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

Συνεργατική Αντιμετώπιση Κατανεμημένων Επιθέσεων Άρνησης Παροχής Υπηρεσίας σε Περιβάλλον Ευφυών Προγραμματιζόμενων Δικτύων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μαρία Α. Αποστολάκη

Επιβλέπων : Βασίλειος Μάγκλαρης, Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2015

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

Συνεργατική Αντιμετώπιση Κατανεμημένων Επιθέσεων Άρνησης Παροχής Υπηρεσίας σε Περιβάλλον Ευφυών Προγραμματιζόμενων Δικτύων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μαρία Α. Αποστολάκη

Επιβλέπων : Βασίλειος Μάγκλαρης, Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Ιούλιο 2015

. . . . .	. . . . .	. . . . .
Β. Μάγκλαρης	Ν.Κοζύρης	Δ. Καλογεράς
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Ερευνητής ΕΠΙΣΕΥ Β

Αθήνα, Ιούλιος 2015

. . . . . .
Μαρία Α. Αποστολάκη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια της φοίτησής μου στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Θα ήθελα να ευχαριστήσω τόν καθητή μου κο Βασίλη Μάγκλαρη για την εμπιστοσύνη που μου έδειξε και την δυνατότητα που μου έδωσε να εκπονήσω την διπλωματική μου στο συγκεκριμένο πολύ ενδιαφέρον θέμα. Παράλληλα, θα ήθελα να ευχαριστήσω πολύ τον Κώστα Γιώτη για την βοήθεια κατά την διάρκεια της διπλωματικής μου και τον χρόνο που αφιέρωσε για να συζητήσουμε τα κομβικά σημεία στην εξέλιξή της. Καταλήγοντας, θα ήθελα να ευχαριστήσω τους γονείς και τους φίλους μου για την συμπαράστασή τους όλα αυτά τα χρόνια.

Περίληψη

Οι κατανεμημένες επιθέσεις άρνησης παροχής υπηρεσίας (Destributed Denial of Service, DDoS) εγείρουν μία σημαντική απειλή για τα σύγχρονα δίκτυα. Το γεγονός ότι προκαλούνται από μηχανήματα που είναι σκορπισμένα σε πολλές διαφορετικές φυσικές τοποθεσίες, υπό διαφορετική διαχείριση, σε συνδυασμό με το ότι, μεμονωμένα, στέλνουν φυσιολογική κίνηση στο θύμα, κάνουν τις επιθέσεις αυτές δύσκολο να ανιχνευτούν και να αντιμετωπιστούν.

Στην πράξη, ένα Αυτόνομο Σύστημα (ΑΣ) που ανιχνεύει μία επίθεση εναντίον του, μπορεί να ακολουθήσει δυο στρατηγικές: Είτε να αποσυνδέσει αμέσως την υπηρεσία ή τον εξυπηρετητή που δέχεται την επίθεση, ολοκληρώνοντας την, αλλά προστατεύοντας ταυτόχρονα το υπόλοιπο δίκτυο, είτε να την αντιμετωπίσει. Η αντιμετώπιση της επίθεσης συνεπάγεται εγκατάσταση κανόνων (φίλτρων) για την επιλεκτική απόρριψη όλων των κακόβουλων ροών. Ο μεγάλος αριθμός των απαιτούμενων φίλτρων όμως, καθιστά αναγκαία είτε την χρησιμοποίηση κάποιας μεθόδου ομαδοποίησης ώστε να εγκατασταθούν γενικότεροι και λιγότεροι κανόνες, είτε την διασπορά τους σε ΑΣ στο μονοπάτι της επίθεσης. Η τελευταία μέθοδος είναι προτιμητέα αφενός επειδή διατηρεί την σύνδεση του θύματος και αφετέρου επειδή, δεν θυσιάζει νόμιμη κίνηση. Ωστόσο, η μέθοδος αυτή δημιουργεί την ανάγκη αποδοτικής επικοινωνίας και συνεργασίας μεταξύ των ΑΣ, καθώς και αυτοματοποίησης και επιτάχυνσης όλων των διαδικασιών που λαμβάνουν χώρα.

Η εργασία αυτή ασχολείται με τα παραπάνω θέματα, προτείνοντας ένα σύστημα το οποίο συντονίζει την διάδοση προτυποποιημένων μηνυμάτων τύπου IODEF κατά μήκος των μονοπατιών από τον στόχο της επίθεση προς τις πηγές της, στα οποία ζητείται η συνεισφορά των ΑΣ στο μονοπάτι της επίθεσης στην αντιμετώπιση της, μέσω εγκατάστασης φίλτρων. Το προτεινόμενο σύστημα εκμεταλλεύεται τα πλεονεκτήματα της προγραμματισιμότητας και της ελαστικής διαχείρισης του δικτύου που προσφέρει η χρήση των Ευφυών Προγραμματιζόμενων δικτύων (SDN). Μέσω της υλοποίηση στο λογισμικό διαφορετικών μοντέλων επιθέσεων, τα SDN διευκολύνουν σημαντικά τον πειραματισμό και μάλιστα σε πραγματική κίνηση ενώ ταυτόχρονα ξεπερνούν τους περιορισμούς που επιβάλει το υλικό και οι προκαθορισμένες λύσεις των κατασκευαστών.

Λέξεις Κλειδιά: Ευφϋή-Προγραμματιζόμενα-Δίκτυα, επιθέσεις κατανεμημένης άρνησης παροχής υπηρεσίας, OpenFlow, συνεργατική αντιμετώπιση, επίθεσης, πίστη, εμπιστοσύνη, OpenDayLight .

Abstract

Distributed denial of service attacks (DDoS) raise a significant threat to modern networks. The fact that they are caused by machines that are in many different physical locations under different network operators, combined with the fact that, they individually send normal traffic to the victim, make these attacks difficult to detect and mitigate.

In practice, an Autonomous System (AS) which detects an attack, has two options. It may either immediately disconnect the host under attack, a practice that completes the attack but protects the overall performance of the network, or try to mitigate the attack. Mitigating the attack, involves installing (firewall-like) rules (filters) that drop malicious flows. The large number of filters required, leads to the need for either aggregating the rules to fewer and more general ones and installing those instead, or cooperating with other ASes to distribute the required filters along the attack path. The latter method is preferred as it keeps the victim connected and does not drop legitimate traffic together with the malicious flows. However, this method implies addressing issues such as the communication and cooperation among ASes as well as automation of all procedures that take place.

This paper addresses these issues by proposing the exchange of a standardized IODEF-type messages along the attack path, to propagate a call to other ASes to contribute to the mitigation of the attack by installing some of the required filters. In this effort, we exploit the advantages of programmability and remote network control offered by Software Defined Networking (SDN). By permitting the implementation of different security algorithms solely on software, SDN facilitates experimentation even on actual traffic. Thus, SDN frees academia from the manufacturers' solutions in mitigating attacks.

Keywords: Software Defined Networks, DDos, OpenFlow, cooperative mitigation, Trust, Reputation, OpenDayLight.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Ερευνητικό Πρόβλημα - Προσέγγιση

Οι επιθέσεις κατανεμημένης άρνησης παροχής υπηρεσίας γίνονται όλο και πιο συχνές και περίπλοκες στα σύγχρονα δίκτυα. Προκαλούνται από κακόβουλο λογισμικό που εγκαθίσταται σε εκτεθειμένα μηχανήματα τα οποία στην συνέχεια παράγουν συντονισμένα και μαζικά κίνηση προς μία υπηρεσία ή ένα εξυπηρετητή. Με τον τρόπο αυτό, ένα απροσδιόριστα μεγάλο σύνολο μηχανημάτων ενισχύει την επίθεση, απρόθυμα και εν αγνοία των χειριστών τους. Το γεγονός αυτό προκαλεί σοβαρές δυσκολίες στον εντοπισμό και την αντιμετώπιση τέτοιων επιθέσεων.

Η χρήση των SDN διευκολύνει σημαντικά την αντιμετώπιση αυτών των επιθέσεων λόγω της άμεσης επιβολής πολιτικών , μέσω του ελεγκτή στο στρώμα δεδομένων. Παρόλα αυτά, ο όγκος των κανόνων που απαιτούνται για την επιλεκτική απόρριψη μόνο των κακόβουλων ροών δεν μπορούν εύκολα να καλυφθούν από τους διαθέσιμους πόρους ενός SDN Domain. Ως SDN domain θα αναφέρουμε ένα δίκτυο το οποίο διαχειρίζεται ένας ή περισσότεροι ελεγκτλες μέσω ενός πρωτοκόλλου όπως το OpenFlow.- Αντίθετα αν συνεργαστούν πολλά SDN Domains ,που βρίσκονται στο μονοπάτι της επίθεσης, κάθε ένα από τα οποία αναλάβει να εγκαταστήσει ένα μικρό υποσύνολο των απαραίτητων κανόνων η αντιμετώπιση μπορεί να είναι γρήγορη και επιτυχής.

Η υλοποίηση όμως, ενός τέτοιου συστήματος σε περιβάλλον SDN, προϋποθέτει την επί-

λυση πολλών ζητημάτων. Ένα τέτοιο ζήτημα, το οποίο μάλιστα έχει απασχολήσει πολύ την ακαδημαϊκή κοινότητα, είναι η ανάγκη αποτελεσματικής επικοινωνίας μεταξύ των SDN Domains. Έχουν γίνει πολλές προτάσεις αρχιτεκτονικών μοντέλων για αυτή την επικοινωνία χωρίς όμως να υπάρχει κάποια καθολικά επικρατέστερη μέχρι στιγμής. Ένα άλλο ζήτημα που είναι άμεσα συνδεδεμένο με την υλοποίηση ενός συστήματος συνεργατικής αντιμετώπισης DDoS επιθέσεων είναι η επιλογή κάποιου μοντέλου εμπιστοσύνης για την σύναψη σχέσεων μεταξύ των συνεργαζόμενων SDN Domains. Στην επιλογή αυτή πρέπει να ληφθεί υπόψιν ο τρόπος λειτουργίας των σημερινών δικτύων αλλά και οι υπάρχουσες σχέσεις μεταξύ αυτών. Τέλος είναι απαραίτητη η χρησιμοποίηση ενός αλγορίθμου ο οποίος θα καθορίζει την συμπεριφορά κάθε τομέα και θα αυτοματοποιεί την διαδικασία της απόφασης που καλείται να πάρει κάθε φορά που λαμβάνει μια αίτηση για συμμετοχή στην αντιμετώπιση μίας επίθεσης από ένα άλλο SDN domain.

Η παρούσα διπλωματική εργασία ασχολείται με τα παραπάνω ζητήματα και παρουσιάζει ένα σύστημα για την αντιμετώπιση επιθέσεων όσο το δυνατόν πιο κοντά στις πηγές τους. Απώτερος στόχος είναι να υλοποιηθεί αυτό ως μία εφαρμογή για τον ελεγκτή OpenDaylight. Η εφαρμογή αυτή θα τρέχει σε κάθε SDN domain πάνω από τον ελεγκτή. Μετά την ανίχνευση μίας επίθεσης και την αποκάλυψη των πηγών της από ένα ξεχωριστό μηχανισμό ανίχνευσης ο οποίος υλοποιείται σε κάθε Domain, η εφαρμογή αυτή αναλαμβάνει να γνωστοποιήσει την επίθεση στα γειτονικά SDN domains μέσω των οποίων δρομολογείται η κίνηση της επίθεσης και να ζητήσει την στήριξη τους για την αντιμετώπισή της. Συντάσσεται έτσι ένα μήνυμα το οποίο ακολουθεί την προτυποποίηση του IODEF αρχείου το οποίο περιέχει μια περιγραφή της επίθεσης μαζί με οδηγίες για την αντιμετώπιση της. Αυτό το μήνυμα διαδίδεται σε γειτονικά domains. Μετά την ανταλλαγή, κάθε παραλήπτης αξιολογεί την αντιμετώπιση σύμφωνα με την εμπιστοσύνη του στον αποστολέα, σε συνδυασμό με την πολιτική του και αποφασίζει ή όχι την εγκατάσταση των προτεινόμενων κανόνων στο επίπεδο των δεδομένων.

1.2 Συνεισφορά Εργασίας

Αξιοποιώντας τις δυνατότητες του SDN και εφαρμόζοντας τις έννοιες της φήμης και της εμπιστοσύνης για την σύναψη σχέσεων μεταξύ των αυτόνομων συστημάτων η εργασία

αυτή έχει ως στόχο τον σχεδιασμό ενός συστήματος το οποίο θα ικανοποιεί τις παραπάνω προδιαγραφές.

Πιο αναλυτικά, το προτεινόμενο συστήμα επιχειρεί αρχικά να επιτύχει την επιλεκτική απόρριψη μόνο των κακόβουλων ροών επιτρέποντας την εξυπηρέτηση όσο το δυνατό μεγαλύτερου ποσοστού της νόμιμης κίνησης. Η απόρριψη αυτή και κατ' επέκταση η αντιμετώπιση της επίθεσης οφείλει, να γίνει, σε σύντομο χρονικό διάστημα δεδομένης της κρισιμότητας και των συνεπειών της κατάστασης για το Domain που δέχεται επίθεση. Ταυτόχρονα, επιχειρείται να επιτευχθεί η βέλτιστη διασπορά των απαραίτητων για την αντιμετώπιση της επίθεσης κανόνων, σε όλα τα ΑΣ από τα οποία περνάει η κίνηση της επίθεσης. Τέλος, το σύστημα στοχεύει στην ελαχιστοποίηση των πόρων που δεσμεύονται, για την συμμετοχή ενός ΑΣ στην αντιμετώπισης της επίθεσης μέσω της επιλεκτικής ικανοποίησης αντίστοιχων αιτημάτων από άλλα ΑΣ.

1.3 Δομή Εργασίας

Το υπόλοιπο αυτής της εργασίας είναι οργανωμένο ως εξής: Στο κεφάλαιο 2 αναφέρονται οι βασικές αρχές των Ευφυών Προγραμματιζόμενων Δικτύων (SDN) και γίνεται ειδική αναφορά στο OpenFlow πρωτόκολλο καθώς και στην αρχιτεκτονική δομή του OpenDaylight. Στην συνέχεια το κεφάλαιο 3 ασχολείται με την επικοινωνία μεταξύ διαφορετικών SDN domains και γίνεται ειδική μνεία στην αρχιτεκτονική του πρωτοκόλλου που επιλέχθηκε. Στο κεφάλαιο 4, αφού περιγραφούν τα χαρακτηριστικά της DDos επίθεσης και των προτεινόμενων στρατηγικών αντιμετώπισης της, περιγράφεται το προτεινόμενο, για την αντιμετώπιση DDoS επιθέσεων σε SDN περιβάλλοντα, σύστημα. Στο κεφάλαιο 5, αναλύονται οι έννοιες της εμπιστοσύνης και της φήμης και η εφαρμογή τούς στο προτεινόμενο σύστημα. Στο κεφάλαιο 6, επιχειρείται η αξιολόγηση της εφαρμογής μέσω προσομοιώσεων. Τέλος, στο κεφάλαιο 7, τα συμπεράσματα συνοψίζονται και προτείνονται μελλοντικές προεκτάσεις .

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο παρελθόν, διάφορες τεχνολογίες αναπτύχθηκαν με σκοπό την δυνατότητα προγραμματισμού σε δίκτυα επικοινωνίας. Τα Ενεργά δίκτυα (ΕΔ) [1] αναπτύχθηκαν στη δεκαετία του 1990. Η βασική ιδέα στην οποία στηρίχθηκε η προσπάθεια αυτή ήταν ότι οι μεταγωγείς μπορούν να εξάγουν και να εκτελούν προγράμματα από πακέτα δεδομένων τα οποία διαχέονται στο διαδίκτυο. Με τη μέθοδο αυτή μπορούν να υλοποιηθούν νέοι μηχανισμοί δρομολόγησης και υπηρεσίες δικτύου, χωρίς να τροποποιηθεί το υλικό προώθησης. Ωστόσο, η προσέγγιση αυτή δεν επικράτησε, λόγω ζητημάτων ασφάλειας και επίδοσης που θα μπορούσαν να προκύψουν σχετικά με την εκτέλεση των προγραμμάτων στις συσκευές προώθησης. Για παράδειγμα, ένας εισβολέας μπορεί να διοχετεύσει κακόβουλα προγράμματα σε πακέτα δικτύου και να τα προωθήσει στο δίκτυο.

Μία άλλη σχετική προσέγγιση ήταν τα Προγραμματιζόμενα Δίκτυα (ΠΔ) [2] [3] που παρέχουν επίσης ένα μέσο προγραμματισμού στο δίκτυο εκτελώντας προγράμματα σε πακέτα με παρόμοιο τρόπο με αυτό των ενεργών δικτύων. Ωστόσο, τα προγράμματα δεν συμπεριλαμβάνονται στα πακέτα δεδομένων όπως στα Ενεργά Δίκτυα. Σε αυτή την προσέγγιση τα προγράμματα είναι εγκατεστημένα στο εσωτερικό των κόμβων του δικτύου, από τα οποία περνάνε τα πακέτα. Αυτό σαφώς μειώνει τις ανησυχίες που υπήρχαν για την ασφάλεια στα ενεργά δίκτυα, διότι ένας κόμβος αποδέχεται και τρέχει μόνο προγράμματα ύστερα από κατάλληλα σήματα και απαραίτητες εγκαταστάσεις πάνω του, που μπορούν να γίνουν μόνο από τον διαχειριστή.

2.1 Βασικές Αρχές

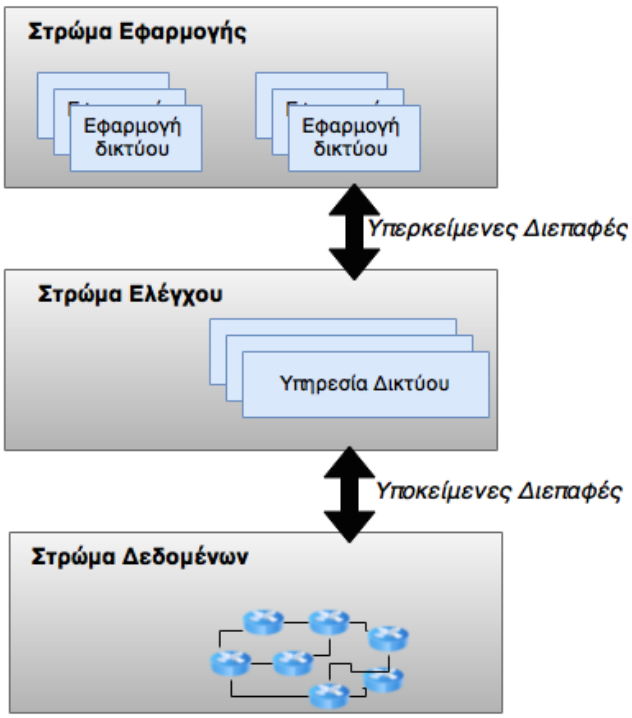
Τα Ευφυή δίκτυα κερδίζουν σταδιακά σε δημοτικότητα επειδή προσφέρουν την δυνατότητα προγραμματισμού από λογισμικό και ταυτόχρονα διευκολύνουν την υιοθέτηση καινοτομικών μεθόδων. Τα Ευφυή -Προγραμματιζόμενα δίκτυα διαχωρίζουν το επίπεδο ελέγχου του δικτύου από αυτό των δεδομένων ή της προώθησης επιτρέποντας έτσι την ανάπτυξη πολύπλοκων εφαρμογών στην κορυφή οι οποίες ελέγχουν το κατώτερο επίπεδο μέσω καθορισμένων διεπαφών. Αυτό έχει ως αποτέλεσμα την σημαντική μείωση της πολυπλοκότητα και του κόστους του υλικού καθώς και την σημαντική αύξηση της ευελιξίας όσο αναφορά την λειτουργικότητα και την διαχείρισή του δικτύου.

Στην συνέχεια θα αναλυθεί η έννοια των Ευφυών - Προγραμματιζόμενων δικτύων χρησιμοποιώντας ως βάση τον ορισμό του οργανισμού ανοικτών δικτύων (Open Networking Foundation (ONF) . Ο παρακάτω ορισμός είναι ο πιο κοινώς αποδεκτός ορισμός παγκοσμίως καθώς σε αυτόν συμμετέχουν οι μεγαλύτεροι ακαδημαϊκοί οργανισμοί και εταιρίες στο χώρο των δικτύων.

Στο Σχήμα 2.1 παρουσιάζεται μια γενικότερη ιδέα της αρχιτεκτονικής των Ευφυών-Προγραμματιζόμενων δικτύων. Το χαμηλότερο επίπεδο αποτελεί το στρώμα δεδομένων ή προώθησης. Τα καθήκοντα του στρώματος δεδομένων είναι η προώθηση πακέτων καθώς και η παρακολούθηση της τοπικής κίνησης και η συγκέντρωση στατιστικών.

Ένα στρώμα υψηλότερα βρίσκουμε το στρώμα ελέγχου. Τα βασικό καθήκον του στρώματος αυτού είναι η διαχείριση του στρώματος δεδομένων. Για το σκοπό αυτό, χρησιμοποιεί τις πληροφορίες που παρέχονται από το επίπεδο δεδομένων και καθορίζει τη λειτουργία του δικτύου και της δρομολόγησης. Αποτελείται από έναν ή περισσότερους ελεγκτές. Ένας ελεγκτής είναι ένα κομμάτι λογισμικού που επικοινωνεί με τα στοιχεία του δικτύου προώθησης μέσω των τυποποιημένων διεπαφών, οι οποίες ονομάζονται υποκείμενες διεπαφές. Το OpenFlow είναι η πιο ευρέως χρησιμοποιούμενη τέτοια διεπαφή και καθορίζει την επικοινωνία του ελεγκτή μόνο με μεταγωγείς σε αντίθεση με άλλες αντίστοιχες διεπαφές οι οποίες λαμβάνουν υπόψιν και την ανάγκη επικοινωνίας του ελεγκτή και με άλλα στοιχεία του δικτύου όπως δρομολογητές.

Το ανώτερο στρώμα είναι το στρώμα εφαρμογής. Το στρώμα εφαρμογής περιέχει εφαρμο-



Σχήμα 2.1: *Αρχιτεκτονική SDN τριών επιπέδων , Πηγή:https://www.opennetworking.org/about/onf-overview*

γές δικτύου και μπορεί να εισαγάγει νέες λειτουργικότητες, σχετικές με την ασφάλεια, την διαχείριση και το συστήματα προώθησης του δικτύου. Το επίπεδο εφαρμογής λαμβάνει μια αφηρημένη και συνολική εικόνα του δικτύου από τους ελεγκτές και την χρησιμοποιεί για να προσφέρει την κατάλληλη καθοδήγηση στο στρώμα ελέγχου. Οι διεπαφές μεταξύ του στρώματος εφαρμογής και του στρώματος ελέγχου αναφέρεται ως υπερκείμενες διεπαφές. Για αυτές, δεν υπάρχει κάποια τυποποίηση σήμερα, και στην πράξη, το λογισμικό του στρώματος ελέγχου παρέχει τις δικές του διεπαφές που εξυπηρετούν τις απαιτήσεις της κάθε εφαρμογής.

2.2 Το πρωτόκολλο OpenFlow

Το πρωτόκολλο OpenFlow είναι το πιο συνηθισμένο πρωτόκολλο για τη διασύνδεση του στρώματος ελέγχου με το στρώμα δεδομένων. Το OpenFlow [4] προτάθηκε αρχικά από το Πανεπιστήμιο του Stanford, και έχει πλέον τυποποιηθεί από το ONF [5]. Το παρόν

κεφάλαιο είναι δομημένο ως εξής:

Πρώτα δίνεται μια επισκόπηση της δομής του OpenFlow και των αλληλεπιδράσεων μεταξύ των ελεγκτών και των μετωγέων. Στη συνέχεια περιγράφονται τα χαρακτηριστικά που υποστηρίζονται από τις διαφορετικές εκδόσεις του.

Η αρχιτεκτονική OpenFlow βασίζεται σε τρεις θεμελιώδεις παραδοχές.

1. Το στρώμα δεδομένων αποτελείται από μεταγωγείς που υποστηρίζουν OpenFlow.

2. Το επίπεδο ελέγχου αποτελείται από έναν ή περισσότερους ελεγκτές που υποστηρίζουν OpenFlow.

3. Οι μεταγωγείς συνδέονται μέσω ενός ασφαλούς καναλιού με το επίπεδο ελέγχου.

2.2.1 Ο Μεταγωγέας OpenFlow

Ένας μεταγωγέας που υποστηρίζει το πρωτόκολλο OpenFlow είναι η βασική συσκευή προώθησης και προωθεί τα πακέτα σύμφωνα με τον πίνακα ροής του. Αυτός ο πίνακας περιέχει ένα σύνολο εγγραφών, καθεμία από τις οποίες αποτελείται από τα αναγνωριστικά πεδία, τα πεδία των μετρητών και από κάποιες οδηγίες, όπως φαίνεται παρακάτω.

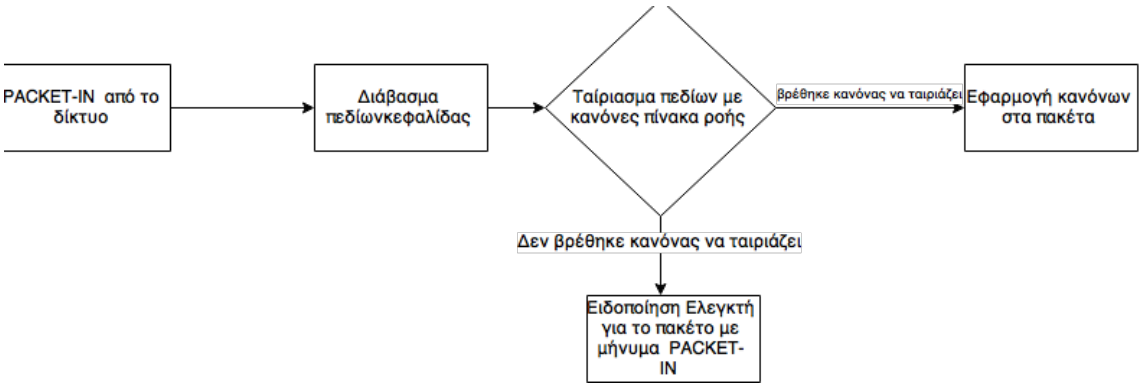
[Πεδία κεφαλίδας | μετρητές | ενέργειες]

Οι εγγραφές στον πίνακα ροής καλούνται επίσης και καταχωρήσεις και κανόνες ροής. Τα **πεδία της κεφαλίδας** σε ένα κανόνα περιγράφουν με ποια πακέτα θα εφαρμοστεί αυτός. Έτσι τα πεδία της επικεφαλίδας κάθε πακέτου συγκρίνονται με τα πεδία κεφαλίδας του κανόνα. Για την διαδικασία του ταιριάσματος μπορούν να χρησιμοποιηθούν οποιαδήποτε από τα πεδία της επικεφαλίδας ενός πακέτου. Τέλος τα χρησιμοποιούμενα για το ταίριασμα πεδία μπορούν να διαφέρουν ανάλογα με το πρωτόκολλο πχ διαφορετικά πεδία για Ethernet, IPv4, IPv6 ή MPLS, με την προϋπόθεση ότι το εκάστοτε πρωτόκολλο υποστηρίζεται από τις προδιαγραφές της έκδοσης του Openflow που χρησιμοποιείται. Οι **μετρητές** χρησιμοποιούνται για τη συλλογή στατιστικών στοιχείων σχετικών με τις ροές πακέτων. Για κάθε κανόνα αποθηκεύονται ο αριθμός των ληφθέντων πακέτων και τα bytes αυτών,

καθώς και η διάρκεια ισχύς του κανόνα. Οι **ενέργειες** ορίζουν πώς οι μεταγωγείς θα χειριστούν τα πακέτα της ροής. Συνηθισμένες ενέργειες είναι η ”προώθηση” η ”διαγραφή” και η ”τροποποίηση”.

2.2.2 Ο Ελεγκτής OpenFlow

Ο Ελεγκτής είναι ένα πρόγραμμα λογισμικού το οποίο είναι υπεύθυνο για τη συμπλήρωση, τροποποίηση ή κατάργηση των πινάκων ροής των μεταγωγέων, με σκοπό τον καθορισμό της συμπεριφοράς τους. Υπάρχουν τρεις κατηγορίες επικοινωνίας στο πρωτόκολλο OpenFlow: (α) η ελεγκτής - μεταγωγέας, (β) η ασύγχρονη και (γ) η συμμετρική επικοινωνία. Όλες υλοποιούνται μέσω ενός ασφαλούς καναλιού ελέγχου. Η επικοινωνία ελεγκτής-μεταγωγέας είναι υπεύθυνη για την ανίχνευση χαρακτηριστικών, την παραμετροποίηση, τον προγραμματισμό του μεταγωγέα και την ανάκτηση πληροφοριών. Μια ασύγχρονη επικοινωνία ενεργοποιείται από τον μεταγωγέα χωρίς καμία πρόσκληση από τον ελεγκτή. Χρησιμοποιείται για να ενημερώσει τον ελεγκτή για τη άφιξη πακέτων, την αλλαγή κατάστασης του και τυχόν λάθη που προέκυψαν. Τέλος, μία συμμετρική επικοινωνία υλοποιείται όταν αποστέλλονται μηνύματα χωρίς πρόσκληση από καμία από τις δύο πλευρές, δηλαδή, τόσο ο μεταγωγέας όσο και ο ελεγκτής είναι ελεύθεροι να εκκινήσουν την επικοινωνία χωρίς πρόσκληση από την άλλη πλευρά. Παραδείγματα για συμμετρική επικοινωνία είναι “hello” ή “echo” μηνύματα που μπορούν να χρησιμοποιηθούν για να προσδιοριστεί εάν το κανάλι ελέγχου εξακολουθεί να είναι διαθέσιμο.



Σχήμα 2.2: *Βασικός μηχανισμός προώθηση πακέτων στο OpenFlow, Πηγή:https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf*

Ο βασικός μηχανισμός προώθηση πακέτων με OpenFlow απεικονίζεται στο Σχήμα 2.2. Όταν ένας μεταγωγέας λαμβάνει ένα πακέτο, αναλύει την κεφαλίδα του η οποία συγκρίνεται με τους κανόνες του πίνακα ροής. Εάν βρεθεί τουλάχιστον ένας κανόνας του οποίου τα πεδία ταιριάζουν με την επικεφαλίδα του πακέτου, τότε αυτός εφαρμόζεται. Εάν υπάρχουν περισσότεροι του ενός τέτοιοι κανόνες επιλέγεται αυτός με την υψηλότερη προτεραιότητα. Στη συνέχεια, ο μεταγωγέας ενημερώνει τους μετρητές του κανόνα που επιλέχθηκε. Τέλος, εκτελεί τις ενέργειες που προσδιορίζονται από τον κανόνα, π.χ. ο μεταγωγέας προωθεί το πακέτο σε μια θύρα. Εάν δεν υπάρχει καταχώρηση του πίνακα ροής που να ταιριάζει με την επικεφαλίδα του πακέτου, ο μεταγωγέας ειδοποιεί τον ελεγκτή του και αποθηκεύει το πακέτο προσωρινά εάν υπάρχει αυτή η δυνατότητα. Ο μεταγωγέας στέλνει ένα PACKET-IN μήνυμα στον ελεγκτή με ενσωματωμένο είτε ολόκληρο το πακέτο (εάν δεν υπάρχει η δυνατότητα αποθήκευσης) είτε τα πρώτα bytes του (εάν υπάρχει η δυνατότητα αποθήκευσης). Ο ελεγκτής που λαμβάνει το PACKET-IN μήνυμα προσδιορίζει την ενέργεια που πρέπει να εφαρμοστεί στο πακέτο και εγκαθιστά ένα ή περισσότερους κανόνες στο επίπεδο δεδομένων. Τα αποθηκευμένα πακέτα στην συνέχεια προωθούνται σύμφωνα με τους κανόνες. Αυτό επιτυγχάνεται είτε αποστέλλοντας τον κωδικό (ID) της θέσης όπου αποθηκεύτηκε το πακέτο μαζί με την ενέργεια που πρέπει να εφαρμοστεί είτε με ρητό μηνύματα PACKET-OUT. Συνήθως, ο ελεγκτής ρυθμίζει ολόκληρη την διαδρομή για το πακέτο στο δίκτυο με την τροποποίηση των πινάκων ροής όλων των μεταγωγεων στην διαδρομή του.

2.3 Προδιαγραφές OpenFlow

Στο κεφάλαιο αυτό θα εξετάσουμε τις διαφορετικές προδιαγραφές OpenFlow προβάλλοντας, τις λειτουργικότητες που υποστηρίζονται και τις αλλαγές σε σχέση με την προηγούμενη κύρια έκδοση τους.

2.3.1 OpenFlow 1.0

Το OpenFlow [6] εκδόθηκε επίσημα τον Δεκέμβριο του 2009. Παρέχει την δυνατότητα ταιριάσματος πακέτων τύπου Ethernet και IP με κανόνες του πίνακα ροής με βάση τις διευ-

θύνσεις πηγής και προορισμού. Επιπλέον μπορούν να χρησιμοποιηθούν τα πεδία Ethernet-type και VLAN για το ταίριασμα πακέτων Ethernet. καθώς επίσης και οι διαφοροποιημένες υπηρεσίες (differentiated services DS) και η ρητή κοινοποίηση της συμφόρησης (Explicit Congestion Notification ECN). Τέλος μπορούν να χρησιμοποιηθούν οι TCP ή UDP πόρτες πηγής και του προορισμού.Το πρότυπο OpenFlow προσδιορίζει ακριβώς το διάβασμα πακέτου και τον αλγόριθμο ταιριάσματος. Ο αλγόριθμος ταιριάσματος του πακέτου ξεκινά με τη σύγκριση των Ethernet και VLAN πεδίων και συνεχίζεται εάν είναι απαραίτητο με τα πεδία της κεφαλίδας IP. Εάν τα μηνύματα είναι τύπου TCP ή UDP, εξετάζονται τα αντίστοιχα πεδία της κεφαλίδας του στρώματος μεταφοράς.

Πολλές ενέργειες μπορούν να χρησιμοποιηθούν για κάθε ροή πακέτων. Η πιο σημαντική είναι η ενέργεια προώθησης. Αυτή η ενέργεια προωθεί ένα πακέτο σε μια συγκεκριμένη θύρα ή σε όλες. Επιπλέον, ο ελεγκτής μπορεί να αναθέσει στο μεταγωγέα, να ενσωματώσει όλα τα πακέτα μιας ροής και να του τα στέλνει. Μια ενέργεια για διαγραφή πακέτων είναι επίσης διαθέσιμη. Αυτή η ενέργεια επιτρέπει την υλοποίηση των λιστών ελέγχου δικτύου (Network Access Control) με OpenFlow. Επιπλέον μια άλλη ενέργεια επιτρέπει την τροποποίηση των πεδίων της επικεφαλίδας του πακέτου. Στατιστικά μπορούν να συγκεντρωθούν παίρνοντας δεδομένα από διάφορους μετρητές στο μεταγωγέα. Ο ελεγκτής μπορεί να ζητήσει έναν πίνακα στατιστικών στοιχείων που περιέχει τον αριθμό των ενεργών καταχωρήσεων ή των πακέτων που έχουν υποστεί επεξεργασία. Τα στατιστικά στοιχεία αποθηκεύονται ανά ροή ως μέρος της αντίστοιχης εγγραφής του πίνακα ροής. Επιπλέον είναι διαθέσιμα και στατιστικά ανά θύρα και ανά ουρά. Τέλος το OpenFlow 1.0 παρέχει βασική υποστήριξη ποιότητας υπηρεσίας (QoS) χρησιμοποιώντας βέβαια μόνο ουρές ελαχίστου ρυθμού. Ένας μεταγωγέας μπορεί δηλαδή να περιέχει μία ή περισσότερες ουρές, και κάθε ουρά συνδέεται σε μια θύρα. Επίσης ένας ελεγκτής OpenFlow μπορεί να ρωτήσει τις πληροφορίες σχετικά με μια ουρά. Η ”Τοποθέτηση στην ουρά” είναι μια ενέργεια που επιτρέπει τη προώθηση σε ουρά. Τα πακέτα στην συνέχεια αντιμετωπίζονται σύμφωνα με τις ιδιότητες της ουράς. Είναι σημαντικό να σημειωθεί ότι οι ελεγκτές OpenFlow είναι μόνο σε θέση να διερευνούν, αλλά όχι να τροποποιούν , τις ιδιότητες μίας ουράς.

2.3.2 Μεταγενέστερες εκδόσεις OpenFlow

OpenFlow 1.1 [7] εκδόθηκε επίσημα τον Φεβρουάριο του 2011 και παρουσιάζει σημαντικές αλλαγές σε σχέση με OpenFlow 1.0. Η επεξεργασία πακέτων λειτουργεί διαφορετικά τώρα. Τα πακέτα υπόκεινται σε επεξεργασία μέσω του αγωγού πολλαπλών πινάκων ροής. Οι κυριότερες αλλαγές είναι η ύπαρξη ενός αγωγού πολλαπλών πινάκων ροής και ενός πίνακα ομάδας. Με το OpenFlow 1.0, το αποτέλεσμα του ταιριάσματος κάθε πακέτου είναι μία λίστα ενεργειών από τις οποίες μία εφαρμόζεται στα πακέτα της ροής. Με το OpenFlow 1.1, το αποτέλεσμα του αγωγού είναι ένα σύνολο ενεργειών που συγκεντρώνονται κατά τη διάρκεια της εκτέλεσης του αγωγού και εφαρμόζονται στο πακέτο, στο τέλος.

Ο πίνακας του αγωγού απαιτεί ένα πεδίο μετα-δεδομένων, οδηγίες και σύνολα ενεργειών έχει δηλαδή την παρακάτω μορφή:

[Πεδία κεφαλίδας | μετρητές | οδηγίες]

Το πεδίο μετα-δεδομένων μπορεί να συλλέγει μετα-δεδομένα για ένα πακέτο κατά τη διαδικασία ταιριάσματος και να τα μεταφέρει από το ένα βήμα αγωγού προς το επόμενο. Οι εγγραφές στον πίνακα ροής περιέχουν οδηγίες αντί ενεργειών, όπως φαίνεται παραπάνω. Οι πιθανές τιμές οδηγιών-εντολών για το OpenFlow 1.1 είναι οι εξής:

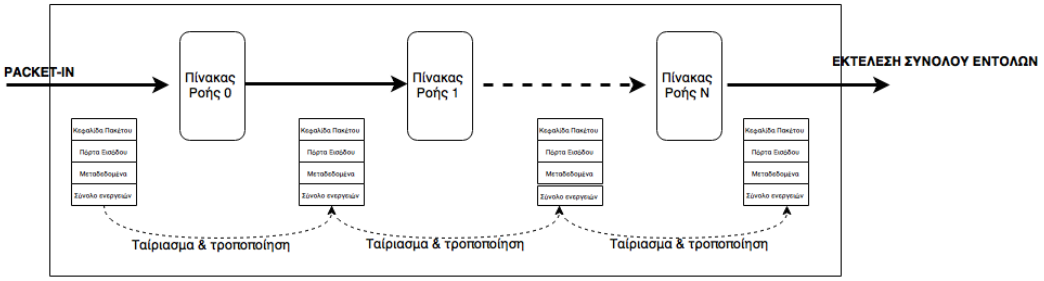
1. “Apply-Actions”: η εντολή επιβάλει την άμεση εφαρμογή ενός κανόνα στο πακέτο. Οι καθορισμένες ενέργειες δεν προστίθενται στο σύνολο ενεργειών.

2. ”Write-Actions”: η εντολή προσθέτει τις καθορισμένες δράσεις στο σύνολο ενεργειών και επιτρέπει την αυξητική κατασκευή της τελικής ενέργειας που θα εφαρμοστεί στη ροή κατά τη διάρκεια της εκτέλεσης του αγωγού.

3. ”Clear-Action”: η εντολή αδειάζει το σύνολο ενεργειών

4. ”Write-Metadata”: η εντολή ενημερώνει το πεδίο μετα-δεδομένων και εφαρμόζεται η καθορισμένη μάσκα στο τρέχον πεδίο μετα-δεδομένων.

