"
10 #include "ns3/uinteger.h"
11 #include "ns3/integer.h"
12 #include "ns3/double.h"
13
14 #include "ns3/ndnSIM/helper/ndn-stack-helper.hpp"
15 #include "ns3/ndnSIM/helper/ndn-fib-helper.hpp"
16 #include "ns3/ndnSIM/model/ndn-common.hpp"
17
18 #include "model/ndn-app-face.hpp"
19
20 #include "ns3/random-variable-stream.h"
21
22
23 NS_LOG_COMPONENT_DEFINE("Router");
24
25 namespace ns3 {
26 namespace ndn {
27
28 // Necessary if you are planning to use ndn::AppHelper
29 NS_OBJECT_ENSURE_REGISTERED(Router);
30
31 TypeId
32 Router::GetTypeId()
33 {
34     static TypeId tid = TypeId("ns3::ndn::Router")
35         .SetGroupName("Ndn")
36         .SetParent<MultimediaConsumer<FileConsumerCbr>>()
37         .AddConstructor<Router>();
38     return tid;
39 }
40
41 Router::Router() : MultimediaConsumer<FileConsumerCbr>()
42 {
43 }
44 }
```

```

46 void
47 Router::StartApplication()
48 {
49     NS_LOG_FUNCTION_NOARGS();
50
51     MultimediaConsumer<FileConsumerCbr>::StartApplication();
52
53     // equivalent to setting interest filter for "/prefix" prefix
54     ndn::FibHelper::AddRoute(GetNode(), "/myprefix", m_face, 0);
55
56     m_isRouter = true;
57 }
58
59 void
60 Router::StopApplication()
61 {
62     NS_LOG_FUNCTION_NOARGS();
63
64     MultimediaConsumer<FileConsumerCbr>::StopApplication();
65 }
66
67 void
68 Router::OnInterest(std::shared_ptr<const ndn::Interest> interest)
69 {
70     ndn::App::OnInterest(interest); // forward call to perform app-level tracing
71
72     NS_LOG_DEBUG("Received Interest: " << interest->getName());
73
74     /*shared_ptr<Interest> interest2 = make_shared<Interest>();
75     interest2->setNonce(m_rand->GetValue(0, std::numeric_limits<uint32_t>::max()));
76     interest2->setName(interest->getName());
77     time::milliseconds interestLifeTime(m_interestLifeTime.GetMilliSeconds());
78     interest2->setInterestLifetime(interestLifeTime);*/
79
80     m_transmittedInterests(interest, this, m_face);
81     m_face->onReceiveInterest(*interest);
82 }
83
84
85 void
86 Router::OnData(std::shared_ptr<const ndn::Data> data)
87 {
88     NS_LOG_DEBUG("Received Data: " << data->getName());
89
90     if (!AreAllSeqReceived()){
91         MultimediaConsumer<FileConsumerCbr>::OnData(data);
92         return;
93     } else if (!m_mpdParsed){
94         OnFileReceived(0,0);
95     }
96     else if (m_currentDownloadType == Segment){
97
98         int pos = data->getName().getSubName(4,1).toUri().find(".");
99
100         if (pos != std::string::npos){
101             std::string seqNo = data->getName().getSubName(4,1).toUri().substr(9, pos - 9);
102
103             if (lastSeqNo != std::stoi(seqNo)) {
104
105                 if (lastSeqNo == NULL) {
106                     std::string prefix = data->getName().getPrefix(4).toUri().substr(0,26);
107                     content.first = prefix;
108                     content.second = reputation;
109                     if (std::stoi(seqNo) != 1) RateData(std::stoi(seqNo), std::stoi(seqNo));
110                 }
111                 else {
112                     RateData(std::stoi(seqNo) - lastSeqNo, std::stoi(seqNo));
113                     content.second = reputation;
114                 }
115                 lastSeqNo = std::stoi(seqNo);
116             }
117         } // if pos
118     } //else if
119
120     uint32_t cs_size = GetNode ()->GetObject<ContentStore>()->GetSize();
121     NS_LOG_DEBUG("CS SIZE " << cs_size);
122 }

```

