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

# ΑΝΑΠΤΥΞΗ ΚΑΙ ΕΦΑΡΜΟΓΗ ΕΝΟΣ ΔΙΚΤΥΑΚΟΥ ΤΗΛΕΣΚΟΠΙΟΥ (NETWORK TELESCOPE) ΓΙΑ ΑΝΙΧΝΕΥΣΗ ΔΙΚΤΥΑΚΩΝ ΑΝΩΜΑΛΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

**ΕΥΣΤΑΘΙΑΣ Ε. ΜΕΓΑΛΙΟΥ**

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

Αθήνα, Οκτώβριος 2005

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

# ΑΝΑΠΤΥΞΗ ΚΑΙ ΕΦΑΡΜΟΓΗ ΕΝΟΣ NETWORK TELESCOPE ΓΙΑ ΑΝΙΧΝΕΥΣΗ ΔΙΚΤΥΑΚΩΝ ΑΝΩΜΑΛΙΩΝ ΜΕ ΧΡΗΣΗ ΤΟΥ "UNALLOCATED ADDRESS SPACE"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

**ΕΥΣΤΑΘΙΑΣ Ε. ΜΕΓΑΛΙΟΥ**

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ……

| | | |
|---|---|---|
| ..... | ..... | ..... |
| Β. Μάγκλαρης | Ε. Συκάς | Σ. Παπαβασιλείου |
| Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. | Επίκουρος Καθηγητής Ε.Μ.Π. |

Αθήνα, Οκτώβριος 2005

.....

**Ευσταθία Ε. Μεγαλιού**

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

## ΠΕΡΙΛΗΨΗ

Τα μεγάλα επιχειρησιακά δίκτυα αντιμετωπίζουν με έναν συνεχώς αυξανόμενο ρυθμό, ποικίλα περιστατικά που απειλούν την ασφάλειά τους, όπως επιθέσεις DDoS, εισβολή σκουληκιών (worms), port scans και άλλες δικτυακές δυσλειτουργίες. Κατά συνέπεια, οι προσεγγίσεις για τη διαχείριση ασφάλειας δικτύων που αποσκοπούν στο γρήγορο και ακριβή εντοπισμό των παραπάνω επιθετικών ενεργειών, αποδεικνύονται πλέον μεγάλης σημασίας. Τα τείχη προστασίας (firewalls) και τα συστήματα ανίχνευσης εισβολέων (IDSs), είναι οι πιο κοινές τεχνικές εντοπισμού των ενεργειών αυτών. Ωστόσο υπάρχουν και νέες, αναπτυσσόμενες τεχνολογίες των οποίων τα αποτελέσματα μπορούν να δώσουν νέα ώθηση στον τομέα της ασφάλειας των δικτύων. Μία από αυτές τις νέες τεχνικές είναι τα Δικτυακά Τηλεσκόπια (Network Telescopes).

Ένα network telescope παρακολουθεί δικτυακή κίνηση από ή προς ένα κομμάτι των διευθύνσεων του δικτύου που δεν έχουν αποδοθεί προς χρήση. Μελετώντας αυτή την "ανεπιθύμητη" κίνηση που καταγράφεται από το τηλεσκόπιο, μας δίνεται η ευκαιρία να παρατηρήσουμε διάφορα μεμονωμένα περιστατικά ασφαλείας, όπως μόλυνση των χρηστών του δικτύου από worms, απόπειρες για network scanning κ.α.

Στην εργασία αυτή θέλουμε να αναπτύξουμε έναν μηχανισμό που υλοποιεί ένα network telescope, ο οποίος θα εφαρμόζεται σε ένα δίκτυο και θα μπορεί αυτόματα να προσδιορίζει το σύνολο των μη χρησιμοποιούμενων διευθύνσεων του δικτύου αυτού, προς τις οποίες την κίνηση μας ενδιαφέρει να μελετήσουμε. Στη συνέχεια, στηριζόμενοι σε ευρέως διαδεδομένα εργαλεία (flow-tools), αναπτύσσουμε λογισμικό που μας επιτρέπει να καταγράφουμε τα δεδομένα της παραπάνω "κακόβουλης" δικτυακής κίνησης. Από τη μελέτη τους, θα μπορούμε να εξάγουμε χρήσιμα συμπεράσματα για την ασφάλεια ενός δικτύου. Επιπλέον, επεκτείνουμε τον παραπάνω μηχανισμό σε μία εφαρμογή ταυτόχρονης συλλογής και καταγραφής δεδομένων που προέρχονται από διαφορετικά δίκτυα, κάνοντας με τον τρόπο αυτό ευκολότερη την υλοποίηση "ISP – level / Backbone" network telescopes.



**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ :** Επιθέσεις DDoS, worms, Δικτυακό Τηλεσκόπιο, Unallocated Address Space, Ανίχνευση Ανωμαλιών, Ανίχνευση Δυσλειτουργιών, Σύστημα Εντοπισμού Εισβολέων, Netflow, flow-tools, RRDTool

## ABSTRACT

Enterprise networks are facing ever-increasing security threats, like DDoS attacks, intrusion of worms, port scans, and network misuse. Effective monitoring approaches to quickly detect these activities are greatly needed. Firewalls and intrusion detection systems (IDSs) are the most common ways to detect these activities. Network Telescopes is a new developing approach in the domain of network security monitoring, that can be a valuable enhancement to existing technologies.

A network telescope is a portion of the network's IP address space which has not been assigned for use. Observing this 'unexpected' traffic arriving at a network telescope, provides the opportunity to view remote network security events such as spoofed DDoS attacks, infection of the network hosts by worms, attempts of network scanning etc.

In this thesis, we develop an application that can ease the deployment of network telescopes by automatically calculating the unallocated address space, using routing information (IGP). We extend the functionality of popular open-source network monitoring tools (the OSU-flow-tools), in order to capture and investigate the 'malicious' traffic that reaches the telescope. Furthermore, our application is designed for parallel collection and recording of network telescope data, that stems from or heads to different networks. Our work can definitely ease the task of building and operating 'transit level network telescopes'.



**KEYWORDS :** DDoS attacks, worms, Network Telescope, Unallocated Address Space, Anomaly Detection, Misuse Detection, Intrusion Detection System (IDS), Netflow, flow-tools, RRDTool

# ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες πρώτα απ'όλα στον επιβλέποντα καθηγητή κ. Βασίλειο Μάγκλαρη, για την ανάθεση αυτής της διπλωματικής εργασίας και την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο θέμα.

Επιπλέον θα ήθελα να ευχαριστήσω τον υπεύθυνο της εργασίας μου και υποψήφιο διδάκτορα Χρήστο Σιατερλή, για την πολύτιμη βοήθειά του, το χρόνο που μου διέθεσε και την καθοδήγηση που μου παρείχε, χωρίς την οποία θα ήταν αδύνατη η ολοκλήρωση της εργασίας αυτής.

Ακόμη, θα ήθελα να ευχαριστήσω όλα τα μέλη του εργαστηρίου Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων (NETMODE) για την προθυμία που έδειξαν στην επίλυση των προβλημάτων που μου παρουσιάστηκαν.

Τέλος, ένα μεγάλο ευχαριστώ οφείλω στην οικογένειά μου και ιδιαίτερα στον πατέρα μου, για τη σημαντική βοήθεια και την ψυχολογική στήριξη που μου παρείχε, καθ' όλη τη διάρκεια των σπουδών μου.

# ΠΕΡΙΕΧΟΜΕΝΑ

# ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ

Το σημερινό παγκόσμιο μοντέλο του Internet, αντιμετωπίζει έναν συνεχώς αυξανόμενο αριθμό απειλών από επιθέσεις κάθε είδους, όπως Denial of service attacks και worms attacks. Η κατανεμημένη φύση του και το χαμηλό κόστος διακίνησης δικτυακών δεδομένων, δίνουν τη δυνατότητα στους επιτιθέμενους να εξασφαλίζουν εύκολη πρόσβαση και να κερδίζουν τον έλεγχο πλήθους χρηστών, γεγονός το οποίο αποτελεί σοβαρή απειλή για την ασφάλεια και την αξιοπιστία ολόκληρου του Διαδικτύου.

Η ικανότητα έγκαιρης ανίχνευσης των επιθετικών μηχανισμών στα πρώτα στάδια της δράσης τους, κρίνεται ιδιαίτερα σημαντική στην προσπάθεια παρεμπόδισης και αντιμετώπισης των μηχανισμών αυτών, μέσω διαφόρων αμυντικών συστημάτων. Το ίδιο ισχύει και για μικρότερα δίκτυα υπολογιστών, τα οποία έχουν ενταχθεί πλέον κανονικά στην καθημερινή μας ζωή.

Ο βασικός μας στόχος στην παρούσα εργασία είναι η δημιουργία ενός μηχανισμού συλλογής και καταγραφής δεδομένων δικτυακής κίνησης που μπορεί να θεωρείται "ύποπτη". Το εργαλείο αυτό ονομάζεται δικτυακό τηλεσκόπιο ή network telescope και η βασική του ιδέα στηρίζεται στη διαχείριση του συνόλου διευθύνσεων ενός δικτύου, οι οποίες δεν έχουν αποδοθεί σε χρήστες και αποτελούν το "unused address space" του δικτύου ή αλλιώς "dark space". Η κίνηση που δρομολογείται προς τις διευθύνσεις αυτές θεωρείται "κακόβουλη", από τη στιγμή που δεν απευθύνεται σε υπαρκτούς χρήστες και μπορεί να θεωρηθεί ως αποτέλεσμα κάποιας δυσλειτουργίας του συστήματος, ή απόκριση λόγω "address spoofing" ή ακόμη και "port scanning" από πιθανά worms. Η παρακολούθηση και η μελέτη των δεδομένων που αφορούν το "unused address space", έχει αποδειχθεί μία αποτελεσματική μέθοδος στη συγκέντρωση πληροφοριών που αφορούν πιθανές επιθέσεις και στην αναγνώριση των επιθετικών τάσεων που επικρατούν σε ένα δικτυακό σύστημα. Η μελέτη αυτή μπορεί να οδηγήσει στην ανάπτυξη κατάλληλων μηχανισμών για αντιμετώπιση μελλοντικών επιθέσεων τέτοιου είδους.

Στο 2ο Κεφάλαιο της παρούσας εργασίας, παρουσιάζουμε κάποια από τα πιο γνωστά είδη Ανωμαλιών Δικτυακής Κίνησης που αποτελούν ιδιαίτερη απειλή για τη σημερινή ασφάλεια των δικτύων, όπως είναι τα worms και οι DDoS attacks. Αναφερόμαστε στα βασικά χαρακτηριστικά τους και σε κάποιους μηχανισμούς ανίχνευσης που έχουν ήδη αναπτυχθεί για την αντιμετώπισή τους.

Στο 3ο Κεφάλαιο κάνουμε μία αναφορά στις υπάρχουσες τεχνικές που αφορούν την ανίχνευση δικτυακών ανωμαλιών και τον εντοπισμό εισβολέων, όπως είναι τα συστήματα IDS (Intrusion Detection Systems) και μέθοδοι που βασίζονται στη διαχείριση του "unused address space". Μάλιστα, γίνεται διάκριση ανάμεσα στις δύο προσεγγίσεις που αφορούν την ανίχνευση ανωμαλιών (anomaly detection) και την ανίχνευση δυσλειτουργιών (misuse detection) αντίστοιχα.

Στο 4ο Κεφάλαιο κάνουμε μία εισαγωγή στην τεχνολογία των Network Telescopes, εξηγούμε τη λειτουργία τους αλλά και τις δυσκολίες που ανακύπτουν από τη μέχρι τώρα χρήση τους. Αναφέρουμε επίσης τα προβλήματα τα οποία στοχεύουμε να ξεπεράσουμε μέσα από τη δική μας προτεινόμενη αρχιτεκτονική.

Στο 5ο Κεφάλαιο γίνεται μία αναλυτική περιγραφή της προτεινόμενης αυτής αρχιτεκτονικής υλοποίησης του telescope. Συγκεκριμένα, παρουσιάζονται τα επιμέρους στοιχεία που συνθέτουν την εφαρμογή και περιγράφεται η λειτουργία τους και ο ακριβής τρόπος υλοποίησής τους. Γίνεται επίσης μία βασική αναφορά στα flow-tools, εργαλεία τα οποία μας βοηθούν στη συλλογή των δεδομένων κίνησης. Τέλος, περιγράφουμε τον τρόπο με τον οποίο επεκτείνουμε μία από τις υπάρχουσες υλοποιήσεις, στην ευρύτερη εφαρμογή της σε περισσότερα του ενός δίκτυα ταυτοχρόνως.

Στο 6ο και τελευταίο Κεφάλαιο παρουσιάζουμε μία εφαρμογή του telescope που περιγράφηκε, σε πραγματικό δίκτυο. Συγκεκριμένα, δίνουμε ένα παράδειγμα εκτέλεσής του για δεδομένα που αφορούν το δίκτυο του Εθνικού Μετσόβιου Πολυτεχνείου (Ε.Μ.Π.) αλλά και αυτό του Πανεπιστημίου Αθηνών (Ε.Κ.Π.Α.). Η μελέτη και η σύγκριση των αποτελεσμάτων που παίρνουμε μπορεί να οδηγήσει σε χρήσιμα συμπεράσματα.

# ΚΕΦΑΛΑΙΟ 2 : ΕΙΔΗ ΑΝΩΜΑΛΙΩΝ ΔΙΚΤΥΑΚΗΣ ΚΙΝΗΣΗΣ

## 2.1 ΣΚΟΥΛΗΚΙΑ (Worms)

Με τον όρο σκουλήκια (worms) περιγράφεται στη βιβλιογραφία ένα είδος προγραμμάτων παραπλήσιων των ιών (viruses), που χρησιμοποιούνται για κακόβουλους σκοπούς και εισβάλλουν στους υπολογιστές ενός δικτύου, προκαλώντας διάφορες δυσλειτουργίες στο σύστημα που προσβάλλουν. Τα προγράμματα αυτά είναι συνήθως μικρού μεγέθους και η μετάδοσή τους βασίζεται στη δικτυακή επικοινωνία. Το πρώτο γνωστό περιστατικό εμφάνισης σκουληκιού, παρουσιάστηκε το 1988 με την ονομασία 'Morris worm' και είχε ιδιαίτερα καταστροφικά αποτελέσματα για τον τότε κόσμο του Internet. Με την εμφάνιση του Code Red II τον Ιούλιο του 2001, ξεκίνησε ένα νέο κύμα επιθέσεων που εξαπολύονταν μέσω διαφόρων τύπων worms. Τα worms αποτελούν πλέον σοβαρή απειλή για την ασφάλεια ολόκληρου του Διαδικτύου και έχει καταστεί ιδιαίτερα επιτακτική η ανάγκη δημιουργίας και λειτουργίας μηχανισμών εντοπισμού και αντιμετώπισης των worms.

### 2.1.1 Χαρακτηριστικά των worms

Κύριο χαρακτηριστικό στοιχείο αυτών των 'κακόβουλων προγραμμάτων' είναι ότι αναπαράγονται αυτόματα, μολύνοντας αρχικά έναν υπολογιστή, ο οποίος στη συνέχεια προσπαθεί να δημιουργήσει αντίγραφά του, μολύνοντας και άλλους υπολογιστές σε ένα δίκτυο-στόχο. Για την εισβολή και την εξάπλωσή τους, τα worms εκμεταλλεύονται διάφορες αδυναμίες και ευαισθησίες του συστήματος και των διεργασιών του.

Η κύρια διαφορά που τα ξεχωρίζει από τους ιούς (viruses) είναι, ότι ενώ οι ιοί απαιτούν κάποια ανθρώπινη ενέργεια προκειμένου να δράσουν στο σύστημα και να εξαπλωθούν, τα worms χαρακτηρίζονται ως "self-propagating", δηλαδή πολλαπλασιάζονται και εξαπλώνονται αυτόματα μεταξύ των κόμβων ενός δικτύου,

χωρίς να απαιτείται κάποια ανθρώπινη παρέμβαση. Γενικά οι ιοί πολλαπλασιάζονται με πιο αργούς ρυθμούς και έχουν αναπτύξει πιο ώριμους μηχανισμούς άμυνας απέναντι σε προγράμματα anti-virus, τα οποία προσπαθούν να εντοπίσουν και να αναστείλουν την εξάπλωσή τους.

Τα βασικά χαρακτηριστικά των worms μπορούν να ταξινομηθούν ως εξής :

- Αυτονομία (autonomy) : εξ'ορισμού, τα worms είναι αυτόνομες οντότητες. Από τη στιγμή που απελευθερώνονται σε ένα σύστημα, είναι σχεδιασμένα έτσι ώστε να δράσουν και να πολλαπλασιαστούν χωρίς κάποια ανθρώπινη παρέμβαση.
- Αναπαραγωγή (propagation) : τα worms αναπαράγονται μόνα τους μεταξύ των διαφόρων χρηστών και χαρακτηρίζονται από πολυμορφία.
- Δυνατότητες αναγνώρισης στόχου (target discovery) : προκειμένου ένα worm να είναι ικανό να αναγνωρίσει έναν πιθανό στόχο, διαθέτει μία "βάση δεδομένων" η οποία διατηρεί στοιχεία με γνωστές αδυναμίες και ευαισθησίες του συστήματος και του λογισμικού. Για τον εντοπισμό τους, χρησιμοποιεί τους πόρους του δικτύου προκειμένου να στείλει διάφορα πακέτα τα οποία μπορούν να αποκαλύψουν την παρουσία των αδυναμιών αυτών. Χρησιμοποιούνται γενικά διάφορες τεχνικές εντοπισμού πιθανών στόχων (port scanning, external target lists, pre-generating target lists…)
- Δυνατότητες επίθεσης : από τη στιγμή που ένας πιθανός στόχος έχει εντοπιστεί, ένα worm θα πρέπει να βρει τον τρόπο με τον οποίο θα εκμεταλλευτεί μία υπάρχουσα αδυναμία για να κερδίσει την πρόσβαση στο στόχο. Εφόσον επιτευχθεί αυτό, τότε εγκαθίσταται στο σύστημα του χρήστη και αρχίζει να εκτελεί τον ανάλογο κώδικα, ενώ στη συνέχεια κάνει τις απαραίτητες ενέργειες προκειμένου να παράγει αντίγραφά του και να δρομολογήσει την επίθεση και προς τα συστήματα των άλλων χρηστών του δικτύου.
- Ποικίλες δυνατότητες εισβολής : ένα εξελιγμένο worm εντοπίζει συνήθως περισσότερες από μία αδυναμίες του συστήματος και έτσι διαθέτει ποικίλους τρόπους προκειμένου να εισβάλει σε ένα σύστημα.