5. ”Goto-Table”: η εντολή αναφέρεται σε έναν πίνακα ροής, και η διαδικασία ταιριάσματος συνεχίζεται με αυτόν τον πίνακα.



Σχήμα 2.3: *OpenFlow Pipeline, Πηγή:http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf*

Για την αποφυγή βρόχων στον αγωγό, μόνο πίνακες με υψηλότερο αναγνωριστικό (ID) από αυτό του τρέχοντος πίνακα επιτρέπεται να αναφέρεται από αυτόν. Έτσι, ο αλγόριθμος ταιριάσματος θα τερματίσει νομοτελειακά Αν δεν υπάρχει κάποια ”Goto” εντολή, η εκτέλεση του αγωγού σταματά, και το συσσωρευμένο σύνολο ενεργειών εκτελείται στο πακέτο.Η δεύτερη σημαντική αλλαγή είναι η προσθήκη ενός πίνακα ομάδας. Ο πίνακας ομάδας υποστηρίζει πιο σύνθετες συμπεριφορές προώθησης, οι οποίες ενδεχομένως να εφαρμοστούν σε ένα σύνολο ροών πακέτων. Αποτελείται από εγγραφές που συντάσσονται με τον παρακάτω τρόπο:

[Αναγνωριστικό Ομάδας | Είδος Ομάδας |Μετρητές | Δοχείο Ενεργειών]

Μια τέτοια εγγραφή θα χρησιμοποιηθεί εάν ένας κανόνας του πίνακα ροής παραπέμψει σε αυτήν χρησιμοποιώντας το αναγνωριστικό της ομάδα της εγγραφής. Ειδικότερα, πολλαπλές εγγραφές πίνακα ροής μπορεί να δείξουν το ίδιο αναγνωριστικό ομάδας, έτσι ώστε η εγγραφή να χρησιμοποιηθεί από πολλαπλές ροές πακέτων.Η εγγραφή πίνακα ομάδας περιέχει τον τύπο της ομάδας, τους μετρητές και το δοχείο ενεργειών. Οι μετρητές χρησιμοποιούνται για τη συλλογή στατιστικών στοιχείων σχετικά με τα πακέτα που υποβάλλονται σε επεξεργασία από αυτή την ομάδα. Το δοχείο ενεργειών περιέχει μια σειρά από ενέργειες που μπορούν να εκτελεστούν, ανάλογα με τον τύπο της ομάδας. Υπάρχουν πιθανώς πολλαπλά δοχεία ενεργειών

για μια καταχώρηση πίνακα της ομάδας. Οι τύποι ομάδας καθορίζουν ποιες από αυτές εφαρμόζονται. Υπάρχουν τέσσερις τύποι ομάδας, και περιγράφονται οι δύο από αυτούς.Ο τύπος της ομάδας **all** χρησιμοποιείται για την υλοποίηση broadcast και multicast. Πακέτα της ομάδας αυτής θα υποστούν επεξεργασία από όλα τα δοχεία δράση. Οι ενέργειες του κάθε δοχείου εφαρμόζονται στο πακέτο διαδοχικά. Ο τύπος της ομάδας **fast failover** χρησιμοποιείται για να εφαρμόσει εφεδρικά μονοπάτια. Ένα δοχείο ενεργειών θεωρείται ενεργό, εάν όλες οι πόρτες και οι γραμμές που εμπλέκονται στην υλοποίηση των ενεργειών του, λειτουργούν κανονικά. Μια ομάδα με τον τύπο ”fast failover” εκτελεί το πρώτο ενεργό δοχείο ενεργειών΄Ετσι, η ”fast failover” τύπος ομάδα υποστηρίζει την αναδρομολόγηση αποφάσεων, που δεν απαιτούν άμεση αλληλεπίδραση ελεγκτή. Έτσι, ένας μηχανισμός ταχείας επιλογής εναλλακτικής δρομολόγησης μπορεί να υλοποιηθεί γεγονός το οποίο εξασφαλίζει ελάχιστη απώλεια πακέτων σε περίπτωση αποτυχίας.

Σε γενικές γραμμές, ο αριθμός των υποστηριζόμενων ενεργειών στο OpenFlow 1.1 είναι μεγαλύτερος από αυτόν του OpenFlow 1.0. Για παράδειγμα, το (TTL) πεδίο Time-To-Live στην επικεφαλίδα IP μπορεί να μειώνεται στο OpenFlow 1.1 Επίσης παρέχει πρόσθετα στατιστικά στοιχεία λόγω της αλλαγής της αρχιτεκτονικής μεταγωγέα. Έτσι ο Ελεγκτής OpenFlow 1.1 μπορεί να ρωτήσει τα στατιστικά στοιχεία για τον πίνακα της ομάδας και της ομάδας εγγραφών, καθώς και για τα δοχεία ενεργειών.

Το OpenFlow 1.2 [8] εκδόθηκε επίσημα τον Δεκέμβριο του 2011 και προσθέτει στα υποστηριζόμενα πρωτόκολλα το IPv6. Στο OpenFlow 1.2 το ταίριασμα μπορεί να γίνει με βάση τα πεδία πηγής και προορισμού IPv6, τον αριθμό πρωτοκόλλου, την ετικέτα ροής και διάφορα πεδία του ICMPv6. Οι χρήστες έχουν τώρα την δυνατότητα να επεκτείνουν το OpenFlow από μόνοι τους προσθέτοντας πεδία που χρησιμοποιούνται στο ταίριασμα μέσω μιας δομής της μορφής τύπος-μήκος-τιμή (TLV), η οποία ονομάζεται επεκτάσιμο ταίριασμα OpenFlow (OpenFlow eXtensible Match)

Με το OpenFlow 1.2, ένας μεταγωγέας μπορεί ταυτόχρονα να συνδέεται με περισσότερους από έναν ελεγκτές. Ο μεταγωγέας ξεκινά τη σύνδεση, και οι άλλοι ελεγκτές αποδέχονται τις συνδέσεις. Ένας από αυτούς τους ελεγκτές ορίζεται κύριος

και προγραμματίζει τον μεταγωγέα. Οι άλλοι ελεγκτές είναι σκλάβοι. Κάθε ελεγκτής σκλάβος μπορεί να προαχθεί σε κύριο, και αντίστοιχα ο κύριος να υποβιβαστεί σε σκλάβο. Αυτό δίνει την δυνατότητα ανακατεύθυνσης ελεγκτή στις εφαρμογές.

OpenFlow 1.3 [9], εισάγει νέες δυνατότητες στην παρακολούθηση, στη λειτουργία και στη διαχείριση του δικτύου. Για τον σκοπό αυτό, προστίθεται ο πίνακας μέτρου στην αρχιτεκτονική του μεταγωγέα. Η δομή των εγγραφών του πίνακα μέτρου είναι η εξής:

[Αναγνωριστικό Μέτρου | δικλείδες μέτρου | μετρητές]

Μια εγγραφή του πίνακα μέτρου συνδέεται άμεσα με ένα κανόνα του πίνακα ροής μέσω του αναγνωριστικού της και μετρά το ποσοστό των πακέτων στα οποία αυτός έχει εφαρμοστεί. Έτσι, μπορεί να χρησιμοποιηθεί μια δικλείδα για να μειώσει το ποσοστό μιας ροής σε σχέση με την συνολική κίνηση που διέρχεται από ένα μεταγωγέα. Η δικλείδα αυτή παρεμπόδιζει τα πακέτα όταν γίνεται υπέρβαση ενός συγκεκριμένου ρυθμού. Αντί να διαγράφει πακέτα, η δικλίδα αυτή μπορεί να τροποποιεί το πεδίο των διαφοροποιημένων υπηρεσιών (DS) τους. Έτσι, απλές πολύπλοκες εφαρμογές QoS μπορούν να υλοποιηθούν από το OpenFlow 1.3 και μετά.

Η υποστήριξη πολλαπλών ελεγκτών χρησιμοποιείται για διαχείριση σφαλμάτων στο OpenFlow 1.2, με χρήση αυθαίρετων βοηθητικών συνδέσεων που στοχεύουν στην συμπλήρωση της σύνδεσης μεταξύ του κύριου ελεγκτή και του μεταγωγέα. Με αυτόν τον τρόπο επιτυγχάνεται καλύτερη εξισορρόπηση φορτίου στο επίπεδο ελέγχου. Επιπλέον εισάγεται η δυνατότητα του ανά σύνδεση φιλτραρίσματος γεγονότων. Αυτό επιτρέπει στους ελεγκτές να εγγράφονται και να δέχονται ειδοποιήσεις μόνο στους τύπους μηνυμάτων που τους ενδιαφέρουν. Τέλος το OpenFlow 1.3 υποστηρίζει το πρωτόκολλο IPv6 και τις επεκτάσεις κεφαλίδας του.

Το OpenFlow 1.4 [10] εκδόθηκε επίσημα τον Οκτώβριο του 2013. Η ONF στην έκδοση αυτή βελτίωσε την δυνατότητα επέκτασης του ταιριάσματος. Προστέθηκαν στο πρωτόκολλο, δομές πεδίου τιμής για πύλες, πίνακες και ουρές. Οι ίδιες δομές χρησιμοποιήθηκαν για να γίνουν παραμετροποιήσιμα κάποια σημεία τα όποια σε προηγούμενες εκδόσεις του πρωτοκόλλου ήταν προκαθορισμένα. Η παραμετροποί-

ηση των εικονικών πυλών είναι επίσης δυνατή. Επιπλέον, οι ελεγκτές μπορούν να στείλουν μηνύματα ελέγχου με ένα ενιαίο πακέτο στους μεταγωγείς. Συμπεριλαμβάνονται, τέλος, μικρές βελτιώσεις των πινάκων ομάδας, η δυνατότητα διαγραφής ροής σε περίπτωση πλήρωσης του πίνακα ροής και δυνατότητες παρακολούθησης.

2.4 Ο ελεγκτής OpenDaylight

Το OpenDaylight Project είναι έργο ανοικτού πηγαίου κώδικα που ιδρύθηκε τον Απρίλιο του 2013, και ανακοίνωσε την πρώτη έκδοση του, τον Φεβρουάριο του 2014. Στοχεύει στην διεύρυνση της χρησιμοποίησης των Ευφυών - Προγραμματιζόμενων Δικτύων (SDN). Πρόκειται για μία συνεργατική πλατφόρμα με την έννοια ότι κάθε χρήστης μπορεί, να συμμετάσχει στον σχεδιασμό και την επέκταση του λογισμικού μοιραζόμενος την δουλειά του με την κοινότητα. Για τον λόγο αυτό ο Opendaylight έχει μία σπονδυλωτή, ευέλικτη πλατφόρμα ελεγκτή, που εξελίσσεται συνεχώς συμπεριλαμβάνοντας επεκτάσεις λογισμικού. Ένας τέτοιος ελεγκτής τρέχει στο δικό του εικονικό μηχάνημα Java (JVM). Ως εκ τούτου, μπορεί να αναπτυχθεί σε οποιαδήποτε πλατφόρμα υλικού εφόσον το λειτουργικό της σύστημα υποστηρίζει Java. Ο OpenDaylight υποστηρίζει ένα ευρύ φάσμα πρωτοκόλλων το οποίο μάλιστα εξελίσσεται με ταχείς ρυθμούς χωρίς όμως να βασίζεται στους σκοπούς κανενός κατασκευαστή

Ο ελεγκτής εκθέτει υπερκείμενες διεπαφές που χρησιμοποιούνται από τις εφαρμογές. Οι εφαρμογές υλοποιούν την επιχειρηματική λογική και τους αλγορίθμους του δικτύου. Χρησιμοποιούν τον ελεγκτή για να μαζέψουν την γνώση του δικτύου, τρέχουν αλγορίθμους παράγουν συμπεράσματα και στην συνέχεια χρησιμοποιούν και πάλι τον ελεγκτή για να παραμετροποιήσουν το δίκτυο σύμφωνα με αυτά. Για την υλοποίηση αυτών των υπερκείμενων διεπαφών ο OpenDaylight υποστηρίζει το πλαίσιο OSGi καθώς και αμφίδρομο REST πρωτόκολλο. Η πλατφόρμα του ελεγκτή περιέχει από μόνη της κάποια βασικά συστατικά λογισμικού ώστε να μπορεί να καλύψει βασικές λειτουργικότητες του δικτύου. Παράλληλα όμως υπηρεσίες ειδικού σκοπού και επεκτάσεις υπηρεσιών μπορούν να προστεθούν, για να αναβαθμίσουν τις δυνατότητες του ελεγκτή. Πιο αναλυτικά:

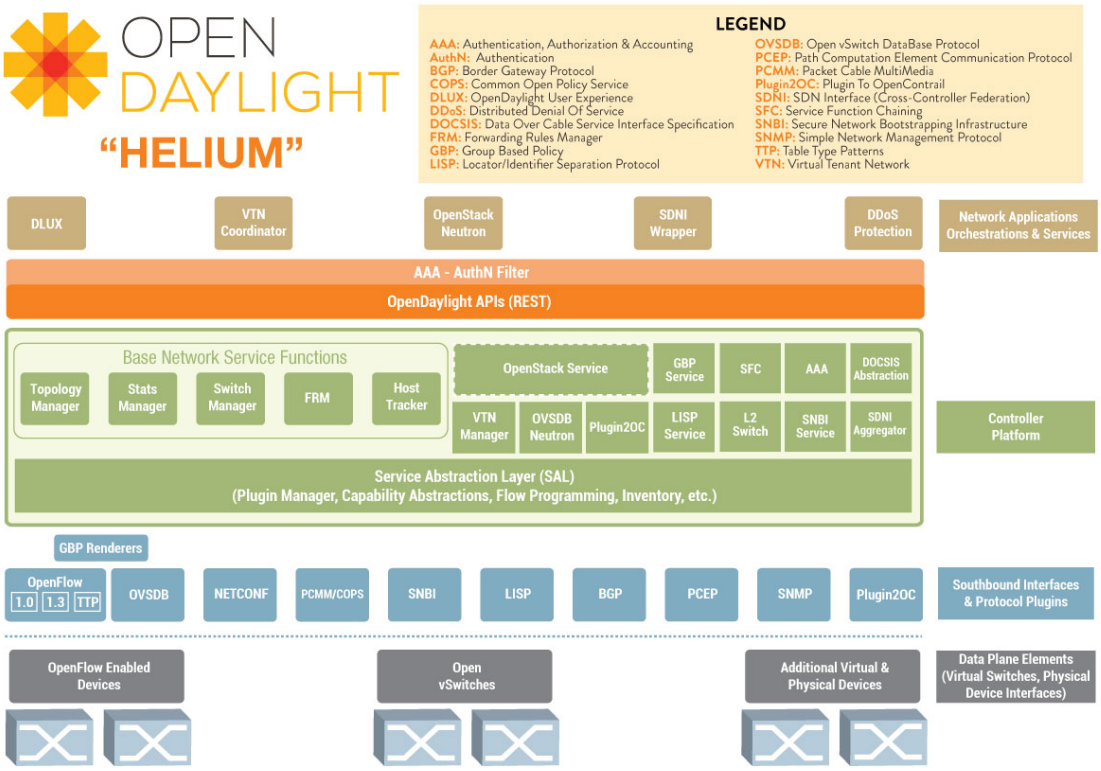
Το OSGi είναι ένα δυναμικό σύστημα στο οποίο προστίθενται συστατικά λογισμικού για συστήματα Java. Καθορίζει την διαδικασία εγκατάστασης, απεγκατάστασης, ενημέρωσης, έναρξης και διακοπής τέτοιων συστατικών. Αυτά τα συστατικά ονομάζονται και δέσμες, και είναι, στην απλούστερη μορφή τους, απλά Java αρχεία (jars). Οι δέσμες μπορούν να εγκατασταθούν, απεγκατασταθούν κλπ χωρίς να σταματήσει ή να επανεκκινηθεί το Java VM του ελεγκτή. Το πλαίσιο OSGi χρησιμοποιείται για εφαρμογές που θα τρέχουν στον ίδιο χώρο διευθύνσεων με τον ελεγκτή. Το Karaf είναι ένα μικρό OSGi που παρέχει ένα ελαφρύ δοχείο για την τοποθέτηση διαφόρων συστατικών λογισμικού.

Από την άλλη το REST (Representational State Transfer) είναι μια απλή αρχιτεκτονική χωρίς καθορισμένη εσωτερική κατάσταση που τρέχει συνήθως πάνω από HTTP. Το REST περιλαμβάνει το διάβασμα ενός αρχείου XML από μία καθορισμένη ιστοσελίδα. Το αρχείο XML αυτό, περιγράφει και περιλαμβάνει το περιεχόμενο που θέλουμε να ανταλλαχθεί. Το REST όντας διεπαφή που στηρίζεται στο διαδίκτυο, χρησιμοποιείται για εφαρμογές που δεν τρέχουν στον ίδιο χώρο διευθύνσεων ή ακόμη και στο ίδιο μηχάνημα με τον ελεγκτή.

2.4.1 Αρχιτεκτονικό Πλαίσιο

Ὅπως φαίνεται στο Σχήμα 2.4. Η πλατφόρμα ελεγκτή επικοινωνεί, με την υποκείμενη υποδομή δικτύου, μέσω υποκείμενων διεπαφών, μέσω των οποίων παρέχει βασικές υπηρεσίες δικτύωσης με μιας σειράς κλάσεων που εμφανίζονται στην Βάση δικτυακών Λειτουργιών. Κάθε ξεχωριστή εφαρμογή μπορεί να χρησιμοποιήσει αυτές τις υπηρεσίες δικτύου. Στη συνέχεια θα δούμε τις λεπτομέρειες των υπηρεσιών δικτύωσης που παρέχονται από την πλατφόρμα ελεγκτή.

- **Συντονιστής Τοπολογίας** (Topology Manager)- αποθηκεύει και διαχειρίζεται πληροφορίες σχετικές με τις διαχειριζόμενες συσκευές δικτύωσης. Όταν ξεκινήσει ο ελεγκτής, ο Συντονιστής Τοπολογίας δημιουργεί τον κόμβο ρίζας στην τοπολογία του επιχειρησιακού υποδέντρου. Στην συνέχεια ακούει τις κοινοποιήσεις και ενημερώνει αυτό το υποδένδρο, με λεπτομέρειες σχετικές με την τοπολογία, συμπεριλαμβανομένων όλων των νέων μεταγωγέων και των δια-



Σχήμα 2.4: *Αρχιτεκτονική OpenDayLight , Πηγή:http://www.opendaylight.org*

συνδέσεων μεταξύ αυτών. Ειδοποιήσεις από άλλα συστατικά, όπως ο Διαχειριστής Μεταγωγέα ή ο Διαχειριστής Συσκευής, μπορούν επίσης να παρέχουν πληροφορίες σχετικές με την τοπολογία.

- **Διαχειριστής Στατιστικών** (Statistics Manager) - υλοποιεί εργαλεία συλλογής στατιστικών, αποστολής αιτήσεων στατιστικών στοιχείων σε όλους τους ενεργούς κόμβους (διαχειριζόμενους μεταγωγείς) και αποθήκευσης των απαντήσεων στο λειτουργικό υποδένδρο στατιστικών. Ο Διαχειριστής Στατιστικών εκθέτει επίσης υπερκείμενες διεπαφές για να επιστρέψει τις σχετικές με την πόρτα μεταγωγέα, το μέτρο το πίνακα ροών και τα στατιστικά ομάδας πληροφορίες. Επιπλέον, η έκδοση Helium του ελεγκτή OpenDaylight επιτρέπει την ρύθμιση του χρονικού διαστήματος μεταξύ διαδοχικών ερωτημάτων στατιστικών.

- **Διαχειριστής Μεταγωγέα** (Topology Manager)- παρέχει λεπτομέρειες για τους κόμβους του δικτύου (μεταγωγείς) και τις υποδοχές τους (θύρες μεταγωγέων). Από τη στιγμή που ο ελεγκτής ανακαλύπτει τα στοιχεία του δικτύου, οι

παράμετροι τους αποθηκεύονται στο δέντρο δεδομένων του Διαχειριστή Μεταγωγεα. Μπορεί να χρησιμοποιηθεί η υπερκείμενη διεπαφή για να ληφθούν πληροφορίες σχετικά με τις κόμβους και τις θύρες των συσκευών.

- **Διαχειριστής Προώθησης Κανόνων** (Statistics Manager) - διαχειρίζεται βασικούς κανόνες προώθησης (όπως οι κανόνες OpenFlow), επιλύει τις διαφορές τους, και να τους επικυρώνει. Ο Διαχειριστής Προώθησης Κανόνων επικοινωνεί με τα υποκείμενα κομμάτια κώδικα και φορτώνει με OpenFlow κανόνες τους διαχειρίζόμενους μεταγωγείς.
- **Διαχειριστής Απογραφής** ρωτάει και κρατάει ενημερωμένες τις αποθηκευμένες πληροφορίες σχετικά με τους μεταγωγείς και τις πόρτες που διαχειρίζεται ο OpenDaylight, και εξασφαλίζει ότι η βάση δεδομένων απογραφής είναι ακριβής και ενημερωμένη.
- **Tracker Χρηστή** - αποθηκεύει πληροφορίες σχετικά με τους τελικούς χρήστες (διεύθυνση στρώμα δεδομένων, τύπος μεταγωγέα, τύπος πόρτας , διεύθυνση του δικτύου), και παρέχει διεπαφές που ανακτούν πληροφορίες για τους τελικούς αυτούς κόμβους. Ο Tracker Χρήστη μπορεί να λειτουργήσει με στατικό ή δυναμικό τρόπο. Στην περίπτωση της δυναμικής λειτουργίας, ο tracker χρησιμοποιεί το ARP για να παρακολουθεί την κατάσταση της βάσης δεδομένων. Στη στατική λειτουργία, η βάση δεδομένων διαφημίζεται μέσω υπερκείμενων διεπαφών.

Οι υποκείμενες διεπαφές που εκθέτει ο ελεγκτής μπορούν να υποστηρίξουν πολλαπλά πρωτόκολλα τα οποία προσθέτονται στην βασική πλατφόρμα ως ξεχωριστά κομμάτια λογισμικού (plugins) τέτοια έιναι το Openflow 1.0, Openflow 1.3 , BGP-LS, κ.λπ. Τα κομμάτια αυτά λογισμικού συνδέονται δυναμικά σε ένα Στρώμα Αφαίρεσης (Service Abstraction Layer -SAL). Το SAL εκθέτει τις λειτουργικές δυνατότητες των συσκευών του δικτύου με βάση τις οποίες τα υπερκείμενα συστατικά λογισμικού γράφονται και καθορίζει το πώς θα υλοποιηθούν αυτές, ανεξάρτητα από το πρωτόκολλο που χρησιμοποιείται μεταξύ του ελεγκτή και των συσκευών του δικτύου. Ουσιαστικά ο OpenDaylight και γενικότερα τα SDN προσαρμόζουν τις δυνατότητες του υλικού στις απαιτήσεις των εφαρμογών που υλοποιούνται στο ανώτερο επίπεδο ελέγχου.

Το SAL είναι σαν μια κοινή γλώσσα κατανοητή από τις υπερκείμενες και υποκείμενες διεπαφές. Όπως ξαναείπαμε το OSGi επιτρέπει την δυναμική σύνδεση εξειδικευμένων κομματιών κώδικα για τα πρωτόκολλα του υποκειμένου. Το SAL με την σειρά του παρέχει βασικές υπηρεσίες, όπως εντοπισμό συσκευών που χρησιμοποιούνται σε συστατικά όπως ο Διαχειριστής Τοπολογίας για να πάρει την τοπολογία και πληροφορίες σχετικά με τις δυνατότητες των συσκευών. Οι υπηρεσίες στην συνέχεια κατασκευάζονται χρησιμοποιώντας τα χαρακτηριστικά γνωρίσματα που εκτίθενται από τα εξειδικευμένα κομμάτια κώδικα και τις συσκευές του δικτύου. Με βάση την αίτηση υπηρεσίας η SAL βρίσκει το κατάλληλο κομμάτι λογισμικού και αποφασίζει ποια είναι η πιο κατάλληλη υποκείμενη διεπαφή για την αλληλεπίδραση με μια συγκεκριμένη συσκευή δικτύου. Η απόφαση αυτή μπορεί επίσης να παρθεί χειροκίνητα .

2.4.2 Ανάγκες σε υλικό

Ο ελεγκτής είναι ένα κοινό εικονικό μηχάνημα Java και συνεπώς μπορεί να τρέξει σε οποιοδήποτε μηχάνημα και οποιοδήποτε λειτουργικό υποστηρίζει Java 1.7. Ταυτόχρονα έχει ένα ενσωματωμένο γραφικό περιβάλλον το οποίο οποίο υλοποιείται σαν εφαρμογή χρησιμοποιώντας την κοινή υπερκείμενη διεπαφή.

Κεφάλαιο 3

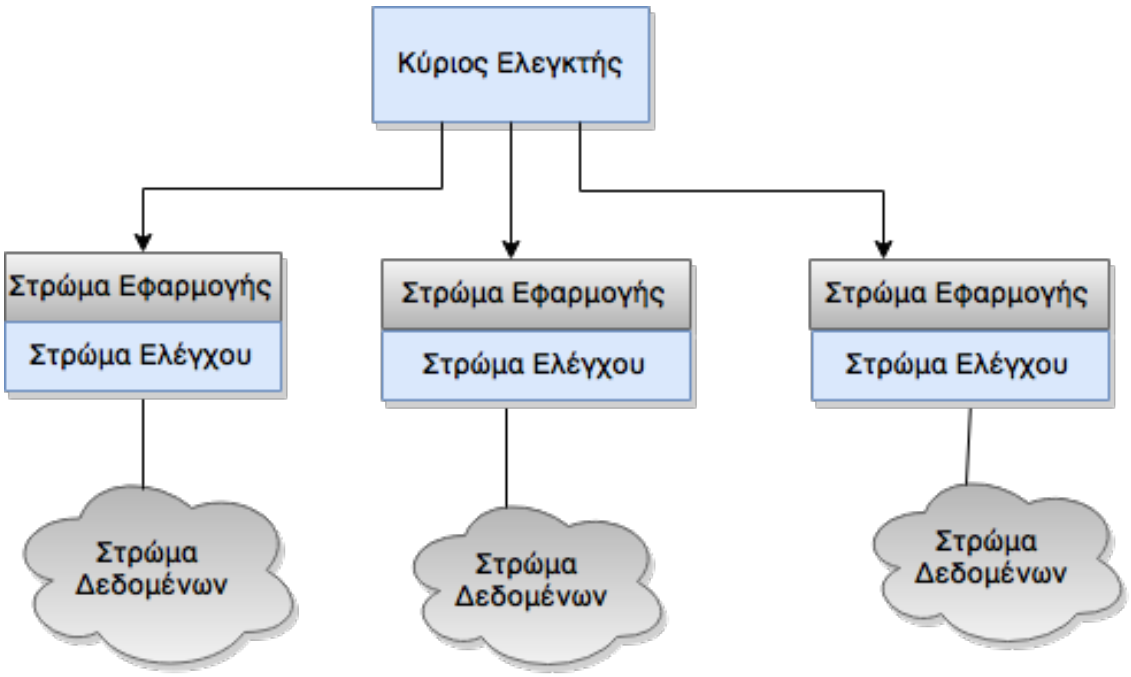
Η Επικοινωνία μεταξύ των SDN Domains

Η επικοινωνία μεταξύ των ελεγκτών που διοικούν διαφορετικούς δικτυακούς τομείς είναι επιτακτική ανάγκη δεδομένης της αύξησης των δικτύων που χρησιμοποιούν SDN. Ως SDN domain θεωρούμε κάθε τμήμα του δικτύου το οποίο υποστηρίζει το SDN, καθορίζεται από το διαχειριστή και διοικείται από έναν ελεγκτή όπως αυτό ορίζεται στο [11]. Υπάρχει πληθώρα διαφορετικών εφαρμογών που προϋποθέτει την επικοινωνία μεταξύ SDN domains από την απλή δρομολόγηση μέχρι την αντιμετώπιση διαδικτυακών επιθέσεων και την δέσμευση πόρων. Το θέμα έχει απασχολήσει μεγάλο μέρος της ακαδημαϊκής αλλά και της εφαρμοσμένης έρευνας χωρίς ακόμα να έχει βρεθεί μια λύση που να ικανοποιεί πλήρως τις απαιτήσεις. Μέχρι στιγμής έχουν προταθεί τρεις βασικές αρχιτεκτονικές προσεγγίσεις οι οποίες θα αναλυθούν εκτενέστερα παρακάτω.

3.1 Το κάθετο αρχιτεκτονικό μοντέλο

Το κάθετο μοντέλο ή το Μοντέλο Συντονιστή όπως φαίνεται στο Σχήμα 3.1 αποτελείται από μία κύρια οντότητα που συντονίζει ένα σύνολο διαφορετικών SDN domains. Κάθε ένας από αυτά τα Domains βρίσκεται κάτω από την διαχείριση κάποιου ελεγκτή. Η κύρια αυτή οντότητα, που θα μπορούσε να θεωρηθεί και ως ελεγ-

κτής των ελεγκτών επικοινωνεί με αυτούς μέσω υπερκείμενων διεπαφών. Στην περίπτωση αυτή ο Συντονιστής οφείλει να έχει πλήρη εικόνα της τοπολογίας και των δυνατοτήτων των domains που βρίσκονται κάτω από την διαχείριση του ενώ ταυτόχρονα να μπορεί να τροποποιήσει την συμπεριφορά τους. Είναι στην συνέχεια υπεύθυνος για την διάδοση αυτών των δεδομένων στους υπόλοιπους ελεγκτές ανάλογα με την πολιτική του εκάστοτε domain.

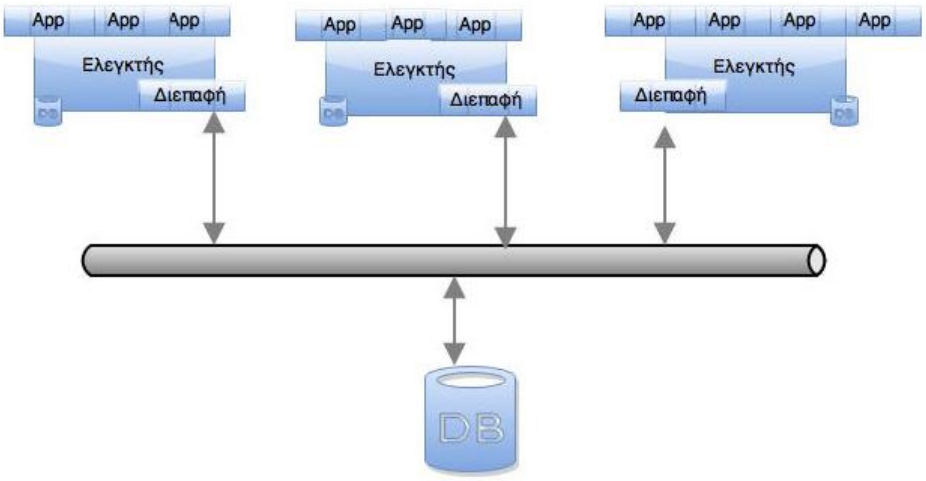


Σχήμα 3.1: *Το κάθετο αρχιτεκτονικό μοντέλο*

Οι απαιτήσεις αυτές, ωστόσο, εγείρουν σοβαρούς κινδύνους όσο αφορά την ασφάλεια των domains. Κομμάτια δικτύων είναι συχνά απρόθυμα να δημοσιοποιήσουν την ακριβή τοπολογία τους πόσο μάλλον να επιτρέψουν σε μια τρίτη οντότητα να έχει πρόσβαση στην παραμετροποίηση και κατ' επέκταση στη διαχείριση του δικτύου τους. Έτσι η λύση αυτή περιορίζεται μόνο σε μικρά δίκτυα που βρίσκονται κάτω από την ίδια διαχειριστική αρχή. Ταυτόχρονα, είναι προφανές ότι αυτή η διατήρηση ενός και μόνο συντονιστή στον οποίο συνδέονται όλα τα domains δεν είναι κλιμακώσιμη ενώ ταυτόχρονα αποτελεί μοναδικό σημείο αποτυχίας. Συνεπώς, για μικρά δίκτυα που βρίσκονται κάτω από κοινή ιδιοκτησία, η αρχιτεκτονική αυτή είναι η πιο αποτελεσματική λόγω της απλότητας και της ευκολίας στην διαχείριση.

3.2 Το αρχιτεκτονικό μοντέλο του διαύλου

Η δεύτερη αρχιτεκτονική προσέγγιση όπως φαίνεται στο Σχήμα 3.2 ακολουθεί το μοντέλο λογισμικού του καναλιού-διαύλου. Τα βασικά δομικά στοιχεία της αρχιτεκτονικής αυτής είναι ένας δίαυλος επικοινωνίας ειδικού σκοπού και μία, είτε κεντρική είτε κατανεμημένη, βάση δεδομένων. Οι ελεγκτές εγγράφονται σε συγκεκριμένες υπηρεσίας που χρειάζονται. Με τον τρόπο αυτό, οι ελεγκτές δηλώνουν ότι θέλουν να ενημερώνονται για αλλαγές σε κάποιο κομμάτι του δικτύου ή ζητούν να τους σταλεί το αποτέλεσμα της εκτέλεσης κάποιου αλγοριθμικού ή στατιστικού υπολογισμού. Οι υπηρεσίες αυτές είναι συνήθως υλοποιημένες σαν ξεχωριστά κομμάτια κώδικα και είτε δημοσιεύουν τα αποτελέσματα τους στο δίαυλο είτε στέλνουν απευθείας μηνύματα.



Σχήμα 3.2: *Το αρχιτεκτονικό μοντέλο του διαύλου*

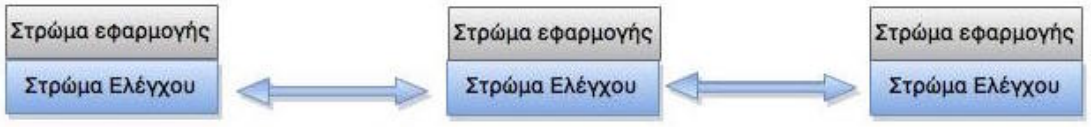
Η αρχιτεκτονική αυτή προσέγγιση είναι πολύ αποτελεσματική από άποψη οικονομίας πόρων, γιατί τα αποτελέσματα μιας υπηρεσίας υπολογίζονται μία φορά και καταναλώνονται από πολλούς ελεγκτές ενώ ταυτόχρονα οι διάφορες ειδοποιήσεις στέλνονται πολυ γρήγορα από έναν ελεγκτή ή υπηρεσία σε όσους ενδιαφέρονται μέσω του διαύλου επικοινωνίας. Ταυτόχρονα είναι πιο εύκολη η δυναμική δέσμευση πόρων αυξάνοντας την ανταποκρισιμότητα του συστήματος. Αν δηλαδή μία υπηρεσία έχει πολύ ζήτηση για κάποιο χρονικό διάστημα αφιερώνονται περισ-

σότεροι υπολογιστικοί πόροι για αυτήν, και εφόσον μιλάμε για λογισμικό δημιουργούνται απλώς περισσότερα στιγμιότυπα μίας κλάσης που υλοποιεί την υπηρεσία. Ταυτόχρονα η αρχιτεκτονική αυτή προσέγγιση είναι πολύ πιο αποδοτική από άποψη κλιμακωσιμότητας.

Παρά τα αδιαμφισβήτητα πλεονεκτήματα της, η βασική αδυναμία της αρχιτεκτονικής έχει και πάλι να κάνει με την ασφάλεια. Τα βασικότερα ερωτήματα που τίθενται είναι ποίος θα ελέγχει το δίαυλο και πώς θα προστατεύονται η ιδιοτικώτητα των ανακοινώσεων που έχουν συγκεκριμένο παραλήπτη χωρίς να αυξάνεται πολύ το υπολογιστικό κόστος. Είναι απαραίτητο επίσης να επισημανθεί ότι για να απολαμβάνει ένα σύστημα τα πλεονεκτήματα που αναφέρθηκαν πρέπει να αποτελείται από ένα σχετικά μεγάλο αριθμό ελεγκτών, διαφορετικά η υλοποίηση του συστήματος δεν έχει νόημα. Συνεπώς η λύση αυτή έχει μικρή πιθανότητα σταδιακής υιοθέτησης σε πραγματικές συνθήκες σύγχρονου δικτύου.

3.3 Το οριζόντιο αρχιτεκτονικό μοντέλο

Η τρίτη αρχιτεκτονική που έχει προταθεί είναι η οριζόντια. Τα βασικότερα της χαρακτηριστικά φαίνονται στο Σχήμα 3.3 . Σύμφωνα με αυτήν οι ελεγκτές επικοινωνούν ο ένας με τον άλλο ανά δύο χωρίς να τηρούν κάποιο συγκεκριμένο σχηματισμό ή ιεραρχία. Οι ελεγκτές απλώς εγκαθιστούν απ' άκρη σε άκρη σύνδεση με γειτονικούς ελεγκτές χρησιμοποιώντας υπάρχοντα πρωτόκολλα επικοινωνίας. Η αρχιτεκτονική αυτή αν και φαίνεται απλουστευτική, διατηρεί το πλεονέκτημα της κλιμακοσιμότητας και της δυνατότητας σταδιακής υιοθέτησης και χρήσης από περισσότερους ελεγκτές χωρίς να επηρεάζεται η αποτελεσματικότητα και η επίδοση του συνολικού συστήματος.



Σχήμα 3.3: *Το οριζόντιο αρχιτεκτονικό μοντέλο*

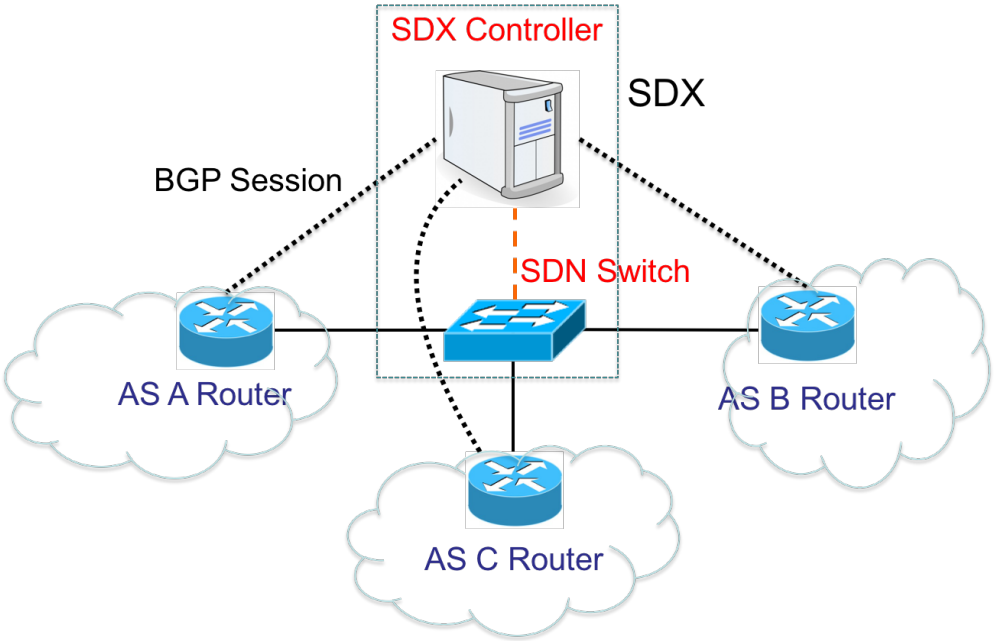
3.4 Χαρακτηριστικά Παραδείγματα Αρχιτεκτονικών Μοντέλων

3.4.1 Software Defined Internet Exchange (SDX)

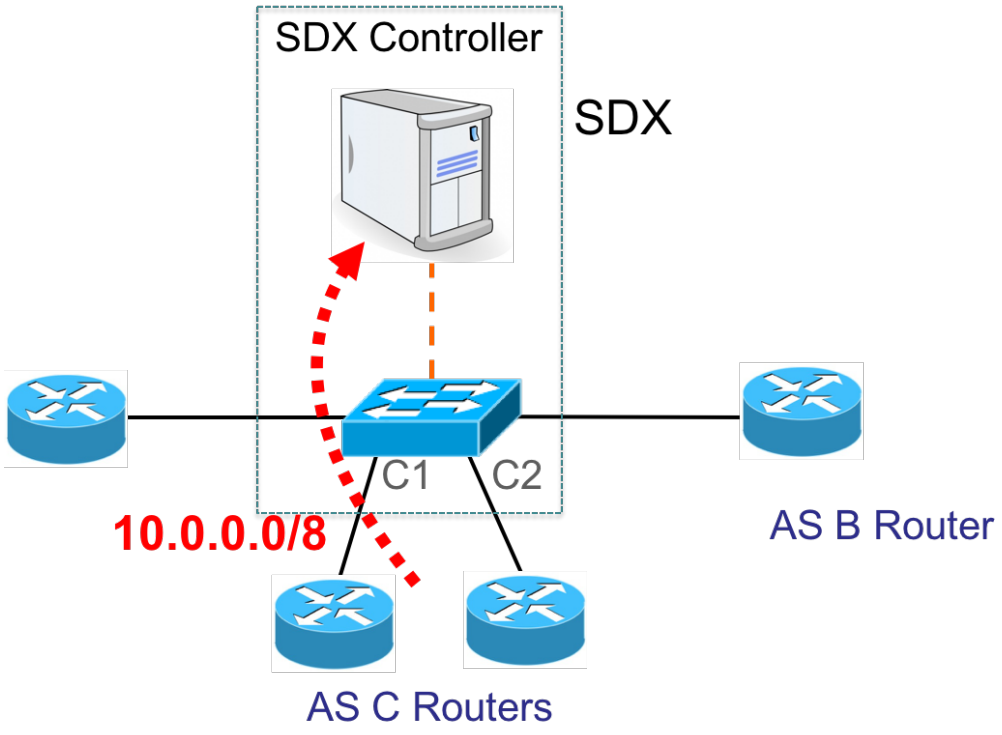
Το SDX [12], [13] μπορεί να θεωρηθεί ως παράδειγμα του κάθετου αρχιτεκτονικού μοντέλου. Στοχεύει στην βελτίωση των λειτουργικών χαρακτηριστικών των σημείων ανταλλαγής κίνησης (IXPs) με τις δυνατότητες των Ευφυών Προγραμματιζόμενων δικτύων. Το SDX είναι μια πλατφόρμα που επιτρέπει σε πολλούς ενδιαφερόμενους φορείς να καθορίζουν τις πολιτικές / εφαρμογές τους πάνω σε μια κοινή υποδομή. Για να γίνουν καλύτερα κατανοητά τα κίνητρα αλλά και η αξία του SDX θα αναφερθούμε επιγραμματικά στα μειονεκτήματα του BGP τα οποία το SDX επιχειρεί να υπερκαλύψει.