```

131 }
132
133 void
134 Router::OnFileReceived(unsigned status, unsigned length){
135
136     MultimediaConsumer<FileConsumerCbr>::OnFileReceived(status, length);
137     NS_LOG_DEBUG("Parsed whole MPD File!");
138
139     lastSeqNo = NULL;
140     gaps=0;
141     reputation = 7; //so that, reputation =4.5 after update
142
143     sum=0;
144     UpdateReputationValue();
145 }
146
147 void Router::RateData(int seqModiff, int nooflastseg){
148
149     double gapratio;
150     float rating;
151
152     if (reputation == 10) rating = 0;
153     else if (seqModiff == 1) rating=1;
154     else {
155         gaps++;
156         gapratio = static_cast<double>(gaps)/nooflastseg;
157         rating = -0.5 * exp (1 - gapratio); // -0* exp(a * (1 - gapratio)), 0 = 0.5, a = 1
158     }
159
160     NS_LOG_DEBUG("rating value = " << rating);
161     if (Simulator::GetContext() == 1) std::cout << "rating value = " << rating << std::endl;
162     /*reputation += rating;
163
164     if (reputation > 10) reputation = 10;
165     if (reputation < 0) reputation = 0;*/
166
167     sum += rating;
168
169     /* print cs
170     Ptr<ContentStore> cs = GetNode ()->GetObject<ContentStore> ();
171     //Ptr<cs::Entry> current = cs->Begin (); // can iterate to another
172     for (auto it = cs->Begin(); it != cs->End(); it= cs->Next(it))
173         std::cout << it->GetName() << std::endl; */
174
175
176
177
178     return;
179 }
180
181 void Router::UpdateReputationValue(){
182
183     reputation = 0.7 * reputation + sum;
184     if (reputation > 10) reputation = 10;
185     if (reputation < 0) reputation = 0;
186
187     NS_LOG_DEBUG("New reputation value = " << reputation);
188
189     Ptr<L3Protocol> L3protocol = GetNode ()->GetObject<L3Protocol>();
190     // Get the forwarder instance
191     shared_ptr<nfd::Forwarder> fw = L3protocol->getForwarder();
192     fw->setReputationValueFromNdnSim(reputation);
193
194     sum=0;
195
196     Simulator::Schedule(Seconds(100),&Router::UpdateReputationValue,this);
197 }
198
199 }
200
201 } //namespace ndn
202 } // namespace ns3
203

```

## *ndn-router2.cpp*

```
1 #include "ndn-router.hpp"
2
3 #include "ns3/ptr.h"
4 #include "ns3/log.h"
5 #include "ns3/simulator.h"
6 #include "ns3/packet.h"
7 #include "ns3/callback.h"
8 #include "ns3/string.h"
9 #include "ns3/boolean.h"
10 #include "ns3/uinteger.h"
11 #include "ns3/integer.h"
12 #include "ns3/double.h"
13
14 #include "ns3/ndnSIM/helper/ndn-stack-helper.hpp"
15 #include "ns3/ndnSIM/helper/ndn-fib-helper.hpp"
16 #include "ns3/ndnSIM/model/ndn-common.hpp"
17
18 #include "model/ndn-app-face.hpp"
19
20 #include "ns3/random-variable-stream.h"
21
22
23 NS_LOG_COMPONENT_DEFINE("Router");
24
25 namespace ns3 {
26 namespace ndn {
27
28 // Necessary if you are planning to use ndn::AppHelper
29 NS_OBJECT_ENSURE_REGISTERED(Router);
30
31 TypeId
32 Router::GetTypeId()
33 {
34     static TypeId tid = TypeId("ns3::ndn::Router")
35         .SetGroupName("Ndn")
36         .SetParent<MultimediaConsumer<FileConsumerCbr>>()
37         .AddConstructor<Router>();
38
39     return tid;
40 }
41
42 Router::Router() : MultimediaConsumer<FileConsumerCbr>()
43 {
44 }
45
```