- Μηχανισμοί αναπαραγωγής και εξάπλωσης (propagation carriers and distribution mechanisms) : τα μέσα τα οποία χρησιμοποιεί ένα worm για την εξάπλωσή του επηρεάζουν την ταχύτητα και τη μυστικότητα που διαθέτει. Μπορεί να μεταφερθεί είτε απευθείας από μηχάνημα σε μηχάνημα, είτε να αποτελέσει κομμάτι μιας συνηθισμένης επικοινωνίας μεταξύ των χρηστών του δικτύου. Για παράδειγμα, η εξάπλωση ενός 'self-carried' worm είναι κομμάτι της μολυσματικής του δράσης και πραγματοποιείται ανεξάρτητα από οποιαδήποτε άλλη ενέργεια μέσα από τις φυσικές συνδέσεις ενός δικτύου. Αντιθέτως, ένα 'embedded' worm προϋποθέτει επιπλέον για τη μετάδοσή του την ύπαρξη ενός καναλιού επικοινωνίας μεταξύ δύο χρηστών, όπου θα εισχωρήσει ανάμεσα σε κάποια μηνύματα ή θα πάρει τη θέση τους. Με αυτόν τον τρόπο, η εισβολή του δεν εμφανίζεται ως μία ιδιαίτερη ανωμαλία, από τη στιγμή που αποτελεί κομμάτι μιας φυσιολογικής κατά τ'άλλα επικοινωνίας.
- Άμυνα : πολλά worms διαθέτουν μηχανισμούς αυτοάμυνας που εμποδίζουν τον εντοπισμό τους
- Πολυμορφισμός : έχοντας την ικανότητα να παίρνουν κάθε φορά και διαφορετική μορφή, ισχυροποιούν τον μηχανισμό αυτοάμυνάς τους, καθώς καθιστούν ακόμα δυσκολότερο τον εντοπισμό και την καταστροφή τους.

***Εικόνα 1:*** *Ένα worm εξαπλώνεται μολύνοντας αρχικά έναν υπολογιστή, ενώ στη συνέχεια προσπαθεί να δημιουργήσει αντίγραφά του, μολύνοντας και άλλους υπολογιστές σε ένα δίκτυο-στόχο.*

Η επικινδυνότητα του κάθε worm μπορεί να κριθεί από 2 παράγοντες :

- η *μολυσματικότητα "virulence",* αφορά το πόσο γρήγορα μπορεί να εξαπλωθεί. Παράγοντες που επηρεάζονται συνήθως από τη δράση του είναι το εύρος των πόρων του δικτύου, η διαθέσιμη μνήμη των δρομολογητών και της Κεντρικής Μονάδας Επεξεργασίας (CPU), ο email server κ.α.

- το *ωφέλιμο φορτίο "payload"*, σχετίζεται με κάποια επιθετική, κακόβουλη ενέργεια, όπως η καταστροφή αρχείων, η αλλοίωση στοιχείων ενός web server κλπ. Πρόκειται για κάποια συγκεκριμένη μορφή κώδικα, ο οποίος αρχίζει να εκτελείται από τη στιγμή που το worm θα εγκατασταθεί σε κάποιο σύστημα. Οι συνέπειες που θα επιφέρει η εκτέλεση του κώδικα αυτού, έχουν να κάνουν με τους συγκεκριμένους σκοπούς του κάθε attacker. Πολλά worms δεν διαθέτουν payload και μοναδικός τους σκοπός είναι απλά να πολλαπλασιάζονται και να εξαπλώνονται.

### 2.1.2 Κίνητρα επιθέσεων μέσω των worms

Τα worms ξεκίνησαν σαν μία ιδέα δημιουργίας ενός προγράμματος που θα διέθετε τη δυνατότητα αυτο-αναπαραγωγής, αλλά έχουν καταλήξει να αποτελούν πλέον ένα ιδιαίτερα αποτελεσματικό μέσο εξαπόλυσης επιθέσεων. Ένα πρώτο βήμα στο σχεδιασμό συστημάτων αντιμετώπισης αυτών των "κακόβουλων" μηχανισμών, είναι η κατανόηση των λόγων που μπορεί να οδηγήσουν σε τέτοιου είδους επιθέσεις. Ενδεικτικά, αναφέρουμε κάποια από τα κίνητρα αυτά :

- Πειραματισμός και περιέργεια : πολλά από τα worms μαρτυρούν απλά τη σύλληψη μίας ιδέας η οποία κατέληξε στη δημιουργία τους. Χαρακτηριστικό παράδειγμα αποτελεί το Morris worm, το οποίο αν και δημιουργήθηκε χωρίς καμία κακόβουλη πρόθεση, κατέληξε να έχει ιδιαίτερα καταστροφικές συνέπειες.

- Εξουσία και φήμη : πολλοί από τους επιτιθέμενους χρήστες υποκινούνται από την επιθυμία να αποκτήσουν μία ευρεία φήμη και να επιδείξουν τις γνώσεις τους αλλά και την επικινδυνότητά τους.
- Οικονομικά κίνητρα και εμπορικό κέρδος : ένα άλλο πιθανό κίνητρο είναι η εξασφάλιση κέρδους μέσα από εγκληματικές ή εκβιαστικές ενέργειες. Σ'αυτή την περίπτωση ο στόχος είναι η οικονομική καταστροφή εταιριών που έχουν ως βάση τους το Διαδίκτυο. Ένα worm που στοχεύει στην υποκλοπή προσωπικών δεδομένων μέσω πιστωτικών καρτών πελατών των εταιριών αυτών, είναι ένα παράδειγμα μίας τέτοιας εγκληματικής ενέργειας.
- Πολιτικά – θρησκευτικά κίνητρα : σε κάποιες ακραίες περιπτώσεις, διάφορες ομάδες ατόμων, υποκινούμενων από πολιτικά ή θρησκευτικά κίνητρα, εκμεταλλεύονται το Διαδίκτυο προκειμένου να διαδώσουν ένα συγκεκριμένο μήνυμα ή σλόγκαν, ή να αποτρέψουν άλλες ομάδες από το να δημοσιοποιήσουν το δικό τους.
- Συγκέντρωση πολύτιμων ή απόρρητων πληροφοριών : κυβερνήσεις, εταιρίες και άλλοι φορείς που επιθυμούν να κερδίσουν πρόσβαση σε εμπιστευτικά δεδομένα, συχνά χρησιμοποιούν τεχνικές "hacking - backdoor install through worms" προκειμένου να εισέλθουν παράνομα σε βάσεις δεδομένων ή άλλες πηγές πολύτιμων πληροφοριών.

### 2.1.3 Μηχανισμοί εντοπισμού και αντιμετώπισης των worms

Καθώς η απειλή των worms παίρνει όλο και μεγαλύτερες διαστάσεις, παράλληλα αναπτύσσονται και νέες τεχνικές για την αντιμετώπισή τους. Οι περισσότερες από αυτές χαρακτηρίζονται ως "proactive" και έχουν ως βασικό στόχο τον έγκαιρο εντοπισμό τους, που με τη σειρά του θα οδηγήσει στην παρεμπόδιση της εξάπλωσής τους ή στην καταπολέμησή τους. Κάποιες από τις πιο διαδεδομένες τεχνικές έγκαιρου εντοπισμού των worms, μπορούν να θεωρηθούν οι εξής :

- Firewalls & routers : Τα τείχη προστασίας ή firewalls και οι δρομολογητές (routers) παρέχουν ένα είδος άμυνας για το δίκτυο στο οποίο ανήκουν, με το να ελέγχουν τα διάφορα πακέτα και να επιτρέπουν την πρόσβαση μόνο σε

όσα πληρούν συγκεκριμένες προϋποθέσεις. Με αυτόν τον τρόπο μπορεί να αποφευχθούν τα λεγόμενα "ύποπτα" πακέτα.

- Network-based intrusion detection systems (IDS) : πρόκειται για συστήματα εντοπισμού εισβολέων ( π.χ. SNORT), τα οποία μπορεί να είναι είτε "proactive" είτε "reactive". Βασίζονται σε κάποιους κοινοποιημένους κανόνες του συστήματος και ελέγχουν για τυχόν παραβιάσεις τους, οι οποίες μπορεί να αποτελούν ενδείξεις επιθετικών ενεργειών.

- Host -based IDS :
  - Checksum-based Detection : πρόκειται για τύπου "proactive" εργαλεία που παρέχουν αμυντικούς μηχανισμούς παρεμπόδισης των worms τα οποία προκαλούν αλλαγές σε αρχεία του σκληρού δίσκου. Διατηρούν βάσεις δεδομένων με "file signatures" για συγκεκριμένα αρχεία εφαρμογών του συστήματος, και συγκρίνουν τα περιεχόμενα των βάσεων αυτών με τα υπάρχοντα αρχεία, προκειμένου να εντοπίσουν τυχόν αλλαγές.
  - Signature-based Detection : τα εργαλεία anti-virus είναι τα πιο γνωστά εργαλεία που ανήκουν σε αυτή την κατηγορία. Βασίζονται σε κοινοποιημένες "virus signatures", προκειμένου να εντοπίσουν, να παρεμποδίσουν και να διαγράψουν worms και ιούς. Χαρακτηρίζονται συνεπώς ως τύπου "reactive" εργαλεία τα οποία βασίζονται σε γνωστούς τύπους επιθέσεων.

- Host-based intrusion prevention systems (IPS) : πρόκειται για έναν σχετικά νέο αλλά ιδιαίτερα αποτελεσματικό "proactive" μηχανισμό άμυνας. Η λειτουργία του βασίζεται στον έλεγχο των διαφόρων εφαρμογών του συστήματος, οι οποίες θα πρέπει να πληρούν συγκεκριμένους κανόνες. Μία πιθανή ενέργεια που δεν ικανοποιεί το σύνολο των προκαθορισμένων κανόνων, αυτόματα μπλοκάρεται και ενεργοποιείται ένα σύστημα συναγερμού. Σκοπός τους δεν είναι απλά η ανίχνευση δικτυακών επιθέσεων όπως συμβαίνει με τα IDS συστήματα, αλλά και η παρεμπόδισή τους ύστερα από τον έγκαιρο εντοπισμό τους.

- Tarpits and Honeynets : πρόκειται για μηχανισμούς που αν και δεν παρέχουν αυστηρή άμυνα ενάντια στα worms, ωστόσο αποτελούν ένα γενικότερο σύστημα προστασίας. Τα "tarpits" είναι σύνολα IP διευθύνσεων που φιλοξενούν υποτιθέμενα "ευπρόσβλητα" συστήματα, τα οποία χρησιμοποιούνται απλά για να προσελκύσουν τα πιθανά worms, παρεμποδίζοντας ταυτόχρονα την εξάπλωσή τους. Τα honeynets είναι ομάδες συστημάτων όπως servers, routers και firewalls που χρησιμοποιούνται για την παρατήρηση πραγματικών επιθέσεων. Και στις δύο περιπτώσεις, τα συστήματα εγκαθίστανται ξεχωριστά από το κανονικό δίκτυο, έτσι ώστε η κίνηση που δρομολογείται προς αυτά να θεωρείται σίγουρα μη αναμενόμενη. Στην ουσία, κανένα από τα δύο συστήματα δεν μπορεί να παρεμποδίσει τη δράση των worms, ωστόσο εάν χρησιμοποιηθούν σε συνδυασμό με κάποιο άλλο εργαλείο διαχείρισης επιθέσεων, μπορούν να λειτουργήσουν ως ανιχνευτές επιθετικών ενεργειών και να βοηθήσουν στη αποτελεσματική ανάλυση της φύσης των επιθέσεων.

Σκοπός των μελλοντικών αμυντικών μηχανισμών που θα αναπτυχθούν είναι, να καταφέρουν να συνδυάσουν την έγκαιρη ανίχνευση των worms με την αποτελεσματική διαχείρισή τους, ώστε να μπορούν να παρέχουν έναν γενικότερο μηχανισμό προστασίας στα ευάλωτα συστήματα όπου θα εφαρμόζονται. Ωστόσο, θα πρέπει να έχουμε υπόψη μας ότι κανένας μηχανισμός δεν μπορεί να εγγυηθεί απόλυτη προστασία, από τη στιγμή που μαζί με τις νέες τεχνολογίες που αναπτύσσονται, εμφανίζονται και νέες αδυναμίες για τα συστήματα, τις οποίες τα worms μπορούν εύκολα να εντοπίσουν και να εκμεταλλευτούν.

## 2.2 ΚΑΤΑΝΕΜΗΜΕΝΕΣ ΕΠΙΘΕΣΕΙΣ ΑΡΝΗΣΗΣ ΥΠΗΡΕΣΙΑΣ (Distributed Denial of service – DDoS attacks)

Οι Κατανεμημένες επιθέσεις Άρνησης Υπηρεσίας (Distributed Denial of Service attacks – DdoS attacks) αποτελούν πλέον ένα αρκετά διαδεδομένο και με ιδιαίτερα δυσάρεστες συνέπειες φαινόμενο του Διαδικτύου. Μία επίθεση DDoS εξαπολύεται με τη δημιουργία αθροιζόμενων ροών κίνησης προς δίκτυα ή υπολογιστές στόχους, με σκοπό τον κατακλυσμό των συνδέσεων που τα ενώνουν με το ευρύτερο δίκτυο και τη μετατροπή τους σε δίκτυα ανίκανα να παρέχουν τις υπηρεσίες που υπό φυσιολογικές συνθήκες προσφέρουν. Αυτή η διογκωμένη ροή κίνησης δεν αποτελεί προφανώς φυσιολογική και νόμιμη κίνηση προς το θύμα, αλλά υποκινείται από διεσπαρμένα στο Διαδίκτυο συστήματα, τα οποία έχουν τεθεί υπό "κακόβουλο" έλεγχο και δρουν σύμφωνα με τις οδηγίες του επιτιθέμενου.

Στην παραπάνω διαδικασία συντελούν ιδιαίτερα δύο βασικά χαρακτηριστικά του Διαδικτύου :

- Τα συστήματα από τα οποία απαρτίζεται το Internet (δικτυακές συσκευές και υπολογιστικά συστήματα), διαθέτουν μία περιορισμένη αλλά και καταναλώσιμη ποσότητα πόρων. Η χωρητικότητά τους, η ταχύτητα επεξεργασίας, αλλά και οι δυνατότητες αποθήκευσης είναι οι κύριοι στόχοι των DDoS επιθέσεων, που σχεδιάζονται προκειμένου να καταναλώσουν όσο το δυνατόν περισσότερους από τους διαθέσιμους πόρους του συστήματος, έτσι ώστε να προκαλέσουν μία γενικότερη αποδιοργάνωση στην απόκρισή του.
- Το Internet διαθέτει ένα ενιαίο σύστημα ασφάλειας, του οποίου τα επιμέρους στοιχεία είναι πλήρως αλληλοεξαρτώμενα και αλληλένδετα. Ανεξαρτήτως από το πόσο ασφαλής θεωρείται ο κάθε χρήστης, η πιθανότητα να δεχτεί κάποια DDoS επίθεση εξαρτάται από το επίπεδο ασφαλείας και των υπόλοιπων χρηστών του παγκόσμιου Internet. Αυτό σημαίνει ότι κάποιος "κακόβουλος" χρήστης μπορεί να εκμεταλλευτεί τις πηγές ενός ξένου συστήματος το οποίο υπόκειται σε κάποιους κανόνες ασφαλείας, έτσι ώστε να ξεκινήσει κάποια επίθεση από το σύστημα αυτό (που θεωρείται ασφαλές) προς ένα άλλο.

  Στόχος λοιπόν είναι, πέρα από την προστασία του συστήματος απέναντι σε πιθανές επιθέσεις που μπορεί να δεχτεί και η παρεμπόδιση χρήσης των πόρων του ίδιου συστήματος για εξαπόλυση επιθέσεων προς άλλες κατευθύνσεις.