Από τις βασικότερες αδυναμίες του BGP, λοιπόν είναι ότι το μοναδικό κριτήριο για την δρομολόγηση κίνησης είναι το πρόθεμα της IP διεύθυνσης. Επίσης οι όποιες αλλαγές κάνουμε στα διαφημιζόμενα προθέματα επηρεάζουν άμεσα μόνο τους γειτονικούς διαχειριστικούς τομείς δηλαδή το επόμενο βήμα του μονοπατιού, ενώ δεν υπάρχει κάποιο μέσο για να οριστεί ολόκληρο το μονοπάτι που θα ακολουθήσει μια ροή πακέτων .Οι αδυναμίες αυτές μπορούν να αντιμετωπιστούν με την χρήση των Ευφυών Προγραμματιζόμενων δικτύων τα οποίο προφανώς επιτρέπουν την επιλογή δρομολόγηση με βάση οποιοδήποτε πεδίο της επικεφαλίδας ενός πακέτου και προσφέρουν άμεσο και ευθύ προγραμματισμό και έλεγχο της δρομολόγησης απ' άκρης σ' άκρη .

Ένα σημείο ανταλλαγής κίνησης (IX ή IXP) είναι μια φυσική υποδομή μέσω της οποίας οι πάροχοι υπηρεσιών Διαδικτύου (ISP) και τα Δίκτυα Παροχής Περιεχομένου (CDNs) ανταλλάσσουν κίνηση στο διαδίκτυο. Ουσιαστικά IXPs είναι μια συμφωνία οικονομικού ενδιαφέροντος καθώς ελαχιστοποιεί το τμήμα της κίνησης ενός παρόχου που πρέπει να παραδοθεί μέσω παρόχων ανώτερου επιπέδου, μειώνοντας έτσι το μέσο κόστος ανά bit παρεχόμενης υπηρεσίας. Ταυτόχρονα με αυτόν τον τρόπο βελτιώνεται η απόδοση της δρομολόγησης και η ανοχή σε σφάλματα.



Σχήμα 3.4: *Παραδοσιακή αρχιτεκτονική Internet Exchange Point (IXP), Πηγή: [11]*



Σχήμα 3.5: *Software Defined Internet Exchange (SDX), Πηγή: [11]*

Πίο αναλυτικά, διατηρώντας τα βασικά χαρακτηριστικά της παραδοσιακής αρχιτεκτονικής των IXPs όπως φαίνεται στο Σχήμα 3.4 χρησιμοποιείται ένας ελεγκτής

ο οποίος είναι υπεύθυνος για το στρώμα ελέγχου αντικαθιστώντας τον παλιότερο εξυπηρετητή δρομολόγησης και έναν SDN μεταγωγέα που είναι υπεύθυνος για το στρώμα δεδομένων. Μια απλουστευμένη εκδοχή της εξέλιξης της αρχιτεκτονικής φαίνεται στο Σχήμα 3.5 . Οι συμμετέχοντες στο SDX εκφράζουν τις πολιτικές δρομολόγησης τους σε μία υψηλού επιπέδου γλώσσα πάνω από την Pyretic. Οι πολιτικές αυτές έχουν την μορφή ”Αν ταίριασμα τότε Ενέργεια” και δηλώνονται στον ελεγκτή ανεξάρτητα για κάθε συμμετέχοντα. Οι συμμετέχοντες μπορούν να ορίσουν πολιτικές για τα πακέτα που ξεκινούν ή κατευθύνονται σε αυτούς χρησιμοποιούν μόνο διαδρομές που τους έχουν γίνει γνωστές μέσω των BGP διαφημίσεων. Στην συνέχεια ο ελεγκτής πρέπει να συνθέσει τις επιμέρους πολιτικές σε μια ενιαία και να εκφράσει αυτήν την πολιτική σε κανόνες δρομολόγησης στον μεταγωγέα. Οι βασικές προκλήσεις που καλείται να αντιμετωπίσει μία τέτοια αρχιτεκτονική είναι η ορθότητα της σύνθεσης των πολιτικών και η κλιμακωσιμότητα. Η ορθότητα των τελικών κανόνων επηρεάζεται από τον τρόπο που θα γίνει η σύνθεση των επιμέρους πολιτικών και από τυχόν αντιφάσεις που υπάρχουν μεταξύ τους. Για να αποφευχθεί αυτό κάθε συμμετέχον βλέπει συνδεδεμένους στον μεταγωγέα μόνο τους συμμετέχοντες με τους οποίους έχει δικαίωμα να επικοινωνήσει Επιπλέον, οι ξεχωριστές πολιτικές προστίθενται διαδοχικά και συγκρίνονται με τα δεδομένα του BGP για να αποφευχθούν σφάλματα.

Το SDX πρέπει να αντιμετωπίσει θέματα κλιμακοσιμότητας τόσο στον χώρο όσο και στον χρόνο. Πιο αναλυτικά η αύξηση των συμμετεχόντων συνεπάγεται μεγάλη αύξηση στους κανόνες που πρέπει να εγκαθίστανται στο στρώμα δεδομένων (μεταγωγέα) καθώς και στον υπολογιστικό χρόνο που απαιτείται για την σύνθεση και την μετατροπή των πολιτικών σε κανόνες.

Για να αντιμετωπιστεί το πρόβλημα του περιορισμένου χώρου του μεταγωγέα το SDX ομαδοποιεί τα διαφορετικά IP προθέματα σε ομάδες ανάλογα με την συμπεριφορά δρομολόγησης και καταχωρεί ένα κανόνα για κάθε ομάδα. Πιο συγκεκριμένα, τα προθέματα αντιστοιχίζονται πρώτα σε κλάσεις που ονομάζονται “κλάσης ισοδύναμης προώθησης” και κάθε κλάση αντιστοιχίζεται στην συνέχεια σε μία ενέργεια. Η λειτουργία αυτή προϋποθέτει δύο πίνακες ο πρώτος από τους οποίους παραμένει πολύ μεγάλος λόγο του μεγάλου αριθμού προθεμάτων. Για τον λόγο αυτό, εκ

πρώτης όψεως δεν φαίνεται να εξοικονομείται καθόλου χώρος. Στην πράξη όμως, αυτός ο πίνακας κατανέμεται στους συνοριακούς δρομολογητές των διαχειριστικών τομέων. Πιο συγκεκριμένα το SDX εκμεταλλεύεται το γεγονός ότι οι δρομολογητές στα άκρα του δικτύου κρατάνε ούτως ή άλλως μια εγγραφή για κάθε πρόθεμα προορισμού και χρησιμοποιούν το next hop ως αναγνωριστικό της ομάδας. Τελικά ένας μεταγωγές μπορεί να εξυπηρετήσει εκατοντάδες συμμετέχοντες με λιγότερους από 30k εγγραφές.

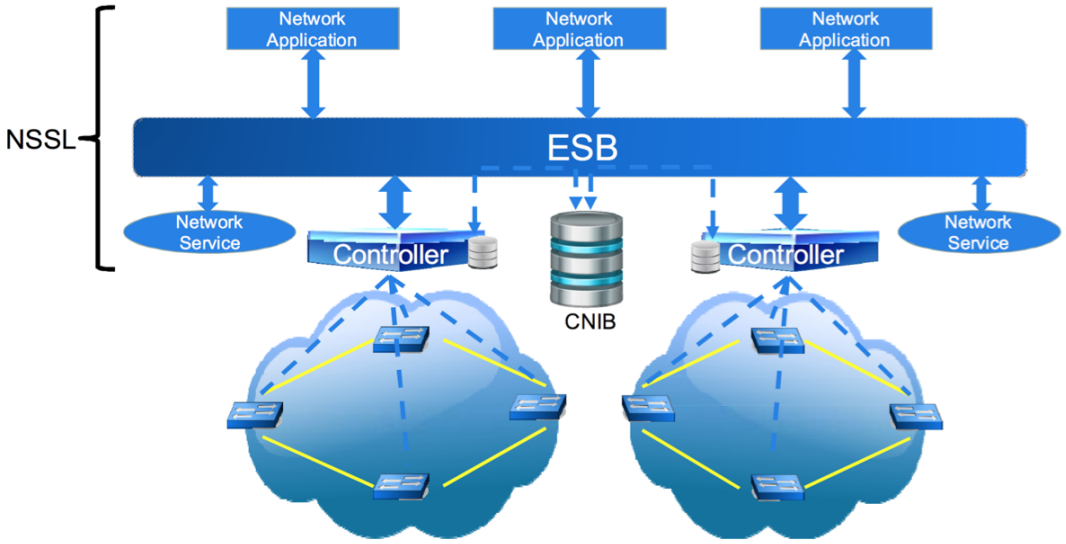
Για τον περιορισμό του υπολογιστικού χρόνου για την σύνθεση των κανόνων το SDX εκμεταλλεύεται κάποια χαρακτηριστικά για να αποφευχθεί η εξαντλητική μέθοδος. Πιο αναλυτικά, αποφεύγεται η προσπάθεια σύνθεσης πολιτικών που δεν έχουν σχέση μεταξύ τους καθώς αυτή η διαδικασία είναι υπολογιστικά ακριβή. Παράλληλα η προσπάθεια επικεντρώνεται στην εκμετάλλευση των παρατηρήσεων ότι οι ενημερώσεις πολιτικής επηρεάζουν μόνο ένα μικρό αριθμό ομάδων και ότι ομάδες που ανανεώθηκαν πριν από μεγάλο σχετικά χρονικό διάστημα έχουν περισσότερες πιθανότητες να επηρεαστούν. Πρέπει να σημειωθεί ότι η προσπάθεια βελτίωσης της κλιμακωσιμότητας στο χώρο και στον χρόνο αποτελούν trade-off. Είναι δυνατό δηλαδή να μειωθεί ο υπολογιστικός χρόνος θυσιάζοντας κάποιους παραπάνω κανόνες από την βέλτιστη λύση και αντίστροφα να αφιερωθεί περισσότερος χρόνος για να βρεθεί η βέλτιστη λύση δηλαδή οι ελάχιστες εγγραφές.

Η μεγάλη διαφορά αυτής της προσπάθειας σε σχέση με άλλες προτεινόμενες αρχιτεκτονικές είναι ότι διατηρεί σε μεγάλο βαθμό την λογική του BGP τόσο για τον έλεγχο της ορθότητας των πολιτικών όσο και για τον υπολογισμό τους.

3.4.2 Service oriented Software Defined Networks (SSDN)

Το SSDN [14] είναι μια τυπική υλοποίηση της αρχιτεκτονικής προσέγγισης του καναλιού, που δίνει έμφαση στις υπηρεσίες. Πιο αναλυτικά, αποτελείται από μία βάση δεδομένων, ένα δίαυλο ένα στρώμα που ονομάζεται "στρώμα υπηρεσιών λογισμικού" και περιέχει εφαρμογές και υπηρεσίες. Οι υπηρεσίες επεξεργάζονται δεδομένα της βάσης ή αποτελέσματα άλλων υπηρεσιών και παράγουν νέα αποτελέσματα τα οποία χρησιμοποιούνται από τις εφαρμογές. Ο ελεγκτής έχει δύο ειδών διεπαφές

ένα για την χρησιμοποίηση εξωτερικών συστατικών λογισμικού που είναι τύπου REST και μία για την υλοποίηση εφαρμογών στο εσωτερικό. Πιο αναλυτικά ένας ελεγκτής είναι το αποτέλεσμα της ένωσης τμημάτων λογισμικού που είτε υλοποιούνται εσωτερικά είτε χρησιμοποιούνται μέσω του διαύλου αφού ο ελεγκτής εγγραφεί στην κατάλληλη υπηρεσία. Από μια πιο αφηρημένη άποψη τόσο ελεγκτές όσο και οι εφαρμογές μπορεί να γίνουν παραγωγοί και καταναλωτές υπηρεσιών. Οι μεν στέλνουν ένα αίτημα για ικανοποίηση και οι δε την απάντηση μέσω του διαύλου. Μία εφαρμογή μπορεί να χρησιμοποιεί τα αποτελέσματα μιας άλλης μαζί με δεδομένα από την βάση και να στέλνει τα δικά της αποτελέσματα σε κάποιους ελεγκτές ή/και σε κάποιες εφαρμογές. Για να εξυπηρετήσει τις ανάγκες αυτής της επικοινωνίας, ο δίαυλος έχει δύο μεθόδους αποστολής μηνυμάτων: από αποστολέα σε μοναδικό παραλήπτη και από έναν αποστολέα πχ μια υπηρεσία σε όλους όσους έχουν εγγραφεί σε αυτήν. Μια απλουστευμένη εκδοχή της αρχιτεκτονικής φαίνεται στο Σχήμα 3.6.



Σχήμα 3.6: *Service Software Defined Networks (SSDN) Πηγή: [11]*

Άλλα οαραδείγματα της αρχιτεκτονικής του διαύλου έιναι τα DIstributed SDN COntrol plane (DISCO) [**?**]

3.4.3 Network Service Interface (NSI)

Το NSI [15] ακολουθεί το οριζόντιο αρχιτεκτονικό μοντέλο. Για την καλύτερη κατανόηση των αρχών και της σύνδεσης του με την επικοινωνία μεταξύ SDN τομέων θα δώσουμε πρώτα μια πιο γενική ιδέα για την αρχιτεκτονική και το σκοπό του.

Το NSI ή αλλιώς Δικτυακή Διεπαφή Υπηρεσιών είναι μια προσπάθεια να παραχθεί μια κοινή διεπαφή που να ορίζει τόσο την οριζόντια επικοινωνία μεταξύ δυο παρόχων δικτυακών υπηρεσιών όσο και την κάθετη επικοινωνία μεταξύ εξωτερικών οντοτήτων όπως χρήστες με αυτούς τους παρόχους. Κινητήρια δύναμη αυτής της προσπάθειας είναι η εύρυθμη λειτουργία και ανταλλαγή δικτυακών υπηρεσιών σε ετερογενή περιβάλλοντα, πολλαπλών διοικητικών τομέων. Το NSI είναι ένα πλαίσιο πρωτοκόλλων δέσμευσης πόρων μεταξύ πολλαπλών διαχειριστικών τομέων. Ορίζει ένα είδος μηνυμάτων , τις επικεφαλίδες τους και και τις αρχές της επεξεργασίας αυτών. Οι βασικές αρχές της αρχιτεκτονικής του NSI θυμίζουν αυτές των Ευφυών Προγραμματιζόμενων δικτύων με την έννοια ότι το δίκτυο χωρίζεται σε τρία επίπεδα αυτά του Ελέγχου, της Μεταφοράς και των Υπηρεσιών.

Το επίπεδο Υπηρεσιών αποτελείται από πράκτορες δικτυακών υπηρεσιών. Οι πράκτορες δικτυακών υπηρεσιών ή NSA (Network Service Agent) υλοποιούνται στο λογισμικό, και είναι υπεύθυνοι για κάποιους πόρους του στρώματος μεταφοράς. Το επίπεδο υπηρεσιών δικτύων είναι σχεδιασμένο για να προσφέρει ένα ευρύ φάσμα υπηρεσιών. Κάθε τέτοια υπηρεσία έχει ένα σχετικό ορισμό ο οποίος καθορίζει το πεδίο εφαρμογής της υπηρεσίας και προσδιορίζει κάθε παράμετρο που πρέπει να συνοδεύει το αίτημα για να εξυπηρετηθεί. Οι υπηρεσίες Δικτύου επιτρέπουν στις εφαρμογές να παρακολουθούν, να ελέγχουν, να ρωτάνε και να υποστηρίζουν τους πόρους του δικτύου που διατίθενται από τον εξυπηρετητή του δικτύου.

Οι πράκτορες (NSA) παρέχουν λοιπόν δικτυακές υπηρεσίες μία από τις οποίες είναι και η σύνδεση με κάποιον άλλο πράκτορα για την αίτηση κάποιας υπηρεσίας. Κάθε υπηρεσία υλοποιείται μέσω της ανταλλαγής μηνυμάτων μεταξύ των πρακτόρων. Η επικοινωνία των πρακτόρων ορίζεται μέσω ενός ειδικού πρωτοκόλλου που ονομάζεται NSI-CS και δημιουργήθηκε το 2011. Πιο συγκεκριμένα, οι πράκτορες στέλνουν και λαμβάνουν NSI μηνύματα και περιέχουν ένα σύνολο δυνατοτήτων

που επιτρέπουν την διεξαγωγή και παράδοση υπηρεσιών. Ένας πράκτορας μπορεί να λειτουργήσει είτε ως πάροχος μίας υπηρεσίας είτε ως αιτών είτε να έχει και τους δύο ρόλους ταυτόχρονα. Το NSI ορίζει στην ουσία την διεπαφή μεταξύ του παρόχου και του αιτούν πράκτορα καθώς και μια σειρά αλληλεπιδράσεων και συναλλαγών μεταξύ τους για την προετοιμασία μίας υπηρεσίας.

Όσο αφορά στην υπηρεσία της σύνδεσης, που είναι και η πιο σχετική με το θέμα του παρόντος κειμένου, πρόκειται για την εγκατάσταση ενός αγωγού που μεταφέρει με διαφανή τρόπο τις πληροφορίες χρήστη από την πόρτα εισόδου στην πόρτα εξόδου. Μία σύνδεση έχει προφανώς κάποια χαρακτηριστικά όπως το μονοπάτι δηλαδή μια ταξινομημένη λίστα από κόμβους δρομολόγησης και κάποιες ιδιότητες σχετικές με την αποδοτικότητα της σύνδεσης όπως η χωρητικότητα, η χρονική στιγμή εκκίνησης, επαλήθευση ταυτότητας και ένα αναγνωριστικό μοναδικό ανά ζεύγος πρακτόρων. Η σύνδεση πραγματοποιείται στο επίπεδο μεταφοράς. Ωστόσο η διαπραγμάτευση γίνεται στο επίπεδο υπηρεσιών από τους πράκτορες που πρέπει να διαπιστώσουν ότι υπάρχουν οι απαραίτητοι πόροι και να τους δεσμεύσουν για συγκεκριμένο χρόνο. Συχνά υλοποιείται ένας πράκτορας ανά διαχειριστικό τομέα. Ο πράκτορας μπορεί να συνδέεται με έναν μοναδικό Διαχειριστής πόρων δικτύου(NRM). Πρόκειται για ένα ξεχωριστό κομμάτι λογισμικού που διαχειρίζεται το τμήμα των πόρων του δικτύου κατά αποκλειστικότητα δηλαδή μπορεί να αποδεχθεί ή να απορρίψει οποιαδήποτε αίτηση από το NSI για τους τοπικούς πόρους.

Όπως έχει γίνει ήδη εμφανές το NSI αν και δεν προοριζόταν για τα Ευφυή Προγραμματιζόμενα δίκτυα εντούτοις έχει χαρακτηριστικά που το καθιστούν συμβατό εννοιολογικά ώστε να μπορεί σχετικά εύκολα να υιοθετηθεί για να καλύψει την ανάγκη της επικοινωνίας διαφορετικών διαχειριστικών τομέων.

Πρώτα από όλα, το NSI είναι είναι αυτόνομο, κατανεμημένο, μοντέλο επικοινωνίας με την έννοια ότι (α) λειτουργεί ανεξάρτητα από το επίπεδο μεταφοράς και συνεπώς είναι εξαιρετικά ευέλικτο. (β)Αναθέτει την διαχειριστή των πόρων εσωτερικά του δικτύου στον (NRM) και(γ) εξασφαλίζει την ασφάλεια της επικοινωνίας ως μέρος της συνολικής αρχιτεκτονικής. Επιπλέον το γεγονός ότι NSI είναι η τεχνολογία η οποία δεν υπαγορεύει συγκεκριμένες τεχνολογίες εντός του τομέα επιτρέπει την υιοθέτηση καινοτόμων τεχνολογιών μεταγωγής, όπως OpenFlow. Το NSI άλλωστε

είναι ένα πλαίσιο που ενσωματώνει πολλές κρίσιμες λειτουργίες των σύγχρονων δικτύων που παραπέμπουν στα Ευφυή Προγραμματιζόμενα δίκτυα. Πιο συγκεκριμένα τα SDN είναι σχετικά με την διαχείριση του στρώματος δεδομένων και της προώθησης ενδιάμεσων συσκευών / τομέων και ανεξάρτητα από το τι αυτά τα πεδία μπορεί να περιέχουν. Το NSI-CS έχει σχεδιαστεί για να παρέχει απλούς διαφανείς αγωγούς για την παράδοση των δεδομένων από τη μία τοποθεσία στην άλλη. Οι δύο τεχνολογίες μπορούν πολύ εύκολα να συνδυαστούν:

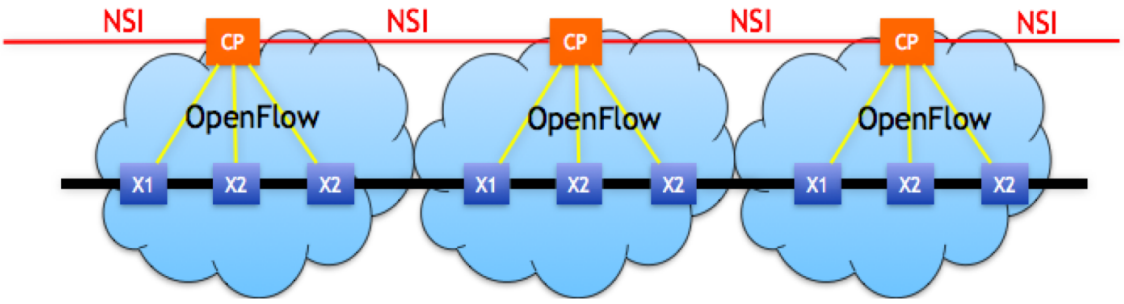
- Από τη σκοπιά του NSI: η λειτουργία NRM αντιστοιχεί σε έναν SDN ελεγκτή ή hypervisor διαχείριση των εσωτερικών μεταγωγέων και της προώθησης
- Από τη σκοπιά του SDN, ο SDN ελεγκτής / hypervisor μπορεί να χρησιμοποιήσει το πρωτόκολλο NSI για την απόκτηση και τη διαχείριση των πόρων των μεταγωγέων εκτός αλλά εντός ενός τομέα που διαχειρίζεται ένας ελεγκτής. Ένας τέτοιος Domain παραπέμπει σε ένα NSI τομέα παροχής υπηρεσιών δικτύου. Αντίστοιχα το NSI είναι σχετικό με τις υπηρεσίες μεταφοράς σε συνδέσμους που μεταφέρουν τα δεδομένα μεταξύ / κατά μήκος και εντός) του Τοπικού Διαχειριστικού Τομέα δηλαδή ο hypervisor μπορεί εύκολα να είναι ένας πράκτορας στον τομέα του. Έτσι καταλήγουμε στην αρχιτεκτονική του Σχήματος 3.7

  Αντίστοιχα παραδείγματα οριζόντιας αρχιτεκτονικής είναι και τα WE-bridge [16] (East-West bridge) και το SDNi το οποίο θα δούμε εκτενέστερα στην συνέχεια.

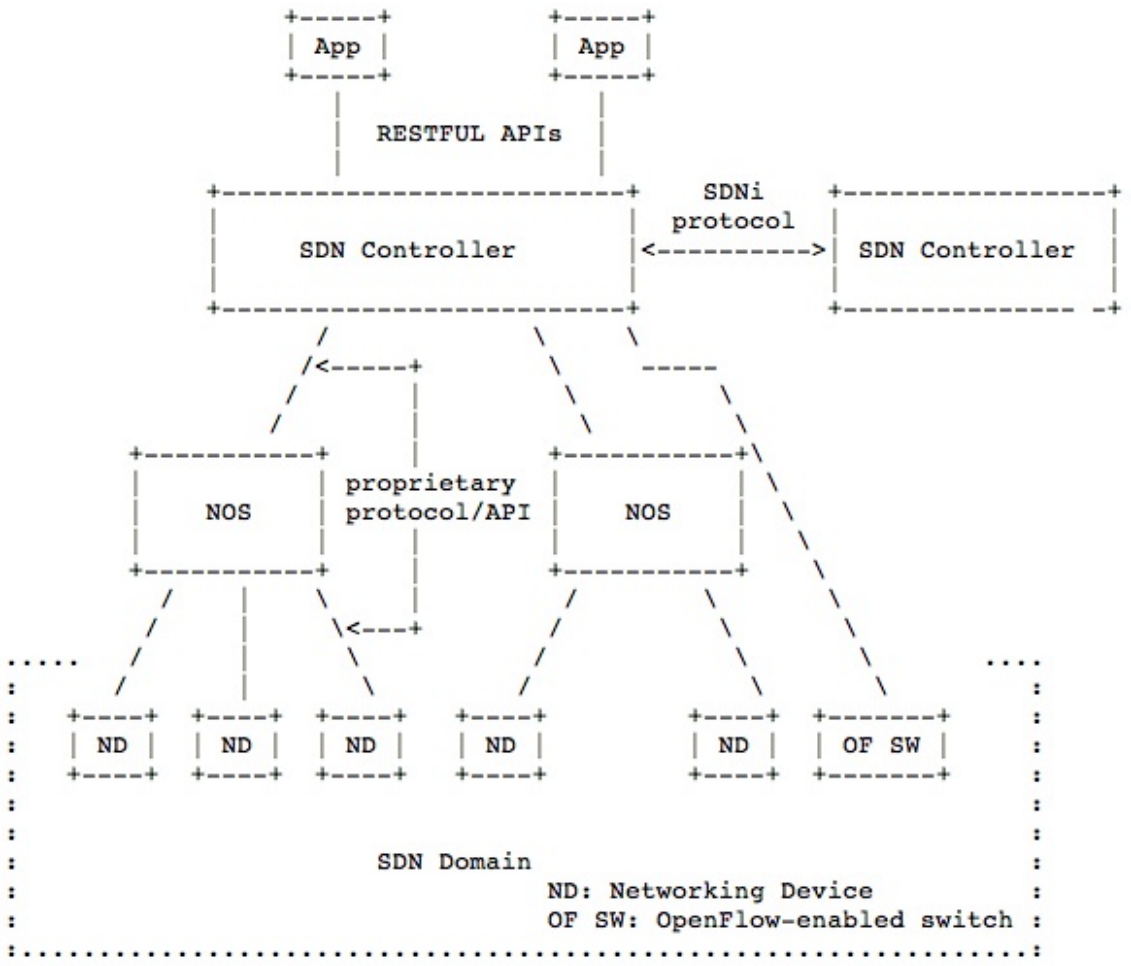
3.4.4 Software Defined Network interface (SDNi)

Το SDNi ή ελεγχόμενη από λογισμικό δικτυακή διεπαφή είναι μια τυπική υλοποίηση του οριζόντιου αρχιτεκτονικού μοντέλου. Περιγράφει την διεπαφή επικοινωνίας μεταξύ των ελεγκτών και τα μηνύματα που ανταλλάσσουν αφού πρώτα ορίσει την αρχιτεκτονική του δικτύου κάτω από τον ελεγκτή.

Πιο συγκεκριμένα, ως SDN domain ορίζεται ένα κομμάτι του δικτύου που καθορίζεται από έναν SDN ελεγκτή. Ο ελεγκτής μπορεί να επικοινωνεί είτε άμεσα με τα στοιχεία του δικτύου μέσω κάποιου πρωτοκόλλου όπως το OpenFlow αν αυτά το



Σχήμα 3.7: *Network Service Interface (NSI) σε περιβάλλον SDN, Πηγή: [13]*



Σχήμα 3.8: *Software Defined Network interface (SDNi), Πηγή:https://tools.ietf.org/html/draft-yin-sdn-sdni-00*

υποστηρίζουν είτε να τα διαχειρίζεται με την βοήθεια ενός δικτυακού λειτουργικού συστήματος. Ένας ελεγκτής μπορεί να επικοινωνεί με πλέον του ενός λειτουργικά και ο ίδιος μπορεί να τρέχει σε κάποιον απομακρυσμένο εξυπηρετητή είτε στο ιδιο το δικτυακό λειτουργικό. Σε κάθε περίπτωση ο ελεγκτής διατηρεί πλήρη εικόνα του δικτύου εγκαθιστά απ' άκρη σ΄άκρη επικοινωνία με τους γειτονικούς ελεγκτές και ανταλλάσσει πληροφορίες σχετικές με τις εφαρμογές και και τον έλεγχο του δικτύου. Πιο συγκεκριμένα, σύμφωνα με το πρωτόκολλο ανταλλάσσονται τριών ειδών μηνύματα :

6. Ενημερώσεις προσβασιμότητας : Η ανταλλαγή πληροφοριών προσβασιμότητας καθιστά εφικτή την δρομολόγηση. Αυτό επιτρέπει σε μία ροή πακέτων να περάσει από διαφορετικούς διαχειριστικούς τομείς και ο ελεγκτής σε κάθε έναν από αυτούς να επιλέγει το καλύτερο μονοπάτι.

7. Αίτηματα κατάστασης / κατάργησης και τροποποίησης κανόνων: Οι ελεγκτές συντονίζουν τις αιτήσεις οι οποίες περιέχουν πληροφορίες όπως τις ανάγκες του μονοπατιού, την ποιότητας υπηρεσιών κτλ μεταξύ διαφορετικών διαχειριστικών τομέων.

8. Ενημερώσεις δυνατοτήτων: Οι ελεγκτές ανταλλάσσουν πληροφορίες σχετικές με τις δυνατότητες του δικτύου όπως το εύρος ζώνης καθώς επίσης και τις δυνατότητες του λογισμικού που τρέχει μέσα στον διαχειριστικό τομέα.

Για την επικοινωνία των ελεγκτών χρειάζεται η εγκαθίδρυση μίας συνόδου με την χρήση είτε του BGP είτε του SIP πάνω απο TCP. Το SIP είναι ένα πρωτόκολλο τύπου ερωτήματος -απάντησης για να την προετοιμασία και την διαχείριση μίας επικοινωνίας Το πρωτόκολλο ορίζει τα μηνύματα που ανταλλάσσονται μεταξύ των συνομιλητών τα οποία διέπουν την εγκατάσταση τον τερματισμό και άλλα απαραίτητα στοιχεία μίας κλήσης. Το BGP από την άλλη διατηρεί επίσης σύνοδο μεταξύ των συνομιλητών ενώ παράλληλα ακολουθεί την προσέγγιση της μηχανής καταστάσεων. Το BGP χρησιμοποιείται για την ανταλλαγή πληροφοριών δρομολόγησης μεταξύ διαφορετικών διαχειριστικών τομέων συνεπώς η ανταλλαγή των SDNi μηνυμάτων συνάδει με την φύση του πρωτοκόλλου. Ταυτόχρονα τα είδη των μηνυμάτων που υποστηρίζει το BGP μπορούν να χρησιμοποιηθούν για την ανταλλαγή των sdni μηνυμάτων με ελάχιστες τροποποιήσεις που δεν διαταράσσουν την υπάρχουσα προτυποποιημένη λειτουργικότητα του πρωτοκόλλου. Για παράδειγμα το μήνυμα

OPEN περιέχει πεδίο προορισμένο για ανταλλαγή πληροφοριών σχετικών με τις δυνατότητες του δικτύου καλύπτει έτσι τις ανάγκες του τρίτου είδους sdni μηνύματος. Ταυτόχρονα το μήνυμα UPDATE χρησιμοποιείται για να κρατάει τους συνομιλητές ενημερωμένους όσο αναφορά την προσβασιμότητα, καλύπτει έτσι τις ανάγκες του πρώτου είδους sdni μηνύματος Τέλος το NLRI είναι ένα ειδικό πεδίο του μηνύματος UPDATE το οποίο αναφέρεται στα δυνατά μονοπάτια που μπορεί να ακολουθήσει μία ροή, καλύπτει έτσι τις ανάγκες του δεύτερου είδους sdni μηνύματος. Συνεπώς παρότι το BGP έχει κατηγορηθεί εντόνως ιδιαίτερα από τους υπερασπιστές των αρχιτεκτονικών του διαδικτύου του μέλλοντος, θεωρείτε καλύτερη επιλογή για την υλοποίηση του SDNi.

3.4.5 Επέκταση του SDNi

Όπως έγινε φανερό παραπάνω το πρωτόκολλο SDNi επικεντρώνεται αυτή την στιγμή στην ρύθμιση της δρομολόγησης μεταξύ των τομέων σε περιοχές που βρίσκονται υπό τον ίδιο φορέα δικτύου. Στην εργασία αυτή, προτείνεται η επέκτασή του σε ένα γενικό περιβάλλον που θα μπορούσε να αναπτυχθεί σταδιακά για να ικανοποιήσει την σηματοδότηση των γειτονικών SDN domains σε ολόκληρο το Διαδίκτυο ανεξάρτητα από τον σκοπό της επικοινωνίας. Γι αυτό χρειάζεται η ενσωμάτωση νέων μορφών μηνυμάτων στο πρωτόκολλο SDNi. Στα πλαίσια αυτής της εργασίας προτείνεται η χρήση του αρχείου IODEF, ενός τυποποιημένου μορφότυπου για να διαβιβάζονται αποδοτικότερα πληροφορίες ασφάλειας μεταξύ τομέων . Για την μεταφορά αυτού του αρχείου με την βοήθεια του BGP προτείνεται η χρήση ενός URI το οποίο μεταφέρεται στο NLRI πεδίο του BGP-UPDATE. Με την βοήθεια της διαδικτυακής υπηρεσίας του rest ένα ερώτημα στο συγκεκριμένο URI από οπουδήποτε θα επιστρέφει το IODEF αρχείο.

Κεφάλαιο 4

Κατανεμημένη Επίθεση άρνησης παροχής υπηρεσίας

4.1 Αντιμετώπιση δικτυακών επιθέσεων με SDN

Σήμερα την προστασία ενός δικτύου αναλαμβάνουν ως επί το πλείστον ενδιάμεσα μηχανήματα (middleboxes) τα οποία είναι κατασκευασμένα για να υλοποιούν συγκεκριμένες λειτουργίες ελέγχου στην διερχόμενη κίνηση. Σταδιακά όμως αναπτύσσονται όλο και περισσότερες εφαρμογές οι οποίες χρησιμοποιούν τις δυνατότητες των Ευφυών Προγραμματιζόμενων Δικτύων για να αντικαταστήσουν αυτά τα μηχανήματα. Στην συνέχεια αναφέρονται κάποια τέτοια παραδείγματα:

Στο [17], οι συγγραφείς προτείνουν το σύστημα ”Resonance” που παρέχει δυναμικές πολιτικές ελέγχου πρόσβασης. Η λύση τους είναι βασισμένη στο OpenFlow για την πραγματοποίηση ελέγχου πρόσβασης στα στοιχεία του δικτύου. Λόγω της ευελιξίας του μπορούν να επιτύχουν άμεση ανταπόκριση και δυναμικό έλεγχο της πρόσβασης στο δίκτυο χωρίς τη χρήση ειδικών middleboxes. Στην δημοσίευση τους τονίζουν, επίσης, τα μειονεκτήματα και τους περιορισμούς των σημερινών αρχιτεκτονικών ελέγχου πρόσβασης που στηρίζονται στα VLAN και δείχνουν πώς τα Ευφυή Προγραμματιζόμενα Δίκτυα συντελούν στην αντιμετώπιση αυτών των περιορισμών και των προβλημάτων.

Ένα παρόμοιο θέμα είναι η εφαρμογή των πολιτικών σε όλο το δίκτυο, το οποίο αναλύ-

εται στο[18]. Οι συγγραφείς προτείνουν την αρχιτεκτονική VeriFlow, η οποία μπορεί να χρησιμοποιηθεί για να ελέγξει και να επιβάλει τις πολιτικές του δικτύου και των σταθερών σε πραγματικό χρόνο. Αυτές οι σταθερές μπορούν να περιλαμβάνουν ελέγχους για βρόχους προώθησης και παραβιάσεις στον ελέγχου πρόσβασης καθώς και εφαρμογή ήμι βέλτιστης δρομολόγησης.

Ο Yao et al. [19] παρουσιάζει μία μέθοδο για την επικύρωση της διεύθυνσης με την βοήθεια των Ευφυών Προγραμματιζόμενων Δικτύων έτσι ώστε η χρήση ψεύτικης διεύθυνσης (spoofing) να μπορεί να αποτραπεί αποτελεσματικά. Η προσέγγιση βασίζεται στην επιλεκτική εγκατάσταση κανόνων που είτε προωθούν είτε διαγράφουν πακέτα. Εάν φτάσει ένα πακέτο που δεν ταιριάζει με κανένα κανόνα εισόδου, ερωτάται ο ελεγκτής . Στη συνέχεια, ο ελεγκτής ελέγχει, αν η συγκεκριμένη διεύθυνση έχει το δικαίωμα να στείλει πακέτα. Όταν ανιχνευτεί πλαστή διεύθυνση, εγκαθίσταται κατάλληλος κανόνας που διαγράφει τα πακέτα που προέρχονται από αυτό το IP για να αποτραπούν περαιτέρω ενέργειες. Η μετακίνηση του στόχου της άμυνας αποτρέπει επιθέσεις που βασίζονται σε στατική διεύθυνση IP, όπως η κρυφή σάρωση και διάδοση κακόβουλου λογισμικού τύπου worm.

Ένα ακόμη παράδειγμα είναι το [20], στο οποίο παρουσιάζεται ένα σύστημα που χρησιμοποιεί την προγραμματισιμότητα των Ευφυών Δικτύων για να ”μετακινεί” τους στόχους των επιθέσεων, αλλάζοντας συχνά τις διευθύνσεις IP κάποιων εξυπηρετητών. Με τον τρόπο αυτό μια υπηρεσία φιλοξενείται πάντα σε ένα εξυπηρετητή με στατική IP όμως στον έξω κόσμο αυτή φαίνεται να αλλάζει, αποτρέποντας αποτελεσματικά τη σάρωση του εξυπηρετητή τόσο από εξωτερικές, όσο και από εσωτερικές επιθέσεις.