```

46 void
47 Router::StartApplication()
48 {
49     NS_LOG_FUNCTION_NOARGS();
50
51     MultimediaConsumer<FileConsumerCbr>::StartApplication();
52
53     // equivalent to setting interest filter for "/prefix" prefix
54     ndn::FibHelper::AddRoute(GetNode(), "/myprefix", m_face, 0);
55
56     m_isRouter = true;
57 }
58
59 void
60 Router::StopApplication()
61 {
62     NS_LOG_FUNCTION_NOARGS();
63
64     MultimediaConsumer<FileConsumerCbr>::StopApplication();
65 }
66
67 void
68 Router::OnInterest(std::shared_ptr<const ndn::Interest> interest)
69 {
70     ndn::App::OnInterest(interest); // forward call to perform app-level tracing
71
72     NS_LOG_DEBUG("Received Interest: " << interest->getName());
73
74     /*shared_ptr<Interest> interest2 = make_shared<Interest>();
75     interest2->setNonce(m_rand->GetValue(0, std::numeric_limits<uint32_t>::max()));
76     interest2->setName(interest->getName());
77     time::milliseconds interestLifetime(m_interestLifetime.GetMilliSeconds());
78     interest2->setInterestLifetime(interestLifetime);*/
79
80     m_transmittedInterests(interest, this, m_face);
81     m_face->onReceiveInterest(*interest);
82 }
83
84
85
86 void
87 Router::OnData(std::shared_ptr<const ndn::Data> data)
88 {
89     NS_LOG_DEBUG("Received Data: " << data->getName());
90
91
92
93
94     if (!AreAllSeqReceived()){
95         MultimediaConsumer<FileConsumerCbr>::OnData(data);
96         return;
97     } else if (!m_mpdParsed){
98         OnFileReceived(0,0);
99     }
100     else if (m_currentDownloadType == Segment){
101
102
103
104         int pos = data->getName().getSubName(4,1).toUri().find(".");
105
106         if (pos != std::string::npos){
107             std::string seqNo = data->getName().getSubName(4,1).toUri().substr(9, pos - 9);
108
109             if (lastSeqNo != std::stoi(seqNo)) {
110
111                 if (lastSeqNo == NULL) {
112                     std::string prefix = data->getName().getPrefix(4).toUri().substr(0,26);
113                     content.first = prefix;
114                     content.second = reputation;
115                     if (std::stoi(seqNo) != 1) RateData(std::stoi(seqNo), std::stoi(seqNo));
116                 }
117                 else {
118                     RateData(std::stoi(seqNo) - lastSeqNo, std::stoi(seqNo));
119                     content.second = reputation;
120                 }
121                 lastSeqNo = std::stoi(seqNo);
122             }
123         } // if pos
124         } //else if
125
126
127         uint32_t cs_size = GetNode ()->GetObject<ContentStore>()->GetSize();
128         NS_LOG_DEBUG("CS SIZE " << cs_size);
129     }
130
131 void
132 Router::OnFileReceived(unsigned status, unsigned length){
133
134
135
136     MultimediaConsumer<FileConsumerCbr>::OnFileReceived(status, length);

```