### 2.2.1 Στρατηγική DDoS επιθέσεων

Επίθεση DDoS έχουμε, όταν ένας αριθμός από ελεγχόμενα μηχανήματα (agents) παράγουν μεγάλο όγκο κίνησης άχρηστων πακέτων προς το θύμα, με στόχο να καταβάλουν τους πόρους του (bandwidth). Είναι λοιπόν προφανές, ότι στις επιθέσεις DDoS μπορεί να περιλαμβάνονται περισσότεροι από ένας επιτιθέμενοι χρήστες και πιθανώς περισσότερα από ένα μηχανήματα – στόχοι. Οι επιθέσεις αυτές καταλήγουν να είναι ιδιαίτερα αποτελεσματικές χρησιμοποιώντας την κατανεμημένη υποδομή του Διαδικτύου. Κατευθύνοντας πολλαπλές "επιθετικές" ροές κίνησης προς μία περιοχή του δικτύου, είναι δυνατό οι συνδέσεις, ακόμα και αν είναι υψηλής χωρητικότητας, να κατακλυστούν και να σταματήσουν να είναι διαθέσιμες για κανονική χρήση. Αυτή η μέθοδος είναι κοινώς γνωστή ως **"packet flooding attacks".**

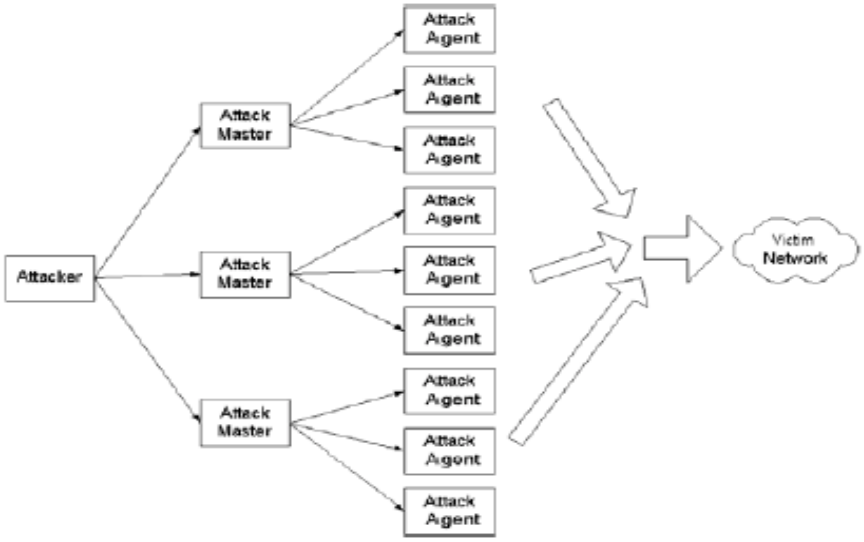
Προκειμένου να πραγματοποιήσει μία DDoS επίθεση ένας attacker, θα πρέπει να ανιχνεύσει το δίκτυο για τον εντοπισμό μεμονωμένων μηχανημάτων που εμφανίζουν διάφορες αδυναμίες ή κενά στην ασφάλεια. Στη συνέχεια, θα εκμεταλλευτεί τις συγκεκριμένες αδυναμίες των "ευάλωτων" μηχανημάτων που έχουν εντοπιστεί προκειμένου να αποκτήσει πρόσβαση σε αυτά και να εγκαταστήσει τους μηχανισμούς επίθεσης.

Πιο αναλυτικά, τα βήματα που ακολουθούνται κατά την εξαπόλυση μίας DDoS επίθεσης είναι τα εξής :

- **Βήμα 1 :** οι επιτιθέμενοι χρήστες διεισδύουν σε έναν αριθμό μηχανημάτων και εγκαθιστούν κώδικα ελέγχου μικρού μεγέθους, με αντίκτυπο στην κανονική λειτουργία του δικτύου ("footprint" κώδικας). Αυτά τα μηχανήματα είναι πλέον οι masters πρώτου επιπέδου.
- **Βήμα 2 :** οι masters πρώτου επιπέδου οργανώνουν την υποδομή της επίθεσης. Μετά από μια διαδικασία διερεύνησης (scanning) διαφόρων μηχανημάτων (από προσωπικούς υπολογιστές μέχρι μεγάλα συστήματα) για γνωστές αδυναμίες, αναγνωρίζουν τα τρωτά μηχανήματα και εγκαθιστούν σε αυτά τον κώδικα που θα παράγει την επίθεση. Αυτά τα μηχανήματα ελέγχονται από τον επιτιθέμενο και ονομάζονται "agents" της επίθεσης, ενώ όσο τα

κακόβουλα προγράμματά τους δεν είναι σε δράση ονομάζονται “zombies” ή “bots”.

- **Βήμα 3 :** μετά από εντολή του επιτιθέμενου, οι agents εξαπολύουν την επίθεση ενεργοποιώντας μεγάλη ροή πακέτων προς το δίκτυο-θύμα. Αυτές οι ροές πακέτων μπορεί να είναι μεμονωμένες και κοντά στην πηγή τους να είναι σχετικά μικρές, αλλά αθροιζόμενες κοντά στο δίκτυο θύμα, καταλήγουν να καταβάλουν σημαντικό τμήμα του διαθέσιμου δικτυακού εύρους (bandwidth).




***Εικόνα 2 :*** *Πολλαπλά επίπεδα μίας DDoS επίθεσης*

Τα πακέτα που αποστέλλονται κατά την εξαπόλυση μίας DDoS επίθεσης ποικίλλουν, αλλά τα πιο συνηθισμένα είναι τα εξής :

- TCP floods (με χρήση των flags SYN, ACK και RST)
- ICMP echo request/reply
- UDP floods

Μερικά εργαλεία επιθέσεων φροντίζουν να τροποποιήσουν διάφορες παραμέτρους των πακέτων που αποστέλλονται προκειμένου να δημιουργήσουν μια γενικότερη σύγχυση. Παράμετροι που μεταβάλλονται συνήθως είναι οι εξής :

- **Source IP address** : μέσω της γνωστής μεθόδου **“IP spoofing”**, μεταβάλλονται τα πεδία μιας σειράς πακέτων που αφορούν τη διεύθυνση του αποστολέα. Σε

πολλές περιπτώσεις, αυτό αποσκοπεί στην έμμεση επίθεση προς έναν επιθυμητό στόχο – θύμα, μέσω ενός ενδιάμεσου κόμβου, ο οποίος θα λάβει τα λανθασμένα πακέτα και θα προκαλέσει με τη σειρά του απαντήσεις στο θύμα το οποίο θα εμφανίζεται ως ο ψεύτικος αποστολέας.

Η τεχνική αυτή των επιθέσεων, πέραν του ότι καταλήγει στην 'άρνηση υπηρεσίας' από το θύμα, έχοντας καταναλώσει τους περισσότερους πόρους του (resources), δίνει στους hackers δύο επιπλέον δυνατότητες : πρώτον, εξασθενίζει τη δυνατότητα μετριασμού της επίθεσης, καθώς η κακόβουλη κίνηση πλέον δεν μπορεί να κατηγοριοποιηθεί με βάση την πηγή της και έτσι δεν είναι εύκολο να εντοπιστεί. Δεύτερον, καθιστά πιο περίπλοκη τη διαδικασία εντοπισμού των hackers και τη λήψη κατάλληλων μέτρων, καθώς η διαδρομή που ακολουθήθηκε από τα επιθετικά πακέτα δεν οδηγεί στη σωστή πηγή και έτσι δεν εξυπηρετεί στον εντοπισμό τους.

- **Source/destination ports** : διάφορα εργαλεία επιθέσεων που βασίζονται σε TCP και UDP πακέτα μεταβάλλουν τους αριθμούς της πόρτας πηγής ή προορισμού, προκειμένου να κάνουν την αντίδραση του συστήματος κατά το φιλτράρισμα των πακέτων πιο δύσκολη.

- **Άλλες επικεφαλίδες IP πακέτων** : υπάρχουν εργαλεία που μεταβάλλουν ακόμη και όλες τις παραμέτρους μιας σειράς πακέτων, εκτός μόνο από τη διεύθυνση προορισμού.

### 2.2.2 Είδη DDoS flooding attacks

Οι DDoS επιθέσεις οι οποίες εφαρμόζουν την παραπάνω στρατηγική μπορούν να διακριθούν σε κάποιες επιμέρους κατηγορίες :

- DIRECT ATTACKS

  Σε μία άμεση επίθεση, ο δράστης σχεδιάζει να αποστείλει ένα μεγάλο αριθμό πακέτων (TCP, ICMP, UDP) απευθείας προς το θύμα. Στην περίπτωση των TCP πακέτων, η πιο γνωστή μέθοδος είναι η **"SYN flooding attack"**, σύμφωνα με την οποία ένας μεγάλος αριθμός TCP SYN πακέτων αποστέλλονται προς τη θύρα

όπου ακούει ο server του θύματος. Εάν ο server αυτός δέχεται αιτήσεις για συνδέσεις με άλλους χρήστες, τότε θα απαντήσει με αποστολή SYN-ACK πακέτων. Τα πακέτα αυτά όμως συνήθως αποστέλλονται προς λανθασμένες διευθύνσεις του Internet, εφόσον το πεδίο του source address είναι αλλοιωμένο λόγω address spoofing. Για το λόγο αυτό, το θύμα αναμεταδίδει συνεχώς τα SYN-ACK πακέτα, με αποτέλεσμα να καταναλώνει ένα μεγάλο μέρος της μνήμης και των πόρων που διαθέτει για τη δημιουργία συνδέσεων. Αυτό εμποδίζει πλέον το server να δεχτεί νέες αιτήσεις για συνδέσεις.

Μία άλλη μέθοδος που βασίζεται σε TCP πακέτα, είναι η δημιουργία συμφόρησης στην κατεύθυνση εισόδου μιας σύνδεσης του θύματος. Σε αυτήν την περίπτωση, το θύμα απαντάει με RST πακέτα (RST flooding). Το ίδιο συμβαίνει και στην περίπτωση ICMP και UDP πακέτων, όπου το θύμα απαντάει με αποστολή ICMP reply και error μηνυμάτων (ICMP flooding).

Πριν την εξαπόλυση μίας άμεσης επίθεσης, θα πρέπει να εγκατασταθεί ένα σύστημα DDos επιθέσεων (attack network), το οποίο θα αποτελείται από έναν ή περισσότερους attacking hosts, έναν αριθμό από masters ή handlers και ένα μεγάλο αριθμό από agents, που αναφέρονται συνήθως ως daemons ή zombies. Ο attacking host είναι ένα καθορισμένο μηχάνημα που χρησιμοποιείται για τον εντοπισμό ευάλωτων συστημάτων τα οποία θα αποτελέσουν τα πιθανά θύματά του, αλλά και για την εγκατάσταση συγκεκριμένων DDoS master και agent προγραμμάτων. Από τη στιγμή που θα εγκατασταθεί το DDoS σύστημα επιθέσεων, ο attacking host μπορεί να δώσει εντολή για επίθεση προς τη διεύθυνση του θύματός του, καθορίζοντας διάφορες παραμέτρους, όπως η διάρκεια της επίθεσης, η μέθοδος που θα χρησιμοποιηθεί και διάφορες άλλες εντολές προς τους masters.

Σήμερα, τα διάφορα εργαλεία επιθέσεων που χρησιμοποιούνται, μπορούν να εξαπολύσουν επιθέσεις προς περισσότερα θύματα ταυτόχρονα, με χρήση διαφόρων τύπων πακέτων (multiple attacks).

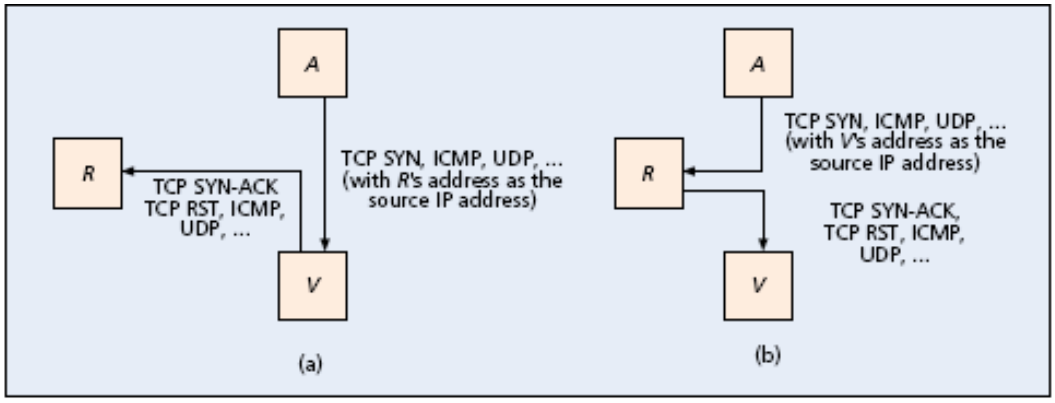
- REFLECTOR ATTACKS

Πρόκειται για μία έμμεση μορφή επίθεσης, όπου ενδιάμεσοι κόμβοι όπως διάφοροι routers και servers χρησιμοποιούνται ως reflectors, δηλαδή ως έμμεσοι υπεύθυνοι των επιθέσεων που θα εξαπολυθούν από αυτούς εν αγνοία τους. Πιο

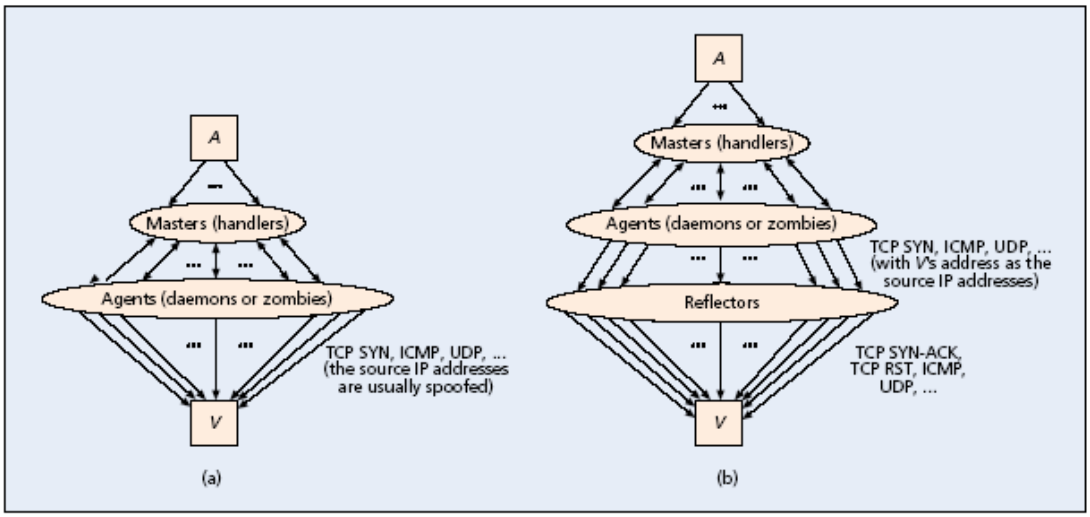
συγκεκριμένα, οι attackers στέλνουν πακέτα προς τους κόμβους-reflectors, τα οποία απαιτούν απαντήσεις προς τις διευθύνσεις που έχουν τεθεί στο πεδίο του source address. Αυτές είναι και οι διευθύνσεις των υποψήφιων θυμάτων. Με αυτόν τον τρόπο και χωρίς να ξέρουν οι reflectors ότι τα πακέτα που λαμβάνουν έχουν υποστεί address spoofing, στέλνουν άλλα πακέτα ως απάντηση προς τους κόμβους-θύματα, με βάση τον τύπο των αρχικών πακέτων. Συνεπώς, τα πακέτα επίθεσης αποστέλλονται από τους reflectors προς το θύμα με τη μορφή κανονικών πακέτων και αν ο αριθμός τους είναι μεγάλος, τότε μπορούν να πλημμυρίσουν το θύμα και να δημιουργήσουν έτσι μεγάλη συμφόρηση στις συνδέσεις του με το υπόλοιπο δίκτυο. Ο τύπος των πακέτων που αποστέλλεται προς τους reflectors μπορεί να ποικίλλει και κατά συνέπεια ποικίλλει και ο τύπος των πακέτων που στέλνονται ως απάντηση από τους reflectors προς το θύμα. Στην επόμενη εικόνα παρουσιάζονται οι πιθανές περιπτώσεις :

| | Packets sent by an attacker to a reflector (with a victim's address as the source address) | Packets sent by the reflector to the victim in response |
|---|---|---|
| Smurf | ICMP echo queries to a subnet-directed broadcast address | ICMP echo replies |
| SYN flooding | TCP SYN packets to public TCP servers (e.g., Web servers) | TCP SYN-ACK packets |
| RST flooding | TCP packets to nonlistening TCP ports | TCP RST packets |
| ICMP flooding | • ICMP queries (usually echo queries)<br>• UDP packets to nonlistening UDP ports<br>• IP packets with low TTL values | • ICMP replies (usually echo replies)<br>• ICMP port unreachable messages<br>• ICMP time exceeded messages |
| DNS reply flooding | DNS (recursive) queries to DNS servers | DNS replies (usually much larger than DNS queries) |

***Εικόνα 3 :*** *Μία σύνοψη των πιο κοινών χρησιμοποιούμενων μεθόδων από τις "reflector attacks"* [6]




***Εικόνα 4 :*** *Δύο τύποι " flooding-based DDoS attacks" : a)direct, b)reflector* [6]

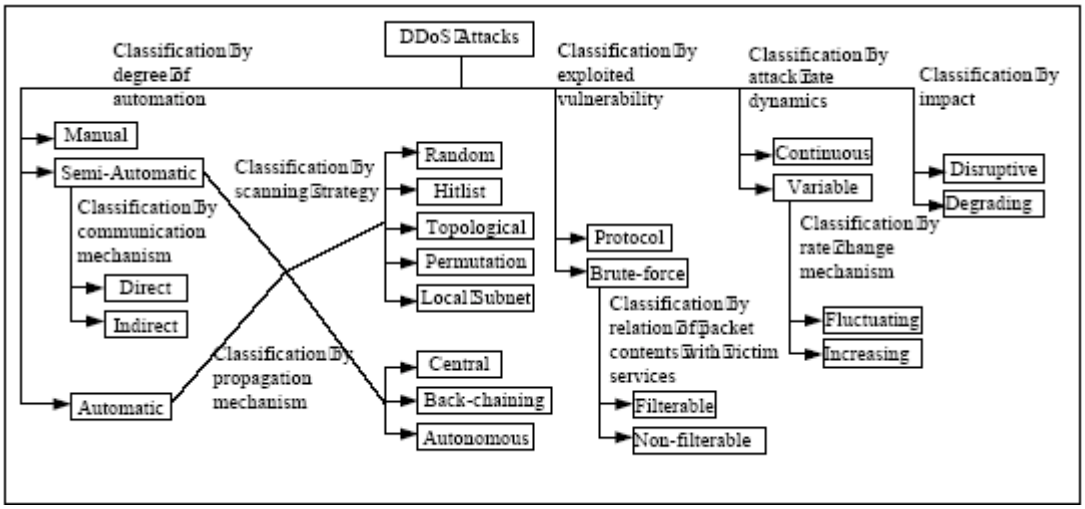



***Εικόνα 5 :*** *Αρχιτεκτονική των DDoS attacks για α)direct και b)reflector attacks* [6]

Σύμφωνα με το παραπάνω σχήμα, η αρχιτεκτονική των direct επιθέσεων είναι παρόμοια με αυτή των reflector επιθέσεων. Ωστόσο υπάρχουν βασικές διαφορές. Για παράδειγμα, μία reflector attack απαιτεί την ύπαρξη ενός καθορισμένου συνόλου από reflectors, όπως για παράδειγμα DNS servers, HTTP servers και routers. Η έκταση που θα πάρει η επίθεση εξαρτάται από το σύνολο των reflectors (και όχι από το σύνολο των agents όπως στις direct επιθέσεις), καθώς και από τη συχνότητα και το μέγεθος των reflected packets. Οι reflectors μπορούν να είναι διασκορπισμένοι σε διάφορα σημεία του Internet και η ύπαρξη των agents πλέον δεν είναι αναγκαία. Επιπλέον, τα πακέτα που προέρχονται από τους reflectors έχουν κανονική μορφή με πραγματικές διευθύνσεις πηγής, σε αντίθεση με τα πακέτα των direct επιθέσεων που έχουν υποστεί address spoofing και για

το λόγο αυτό είναι πιο εύκολο να φιλτραριστούν και να εντοπιστούν από ανάλογους μηχανισμούς.