Τέλος η ανίχνευση και η αντιμετώπιση της Κατανεμημένης Επίθεση άρνησης παροχής υπηρεσίας έχει διερευνηθεί αρκετές φορές με βάση τα Ευφυή Προγραμματιζόμενα Δίκτυα. Τα [21], [22] παρουσιάζουν μεθόδους ανίχνευσης επιθέσεων DDoS μετρώντας τη συχνότητα και την εντροπία της κίνησης κάθε ροής αντίστοιχα. Εάν τα μεγέθη αυτά της ροής υπερβαίνουν ένα συγκεκριμένο όριο, υποθέτουν ότι συμβαίνει μια επίθεση . Αντίστοιχες προσπάθειες περιγράφονται στα [] , [23] ενώ στο [24] μεταφέρει κάποιες από τις γνωστές μεθόδους ανίχνευσης σε περιβάλλον SDN. Σε μια τέτοια περίπτωση, ο ελεγκτής δίνει εντολή στους μεταγωγείς να διαγράψουν τα πακέτα που ανήκουν στην κακόβουλη ροή για την άμβλυνση των συνεπειών της επίθεσης. Ένας άλλος τρόπος που προτείνε-

ται από τον Braga et al.[25] βασίζεται στην αυτό-οργανούμενο χάρτη για την ταξινόμηση των κίνησης σε ακολουθίες. Όταν παρατηρείται κίνηση η οποία έχει ταξινομηθεί ως κακόβουλη, ο ελεγκτής άμεσα προγραμματίζει το δίκτυο ώστε να προωθεί πακέτα της.

4.2 Ορισμός - Γενικά Στοιχεία

Η επίθεση εκδηλωμένη ως άρνηση (παροχής) υπηρεσίας είναι μια μεγάλης κλίμακας, συντονισμένη επίθεση στη διαθεσιμότητα των υπηρεσιών ενός συστήματος ή στους πόρους ενός δικτύου, που ξεκινάει έμμεσα μέσω εκμετάλλευσης πολλών εκτεθειμένων υπολογιστών στο διαδίκτυο. Η επίθεση αυτή μπορεί να ακολουθήσει δύο τύπους αρχιτεκτονικών: την αρχιτεκτονική Πράκτορα - Χειριστή και την αρχιτεκτονική διαδικτυακής αναμετάδοσης κίνησης.

Η αρχιτεκτονική Πράκτορα - Χειριστή αποτελείται από πελάτες, πράκτορες και χειριστές.Οι χειριστές είναι πακέτα λογισμικού που βρίσκονται σε όλο το διαδίκτυο και χρησιμοποιούνται από τους πελάτες για να επικοινωνήσουν με τους πράκτορες. Στιγμιότυπα του λογισμικού των πρακτόρων τοποθετούνται σε εκτεθειμένα μηχανήματα που τελικά πραγματοποιούν την επίθεση. Η επιλογή τους προφανώς δεν είναι τυχαία. Επιλέγονται μηχανήματα που μπορούν να παραβιαστούν και έχουν την δυνατότητα να στείλουν μεγάλο όγκο κίνησης στο θύμα. Οι ιδιοκτήτες και οι χρήστες των μηχανημάτων αγνοούν την κατάσταση καθώς ο κακόβουλος κώδικας που εγκαθίσταται δεν επηρεάζει σημαντικά την απόδοση του εκάστοτε μηχανήματος.

Στην αρχιτεκτονική διαδικτυακής αναμετάδοσης κίνησης (IRC), ένα κανάλι επικοινωνίας χρησιμοποιείται για τη σύνδεση του πελάτη με τους πράκτορες. Οι θύρες IRC μπορούν να χρησιμοποιηθούν για την αποστολή εντολών στους πράκτορες. Αυτό κάνει τα πακέτα της επίθεσης πιο δύσκολο να εντοπιστούν. Επιπλέον, είναι πιο εύκολο για έναν εισβολέα να κρύψει την παρουσία του σε ένα τέτοιο κανάλι, καθώς αυτά τείνουν να έχουν μεγάλο όγκο κίνησης. Μια IRC επίθεση ακολουθεί μοντέλο παρόμοιο με το μοντέλο επίθεση Πράκτορα -Χειριστή με την διαφορά ότι αντί να χρησιμοποιεί ένα πρόγραμμα χειρισμού εγκατεστημένο σε ένα εξυπηρετητή ιστού, χρησιμοποιεί έναν IRC διακομιστή ο οποίος παρακολουθεί τις διευθύνσεις των συνδεδεμένων πρακτόρων και χειριστών και διευκο-

λύνει την επικοινωνία μεταξύ τους.

Οι επιθέσεις αυτές είναι δύσκολο να αντιμετωπιστούν γιατί στηρίζονται στις αδυναμίες της λειτουργίας του σύγχρονου διαδικτύου. Πίο συγκεκριμένα το πόσο εκτεθειμένο σε τέτοιου είδους επιθέσεις είναι ένα δίκτυο δεν εξαρτάται τόσο από το ίδιο αλλά κυρίως από το υπόλοιπο διαδίκτυο. Οι επιθέσεις αυτές στηρίζονται επίσης στην υπεροχή του επιτιθέμενου σε όγκο πόρων σε σχέση με τους πόρους του θύματος. Η υπεροχή αυτή είναι δεδομένη καθώς οι πόροι του επιτιθέμενου συντίθενται από μη φραγμένο σύνολο εκτεθειμένων υπολογιστών σε όλο το διαδίκτυο.

4.3 Κατηγορίες Μηχανισμών Άμυνας

Με βάση το σημείο στο οποίο πραγματοποιείται η ανάπτυξη του συστήματα υπεράσπισης εναντίον των επιθέσεων η μέθοδοι άμυνας μπορούν να χωριστούν σε τρεις κατηγορίες: άμυνας στην περιοχή του θύματος, άμυνας στην περιοχή της πηγής, και άμυνας στο ενδιάμεσο δίκτυο. Κάθε προσέγγιση έχει τα δικά της πλεονεκτήματα και μειονεκτήματα τα οποία θα συζητήσουμε.

4.3.1 Μηχανισμός άμυνας στην περιοχή του θύματος

Οι σχετικές προσεγγίσεις ανίχνευσης και αντιμετώπισης της επίθεσης υλοποιούνται εσωτερικά ή/και στα άκρα του διαχειριστικού τομέα του θύματος, δηλαδή, του δικτύου που παρέχει κρίσιμες δικτυακές υπηρεσίες. Στην περίπτωση αυτή υλοποιείται εξειδικευμένη μονάδα που επωμίζεται την ευθύνη του εντοπισμού της επίθεσης. Κρατούνται επίσης συνήθως δεδομένα αναφοράς δηλαδή πληροφορίες σχετικές με γνωστά είδη επιθέσεων και το προφίλ της φυσιολογικής λειτουργίας του τομέα. Αυτά τα δεδομένα ανανεώνονται καθώς η γνώση επαυξάνεται με νέες παρατηρήσεις. Η ανίχνευση και αντιμετώπιση επιθέσεων στα άκρα του διαχειριστικού τομέα του θύματος είναι σχετικά εύκολη λόγω του υψηλού ποσοστού κατανάλωσης πόρων. Είναι επίσης η πιο πρακτικά εφαρμόσιμη καθώς οι εξυπηρετητές ιστού που παρέχουν κρίσιμες υπηρεσίες προσπαθούν πάντα να εξασφαλίσουν τους πόρους τους για τους καλούς χρήστες. Ωστόσο η αποτελεσματική αντιμετώπιση της επίθεσης είναι συχνά αδύνατη με αυτόν τον τρόπο. Όσο αφορά την αντιμετώ-

πιση,αυτό οφείλεται στο ότι κατά τη διάρκεια της επίθεσης, οι πόροι του θύματος, π.χ., το εύρος ζώνης ή ο διαθέσιμος χώρος για κανόνες , συχνά κατακλύζονται από τον όγκο της εισερχόμενης κίνησης. Ο τομέας έτσι αναγκάζεται να εγκαταστήσει πιο γενικούς κανόνες και να κόψει συνεπώς πολύ μεγάλο μέρος της κίνησης του για να εξυπηρετήσει έστω και κάποιους από τους πελάτες του. Στο μεταξύ η κίνηση της επίθεσης εξακολουθεί να πλημμυρίζει τους γύρω διαχειριστικούς τομείς. Ένα ακόμα σημαντικό μειονέκτημα όσο αφορά την ανίχνευση, αυτή τη φορά, στη μεριά του θύματος είναι ότι, ανιχνεύεται η επίθεση μόνο μετά την άφιξή της στο θύμα και η αντιμετώπιση της καθυστερεί. Αυτό σημαίνει ότι μέχρι τότε οι καλοί πελάτες στερούνται την υπηρεσία. Παρά τα μειονεκτήματα της είναι πολύ σημαντικό να τονιστεί ότι ελαχιστοποιεί τον αριθμό των λάθος συναγερμών.

4.3.2 Μηχανισμός άμυνας στην περιοχή της πηγής

Στόχος του μηχανισμού αυτού είναι η ανίχνευση και αντιμετώπιση της επίθεσης στο σημείο των πηγών πριν καν η κίνηση δημιουργήσει πρόβλημα στο θύμα. Σε αυτήν την περίπτωση ένα συστατικό κομμάτι λογισμικού σε κάθε διαχειριστικό τομέα επωμίζεται την ευθύνη της ανίχνευσης επιθέσεων που ξεκινούν από τον ίδιο τον τομέα. Πολλοί τρόποι έχουν προταθεί για την υλοποίηση αυτής της ανίχνευσης. Ένας τέτοιος τρόπος είναι η επιβολή ορίου στις εξερχόμενες συνδέσεις. Γι αυτό συγκρίνονται τα στατιστικά της εισερχόμενης και εξερχόμενης κίνησης με τις τιμές της προκαθορισμένη κανονικής συμπεριφοράς. Άλλος τρόπος υλοποίησης είναι ο συνδυασμός των εσωτερικών του τομέα παρατηρήσεων με ειδοποιήσεις που προέρχονται από το θύμα της επίθεσης. Ο εντοπισμό και η αντιμετώπιση μιας επίθεσης στην πηγή της είναι σαφώς η καλύτερη δυνατή άμυνα. Εμποδίζει τις πλημμύρες και την άσκοπη κατανάλωση πόρων όχι μόνο στην πλευρά του θύματος, αλλά και σε ολόκληρο το ενδιάμεσο δίκτυο. Ταυτόχρονα το θύμα βρίσκεται για ελάχιστο χρόνο στην δυσμενή κατάσταση της αποσύνδεσης από τον διαδίκτυο ή της αδυναμίας να εξυπηρετήσει τους καλούς του πελάτες. Η βασική δυσκολία αυτής της αντιμετώπισης όμως είναι η ανίχνευση της επίθεσης ή των πηγών αυτής από το θύμα με ακρίβεια. Αυτό συμβαίνει διότι οι πηγές είναι ευρέως κατανεμημένες και κάθε μία συμπεριφέρεται σχετικά φυσιολογικά. Ένα ακόμη πρόβλημα είναι η δυσκολία υλοποίησης συγκεκριμένης πολιτικής σε τόσες διαφορετικές τοποθεσίες και διαχειριστικές αρχές.

4.3.3 Μηχανισμός άμυνας στο ενδιάμεσο δίκτυο

Ο Μηχανισμός άμυνας στο ενδιάμεσο δίκτυο εξισορροπεί το trade-off μεταξύ της ακρίβειας ανίχνευσης και της κατανάλωσης εύρους ζώνης που είναι τα βασικότερα προβλήματα των προηγούμενων προσεγγίσεων. Ένα τέτοιο σύστημα έχει γενικά συλλογικό χαρακτήρα και οι διαχειριστικοί τομείς μοιράζονται τις παρατηρήσεις τους μεταξύ τους. Όπως στον μηχανισμό άμυνας στην πηγή, τα συστήματα αυτά μπορούν επιβάλλουν όρια ποσοστού για τις συνδέσεις που διέρχονται από τους τομείς που διαχειρίζονται μετά από σύγκριση με τα αποθηκευμένα ποσοστά του συνηθισμένου προφίλ καθώς και να δέχονται υποδείξεις από άλλους τομείς. Η ανίχνευση της επίθεσης και εύρεση των πηγών είναι εύκολη σε αυτήν την προσέγγιση καθώς στηρίζεται στην συλλογική λειτουργία. Οι διαφορετικοί τομείς μπορούν να σχηματίσουν ένα πλέγμα επικάλυψης και να μοιραστούν τις παρατηρήσεις τους. Όμοια η αντιμετώπιση της επίθεσης είναι η βέλτιστη δυνατή καθώς η εγκατάσταση κανόνων είναι βέλτιστα κατανεμημένη. Προφανώς, η εφαρμογή της εν λόγω προσέγγισης στην πράξη είναι εξαιρετικά δύσκολη καθώς προϋποθέτει υψηλού επιπέδου συνεργασία και εμπιστοσύνη μεταξύ των τομέων.

4.4 Παραδείγματα αρχιτεκτονικών αντιμετώπισης επιθέσεων

Η παρούσα εργασία εστιάζει στην βελτίωση της αντιμετώπισης της επίθεσης και όχι της ανίχνευσης. Είναι εμφανές ότι η αντιμετώπιση μίας επίθεσης είναι πολύ πιο αποτελεσματική και επιτυχημένη όταν είναι συνεργατική. Όταν πολλοί διαχειριστικοί τομείς συνεργάζονται αφιερώνοντας πόρους για την αντιμετώπιση μίας επίθεσης, αυξάνεται το σύνολο των διαθέσιμων πόρων με αποτέλεσμα (α) να μην είναι απαραίτητη η επιβολή γενικότερων φίλτρων που κόβουν μαζί με την κακή και καλή κίνηση καθώς και (β)να αίρεται το βασικότερο στοιχείο που κάνει τις DDos επιθέσεις πετυχημένες που είναι η υπεροχή σε πόρους. Ταυτόχρονα με την συνεργατική αντιμετώπιση το σημείο άμυνας μεταφέρεται πιο κοντά στις πηγές με αποτέλεσμα να μην μολύνεται το δίκτυο με κακόβουλη κίνηση.

4.4.1 Active Internet Traffic Filtering (AITF)

Μία από τις πιο ολοκληρωμένες προτάσεις συνεργατικών συστημάτων άμυνας είναι το AITF [26]. Στο έργο αυτό ο εξυπηρετητής ή υπηρεσία που βρίσκεται κάτω από επίθεση ειδοποιεί τον συνοριακό εξυπηρετητή πύλης του, ο οποίος εγκαθιστά προσωρινά κανόνες και στη συνέχεια ειδοποιεί τους συνοριακούς εξυπηρετητές πύλης των πηγών της επίθεσης. Οι συνοριακοί εξυπηρετητές πύλης ειδοποιούν τους χρήστες που πραγματοποιούν την επίθεση να σταματήσουν να στέλνουν κίνηση στο θύμα. Για να είναι εφικτή ωστόσο η επικοινωνία μεταξύ των συνοριακών εξυπηρετητών πηγής και προορισμού, ο συνοριακός εξυπηρετητής του θύματος (του προορισμού) πρέπει να γνωρίζει ποιος είναι ο υπεύθυνος αντίστοιχος εξυπηρετητής της κάθε πηγής της επίθεσης. Αυτό εξασφαλίζεται υπερφορτώνοντας κάποια από τα πακέτα όλης της κίνησης με τις ταυτότητες των συνοριακών δρομολογίων από τους οποίους πέρασαν μέχρι να φτάσουν στον προορισμό τους. Οι τιμές της χρονικής περιόδου που κρατούνται προσωρινά και μακροπρόθεσμα τα φίλτρα είναι προκαθορισμένες. Επιπλέον, στην προσέγγιση αυτή ο συνοριακός εξυπηρετητής του θύματος ζητάει βοήθεια από αυτόν της κάθε πηγής απευθείας. Ωστόσο, ένα ΑΣ έχει ελάχιστον ή ανύπαρκτο κίνητρο για να εμποδίσει την κυκλοφορία κάποιου χρήστη για λογαριασμό ενός άλλου ΑΣ το οποίο δεν γνωρίζει ούτε κατ ανάγκη εμπιστεύεται Τέλος, Στην περίπτωση που ένα ΑΣ δεν συνεργαστεί, αυτό μπορεί να τιμωρηθεί με αποσύνδεση από το θύμα και το οποίο στην συνέχεια ζητάει βοήθεια κάποιου ΑΣ στο μονοπάτι, πολιτική που στην πράξη δεν είναι αποδεκτή.

4.4.2 Το πρωτόκολλο BGP FlowSpec και η εφαρμογή του

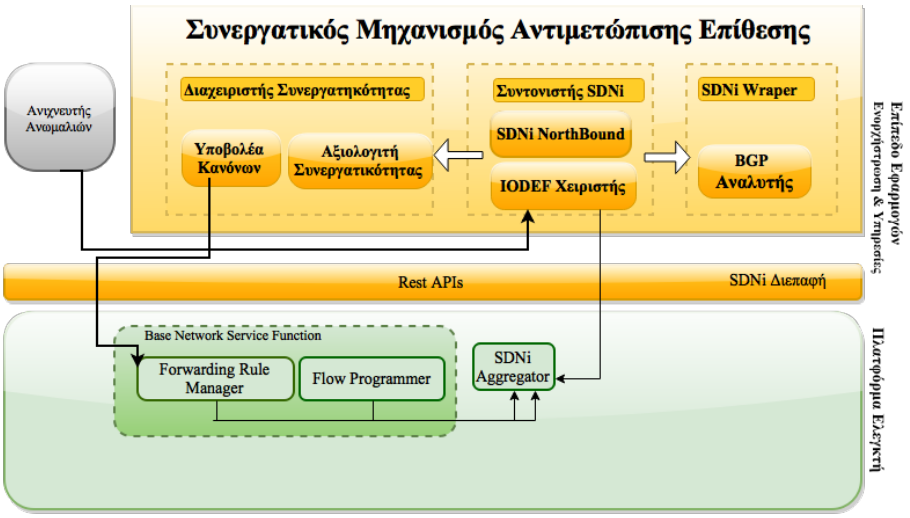
Το BGP FlowSpec [27] είναι μία επέκταση του BGP που υποστηρίζεται από συγκεκριμένους κατασκευαστές υλικού και επιτρέπει στο BGP να διαδίδει φίλτρα που αποτελούνται από μία πλειάδα που αφορά τα κριτήρια ταιριάσματος του φίλτρου με μία ροή πακέτων και από κάποιες ενέργειες για να εφαρμοστούν στις ροές αυτές. Τα βασικότερα κριτήρια ταιριάσματος είναι το πρόθεμα πηγής / προορισμού, η πόρτα πηγής / προορισμού, ο τύπος ή/και κωδικός ICMP , το μέγεθος του πακέτου, το DSCP, η σημαία TCP και η κωδικοποίηση θραύσματος. Οι βασικότερες ενέργειες που τα φίλτρα προβλέπουν είναι η αποδοχή,

η απόρριψη, το ποσοστό-όριο δειγμάτων και η προώθηση. Οι πληροφορίες και τα φίλτρα που ανταλλάσσονται είναι ανεξάρτητα από την δρομολόγηση καθώς υλοποιούνται σε διαφορετικά NLRI, αλλά αυτόματα επαληθεύονται σε σχέση με αυτήν, πρόκειται δηλαδή για μια προσθήκη στους υπάρχοντες μηχανισμούς. Τα θέματα πολυπλοκότητας και κλιμάκωσης έχουν ήδη λυθεί λόγω της αποδεδειγμένης επεκτασιμότητας και ευελιξίας του BGP στην προσθήκη νέων υπηρεσιών (Multicast, IPv6, L3 VPN, L2 VPN, VPLS). Το BGP FlowSpec επαναχρησιμοποιεί την υποδομή εξωτερικής δρομολόγησης και το υφιστάμενο μοντέλο εμπιστοσύνης. ’Ετσι ένας δρομολογητής αποδέχεται φίλτρα όταν διαφημίζονται από το επόμενο-βήμα του προθέματος προορισμού.

Το πρωτόκολλο αυτό χρησιμοποιείται για την συνεργατική αντιμετώπιση των πελατών του GEANT μέσα από ένα σύστημα που ονομάζεται FireCycle. firecircle. Πίο αναλυτικά το σύστημα λειτουργεί ως εξής: Κάποιος εκπρόσωπος Πελάτη του NOC μπαίνει σε μια ιστοσελίδα και περιγράφει τις ροές των επιτιθεμένων καθώς και ενέργειες που επιθυμεί να εφαρμοστούν στις ροές αυτές. Η διεύθυνση IP προορισμός αυτών των ροών πρέπει να ταυτίζεται με το IP του πελάτη για λόγους ασφαλείας. Ένας ειδικός δρομολογητής έχει ρυθμιστεί να διαφημίζει τη διαδρομή μέσω BGP flowspec. Συνόδοι eBGP διαδίδουν τις n-άδες στους δρομολογητή (ες) του ΕΔΕΤ ενώ ταυτόχρονα iBGP σύνοδοι τις διαδίδουν περαιτέρω στο εσωτερικό των ΑΣ. Παρά τα πλεονεκτήματα αυτού του συστήματος δεν πρέπει να ξεχνάμε ότι στηρίζεται σε μία λύση που μόνο ένας κατασκευαστής υλικού υποστηρίζει.Ταυτόχρονα το σύστημα δεν επιλύει τα θέματα της εμπιστοσύνης μεταξύ των συμμετεχόντων περιορίζοντας έτσι εξ αρχής την δυνατότητα υλοποίησης εντός των πλαισίων ενός και μόνο διοικητικού τομέα.

4.5 Συνεργατικός Μηχανισμός Αντιμετώπισης Επίθεσης

Για την αντιμετώπιση των προβλημάτων προηγούμενων μεθόδων προτείνεται ένα συνεργατικό σχήμα αντιμετώπισης DDos που συνδυάζει τις δυνατότητες των SDN με αλγορίθμους εμπιστοσύνης. Ο Συνεργατικός Μηχανισμός Αντιμετώπισης Επίθεσης (ΣΜΑΕ) η αρχιτεκτονική του οποίου, φαίνεται στο Σχήμα 4.1 τρέχει πάνω από τον ελεγκτή σε κάθε SDN τομέα. Το σύστημα εξασφαλίζει την αντιμετώπιση μίας επίθεσης μέσω μετάδοσης μηνυμάτων-εκκλήσεων για βοήθεια από τον τομέα θύμα προς τους τομείς που βρίσκο-



Σχήμα 4.1: *Αρχιτεκτονική Συνεργατικού Μηχανισμού Αντιμετώπισης Επίθεσης*

νται στα μονοπάτια της κίνησης της επίθεσης. Κάθε ένας από αυτούς τους τομείς καλείται να βοηθήσει στην αντιμετώπιση εγκαθιστώντας κανόνες που διαγράφουν τα πακέτα των επιτιθέμενων και να διαδώσει περαιτέρω την έκκληση στους τομείς από τους οποίους τα πακέτα έφτασαν σε αυτόν. Για την υλοποίηση του συστήματος χρησιμοποιήθηκα το sdni με την επέκταση του IODEF όπως αυτό περιγράφηκε στα προηγούμενα κεφάλαια καθώς και ένας αλγόριθμος για την αυτοματοποίηση της απόφασης ο οποίος θα μελετηθεί στο επόμενο κεφάλαιο.

Ο Συνεργατικό Μηχανισμό Αντιμετώπισης Επίθεσης (ΣΜΑΕ) αποτελείται από τρία συστατικά (στοιχεία) συστημικού λογισμικού.Στην συνέχεια θα εξεταστούν οι λεπτομέρειες της υλοποίησης.

1. Ο Διαχειριστής Συνεργατικότητας συγκεντρώνει την ευφυΐα του Τομέα, καθώς καθορίζει και εφαρμόζει τη στρατηγική που θα ακολουθηθεί ανάλογα με τη ληφθείσα κλήση για βοήθεια από έναν άλλο τομέα. Αποτελείται από δύο κλάσεις τον Αξιολογιτή Συνεργατικότητας και τον Υποβολέα Κανόνων. Ο πρώτος παρακολουθεί την συμπεριφορά των γειτονικών και μη Τομέων ενώ ο δεύτερος υλοποιεί την απόφαση για βοήθεια εγκαθιστώντας κανόνες στο επίπεδο δεδομένων.

2. Ο Συντονιστής SDNi είναι υπεύθυνος για το χειρισμό των sdni- μηνυμάτων. Περιέχει το IODEF Χειριστή που είναι μια συνάρτηση υπεύθυνη για την χειρισμό των μηνυμάτων που παραπέμπουν σε αρχείο της μορφή αυτής IODEF. Ο IODEF

Χειριστής έχει τρεις βασικές λειτουργίες: (α) τη δημιουργία ενός αρχείου για να περιγραφεί επαρκώς η επίθεση, μαζί με ένα αίτημα για βοήθεια το οποίο θα αποσταλεί σε άλλους τομείς (β) την αποκωδικοποίηση του αρχείου - μηνύματος και τη μετάδοση των κατάλληλων πληροφοριών στο Διαχειριστής Συνεργατικότητας. (γ) την περαιτέρω διάδοση των ληφθέντων μηνυμάτων στους κατάλληλους γειτονικούς τομείς.

3. Ο SDNi Wrapper είναι υπεύθυνο για τη αποστολή και συλλογή των sdni μηνύματα από τους ελεγκτές πάνω από το πρωτόκολλο BGP. Περιέχει τον BGP Αναλυτής που αναλύει κάθε μήνυμα BGP, το αποκωδικοποιεί, εξάγει τις χρήσιμες πληροφορίες και το αποδίδει στον κατάλληλο για την περαιτέρω επεξεργασίας του μηχανισμό. Το BGP Αναλυτής εσωκλείνει επίσης τις πληροφορίες που πρέπει να ανταλλαχθούν για τις ανάγκες της εφαρμογής σε ένα μήνυμα κατάλληλο για να μεταδοθεί μέσω BGP.

Για να γίνει πιο κατανοητή η λειτουργία του συστήματος και οι αλληλεπιδράσεις μεταξύ των συστατικών στοιχείων του λογισμικού θα περιγραφούν τρεις διαδικασίες και πιο συγκεκριμένα η άφιξη, η αποστολή και η σύνταξη ενός μηνύματος.

4.5.1 Δημιουργία - Τροποποίηση του IODEF

’Οταν ο μηχανισμός Ανίχνευσης Ανωμαλιών ενός τομέα αναγνωρίσει μία επίθεση ειδοποιεί τον Χειριστή IODEF του τομέα ο οποίος αναλαμβάνει να συντάξει το IODEF. Για την σύνταξη του IODEF ο Χειριστής λαμβάνει από τον μηχανισμό Ανίχνευσης το είδος της επίθεσης, την διεύθυνση IP του θύματος και τις IP διευθύνσεις των μηχανημάτων που αποτελούν τις πηγές επίθεσης. Η λειτουργία του χειριστή είναι απαραίτητη καθώς κάθε διαφορετικός μηχανισμός Ανίχνευσης Ανωμαλιών που υλοποιείται είτε στο υλικό είτε στο λογισμικό πιθανώς να έχει διαφορετική έξοδο. Ο Χειριστής αναλαμβάνει να μετατρέψει όλες τις πιθανές αυτές εξόδους σε ένα αρχείο που καταλαβαίνει κάθε στιγμιότυπο της εφαρμογή λογισμικού ΣΜΑΕ. Παράγεται έτσι ένα ειδικό αρχείο για κάθε τομέα παραλήπτη. Κάθε μοναδικό τέτοιο αρχείο είναι συνδεδεμένο με ένα κλειδί το οποίο μεταφέρεται μέσω του BGP UPDATE στους παραλήπτες. Μετά την σύνταξη του, αποθηκεύεται τοπικά

και κάθε ένα από αυτά συνδέεται με μοναδικό κλειδί . Η συνάρτηση sdni Northbound του sdni Συντονιστή αντλεί από αυτό το σημείο αποθήκευσης το αρχείο και το προσφέρει σαν απάντηση στα αιτήματα κάθε τομέα παραλήπτη που συνοδεύονται από αυτό το μοναδικό κλειδί.

Μεταξύ αυτών των αρχείων το πεδίο που διαφοροποιείται είναι ο κατάλογος των IP ο οποίος περιέχει μόνο ένα υποσύνολο του αρχικού συνόλου των κακόβουλων IP διευθύνσεις τα οποία γνωστοποιήθηκαν στον IODEF Χειριστή. Το υποσύνολο αυτό αποτελείται από τις διευθύνσεις πηγής αυτές που αντιστοιχούν σε ροές πακέτων οι οποίες δρομολογούνται μέσω του τομέα για τον οποίο φτιάχνεται το IODEF. Για τον διαχωρισμό αυτό ο Χειριστής συνεργάζεται με τον SDNi Wrapper ο οποίος ζητά τις καταχωρήσεις του πίνακα ροής που αφορούν όλες τις κακόβουλες ροές πακέτων που γνωρίζει. Στην συνέχεια ο Χειριστής ελέγχει την διεύθυνση MAC της πηγής κάθε κακόβουλης ροής, διαπιστώνει από ποιον τομέα αυτή προήλθε και την προσθέτει στο IODEF αρχείο αυτού του τομέα. Στην συνέχεια ο IODEF Χειριστής επικοινωνεί με το SDNi wrapper για να στείλει τα μηνύματα. Η διαδικασία της προώθησης παρουσιάζεται αναλυτικότερα παρακάτω.

Η διαδικασία τροποποίησης ενός ληφθέντος IODEF έχει ελάχιστες διαφορές από την διαδικασία δημιουργίας νέου. Στην περίπτωση αυτή η διαδικασία πυροδοτείται από τον Διαχειριστή Συνεργατικότητας. Η γνώση σχετικά με την επίθεση παρέχεται κατευθείαν από το IODEF αρχείο αντί να δίνεται από τον μηχανισμό Ανίχνευσης Ανωμαλιών. Επιπλέον ο Χειριστής μέσα σε κάθε νέο IODEF εσωκλείει την γνώμη του για το επίπεδο συνεργατικότητας του τομέα που του έστειλε το IODEF.

4.5.2 Αποστολή μηνύματος

Για την αποστολή ενός μηνύματος ο BGP Αναλυτής εκκινεί μια σύνοδο BGP με ένα άλλο ελεγκτή χρησιμοποιώντας ένα OPEN μήνυμα στο οποίο εδραιώνεται η δυνατότητα χρήσης Content-URI [28] μεταξύ των συνδιαλεγόμενων ελεγκτών. Το Content-URI είναι ένα τυποποιημένο αναγνωριστικό πόρου που παραπέμπει σε κάποιο περιεχόμενο. Μετά την ανταλλαγή του OPEN, οι ελεγκτές έχουν την δυνατότητα να ανταλλάσσουν BGP-UPDATE [29] μηνύματα που περιέχουν την οικογένεια διευθύνσεων (AF) τύπου Content-URI η οποία περιέχει την παραπομπή στην πηγή που είναι το IODEF αρχείο, μέσα στο

NLRI [30] πεδίο [30]. Γενικά, το URI αποτελεί ένα σύστημα το οποίο περιγράφει το μηχανισμό για να ανακτηθεί ένα περιεχόμενο, ο μηχανισμός αυτός στην περίπτωσή μας είναι το rest. Το URI περιγράφει επίσης την θέση όπου το περιεχόμενο ή αλλιώς ο πόρος που στην περίπτωση μας αντιστοιχεί στο αρχείο IODEF βρίσκεται. Για την ικανοποίηση των αιτημάτων των παραληπτών του μηνύματος ο SDNi Wrapper του τομέα που τα στέλνει υλοποιεί την sdni Northbound, μια συνάρτηση η οποία είναι υπεύθυνη για την αποστολή του κατάλληλου IODEF στους ”πελάτες” που χτυπάνε το URI που μεταφέρθηκε.

4.5.3 Άφιξη μηνύματος

Ο BGP Αναλυτής είναι υπεύθυνος και για την λήψη μηνυμάτων BGP και την ταξινόμηση τους. Πιο αναλυτικά, μόλις ληφθεί ένα BGP μήνυμα από κάποιον τομέα, ο BGP Αναλυτής ελέγχει το πεδίο των Διαδρομών προς απόσυρση και το πεδίο των χαρακτηριστικών του μονοπατιού του μηνύματος BGP. Αν και τα δύο αυτά πεδία είναι άδεια, στέλνει το NLRI μέρος του μηνύματος στον SDNi Wrapper. Ο SDNi wrapper χρησιμοποιεί το URI, που περικλείεται στο NLRI μέρος του μηνύματος και παίρνει το αρχείο IODEF μέσω αιτήματος στην δικτυακής υπηρεσία rest. Αφού ανακτήσει το αρχείο, το αποδίδει στον IODEF Χειριστής. Ο Χειριστής IODEF διαβάζει το IODEF και μεταφέρει τα απαραίτητα δεδομένα στον Διαχειριστή Συνεργατικότητας. Ο διαχειριστής Συνεργατικότητας υλοποιώντας κατάλληλους αλγορίθμους που θα αναλυθούν στο κεφάλαιο 7 αποφασίζει εαν το αίτημα για βοήθεια θα πρέπει να ικανοποιηθεί ή όχι. Στην περίπτωση αυτή ο Χειριστής ειδοποιεί τον Υποβολέα κανόνων να εφαρμόσει τις οδηγίες της έκκλησης για βοήθεια εγκαθιστόντας κανόνες στο επίπεδο δεδομένων και τον Χειριστή IODEF να συντάξει το μήνυμα που θα προωθηθεί περαιτέρω εφόσον ο τομέας δεν είναι (μόνο) πηγή.

4.6 Τυποποίηση αναφοράς γεγονότων ασφάλειας

Στο παρών κεφάλαιο θα αναλυθούν τα κίνητρα της χρήσης του και η ακριβής σύνταξη του. Κίνητρα χρήσης του είναι η ανάγκη για :

- αυτοματοποίηση στην επεξεργασία των δεδομένων. Αυτό εξασφαλίζεται διότι οι

υπολογιστικοί πόροι που χρειάζονται για την ανάλυση κειμένου σε ελεύθερη μορφή είναι αρκετά περισσότεροι σε σύγκριση με την περίπτωση μιας τυποποιημένης αναπαράστασης

- μείωση της προσπάθειάς και του κόστους ομαδοποίησης και συγκέντρωσης ανάλογων στοιχείων από διαφορετικές πηγές

- μια κοινή μορφή στην οποία μπορούμε να στηριχθούμε για να φτιάξουμε διαλειτουργικά εργαλεία για την συλλογή περιστατικών και την μετέπειτα ανάλυση που αυτά συνεπάγονται ειδικά όταν προέρχονται από πολλαπλές διαφορετικές τεχνολογικές μονάδες.

Στην βιβλιογραφία μπορεί να βρει κανείς αρκετές τέτοιες προσπάθειες τυποποίησης μηνυμάτων κάποια από τα οποία έγιναν RFC και όπως τα IETF (RFC 4765) [31], IODEF (RFC 5070)[32], x-ARR (RFC 5965) [**?**]. Στο Σχήμα 4.3 βλέπουμε ένα συγκριτικό πίνακα των χαρακτηριστικών τους.

	IDMEF	**IODEF**	**Warden**	**MAEC**	**X-ARF**	**AVDL**
Language	XML	XML		Is A	php, python,java	XML
RFC	Yes	Yes	No	No	No	OASIS Standard
Extensibility	Poor	Poor	Poor	Free text	Yes	
Format	Tags	Tags	free text		Schemes	XML file
Dynamic	Yes	Yes	No		No	
read H-C	H+	2	2		H-C	M
learning curve	Easy	Moderate		Steep	Easy	Steep
incident types diversity	Best	Best			Only 4 Schemes	Poor
communication method	transport independent (BEEP)	ND	HTTP/RCP	ND	EMail	files
Structure	deep	deep	Medium	Medium	Depends	deep

Σχήμα 4.2: *Συγκριτικός πίνακα τυποποίησεων μηνύματος*

4.7 Η δομή του αρχείου IODEF

Επιλέχθηκε η χρήση το μήνυμα IODEF καθώς γίνεται εύκολα κατανοητό από τον άνθρωπο και μπορεί να διαβαστεί γρήγορα από μηχανή προκαλώντας έτσι την ελάχιστη επιβάρυνση και τις ελάχιστες προϋποθέσεις για χρήσης. Το IODEF ή συντακτικό αντικειμένου ανταλλαγής, περιγραφής περιστατικού είναι ένα πρότυπο για την αναπαράσταση των πληροφοριών ασφαλείας των υπολογιστών που συνήθως ανταλλάσσονται μεταξύ Ομάδων Αντιμετώπισης Περιστατικών Ασφάλειας (CSIRT). Πρόκειται για μια αναπαράσταση σε XML που χρησιμοποιείται για τη διαβίβαση πληροφοριών κάποιου περιστατικού μεταξύ των διοικητικών τομέων. Η υιοθέτηση του, υπόσχεται σημαντική βελτίωση στην αποτελεσματικότητα των μεθόδων αντιμετώπισης επιθέσεων καθώς απλουστεύει την συνεργασία μεταξύ των εμπλεκομένων ομάδων και την ανταλλαγής δεδομένων. Το συνολικό συντακτικό του IODEF φαίνεται στο Σχήμα 4.2 είναι αρκετά περίπλοκο καθώς στοχεύει στην κάλυψη όσο το δυνατόν μεγαλύτερου μέρους των αναγκών σηματοδότησης για θέματα ασφάλειας. Στο Σχήμα 4.4 δίνεται μία δομή IODEF που είναι προσαρμοσμένη στις ανάγκες της εφαρμογής. Αρχικά παρατίθεται ένα παράδειγμα τέτοιου αρχείου και στην συνέχεια θα αναλυθούν μόνο τα πεδία που χρησιμοποιούνται στην εφαρμογής μας.

```
<?xml version="1.0" encoding="UTF-8" ?>
<IODEF-Document version="1.00" lang="en"
 xmlns="urn:ietf:params:xml:ns:iodef-1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:schema:iodef-1.0">
 <Incident purpose="mitigation">
 <IncidentID name="csirt.example.com">908711</IncidentID>
 <ReportTime>2006-06-08T05:44:53-05:00</ReportTime>
 <Description>Large bot-net</Description>
 <Assessment>
  <Impact type="dos" severity="high" completion="succeeded" />
 </Assessment>
 <Method>
  <!-- References a given piece of malware, "GT Bot" -->
  <Reference>
  <ReferenceName>GT Bot</ReferenceName>
  </Reference>
  <!-- References the vulnerability used to compromise the
   machines -->
  <Reference>
  <ReferenceName>CA-2003-22</ReferenceName>
  <URL>http://www.cert.org/advisories/CA-2003-22.html</URL>
  <Description>Root compromise via this IE vulnerability to
     install the GT Bot</Description>
  </Reference>
 </Method>
 <!-- A member of the CSIRT that is coordinating this
   incident -->
 <Contact type="person" role="irt">
```

```
 <ContactName>Joe Smith</ContactName>
 <Email>jsmith@csirt.example.com</Email>
</Contact>
<EventData>
 <Description>These hosts are compromised and acting as bots
    communicating with irc.example.com.</Description>
 <Flow>
 <!-- bot running on 192.0.2.1 and sending DoS traffic at
   10,000 bytes/second -->
 <System category="source">
  <Node>
  <Address category="ipv4-addr">192.0.2.1</Address>
  </Node>
  <Counter type="byte" duration="second">10000</Counter>
  <Description>bot</Description>
 </System>
 <!-- a second bot on 192.0.2.3 -->
 <System category="source">
  <Node>
  <Address category="ipv4-addr">192.0.2.3</Address>
  </Node>
  <Counter type="byte" duration="second">250000</Counter>
  <Description>bot</Description>
 </System>
 <!-- Command-and-control IRC server for these bots-->
 <System category="target">
  <Node>
  <NodeName>irc.example.com</NodeName>
  <Address category="ipv4-addr">192.0.2.20</Address>
  <DateTime>2006-06-08T01:01:03-05:00</DateTime>
  </Node>
  <Description>IRC server on #give-me-cmd channel</Description>
 </System>
 </Flow>
 <!-- Request to take these machines offline -->
 <Expectation action="investigate">
 <Description>Confirm the source and take machines off-line and
    remediate</Description>
 </Expectation>
</EventData>
</Incident>
</IODEF-Document>
```

Κάθε περιστατικό-επίθεση αντιπροσωπεύεται από ένα στιγμιότυπο της κλάσης περιστατικού. Κάθε στιγμιότυπο της κλάσης έχει τα κάποια ιδιο-χαρακτηριστικά το πιο σημαντικό από τα οποία είναι αυτό που ονομάζεται **"purpose"** Το χαρακτηριστικό αυτό της κλάσης χρησιμοποιείται για να δηλώσει τον σκοπό της αποστολής του αρχείου IODEF. Στην περίπτωση μας η τιμή του είναι **"mitigation"** και δηλώνει ότι το αρχείο στάλθηκε με σκοπό την αίτηση βοήθειας για την αντιμετώπιση της περιγραφόμενης κακόβουλης συμπεριφοράς. Θα δούμε τα πεδία που συνθέτουν την κλάση του περιστατικού χωριστά. Κάποια από τα πεδία έχουν πρωτόγονο τύπο, άλλα είναι και τα ίδια κλάσεις και άλλα είναι σύνολα:

- **IncidentID**: Η κλάση IncidentID περιέχει τον αριθμό που χρησιμοποιείται για

τον εντοπισμό ενός περιστατικού. Αυτό το αναγνωριστικό χρησιμοποιείται και ως κλειδί για την ανάκτηση του αρχείου μέσω μίας υπηρεσίας Rest. Η κλάση IncidentID έχει τρία ιδιο-χαρακτηριστικά από τα οποία χρησιμοποιούμε μόνο το **name** που είναι μία συμβολοσειρά η οποία αναφέρεται στον τομέα που δημιούργησε το αρχείο. Ο συνδυασμός του ονόματός και του περιεχομένου του IncidentID πρέπει να είναι ένα καθολικά μοναδικό αναγνωριστικό που περιγράφει το περιστατικό. Τα domains δηλαδή που παρήγαγαν ένα αρχείο δεν πρέπει να ξαναχρησιμοποιήσουν την ίδια τιμή, εκτός εάν αναφέρονται στο ίδιο περιστατικό.Γι τον σκοπό αυτό, πρέπει να χρησιμοποιηθεί το πλήρως αναγνωρισμένο όνομα τομέα που σχετίζονται με το στιγμιότυπο IODEF και ένα αναγνωριστικό που να αναφέρεται σε ένα υποσύνολο των δηλωμένων περιστατικών

- **AlternativeID**: Το πεδίο αυτό σύμφωνα με την προτυποποίηση του IODEF χρησιμοποιείται για τον εντοπισμό του ίδιου περιστατικού από μια διαφορετική ομάδα από αυτή που έχει παράξει το αρχείο. Στην περίπτωση μας χρησιμοποιείται για να μεταφέρει το αναγνωριστικό του περιστατικού όπως αυτό ορίστηκε για το αρχικό IODEF αρχείο που έφτιαξε ο διαχειριστικός τομέας που δεχόταν επίθεση. Χρησιμοποιείται δηλαδή σαν ασφαλιστική δικλίδα για να επικυρώσει ότι δεν πρόκειται για προσωπικές παρατηρήσεις του αποστολέα αλλά για απλή προώθηση. Η πληροφορία που μεταφέρει εμπλουτίζεται από το παρακάτω πεδίο.