```

138 NS_LOG_DEBUG("Parsed whole MPD File!");
139
140 lastSeqNo = NULL;
141 gaps=0;
142
143 a=1; //model_2
144 b=1; //so that, reputation = 0.5 after update
145
146 //reputation = 7; //model_1 so that, reputation =5.25 after update
147 //sum=0; //model_1
148
149 UpdateReputationValue();
150 }
151
152
153 void Router::RateData(int seqNodiff, int nooflastseg){
154
155     double gapratio;
156     float rating;
157
158     if (seqNodiff == 1) rating=0;
159     else {
160         rating = 1;
161     }
162
163     NS_LOG_DEBUG("rating value = " << rating);
164     if (Simulator::GetContext() == 1) std::cout << "rating value = " << rating << std::endl;
165     /*reputation += rating;
166
167     if (reputation > 10) reputation = 10;
168     if (reputation < 0) reputation = 0;*/
169
170     a = 0.9*a + rating;
171     b = 0.9*b + (1-rating);
172
173     /* print cs
174     Ptr<ContentStore> cs = GetNode ()->GetObject<ContentStore> ();
175     //Ptr<cs::Entry> current = cs->Begin (); // can iterate to another
176     for (auto it = cs->Begin(); it != cs->End(); it= cs->Next(it))
177         std::cout << it->GetName() << std::endl; */
178
179     return;
180 }
181
182
183 void Router::UpdateReputationValue(){
184
185     reputation = 10*( a/(a + b) );//model_2
186     reputation = 10 - reputation;
187
188     //reputation = 0.75 * reputation + sum;
189     //if (reputation > 10) reputation = 10; //model_1
190     //if (reputation < 0) reputation = 0;
191
192     NS_LOG_DEBUG("New reputation value = " << reputation);
193
194     Ptr<I3Protocol> i3protocol = GetNode ()->GetObject<I3Protocol>();
195     // Get the forwarder instance
196     shared_ptr<nfd::Forwarder> fw = i3protocol->getForwarder();
197     fw->setReputationValueFromNdnSim(reputation);
198
199     //sum=0;
200     //a=0;
201     //b=0;
202
203     Simulator::Schedule(Seconds(100), &Router::UpdateReputationValue, this);
204
205 }
206
207
208 } //namespace ndn
209 } // namespace ns3
210

```

## *ndn-router3.cpp*

```
1 #include "ndn-router.hpp"
2
3 #include "ns3/ptr.h"
4 #include "ns3/log.h"
5 #include "ns3/simulator.h"
6 #include "ns3/packet.h"
7 #include "ns3/callback.h"
8 #include "ns3/string.h"
9 #include "ns3/boolean.h"
10 #include "ns3/uinteger.h"
11 #include "ns3/integer.h"
12 #include "ns3/double.h"
13
14 #include "ns3/ndnSIM/helper/ndn-stack-helper.hpp"
15 #include "ns3/ndnSIM/helper/ndn-fib-helper.hpp"
16 #include "ns3/ndnSIM/model/ndn-common.hpp"
17
18 #include "model/ndn-app-face.hpp"
19
20 #include "ns3/random-variable-stream.h"
21
22
23 NS_LOG_COMPONENT_DEFINE("Router");
24
25 namespace ns3 {
26 namespace ndn {
27
28 // Necessary if you are planning to use ndn::AppHelper
29 NS_OBJECT_ENSURE_REGISTERED(Router);
30
31 TypeId
32 Router::GetTypeId()
33 {
34     static TypeId tid = TypeId("ns3::ndn::Router")
35         .SetGroupName("Ndn")
36         .SetParent<MultimediaConsumer<FileConsumerCbr>>()
37         .AddConstructor<Router>();
38     return tid;
39 }
40
41 Router::Router() : MultimediaConsumer<FileConsumerCbr>()
42 {
43 }
44
45
```

```

46 void
47 Router::StartApplication()
48 {
49     NS_LOG_FUNCTION_NOARGS();
50
51     MultimediaConsumer<FileConsumerCbr>::StartApplication();
52
53     // equivalent to setting interest filter for "/prefix" prefix
54     ndn::FibHelper::AddRoute(GetNode(), "/myprefix", m_face, 0);
55
56     m_isRouter = true;
57
58 }
59
60 void
61 Router::StopApplication()
62 {
63     NS_LOG_FUNCTION_NOARGS();
64
65     MultimediaConsumer<FileConsumerCbr>::StopApplication();
66 }
67
68 void
69 Router::OnInterest(std::shared_ptr<const ndn::Interest> interest)
70 {
71     ndn::App::OnInterest(interest); // forward call to perform app-level tracing
72
73     NS_LOG_DEBUG("Received Interest: " << interest->getName());
74
75     /*shared_ptr<Interest> interest2 = make_shared<Interest>();
76     interest2->setNonce(m_rand->GetValue(0, std::numeric_limits<uint32_t>::max()));
77     interest2->setName(interest->getName());
78     time::milliseconds interestLifetime(m_interestLifetime.GetMilliSeconds());
79     interest2->setInterestLifetime(interestLifetime);*/
80
81     m_transmittedInterests(interest, this, m_face);
82     m_face->onReceiveInterest(*interest);
83
84 }
85
86 void
87 Router::OnData(std::shared_ptr<const ndn::Data> data)
88 {
89     NS_LOG_DEBUG("Received Data: " << data->getName());
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