Ο παρακάτω πίνακας δείχνει μία περαιτέρω ταξινόμηση των DDoS attacks :




***Εικόνα 6 :*** *Ταξινόμηση των κατανεμημένων DDoS επιθέσεων* [4]

### 2.2.3 Συστήματα εντοπισμού και αντιμετώπισης DDoS επιθέσεων

Η σοβαρότητα και η συχνότητα των προβλημάτων που ανακύπτουν από τις DDoS επιθέσεις, έχει οδηγήσει στην εμφάνιση ποικίλων τεχνικών άμυνας και μηχανισμών αντιμετώπισής τους. Κάποιες από αυτές εστιάζουν σε ένα συγκεκριμένο είδος επίθεσης, ενώ άλλες προσπαθούν να αντιμετωπίσουν το πρόβλημα των DDoS επιθέσεων γενικότερα.

Σε μία αρχική προσπάθεια προσέγγισης των τεχνικών αυτών, μπορούμε να διακρίνουμε τρεις κύριες μορφές άμυνας ενάντια στις DDoS επιθέσεις :

- Πρόληψη και παρεμπόδιση επιθέσεων (attack prevention and preemption –before the attack)

  Σε αυτή την περίπτωση άμυνας, ο βασικός στόχος είναι να εμποδιστούν οι DDoS επιθέσεις πριν προλάβουν να εμφανιστούν. Για το λόγο αυτό θα πρέπει

τα συστήματα των χρηστών να είναι πλήρως προστατευμένα ή να υπάρχουν ειδικά προγράμματα που θα εντοπίζουν έγκαιρα και θα διαχειρίζονται κατάλληλα την επιθετική κίνηση του δικτύου τους.

Πρόσφατα, έχουν προταθεί διάφορες μέθοδοι που προσβλέπουν σε ένα γενικότερο μηχανισμό άμυνας, ένα "internet firewall", που θα μπορεί να προστατεύει ολόκληρο το Internet από DDoS attacks. Σκοπός αυτού του "internet firewall" είναι να εντοπίζει τέτοιου είδους επιθέσεις στον κορμό του Internet και να απορρίπτει τα ύποπτα πακέτα προτού φτάσουν στα θύματά τους. Εάν εφαρμοστεί σωστά και αποτελεσματικά αυτή η προσέγγιση, έχει τη δυνατότητα να διατηρήσει τη λειτουργία του θύματος σε κανονικά επίπεδα, ακόμα και κατά τη διάρκεια μίας επίθεσης.

Ωστόσο, η μέθοδος αυτή δεν είναι η καλύτερη δυνατή από τη στιγμή που υπάρχουν πολλοί απρόσεχτοι χρήστες που αφήνουν τα συστήματά τους ευάλωτα απέναντι σε DDoS agents, ενώ παράλληλα οι διάφοροι ISP's δεν προτίθενται να διαχειριστούν τυχόν επιθετική κίνηση. Τέλος, τα προγράμματα εντοπισμού απαιτούν εις βάθος γνώση όλων των πιθανών επιθετικών μεθόδων, κάτι που είναι ιδιαίτερα δύσκολο, λόγω της πολυπλοκότητας αλλά και της εύκολης τροποποίησης των μεθόδων αυτών.

- <u>Εντοπισμός επιθέσεων και φιλτράρισμα ( attack detection and filtering – during the attack)</u>

  Στην πρώτη φάση της μεθόδου αυτής, σκοπός είναι ο εντοπισμός DDoS πακέτων. Μετά την αναγνώριση της κίνησης που δημιουργείται από τα attack packets και σε δεύτερη φάση, ειδικά φίλτρα αναλαμβάνουν να απομονώσουν και στη συνέχεια να απορρίψουν τα πακέτα αυτά.

  Σε μία ιδανική περίπτωση, οι επιθέσεις θα μπορούσαν να παρεμποδιστούν όσο το δυνατόν πιο κοντά στην πηγή τους. Παράδειγμα ενός αμυντικού συστήματος που έχει αναπτυχθεί βασιζόμενο στη λογική αυτή είναι το D-WARD, το οποίο εφαρμόζεται στην αφετηρία των συνδέσεων ενός δικτύου και εμποδίζει τους χρήστες του από τη συμμετοχή τους σε DDoS επιθέσεις. Το D-WARD καθορίζει αρχικά ένα σύνολο διευθύνσεων (police address set) των οποίων η εξερχόμενη κίνηση υπόκειται σε συγκεκριμένους κανόνες ελέγχου και στη συνέχεια διαχειρίζεται την κίνηση μεταξύ αυτού του

συνόλου και των υπόλοιπων διευθύνσεων του Internet και προς τις δύο κατευθύνσεις. Σκοπός του είναι να προσφέρει καλές και αξιόπιστες υπηρεσίες προς τη νόμιμη κίνηση ακόμα και κατά τη διάρκεια μίας επίθεσης και ταυτόχρονα να μειώσει σε αμελητέα επίπεδα την κίνηση DDoS. Το σύστημα D-WARD εγκαθίσταται στο δίκτυο (source network) σαν ένας source router που χρησιμεύει σαν δίοδος μεταξύ του συγκεκριμένου δικτύου και του υπόλοιπου Internet. Διαχειρίζεται τη συμπεριφορά κάθε σταθμού με τον οποίο επικοινωνεί το source network και ψάχνει για σημάδια που προδίδουν τυχόν δυσκολίες στην επικοινωνία, όπως η μείωση στον αριθμό των πακέτων απόκρισης ή η αύξηση του χρόνου που μεσολαβεί ανάμεσα στις αφίξεις των πακέτων. Το D-WARD κάνει περιοδικές συγκρίσεις μεταξύ των στατιστικών στοιχείων που προέκυψαν από τις παρατηρήσεις του προς κάθε κατεύθυνση και ενός προκαθορισμένου συνόλου τιμών που αφορά την κανονική κίνηση. Εάν η σύγκριση αποκαλύψει την πιθανότητα μίας DDoS επίθεσης, το D-WARD αποκρίνεται με την επιβολή ενός φραγμού (rate limit) στην ύποπτη εξερχόμενη κίνηση από το συγκεκριμένο σταθμό. Οι περαιτέρω παρατηρήσεις είτε θα επιβεβαιώσουν είτε θα διαψεύσουν την προηγούμενη υπόθεση και το D-WARD θα αυξομειώσει ανάλογα το επιβαλλόμενο rate limit.

Η τοποθέτηση αμυντικών DDoS συστημάτων (όπως το D-WARD) κοντά στην πηγή των επιθέσεων έχει πολλά πλεονεκτήματα. Η επιθετική κίνηση μπορεί να παρεμποδιστεί προτού να εισέλθει στον πυρήνα του Internet και αναμειχθεί με την υπόλοιπη κανονική κίνηση, δημιουργώντας έτσι επιπλέον συμφόρηση. Με τον τρόπο αυτό διευκολύνεται επίσης η ανίχνευση και η αναγνώριση των πηγών προέλευσης της κάθε επίθεσης και οι routers που είναι τοποθετημένοι κοντά σε αυτές έρχονται αντιμέτωποι με μικρότερη κίνηση σε σχέση με τους κεντρικούς routers. Για το λόγο αυτό μπορούν να διαθέσουν περισσότερους από τους πόρους τους στην αντιμετώπιση των DDoS επιθέσεων.

Ωστόσο, η μέθοδος αυτή δεν είναι πάντοτε αποτελεσματική, διότι πολύ συχνά εκτός από τα πακέτα επίθεσης, απορρίπτονται και κανονικά πακέτα εφόσον και τα δύο περιέχουν signatures που ταιριάζουν στα στοιχεία του θύματος.

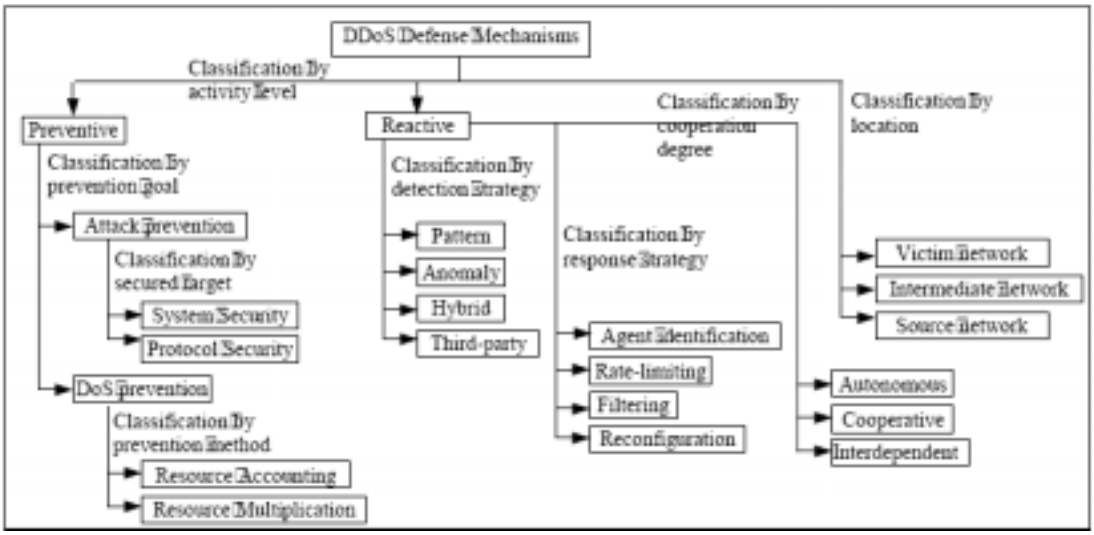
- Εξιχνίαση και αναγνώριση της πηγής προέλευσης των επιθέσεων (attack source traceback and identification – during and after the attack)

  Η μέθοδος αυτή εφαρμόζεται μετά από μία DDoS επίθεση και απώτερος στόχος της είναι ο εντοπισμός της πραγματικής πηγής των επιθέσεων, χωρίς να βασίζεται όμως στις πληροφορίες σχετικά με την πηγή που μεταφέρουν τα πακέτα, καθώς αυτή μπορεί να έχει υποστεί αλλοίωση.

  Ωστόσο, δεν είναι πάντοτε εφικτό να εντοπιστούν οι πραγματικές πηγές των attack packets, ιδιαίτερα στην περίπτωση των reflector attacks όπου τα πακέτα φαίνονται να προέρχονται από νόμιμες διευθύνσεις. Ακόμη όμως και αν οι πηγές εντοπιστούν, δεν είναι πάντα εύκολο να μπλοκαριστούν από την αποστολή πακέτων, ειδικά στην περίπτωση που είναι διασκορπισμένες σε διαφορετικά αυτόνομα συστήματα (AS).

  Παρ'όλα αυτά όμως, η μέθοδος "IP traceback" αποδεικνύεται πολύ χρήσιμη στον εντοπισμό των attackers και στη συγκέντρωση πληροφοριών για χρήση από τις νόμιμες αρχές προς επιβολή ποινών.

Μία περισσότερο εις βάθος προσέγγιση στηριγμένη σε πιο αναλυτικά κριτήρια, μας επιτρέπει να κάνουμε την παρακάτω ταξινόμηση των αμυντικών μηχανισμών :




***Εικόνα 7 :*** *Ταξινόμηση των αμυντικών μηχανισμών για κατανεμημένες DDoS επιθέσεις* [4]

Γενικά, για την αντιμετώπιση μίας επίθεσης DDoS, είναι αναγκαία η συνεργασία μεταξύ των διαφόρων δικτυακών τόπων (sites), αφού η φύση της επίθεσης εμποδίζει την αποτελεσματική αντιμετώπισή της από ένα μεμονωμένο δικτυακό τόπο. Το IP spoofing, η εξάπλωση της επίθεσης σε πολλά δίκτυα-στόχους και η αδυναμία ενός δικτύου να διαμορφώσει τον όγκο της εισερχόμενης κίνησης (traffic shaping) καθιστούν μη αποτελεσματικές τις προσπάθειες ενός και μόνο δικτυακού τόπου να αντιμετωπίσει την επίθεση.

Η συνεργασία ανάμεσα στους δικτυακούς τόπους μπορεί να λάβει χώρα με τις ακόλουθες ενέργειες :

α) καθορισμός των χαρακτηριστικών της επίθεσης (χρησιμοποιούμενο πρωτόκολλο, ports κ.α.) σε κάθε σημείο κατά μήκος της διαδρομής της επίθεσης, έτσι ώστε να μπορούν να τεθούν σε λειτουργία τα κατάλληλα φίλτρα αντιμετώπισης,

β) διάδοση αυτών των χαρακτηριστικών σε όλα τα δίκτυα που βρίσκονται πάνω στο μονοπάτι της επίθεσης και επικοινωνία ανάμεσα στο θύμα και στα δίκτυα προέλευσης της κίνησης, για έλεγχο της αποτελεσματικότητας των φίλτρων (προσαρμογή τους στα εναλλασσόμενα μοτίβα επιθέσεων).

# ΚΕΦΑΛΑΙΟ 3 : ΑΝΙΧΝΕΥΣΗ ΔΙΚΤΥΑΚΩΝ ΑΝΩΜΑΛΙΩΝ ΚΑΙ ΕΝΤΟΠΙΣΜΟΣ ΕΠΙΘΕΣΕΩΝ

## 3.1 ΔΙΑΚΡΙΣΗ ΤΩΝ ΔΙΑΦΟΡΩΝ ΠΡΟΣΕΓΓΙΣΕΩΝ ΓΙΑ ΤΟΝ ΕΝΤΟΠΙΣΜΟ ΔΙΚΤΥΑΚΩΝ ΑΝΩΜΑΛΙΩΝ

Μέχρι στιγμής έχουν αναπτυχθεί δύο βασικές προσεγγίσεις σχετικά με την ανίχνευση επιθέσεων :

- Η πρώτη προσέγγιση είναι γνωστή ως **"anomaly detection"** και σκοπός της είναι ο καθορισμός και ο χαρακτηρισμός ενός σωστού και αποδεκτού τρόπου λειτουργίας και δυναμικής συμπεριφοράς τους συστήματος, με βάση τον οποίο μπορεί να εντοπιστεί οποιαδήποτε λανθασμένη συμπεριφορά ή μεταβολή που αποκλίνει από αυτόν. Όπως είναι φανερό, η προσέγγιση αυτή βασίζεται στη διάκριση μεταξύ της επιθυμητής από τη μη αποδεκτή συμπεριφορά ενός συστήματος, που μπορεί να χαρακτηριστεί ως μία ανωμαλία. Το όριο διάκρισης μεταξύ της αποδεκτής και της ανώμαλης μορφής ενός καταχωρημένου κώδικα είναι πλήρως καθορισμένο. Ακόμη και ένα διαφορετικό bit μπορεί να υποδείξει κάποιο πρόβλημα. Αντιθέτως, το όριο διάκρισης μεταξύ αποδεκτής και ανώμαλης συμπεριφοράς είναι πολύ πιο δύσκολο να καθοριστεί.

- Η δεύτερη προσέγγιση, το **"misuse detection",** αναφέρεται σε γνωστές μεθόδους εισβολής σε ένα σύστημα, καθεμία από τις οποίες ονομάζεται σαν ένα αναγνωρίσιμο "πρότυπο". Ένα **"misuse detection system"** ερευνά για συγκεκριμένα πρότυπα. Τα πρότυπα αυτά μπορεί να ποικίλουν. Για παράδειγμα, μπορεί να έχουν να κάνουν με ένα συγκεκριμένο bit που υποδεικνύει κάποιον ιό ή με μία σειρά ενεργειών που θεωρούνται ύποπτες.

Τα μέχρι τώρα "intrusion detection systems" έχουν κατασκευαστεί έτσι ώστε να εφαρμόζουν και τις δύο προσεγγίσεις ξεχωριστά ή και σε συνδυασμό μεταξύ τους (hybrid systems).