- **RelatedActivity** Το πεδίο αυτό σύμφωνα με την προτυποποίηση του IODEF χρησιμοποιείται για να μεταφέρει είτε το αναγνωριστικό είτε κάποιο URL ενός γεγονότος που είναι σχετικό με το περιγραφόμενο συμβάν. Στην περίπτωση μας το πεδίο χρησιμοποιείται για να στεγάσει το URL το οποίο ο εκάστοτε παραλήπτης πρέπει να χτυπήσει μαζί με το αναγνωριστικό που δόθηκε στο προηγούμενο πεδίο ώστε να πάρει το αρχικό IODEF που περιγράφει την επίθεση από την μεριά του τομέα που την δέχεται.

- **ReportTime**: Το πεδίο αυτό χρησιμοποιείται για την χρονική σήμανση της παράδοσης του μηνύματος, που διαμορφώνεται σύμφωνα με τον τύπο δεδομένων DATETIME. Ανήκει στο μοντέλο δεδομένων Time το οποία χρησιμοποιεί πέντε διαφορετικές κατηγορίες για να εκφράσει την χρονική στιγμή. Οι κατηγορίες αυτές

έχουν προφανώς ίδιο τύπο δεδομένων αλλά κάθε μια έχει ένα ξεχωριστό όνομα για να μεταφέρει τη διαφορά στην σημασιολογία .

- **Description** : Το πεδίο χρησιμοποιείται για μια περιγραφή ελεύθερης μορφής για το περιστατικό

- **Assessment** : Το πεδίο περιγράφει τις τεχνικές και μη τεχνικές επιπτώσεις του συμβάντος στον διοικητικό τομέα που δέχεται επίθεση. Σύμφωνα με το RFC έχει ένα ιδιοχαρακτηριστικό που δείχνει κατά πόσο πρόκειται για γεγονός που έγινε ή που προβλέπεται ότι θα γίνει. Στην περίπτωση μας έχει την τιμή "actual" γιατί αναφέρεται σε γεγονός που έχει πραγματοποιηθεί. Η κλάση Assessment αποτελείται από κλάσεις των οποίων οι πιθανές τιμές για τις ανάγκες της εφαρμογής μας δίνονται στις αγκύλες ενώ ταυτόχρονα σχολιάζονται όσες είναι απαραίτητο να αναλυθούν.

  – Impact : πεδίο που αναφέρεται στις τεχνικές επιπτώσεις με χρήση ελεύθερης γλώσσας. ΄Έχει κάποια επιπλέον ιδιοχαρακτηριστικά που παρουσιάζονται παρακάτω.

    * lang Πεδίο που περιέχει ένα έγκυρο κωδικό γλώσσας σύμφωνα με το RFC 4646 [7]
    * severitylow , medium , high Πεδίο που αναφέρεται στην σοβαρότητα των συνεπειών της επίθεσης
    * completion succeded , failed Πεδίο που αναφέρεται στο κατά πόσο η επίθεση ολοκληρώθηκε- πέτυχε τον σκοπό της.
    * type dos Πεδίο που αναφέρεται στο είδος του περιστατικού από ένα σύνολο πιθανών περιστατικών που ορίζονται στο RFC.
    * ext-type

  – TimeImpact : πεδίο που αναφέρεται στις σχετικές με τον χρόνο επιπτώσεις. Το πεδίο αυτό είναι ένας θετικός πραγματικός αριθμός που εκφράζει ποσοτικά τη χρονική διάρκεια του περιστατικού. Το πεδίο αυτό έχει και ιδιοχαρακτηριστικά τα οποία προσδίδουν στην απόλυτη αυτή τιμή την σημασιολογία της. Πιο συγκεκριμένα τα ιδιοχαρακτηριστικά.

    * severity low , medium , high Πεδίο που αναφέρεται στην σοβαρότητα των συνεπειών της επίθεσης σε σχέση με τον χρόνο.

        * metric labor Συνολικός χρόνος που χρειάζεται για να αντιμετωπιστεί η επίθεση
        * duration second,minites,hours,days... : Πεδίο που δηλώνει τις μονάδες στις οποίες είναι εκφρασμένη η χρονική διάρκεια.
    – MonetaryImpact : πεδίο που αναφέρεται στις επιπτώσεις της επίθεσης από οικονομική άποψη. Περιέχει την απόλυτη τιμή της ζημιάς σε χρηματικές μονάδες .
        * severity low , medium , high Πεδίο που αναφέρεται στην σοβαρότητα των συνεπειών της επίθεσης από οικονομική σκοπιά
        * currency second,minites,hours,days... το είδος των χρηματικών μονάδων στο οποίο είναι εκφρασμένη η χρηματική ζημία.
    – Confidence: Πεδίο που εκφράζει την σιγουριά για την ορθότητα του συναγερμού rating low , medium , high, znumeric

- **Method**: Πεδίο που αναφέρεται στην τεχνική που χρησιμοποιήθηκε στην επίθεση. Αποτελείται από μια λίστα από αναφορές που περιγράφουν την μέθοδο επίθεση και ένα πεδίο που ονομάζεται “Description” και είναι ελευθέρας μορφής κείμενο που περιγράφει την μεθοδολογία των εισβολέων.

- **History** : Το πεδίο αυτό περιέχει την γνώμη του αποστολέα του IODEF σχετικά με το πόση διάθεση για συνεργασία έχει δείξει ο διαχειριστικός τομέας που του έστειλε την έκκληση για βοήθεια ανεξάρτητα αν αυτός δέχεται επίθεση ή είναι κάποιος ενδιάμεσος κόμβος. Το πεδίο αυτό χρησιμοποιείται σύμφωνα με το RFC για την ενημέρωση του αποστολέα για μία σειρά ενεργειών ενός τομέα δηλαδή περιέχει πολλα στιγμιότηπα της κλάσης “HistoryItem”. Στην περίπτωση μας χρειάζεται μόνο η άποψη του αποστολέα (τομέα που φτιάχνει το αρχείο IODEF) καθώς και κάποιες σταθερές που δηλώνουν το κατά πόσο η γνώμη αυτή είναι αντιπροσωπευτική. Για τον λόγο αυτό χρησιμοποιούνται τα εξής πεδία
    – DateTime : χρονική στιγμή τελευταίας εξέτασης συνεργατικότητας, δηλαδή της τελευταίας φοράς που ο τομέας που φτιάχνει το αρχείο IODEF έστειλε κάποιο αίτημα συνεργασίας και έλεγξε κατά πόσο ο παραλήπτης του συνεργάστηκε ή όχι.

  - AditionalData: Πρόκειται για ένα είδος πεδίου το οποίο υπάρχει σε πολλά σημεία της δομής του IODEF και εξυπηρετεί ανάγκες επέκτασης. Στην περίπτωση μας περιέχει την τιμή του επιπέδου συνεργατικότητας καθώς και των αριθμό των συναλλαγών μεταξύ του αποστολέα και του περιγραφόμενου τομέα.

- **EventData**: Πρόκειται για πολλά στιγμιότυπα που περιέχονται μέσα σε ένα στιγμιότυπο τύπου EventData και κληρονομούν τις τιμές του προγόνου τους. Αυτός ο αναδρομικός ορισμός μπορεί να χρησιμοποιηθεί για μία ομάδα από κοινά δεδομένα σχετικά με πολλαπλά συμβάντα. Όταν τα στιγμιότυπα της κλάσης EventData ορίζονται αναδρομικά, μόνο τα φύλλα αντιπροσωπεύουν πραγματικά γεγονότα. Κάθε στιγμιότυπο EnentData μπορεί να έχει Description, DetectTime, StartTime, EndTime, Contact, Assesment, Method, Flow, Expectation,Record EventData και AdditionalData. Είναι φανερό δηλαδή ότι υπάρχει σημαντική επικάλυψη στις κατηγορίες Incedent και EventData. Ωστόσο, η σημασιολογία αυτών των κατηγοριών είναι αρκετά διαφορετική. Η κλάση Incident παρέχει συνοπτικές πληροφορίες σχετικά με το σύνολο του περιστατικού, ενώ η κατηγορία EventData παρέχει πληροφορίες σχετικά με τα ατομικά γνωρίσματα που περιλαμβάνει το περιστατικό. Στην πιο συνηθισμένη περίπτωση,η κατηγορία EventData θα παρέχει πιο συγκεκριμένες πληροφορίες για την γενική περιγραφή που παρέχεται στην κλάση. Στην περίπτωση μας στο IncidentData περιγράφεται η επίθεση DDos συνολικά και στο EventData περιγράφονται οι πηγές της κίνησης. Αξίζει να σημειωθεί για να γίνει πιο κατανοητή η διαφορά μεταξύ των δύο πεδίων ότι είναι πιθανό η συνολική συνοπτική πληροφορία σχετικά με το περιστατικό να έρχεται σε αντίθεση με ατομικές πληροφορίες σε EventData όταν υπάρχει μια ουσιαστική σύνθεση διαφόρων εκδηλώσεων σε ένα μόνο περιστατικό. Σε μια τέτοια περίπτωση, η ερμηνεία των πιο ειδικών EventData πρέπει να αντικαταστήσουν τις γενικότερες πληροφορίες που παρέχονται στα IncidentData. Στην συνέχεια θα εξετάσουμε μόνο τα πεδία της κλάσης EventData που δεν έχουν ήδη αναλυθεί δηλαδή δεν περιέχονται και στην κλάση IncidentData,

  - Expectation :Το πεδίο αυτό μεταβιβάζει στον παραλήπτη του IODEF την ενέργεια που ο αποστολέας του ζητάει να εφαρμόσει σε συγκεκριμένες ροές. Τα

πεδία που περιέχει αναλύονται παρακάτω:

* Description : Περιγραφή σε κείμενο ελεύθερης γραφής
* StartTime: Δηλώνει την χρονική στιγμή κατά την οποία ο παραλήπτης καλείται να εφαρμόσει τις περικλειόμενες οδηγίες. Αν αυτή η τιμή είναι μικρότερη από την χρονική στιγμή της λήψης τότε αυτό δηλώνει την παράκληση του αποστολέα να εφαρμόσει τις οδηγίες όσο νωρίτερα μπορεί.

Η κλάση Expectations έχει τέσσερα ιδιοχαρακτηριστικά τα σημαντικότερα από τα οποία είναι το "action" το οποίο ορίζει πως πρέπει οι παραλήπτες να χειριστούν τις περιγραφόμενες ροές. Αυτή μπορεί να πάρει αρκετές τιμές σύμφωνα με την προτυποποίηση όμως για τις ανάγκες της εφαρμογής μας μπορούν να χρησιμοποιηθούν μόνο οι :

* contact-source-site: ζητάει από τον παραλήπτη να επικοινωνήσει με τον τελικό χρήστη. Χρησιμοποιείται στο τελευταίο στάδιο της αντιμετώπισης της επίθεσης δηλαδή όταν το IODEF έχει φτάσει ένα βήμα πριν το φύλλο του δέντρου της επίθεσης.
* block-host: ζητάει από τον παραλήπτη να κόψει την κίνηση που περιγράφεται ως πηγή της επίθεσης
* rate-limit-host: ζητάει από τον παραλήπτη να μειώσει την κίνηση που περιγράφεται ως πηγή της επίθεσης

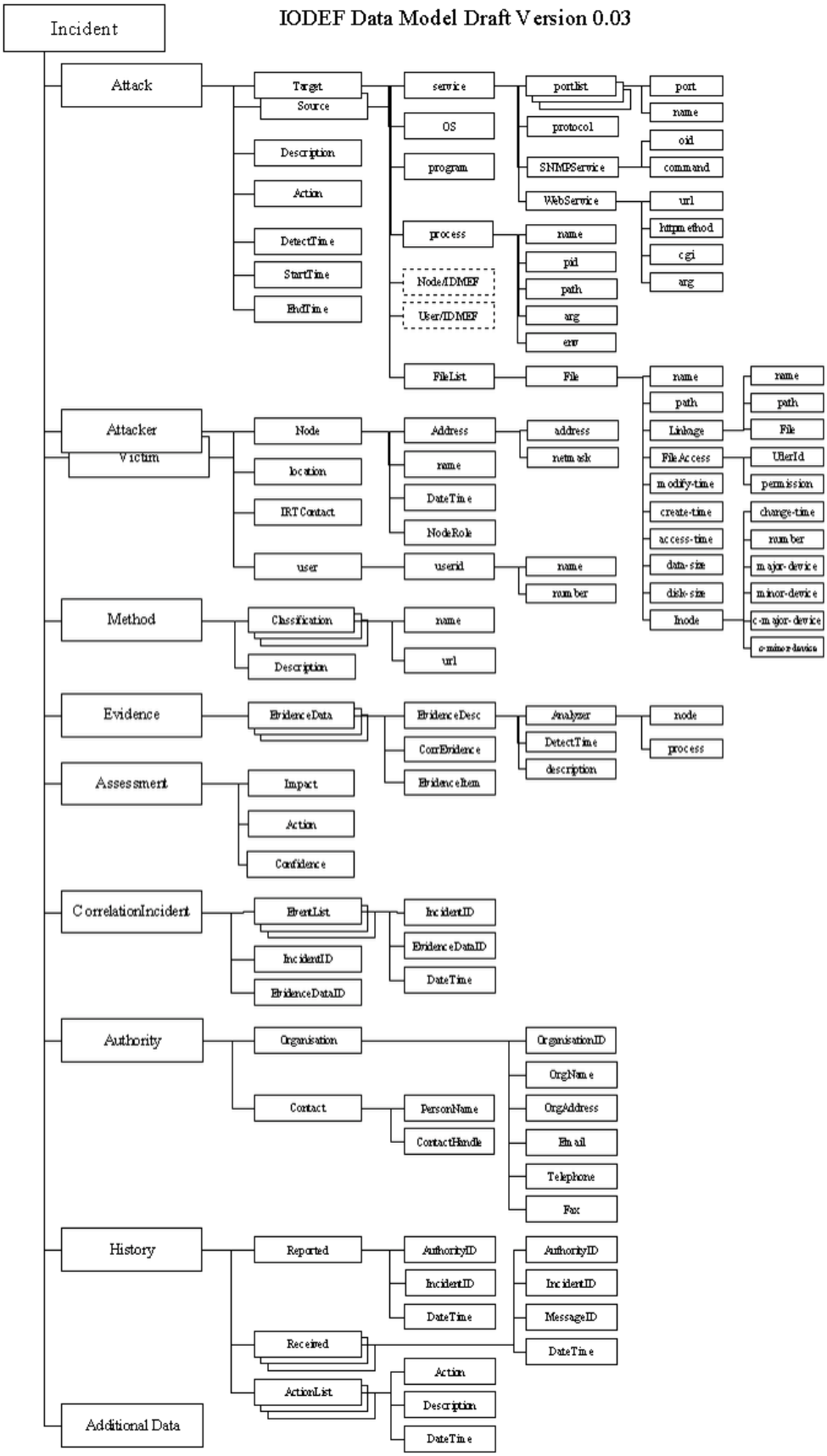
– Flow: Το πεδίο αυτό περιγράφει τα συστήματα που εμπλέκονται στο περιγραφόμενο γεγονός. Στην ουσία αποτελείται από στιγμιότυπα της κλάσης System και χρησιμοποιείται για να διαχωρίσει τους χρήστες που είναι πηγές της επίθεσης από τα θύματα - προορισμούς. Η κλάση System περιγράφει το σύστημα ή δίκτυο που συμμετέχει στο περιστατικό. Είχε δύο ιδιοχαρακτηριστικά:

* category: Τα συστήματα ή δίκτυα που αναπαρίστανται από αυτή την κλάση κατηγοριοποιούνται με βάση τον ρόλο που έπαιξαν στο περιστατικό μέσω του ιδιοχαρακτηριστικού "category". Η τιμή αυτού αποδίδει σημασιολογία στα πεδία της κλάσης του συστήματος που αναφέρονται αναλυτικότερα παρακάτω. Εάν το ιδιοχαρακτηριστικό category"έχει την τιμή "source", τότε η συγκεντρωτική κλάση δηλώνει ένα μηχάνημα από

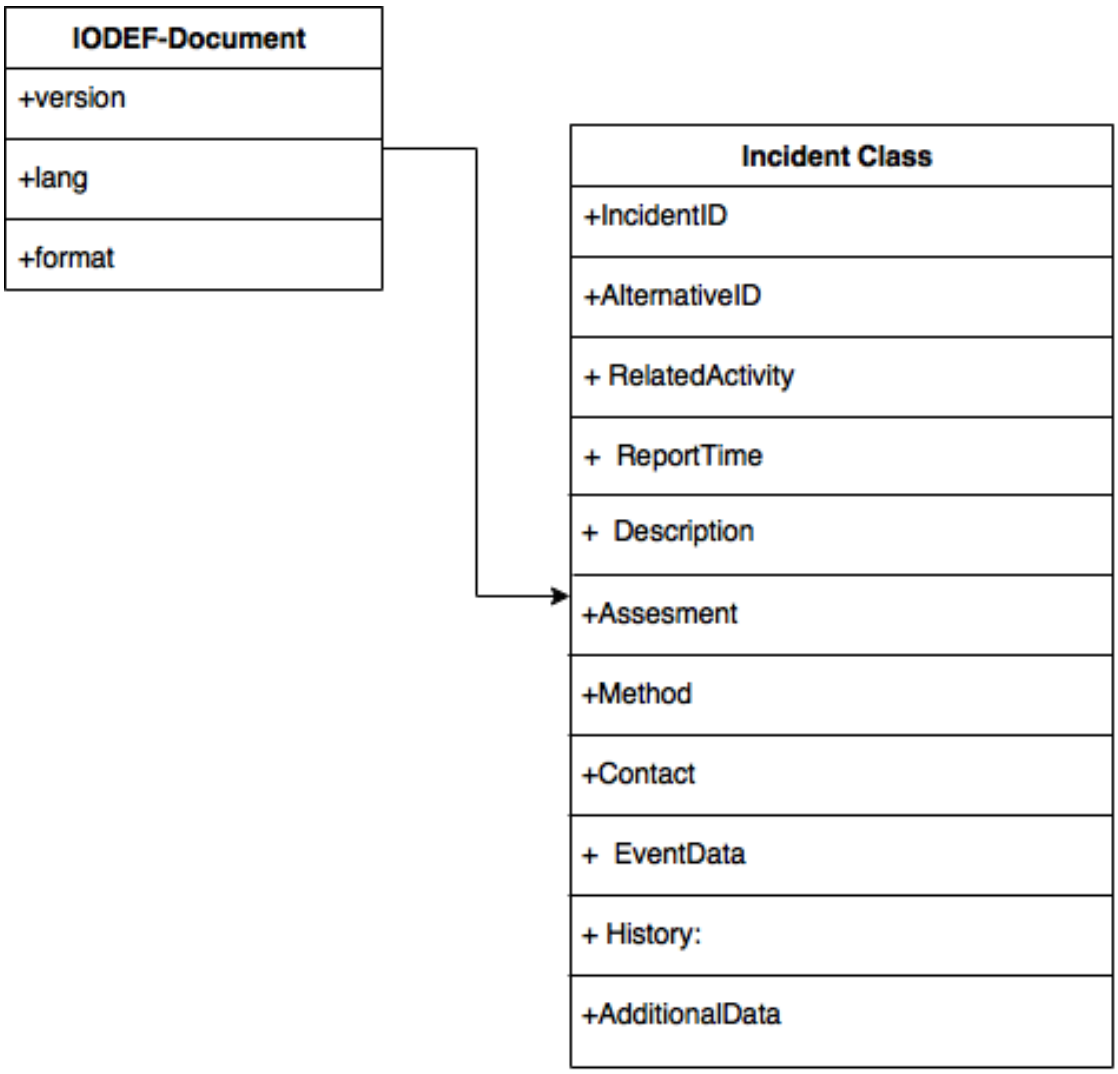
το οποίο η επίθεση ξεκίνησε. Εάν η τιμή είναι ”target” τότε πρόκειται για το μηχάνημα ή την υπηρεσία που βρίσκεται στο στόχαστρο της επίθεσης. ‘

* spoofed : Το ιδιοχαρακτηριστικό αυτό χρησιμοποιείται για να στεγάσει την υποψία ότι τα IP των πηγών δεν είναι πραγματικά. Η default τιμή του πεδίου είναι “unknown” αλλά μπορεί να μετατραπεί σε “yes” από οποιονδήποτε διαχειριστικό τομέα εάν αυτός διαπιστώσει για παράδειγμα ότι έρχεται κίνηση από το ίδιο IP από περισσότερους του ενός διαφορετικούς τομείς.Η κλάση system αποτελείται από τις παρακάτω κλάσεις.

* Node: Η κλάση αυτή κατονομάζει το σύστημα (υπολογιστή, δρομολογητή) ή δίκτυο. Οι κλάσεις που συνιστούν την κλάση node φαίνονται παρακάτω
  - NodeName : πεδίο που δηλώνει το όνομα του κόμβου
  - Address : πεδίο που δηλώνει την διεύθυνση του χρήστη του υλικού, του δικτύου ή και της εφαρμογής . Σύμφωνα με το πρότυπο έχει αρκετά πεδία εμείς χρησιμοποιούμε μόνο ένα από αυτά και συγκεκριμένα την κατηγορία που αναφέρεται στο είδος της διεύθυνσης που δηλώνεται. Η τιμή αυτού του πεδίου εξαρτάται από την έξοδο του μηχανισμού αναγνώρισης των πηγών και στην περίπτωση μας είναι category IPv4-addr
  - Service : πεδίο που δηλώνει την δικτυακή υπηρεσία που έτρεχε στο σύστημα
  - Counter: πεδίο που δηλώνει ένα μετρητή της δραστηριότητας του χρήστη. Στην περίπτωση που πρόκειται για πηγή μπορεί να δοθεί ο αριθμός των πακέτων που αυτός έστειλε εαν υποστηρίζεται από τον μηχανισμό ανίχνευσης.
* description Περιγραφή σε κείμενο ελεύθερης γραφής



Σχήμα 4.3: *Πλήρης δομή αρχείου IODEF Πηγή:https://www.terena.org/activities/tf-csirt/iodef/docs/draft-terena-iodef-xml-003-final.txt*



Σχήμα 4.4: *Δομή IODEF, προσαρμοσμένη στις ανάγκες της εφαρμογής*

Κεφάλαιο 5

Η εμπιστοσύνη και η φήμη

5.1 Ορισμός- Γενικά Στοιχεία

Η εμπιστοσύνη και η φήμη στους χρήστες ενός συστήματος αντιπροσωπεύουν μια σημαντική τάση στην στήριξη αποφάσεων στο διαδίκτυο με τη μεσολάβηση της παροχής υπηρεσιών. Η βασική ιδέα είναι να αφήσουμε τους χρήστες να αξιολογήσουν ο ένας τον άλλον. Μετά την ολοκλήρωση της όποιας συναλλαγής, όλες οι αξιολογήσεις μπορούν να συνδυαστούν για να αντληθεί για έναν συγκεκριμένο χρήστη μία μετρική που να εκφράζει την φήμη ή την εμπιστοσύνη που οι άλλοι έχουν σε αυτόν. Η τιμή αυτή μπορεί βοηθήσει στην συνέχεια τους άλλους χρήστες να αποφασίσουν κατά πόσον αξίζει να συναλλάσσονται με τον εν λόγω χρήστη στο μέλλον. Μια φυσική συνέπεια είναι η μετρική αυτή, να αποτελεί επίσης κίνητρο για την καλή συμπεριφορά των χρηστών, αφού γνωρίζουν ότι παρακολουθούνται, και ως εκ τούτου η χρήση της μετρικής τείνει να έχει μια θετική επίδραση στην ποιότητα του συνολικού συστήματος.

Συστήματα φήμης μπορούν να αναφέρονται και ως συνεργατικά συστήματα επιβολής κυρώσεων ώστε ο όρος να αντικατοπτρίζει την συνεργατική φύση, των συστημάτων και να σχετίζεται με συνεργατικά συστήματα φιλτραρίσματος. Συστήματα Φήμης χρησιμοποιούνται ήδη σε επιτυχημένες εμπορικές εφαρμογές στο διαδίκτυο. Υπάρχει επίσης ταχέως αυξανόμενη βιβλιογραφία γύρω από τα συστήματα εμπιστοσύνης και φήμης, αλλά, δυστυχώς, αυτή η δραστηριότητα δεν είναι πολύ συνεπής.

Ένα από τα βασικότερα προβλήματα των συνεργατικών συστημάτων αντιμετώπισης επιθέσεων είναι η μη αυτοματοποιημένη διαδικασία λήψη απόφασης σχετικά με την συμμετοχή ή μη στην υπεράσπιση του στόχου της επίθεσης γεγονός που επιφέρει σημαντικές καθυστερήσεις οι οποίες επηρεάζουν καθοριστικά την αποτελεσματικότητα των σχημάτων αντιμετώπισης επιθέσεων. Στην πράξη σήμερα σε ελάχιστες περιπτώσεις εάν ο κίνδυνος είναι πολύ μεγάλος οι πάροχοι μπορούν να επικοινωνήσουν μεταξύ τους για να αλληλοβοηθηθούν. Μία τέτοια πρακτική ωστόσο, δεν κλιμακώνεται και ξοδεύει πολύτιμο χρόνο για την αντιμετώπιση της επίθεσης, με αποτέλεσμα συχνά να συντελεί στην ολοκλήρωση της. Συνεπώς,ένα σύστημα φήμης- εμπιστοσύνης θα διευκολύνει, θα αυτοματοποιήσει και συνεπώς θα επιταχύνει την διαδικασία λήψης αποφάσεων σε κάθε τομέα που δέχεται κάποιο αίτημα για βοήθεια.

Ένα επιπλέον μελανό σημείο, των συνεργατικών συστημάτων αντιμετώπισης επιθέσεων είναι η υπόθεση ότι όλοι οι διοικητικοί τομείς θα αφιέρωναν ένα κομμάτι των πόρων τους, με σκοπό να συνεισφέρουν στην αντιμετώπιση επίθεσεις σε κάποιο άλλο τομέα. Στην πράξη αυτό προφανώς δεν ισχύει ειδικά εάν, η διαθεσιμότητα του δεύτερου δεν κινδυνεύει λόγω της επίθεσης στον πρώτο ή εάν οι δύο τομείς δεν ανήκουν σε συνεργαζόμενες δικτυακές διαχειριστικές αρχές. Επομένως ένα σύστημα φήμης- εμπιστοσύνης θα προσέφερε κίνητρο στους διαχειριστικούς τομείς, να συμμετέχουν στην διαδικασία αντιμετώπισης. Το κίνητρο αυτό είναι η αύξηση της μετρικής που εκφράζει πόσο συνεργατική συμπεριφορά επιδεικνύουν που έχει σαν αποτέλεσμα και οι ίδιοι να απολαύσουν κάποια βοήθεια σε περίπτωση που δεχτούν επίθεση.

5.2 Η έννοια της εμπιστοσύνης και η εφαρμογή της

Οι εκδηλώσεις της εμπιστοσύνης είναι εύκολο να αναγνωριστούν γιατί τις βιώνουμε και βασιζόμαστε σε αυτές κάθε μέρα, αλλά την ίδια στιγμή η εμπιστοσύνη σαν έννοια είναι πολύ δύσκολο να οριστεί καθώς εκδηλώνεται σε πολλές διαφορετικές μορφές. Η βιβλιογραφία σχετικά με την εμπιστοσύνη εσωκλύει επίσης αρκετή σύγχυση επειδή ο όρος χρησιμοποιείται με πολλαπλές σημασίες. Ο δύο βασικοί ορισμοί στην βιβλιογραφία σχετικά με την εμπιστοσύνη δημιουργούν ένα διαχωρισμό μεταξύ της εμπιστοσύνης της αξιοπιστίας και της εμπιστοσύνης της απόφασης αντίστοιχα [33]. Εμείς θα ασχοληθούμε μόνο

με την πρώτη καθώς καλύπτει καλύτερα σημασιολογικά το πρόβλημά μας. Ο ορισμός της είναι ο παρακάτω:

> Εμπιστοσύνη Αξιοπιστίας: Εμπιστοσύνη είναι η υποκειμενική πιθανότητα με την οποία ένα άτομο Α, αναμένει ότι κάποιο άλλο άτομο Β, εκτελεί μια συγκεκριμένη ενέργεια από την οποία ο Α εξαρτάται.

Ο ορισμός αυτός περιλαμβάνει την έννοια της εξάρτησης από έναν αξιόπιστο χρήστη, και της αξιοπιστίας του, που στην ουσία είναι μία πιθανότητα. Ταιριάζει σημασιολογικά με την περίπτωση μας, όπου η εμπιστοσύνη είναι η υποκειμενική πιθανότητα, με την οποία ένας τομέας Α αναμένει ότι ένας τομέας Β, θα συνεισφέρει στην αντιμετώπιση μίας επίθεσης εφόσον του ζητηθεί από τον Α.

Στην συνέχεια, θα εξεταστεί με ποιον τρόπο επιτυγχάνονται οι δύο απαιτήσεις (παροχή κινήτρου για καλύτερη συμπεριφορά και αυτοματοποίηση της απόφασης) που περιγράφτηκαν παραπάνω σε τρία διαφορετικά σενάρια όσο αναφορά την σχετική θέση μεταξύ των Α και Β.

Σενάριο 1

Όταν μιλάμε για domains εντός ενός αυτόνομου συστήματος στο διαδίκτυο τότε η εμπιστοσύνη και η συνεργασία μεταξύ τους θεωρείται τετριμμένη και δεδομένη. Σε αυτή την περίπτωση τόσο το θέμα της λήψης απόφασης όσο και του κινήτρου για καλή συμπεριφοράς είναι λυμένα. Πιο συγκεκριμένα, στην έκκλιση για βοήθεια από ένα τομέα του ίδιου ΑΣ η απάντηση είναι πάντα θετική. Συνεπώς, δεν χρειάζεται κάποιο επιπλέον σύστημα για την εξασφάλιση της αυτοματοποίησης πέρα από την αναγνώριση ότι το μήνυμα προέρχεται από ένα τομέα του ίδιου ΑΣ. Επιπλέον, όντας κάτω από κάποια κοινή διαχειριστική οντότητα είναι βέβαιο ότι όλοι οι τομείς που απαρτίζουν ένα αυτόνομο σύστημα θα είναι πρόθυμοι να συνεισφέρουν στην αντιμετώπιση μίας επίθεσης που λαμβάνει χώρα στο ΑΣ τους. Συνεπώς, όσο αφορά την παροχή κινήτρου, η συνολική ευημερία και εύρυθμη λειτουργία του συνολικού ΑΣ είναι υπεραρκετή.

Σενάριο 2

Ωστόσο, όταν μιλάμε για τομείς σε διαφορετικά αυτόματα συστήματα, τότε για να ικανοποιήσουμε τις απαιτήσεις μας πρέπει να εισάγουμε μια μετρική για την ποσοτικοποίηση της έννοιας της εμπιστοσύνης, όπως αυτή ορίστηκε παραπάνω. Η μετρική που εισάγουμε ονομάζεται επίπεδο συνεργατικότητας και συνάδει με την έννοια της εμπιστοσύνης.

Επίπεδο συνεργατικότητας : εκφράζει την πιθανότητα ο Β να συνεισφέρει στην αντιμετώπιση μελλοντικών επιθέσεων εναντίον του Α από την οπτική γωνία του Α.

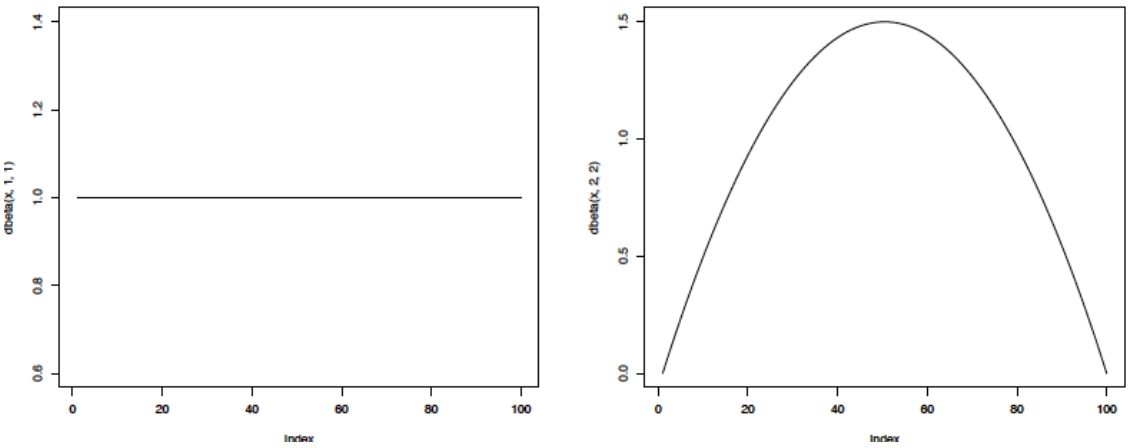
Η πυκνότητα πιθανότητας αυτής της μετρικής ακολουθεί την βήτα κατανομή . Η συνάρτηση Βήτα χρησιμοποιείται γενικά για περιπτώσεις όπου η ακριβής μορφή της συνάρτησης πυκνότητας πιθανότητας δεν είναι γνωστή, αλλά υπάρχουν ορισμένες ενδείξεις ότι η πυκνότητα παρουσιάζει μέγιστο κοντά σε κάποια τιμή. Η συνάρτηση πυκνότητας πιθανότητας της Βήτα κατανομής,έχει δύο θετικές παραμέτρους α, β ανάλογα με την επιλογή των οποίων παίρνει διάφορες μορφές. Επομένως, είναι δυνατόν να προσαρμοσθεί στα εμπειρικά δεδομένα. Σε κάθε χρονική στιγμή το επίπεδο εμπιστοσύνης δίνεται από την μέση τιμή αυτής της κατανομής. Συνεπώς η μετρική αυτή μπορεί να πάρει τιμές από μηδέν μέχρι ένα με το ένα να εκφράζει την απόλυτη εμπιστοσύνη ή την απόλυτη σιγουριά του Α ότι ο Β θα τον στηρίξει αν χρειαστεί. Σε κάθε χρονική στιγμή το επίπεδο εμπιστοσύνης δίνεται από την μέση τιμή αυτής της κατανομής. Συνεπώς η μετρική αυτή μπορεί να πάρει τιμές από μηδέν μέχρι ένα με το ένα να εκφράζει την απόλυτη εμπιστοσύνη ή την απόλυτη σιγουριά του Α ότι ο Β θα τον στηρίξει αν χρειαστεί.Το επίπεδο συνεργατικότητας ανανεώνεται μέσω των παραμέτρων α και β ανάλογα με τις παρατηρήσεις του Α κάθε φορά δηλαδή που ο Α ζητάει από τον Β βοήθεια είτε επειδή ο ίδιος δέχεται επίθεση είτε εκ μέρους κάποιου άλλου. Πιο συγκεκριμένα αν ο Β βοήθησε στην αντιμετώπιση της επίθεσης δηλαδή εγκατέστησε τους ζητούμενους κανόνες η παράμετρος s τίθεται στο 1 διαφορετικά τίθεται στο μηδέν. Με βάση αυτό το s ανανεώνονται τα α και β σύμφωνα με τον τύπο.

$$\alpha := u\alpha + s \tag{5.1}$$

$$\beta := u\beta + (1 - s) \tag{5.2}$$

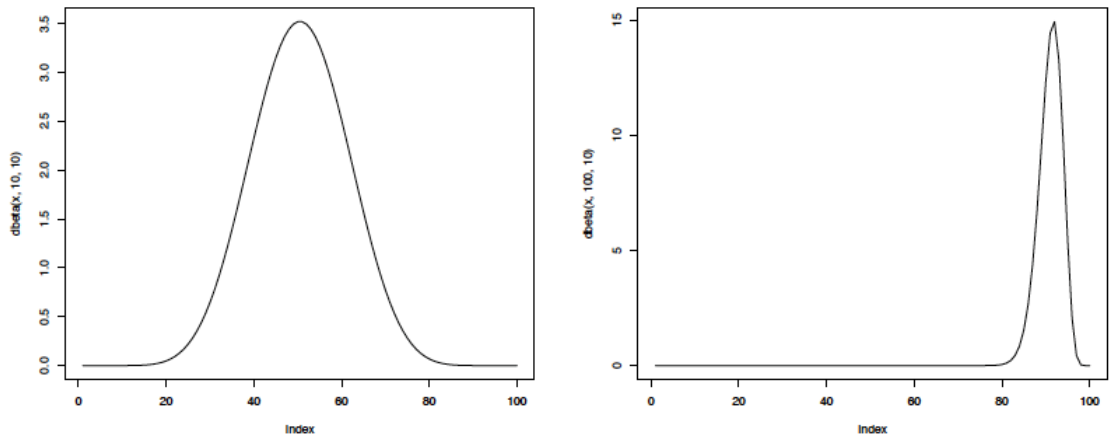
Άξια σχολιασμού είναι και η χρήση της παραμέτρου υ. Η παράμετρος αυτή ονομάζεται

παράγοντας μνήμης είναι κοντά στην μονάδα αλλά μικρότερη από αυτήν και χρησιμοποιείται για να εξασφαλίσουμε ότι οι πιο πρόσφατες παρατηρήσεις θα έχουν μεγαλύτερη ισχύ στον υπολογισμό του τελικού επιπέδου συνεργατικότητας και συνεπώς η τιμή αυτή θα προσαρμόζεται στην συμπεριφορά του εκάστοτε τομέα. Είναι σημαντικό επίσης να τονιστεί ότι η ορθότητα της πρόβλεψης της καμπύλη βήτα εξαρτάται από τον αριθμό των παρατηρήσεων. Δηλαδή όσο αυξάνονται οι παρατηρήσεις η καμπάνα της καμπύλης στενεύει ή ισοδύναμα οι πιθανότητες είναι μαζεμένες στην μέση τιμή . Το φαινόμενο αυτό φαίνεται καλύτερα στα Σχήματα 5.2 και 5.1 . Στο πρώτο, φαίνεται η καμπύλη όπως αρχικοποιήθηκε δηλαδή χωρίς καμία παρατήρηση από το δεύτερο και μετά προστίθενται παρατηρήσεις και παρουσιάζονται ενδεικτικά οι γραφικές της Beta(2,2) , Beta (10,10) και Beta(100,10) αντίστοιχα. Εκτός από το στένευμα της καμπάνας λόγω της αύξησης των παρατηρήσεων, βλέπουμε ότι το μέγιστο της καμπύλης τείνει προς τιμές κοντά στη μονάδα. Αυτό συμβαίνει διότι στην συντριπτική πλειοψηφία των παρατηρήσεων ο Β συνεργάζεται και άρα μεγαλώνει και η πίστη του Α ότι θα έχει την στήριξη του Β όταν και αν την χρειαστεί.