138 NS_LOG_DEBUG("Parsed whole MPD File!");
139
140 lastSeqNo = NULL;
141 gaps=0;
142
143 nc=1; //model_3
144 np=1;//so that, reputation = 0.5 after update
145
146 gaps = 0; //model 3 arxikopoiw
147 correctseq = 0;
148 gapsold = 0;
149 correctseqold = 0;
150 f=0.9;
151
152 UpdateReputationValue();
153 }
154
155
156 void Router::RateData(int seqModiff, int nooflastseg){
157
158     if (seqModiff != 1) {
159
160         gaps = gaps + seqModiff; // o arithmos tw'n segments pou exei paraleipsei
161         correctseq = nooflastseg - gaps; //ta swsta seq mexri twra
162
163     }
164     else{
165         correctseq = nooflastseg - gaps; // ta swsta mexri twra
166     }
167
168     //NS_LOG_DEBUG("rating value = " << rating);
169     //if (Simulator::GetContext() == 1) std::cout << "rating value = " << rating << std::endl;
170
171     /* print cs
172     Ptr<ContentStore> cs = GetNode ()->GetObject<ContentStore> ();
173     //Ptr<cs::Entry> current = cs->Begin (); // can iterate to another
174     for (auto it = cs->Begin(); it != cs->End(); it= cs->Next(it))
175         std::cout << it->GetName() << std::endl; */
176
177     return;
178 }
179
180 void Router::UpdateReputationValue(){
181
182
183     np = np * f + gaps - gapsold;
184     nc = nc * f + correctseq - correctseqold;
185
186     reputation = 10*nc/(nc + np);
187
188     NS_LOG_DEBUG("New reputation value = " << reputation);
189
190     Ptr<L3Protocol> l3protocol = GetNode ()->GetObject<L3Protocol>();
191     // Get the forwarder instance
192     shared_ptr<nfd::Forwarder> fw = l3protocol->getForwarder();
193     fw->setReputationValueFromNdnSim(reputation);
194
195     if (reputation >= 6.5) f = 0.1; // f = e
196     if (reputation < 6.5) f = 0.9; // f = 1 - e , (e=0.1)
197
198     gapsold = gaps;
199     correctseqold = correctseq;
200
201     Simulator::Schedule(Seconds(100), &Router::UpdateReputationValue, this);
202 }
203
204 } //namespace ndn
205 } // namespace ns3
206

```

## Βιβλιογραφία

---

- [1] Named Data Networking: Motivation & Details - [named-data.net](http://named-data.net)
- [2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," Commun. ACM, vol. 55, no. 1, p. 117–124, Jan. 2012. [Online]. Available: <https://doi.org/10.1145/2063176.2063204>
- [3] Kapetanidou, I.A.; Sarros, C.-A.; Tsaoussidis, V. Reputation-Based Trust Approaches in Named Data Networking. Future Internet 2019, 11, 241. <https://doi.org/10.3390/fi11110241>
- [4] Ghali, C.; Tsudik, G.; Uzun, E. Needle in a haystack: Mitigating content poisoning in named-data networking. In Proceedings of the NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA, USA, 23–26 February 2014.
- [5] Ramani, S.K.; Tourani, R.; Torres, G.; Misra, S.; Afanasyev, A. NDN-ABS: Attribute-Based Signature Scheme for Named Data Networking. In Proceedings of the 6th ACM Conference on Information-Centric Networking. ACM, Macau, China, 24–26 September 2019.
- [6] Ghali, C.; Tsudik, G.; Uzun, E. Needle in a haystack: Mitigating content poisoning in named-data networking. In Proceedings of the NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA, USA, 23–26 February 2014.
- [7] Rezaeifar, Z.; Wang, J.; Oh, H. A trust-based method for mitigating cache poisoning in Name Data Networking. J. Netw. Comput. Appl. 2018, 104, 117–132.
- [8] Umeda, S.; Kamimoto, T.; Ohata, Y.; Shigeno, H. Interest flow control method based on user reputation and content name prefixes in named data networking. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015.
- [9] I. A. Kapetanidou, M. Hassan, C. -A. Sarros, M. Conti and V. Tsaoussidis, "Reputation-based Trust: A robust mechanism for Dynamic Adaptive Streaming