### 3.1.1 Anomaly Detection

Όπως ήδη αναφέρθηκε, ένα σύστημα εντοπισμού ανωμαλιών πρέπει να έχει τη δυνατότητα να διακρίνει μεταξύ μίας φυσιολογικής και μίας ανώμαλης κατάστασης. Υπάρχουν δύο τύποι συστημάτων εντοπισμού ανωμαλιών :

- Η **"static anomaly detection"** βασίζεται στην υπόθεση ότι υπάρχει ένα κομμάτι του διαχειριζόμενου συστήματος που θα πρέπει να παραμένει σταθερό. Το κομμάτι αυτό αφορά ένα μέρος κώδικα και ένα μέρος δεδομένων που θεωρείται ότι παραμένουν σταθερά και αντιπροσωπεύονται από μία σειρά δυαδικών ψηφίων (strings and files). Η αναπαράσταση αυτής της σταθερής κατάστασης που συνήθως αποθηκεύεται σαν μία συμπιεσμένη ακολουθία bit strings ονομάζεται *signature*. Τα signatures περιέχουν συνήθως checksums, message-digest algorithms and hash functions.

Οι "static anomaly detectors", ελέγχουν για την ακεραιότητα και την πληρότητα της κατάστασης του συστήματος που θεωρείται σταθερή, κάνοντας περιοδικές συγκρίσεις μεταξύ αυτής και των εκάστοτε δεδομένων (bit strings) που προκύπτουν από τις διαδοχικές μετρήσεις. Οποιαδήποτε παρέκκλιση από την αρχική κατάσταση, υποδεικνύει είτε κάποιο σφάλμα του hardware, είτε κάποιο χρήστη που εισέβαλε στο σύστημα και προκάλεσε μεταβολές.

Η μέθοδος της **"dynamic anomaly detection"** προϋποθέτει την ύπαρξη μίας καθορισμένης συμπεριφοράς του συστήματος, η οποία αντιπροσωπεύεται από μία σειρά διακριτών γεγονότων τα οποία καθορίζουν ένα βασικό προφίλ (base profile). Το προφίλ αυτό αποτελείται από διάφορες παρατηρούμενες παραμέτρους και θεωρείται ότι χαρακτηρίζει τη φυσιολογική, ομαλή συμπεριφορά του συστήματος.

Ύστερα από την αρχικοποίηση του βασικού προφίλ, η ανίχνευση επιθέσεων μπορεί να ξεκινήσει. Οι dynamic detectors, παρόμοια με τους static detectors, ερευνούν περιοδικά για διαφορές μεταξύ της παρατηρούμενης συμπεριφοράς του συστήματος και της αναμενόμενης, η οποία στηρίζεται στο βασικό προφίλ. Παράλληλα,

διαμορφώνουν δυναμικά το νέο προφίλ του συστήματος (current profile) με βάση τα νέα παρατηρούμενα στοιχεία. Με τον τρόπο αυτό, προσπαθούν να εξάγουν συμπεράσματα και να εντοπίσουν διάφορα "ενδιαφέροντα" γεγονότα που μπορεί να αποτελούν είδη επιθέσεων. Για παράδειγμα, πολλά συστήματα εντοπισμού επιθέσεων χρησιμοποιούν σειρές καταγεγραμμένων στοιχείων κίνησης που παράγονται από τα αντίστοιχα λειτουργικά συστήματα, μέσα από τα οποία προσπαθούν να εντοπίσουν γεγονότα "ενδιαφέροντος" που συνιστούν μία συγκεκριμένη ύποπτη συμπεριφορά. Τα πιο εξελιγμένα συστήματα, δημιουργούν μία ξεχωριστή βάση για γεγονότα που χαρακτηρίζουν μία ανωμαλία, την οποία χρησιμοποιούν στη συνέχεια ως αναφορά για τη σύγκριση με τα παρατηρούμενα δεδομένα. Συνήθως και σε αντίθεση με τη "static anomaly detection", το όριο ανάμεσα στη φυσιολογική και στην "ανώμαλη" συμπεριφορά δεν είναι απόλυτα διακριτό. Η φυσιολογική συμπεριφορά καθορίζεται με βάση τις τιμές διαφόρων παραμέτρων κατά την αρχικοποίηση του συστήματος, οι οποίες χρησιμοποιούνται στη συνέχεια ως σημείο αναφοράς για σύγκριση και για την εξαγωγή των κατάλληλων συμπερασμάτων. Εάν μία αμφίβολη συμπεριφορά δε χαρακτηριστεί ως "ανώμαλη" , τότε μία απόπειρα εισβολής στο σύστημα πιθανόν να μην εντοπιστεί. Σε αντίθετη περίπτωση, όπου μία συμπεριφορά χαρακτηριστεί ανώμαλη χωρίς πραγματικά να είναι, τότε οι διαχειριστές του συστήματος θα ειδοποιηθούν από ψευδή μηνύματα συναγερμού, χωρίς να υπάρχει κάποια επίθεση. Ο πιο συνήθης τρόπος για τον καθορισμό αυτού του ορίου βασίζεται σε στατιστικά στοιχεία με συγκεκριμένη μέση τιμή και τυπική απόκλιση και σε εμπειρικά δεδομένα.

Τέλος, ένα επιπλέον πρόβλημα που προκύπτει για τα dynamic anomaly detection systems, είναι ότι καλούνται να δημιουργήσουν βασικά προφίλ με ακριβή στοιχεία, ώστε στη συνέχεια να μπορούν να αναγνωρίζουν τις αποκλίνουσες από αυτά συμπεριφορές. Η κατασκευή αυτών των προφίλ μπορεί να γίνει είτε δυναμικά κατά τη διάρκεια της λειτουργίας του συστήματος, είτε με παρατήρηση της φυσιολογικής συμπεριφοράς ενός άλλου συστήματος για ένα επαρκές χρονικό διάστημα.

Οι τεχνικές που βασίζονται στην προσέγγιση της **anomaly detection** εφαρμόζουν τόσο τη στατική όσο και τη δυναμική μέθοδο. Ειδικότερα, τα συστήματα εντοπισμού επιθέσεων δεύτερης γενιάς είναι πιο πολύπλοκα, έτσι ώστε να εντοπίζουν διάφορα γεγονότα και συμπεριφορές "ενδιαφέροντος", με βάση συγκεκριμένα πρότυπα και να κάνουν χρήση στατιστικών μεθόδων για τη διάκριση μεταξύ φυσιολογικής και

ανώμαλης συμπεριφοράς. Παρ'όλα αυτά, εξακολουθεί να υπάρχει το πρόβλημα του μεγέθους κάλυψης, για παράδειγμα το ποσοστό των επιθέσεων που ένας συγκεκριμένος detector μπορεί να αναγνωρίσει.

### 3.1.2 Misuse Detection

Η προσέγγιση αυτή αναφέρεται στον εντοπισμό εισβολέων, οι οποίοι επιχειρούν να κερδίσουν παράνομη πρόσβαση σε κάποιο σύστημα, χρησιμοποιώντας κάποια ήδη γνωστή τεχνική και εκμεταλλευόμενοι συγκεκριμένες αδυναμίες του συστήματος. Τα misuse detection systems ερευνούν συνεχώς για συγκεκριμένα "intrusion scenarios", δηλαδή για ακολουθίες γνωστών ενεργειών, οι οποίες όταν εκτελεστούν οδηγούν σε κάποιο είδος επίθεσης, εκτός και αν κάποια εξωτερική παρέμβαση εμποδίσει την ολοκλήρωσή τους. Οι παρατηρήσεις αυτές που χαρακτηρίζουν τη συμπεριφορά του συστήματος ανά πάσα στιγμή μπορεί να στηρίζονται είτε σε real-time δεδομένα είτε σε audit records που έχουν ήδη καταγραφεί από το λειτουργικό σύστημα.

Η διαφορά μεταξύ των διαφόρων misuse detection systems έγκειται στον τρόπο με τον οποίο το κάθε σύστημα περιγράφει μία ύποπτη συμπεριφορά η οποία συνιστά ένα είδος επίθεσης. Τα συστήματα πρώτης γενιάς χρησιμοποιούσαν συγκεκριμένους κανόνες για να περιγράψουν ύποπτες συμπεριφορές. Όμως ο μεγάλος αριθμός των κανόνων αυτών που συσσωρεύονταν, καθιστούσε δύσκολη τόσο την επεξεργασία όσο και τη διαμόρφωσή τους. Τώρα πλέον, τα διάφορα intrusion scenarios μπορούν να προσδιοριστούν προσεγγιστικά, με εξέταση των διαφόρων ακολουθιακών ενεργειών που τα αποτελούν. Κατά τη διάρκεια των ενεργειών αυτών, ένα intrusion detection system μπορεί να προβλέψει το επόμενο βήμα της πιθανής επίθεσης.

## 3.2 ΤΕΧΝΙΚΕΣ ΑΝΙΧΝΕΥΣΗΣ ΔΙΚΤΥΑΚΩΝ ΑΝΩΜΑΛΙΩΝ

### 3.2.1 Συστήματα εντοπισμού εισβολέων – Intrusion Detection Systems (IDS)

Με τον όρο "εισβολή" ή "intrusion", αναφερόμαστε σε μία "παράνομη" ενέργεια που πραγματοποιείται από έναν ή περισσότερους χρήστες ενός πληροφοριακού συστήματος. Ο εισβολέας μπορεί να προέρχεται από το εξωτερικό του συστήματος, ή να είναι ένας εσωτερικός χρήστης ο οποίος υπερβαίνει τα όρια της δικαιοδοσίας που έχει στο σύστημα. Σε κάθε περίπτωση, μία τέτοια ενέργεια μπορεί να αποβεί επιβλαβής για τη σωστή λειτουργία του συστήματος και την παροχή των υπηρεσιών του. Καθώς η τεχνολογία των πληροφοριακών συστημάτων γίνεται όλο και πιο κατανοητή, τα Συστήματα Εντοπισμού Εισβολέων ή Intrusion Detection Systems, έχουν ενσωματωθεί πλέον στη στα ευρύτερα λειτουργικά συστήματα των μεγάλων δικτύων.

Ένα NIDS σύστημα είναι ουσιαστικά ένα σύστημα παρακολούθησης του δικτύου (network sniffer). Δίνει τη δυνατότητα στο διαχειριστή να παρατηρεί τι συμβαίνει στο δίκτυο και του προσφέρει μία γενική εικόνα της κίνησης προς / από ένα σύστημα. Στη συνέχεια, με τη βοήθεια ενός firewall ή με αποστολή κατάλληλων μηνυμάτων, ο διαχειριστής μπορεί να ελέγξει τα διάφορα είδη κίνησης και να μπλοκάρει τους κακόβουλους χρήστες.

Ένα IDS σύστημα αποτελείται συνήθως από 2 στοιχεία :

- έναν αισθητήρα (sensor), παθητικό στοιχείο με πολλά interfaces, που συλλέγει τα πακέτα που διέρχονται από το δίκτυο
- ένα σταθμό ανάλυσης, που διαθέτει επεξεργαστή και μνήμη και χρησιμεύει στην ανάλυση των δεδομένων που συλλέγει ο sensor

Οι πιο κοινοί τύποι IDS είναι οι εξής :

- **Host Based IDS** (Norton, McAfee)
- **Network Intrusion Detection System** (Snort, Shadow) : έχει τη δυνατότητα να εντοπίζει κάθε είδους ανωμαλία στο δίκτυο, σε αντίθεση με

το Host Based IDS που μπορεί να μην εντοπίσει ανωμαλίες που προέρχονται από το εσωτερικό του δικτύου.

Προσφέρει μία καλύτερη γενική εικόνα της κίνησης του δικτύου που οδηγεί σε πιο εύκολο εντοπισμό προβλημάτων.

Γενικά, ένα σύστημα IDS συλλέγει δεδομένα από το δίκτυο και παρουσιάζει μία γενική εικόνα της κίνησης, χωρίς όμως να μπορεί να μπλοκάρει την κακόβουλη κίνηση που εισέρχεται στο δίκτυο. Στη συνέχεια, ο ίδιος ο χρήστης, έχοντας στη διάθεσή του πλέον τα παραπάνω δεδομένα, μπορεί με χρήση φίλτρων, firewalls και άλλων πολιτικών ασφαλείας, να μπλοκάρει άλλους χρήστες και να εμποδίσει πιθανές επιθέσεις. Επιπλέον, παρατηρώντας τα δεδομένα που συλλέγονται, είναι δυνατό να καθοριστεί πού εντοπίζονται οι περισσότερες επιθέσεις, για ποιους λόγους και από ποιους χρήστες προέρχονται και έτσι να εξαχθούν διάφορα γενικά συμπεράσματα που θα βοηθήσουν τους διαχειριστές των δικτύων να λάβουν κατάλληλα μέτρα προφύλαξης.

### 3.2.2 Μέθοδοι ανίχνευσης δικτυακών ανωμαλιών με χρήση του 'unallocated address space'

Καθώς τα φαινόμενα δικτυακών απειλών γίνονται όλο και πιο έντονα, ο εντοπισμός, η διαχείριση και η αντιμετώπισή τους είναι πολύ σημαντική για τους χρήστες δικτύων, μεγάλων οργανισμών, αλλά και ολόκληρου του Internet. Για την καλύτερη ανίχνευση και παρακολούθησή τους, πολλοί ερευνητές και διαχειριστές δικτύων, εφαρμόζουν τη μέθοδο του "unused address space".

Η μέθοδος διαχείρισης των μη χρησιμοποιούμενων διευθύνσεων που έχουν αποδοθεί σε ένα δίκτυο, έχει αποδειχθεί αποτελεσματική για την ανίχνευση και τον εντοπισμό δικτυακών ανωμαλιών και δυσλειτουργιών, όπως τα worms και οι DDOS attacks. Η βασική της ιδέα στηρίζεται στην υπόθεση ότι η κίνηση που δρομολογείται προς μη χρησιμοποιούμενες διευθύνσεις, δηλαδή προς μη υπαρκτούς χρήστες, θεωρείται ύποπτη και μη παραγωγική. Μπορεί να είναι το αποτέλεσμα κάποιας δυσλειτουργίας του συστήματος, ή να οφείλεται σε 'backscatter from spoofed source addresses', ή ακόμα και σε 'network scanning from worms or probing'. Η κίνηση

αυτή που αφορά το συγκεκριμένο κομμάτι διευθύνσεων ενός δικτύου, αναφέρεται συχνά και ως “background radiation”.

Η τεχνική διαχείρισης των πακέτων που δρομολογούνται προς μη χρησιμοποιούμενες διευθύνσεις, έχει επομένως δύο σημαντικά πλεονεκτήματα : πρώτον, όλα τα πακέτα με προορισμό αυτές τις διευθύνσεις μπορούν να θεωρηθούν με μεγάλη βεβαιότητα κακόβουλα, εκτός από κάποιες περιπτώσεις δυσλειτουργιών. Δεύτερον, σε αντίθεση με άλλα συστήματα ανίχνευσης που διαχειρίζονται παθητικά το κάθε είδος κίνησης, ένα εργαλείο ανίχνευσης (detection tool) που διερευνά την κίνηση προς τις unused addresses, μπορεί να ανταποκριθεί στις αιτήσεις συνδέσεων που γίνονται προς αυτές και με τον τρόπο αυτό να συλλέξει τα πακέτα που περιέχουν κάποια πληροφορία χαρακτηριστική επιθέσεων (attack-specific information) και έτσι να παρεμποδίσει την είσοδό τους στο δίκτυο.

Η διαχείριση της παραπάνω δικτυακής κίνησης για εντοπισμό διαφόρων κακόβουλων στοιχείων, μπορεί να πάρει πολλές διαφορετικές μορφές. Η πιο συνηθισμένη τεχνική ανίχνευσης βασίζεται στην παθητική μέτρηση και ανάλυση των στοιχείων κίνησης ενός δικτύου (flows) , με χρήση ειδικών εργαλείων συλλογής και ανάλυσης (flow-tools), όπως το Netflow.

Μέχρι στιγμής έχουν αναπτυχθεί αρκετές τεχνικές για την ανίχνευση επιθέσεων μέσω της μεθόδου του “unused address space”, έτσι όπως αυτή περιγράφηκε παραπάνω. Κάποιες από αυτές είναι τα network telescopes, blackholes and darknets. Ωστόσο, υπάρχουν εμφανείς διαφορές τόσο στον όγκο όσο και στη σύνθεση των συλλεγόμενων δεδομένων κίνησης, που εξαρτώνται από το είδος της τεχνικής που χρησιμοποιείται αλλά και από το μέγεθος του “unused address space”. Συνεπώς, τα συμπεράσματα τα οποία προκύπτουν κάθε φορά από την επεξεργασία των συλλεγόμενων στοιχείων κίνησης, δεν είναι εύκολο να γενικευτούν χωρίς να ληφθεί υπόψη το μέγεθος του εκάστοτε δείγματος. Ένας τρόπος για την εξαγωγή περισσότερο αντιπροσωπευτικών συμπερασμάτων, είναι η αύξηση του αριθμού και του μεγέθους των “unused address blocks”. Γι’αυτό και η δυνατότητα διαχείρισης των “unused address spaces” είναι μεγαλύτερη σε δίκτυα κλάσης A ή B, όπου ο αριθμός των διευθύνσεων αυτών είναι σαφώς πιο σημαντικός. Ωστόσο, αυτό που έχει επίσης πολύ μεγάλη σημασία για την εγκυρότητα των αποτελεσμάτων της διαχειριζόμενης κίνησης, είναι η κατάλληλη τοποθέτηση των “monitoring blocks” στα διάφορα σημεία του δικτύου.