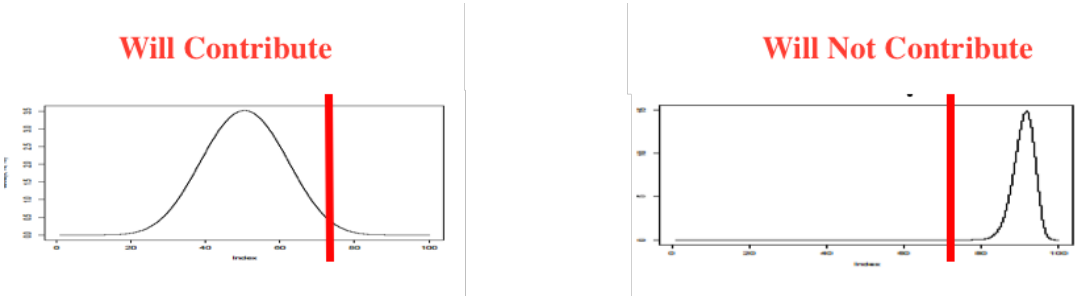
Σχήμα 5.1: *Αρχικοποίηση και Σταδιακή προσαρμογή της βήτα κατανομής*

Κάθε φορά που ο τομέας Α ικανοποιήσει μία αίτηση για συμμετοχή στην αντιμετώπιση μίας επίθεσης, από κάποιον γειτονικό τομέα Β τότε ο Α θα ελέγξει το επίπεδο συνεργατικότητας που έχει υπολογίσει για τον Β και αν αυτό ξεπερνάει ένα καθορισμένο όριο τότε θα συνεισφέρει. Στην περίπτωση αυτή θα λέμε ότι ο Β ικανοποιεί τα κριτήρια συνεργασιμότητας του Α. Η γραφική αναπαράσταση αυτής της απόφασης φαίνεται στην Σχήμα 5.3 όπου η κόκκινη γραμμή είναι αυτό το καθορισμένο όριο. Αν η Βήτα συνάρτηση παρουσιά-



Σχήμα 5.2: *Προσαρμογή βήτα κατανομής στα πειραματικά δεδομένα*

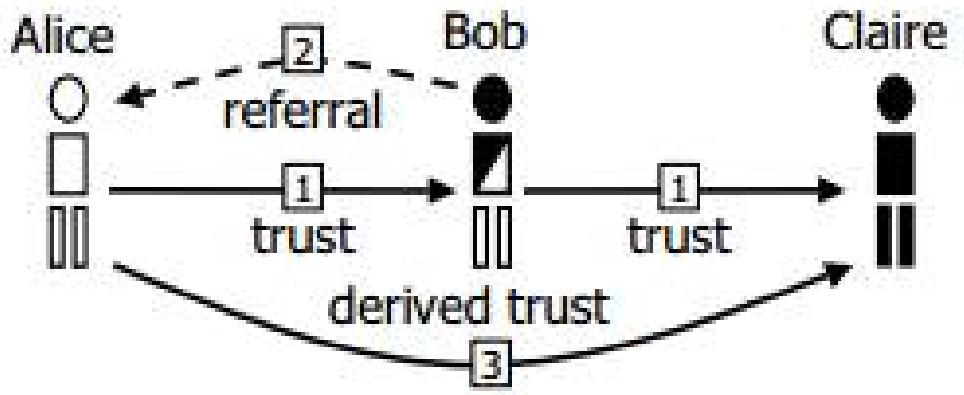
ζει μέγιστο αριστερά από την κόκκινη γραμμή τότε ο Α αποφασίζει να μην συνεισφέρει αν παρουσιάζει δεξιά τότε συνεισφέρει.



Σχήμα 5.3: *Λήψη απόφασης βάσει του μεγίστου της βήτα κατανομής*

Αυτό που αξίζει να αναφερθεί εδώ είναι ότι κάθε τομέας Α μπορεί να διαπιστώσει εάν ο Β συμμετείχε ή όχι εφόσον αυτοί είναι γειτονικοί, εκμεταλλευόμενος τις δυνατότητες του OpenFlow. Πιο συγκεκριμένα ο Α πριν ζητήσει την συμμετοχή στην αντιμετώπιση μίας επίθεσης από τον Β με ένα σύνολο κανόνων, εγκατέστησε ο ίδιος τους κανόνες αυτούς στον πίνακα ροής του. Εάν ο Β αποφασίσει να συνεισφέρει στην αντιμετώπιση και να εγκαταστήσει αυτό το σύνολο τότε οι κανόνες στο πίνακα ροής του Α θα λήξουν καθώς τα πακέτα δεν θα φτάνουν πια σε αυτόν επειδή επειδή θα μπλοκάρονται από τον Β που είναι πιο μπροστά στο μονοπάτι από την πηγή στο θύμα. Με τον τρόπο αυτό ο Α γνωρίζει την συμπεριφορά του Β και ανανεώνει αντίστοιχα την μετρική του επιπέδου συνεργατικότητας.

Σενάριο 3



Σχήμα 5.4: *Παράδειγμα μεταβατικής ιδιότητας στην εμπιστοσύνη, Πηγή: [22]*

Ωστόσο, τι συμβαίνει με τις εκκλήσεις που πηγάζουν από ένα τομέα ενός διαφορετικού αυτόνομου συστήματος με τον οποίο ο παραλήπτης δεν έχει άμεση σύνδεση; Προφανώς, ο παραλήπτης δεν έχει κάποια γνώμη για το επίπεδο συνεργατικότητας του αποστολέα γιατί δεν έχει σχετικές παρατηρήσεις. Για να αντιμετωπιστεί αυτό το πρόβλημα θα χρησιμοποιήσουμε στην έννοια της μεταβατικής ιδιότητας της εμπιστοσύνης. Η μεταβατικότητα ως μέθοδος για την εξαγωγή εμπιστοσύνη από μια μεταβατική πορεία εμπιστοσύνης είναι ένα στοιχείο που συναντάται πολύ συχνά σε συστήματα φήμης. Η ιδέα πίσω από την μεταβατικότητα της εμπιστοσύνη μπορεί να γίνει εύκολα αντιληπτή μέσα από το παράδειγμα του Σχήματος 5.4 ’Οταν η Alice εμπιστεύεται ο Bob και ο Bob εμπιστεύεται Claire, και ο Bob αναφέρει την Claire στην Alice, τότε η Alice μπορεί να αντλήσει ένα μέτρο για την εμπιστοσύνης που θα μπορούσε να έχει στην Claire από την αναφορά του Bob σε συνδυασμό με την εμπιστοσύνη της στον Bob. Προφανώς για να είναι σωστή η μετάβαση πρέπει, σύμφωνα με την βιβλιογραφία, να τηρούνται κάποιοι σημασιολογικοί περιορισμοί οι οποίοι δηλώνονται παρακάτω:

1. Η Alice πρέπει να εμπιστεύονται τον Bob
2. Ο Bob να συστήσει Claire για ένα συγκεκριμένο σκοπό
3. Ο Bob πρέπει να εμπιστευτούν Claire για τον ίδιο σκοπό

Αντίστοιχα εμείς χρησιμοποιούμε αυτήν την ιδιότητα για να αποδείξουμε την ορθότητα της μετάβασης της τιμής του επιπέδου συνεργατικότητας που αφορά ένα τομέα εκτός του

ΑΣ σε τομείς εντός αυτού. Οι σημασιολογικοί περιορισμοί ισχύουν καθώς:

1. Όλοι οι τομείς στο ίδιο ΑΣ εμπιστεύονται ο ένας τον άλλο
2. Για να κληθεί ένα άλλος τομέας στο ΑΣ του Α να αποφασίσει εάν θα συνεισφέρει το Β θα πρέπει ο Α να έχει προωθήσει το αίτημα, να έχει δηλαδή διαφημίσει- συστήσει το επίπεδο της συνεργατικότητας του Β
3. Ο τομέας του ΑΣ έστω Α που πρώτος από όλο το ΑΣ πήρε την έκκληση για βοήθεια ενός τομέα Β σε άλλο ΑΣ το προωθεί στους τομείς του ΑΣ στο οποίο ανήκει μόνο εφόσον η συμπεριφορά του Β ικανοποιεί τα κριτήρια συνεργατικότητας του Α που είναι ο σκοπός που θα τον εμπιστευτούν και οι άλλοι τομείς στο ΑΣ του Α.

Με τον τρόπο αυτό, η απόφαση για το εάν ένα αυτόματο σύστημα θα συνεισφέρει ή όχι είναι συνολική για όλο το ΑΣ και παίρνεται από τούς τομείς που βρίσκονται στα άκρα δηλαδή επικοινωνούν απευθείας με κάποιο τομέα εκτός του ΑΣ τους. Η επιλογή αυτή είναι απόλυτα λογική καθώς μόνο αυτοί μπορούν να παρατηρήσουν την συμπεριφορά τομέων που ανήκουν σε άλλα ΑΣ. Η μετρική αυτή που υπολογίζουν όμως δεν αφορά μόνο τον τομέα που την υπολογίζει αλλά ολόκληρο το ΑΣ εφόσον η απόφαση για βοήθεια ή μη είναι κοινή. Τέλος αφού η απόφαση είναι κοινή για όλο το ΑΣ το όριο πάνω από το οποίο το συνεργατικό επίπεδο ενός ΑΣ θεωρείται επαρκές για να του παρασχεθεί βοήθεια τίθεται από τον διαχειριστή του ΑΣ.

Σενάριο 4

Μένει ακόμα να ικανοποιηθούν οι αρχικές απαιτήσεις για την περίπτωση ενός αιτήματος που έρχεται από ένα ΑΣ Α σε ένα ΑΣ Β τα οποία δε είναι γειτονικά. Σε αυτήν την περίπτωση χρησιμοποιείται η έννοια της φήμης. Η έννοια της φήμης είναι στενά συνδεδεμένη με εκείνη της εμπιστοσύνης, αλλά είναι προφανές ότι υπάρχει μία σαφής και σημαντική διαφορά. Για τους σκοπούς της παρούσας μελέτης, μπορούμε να ορίσουμε τη φήμη σύμφωνα με το λεξικό της Οξφόρδης όπως γίνεται και στο [33].

> Φήμη είναι αυτό που λέγεται ή πιστεύεται για τον χαρακτήρα ή την συμπεριφορά ενός πράγματος ή προσώπου.

Ο ορισμός αυτός είναι σε συμφωνία με τον ορισμό των ερευνητών των κοινωνικών δικτύων που ορίζουν την φήμη ως μια ποσότητα που προέρχεται από το υποκείμενο κοινωνικό δίκτυο η οποία είναι ορατή σε παγκόσμιο επίπεδο σε όλα τα μέλη του δικτύου. Η διαφορά μεταξύ εμπιστοσύνης και φήμη είναι ότι τα συστήματα Εμπιστοσύνη παράγουν ένα αποτέλεσμα που αντανακλά την υποκειμενική άποψη μίας οντότητας για την αξιοπιστίας μίας άλλης, ενώ τα συστήματα φήμης παράγουν μιας αξιολόγηση, όπως συντίθεται από ολόκληρη την κοινότητα. Επιπλέον, η μεταβατικότητα είναι μία ρητή ιδιότητα σε συστήματα εμπιστοσύνης, ενώ τα συστήματα φήμης συνήθως λαμβάνουν την μεταβατικότητα μόνο εμμέσως υπόψιν.

Η έννοια της φήμης μπορεί να χρησιμοποιηθεί για την επίλυση του προβλήματός μας. Δηλαδή σε ένα αυτόνομο σύστημα Α που δεν συνορεύει με ένα άλλο Αυτόνομο σύστημα Β δεν υπάρχει κανένα domain που να έχει προσωπική γνώμη, (γνώμη δηλαδή που προέκυψε από τις προσωπικές του παρατηρήσεις) αλλά δημιουργεί μια άποψη για το επίπεδο συνεργατικότητας του Β συνθέτοντας τις απόψεις των άλλων αυτόνομων συστημάτων. Για τον σκοπό αυτό είναι απαραίτητο τα αυτόματα συστήματα να ανταλλάσσουν τις γνώμες τους. Έτσι κάθε φορά που ένα μήνυμα προωθείται ο αποστολέας εσωκλείνει τις τιμές του επιπέδου συνεργατιικότητας και τον αριθμό των συναλλαγών του με το domain του ΑΣ που του έστειλε το μήνυμα ανεξάρτητα από το εάν το δεύτερο είναι ο στόχος της επίθεσης ή όχι. Εάν ο αποδέκτης του μηνύματος έστω Α δεν έχει προσωπική γνώμη για το θύμα Β τότε αποθηκεύει την τιμή του επιπέδου συνεργατικότητας που αναφέρει ο γείτονας του θύματος εφόσον ο ίδιος ή το προηγούμενο ΑΣ στο μονοπάτι εμπιστεύεται τον γείτονα αυτόν. Τελικά ο Α συνδυάζεται με την παλαιότερη γνώμη που είχε με όμοιο τρόπο για τον Β με αυτήν που αποθήκευσε μόλις με χρήση του σταθμισμένου μέσου.

$$Cop\_level_{new} := \frac{Cop\_level\,[B]_{old} * N_{old} + Cop\_level\,[B]_{neighbor} * M_{neighbor}}{M_{neighbor} + N_{old}} \quad (5.3)$$

$$N_{new} = M_{neighnor} + N_{old} \quad (5.4)$$

Η απόφαση ωστόσο για τον αν ο Α θα συνδράμει στην αντιμετώπιση της επίθεσης του Β δεν θα έπρεπε να έχει να κάνει μόνο με τον Β εφόσον τα δύο ΑΣ απέχουν περισσότερο από ένα βήμα μεταξύ τους. Αυτό συμβαίνει διότι εάν ο Α αποφασίσει να εγκαταστήσει

κανόνες και το ένα ΑΣ ανάμεσα στους Α και Β έστω Γ τους έχει εγκαταστήσει ήδη τότε πρακτικά επωφελείται άμεσα το Γ διότι οι δικοί του κανόνες θα λήξουν. Συνεπώς για την λήψη της απόφασης λαμβάνεται υπόψιν και το επίπεδο συνεργατικότητας του ΑΣ που προώθησε το μήνυμα,του ΑΣ δηλαδή που είναι πιο μπροστά στο μονοπάτι από το θύμα στην πηγή.

Όταν τελικά το Α κληθεί να αποφασίσει αν θα συνεισφέρει ή όχι στην αντιμετώπιση της επίθεσης που δέχεται ένας domain που βρίσκεται σε ΑΣ το οποίο δεν συνορεύει με το δικό του ΑΣ, θα δει την τιμή εμπιστευτικότητας του αποστολέα όσο και αυτή που έχει συνθέσει μέσω της φήμης του ΑΣ -θύματος. Εάν το μεγαλύτερο από του αυτά ικανοποιεί τα κριτήρια συνεργασιμότητας του Α τότε αυτός θα συνεργάζεται. Παρακάτω δίνεται ο κώδικας που τρέχει για την ανανέωση του επιπέδου συνεργατικότητας τόσο για κάποιον γειτονικό όσο και για απομεμακρυσμένο τομέα.

```
public class nonadjacent {
  private double cop_level;
  private int id;
  private int N;
  private isp me;
  private int port;
  private void updateNonAdjacent(ArrayList<opinion> opinions)
  {
    int newN=0;
    double newCop=0;
    if (opinions.size()> 3)
    {
      if (opinions.get(2).cop_level>me.threshold_clients)
      {
        newCop=opinions.get(1).cop_level;
        newN=opinions.get(1).transactions;
      }
    }
    else
    {
      if (me.adjacents.contains(opinions.get(1).id))
      {
        adjacent a= me.adjacents.get(opinions.get(1).id);
        if (a.cop_level>me.threshold_clients)
        {
          newCop=opinions.get(1).cop_level;
          newN=opinions.get(1).transactions;
        }
      }
    }
    N+=newN;
    cop_level= (cop_level*N + newCop*newN )/N;

  }
}
public class adjacent {
    public double cop_level;
    private int id;
    private double a;
```

```
    private double b;
    private double u;
    private int port;
    private adjacent(int id, int port)
    {
      this.id=id;
      this.a=1;
      this.b=1;
      this.cop_level=0.5;
      this.u=0.99;
      this.port=port;
    }
    private void updateAdjacent(boolean helped)
    {
      int s=1;
      if (helped) s=0;
      a = u*a + s;
       b = u*b + (1-s);
       cop_level= (a+b)/ 2.0;
    }
}
```

Κεφάλαιο 6

Πειραματατική Διαδικασία - Συμπεράσματα

6.1 Αρχές Προσομοίωσης

Για την αξιολόγηση της αποτελεσματικότητας της ενδεχόμενης εφαρμογής ενός τέτοιου συστήματος υπό πραγματικές συνθήκες διαδικτύου πραγματοποιήθηκαν μία σειρά από προσομοιώσεις. Για τις ανάγκες αυτών μια κατανεμημένη επίθεση άρνησης υπηρεσίας προσομοιώνεται με ένα δέντρο του οποίου οι κόμβοι είναι αυτόματα συστήματα. Η ρίζα του δέντρου είναι το ΑΣ στο οποίο βρίσκεται το domain που δέχεται επίθεση. Τα φύλλα είναι τα Αυτόματα συστήματα από τα οποία ξεκινάει η επίθεση, δηλαδή είναι τα ΑΣ που αποτελούν την φυσική τοποθεσία των μηχανημάτων στα οποία ο επιτιθέμενος έχει εγκαταστήσει κάποιου είδους κακόβουλο λογισμικό το οποίο στέλνει κίνηση στο θύμα. Η κίνηση αυτή φτάνει στο θύμα μέσω του μοναδικού μονοπατιού από το αντίστοιχο φύλλο στη ρίζα. Το δέντρο σε κάθε επίθεση είναι ένα μέρος του γράφου που αναπαριστά την τοπολογία του συνόλου των Αυτόνομων Συστημάτων στο διαδίκτυο.

6.2 Τοπολογία και Επιθέσεις

Ο εν λόγω γράφος δημιουργήθηκε με βάση μία τοπολογία του διαδικτύου σε επίπεδο Αυτόνομων Συστημάτων. Η τοπολογία αυτή προέρχεται από τα δημοσιευμένα δεδομένα της Caida. Η Caida ή το Κέντρο Εφαρμοσμένης Ανάλυσης Δεδομένων Διαδικτύου είναι μια συλλογική προσπάθεια μεταξύ των οργανώσεων που δραστηριοποιούνται στον εμπορικό, κυβερνητικό και ερευνητικό τομέα με στόχο την διατήρηση μίας ισχυρής, κλιμακούμενης παγκόσμιας υποδομής του διαδικτύου. Τα δεδομένα αυτά είναι διαθέσιμα από το 2004, με ένα νέο αρχείο ανά εβδομάδα από το 2006 και ένα ανά μήνα στα προηγούμενα έτη. Κάθε αρχείο περιέχει ένα πλήρες γράφημα Αυτόνομων Συστημάτων που προέρχονται από στιγμιότυπα των πινάκων δρομολόγησης BGP που λαμβάνονται ανά διαστήματα οκτώ ωρών για μια περίοδο πέντε ημερών. Οι σχέσεις μεταξύ ΑΣ που είναι διαθέσιμες είναι οι εξής: πελάτης- πάροχος (και πάροχος - πελάτης προς την αντίθετη κατεύθυνση), ομότιμος-ομότιμος (peer to peer), και αδελφός- αδελφός (που ανήκουν στον ίδιο οργανισμό).Η γενική διαδικασία για τη δημιουργία ενός τέτοιου αρχείου είναι η εξής:

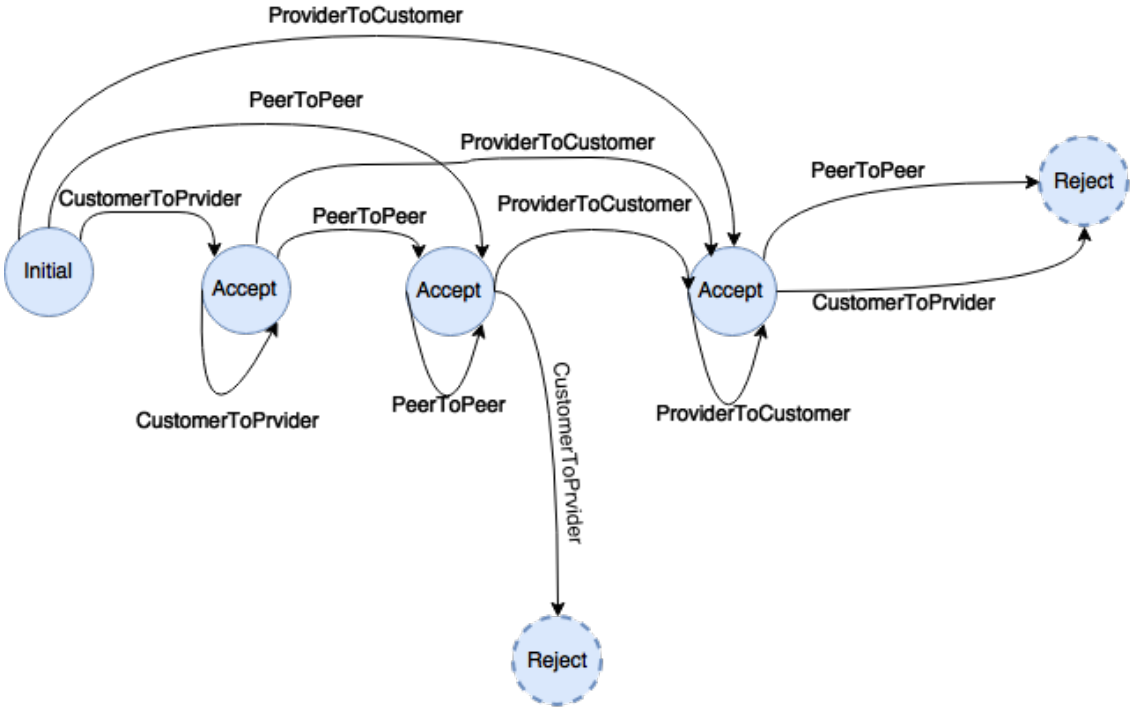
1. Απόσπαση όλων των συνδέσεων από τα στιγμιότυπα RouteViews.
2. Εξαγωγή συμπερασμάτων για σχέσεις πελάτη - παρόχου, και σημείωση αντίστοιχων συνδέσμων.
3. Εξαγωγή συμπεράσματων για σχέσεις ομότιμος-ομότιμος και σημείωση αντίστοιχων συνδέσμων, ενδεχομένως αντικαθιστώντας σχέσεις πελάτη -παρόχου που είχαν συναχθεί στο βήμα 2.
4. Καθορισμός και διόρθωση με Heuristic μεθόδους ύποπτων σχέσεων (π.χ., ένα χαμηλού βαθμού ΑΣ ενεργεί ως πάροχος σε υψηλό βαθμού ΑΣ).
5. Εξαγωγή συμπεράσματα για σχέσεις αδελφών ΑΣ από ερωτήματα WHOIS, και σημείωση αντίστοιχων συνδέσμων, ενδεχομένως αντικαθιστώντας σχέσεις που είχαν συναχθεί προηγουμένως .

Οι λεπτομέρειες των αλγορίθμων που χρησιμοποιούνται ώστε να συναχθούν οι σχέσεις αυτές αναφέρονται ενδελεχώς στα ακόλουθα [34] [35]

Το dataset της CAIDA [36] αναφέρεται τόσο στην τοπολογία όσο και στις επιχειρηματικές σχέσεις των ΑΣ, είναι δηλαδή σχετικό και με τις δύο πτυχές της δομής του Διαδικτύου την τεχνική και και την οικονομική. Χρησιμοποιήθηκε αυτό το dataset αφενός για να δουλέψουμε πάνω σε μία πραγματική τοπολογία και αφετέρου επειδή περιέχει σχέσεις μεταξύ των ΑΣ οι οποίες καθορίζουν τις πολιτικές δρομολόγησης και εισάγουν ένα μη τετριμμένο σύνολο περιορισμών σε διαδρομές στις οποίες τα πακέτα μπορούν να ρέουν στο διαδίκτυο. Χρησιμοποιήσαμε αυτά τα δεδομένα για να παράγουμε ρεαλιστικά μονοπάτια κίνησης που συνιστούν την επίθεση. Έτσι κάθε επίθεση συνίσταται από μονοπάτια από το ΑΣ - πηγή στο ΑΣ- θύμα τα οποία δημιουργήθηκαν εφαρμόζοντας τον αλγόριθμο που παρουσιάζει ο Gao [37] σχετικά με τα επιτρεπτά μονοπάτια. Ο αλγόριθμος αυτός, χρησιμοποιήθηκε ως επέκταση του αλγορίθμου του BFS για την δημιουργία των δέντρων που αναπαριστούν επιθέσεις. Δωθήσας της ρίζας, για την είσοδο κάθε νέου κόμβου στο BFS δέντρο απαιτείται πέρα από το να μην έχει συμπεριληφθεί ήδη και να μην δημιουργεί κύκλο στο υπάρχον δέντρο, η ακμή που θα προστεθεί να μην χαλάει την ορθότητα του μονοπατιού και το μέγιστο επιτρεπτό ύψος δέντρου όπως αυτό ορίστηκε στις αρχικές απαιτήσεις. Ορθά μονοπάτια είναι αυτά, τα οποία μπορεί να ακολουθήσει μια ροή πακέτων στο διαδίκτυο. Ισοδύναμα είναι τα μονοπάτια που μπορούν να αναπαρασταθούν ως συμβολοσειρές που αναγνωρίζει το αυτόματο του Σχήματος 6.1. Δημιουργήθηκε ένα Deterministic Finite Automaton (DFA) [38] για την καλύτερη κατανόηση της διαδικασίας που ακολουθούμε και είναι πιστή υλοποίηση του αλγορίθμου που παρουσιάζεται [37].

Οι δύο βασικότερες συναρτήσεις για την παραγωγή ενός δέντρου επίθεσης είναι οι παρακάτω. Η συνάρτηση **getAllPaths** επιστρέφει το δέντρο μίας επίθεσης πάνω στην τοπολογία δηλαδή μια ομάδα από μονοπάτια - ακολουθίες από ΑΣ που απαρτίζουν την επίθεση. Η επίθεση αυτή είναι παραμετροποιήσιμη ως προς το αυτόματο σύστημα που δέχεται επίθεση, το μέγιστο μονοπάτι από αυτόματα συστήματα που μπορεί να διατρέχει μια ροή κακόβουλων πακέτων που συμμετέχουν στην επίθεση, την τοπολογία πάνω στην όποια θα δημιουργηθεί η επίθεση και τις οικονομικές σχέσεις μεταξύ αυτών των αυτόνομων συστημάτων. Ο κώδικας δίνεται παρακάτω:

```
public static Hashtable <Integer, ArrayList<Integer>> getAllPaths( Hashtable<Integer, ArrayList<Integer>> edges, int root, int dep,
      Hashtable<String, Integer> relation,int[] Level, ArrayList<Integer> queue2, Hashtable<Integer, ISP> iSPs )
  {
    for (int i=0; i<6; i++){Level[i]=0;}
    Hashtable<Integer,ArrayList<Integer> > bfstree= new Hashtable <Integer, ArrayList<Integer>>();
    Hashtable<Integer, Integer> indexes= new Hashtable<Integer, Integer>(); // Key= ASN , Boolean= visited==added in Queue==
```



Σχήμα 6.1: *DFA Επιτρεπτών Μονοπατιών*

```
ArrayList<Tupla> queue = new ArrayList<Tupla>();
queue.add(new Tupla(root, 1));
queue2.add(root);
Level[1]=1;
indexes.put(root, 1);
while (!queue.isEmpty())
{
   Tupla temp=queue.get(0); //root & key
   queue.remove(0);
   Integer prev=indexes.get(temp.a);
   ArrayList<Integer> row = new ArrayList<Integer>();
   if ((temp.b+1)<=dep)
   {
      if (edges.containsKey(temp.a))
      {
         for(int j=0 ; j<(edges.get(temp.a).size()); j++)
         {
            Integer node = edges.get(temp.a).get(j); //node that could be added to bfs tree
            {
               if (!indexes.containsKey(node))
               {
                  int tt=isvalid(Integer.toString(temp.a),node.toString() , prev, relation,temp.b+1 ,dep, iSPs);
                  if (tt!=-2)// valid
                  {
                     queue.add(new Tupla(node, temp.b+1));
                     queue2.add(node);
                     Level[temp.b+1]++;
                     row.add(node);
                     indexes.put(node, tt);
                  }
               }
            }
         }
         //System.out.println("the node="+temp.a+ "has the egdes"+row.toString());
```

```
            bfstree.put(temp.a,row);
          }
        }
      }
      return bfstree ;
  }
```

Η συνάρτηση **isvalid()** ελέγχει κατά πόσο μια ακμή είναι επιτρεπτή με βάση το υπάρχον δέντρο δηλαδή εάν η ακολουθία που εκφράζει το μονοπάτι που δημιουργείται από τον κόμβο του θύματος στον κόμβο που επιχειρείται να μπει στο δέντρο, αναγνωρίζεται από το αυτόματο του Σχήματος 6.1.Ο κώδικας δίνεται παρακάτω:

```
static int isvalid(String temp, String node, Integer prev, Hashtable<String, Integer> relation, int j, int dep, Hashtable<Integer,
      ISP> iSPs)
  {
    String tupla = temp+node;
    String revtupla = node+temp;
    Integer cur = null;
    if (relation.containsKey(revtupla))
    {
      int i=relation.get(revtupla);
      if (i==-1) cur=1; else cur=i;
    }
    else if (relation.containsKey(tupla)) cur=relation.get(tupla);
    else System.out.println("den vrhka tpt");
    if(j<dep){
      if ((prev==1)&&(cur==0)) {return cur;} //CP-->PP
      else if ((prev==0)&&(cur==-1)) { return cur;}//PP-->PC
      else if ((prev==1)&&(cur==-1)){return cur;}//CP--> PC
      else if ((cur==prev)) {return cur;}
      else return -2;
    }
    else {
      if ((cur==-1)) return cur;
      else if ((cur==1) && iSPs.get(Integer.parseInt(node)).Customers.isEmpty()) return cur;
      else return -2;
    }
  }
```

6.3 Αξιολόγηση συστήματος ως προς την ταχύτητα αντιμετώπισης

Για τις ανάγκες της αξιολόγησης του συστήματος όσο αφορά στην ταχύτητα αντιμετώπισης της επίθεσης και κατ' επέκταση την δυνατότητα διασποράς των απαιτούμενων για την αντιμετώπιση της επίθεσης κανόνων, χρησιμοποιήθηκε ένα επιπλέον dataset που παρέχεται από την Caida και συγκεκριμένα το το CAIDA'DDoS Attack 2007' Dataset [39]. Το dataset αυτό περιέχει μόνο πακέτα (traces) κακόβουλων αποστολέων και καθόλου ομαλή

κίνηση. Για τον λόγο αυτό χρησιμοποιήσαμε ταυτόχρονα στις προσομοιώσεις και traces κίνησης από το δίκτυο του Εθνικού Μετσόβιου Πολυτεχνείου.

6.3.1 Θέματα Υλοποίησης

Η πειραματική μας διάταξη αποτελείται από τρία μηχανήματα. Το πρώτο αναπαράγει την παραπάνω κίνηση με χρήση της συνάρτησης tcpreplay δηλαδή στέλνει τα ίδια πακέτα ποιοτικά και ποσοτικά, με τον ίδιο ρυθμό που θα τα έστελναν όλα τα μηχανήματα που λειτούργησαν είτε ως πηγές της επίθεσης του 2007 είτε ως απλοί νόμιμοι χρήστες. Ουσιαστικά δηλαδή το πρώτο μηχάνημα προσομοιώνει όλους τους κόμβους που στέλνουν πακέτα στο ΑΣ που βρίσκεται ο στόχος. Η παραγόμενη αυτή κίνηση φτάνει στο δεύτερο μηχάνημα το οποίο εκπροσωπεί όλα τα δίκτυα ανάμεσα στις πήγες και το ΑΣ του στόχου της επίθεσης. Πρόκειται ουσιαστικά για ένα μεταγωγέα που υποστηρίζει OpenFlow στον οποίο συνοψίζονται και εγκαθίστανται όλοι οι κανόνες που ζήτησε το ΑΣ θύμα και θα ήταν στην πράξη εγκατεστημένα σε όλο το ενδιάμεσο δίκτυο. Συνεπώς στην αρχή ο μεταγωγέας έχει εγκατεστημένους OpenFlow κανόνες οι οποίοι κατευθύνουν όλη την κίνηση που προορίζεται για το ΑΣ του στόχου σε αυτό, συμπεριλαμβανομένης και της κίνησης της επίθεσης. Αργότερα καθώς η προσομοίωση συνεχίζεται (δηλαδή το SDN domain ανιχνεύει την επίθεση και στέλνει εκκλήσεις για συνεργασία) κάποια αυτόματα συστήματα εγκαθιστούν τους κανόνες που ζήτησε ο στόχος και έτσι τα αντίστοιχα πακέτα δεν φτάνουν σε αυτόν. Οι εν λόγω OF κανόνες εγκαθίστανται με όμοιο τρόπο στο δεύτερο μηχάνημα με αποτέλεσμα στο μηχάνημα που αντιπροσωπεύει το SDN domain του θύματος να φτάνουν μόνο τα πακέτα που θα έφταναν υπό πραγματικές συνθήκες. Ο SDN τομέας που δέχεται επίθεση εκπροσωπείται από ένα τρίτο μηχάνημα. Στο μηχάνημα αυτό μετράμε τόσο τα πακέτα σαν συνάρτηση χρόνου (packets / sec) όσο και την ταχύτητα διέλευσης (σε bps) .

Με τον τρόπο αυτό όλη η περιπλοκότητα του πειράματος μεταφέρεται στον υπολογισμό των κανόνων και της χρονικής στιγμής που αυτοί εγκαθίστανται στο ενδιάμεσο δίκτυο. Ο υπολογισμός αυτός γίνεται σε ξεχωριστό τοπικό μηχάνημα σε κώδικα java ο οποίος επικοινωνεί μέσω socket με το απομακρυσμένο μηχάνημα του μεταγωγέα (δεύτερο μηχάνημα) για να τον ειδοποιεί κάθε φορά που εγκαθίσταται ένας κανόνας σε κάποιο domain

σύμφωνα με την προσομοίωση. Ο κώδικας που τρέχει στο μηχάνημα του μεταγωγέα (2ο μηχάνημα) χρησιμοποιεί ένα στιγμιότυπο της κλάσης Runtime της Java που επιτρέπει στον κώδικα να διασυνδέεται άμεσα με το περιβάλλον στο οποίο εκτελείται και να εγκαθιστά ισοδύναμους OpenFlow κανόνες στον πίνακα ροής του. Συνεπώς, η κίνηση που βλέπει το domain που είναι στόχος (τρίτο μηχάνημα) είναι η σωστή δηλαδή αυτή που θα έβλεπε αν ήταν στο πραγματικό διαδίκτυο και συνέβαινε η ίδια επίθεση και ένα μέρος των ΑΣ χρησιμοποιούσαν το σύστημα μας για να αντιμετωπίσουν την επίθεση.

Ο κώδικας που τρέχει στην μεριά του μεταγωγέα δηλαδή στο απομακρυσμένο μηχάνημα 2 δίνεται παρακάτω.

```
public class InsjectorFinal implements Runnable {
  static int port=9897;

  public static void main(String[] args)
  {
    InsjectorFinal in= new InsjectorFinal();
    (new Thread(in)).start();
  }
  public void run()
  {
    String IP;
    System.out.println("Injector waits for delays=:");
    String s;
    ServerSocket server = null;
    try
    {
      server = new ServerSocket( port);
    } catch (IOException e) {
      e.printStackTrace();
    }
    while(true)
    {
      Socket socket = null;
      try {
        socket = server.accept();
      } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
      System.out.println("Accepted Connection for delays");
      //read from socket to ObjectInputStream object
      ObjectInputStream ois = null;
      try {
        ois = new ObjectInputStream(socket.getInputStream());
      } catch (IOException e) {
        e.printStackTrace();
      }
      //convert ObjectInputStream object to String
      ArrayList<Integer> delays = null;
      try {
        try {
          ArrayList<Integer> readObject = (ArrayList<Integer>) ois.readObject();
          delays = readObject;
        } catch (ClassNotFoundException e) {
          e.printStackTrace();
```

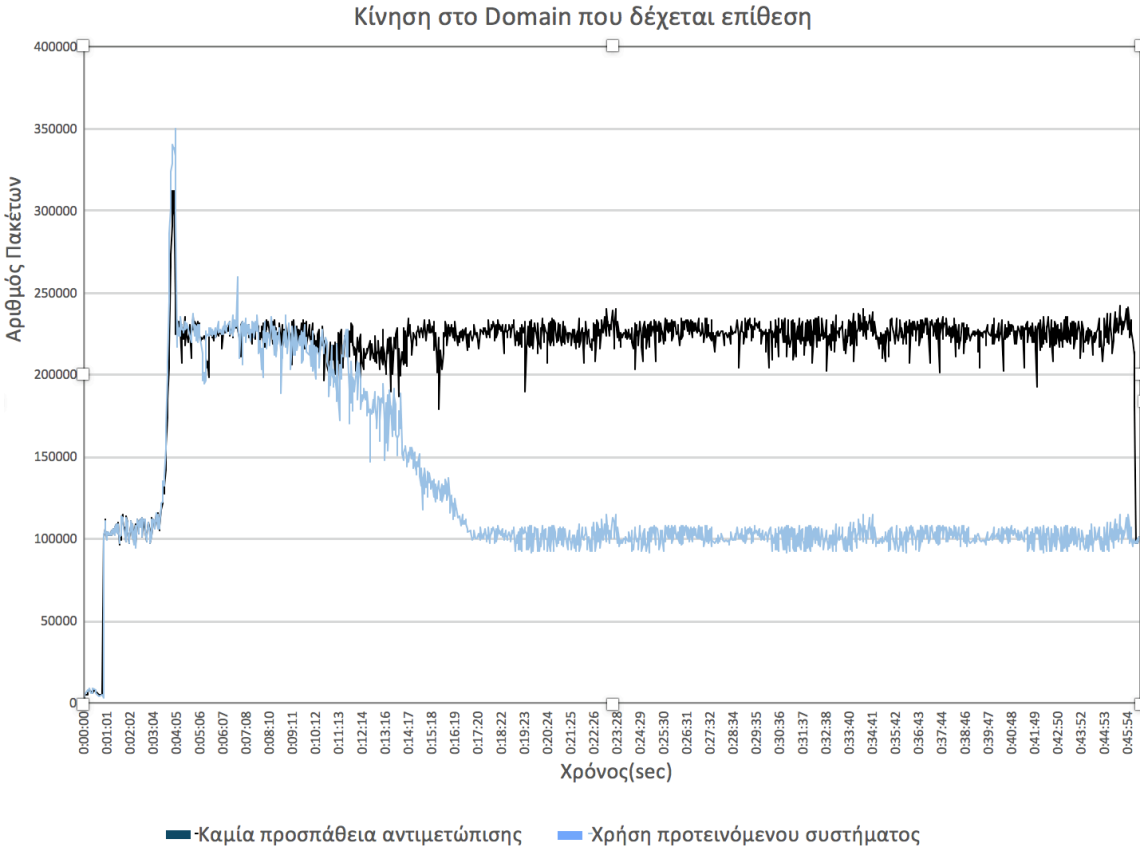
```
    }
}catch (IOException ee) {
    // TODO Auto-generated catch block
    ee.printStackTrace();
}
try {
    socket.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
ArrayList<ArrayList<String>> attackIP= new ArrayList<ArrayList<String>>();
for (int k =0; k<delays.size(); k++)
{
    try {
        try {
            ArrayList<String> line = new ArrayList<String>();
            line = (ArrayList<String>) ois.readObject();
            attackIP.add(line);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    catch (IOException ee) {
        ee.printStackTrace();
    }
}
double cur= 0.0;
System.out.println("________New message__message.size___" + attackIP.size()+"____first____"+attackIP.get(0).size()+
        "____2o____"+attackIP.get(1).size()+ "____30____"+attackIP.get(2).size());
for(int i = 0; i<delays.size(); i++)
{
    if ( delays.get(i)>cur)
    {
        try {
            System.out.println("will wait for "+(delays.get(i)-cur));
            Thread.sleep((long) ((delays.get(i)-cur)*1000)); //1000 milliseconds is one second.
        } catch(InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
        cur=delays.get(i);
    }
    int j=0;
    for ( j=0; j<attackIP.get(i).size();j++)
    {
        IP=attackIP.get(i).get(j);

        try {
            Process p = Runtime.getRuntime().exec("ovs-ofctl add-flow br0 priority=100,in_port=2,ip,nw_src="+IP+",actions=");

            p.waitFor();
            BufferedReader stdInput = new BufferedReader(new
                 InputStreamReader(p.getInputStream()));

            while (( s = stdInput.readLine()) != null)
            {
               System.out.println(s);
            }
        } catch (IOException e1) {
            e1.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

```
        System.out.println("________In stalles ______"+j);
      }
    }
  }
```

Ο κώδικας που τρέχει τοπικά για την αποστολή των δεδομένων της προσομοίωσης δίνεται παρακάτω.