- over Named Data Networking," 2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE), 2020, pp. 114-121, doi: 10.1109/BigDataSE50710.2020.00023.
- [10] "Cisco visual networking index: Forecast and methodology, 2015-2020,"
- [11] W. Li, S. M. A. Oteafy and H. S. Hassanein, "Rate-Selective Caching for Adaptive Streaming Over Information-Centric Networks," in IEEE Transactions on Computers, vol. 66, no. 9, pp. 1613-1628, 1 Sept. 2017, doi: 10.1109/TC.2017.2687920.
- [12] S. Lederer, C. Mueller, B. Rainer, C. Timmerer and H. Hellwagner, "An experimental analysis of Dynamic Adaptive Streaming over HTTP in Content Centric Networks," 2013 IEEE International Conference on Multimedia and Expo (ICME), 2013, pp. 1-6, doi: 10.1109/ICME.2013.6607500.
- [13] M. F. Majeed, S. H. Ahmed, S. Muhammad, H. Song, and D. B. Rawat, "Multimedia streaming in informationcentric networking: A survey and future perspectives," Computer Networks, vol. 125, no. Supplement C, pp. 103 – 121, 2017, softwarization and Caching in NGN. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389128617302414>
- [14] B. Rainer, D. Posch, and H. Hellwagner, "Investigating the performance of pull-based dynamic adaptive streaming in ndn," IEEE Journal on Selected Areas in Communications, vol. 34, no. 8, pp. 2130–2140, Aug 2016.
- [15] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi, "Dynamic adaptive video streaming: Towards a systematic comparison of icn and tcp/ip," IEEE Transactions on Multimedia, vol. 19, no. 10, pp. 2166–2181, Oct 2017.
- [16] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," IEEE Network, vol. 28, no. 6, pp. 91–96, Nov 2014.
- [17] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner, "Adaptive streaming over content centric networks in mobile networks using multiple links," in 2013 IEEE International Conference on Communications Workshops (ICC), June 2013, pp. 677–681.
- [18] S. Petrangeli, N. Bouten, M. Claeys, and F. D. Turck, "Towards svcbased adaptive streaming in information centric networks," in 2015 IEEE

- International Conference on Multimedia Expo Workshops (ICMEW), June 2015, pp. 1–6.
- [19] Y. Liu, J. Geurts, J. C. Point, S. Lederer, B. Rainer, C. Mller, C. Timmerer, and H. Hellwagner, “Dynamic adaptive streaming over ccn: A caching and overhead analysis,” in 2013 IEEE International Conference on Communications (ICC), June 2013, pp. 3629–3633.
  - [20] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, “Offloading cellular networks with information-centric networking: The case of video streaming,” in 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2012, pp. 1–3.
  - [21] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, “Probe and adapt: Rate adaptation for http video streaming at scale,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, April 2014.
  - [22] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, “Bola: Near-optimal bitrate adaptation for online videos,” in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.
  - [23] C. Sieber, T. Hofeld, T. Zinner, P. Tran-Gia, and C. Timmerer, “Implementation and user-centric comparison of a novel adaptation logic for dash with svc,” in 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), May 2013, pp. 1318–1323.
  - [24] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, “An experimental evaluation of rate-adaptive video players over http,” *Image Commun.*, vol. 27, no. 4, pp. 271–287, Apr. 2012.
  - [25] G. Acs, M. Conti, P. Gasti, C. Ghali, G. Tsudik, and C. Wood, “Privacyaware caching in information-centric networking,” *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, 2017