### 3.2.3 Επέκταση σε κατανεμημένα συστήματα δικτύων (Distributed Network Systems)

Οι τεχνικές για την ανίχνευση επιθέσεων σε ένα κατανεμημένο σύστημα ή δίκτυο υπολογιστών είναι παρόμοιες με αυτές που περιγράφηκαν για ένα μεμονωμένο λειτουργικό σύστημα. Ωστόσο, οι πολλαπλοί χρήστες ενός δικτυακού συστήματος μπορούν να συνεργαστούν προκειμένου να εξαπολύσουν μία πολλαπλή επίθεση, στην οποία συμμετέχουν πολλές ξεχωριστές οντότητες. Η εμπειρία μάλιστα έχει δείξει ότι οι πολλαπλές επιθέσεις σε ένα δίκτυο ή σε ένα κατανεμημένο σύστημα είναι πολύ πιο συχνές από τις μεμονωμένες επιθέσεις. Κι αυτό, γιατί παρέχουν περισσότερες δυνατότητες εκμετάλλευσης των πόρων ενός δικτύου από τους εισβολείς, οι οποίοι έχουν τη δυνατότητα να καλύψουν πιο εύκολα τις ενέργειές τους μεταξύ των πολλαπλών κόμβων και να εκμεταλλευτούν το γεγονός ότι το ένα λειτουργικό σύστημα – στόχος μπορεί να μην έχει ενημερωθεί για την παρόμοια κατάσταση κάποιου άλλου.

Είναι λοιπόν φανερό, πως για να εντοπιστούν οι πολλαπλές επιθέσεις, θα πρέπει οι detectors να είναι ικανοί να συσχετίζουν τις ενέργειες μεταξύ των διαφόρων χρηστών ενός συστήματος. Για το σκοπό αυτό εφαρμόζεται είτε κάποια συγκεντρωτική μέθοδος, όπου όλες οι πληροφορίες συγκεντρώνονται σε ένα συγκεκριμένο σημείο και στη συνέχεια αναλύονται, είτε μία αποκεντρωτική, ιεραρχική προσέγγιση, όπου οι διάφορες πληροφορίες επεξεργάζονται τοπικά και στη συνέχεια συγκεντρώνονται μόνο τα κρίσιμα κομμάτια τους για περαιτέρω ανάλυση από τα στοιχεία των intrusion detection systems.

Δυσκολίες που ανακύπτουν από την ανίχνευση δικτυακών ανωμαλιών και δυσλειτουργιών, αφορούν το κλασικό πρόβλημα του συγχρονισμού μεταξύ των καταγραφών που γίνονται σε διαφορετικούς κόμβους, καθώς η ακολουθία των διαφόρων γεγονότων που λαμβάνουν χώρα σε όλο το δίκτυο, έχει μεγάλη σημασία από τη στιγμή που αυτή καθορίζει ένα πιθανό σενάριο επίθεσης.

### 3.2.4 Αυτοπροστασία των Intrusion Detection Systems

Το λογισμικό των μηχανισμών ανίχνευσης επιθέσεων που περιγράφηκαν παραπάνω, δεν είναι ανεξάρτητο από το υπόλοιπο λειτουργικό σύστημα. Θα πρέπει να ληφθούν και γι'αυτό τα κατάλληλα μέτρα προστασίας του από πιθανές εισβολές

που σκοπεύουν να παρεμποδίσουν τη σωστή λειτουργία του, δηλαδή την ανίχνευση για άλλες εισβολές στο υπόλοιπο σύστημα. Για το λόγο αυτό φροντίζεται να διατηρηθεί η φυσική προστασία του συστήματος, αλλά και των δεδομένων τα οποία επεξεργάζονται από αυτό, μέσω διαφόρων μηχανισμών, όπως read-only access στη βάση δεδομένων του κλπ.

Από τη στιγμή που η λειτουργία των συστημάτων αυτών στηρίζεται σε audit records data, αυτά θα πρέπει να είναι διαθέσιμα τη σωστή χρονική στιγμή. Μεγάλα χρονικά κενά μεταξύ των λαμβανόμενων δεδομένων μπορούν να καταστήσουν άχρηστη τη λειτουργία ενός intrusion detection system, από τη στιγμή που δε θα μπορούν να εντοπιστούν οι εκάστοτε εισβολές τη σωστή χρονική στιγμή.

Για τους παραπάνω λόγους, οι κατασκευαστές των εν λόγω συστημάτων θα πρέπει να λαμβάνουν υπόψη τους την ανάγκη συνύπαρξής τους με το υπόλοιπο σύστημα. Τόσο το προφυλλασσόμενο σύστημα, όσο και το ίδιο το σύστημα ανίχνευσης, δε θα πρέπει να ανταγωνίζονται για τον ίδιο επεξεργαστή και τους ίδιους πόρους, γιατί αυτό θα καθιστούσε το δεύτερο ευάλωτο σε denial-of-service attacks. Η χρήση ξεχωριστής μνήμης και επεξεργαστή από τα συστήματα ανίχνευσης λύνει τα περισσότερα προβλήματα αυτού του είδους.

Τα Συστήματα Ανίχνευσης Εισβολέων της επόμενης γενιάς, αναμένεται να εφαρμόζουν περισσότερο ακριβείς τεχνικές, καθώς οι παράμετροι των λειτουργικών συστημάτων όπως και των πρωτοκόλλων που μεσολαβούν στη σύνδεση πολλαπλών υπολογιστών σε ένα αυτόνομο δίκτυο, θα είναι καθορισμένες με μεγαλύτερη ακρίβεια. Αυτό θα δίνει τη δυνατότητα αποτελεσματικότερης διαχείρισης και εντοπισμού κάποιας ανώμαλης συμπεριφοράς, η οποία θα προσδιορίζεται πλέον με βάση πιο σαφή κριτήρια.

Η διαδικασία εντοπισμού των παράνομων ενεργειών εκ μέρους των εισβολέων θα γίνεται πιο εύκολα, από τη στιγμή που τα νέα δικτυακά πρωτόκολλα θα επιτρέπουν εξακρίβωση της προέλευσης και αναγνώριση των εξωτερικών εισβολέων, ακολουθώντας τη διαδρομή των επιθετικών πακέτων από τον κόμβο-προορισμού πίσω στον κόμβο-πηγή, μέσω των ενδιάμεσων κόμβων.

# ΚΕΦΑΛΑΙΟ 4 : ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΩΝ NETWORK TELESCOPES

## 4.1 ΤΙ ΕΙΝΑΙ ΕΝΑ NETWORK TELESCOPE

Τα τελευταία τέσσερα χρόνια, μελετάται μία νέα προσέγγιση αναφορικά με την ασφάλεια των δικτύων και τον εντοπισμό επιθέσεων, η οποία βασίζεται στη μέθοδο του "unused address space", δηλαδή στην παρατήρηση της θεωρητικά "κακόβουλης" κίνησης που αφορά το σύνολο των μη χρησιμοποιούμενων διευθύνσεων του δικτύου. Πρόκειται για τα δικτυακά τηλεσκόπια ή "network telescopes" όπως συνήθως αποκαλούνται.

Το network telescope είναι ένα εργαλείο που παρακολουθεί δικτυακή κίνηση από ή προς ένα κομμάτι δρομολογημένων IP διευθύνσεων ενός δικτύου, οι οποίες δεν έχουν αποδοθεί προς χρήση και αποτελούν το λεγόμενο "dark address space". Η διαχείριση αυτής της μη αναμενόμενης, κακόβουλης κίνησης που φτάνει σε ένα "network telescope", παρέχει τη δυνατότητα παρακολούθησης και αξιολόγησης μεμονωμένων γεγονότων, μεγάλης συνήθως έκτασης, που αφορούν την ασφάλεια ενός δικτύου, όπως περιπτώσεις denial-of-service attacks, μόλυνση συστημάτων από worms κλπ.

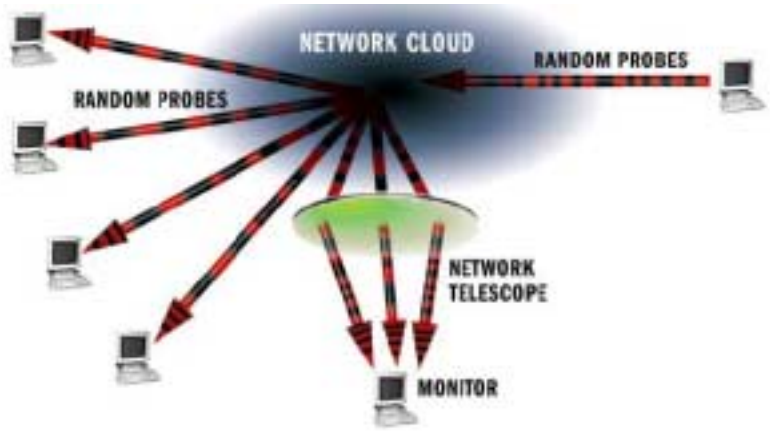
Τα κλασικά network telescopes λειτουργούν με έναν παθητικό τρόπο, εξετάζοντας "παράνομες" μορφές κίνησης, που έχουν ως προορισμό μη χρησιμοποιούμενες IP διευθύνσεις, δηλαδή διευθύνσεις που κανονικά δεν έχουν αποδοθεί σε χρήστες (unallocated addresses). Σκοπός τους είναι να καταγράφουν τα πακέτα που συνιστούν την παραπάνω κίνηση και να συγκεντρώνουν κατάλληλα δεδομένα. Η βασική ιδέα ενός telescope στηρίζεται στο γεγονός ότι, εάν ένας χρήστης στείλει πολλά πακέτα προς τυχαίες IP διευθύνσεις προσπαθώντας να εντοπίσει έναν στόχο, τότε κάποια από τα πακέτα αυτά θα καταγραφούν από το network telescope, ανάλογα με το address space που διαχειρίζεται. Μπορεί να παρατηρηθεί επιπλέον η συμπεριφορά συγκεκριμένων χρηστών και να εξαχθούν συμπεράσματα που αφορούν χρόνους έναρξης και λήξης αποστολής πακέτων, μέγεθος κίνησης, και πηγή προέλευσής της.

Όσο μεγαλύτερο είναι το εύρος των διευθύνσεων που αποδίδεται σε ένα network telescope, τόσο περισσότερο αυξάνεται και η ανάλυση των παρατηρούμενων γεγονότων, παρέχοντας έτσι περισσότερες και πιο ακριβείς πληροφορίες. Επιπλέον,

μπορούν να εντοπιστούν και γεγονότα μικρότερης κλίμακας, που μπορεί να σχετίζονται είτε με αποστολή μικρότερου αριθμού πακέτων είτε με μικρή διάρκεια του γεγονότος.

Για ένα network telescope, η καταγραφή των στοιχείων κίνησης που αφορούν το unallocated IP address space, μπορεί να γίνει ως εξής :

- με χρήση του πρωτοκόλλου BGP, μπορεί να καθοριστεί μία συγκεκριμένη διαδρομή προς το διαχειριζόμενο κομμάτι των IP διευθύνσεων
- με χρήση flowtools σε συνδυασμό με κατάλληλα φίλτρα, καθίσταται δυνατή η συλλογή δεδομένων από routers
- καθορίζοντας εξαρχής συγκεκριμένα υποδίκτυα (σύνολα IP διευθύνσεων) τα οποία δεν έχουν αποδοθεί προς χρήση.




***Εικόνα 8 :*** *Εάν ένα μηχάνημα στέλνει με τυχαίο τρόπο πλήθος πακέτων σε IP διευθύνσεις, τότε κάποια από τα πακέτα αυτά καταγράφονται από το telescope, ανάλογα με το μέγεθος του διαχειριζόμενου address space.*

Πέρα από τα κλασικά network telescopes, μπορούμε να διακρίνουμε τις εξής κατηγορίες :

- Distributed Telescopes

  Πρόκειται για network telescopes που παρακολουθούν ένα ευρύτερο σύνολο διευθύνσεων, αποτελούμενο από περισσότερα του ενός address blocks, τα οποία δεν είναι απαραίτητα συνεχόμενα μεταξύ τους. Κάθε περιοχή διευθύνσεων παρακολουθείται από ένα μικρού μεγέθους telescope, και όλα μαζί συνδυάζονται

για να σχηματίσουν το distributed network telescope. Με αυτόν τον τρόπο η αποτελεσματικότητά τους αυξάνεται, καθώς ο εντοπισμός της κακόβουλης κίνησης δεν εξαρτάται πλέον από ένα μοναδικό address block, ενώ ταυτόχρονα επιτυγχάνεται καλύτερος χρόνος και ακρίβεια ανίχνευσης. Επίσης είναι πιο δύσκολο πλέον να αποφευχθούν από τους κακόβουλους χρήστες μέσω εξειδικευμένων επιλεκτικών αλγοριθμικών τεχνικών. Τέλος, οι διαθέσιμοι πόροι ολόκληρου του συστήματος ανίχνευσης, μπορούν να διαμοιραστούν μεταξύ των πολλαπλών telescopes και με αυτόν τον τρόπο να γίνει καλύτερη αξιοποίηση της λειτουργικής τους ικανότητας και πιο σωστή διαχείριση του φορτίου κίνησης.

Ωστόσο, η χρήση ενός distributed network telescope έχει ένα σημαντικό μειονέκτημα. Η συσχέτιση των γεγονότων που συλλέγονται σε κάθε block εξαρτάται από το συγχρονισμό των επιμέρους telescopes. Συνεπώς, η ακρίβεια των αναλύσεων που αφορούν χρονικά εξαρτώμενες παραμέτρους, εξαρτάται άμεσα από την ακρίβεια στο συγχρονισμό, ειδικά όταν πρόκειται για real-time δεδομένα. Όπως φαίνεται λοιπόν, μειονέκτημα αποτελεί η δυσκολία στην επεξεργασία των δεδομένων και στην εξαγωγή συμπερασμάτων που οφείλεται σε διαφορετικούς χρόνους έναρξης των γεγονότων σε κάθε block.

- Anycast Telescopes

  Πρόκειται για ένα άλλο είδος network telescopes, όπου η δρομολόγηση προς το διαχειριζόμενο address space καθίσταται πιο εύκολη, καθώς υπάρχουν πολλαπλές δυνατές διαδρομές που οδηγούν σε αυτό και καθορίζονται από διαφορετικά σημεία του δικτύου. Εμφανίζουν παρόμοια πλεονεκτήματα και μειονεκτήματα με αυτά των Distributed Telescopes, εκτός του ότι δεν μπορούν να παρακολουθήσουν μεγαλύτερα address blocks. Ένα βασικό τους πλεονέκτημα έναντι στα distributed telescopes, είναι ότι προσφέρουν περισσότερες δυνατές διαδρομές για την κίνηση που κατευθύνεται προς το unallocated address space, με αποτέλεσμα να εντοπίζουν τα διάφορα γεγονότα σε μικρότερο χρόνο, ανάλογα πάντα με τη σχετική θέση των telescopes και των χρηστών του δικτύου. Με τον τρόπο αυτό, ένα anycast telescope επιτυγχάνει τη διανομή του φορτίου κίνησης μεταξύ των διαφόρων σημείων του δικτύου, χωρίς να υπερφορτώνει ιδιαίτερα κάποιο κανάλι, καθιστώντας ταυτόχρονα πιο γρήγορη τη διακίνηση των πακέτων και την ανίχνευση της "ανεπιθύμητης" κίνησης.

- Transit Network Telescopes

  Πρόκειται για μία άλλη κατηγορία network telescopes που διαχειρίζονται ένα σύνολο από address blocks, τα οποία μπορεί να ανήκουν ακόμα και σε διαφορετικά αυτόνομα συστήματα (ASs). Το χαρακτηριστικό τους είναι, ότι τοποθετούνται σε κεντρικά σημεία του δικτύου, έτσι ώστε να μπορούν να παρατηρούν την κίνηση που προέρχεται ή που κατευθύνεται προς τα επιμέρους address blocks και διαχειρίζονται τα παρατηρούμενα στοιχεία κίνησης στα σημεία αυτά, χωρίς να κάνουν χρήση μικρότερων telescopes τοποθετούμενων σε κάθε network edge ξεχωριστά.

  Το βασικό πλεονέκτημα ενός transit telescope, είναι ότι μπορεί να παρατηρεί την κίνηση σε ένα μεγάλο εύρος διευθύνσεων από ένα κεντρικό σημείο, αποφεύγοντας έτσι προβλήματα συγχρονισμού και ανταλλαγής δεδομένων που εμφανίζει για παράδειγμα ένα distributed telescope. Ωστόσο, υπάρχουν και κάποια βασικά μειονεκτήματα. Ένα transit telescope, μπορεί να παρατηρήσει την κίνηση κάθε μορφής, η οποία αφορά διευθύνσεις που ανήκουν μέσα στο δικό του address space ή σε single-homed customer networks. Για κάθε άλλο εύρος διευθύνσεων όμως, μπορεί να παρατηρήσει μόνο κάποιο περιστατικό κίνησης (event traffic) που δρομολογείται μέσα από το δίκτυο στο οποίο ανήκει. Συνεπώς, χωρίς να διαθέτει μία ολοκληρωμένη εικόνα των περιστατικών κίνησης, ο ακριβής χαρακτηρισμός τους και η εξαγωγή συμπερασμάτων καθίσταται αρκετά δύσκολη.

  Με βάση τα παραπάνω, θα μπορούσαμε να πούμε πως ένα transit telescope είναι περισσότερο χρήσιμο για την παρατήρηση και καταγραφή γεγονότων κίνησης που αφορούν ένα αρκετά μεγάλο εύρος διευθύνσεων, και λιγότερο για τον ακριβή χαρακτηρισμό τους.

- Organizational – Internal Telescopes

  Πρόκειται για μικρά network telescopes τα οποία παρατηρούν εσωτερικά γεγονότα που λαμβάνουν χώρα σε ένα δίκτυο. Παρ'ότι δε μπορούν να μελετήσουν την εξωτερική κίνηση που μπαίνει στο δίκτυο, αποδεικνύονται χρήσιμα στο γρήγορο εντοπισμό και την αντιμετώπιση διαφόρων εσωτερικών προβλημάτων του. Για παράδειγμα, μπορούν να εντοπίσουν γρήγορα κάποια μηχανήματα που έχουν προσβληθεί από worms, ή από hackers ή εμφανίζουν κάποια άλλη δυσλειτουργία.