```
ublic class Main2 {
  static int proc=5;
  static int rules;
  public static void m(ArrayList<Double> arr) throws FileNotFoundException, IOException{
    double timestamp;
    String src;
    String line;
    Hashtable<String, Tripletes> attack = new Hashtable<String, Tripletes>();
    ArrayList<String> attackAr = new ArrayList<>();
    try
    {
      BufferedReader br = new BufferedReader(new FileReader("/Users/mariaapostolake/maria.txt"));
      //BufferedReader br = new BufferedReader(new FileReader("/Users/mariaapostolake/Desktop/topo/ddos.txt"));
      while ((line = br.readLine()) != null)
      {
        String[] parts = line.split(" ");
        timestamp=( Double.parseDouble(parts[1]));
        src=(parts[2]);// id to
        Tripletes t;
        if ((t=(Tripletes) attack.get(src))!=null)
        {
          t.end=timestamp;
          t.counter++;
        }
        else
        {
          Tripletes tt= new Tripletes(src, timestamp);
          attack.put(src, tt);
          attackAr.add(src);
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    int step =(attackAr.size()/proc);
    ArrayList<ArrayList<String>> com = new ArrayList<ArrayList<String>>();
    for (int i=0; i<attackAr.size(); i+=step)
    {
      ArrayList<String> comi = new ArrayList<String>();
      if((i+2*step)>(attackAr.size()-1)) step=step+ (attackAr.size() % proc);
      for (int j=0; j<step; j++)
      {
        if ((i+j)>=attackAr.size()){break;}
        comi.add(attackAr.get(i+j));
      }
      com.add(comi);
      System.out.println(""+comi.size());
    }
    while (true)
    {
      Scanner input = new Scanner(System.in);
      int numOfStudents = input.nextInt();
      System.out.println("OK");
```

```
        Socket socket = null;
        ObjectOutputStream oos = null;
        socket = new Socket("147.102.13.241", 9897);
        oos = new ObjectOutputStream(socket.getOutputStream());
        oos.writeObject(arr);
        for (int k=0; k<com.size(); k++)
        {
          //oos =new ObjectOutputStream(socket.getOutputStream());
          oos.writeObject(com.get(k));
        }
        oos.close();
    }
}
```



Σχήμα 6.2: *Πακέτα που φτάνουν στο Domain που δέχεται επίθεση*

6.3.2 Αποτελέσματα

Στο Σχήμα 6.2 μπορούμε να δούμε με κόκκινο χρώμα την κίνηση που βλέπει η υπηρεσία - εξυπηρετητής που δέχεται επίθεση και μετρήθηκε στο μηχάνημα 3 σε αριθμό πακέτων σαν συνάρτηση του χρόνου προτού γίνει οποιαδήποτε προσπάθεια αντιμετώπισης της επί-

θεσης. Με μπλε χρώμα φαίνεται στην συνέχεια η κίνηση στην περίπτωση που όλα τα ΑΣ στο δέντρο της επίθεσης συμμετέχουν στην διαδικασία αντιμετώπισης και εγκαθιστούν κανόνες. Σε κάθε περίπτωση και για λόγους καλύτερης κατανόησης το ΑΣ στο οποίο ανήκει το θύμα δεν εγκαθιστά κανόνες. Είναι προφανές ότι ο διαμοιρασμός των κανόνων στα ΑΣ προς τα μονοπάτια της επίθεσης είναι πολύ αποτελεσματικός και γρήγορος. Υπολογίζεται θεωρητικά άλλωστε ότι ο χρόνος αντιμετώπισης, δηλαδή το χρονικό διάστημα ανάμεσα στην χρονική στιγμή που το domain στόχος ανιχνεύει την επίθεση και στέλνει αίτημα για βοήθεια μέχρι που η περιγραφόμενη ροή πακέτων μπλοκαριστεί στο ΑΣ που συνορεύει με το ΑΣ του θύματος είναι ίσο με το χρόνο μονής διαδρομής ενός πακέτου μεταξύ των δύο ΑΣ αυξανόμενο κατά ένα μικρό χρονικό διάστημα για την επεξεργασία του αιτήματος. Παρατηρήθηκε επίσης ότι η άρνηση για βοήθεια από το 1/5 των ΑΣ δεν επηρεάζει σημαντικά την επίδοση του συστήματος όσο αφορά το θύμα. Επηρεάζει προφανώς τον αριθμό των κανόνων που αναγκάζεται να εγκαταστήσει το θύμα καθώς επίσης και τον χρόνο που τα άλλα ΑΣ κρατούν τους κανόνες στον πίνακα ροής τους αλλά αυτό θα εξεταστεί ξεχωριστά παρακάτω.

6.4 Αξιολόγηση συστήματος ως προς την κατανομή πόρων στον χώρο

Στο δεύτερο πείραμα επικεντρωθήκαμε στην αξιολόγησης του συστήματος όσο αφορά στην διαχείριση πόρων του τομέα που δέχεται επίθεση. Για τις ανάγκες αυτής της μελέτης, παρακολουθήσαμε την μεταβολή του αριθμού των κανόνων του πίνακα ροής που απαιτούνται ώστε να επιτευχθεί επιλεκτικό μπλοκάρισμα μόνο της κακόβουλης κίνηση, ανάλογα με την συμπεριφορά των υπόλοιπον Αυτόνομων Συστημάτων. Η κακόβουλη κίνηση αφορά την επίθεση της CAIDA και το ίδιο ΑΣ - στόχο.

6.4.1 Θέματα Υλοποίησης

Το dataset αποτελείται από 9312 μοναδικές IP διευθύνσεις πηγής οι οποίες και μπορούν να ομαδοποιηθούν σε 4186 /16 υπο-δίκτυα. Για τις ανάγκες της προσομοίωσης μας έπρεπε

να διαμοιράσουμε αυτές τις πηγές δηλαδή τα ”μολυσμένα” μηχανήματα στην τοπολογία μας. Δεν μπορούμε να αναπαράγουμε απόλυτα την τοπολογία της επίθεσης καθώς οι διευθύνσεις IP που δίνονται από το dataset είναι ανωνυμοποιημένες. Συνεπώς, βασιστήκαμε στην υπόθεση ότι κάθε κλάση B IP διευθύνσεων πηγής προέρχεται από ένα ΑΣ. Κατασκευάσαμε ένα δέντρο για την αναπαράσταση της επίθεσης με την μέθοδο που περιγράψαμε παραπάνω θέτοντας τις εξής παραμέτρους. (α) τυχαία επιλεγμένη ρίζα - ΑΣ που δέχεται επίθεση (ίδια με το προηγούμενο πείραμα) (β) μέγιστη απόσταση πηγών από την ρίζα 6, λαμβάνοντας υπόψιν την παρατήρηση ότι δύο οποιαδήποτε αυτόματα συστήματα στο διαδίκτυο απέχουν 6 βήματα. [40] (γ)συνολικός αριθμός των ΑΣ που είναι πηγές είναι ίσος με τον αριθμό των διαφορετικών /16 υποδικτύων. Η προσομοίωση της επίθεσης αυτής έγινε με την βοήθεια κάποιων επιπλέον συναρτήσεων που παρουσιάζονται συνοπτικά παρακάτω:

Το δέντρο που δημιουργεί η getallpaths είναι πολύ πυκνό ως αποτέλεσμα του πυκνού γράφου της τοπολογίας και κατ επέκταση ως αποτέλεσμα των πολύπλοκων σχέσεων των ΑΣ στο διαδίκτυο. Συνεπώς, τα φύλλα του δέντρου που κατασκευάζονται ξεπερνούν κατά πολύ τον επιθυμητό αριθμό πηγών. Για τον λόγο αυτό χρησιμοποιείται η συνάρτηση **deletesome()** η οποία διαγράφει τυχαία κλαδιά από το δέντρο που δημιούργησε η gellAllPaths μέχρι να φτάσουμε σε συγκεκριμένο αριθμό φύλλων. Η συνάρτηση δημιουργήθηκε για να επιτρέψει την διασπορά δεδομένου αριθμό αυτόματων συστημάτων που συμμετείχαν ώς πηγές στην επίθεση που προσομοιώνεται, στο αρχικό δέντρο για την δημιουργία του υποδέντρου της επίθεσης.

```
public static void deletesome(ArrayList<ArrayList<Integer>> all, int limit)
  {
     while(all.size()>limit)
     {
        int max =all.size()-1;
        int min =1;
        Random rand = new Random();
        int del = rand.nextInt((max - min) + 1) + min;
        all.remove(del);
     }
  }
```

Η συνάρτηση **level()** χρησιμοποιείται για να βρεθεί ο αριθμός των διαθέσιμων κανόνων ανά επίπεδο δέντρου επίθεσης. Παραμετροποιείται ως προς τον αριθμό των διαθέσιμων κανόνων ανά ΑΣ. Ταυτόχρονα η συνάρτηση λαμβάνει υπόψιν το αν αυτά τα AS συμμε-

τέχουν ή όχι στην διαδικασία.

```
public static int[] level (ArrayList<ArrayList<Integer>> paths, Hashtable<Integer, Integer> maliscious)
    {
      int [] last= new int[10];
      int [] levels= new int[10];
      for (int i=0; i<10; i++) {levels[i]=0;last[i]=-1;}
      for (int i=0; i<paths.size(); i++)
      {
        for(int j=0; j<paths.get(i).size(); j++)
        {

          if (last[j]!=paths.get(i).get(j))
          {
            last[j]=paths.get(i).get(j);
            if (!(maliscious.containsKey(last[j])))levels[j]++;
          }
        }
      }
      return levels;
    }
```

Παρακάτω δίνεται η συνάρτηση που συνδυάζει τα παραπάνω για τις ανάγκες της προσομοίωσης. Η συνάρτηση αυτή διαβάζει το αρχείο της CAIDA και φτιάχνει ένα στιγμιότυπο της κλάση ISP (Autonomus System) για κάθε αυτόνομο σύστημα που αναφέρεται και ενσωματώνει την πληροφορία για τις σχέσεις μεταξύ τους στο πρόγραμμα. Πιο αναλυτικά κάθε AS (κάθε στιγμιότυπο της κλάσης AS) περιέχει στο τέλος του διαβάσματος τρείς δομές hash με τα ΑΣ που είναι πελάτες του , πάροχοι και ομότιμοι. Τέλος δημιουργεί μία hash δομή για τις ακμές του γράφου που ενσωματώνουν και την σχέση που η κάθε μία αναπαριστά. Στην συνέχεια δημιουργείται η επίθεση και καταμετρούνται οι κανόνες που χρειάζονται σε κάθε περίπτωση.

```
public class FirstExperiment {
    public static void main(String[] args) throws FileNotFoundException, UnsupportedEncodingException, InterruptedException {
      System.out.println("FIle opened");
      Hashtable<String, Integer> relation = new Hashtable<String, Integer>();
      Hashtable<Integer,ArrayList<Integer>> edges = new Hashtable<Integer,ArrayList<Integer>>();
      int root=7151;
      int malNum=10000;
      Hashtable<Integer, Integer> Maliscious= new Hashtable<Integer, Integer>();
      Hashtable<Integer, ISP> ISPs = new Hashtable<Integer, ISP> ();
      PrintWriter writerSome = new PrintWriter("/Users/mariaapostolake/Desktop/topo/rulesTobeinstalledWhenSomeHelp.txt", "UTF-8");
      PrintWriter writerAll = new PrintWriter("/Users/mariaapostolake/Desktop/topo/rulesTobeinstalledWhenSAllHelp.txt", "UTF-8");
      PrintWriter writerMal = new PrintWriter("/Users/mariaapostolake/Desktop/topo/Maliscious.txt","UTF-8");
      try
      {
        BufferedReader br = new BufferedReader(new FileReader("/Users/mariaapostolake/Desktop/topo/as-rel2015.txt"));

        String line;
        while ((line = br.readLine()) != null)
        {
          String[] parts = line.split(" ");
          String tupla = parts[0]+parts[1];
          relation.put(tupla, (Integer.parseInt(parts[2])));
```

```
        if (!(ISPs.containsKey(Integer.parseInt(parts[1]))))
        {
          ISP isp = new ISP(Integer.parseInt(parts[1]));
          ISPs.put(Integer.parseInt(parts[1]), isp);
        }
        if (! ISPs.containsKey(Integer.parseInt(parts[0])))
        {
          ISP isp = new ISP(Integer.parseInt(parts[0]));
          ISPs.put(Integer.parseInt(parts[0]), isp);
        }
        if((parts[2]).equals("-1"))
        {
          ISP isp = ISPs.get(Integer.parseInt(parts[0]));
          isp.addclient(parts[1]);
          isp = ISPs.get(Integer.parseInt(parts[1]));
          isp.addProvider(parts[0]);
        }
        else
        {
          ISP isp = ISPs.get(Integer.parseInt(parts[1]));
          isp.addPeer(parts[0]);
          isp = ISPs.get(Integer.parseInt(parts[0]));
          isp.addPeer(parts[1]);
        }
      }
    }catch (Exception e){e.printStackTrace();}
    Set<Integer> keys = ISPs.keySet();
    System.out.println("size "+keys.size());
    for(Integer i:keys)
    {
      ISP isp= ISPs.get(i);
      if ((Math.random() < 0.4)&&(malNum>0))
      {
        if (!(i==root) && ISPs.containsKey(i))
        {
          ISPs.get(i).helps=false;
          Maliscious.put(i, 1);
          malNum--;
           writerMal.println(i);
        }
      }
      {
        ArrayList<Integer> temp = new ArrayList<Integer>();
        Set<String> clients = isp.Customers.keySet();
        for(String j:clients)
        {
          temp.add(Integer.parseInt(j));
        }
        Set<String> providers = isp.Providers.keySet();
        for(String j:providers)
        {
          temp.add(Integer.parseInt(j));
        }
        Set<String> peers = isp.Peers.keySet();
        for(String j:peers)
        {
          temp.add(Integer.parseInt(j));
        }
        edges.put(isp.id,temp);
      }
    }
    writerMal.close();
    System.out.println( "Nodes are"+ ISPs.size());
    Hashtable<Integer,ArrayList<Integer>> paths = new Hashtable<Integer,ArrayList<Integer>>();
```

```
  System.out.println("Root is "+ root);
  int depth=6;
  int[] Level= new int[10];
ArrayList<Integer> queue= new ArrayList<Integer>();
  paths=getAllPaths(edges,root,depth, relation,Level);
  System.out.println( "tree created"+paths.size());
  ArrayList<ArrayList<Integer>> pathss = new  ArrayList<ArrayList<Integer>> ();
  pathss=makeallpaths(depth, 0, paths, root, -5);//????
  for (int k=0; k<pathss.size(); k++)
  {
     Collections.reverse(pathss.get(k));
  }
  //System.out.println("paths after reverse created" +pathss.toString());
  ArrayList<ArrayList<Tripletes>> ips =Read.read();
  System.out.println("*****All subnets are****** "+ ips.size());
  deletesome(pathss, ips.size());
  System.out.println("paths after remove are "+ pathss.size() );
  Hashtable<String, Integer> keptIPs ;
  int c=findKept(pathss);
  System.out.println("C is +" + c);
  keptIPs=PutKeptInHash(c);
  System.out.println("Rules that won't be installed "+ keptIPs.size());
  int [] rules = new int[depth];
  Level=level(pathss,Maliscious);
  rules[0]=0;
  for (int i=1; i<depth; i++)
  {
     rules[i]=(300* Level[i])-(rules[i-1]);
     if (rules[i]<0) rules[i]=0;
     System.out.println("level "+i+"is "+Level[i]+ "new rules "+rules[i]);
  }
  int j=1;
  int rul=rules[1];
  boolean flag=true;
  int n=0;
  int k=1;
  while(flag)
  {
     flag=false;
     for(int i=0; i<ips.size(); i++)
     {
        if (ips.get(i).size()>0)
        {
           flag=true;
           Tripletes tt = ips.get(i).get(0);
           int temp=(int)(tt.start/3)*3+3+k ;
           int temp2=(int)(tt.start/3)*3+3;
           if ((temp<tt.end) && (!(keptIPs.containsKey(tt.src))))
              {
              writerSome.println(" "+tt.src+" "+tt.start+" "+temp);
              writerAll.println(" "+tt.src+" "+tt.start+" "+temp2);
              n++;
              }
           else
           {
              writerSome.println(" "+tt.src+" "+tt.start+" "+tt.end);
              writerAll.println(" "+tt.src+" "+tt.start+" "+temp2);
           }

           ips.get(i).remove(0);
           rul--;
        }
        if (rul<0){k=k+1; rul=rules[j++];}
     }
```

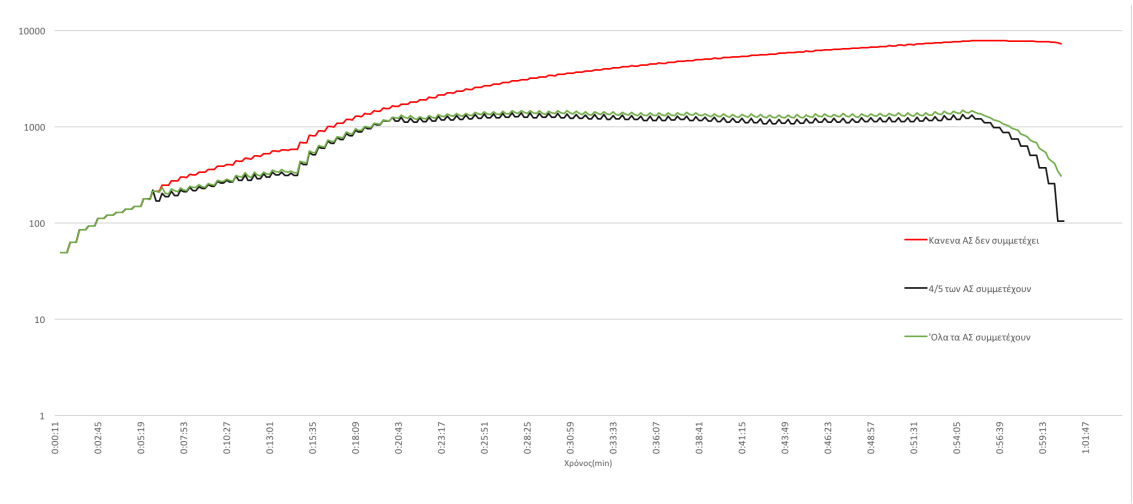
```
    }
    writerSome.close();
    writerAll.close();
    System.out.println("n="+n);
    Level=level(pathss, Maliscious);
    int [] rules2 = new int[depth];
    rules2[0]=0;
    for (int i=1; i<depth; i++)
    {
      rules2[i]=(200* Level[i])-(rules2[i-1]);
      //System.out.println("level "+i+"is "+Level[i]+ "new rules "+rules2[i]);
    }
  }
  private static Hashtable<String, Integer> PutKeptInHash(int c) {
    Hashtable< String, Integer> a = new Hashtable<>();
    int read=0;
    int writeinhash=0;
    int notwriteinhash=0;
    try
    {
      FileReader fl = new FileReader("/Users/mariaapostolake/Desktop/topo/subnets2.txt");
      BufferedReader br = new BufferedReader(fl);
      System.out.println("read from subnets");
      String line=br.readLine();
      boolean flag=true;
      while (((line = br.readLine()) != null)&& c>0)
      {
        if (flag){
          String[] parts=line.split(" ");
          while (!(parts[0].equalsIgnoreCase("subnet:")))
          {
            String[] part=line.split(" ");
            a.put(part[0], 1);
            System.out.println("put "+part[0]);
            c--;
            writeinhash++;
            if ((line = br.readLine()) == null) break;
            read++;
            parts=line.split(" ");
          }
          flag=false;
        }
        else {flag=true; notwriteinhash++;read++;}
      }
      br.close();
      System.out.println("read "+read +"writeinhash "+writeinhash+" notwriteinhash"+notwriteinhash);
    }catch(Exception e){}
    return a;
  }
  private static int findKept (ArrayList<ArrayList<Integer>> pathss)
  {
    int c=0;
    for (int i=0; i<pathss.size(); i++)
    {
      if ( (pathss.get(i).get(1)==174) ||(pathss.get(i).get(1)==6461))
      {
        c++;
      }
    }
    return c;
  }
```

6.4.2 Αποτελέσματα

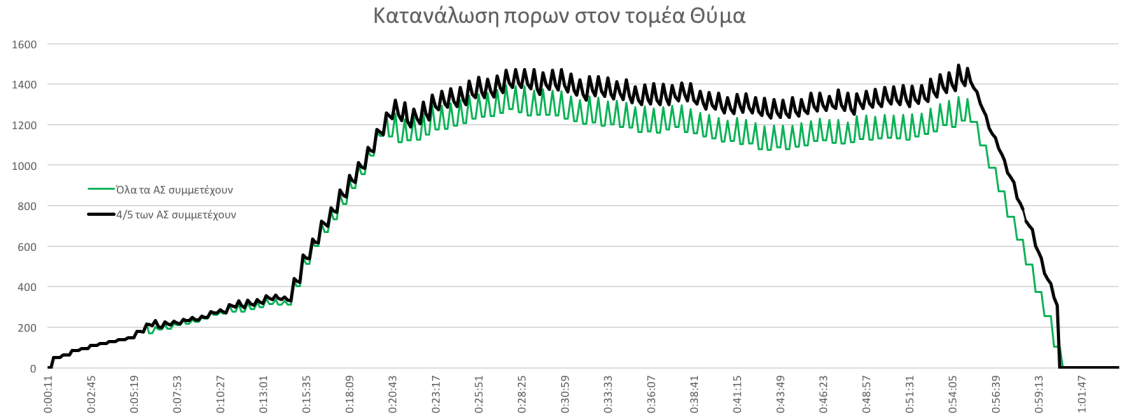
Με την παραπάνω διαδικασία κατασκευάσαμε τρείς γραφικές όλες σχετικές με την επίθεση της CAIDA που περιγράφηκε παραπάνω που παρουσιάζονται στο Σχήμα 6.3 . Η κόκκινη γραμμή δείχνει τους κανόνες που θα χρειαζόταν ο τομέας για να σταματήσει την επίθεση χωρίς κάποια εξωτερική βοήθεια. Έπειτα, βλέπουμε τους κανόνες που θα χρειαζόταν εάν όλοι οι άμεσοι γείτονες του, συμμετείχαν στην διαδικασία αντιμετώπισης με πράσινο χρώμα. Τέλος, με μαύρο χρώμα βλέπουμε το πιο ρεαλιστικό σενάριο σύμφωνα με το οποίο μόνο τα 4/5 των εμπλεκομένων ΑΣ επιδεικνύουν συνεργατική συμπεριφορά. Σε κάθε περίπτωση το θύμα υλοποιεί ένα μηχανισμό για την αναγνώριση των κακόβουλων ροών το οποίο περιγράφεται αναλυτικά στο [22] και παράγει αποτελέσματα κάθε 30 sec. Έτσι κάθε 30 δευτερόλεπτα το θύμα εγκαθιστά κανόνες και στέλνεται και μήνυμα στους γειτονικά ΑΣ για βοήθεια. Εάν αυτά συνεισφέρουν οι κανόνες στον πίνακα του θύματος λήγουν ύστερα από 300 sec. Η τιμή αυτή αντιπροσωπεύει το iddle timeout των κανόνων είναι προφανώς παραμετροποιήσιμη αλλά η τιμή επιλέχθηκε για τις ανάγκες της προσομοίωσης σύμφωνα με την default τιμή που έχει επιλέγει από τους κατασκευαστές για αντανακλαστικές λίστες πρόσβασης (reflective access list). Οι γραφικές στο Σχήμα 6.3 σχεδιάστηκαν σε λογαριθμική κλίμακα λόγω της μεγάλης διαφοράς των μεγεθών μεταξύ της περίπτωσης που κανένα ΑΣ δεν συνεισφέρουν και κάποια συνεισφέρουν. Αυτό όμως έχει σαν αποτέλεσμα να μην είναι τόσο ευκρινής η διαφορά του αριθμού των κανόνων που απαιτούνται μεταξύ των περιπτώσεων όπου όλα τα ΑΣ ή τα 4/5 αυτών συμμετέχουν οπότε οι δύο τελευταίες περιπτώσεις και δίνονται σε ξεχωριστό μη λογαριθμικό διάγραμμα στο Σχήμα 6.4.

6.5 Αξιολόγηση συστήματος ως προς την κατανομή πόρων στον χρόνο

Μέχρι στιγμής έχουμε αναφερθεί μόνο στο κέρδος που έχει ένας τομέας από την συμμετοχή του σε ένα συνεργατικό σχήμα αντιμετώπισης επιθέσεων που προκύπτει από την βοήθεια που λαμβάνει σε περίπτωση που δεχθεί κάποια επίθεση. Για να χαίρει όμως αυτής



Σχήμα 6.3: *Αριθμός κανόνων που το Domain που δέχεται επίθεση εγκαθιστά για την υπεράσπιση του (λογαριθμική κλίμακα)*



Σχήμα 6.4: *Αριθμός κανόνων που το Domain που δέχεται επίθεση εγκαθιστά*

της αντιμετώπισης από τα άλλα ΑΣ πρέπει και ο ίδιος να έχει δαπανήσει κάποιους πόρους για να συνεισφέρει στην αντιμετώπιση επιθέσεων που αυτά δέχονται. Σκοπός ενός συστήματος φήμης είναι να ελαχιστοποιήσει τους πόρους αυτούς για κάθε τομέα χωρίς να επηρεαστεί το κέρδος σε περίπτωση επίθεσης εναντίον του.

Γι να αξιολογηθεί το σύστημα μας ως προς την αποτελεσματικότητα του όσο αφορά στον καταμερισμό στον χρόνο των διαθέσιμων κανόνων προσημειώθηκαν 250 επιθέσεις εναντίον τυχαία επιλεγμένων ΑΣ στην τοπολογία . Για να είναι η προσομοίωση των επιθέσεων πιο ρεαλιστική αναπαράχθηκαν τα χαρακτηριστικά της επίθεσης του dataset της caida. Έτσι όλες οι επιθέσεις που προσημειώθηκαν είχαν την ίδια διάρκεια με την επίθεση που πραγματοποιήθηκε το 2007 καθώς και τον ίδιο αριθμό διαφορετικών IP πηγών

ανά υποδίκτυο. Το πείραμα μας πραγματοποιήθηκε σε δύο στάδια. Σε κάθε περίπτωση το 1/5 τυχαία επιλεγμένων ΑΣ (αλλά ίδιων με τα προηγούμενα πειράματα) δεν συμμετέχουν στο συνεργατικό σχήμα, δηλαδή δεν εγκαθιστούν ποτέ κανόνες εκ μέρους κάποιου άλλου ΑΣ. Στο πρώτο στάδιο τα 4/5 των ΑΣ δηλαδή τα ΑΣ που συμμετέχουν στο σχήμα εγκαθιστούν κανόνες ανεξάρτητα από την τιμή επίπεδου συνεργασιμότητας των εμπλεκομένων αυτόνομων συστημάτων. Στο δεύτερο στάδιο τα ΑΣ που συμμετέχουν αποφασίζουν αν θα συμμετέχουν ή όχι στην αντιμετώπιση μίας επίθεσης που δεν συμβαίνει εντός του ΑΣ τους, σύμφωνα με τον αλγόριθμο που περιγράφτηκε στο προηγούμενο κεφάλαιο. Σε κάθε στάδιο υπολογίστηκε η δαπάνη πόρων σε ένα συγκεκριμένο ΑΣ που για λόγους συνάφειας με τα προηγούμενα αποτελέσματα επιλέχθηκε να ταυτίζεται με το προηγούμενο θύμα της επίθεσης εκφρασμένη σε αριθμό κανόνων και διάρκεια ισχύος αυτών καθ όλη την περίοδο των επιθέσεων.

6.5.1 Θέματα Υλοποίησης

Το τρίτο πείραμα υλοποιήθηκε με την χρήση των παρακάτω συναρτήσεων

Η συνάρτηση **startAttacks** είναι υπεύθυνη για την προσομοίωση πολλαπλών επιθέσεων με τυχαία επιλεγμένα θύματα στην δοσμένη τοπολογία. Οι επιθέσεις αυτές δημιουργούνται και πάλι με πολλαπλές κλήσεις της getAllPaths η οποία φτιάχνει κάθε φορά ένα δέντρο που αντιπροσωπεύει την επίθεση με διαφορετική κάθε φορά ρίζα. Πριν τη προσομοίωση των επιθέσεων η startAttacks προδιαγράφει την συμπεριφορά κάποιων (1/5 του συνολικού αριθμού) Αυτόματων Συστημάτων ως μη συνεργατικά. Σε κάθε δημιουργία επίθεσης η startAttacks καλεί την charge.

```
public static void startAttacks( ArrayList<payment> payment,ArrayList<Integer> clients, int malNum, Hashtable<Integer,
      ArrayList<Integer>> edges, Hashtable<String, Integer> relation, Hashtable<Integer, ISP> iSPs, String file, int victim)
  {
    Set<Integer> keys = iSPs.keySet();
    for(Integer i:keys)
    {
      if (Math.random() < 0.4)
      {
        if (!(i==victim)&& iSPs.containsKey(i))
        {
          iSPs.get(i).helps=false;
          System.out.println("Maliscious"+ i);
          malNum--; if (malNum<0) break;
        }
      }
    }
```

```
    System.out.println("Victim is"+victim);
    Set<String> provs = iSPs.get(victim).Providers.keySet();
    int np= (provs.size()/4)+2;
    boolean flag=true;
    for(String i:provs)
    {
      iSPs.get(Integer.parseInt(i)).helps=flag;
      flag=false;
      System.out.println("Maliscious Provider"+ i);
      np--;
      if (np<0)break;
    }
    try
    {
      String line;
      BufferedReader brr = new BufferedReader(new FileReader(file));
      for (int h=0; h<400; h++)

      {
        line = brr.readLine();
        int root =Integer.parseInt(line);

        Hashtable<Integer,ArrayList<Integer>> paths = new Hashtable<Integer,ArrayList<Integer>>();
        int depth=6;
        ArrayList<Integer> queue= new ArrayList<Integer>();
        int[] Level= new int[10] ;
        paths=getAllPaths(edges,root,depth, relation,Level, queue,iSPs);
        //System.out.println("lene"+paths.size());

        Hashtable<Integer,ArrayList<Integer>> paths1 = (Hashtable<Integer, ArrayList<Integer>>) paths.clone();
        ArrayList<ArrayList<Integer>> pathss = new  ArrayList<ArrayList<Integer>> ();
        pathss=Tree.makepaths(depth, 0, paths, root, -5);
        //System.out.println("lene"+line);

        for (int k=0; k<pathss.size(); k++)
        {
          Collections.reverse(pathss.get(k));
        }
        Hashtable<Integer, Boolean> helped = new Hashtable<Integer, Boolean> ();

        helped.put((Integer)root, true);
        int yes=0,no=0;
        for(int j=0; j<queue.size(); j++)
        {
          if (!(paths1.containsKey(queue.get(j)))) no++;
          else
          {
            yes++;
            Tochildren(queue.get(j),iSPs, paths1,helped );
          }
        }
        //System.out.println("yes"+yes+ " no "+ no);
        charge(payment, pathss,helped,iSPs,victim,root );
      }
    }catch (Exception e){}
  }
```

Η συνάρτηση **charge()** είναι υπεύθυνη για τον υπολογισμό του κόστους σε πόρους - κανόνες δηλαδή που εγκατέστησε ένα ΑΣ το οποίο προφανώς εμπλέκεται στην επίθεση δηλαδή κάποιο μέρος της κακόβουλης κίνησης δρομολογείται μέσω αυτού και και το οποίο απο-

φάσισε να συνεισφέρει στην αντιμετώπιση της επίθεσης του ΑΣ. Το κόστος εκφράζεται σε χρόνο ισχύος των κανόνων και επηρεάζεται από την απόφαση των γειτονικών του Α Αυτόνομων Συστημάτων, τα οποία εμπλέκονται επίσης στην επίθεση. Αν αυτά αποφασίσουν να συνεισφέρουν τότε οι κανόνες λήγουν και το κόστος είναι μικρότερο. Αντίθετα αν δεν συνεισφέρουν το κόστος είναι μεγαλύτερο διότι υπάρχουν πακέτα που χτυπάνε τους κανόνες οπότε αυτοί μένουν σε ισχύ είτε μέχρι την λήξη του hard timeout είτε μέχρι την λήξη της επίθεσης.

```
  private static void charge( ArrayList<payment> allp ,ArrayList<ArrayList<Integer>> pathss,
       Hashtable<Integer, Boolean> helped, Hashtable<Integer, ISP> iSPs, int ispid, int attackid)

  {
    boolean flag= false;
    boolean afterfirst=false;
//Rules 1.victim 2.attack length 3.hops away from the victim 4.(-1) if not helping
    int temp;
    //gia ola ta paths=grammes
    for (int i =0; i< pathss.size(); i++)
    {
      //psaxnw to charge mesa sth grammh
      for(int j=0; j<pathss.get(i).size(); j++)
      {
        temp=pathss.get(i).get(0)%10+1;
        if ((pathss.get(i).get(0)==3356)||(pathss.get(i).get(0)==6461)||(pathss.get(i).get(0)==174)) temp=25;
        if (pathss.get(i).get(0)==14610)System.out.println("path "+pathss.get(i));
        Integer elem =pathss.get(i).get(j);
        if (pathss.get(i).get(j)==ispid || (ispid==pathss.get(0).get(0)))
        {
          payment p;
          if (helped.get(ispid)){
          afterfirst=true;
          flag=true;
          int k;
          for ( k=j+1; k<pathss.get(i).size(); k++)
          {
            elem =pathss.get(i).get(k);
            if (helped.get(elem)){break;}
          }
          if (k==pathss.get(i).size()) k=6;
           p= new payment(temp,j,k,attackid);
          }
          else{ p= new payment(-1,j,j,attackid);}
          allp.add(p);
      if (pathss.get(i).get(0)==174) System.out.println("ffor"+pathss.get(i).toString()+ " will pay "+ p.temp+ " from "+p.start+
            "to "+ p.end);
          break;
        }
      }
      if ((!flag)&&afterfirst) break;
      else flag=false;
    }
    if (!afterfirst) {payment p= new payment(0,0,0,attackid);  allp.add(p);
    }
  }
```

Η συνάρτηση **Tochildren** καλείται επίσης από την startArracks ύστερα από την δημιουρ-

γία κάποιας επίθεσης και σκοπό έχει να συγκεντρώσει τις αποφάσεις όλων των τομέων σχετικά με την συνεισφορά ή μη στο θύμα. Για την υλοποίηση αυτού διατρέχεται το δέντρο της επίθεσης από την ρίζα προς τα φύλλα και με την σειρά που οι κόμβοι μπήκαν αρχικά στο BFS δέντρο καθώς για να διαπιστωθεί η απόφαση ενός παιδιού κόμβου πρέπει πρώτα να είναι γνωστή η απόφαση του πατέρα. Τα αποτελέσματα της συνάρτησης αποθηκεύονται σε μία δομή hash και καταναλώνονται από την charge.