  - [26] F. Zhang, W. He, X. Liu, and P. G. Bridges, “Inferring users’ online activities through traffic analysis,” in *Proceedings of the Fourth ACM Conference on Wireless Network Security*, ser. WiSec ’11. New York, NY, USA: ACM, 2011, pp. 59–70. [Online]. Available: <http://doi.acm.org/10.1145/1998412.1998425>
  - [27] M. Liberatore and B. N. Levine, “Inferring the source of encrypted http connections,” in *Proceedings of the 13th ACM Conference on Computer and*

- Communications Security, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 255–263. [Online]. Available: <http://doi.acm.org/10.1145/1180405.1180437>
- [28] D. Wu, Z. Xu, B. Chen and Y. Zhang, "What If Routers Are Malicious? Mitigating Content Poisoning Attack in NDN," 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 481-488, doi: 10.1109/TrustCom.2016.0100.
- [29] Hassan, Muhammad & Droms, Ralph & Conti, Mauro & Lal, Chhagan. (2018). Fair-RTT-DAS: A robust and efficient dynamic adaptive streaming over ICN. Computer Communications. 129. 10.1016/j.comcom.2018.07.033.
- [30] Network Simulations with the ns-3 Simulator - Thomas R. Henderson, Mathieu Lacage, George F. Riley
- [31] What is ns-3 - nsnam.org
- [32] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. 2017. On the Evolution of ndnSIM: an Open-Source Simulator for NDN Experimentation. SIGCOMM Comput. Commun. Rev. 47, 3 (July 2017), 19–33. DOI:<https://doi.org/10.1145/3138808.3138812>
- [33] Modeling and Tools for Network Simulation - Gross, James, Wehrle, Klaus, Güneş, Mesut
- [34] Christian Kreuzberger, Daniel Posch, Hermann Hellwagner "AMuSt Framework - Adaptive Multimedia Streaming Simulation Framework for ns-3 and ndnSIM", <https://github.com/ChristianKreuzberger/AMust-Simulator/>
- [35] James O. Berger. Statistical Decision Theory and Bayesian Analysis. Springer, second edition edition, 1985.
- [36] Anthony Davison. Bayesian Models. Chapter 11 in Manuscript, 2002
- [37] Buchegger, Sonja & Le Boudec, Jean-Yves. (2004). A Robust Reputation System for P2P and Mobile Ad-hoc Networks. P2P and Mobile Ad-hoc Networks, Second Workshop on the Economics of Peer-to-peer Systems.
- [38] Y. Sun, Z. Han, and K. J. R. Liu, "Defense of trust management vulnerabilities in distributed network," IEEE Comm. Magazine, vol. 46, no. 2, pp. 112–119, Feb. 2008.
- [39] X. Li and L. Liu, "PeerTrust: Supporting reputation-based trust for Peerto-Peer electronic communities," IEEE Trans. on Knowledge and Data Engineering, vol. 16, no. 7, pp. 843–857, July 2004.

- [40] Bo Hu and H. V. Zhao, "Pollution-resistant peer-to-peer live streaming using trust management," *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 3057-3060, doi: 10.1109/ICIP.2009.5414199.
- [41] C. Mueller, S. Lederer, J. Poecher, and C. Timmerer, "Demo paper: Libdash- an open source software library for the mpeg-dash standard," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2013, pp. 1–2.
- [42] S. Lederer, C. Muller, and C. Timmerer, "Dynamic adaptive " streaming over http dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94. [Online]. Available: <http://doi.acm.org/10.1145/2155555.2155570>.
- [43] Y. Liu, S. Dey, F. Ulupinar, M. Luby and Y. Mao, "Deriving and Validating User Experience Model for DASH Video Streaming," in *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 651-665, Dec. 2015, doi: 10.1109/TBC.2015.2460611.
- [44] D. Zegarra Rodríguez, R. Lopes Rosa, E. Costa Alfaia, J. Issy Abrahão and G. Bressan, "Video Quality Metric for Streaming Service Using DASH Standard," in *IEEE Transactions on Broadcasting*, vol. 62, no. 3, pp. 628-639, Sept. 2016, doi: 10.1109/TBC.2016.2570012.