- Honeyfarm Telescopes

Η διαφορά των honeyfarm telescopes από τα προηγούμενα είδη που περιγράφηκαν, είναι ότι δεν περιορίζονται απλά στην παθητική παρατήρηση των περιστατικών κίνησης, αλλά ανταποκρίνονται ενεργά σε κάποια ή και σε όλα από τα γεγονότα αυτά, με τη βοήθεια των honeypots. Το address space το οποίο ελέγχουν μπορεί να είναι πολύ μεγάλο (για παράδειγμα 16 εκατομμύρια διευθύνσεις αντιστοιχούν σε ένα /8 prefix) και γι'αυτό χρειάζεται να αποφασίσουν ποιες από τις διευθύνσεις αυτές θα ανταποκριθούν ενεργά στην εισερχόμενη κίνηση, καθώς και σε ποια είδη κίνησης είναι σκόπιμο να επικεντρωθούν. Βέβαια, η παραπάνω ενεργή απόκριση προσθέτει επιπλέον φορτίο στο δίκτυο, το οποίο μπορεί να αναμιχθεί με την κίνηση των υπόλοιπων γεγονότων, να επιδεινώσει την κατάσταση του δικτύου σε περιπτώσεις υπερφόρτωσης, ή ακόμη και να μη γίνει καθόλου αποδεκτή εάν το κόστος λόγω φορτίου την αντίστοιχη χρονική στιγμή δεν το επιτρέπει.

## 4.2 ΣΗΜΑΣΙΑ ΤΗΣ ΧΡΗΣΗΣ ΕΝΟΣ TELESCOPE ΣΤΗΝ ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΤΗΣ ΔΙΚΤΥΑΚΗΣ ΚΙΝΗΣΗΣ

Η αρχική χρήση του telescope, επικεντρωνόταν στον εντοπισμό spoofed DDoS επιθέσεων. Αργότερα, ο ίδιος μηχανισμός επεκτάθηκε και στον εντοπισμό worms. Στις μέρες μας πλέον, το telescope αποτελεί μία κοινή τεχνική εντοπισμού worms, DDoS και τύπου 'scanning' επιθέσεων που εφαρμόζεται κυρίως από έναν πάροχο (ISP) ή μια εταιρία.

Ο εντοπισμός και στη συνέχεια η κατανόηση των επιπτώσεων από DDoS και worms επιθέσεις, αποτελούν μία πρόκληση για το σημερινό τεχνολογικό κόσμο. Η ικανότητα να εντοπιστούν επιθέσεις DDoS ή worms στο αρχικό στάδιο της εξάπλωσής τους, είναι ένα κρίσιμο θέμα στην προσπάθεια εύρεσης και χρήσης μιας τεχνικής που θα απαλλάσσει ένα μολυσμένο σύστημα από τις δυσάρεστες επιπτώσεις επιθέσεων τέτοιου είδους.

Τα δεδομένα της δικτυακής κίνησης που συγκεντρώνονται από ένα network telescope (π.χ με τη μορφή Netflow), μας δίνουν τη δυνατότητα παρακολούθησης και καταγραφής των τάσεων της επιθετικής κίνησης (όπως νέοι τύποι επιθέσεων και οι στόχοι τους) στην πάροδο του χρόνου. Με βάση αυτά τα πραγματικού χρόνου

δεδομένα, μπορούν να μελετηθούν και να αντιμετωπιστούν πιο αποδοτικά κάποιες μελλοντικές επιθέσεις.

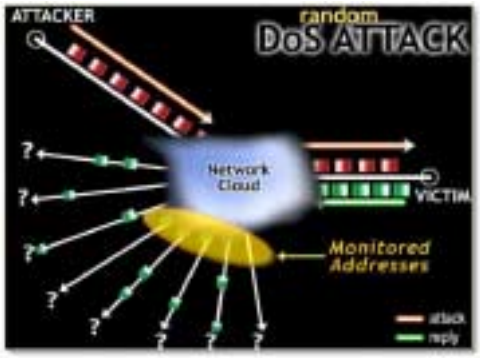
Τα πακέτα που μπορεί να εντοπίσει και να καταγράψει ένα telescope μπορούν να ταξινομηθούν χοντρικά σε 4 κατηγορίες :

1. **Host / Port scanning:** πρόκειται για μία τεχνική που εφαρμόζεται συνήθως από hackers προκειμένου να εντοπιστούν ευπρόσβλητα μηχανήματα ή ανοιχτά ports ενός δικτύου. Σε μία τέτοια περίπτωση, το telescope θα μπορέσει να καταγράψει τα πακέτα – ανιχνευτές που προσπαθούν να εντοπίσουν τους πιθανούς τους στόχους. Στην κατηγορία αυτή ανήκουν και τα worms, τα οποία στην προσπάθειά τους να εξαπλωθούν και να μολύνουν τα μηχανήματα ενός δικτύου, ψάχνουν για τα πιθανά θύματά τους, αποστέλλοντας κατάλληλα πακέτα, κάποια από τα οποία φτάνουν σε μη υπάρχοντες κόμβους ή ports. Μία μέθοδος εντοπισμού τέτοιου είδους κίνησης που καταγράφεται από το telescope, βασίζεται στην παρατήρηση της κίνησης που αντιστοιχεί σε μια μη ολοκληρωμένη σύνδεση, στην περίπτωση δηλαδή όπου εστάλη ένα SYN πακέτο χωρίς να υπάρξει αντίστοιχα ένα SYN-ACK ή ένα RST πακέτο.


***Εικόνα 9 :*** *Ένα worm αποστέλλει SYN πακέτα για εγκατάσταση σύνδεσης προς έναν μη υπαρκτό χρήστη*

2. **Backscatter from spoofed DDOS attacks :** στην περίπτωση αυτή, ένα μεγάλο πλήθος πακέτων αποστέλλεται προς το θύμα, προκειμένου να εξαντλήσει τους πόρους του server ή του ίδιου του δικτύου. Μία από τις πιο διαδεδομένες 'spoofed DDoS' επιθέσεις είναι η SYN επίθεση, όπου ο χρήστης αποστέλλει ένα πλήθος από SYN πακέτα, τα οποία υπερφορτώνουν το θύμα και κατά συνέπεια η δημιουργία μίας πραγματικής σύνδεσης με τον server, καθίσταται αδύνατη. Αν θεωρήσουμε την περίπτωση όπου το θύμα αποκρίνεται με ένα SYN-ACK πακέτο, στην ψεύτικη διεύθυνση πηγής που θα λάβει με το πρώτο SYN πακέτο, η κίνηση αυτή είναι πιθανό να δρομολογηθεί

προς μία μη χρησιμοποιούμενη διεύθυνση, εφόσον αυτή έχει επιλεχθεί τυχαία. Η κίνηση αυτή θα καταγραφεί από το telescope το οποίο διαχειρίζεται το σύνολο των μη χρησιμοποιούμενων διευθύνσεων και θα είναι προφανώς ανάλογη με το μέγεθος του συνόλου αυτού. Μελέτη και παρακολούθηση της συνολικής SYN-ACK απόκρισης που λαμβάνεται από το telescope, μπορεί να οδηγήσει σε εξίσου χρήσιμα συμπεράσματα για το είδος και το μέγεθος της επίθεσης.




***Εικόνα 10 :*** *Το θύμα πλημμυρίζεται με πακέτα που μεταφέρουν τυχαίες, "spoofed" source IP διευθύνσεις. Θεωρώντας γνήσια τα πακέτα αυτά, αποκρίνεται στις αιτήσεις που αφορούν τις πλαστές διευθύνσεις.*

3. **Configuration Mistakes:** ένα flow το οποίο έχει πολύ μικρή διάρκεια ζωής και δεν μπορεί να καταταγεί σε μία από τις παραπάνω κατηγορίες, αποδίδεται συνήθως σε κάποια δυσλειτουργία του συστήματος και χαρακτηρίζεται ως "configuration mistake".

4. **Άλλου τύπου :** ένα "ανεπιθύμητο" flow το οποίο δεν ανήκει σε καμία από τις παραπάνω κατηγορίες.

Η συνήθης κατανομή των πακέτων στις παραπάνω κατηγορίες έχει ως εξής [14] :

| **Type of packet** | **percentage** |
|---|---|
| Host/port scanning | 92% |
| DDOS backscatter | 5% |
| Configuration mistakes | 2% |
| Other | 1% |

## 4.3 ΔΥΣΚΟΛΙΕΣ ΣΤΗ ΧΡΗΣΗ, ΛΕΙΤΟΥΡΓΙΑ ΚΑΙ ΕΓΚΑΤΑΣΤΑΣΗ ΕΝΟΣ TELESCOPE

Πέρα από τα πλεονεκτήματα που απορρέουν από τη χρήση ενός telescope, η εγκατάσταση και η λειτουργία του δεν παύει να παρουσιάζει κάποιες δυσκολίες. Μία από αυτές αφορά τον καθορισμό του παρατηρούμενου συνόλου διευθύνσεων, δηλαδή το unallocated address space του δικτύου. Η λειτουργία του telescope απαιτεί από το χρήστη να γνωρίζει ο ίδιος και να μπορεί να καθορίσει το κομμάτι των διευθύνσεων του δικτύου, προς τις οποίες την κίνηση ενδιαφέρεται να μελετήσει. Η διαδικασία αυτή είναι αρκετά επίπονη, ιδιαίτερα στην περίπτωση όπου το μέγεθος του δικτύου είναι μεγάλο και χρειάζεται να μελετηθεί ολόκληρο το unallocated address space. Επιπλέον, το σύνολο των διευθύνσεων αυτών μπορεί να μεταβάλλεται, καθώς νέες διευθύνσεις αποδίδονται σε νέους χρήστες και έτσι ο διαχειριστής του δικτύου ή ένας άλλος χρήστης που είναι υπεύθυνος για τη διαδικασία αυτή, θα πρέπει να γνωρίζει ανά πάσα στιγμή το νέο unallocated address space, αλλά και να το μεταβάλλει ο ίδιος ανάλογα, πριν από τη λειτουργία του telescope.

Μέσα από την παρούσα εργασία υπερβαίνουμε τη δυσκολία αυτή με τη δημιουργία μιας εφαρμογής, η οποία αυτοματοποιεί τη διαδικασία εύρεσης του unallocated address space και μπορεί να εφαρμοστεί σε ένα δίκτυο, γνωρίζοντας απλά το σύνολο των IP διευθύνσεων που διαθέτει (total address space). Το σύνολο των IP διευθύνσεων που διαθέτει ένα δίκτυο δεν μεταβάλλεται συχνά και μπορεί να βρεθεί με την υπηρεσία *'whois'* [27].

Ένα επιπλέον ζήτημα που ανακύπτει, είναι αυτό που αφορά τις δυνατότητες εφαρμογής του telescope σε περισσότερα από ένα δίκτυα και τη συγκέντρωση αντίστοιχων δεδομένων. Μέχρι στιγμής, με τα υπάρχοντα εργαλεία όπως είναι τα flow-tools, η καταγραφή δεδομένων μπορεί να γίνει για ένα συγκεκριμένο και μόνο δίκτυο, με βάση το διαχειριζόμενο σύνολο διευθύνσεων που έχει οριστεί γι'αυτό. Προκειμένου να διευκολύνουμε την παράλληλη συλλογή στοιχείων για σύνολα διευθύνσεων που ανήκουν σε διαφορετικά δίκτυα, επεκτείνουμε κάποια από τις ήδη υπάρχουσες υλοποιήσεις των flow-tools, έτσι ώστε να είναι δυνατή η ταυτόχρονη εφαρμογή του εργαλείου σε περισσότερα από ένα δίκτυα. Για καθένα από τα δίκτυα αυτά μάλιστα, μπορεί να χρησιμοποιηθεί ένα ξεχωριστό φίλτρο, ανάλογα με το κομμάτι των διευθύνσεων που θέλουμε να μελετήσουμε για το κάθε δίκτυο. Το

γεγονός αυτό ουσιαστικά διευκολύνει τη διαδικασία σχεδιασμού και εφαρμογής ενός ‘transit telescope’.

# ΚΕΦΑΛΑΙΟ 5 : ΠΡΟΤΕΙΝΟΜΕΝΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΛΟΠΟΙΗΣΗΣ ΕΝΟΣ NETWORK TELESCOPE

## 5.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΥΛΟΠΟΙΗΣΗΣ

'Ενα δικτυακό τηλεσκόπιο (network telescope) είναι ένα σύστημα που παρατηρεί ένα μέρος της κίνησης ενός δικτύου, το οποίο δρομολογείται κανονικά αλλά δεν έχει αποδοθεί για χρήση (unallocated address space). Παράδειγμα αποτελούν τμήματα του συνολικού address space που έχουν ανατεθεί σε ένα δίκτυο (π.χ. Ε.Μ.Π: 147.102.0.0/16) και τα οποία δεν έχουν αποδοθεί προς χρήση (πχ 147.102.17.0/24).

Στόχος της παρούσας εργασίας είναι η δημιουργία ενός συνόλου εφαρμογών-εργαλείων που απλοποιούν την εγκατάσταση και λειτουργία ενός network telescope. Σημαντικό σημείο αποτελεί η αυτοματοποίηση της ρύθμισης του non-allocated address space. Το λογισμικό που αναπτύχθηκε βασίστηκε σε υπάρχουσες υλοποιήσεις των flow-tools. Τα εργαλεία που υλοποιήθηκαν, χρησιμοποιήθηκαν για την εγκατάσταση ενός στοιχειώδους transit telescope και έγινε μια πρώτη μελέτη των αποτελεσμάτων του. Επιπλέον, έγινε μία απόπειρα μελέτης της κίνησης που αφορά δύο ξεχωριστά δίκτυα ταυτόχρονα, μέσω μίας κοινής εφαρμογής που βασίζεται σε flow-tools, η οποία επεκτείνεται έτσι ώστε να έχει ευρύτερη χρήση σε παραπάνω του ενός δίκτυα. Το γεγονός αυτό διευκολύνει την παράλληλη συγκέντρωση δεδομένων και εξαγωγή χρήσιμων συμπερασμάτων μέσα από την κοινή μελέτη και σύγκριση των αποτελεσμάτων.

Η διαδικασία της υλοποίησης του παραπάνω network telescope, βασίζεται στην υλοποίηση κάποιων επιμέρους δομικών στοιχείων και εφαρμογών, που σε συνδυασμό μεταξύ τους επιτυγχάνουν την τελική λειτουργία συλλογής και παρατήρησης δεδομένων, για την κίνηση του δικτύου στο οποίο εφαρμόζεται το network telescope. Πιο αναλυτικά, το σύνολο της υλοποίησης συνίσταται από τα εξής επιμέρους βήματα:

**1)** Αρχικά συλλέγουμε τις διευθύνσεις του δικτύου που έχουν αποδοθεί κανονικά προς χρήση (allocated addresses). Αυτό γίνεται με τη βοήθεια στοιχείων που συγκεντρώνουμε με τη βοήθεια του πρωτοκόλλου διαχείρισης SNMP για παράδειγμα μέσα από τις πληροφορίες της OSPF MIB. Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί και η MIB-2 routing table.