```
private static void Tochildren(Integer root, Hashtable<Integer, ISP> iSPs, Hashtable<Integer, ArrayList<Integer>> paths,
      Hashtable<Integer, Boolean> helped)
{

  if (paths.containsKey(root)){
    ISP father= iSPs.get(root);

    ArrayList<Integer> kids = paths.get(root);
    boolean help;
    boolean trust;
    for (int i =0; i<kids.size(); i++)
    {
      help=true;
      trust = false;// how much the kid trust its father
      Integer kidId= kids.get(i);

      ISP kid = iSPs.get(kidId);// will that kid help?
      //System.out.println("father "+ root+ " child "+kidId+" or "+kid.id);
      if (!kid.helps) help=false;
      else
      {
        trust= kid.willhelp(root);
        if (trust)
        {
          help=helped.get(root);
        }
      }
      if (helped.get(root)) father.update(kidId,help);
      helped.put(kidId, help);
    }
  }
  else
  {
    System.out.println("Problem with "+ root);
  }
}
```

Ένα αξιοσημείωτο στοιχείο της υλοποίησης είναι η χρήση ενός υπογράφου της αρχικής τοπολογίας στον οποίο λαμβάνουν χώρα οι επιθέσεις. Θεωρήθηκε σκόπιμο για λόγους παρουσίασης και γρηγορότερες σύγκλησης να διαμοιράσουμε τις επιθέσεις σε μικρότερο αριθμό ΑΣ από τα 50000 της αρχικής τοπολογίας. Αυτή η σύμβαση έγινε χωρίς βλάβη της γενικότητας καθώς ο υπογράφος που χρησιμοποιήσαμε αν και έχει μόνο 10000 κόμβους δηλαδή το 1/5 του αρχικού διατηρεί τις ιδιότητες του ως προς την πυκνότητα και τις σχέσεις που υποδηλώνουν οι ακμές του. Ο κώδικας της παραπάνω διαδικασίας δίνεται

παρακάτω:

```
package cop_leve;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Hashtable;
import java.util.Set;

public class FindClients {
  public static void main(String[] args) throws FileNotFoundException, UnsupportedEncodingException
  {
    System.out.println("FIle opened");
    Hashtable<String, Integer> relation = new Hashtable<String, Integer>();
    Hashtable<Integer,ArrayList<Integer>> edges = new Hashtable<Integer,ArrayList<Integer>>();
    Hashtable<Integer, ISP> ISPs = new Hashtable<Integer, ISP> ();
    try
    {
      BufferedReader br = new BufferedReader(new FileReader("/Users/mariaapostolake/Desktop/topo/as-rel2015.txt"));
      String line;
      while ((line = br.readLine()) != null)
      {
        String[] parts = line.split(" ");
        String tupla = parts[0]+parts[1];
        relation.put(tupla, (Integer.parseInt(parts[2])));
        if (!(ISPs.containsKey(Integer.parseInt(parts[1]))))
        {
          ISP isp = new ISP(Integer.parseInt(parts[1]));
          ISPs.put(Integer.parseInt(parts[1]), isp);
        }
        if (! ISPs.containsKey(Integer.parseInt(parts[0])))
        {
          ISP isp = new ISP(Integer.parseInt(parts[0]));
          ISPs.put(Integer.parseInt(parts[0]), isp);
        }
        if((parts[2]).equals("-1"))
        {
          ISP isp = ISPs.get(Integer.parseInt(parts[0]));
          isp.addclient(parts[1]);
          isp = ISPs.get(Integer.parseInt(parts[1]));
          isp.addProvider(parts[0]);
        }
        else
        {
          ISP isp = ISPs.get(Integer.parseInt(parts[1]));
          isp.addPeer(parts[0]);
          isp = ISPs.get(Integer.parseInt(parts[0]));
          isp.addPeer(parts[1]);
        }
      }
    }catch (Exception e){e.printStackTrace();}
    Set<Integer> keys = ISPs.keySet();
    Hashtable<Integer,Object> tobeRemoved = new Hashtable<Integer,Object>();
    for(Integer i:keys)
    {
      ISP isp= ISPs.get(i);
      if (isp.Customers.isEmpty()&&isp.Peers.isEmpty())
      {
        tobeRemoved.put(i,1);
      }
```

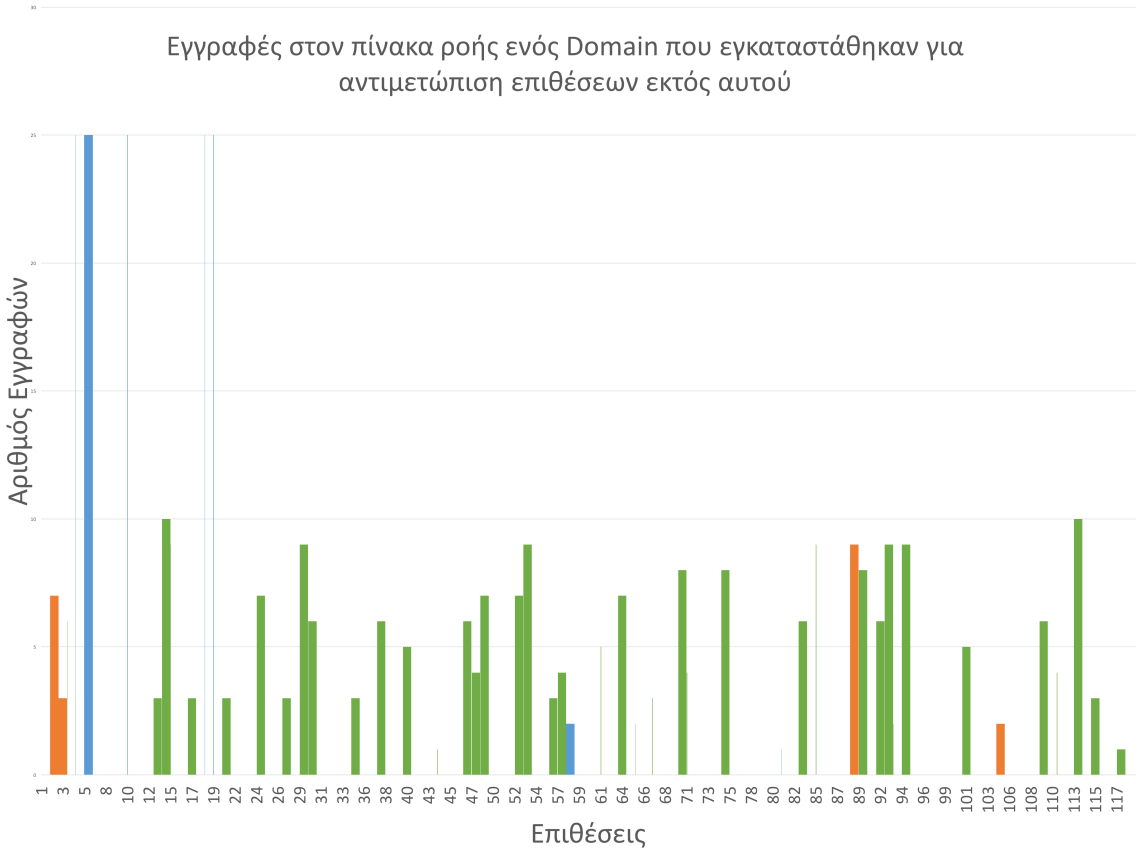
```
      {
        ArrayList<Integer> temp = new ArrayList<Integer>();
        Set<String> clients = isp.Customers.keySet();
        for(String j:clients)
        {
          temp.add(Integer.parseInt(j));
        }
        Set<String> providers = isp.Providers.keySet();
        for(String j:providers)
        {
          temp.add(Integer.parseInt(j));
        }
        Set<String> peers = isp.Peers.keySet();
        for(String j:peers)
        {
          temp.add(Integer.parseInt(j));
        }
        edges.put(isp.id,temp);
      }
    }
    //UNCOMMEND TO CREATE NEW
    PrintWriter writer = new PrintWriter("/Users/mariaapostolake/Desktop/topo/subgraph.txt", "UTF-8");
    int root=7151;//9351;//23936;//7151;//262993;//6730;//6805;//174;//132346;//157;
    Hashtable<Integer,ArrayList<Integer>> paths = new Hashtable<Integer,ArrayList<Integer>>();
    int depth=6;
    paths=getAllPaths(edges,root,depth, relation);
    Set<Integer> k = paths.keySet();
    System.out.println("In "+ root+ "nodes are "+paths.size()+"Provs "+ ISPs.get(root).Providers.size()+"Peers
          "+ISPs.get(root).Peers.size()+"Provs "+ ISPs.get(root).Customers.size());
    int p=0;
    for(Integer i:k)
    {
      p++;
      writer.println(i);
    }
    writer.close();
    System.out.println("all are"+p);
}
public static Hashtable <Integer, ArrayList<Integer>> getAllPaths ( Hashtable<Integer, ArrayList<Integer>> edges, int root, int
       dep, Hashtable<String, Integer> relation )
{
    Hashtable<Integer,ArrayList<Integer> > bfstree= new Hashtable <Integer, ArrayList<Integer>>();
    Hashtable<Integer, Integer> indexes= new Hashtable<Integer, Integer>(); // Key= ASN , Boolean= visited==added in Queue==
    ArrayList<Tupla> queue = new ArrayList<Tupla>();
    queue.add(new Tupla(root, 1));
    indexes.put(root, 1);
    //int flag=5;
    while (!queue.isEmpty())
    {
      Tupla temp=queue.get(0);//root & key
      queue.remove(0);
      Integer prev=indexes.get(temp.a);
      ArrayList<Integer> row = new ArrayList<Integer>();
      if ((temp.b+1)<=dep)
      {
        if (edges.containsKey(temp.a))
        {
          // flag=10;
          for(int j=0 ; j<(edges.get(temp.a).size()); j++)
          {
            Integer node = edges.get(temp.a).get(j); //node that could be added to bfs tree
            {
              if (!indexes.containsKey(node))
              {
```

```
                {
                  queue.add(new Tupla(node, temp.b+1));
                  row.add(node);
                  indexes.put(node, 1);
                }
              }
            }
          }
          bfstree.put(temp.a,row);
        }
      }
    }
    return bfstree ;
  }
}
```

6.5.2 Αποτελέσματα

Στα παρακάτω διαγράμματα φαίνονται τα μετρούμενα μεγέθη του πειράματος. Στο Σχήμα 6.5 βλέπουμε τους κανόνες που θα έπρεπε να εγκαταστήσει ένα ΑΣ για να συμμετέχει στο συνεργατικό σχήμα αν απλώς ικανοποιεί άκριτα κάθε αίτημα που του έρχεται. Ο χρωματικός κώδικας χρησιμοποιήθηκε για να είναι εμφανές ότι το θύμα της επίθεσης είναι αφενός διαφορετικό από το ΑΣ του οποίου την κατανάλωση πόρων παρατηρούμε και αφετέρου απέχει από αυτό μεταβλητή απόσταση αφού κάθε διαφορετικό χρώμα δηλώνει άλλη απόσταση από θύμα . Πιο συγκεκριμένα το μπλε δηλώνει ότι το ΑΣ του θύματος βρίσκεται 1 βήμα από το ΑΣ του οποίου παρακολουθούμε την κατανάλωση πόρων, το κόκκινο χρώμα δύο και το πράσινο τρία αντίστοιχα. Στο Σχήμα 6.6 βλέπουμε τους κανόνες που πρέπει να εγκαταστήσει εάν γύρω του λαμβάνουν χώρα οι ίδιες επιθέσεις με πριν και συνεπώς τα μηνύματα που δέχεται είναι τα ίδια αλλά ο ίδιος αποφασίζει αν θα συνεισφέρει ή όχι με βάση το σύστημα φήμης που περιγράφηκε στο προηγούμενο κεφάλαιο. Είναι φανερό ότι η χρήση του συστήματος είναι πολύ ωφέλιμη για ένα ΑΣ διότι ελαχιστοποιεί τους πόρους που αυτό καταναλώνει για να συμμετέχει στο συνεργατικό σύστημα. Σε μία πιο αφηρημένη περιγραφή θα μπορούσαμε να πούμε ότι το σύστημα κατανέμει τους διαθέσιμους πόρους στον χρόνο μειώνοντας τις απαιτήσεις σε υλικό. Για την ασφάλεια του ένα αυτόνομο σύστημα πρέπει να εξασφαλίσει την δυνατότητα εγκατάσταση πολλών κανόνων ανά πάσα στιγμή αυτό δεχτεί επίθεση. Αυτό μπορεί να επιτευχθεί είτε με αγορά περισσότερου hardware είτε ασφαλίζοντας (δανεισμένους) πόρους σε περίπτωση επίθεσης τους οποίους αποπληρώνει ή προπληρώνει αφιερώνοντας ένα ποσοστό των πόρων

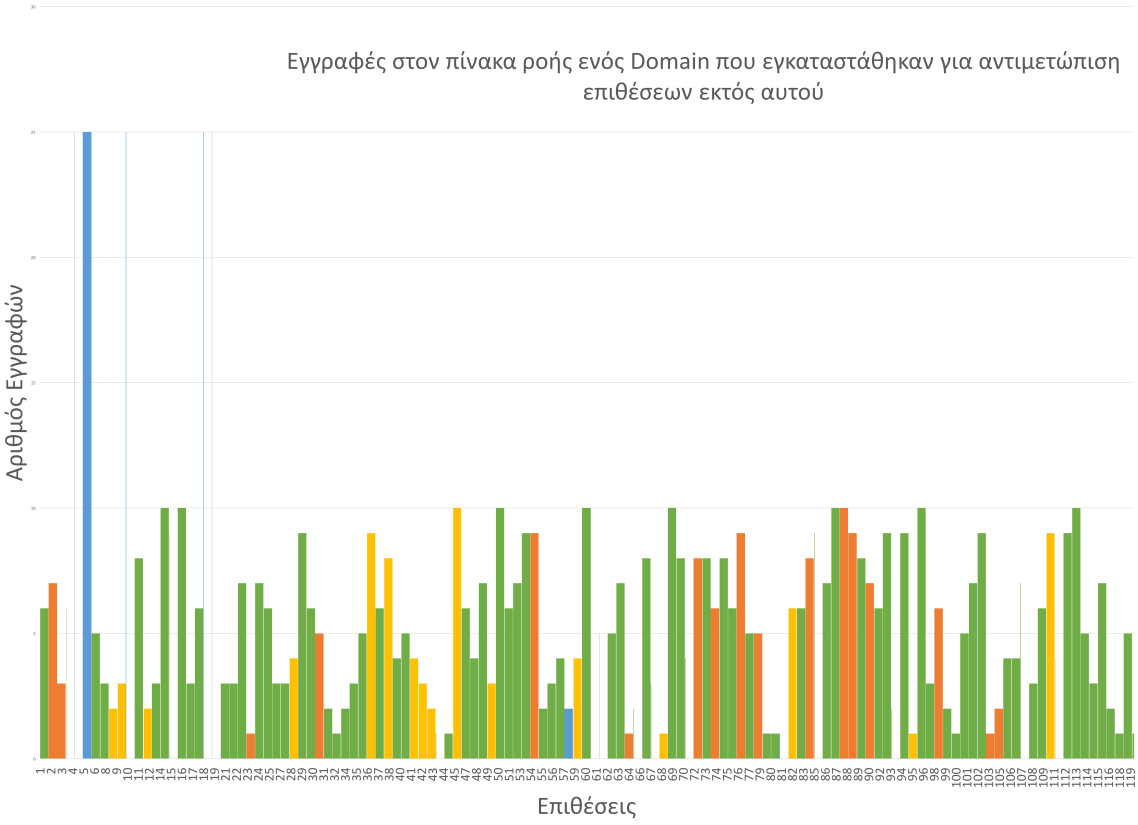
του στα γειτονικά ΑΣ για τον ίδιο σκοπό.



Σχήμα 6.5: *Εγγραφές στον πίνακα ροής ενός Domain που εγκαταστάθηκαν για αντιμετώπιση επιθέσεων εναντίον άλλων Domains, με ικανοποίηση όλων των αιτημάτων συνεργασίας*

6.6 Λοιπός κώδικας

Για την πρώτη προσομοίωση που πραγματοποιήθηκε χρησιμοποιήθηκε η τοπολογία που παρέχεται από το ”The Internet Topology Zoo”. Η διαδικτυακή Τοπολογία Zoo είναι ένα εν εξελίξει έργο για τη συλλογή δεδομένων σχετικά με τις τοπολογίες δικτύου από όλο τον κόσμο. Μέχρι στιγμής περιέχει πάνω από διακόσια πενήντα δίκτυα, σε μια ποικιλία γραφημάτων για στατιστική ανάλυση, σχεδίασης, ή άλλου είδους έρευνας σε θέματα δικτύου. Το μοντέλο που χρησιμοποιεί βασίζεται στο GÉANT - το Ευρωπαϊκό Ερευνητικό Δίκτυο Κορμού - και στα συνδεδεμένα ευρωπαϊκά Εθνικό δίκτυα έρευνας και εκπαίδευσης του (NRENs).Παρέχει διασύνδεσης επιπέδου δρομολόγησης των πολλαπλών αυτόνομων συ-



Σχήμα 6.6: *Εγγραφές στον πίνακα ροής ενός Domain που εγκαταστάθηκαν για αντιμετώπιση επιθέσεων εναντίον άλλων Domains, με επιλεκτική ικανοποίηση κάποιων αιτημάτων συνεργασίας*

στημάτων που συνθέτουν τα ευρωπαϊκά δίκτυα έρευνας. Επικεντρωθήκαμε σε αυτά τα δίκτυα, επειδή έχουν ενδιαφέρουσα ποικιλία και δομή και διότι οι φορείς εκμετάλλευσης των δικτύων αυτών παρέχουν, κατά κανόνα, υψηλής ποιότητας πληροφορίες, σχετικά με τη δομή του δικτύου τους. Το προκύπτον σύστημα περιλαμβάνει περισσότερα από 40 δίκτυα και πάνω από 1.000 δρομολογητές. Αποτέλεσμα αρκετά ικανοποιητικό δεδομένων των σημερινών εργαλείων προσομοίωση. Στην προσομοίωση αυτή κάθε ΑΣ παριστάνεται σαν ένα στιγμιότυπο της κλάσης isp και κάθε εσωτερικός του διοικητικού τομέας σαν ένα στιγμιότυπο της κλάσης client. Και οι δύο κλάσεις υλοποιούν την διπροσωπία Person πράγμα το οποίο σημαίνει ότι λαμβάνουν και στέλνουν μηνύματα μέσω ενός ξεχωριστού socket. Ταυτόχρονα είναι υποχρεωμένες να υλοποιούν συναρτήσεις που να χειρίζονται τα εισερχόμενα μηνύματα (receivemsg) να τα προωθούν (forwardmsg) να συνθέτουν μηνύματα και να τροποποιούν τις μετρικές των συστημάτων φήμης που χρησιμοποιούν ανάλογα με τα γεγονότα. Η ισχύς των κανόνων υλοποιείται μέσω νημάτων.

6.7 Συμπεράσματα

Η εκμετάλλευση των δυνατοτήτων του SDN προς την αντιμετώπιση μίας επίθεσης κατανεμημένης άρνησης υπηρεσίας είναι πολύ αποτελεσματική. Τα πιο αξιοσημείωτα πλεονεκτήματα της χρήσης SDN σε σχέση με τις παραδοσιακές μεθόδους αντιμετώπισης είναι η δυνατότητα άμεσου προγραμματισμού της συμπεριφοράς του στρώματος προώθησης δεδομένων. Υπάρχει δηλαδή δυνατότητα ρητής εγκατάστασης κανόνων που απορρίπτουν ροές με πολύ συγκεκριμένα χαρακτηριστικά σε κάθε SDN domain, αποφεύγοντας έτσι ταυτόχρονα και την απόρριψη νόμιμων ροών πακέτων. Επιπλέον, η χρήση του OpenFlow και συγκεκριμένα η εκμετάλλευση των χαρακτηριστικών της αυτόματης λήξης των κανόνων όταν δεν χρησιμοποιούνται και της ειδοποίησης του ελεγκτή για αυτό το γεγονός, εξασφαλίζει την ελαχιστοποίηση του χρόνου ισχύς των κανόνων και την διευκόλυνσης της διαδικασίας ελέγχου της συμπεριφοράς ενός γειτονικού domain αντίστοιχα. Τέλος η χρήση του OpenFlow διευκολύνει την διαδικασία εύρεσης των πηγών της επίθεσης χρησιμοποιώντας την διεύθυνση MAC των κανόνων που εφαρμόστηκαν σε κάθε domain για την δρομολόγηση της κακόβουλης κίνησης. Με τον τρόπο αυτό για την εύρεση των πηγών δεν υπερφορτώνονται τα πακέτα με επιπλέον πληροφορία, ενώ ταυτόχρονα η υπολογιστική ισχύ που απαιτείται για την εύρεση των μονοπατιών κατανέμεται στα domains από τα οποία διέρχεται η κακόβουλη κίνηση.

Η αυξημένη ταχύτητα αντιμετώπισης οφείλεται όμως σε μεγάλο βαθμό και στην αυτοματοποίηση τόσο της διαδικασίας αποστολής και λήψης των μηνυμάτων που ανταλλάσσονται για τον συντονισμό της αντιμετώπισης της επίθεσης, όσο και της διαδικασίας απόφασης που καλείται να πάρει ένα domain για το αν θα συμμετάσχει στην αντιμετώπιση μίας επίθεσης όταν του ζητηθεί.

Όσο αφορά την κατανάλωση πόρων από την μεριά του domain που δέχεται επίθεση, έγινε εμφανές ότι ο καταμερισμός των κανόνων σε άλλα SDN domains μειώνει σημαντικά τον φόρτο του και του επιτρέπει αφενός να μην αποσυνδεθεί από το internet και αφετέρου να προστατέψει την νόμιμη κίνηση του. Αξιοσημείωτη είναι και η παρατήρηση ότι ακόμα και στην περίπτωση που δεν συμμετέχουν όλα τα SDN Domains στην αντιμετώπιση μίας επίθεσης, πού είναι άλλωστε και το πιο ρεαλιστικό σενάριο, ο φόρτος που επωμίζεται

το θύμα όσον αφορά στους πόρους που χρειάζεται να δεσμεύσει για την υπεράσπιση του, είναι σημαντικά μειωμένος σε σχέση με με αυτούς που θα χρειαζόταν για να αντιμετωπίσει την επίθεση μόνος του.

Τέλος, στόχος του συστήματος ήταν να μειώσει τους πόρους που ένα SDN domain καταναλώνει για να συμμετάσχει στο συνεργατικό σύστημα. Ο αλγόριθμος μας εκπληρώνει αυτόν τον στόχο επιλέγοντας για κάθε domain τις επιθέσεις στις οποίες θα συνδράμει με απώτερο στόχο την μεγιστοποίηση της πιθανότητας συγκεκριμένα Domains να συνδράμουν αντίστοιχα σε περίπτωσή που αυτό δεχτεί επίθεση. Σε μία πιο αφηρημένη περιγραφή θα μπορούσαμε να πούμε ότι το σύστημα μας κατανέμει τους διαθέσιμους πόρους κάθε Domain στον χρόνο μειώνοντας έτσι τις απαιτήσεις σε υλικό. Αυτό συμβαίνει γιατί η υιοθέτηση του συστήματος προσφέρει μια εναλλακτική για την αντιμετώπιση DDos επιθέσεων. Αντί δηλαδή ένα SDN domain να επωμίζεται το κόστος για την αγορά περισσότερου hardware, ώστε να το χρησιμοποιήσει ολοκληρωτικά μόνο την στιγμή της επίθεσης, εξασφαλίσει τους ίδιους αριθμητικά πόρους την στιγμή της επίθεσης με την μορφή υποστήριξης από άλλα Domains, αφιερώνοντας ο ίδιος ένα μικρό κομμάτι των πόρων του για αυτούς τον υπόλοιπο καιρό.

Κεφάλαιο 7

Επίλογος- Μελλοντικές Επεκτάσεις

Λαμβάνοντας υπόψιν τις σημερινές τάσεις όσον αφορά την εξέλιξη των DDos επιθέσεων θα συνεχίσουν να αποτελούν μία από τις σημαντικότερες απειλές των σύγχρονων δικτύων. Αναμένεται επίσης στο μέλλον να γίνουν πολύ πιο σύνθετες και σφοδρές με αποτέλεσμα την μεγαλύτερη δυσκολία τόσο στην ανίχνευση όσο και στην αντιμετώπιση τους. Η συνεργασία μεταξύ διαφορετικών δικτύων - παρόχων ή αυτόνομων συστημάτων είναι επιτακτική ανάγκη καθώς η συνεχή αγορά εξειδικευμένου υλικού για να ακολουθήσει τις αυξανόμενες ανάγκες της αντιμετώπισης τους δεν είναι βιώσιμη λύση.

Προς αυτή την κατεύθυνση, στην παρούσα εργασία εκμεταλλευτήκαμε τα πλεονεκτήματα του SDN για να γίνει πιο αποτελεσματική η αντιμετώπιση τέτοιων επιθέσεων. Η χρήση των SDN προσέφερε την δυνατότητα άμεσης εφαρμογής πολιτικών ασφάλειας στο στρώμα δεδομένων ρητά και χωρίς την ανάγκη εξειδικευμένου hardware. Ταυτόχρονα έδωσε την δυνατότητα ελέγχου της συμπεριφοράς των γειτονικών ΑΣ καθώς και εύρεσης των μονοπατιών προς την πηγή της επίθεσης χωρίς υψηλό υπολογιστικό κόστος, επιβάρυνση της δρομολόγησης, κατανάλωση εύρους ζώνης ή οποιαδήποτε επηρεασμό της ανταποκρισιμότητας του δικτύου γενικότερα . Ταυτόχρονα η εφαρμογή εννοιών όπως η φήμη και η εμπιστοσύνη που χρησιμοποιούνται κατά κόρων στα peero-to-peer δίκτυα είναι μία πολλά υποσχόμενη προσθήκη στις υπάρχουσες μεθόδους συνεργατικής αντιμετώπισης επιθέσεων καθώς προσφέρει κίνητρο συνεργασίας και αίρει κάποια από τα μειονεκτήματα που επιφέρει η κατανεμημένη και μη ελεγχόμενη δομή του διαδικτύου.

Ωστόσο μένει ακόμα αρκετή ερευνά ώστε τέτοιου είδους συστήματα να μπορούν αν εφαρμοστούν στην πράξη. Είναιι απαραίτητο, πρώτα από όλα, τέτοια συστήματα να δοκιμαστούν σε πραγματικές συνθήκες για να είναι πιο ρεαλιστική η συμπεριφορά των Αυτομάτων Συστημάτων και των SDN Domains. Με μια σειρά πειραμάτων θα μπορούσε να βρεθεί το βέλτιστο όριο επιπέδου συνεργατικότητας για κάθε ξεχωριστό Αυτόνομο σύστημα με βάση την συμπεριφορά των γειτονικών ΑΣ καθώς και μέγιστος αριθμός κανόνων που ένα αυτόνομο σύστημα πρέπει να δεσμεύει ώστε να εξασφαλίσει όσους συμμάχους θεωρεί φρόνιμο.

Η αντιμετώπιση όμως μίας επίθεσης DDoS θα ήταν πολύ πιο αποτελεσματική αν ξεκινούσε προληπτικά και όχι αφού ήδη έχει προκαλέσει κάποια εμφανή ζημιά στο στόχο. Το συνεργατικό σχήμα που παρουσιάστηκε δημιουργεί ουσιαστικά ένα προφίλ για κάθε ΑΣ. Στο μέλλον θα μπορούσε να δημιουργείται και ένα προφίλ για κάθε χρήστη ώστε να έχουμε μία εκτίμηση της πιθανότητας να έχει άθελα του γίνει πηγή μίας επίθεσης και να αντιμετωπίζεται αμέσως. Θα μπορούσε ακόμα να δημιουργείται και ένα προφίλ για κάθε domain ώστε να μπορούμε οποιαδήποτε στιγμή να εκτιμήσουμε πόσο αξιόπιστη είναι η μέθοδος ανίχνευσης ανωμαλιών που χρησιμοποιεί ή πόσο κομβικής σημασίας για την επίδοση του δικτύου είναι η συμμετοχή στην αντιμετώπιση μίας επίθεση εναντίων του.

Παράρτημα Α΄

Παράρτημα Κώδικα

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
public class isp extends Person implements Serializable, Runnable
{
  public ArrayList<SocketClientExample> clients = new ArrayList<SocketClientExample>();
  public ArrayList<colleque> colleques = new ArrayList<colleque>();
  public double threshold_clients;
  public double threshold_collegues;
  public isp(ArrayList<String> me , ArrayList<Integer> edges) throws IOException
  {
    this.id=Integer.parseInt(me.get(0));
    this.port=9898+id;
    this.threshold_clients=0.9;
    this.threshold_collegues=0.1;
    for(int i=0; i <= Integer.parseInt(me.get(2)); i++)
    {
      SocketClientExample client = new SocketClientExample(i,id);
      clients.add(client);
    }
    for( int i = 1; i <edges.size(); i++)
    {
      colleque collegue = new colleque(edges.get(i),(9898+edges.get(i)), true);
      colleques.add(collegue);
    }
  }

  public void forward(message m) throws UnknownHostException, IOException, ClassNotFoundException
  {
    int index;
    int portt;
    m.sender=id;
    if (m.path.children.get(0).data.isleaf)
    {
      for (int i=0; i< m.path.children.size(); i++)
      {
```

```
        int index1= m.path.children.get(i).data.id;
        int port2=clients.get(index1).port;// Kalutera na valw ASN number h toulaxiston na allaksw to index
        System.out.println("ISP__"+id+":"+port+"__Forward to client__\n"+ port2);
        Agent myagent = new Agent(m,port2);
        feedback feed= new feedback( myagent.send(m, port2).helped);
        (new Thread(myagent)).start();
        System.out.println("ISP__"+id+":"+port+"__I will update the scop_level of__"+ index1 +"\n");
        clients.get(index1).update(feed.helped);
      }
    }
    else
    {
      int i;
      message m2 = new message(m.from, m.to, m.falsepositive, m.nrc, m.sender, m.isclient);
      m2.path=m.path;
      m2.StartTree=m.StartTree;
      for ( i=0; i< m2.path.children.size(); i++)
      {
        message mm = new message(m2.from, m2.to, m2.falsepositive, m2.nrc, m2.sender, m2.isclient);
        mm.StartTree=m2.StartTree;
        mm.isclient= false;
        index= m2.path.children.get(i).data.id;
        int newindex=this.idToIndex(index);
        portt= colleques.get(newindex).port;
        mm.path= m2.path.children.get(i);
        Agent myagent = new Agent(mm,portt);
        System.out.println("ISP__"+id+":"+port+"__Forward to ISP__"+portt);
        myagent.send(m, portt);
        (new Thread(myagent)).start();
        if (m.from==id) colleques.get(index-1).update(feed.helped,true);
      }
    }
}

public void receivemsg(message m) throws UnknownHostException, ClassNotFoundException, IOException, InterruptedException
{
    if (m.isclient)
    {
      if (clients.get(m.from).sus_level< threshold_clients)
      {
        System.out.println("ISP__"+id+":"+port+"__client__"+m.from+ "__is trustworthy\n");
        m.isclient=false;
         install (m.nrc, feed, m.time );
        decisionToTree( m.StartTree,id,true,1.0);
        resourses-=m.nrc;
        System.out.println("ISP__"+id+":"+port+"__installed rules now resources are__"+resourses);
        forward(m);
        (new Thread(new AutoexpireThread(this, m.nrc , m.StartTree))).start(); TO BE FIXED ################
      }
      else
      {
        decisionToTree(m.StartTree,id,false,1.0);
        System.out.println("ISP__"+id+":"+port+"__client__"+m.from+ "__is not trustworthy\n");
        /install (m.nrc,feed,m.time);
        (new Thread(new AutoexpireThread(this, m.nrc, feed))).start();TO BE FIXED #########
      }
    }
    else
    {
      Tuple myTuple;
      int index= idToIndex(m.from);
      colleque myColleque = colleques.get(index);

      if (!(myColleque.isneighbor))
```

```
      {
        //myColleque.updateStanger(m.isps);
        //myColleque.cop_level_gen=m.isps.get(0).cop_level;
      }
      else
      {
      if (myColleque.Gn> myColleque.In)
      {
        myTuple = new Tuple(myColleque.cop_level_gen , myColleque.Gn) ;
      }
      else
      {
        myTuple = new Tuple(myColleque.cop_level_in , myColleque.In) ;
      }
      m.isps.add(myTuple);
      System.out.println("ISP__"+id+":"+port+"__ISP__added in m\n");
      }
      forward(m);
      //install (m.nrc, feed , m.time);
      // (new Thread(new AutoexpireThread(this, m.nrc, feed))).start();TO BE FIXED #########
      double cop= max (myColleque.cop_level_gen,myColleque.cop_level_in);
      if (cop >threshold_collegues)
      {
        //decisionToTree(m.StartTree,id,true,1.0);
        System.out.println("ISP__"+id+":"+port+"__ISP__"+m.from+":"+ "__is trustworthy\n");
       resourses-=m.nrc;
       // System.out.println("ISP__"+id+":"+port+"__installed rules now resources are__"+resourses);
      }
      else
      {
        //decisionToTree(m.StartTree,id,false,1.0);
        System.out.println("ISP__"+id+":"+port+"__ISP__"+m.from+":"+ "__is not trustworthy\n");
      }
      //   Tuple myTuple = new Tuple(-1, d, 1);
      //TODO    m.isps.add(t);
    }
  }
  private double max(double a, double b) {
    if (a>b) return a;
    // TODO Auto-generated method stub
    else return b;
  }
  public int getindex (message m)
  {
    int i;
    boolean flag=false;
    for(i=0 ; i< colleques.size(); i++)
    {
      if (colleques.get(i).id == (m.from))
      {
        flag=true;
        break;
      }
    }
    if (!flag)
    {
      System.out.println("ISP__"+id+":"+port+"__save cop_level for__"+m.from);
      //Tuple t = new Tuple(m.from, 0.0, 0);
      colleque e = new colleque(m.from, 0, false);
      colleques.add(e);
    }
    return i;
  }
```

```
public void run(){
  //create the socket server object
  //keep listens indefinitely until receives 'exit' call or program terminates
  //isp isp = new isp(1,2,2,0.1,0.8);
  System.out.println("ISP__"+id+":"+port+ "__ listens to___" + (port) );
  ServerSocket server = null;
  try {
    server = new ServerSocket( port);
  } catch (IOException e) {
    e.printStackTrace();
  }
  while(true)
  {
    Socket socket = null;
    try {
      socket = server.accept();
    } catch (IOException e) {
      e.printStackTrace();
    }
    //read from socket to ObjectInputStream object
    ObjectInputStream ois = null;
    try {
      ois = new ObjectInputStream(socket.getInputStream());
    } catch (IOException e) {
      e.printStackTrace();
    }
    message message = null;
    try {
      message = (message) ois.readObject();
    } catch (ClassNotFoundException | IOException e) {
      e.printStackTrace();
    }
    if (message.isclient)
      System.out.println("ISP__"+id+":"+port+"__received from client__" + message.sender+ "__this-->"+ message.falsepositive );
    else
      System.out.println("ISP__"+id+":"+port+"__received from ISP__" + message.sender+ "__this-->"+ message.falsepositive );
    try {
      receivemsg(message);
    } catch (ClassNotFoundException | IOException
        | InterruptedException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } // install + forward h sendIODEF
    //create ObjectOutputStream object
    ObjectOutputStream oos = null;
    try {
      oos = new ObjectOutputStream(socket.getOutputStream());
    } catch (IOException e) {
      e.printStackTrace();
    }
    // close resources
    try {
      ois.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
    try {
      oos.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
    try {
      socket.close();
    } catch (IOException e) {
```

```
          e.printStackTrace();
        }
      }
    }

    public int idToIndex(int id)
    {
      int i=0;
      while( i<colleques.size())
      {
        if (colleques.get(i).id==id) return i;
        else i++;
      }
      System.out.println("ISP__"+id+":"+port+"__save cop_level for__"+id);
      colleque e = new colleque(id, 0, false);
      colleques.add(e);
      return i;
    }
}
```

Βιβλιογραφία

[1] D. L. Tennenhouse and D. J. Wetherall, “Towards an active network architecture,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 5, pp. 81–94, Oct. 2007. [Online]. Available: http://doi.acm.org/10.1145/1290168.1290180

[2] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, “A survey of programmable networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, Apr. 1999. [Online]. Available: http://doi.acm.org/10.1145/505733.505735

[3] N. Feamster, J. Rexford, and E. Zegura, “The road to sdn: An intellectual history of programmable networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602204.2602219

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: Enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[5] Open Networking Foundation., Site : https://www.opennetworking.org/.

[6] *OpenFlow Switch Specification Version 1.0.0. 2009*, OpenFlow Switch Consortium, Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf, 3 2013.

[7] *OpenFlow Switch Specification Version 1.1.0. 2011*, OpenFlow Switch Consortium and Others, Available online: http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf, 3 2015.

[8] *OpenFlow Switch Specification Version 1.1.0. 2011*, Open Networking Foundation., Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.pdf, 11 2013.

[9] *OpenFlow Switch Specification Version 1.1.0. 2011*, Open Networking Foundation., Available online: https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/openflow-spec-v1.3.2.pdf, 1 2014, rev. 3.

[10] *OpenFlow Switch Specification Version 1.1.0. 2011*, Open Networking Foundation., Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, 1 2014.

[11] *SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains*, Available online:http://tools.ietf.org/id/draft-yin-sdn-sdni-00.txtf, 6 2012.

[12] A. Gupta et al., "Sdx: A software defined internet exchange," in *Proceedings of ACM SIGCOMM*, 2014, pp. 579–580.

[13] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. J. Clark, and E. Katz-Bassett, "SDX: a software defined internet exchange," in *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, 2014, pp. 551–562. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626300

[14] J. Zhu, W. Xie, L. Li, M. Luo, and W. Chou, "Software service defined network: Centralized network information service," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE, 2013, pp. 1–7.

[15] Ł. Podleski, E. JACOB, J. AZNAR-BARANDA, X. JEANNIN, K. BAUMANN, and C. ARGYROPOULOS, "Multi-domain software defined network: exploring possibilities in," *TNC 2014*.

[16] P. Lin, J. Bi, Z. Chen, Y. Wang, H. Hu, and A. Xu, "We-bridge: West-east bridge for SDN inter-domain network peering," in *2014 Proceedings IEEE INFOCOM*

Workshops, Toronto, ON, Canada, April 27 - May 2, 2014, 2014, pp. 111–112. [Online]. Available: http://dx.doi.org/10.1109/INFCOMW.2014.6849180

[17] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: Dynamic access control for enterprise networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: ACM, 2009, pp. 11–18. [Online]. Available: http://doi.acm.org/10.1145/1592681.1592684

[18] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "Veriflow: verifying network-wide invariants in real time," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 467–472, 2012.

[19] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with openflow/nox architecture," in *ICNP'11*, 2011, pp. 7–12.

[20] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 127–132.

[21] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, "A novel design for future on-demand service and security," in *Communication Technology (ICCT), 2010 12th IEEE International Conference on*, nov. 2010, pp. 385 –388.

[22] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Comput. Netw.*, vol. 62, pp. 122–136, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.10.014

[23] K. Giotis, G. Androulidakis, and V. Maglaris, "Leveraging SDN for efficient anomaly detection and mitigation on legacy networks," in *Third European Workshop on Software Defined Networks, EWSDN 2014, Budapest, Hungary, September 1-3, 2014*, 2014, pp. 85–90. [Online]. Available: http://dx.doi.org/10.1109/EWSDN.2014.24

[24] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in *Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection*, ser. RAID’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 161–180. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23644-0_9

[25] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, Oct. 2010, pp. 408 –415.

[26] K. J. Argyraki and D. R. Cheriton, “Active internet traffic filtering: Real-time response to denial-of-service attacks.” in *USENIX annual technical conference, general track*, 2005, pp. 135–148.

[27] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and D. McPherson, “Dissemination of flow specification rules,” Internet Requests for Comments, RFC Editor, RFC 5575, August 2009, http://www.rfc-editor.org/rfc/rfc5575.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5575.txt

[28] T. Berners-Lee, R. T. Fielding, and L. Masinter, “Uniform resource identifier (uri): Generic syntax,” Internet Requests for Comments, RFC Editor, STD 66, January 2005, http://www.rfc-editor.org/rfc/rfc3986.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3986.txt

[29] Y. Rekhter, T. Li, and S. Hares, “A border gateway protocol 4 (bgp-4),” Internet Requests for Comments, RFC Editor, RFC 4271, January 2006, http://www.rfc-editor.org/rfc/rfc4271.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4271.txt

[30] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, “Multiprotocol extensions for bgp-4,” Internet Requests for Comments, RFC Editor, RFC 4760, January 2007, http://www.rfc-editor.org/rfc/rfc4760.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4760.txt

[31] H. Debar, D. Curry, and B. Feinstein, “The intrusion detection message exchange format (idmef),” Internet Requests for Comments, RFC Editor, RFC

4765, March 2007, http://www.rfc-editor.org/rfc/rfc4765.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4765.txt

[32] R. Danyliw, J. Meijer, and Y. Demchenko, "The incident object description exchange format," Internet Requests for Comments, RFC Editor, RFC 5070, December 2007, http://www.rfc-editor.org/rfc/rfc5070.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5070.txt

[33] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[34] X. Dimitropoulos, D. Krioukov, B. Huffaker, k. claffy, and G. Riley, "Inferring AS Relationships: Dead End or Lively Beginning?" Santorini, Greece, pp. 113–125, May 2005.

[35] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, k. claffy, and G. Riley, "AS Relationships: Inference and Validation," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 37, no. 1, pp. 29–40, Jan 2007.

[36] The CAIDA AS Relationships Dataset, <2015>., http://www.caida.org/data/as-relationships/.

[37] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1109/90.974527

[38] J. E. Hopcroft and J. D. Ullman, *Introduction To Automata Theory, Languages, And Computation*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[39] The CAIDA UCSD ?DDoS Attack 2007 Dataset., http://www.caida.org/data/passive/ddos-20070804$_{d}ataset.xml$.

[40] D. Magoni and J. J. Pansiot, "Analysis of the autonomous system network topology," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 3, pp. 26–37, Jul. 2001. [Online]. Available: http://doi.acm.org/10.1145/505659.505663