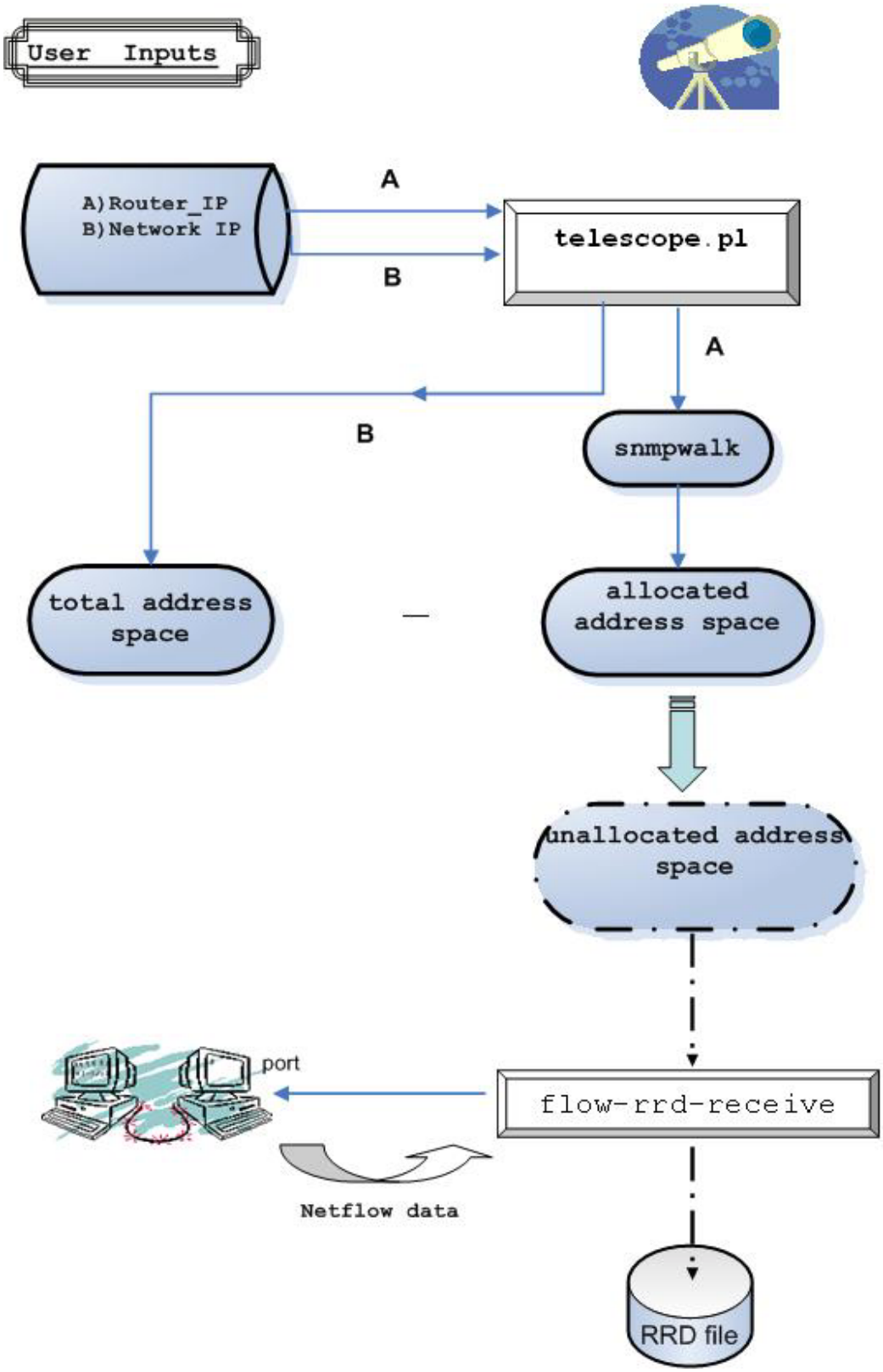
**2)** Στη συνέχεια, έχοντας στη διάθεσή μας το σύνολο των διευθύνσεων που αντιστοιχούν στο προς διαχείριση δίκτυο (total address space), εφαρμόζουμε ένα είδος αφαίρεσης μεταξύ του συνόλου αυτού και του συνόλου των αποδιδόμενων διευθύνσεων (allocated addresses) που συγκεντρώσαμε από το προηγούμενο βήμα. Με τον τρόπο αυτό, προκύπτει το unallocated address space του δικτύου, δηλαδή το σύνολο των μη χρησιμοποιούμενων διευθύνσεων, την κίνηση προς το οποίο μας ενδιαφέρει να εξετάσουμε στη συνέχεια. Έτσι παράλληλα αυτοματοποιείται η διαδικασία εύρεσης του unallocated address space, το οποίο δε χρειάζεται να καθορίζεται εξαρχής από τον διαχειριστή του εκάστοτε δικτύου, καθώς η παραπάνω διαδικασία μπορεί να εφαρμοστεί αυτόματα σε κάθε δίκτυο. Το σύνολο των διευθύνσεων αυτών που δημιουργείται, θα αποτελέσει στη συνέχεια ένα είδος φίλτρου για την καταγραφή της κίνησης που προορίζεται προς αυτό. Για την υλοποίηση όλων των παραπάνω κάνουμε χρήση μίας εφαρμογής σε γλώσσα perl.

**3)** Σειρά έχει η συλλογή των δεδομένων κίνησης που αφορούν το unallocated address space του δικτύου, από τα οποία θα εξαχθούν κατάλληλα συμπεράσματα. Τα δεδομένα αυτά συλλέγονται υπό τη μορφή flow records μέσα από το σύστημα Netflow. Τα flows αποτελούν σειρές από πακέτα που δρομολογούνται προς μία συγκεκριμένη κατεύθυνση και τα flow-records που παρέχονται από το σύστημα Netflow δίνουν πληροφορίες για τον όγκο της παρατηρούμενης κίνησης ανά flow καλλά όχι και για το περιεχόμενό της. Η συλλογή των παραπάνω δεδομένων (flow-records), γίνεται με τη βοήθεια κατάλληλων εργαλείων, flow-tools (όπως το flow-capture ή το flow-receive) τα οποία συνδυάζουν και τη χρήση του φίλτρου που αναπτύχθηκε, προκειμένου να καταγράψουν μόνο τα γεγονότα που προορίζονται προς τις μη αποδιδόμενες διευθύνσεις. Έτσι δημιουργείται ένα σύνολο από flow

records τα οποία χαρακτηρίζουν την κίνηση προς τις διευθύνσεις αυτές του δικτύου, η οποία θεωρείται ύποπτη. Επιπλέον, διευρύνουμε την παραπάνω διαδικασία έτσι ώστε να έχει ταυτόχρονη εφαρμογή σε περισσότερα από ένα δίκτυα.

**4)** Τέλος, ύστερα από την εφαρμογή όλων των παραπάνω σε πραγματικό δίκτυο (ή δίκτυα), γίνεται παρατήρηση και ανάλυση των δεδομένων που έχουν συγκεντρωθεί, για την εξαγωγή κατάλληλων συμπερασμάτων που αφορούν τη θεωρούμενη "ύποπτη" ή "κακόβουλη" κίνηση που έχει παρατηρηθεί από το network telescope.

Ακολουθεί μία σχηματική αναπαράσταση για τη δομή της εφαρμογής που πρόκειται να αναπτυχθεί :




***Εικόνα 11:*** *Αναπαράσταση της δομής της προτεινόμενης εφαρμογής που αναπτύχθηκε για την υλοποίηση ενός Network Telescope.*

Προτού προχωρήσουμε στην αναλυτική περιγραφή, θα αναφερθούμε σε δύο σημαντικά πρωτόκολλα τα οποία χρησιμοποιούνται από την εφαρμογή μας για την εξαγωγή χρήσιμων δεδομένων. Πρόκειται για τα πρωτόκολλα SNMP και OSPF.

### 5.1.1 Πρωτόκολλο διαχείρισης SNMP

Η αρχιτεκτονική που προτείνεται και χρησιμοποιείται σήμερα για την διαχείριση δικτύων υπολογιστών αποτελείται από το σύστημα διαχείρισης του δικτύου (Network Management System, **NMS**) ή το Σύστημα Λειτουργίας (Operation System, **OS**) και τα στοιχεία εκείνα των δικτύων (Network Elements, **NE**) τα οποία θέλουμε να διαχειριστούμε. Τέτοια NE's σε ένα δίκτυο είναι κυρίως μηχανήματα αποθήκευσης ή επεξεργασίας πληροφοριών, όπως hosts, workstations, καθώς και μηχανήματα διασύνδεσης δικτύων, όπως routers, bridges, repeaters κ.ά. στα οποία τρέχουν διαδικασίες διαχείρισης, που ονομάζονται αντιπρόσωποι διαχείρισης (**agents**) και είναι υπεύθυνες για την εκτέλεση των συναρτήσεων που καλούν οι διαχειριστές.

Για τη μεταφορά της πληροφορίας μεταξύ των διαχειριστικών συστημάτων και των διαχειριζόμενων στοιχείων, χρησιμοποιούνται κατάλληλα πρωτόκολλα μεταφοράς της πληροφορίας που αφορά τη διαχείριση. Τα πρωτόκολλα αυτά ονομάζονται **Πρωτόκολλα Διαχείρισης Δικτύων** (Network Management Protocols, NMP's) και καθορίζουν με σαφήνεια τον τρόπο επικοινωνίας, τη μορφή και τη σημασία των μηνυμάτων που θα ανταλλαχθούν, όπως επίσης και τον τρόπο ορισμού και περιγραφής των στοιχείων που θέλουμε να διαχειριστούμε.

Ένα από τα πιο γνωστά και ευρέως χρησιμοποιούμενα πρωτόκολλα διαχείρισης είναι το **SNMP** (Simple Network Management Protocol). Η αρχιτεκτονική που προτείνει το SNMP ακολουθεί το μοντέλο διαχειριστή – αντιπροσώπου (manager–agent model), με τους σταθμούς διαχείρισης και τα στοιχεία του δικτύου τα οποία θέλουμε να διαχειριστούμε. Ο agent είναι κάποιο πρόγραμμα εξυπηρετητής που προσφέρει πληροφορία σχετική πάντα με τη διαχείριση και έχει στην κατοχή του μια συλλογή από μεταβλητές (στιγμιότυπα διαχειριζόμενων αντικειμένων – Managed Objects), όπως διευθύνσεις, τύπους interfaces κ.α. των οποίων οφείλει να γνωρίζει τις τιμές και να τις αποδίδει.

Το σύνολο των μεταβλητών αυτών αποτελεί τη Βάση Πληροφορίας Διαχείρισης (Management Information Base, **MIB**). Κάθε MIB δομείται βάσει μίας δενδρικής δομής, τα στοιχεία της οποίας αποτελούν τα διαχειριζόμενα αντικείμενα (objects). Κάθε στοιχείο έχει τη δική του ταυτότητα που το προσδιορίζει (object id), η οποία προκύπτει ανάλογα με τη θέση του αντικειμένου στο δέντρο της MIB.

Η στρατηγική διαχείρισης που υπονοείται στο SNMP, απαιτεί η παρακολούθηση της κατάστασης ενός δικτύου να πραγματοποιείται με αναζήτηση της κατάλληλης πληροφορίας μέσω της MIB. Με τον τρόπο αυτό είναι δυνατό να παρακολουθηθεί η απόδοση και η κατάσταση ενός δικτύου, να ελεγχθούν παράμετροι που αφορούν τη λειτουργία του, να αναφερθούν, να απομονωθούν και να αναλυθούν σφάλματα.

Η μεγάλη ανάπτυξη της τεχνολογίας των TCP/IP δικτύων οδήγησε στην ανάγκη για εργαλεία διαχείρισης των δικτύων αυτών. Το SNMP είχε σαν στόχο μια βραχυπρόθεσμη λύση για τη διαχείρισή τους και πραγματικά πέτυχε το στόχο του, ελαχιστοποιώντας την πολυπλοκότητα των συναρτήσεων διαχείρισης που πρέπει να πραγματοποιήσει κάποιος agent, αφήνοντας βέβαια την πολλή επεξεργασία στους managers. Άλλοι στόχοι που είχαν τεθεί κατά την ανάπτυξη του SNMP ήταν η επεκτασιμότητα και η ανεξαρτησία από την αρχιτεκτονική των μηχανημάτων που θα διαχειριζόταν, στόχοι οι οποίοι επιτεύχθηκαν μέχρι κάποιο σημείο.

### 5.1.2 Πρωτόκολλο δρομολόγησης OSPF

Το **OSPF** (Open Shortest Path First) είναι ένα εσωτερικό gateway πρωτόκολλο δρομολόγησης, σχεδιασμένο έτσι ώστε να χρησιμοποιείται εντός ενός μοναδικού αυτόνομου συστήματος. Βασίζεται στην τεχνολογία του **SPF** (shortest path first), που είναι παρόμοια με αυτή του αλγορίθμου Bellman-Ford που συναντάται σε άλλα πρωτόκολλα δρομολόγησης όπως το RIP – Routing Information Protocol. Στο OSPF, υπάρχουν μεμονωμένα link-state advertisements (**LSAs**) που περιγράφουν τα επιμέρους στοιχεία τα οποία ανήκουν στην περιοχή δρομολόγησης (routing domain) του πρωτοκόλλου. Τα LSAs είναι μικρά και καθένα από αυτά περιγράφει ένα κομμάτι του OSPF routing domain και ιδιαίτερα τα στοιχεία που γειτονεύουν με ένα συγκεκριμένο router ή ένα μεμονωμένο transit network, μία εσωτερική ή εξωτερική

διαδρομή του δικτύου, ή ακόμη και ένα ολόκληρο αυτόνομο σύστημα (AS). Τα συγκεκριμένα LSAs διοχετεύονται μέσω του routing domain, σχηματίζοντας μία αντίστοιχη βάση δεδομένων (Link-State Database). Κάθε router σε ένα domain έχει τη δική του χαρακτηριστική link-state database. Με βάση αυτήν, κάθε router δημιουργεί δυναμικά ένα δέντρο δρομολόγησης (routing table), υπολογίζοντας το ελάχιστο μονοπάτι του δέντρου (shortest-path tree), με τον ίδιο τον δρομολογητή να αποτελεί κάθε φορά τη ρίζα του αντίστοιχου δέντρου. Ο συγκεκριμένος υπολογισμός αναφέρεται κοινώς ως ο αλγόριθμος του Dijkstra.

#### 5.1.2.1 Link State Advertisements (LSAs)

Όπως αναφέρθηκε και προηγουμένως, κάθε δρομολογητής σε ένα Αυτόνομο Σύστημα δημιουργεί ένα ή περισσότερα LSAs, το σύνολο των οποίων αποτελεί τη Link-State Database. Κάθε LSA ξεκινάει με μία καθορισμένης μορφής επικεφαλίδα (header) των 20 bytes, η οποία αποτελείται μεταξύ άλλων από τα εξής πεδία :
**LS type, Link State ID** και **Advertising Router.** Ο συνδυασμός των τριών αυτών πεδίων χαρακτηρίζει με μοναδικό τρόπο το κάθε LSA.

Σε ένα Αυτόνομο Σύστημα μπορεί να υπάρχουν ταυτόχρονα διαφορετικά στιγμιότυπα από ένα συγκεκριμένο LSA. Στην περίπτωση αυτή, θα πρέπει να καθοριστεί ποιο είναι το πιο πρόσφατο. Ο καθορισμός αυτός μπορεί να γίνει ύστερα από εξέταση των **LS sequence**, **LS checksum** και **LS age** πεδίων. Τα πεδία αυτά περιλαμβάνονται επίσης μέσα στην επικεφαλίδα των 20 bytes.

Αρκετοί τύποι OSPF πακέτων δημιουργούν LSAs. Στην περίπτωση που η χρονική στιγμή δημιουργίας τους δεν έχει ιδιαίτερη σημασία, τότε το LSA χαρακτηρίζεται από τα τρία πεδία των LS type, Link State ID and Advertising Router. Διαφορετικά, τα πεδία των LS sequence number, LS age and LS checksum θα πρέπει επίσης να αναφέρονται.

Πιο αναλυτικά, τα τρία πρώτα πεδία της επικεφαλίδας έχουν ως εξής :

- **LS Type**

  Το πεδίο LS type είναι αυτό που καθορίζει τη μορφή και τη λειτουργία του LSA. Διαφορετικού τύπου LSAs έχουν και διαφορετικά ονόματα (π.χ. router-LSAs ή network-LSAs). Όλοι οι τύποι των LSAs που θα αναφερθούν και στη συνέχεια,

διοχετεύονται μέσω μίας μοναδικής περιοχής, εκτός των AS-external-LSAs (LS type = 5), που διοχετεύονται μέσα από ολόκληρο το Αυτόνομο Σύστημα.
Υπάρχουν 5 βασικοί διακριτοί τύποι LSAs, καθένας εκ των οποίων έχει μία ξεχωριστή λειτουργία. Ο παρακάτω πίνακας εξηγεί συνοπτικά τη λειτουργία του κάθε τύπου :

| **LS Type** | **LSA description** |
|---|---|
| 1<br>**router-LSAs** | Κάθε δρομολογητής σε μία περιοχή του δικτύου παράγει router-LSAs τα οποία περιγράφουν την κατάσταση και το κόστος των στοιχείων της περιοχής (για παράδειγμα τα interfaces) που είναι συνδεδεμένα στο δρομολογητή. Όλες οι συνδέσεις ενός router σε μία περιοχή περιγράφονται από ένα μοναδικό router-LSA. |
| 2<br>**network-LSAs** | Παράγονται από κάθε broadcast και NBMA δίκτυο σε μία περιοχή η οποία υποστηρίζει δύο ή περισσότερους δρομολογητές. Τα network-LSAs παράγονται από έναν καθορισμένο δρομολογητή του δικτύου (Designated Router) και καθένα από αυτά περιγράφει τους δρομολογητές που είναι συνδεδεμένοι στο δίκτυο, συμπεριλαμβανομένου και του ίδιου του Designated Router. |
| 3 ή 4<br>**summary-LSAs** | Παράγονται από δρομολογητές που βρίσκονται στα σύνορα της περιοχής του δικτύου. Τα summary-LSAs τύπου |

| | |
|---|---|
| | 3 χρησιμοποιούνται όταν ο προορισμός είναι ένα IP δίκτυο και περιγράφουν διαδρομές για προορισμούς εντός της περιοχής του δικτύου (inter-area routes), ενώ τα summary-LSAs τύπου 4 περιγράφουν διαδρομές προς τους δρομολογητές που βρίσκονται στα σύνορα των Αυτόνομων Συστημάτων. |
| 5<br>**AS-external-LSAs** | Πηγάζουν από τους δρομολογητές στα σύνορα των Αυτόνομων Συστημάτων και περιγράφουν διαδρομές για εξωτερικούς προορισμούς . Μία default διαδρομή για ένα Αυτόνομο Σύστημα μπορεί επίσης να περιγραφεί από ένα AS-external-LSA. |

- **Link State ID**

Στο πεδίο αυτό καθορίζεται η ταυτότητα της περιοχής δρομολόγησης που περιγράφεται από το κάθε LSA και ανάλογα με τον τύπο του, το Link State ID παίρνει διαφορετικές τιμές, οι οποίες αναγράφονται στον παρακάτω πίνακα. Για τα summary-LSAs τύπου 3 και τα AS-external-LSAs τύπου 5, το Link State ID μπορεί να έχει επιπλέον ένα ή περισσότερα bits καθορισμένα, τα οποία αναφέρονται στον destination network host. Αυτή η δυνατότητα να καθορίζονται συγκεκριμένα bits για τον host, επιτρέπει στον router να δημιουργεί ξεχωριστά LSAs για δύο δίκτυα με την ίδια διεύθυνση αλλά διαφορετική μάσκα.

Όταν ένα LSA τύπου 2,3 ή 5 περιγράφει ένα δίκτυο, τότε η IP του δικτύου παράγεται εύκολα, εφαρμόζοντας στο πεδίο του Link State ID τη μάσκα της

μορφής δίκτυο/υποδίκτυο που περιέχεται στον κορμό του LSA και συγκεκριμένα στο πεδίο Network Mask. Όταν το LSA είναι τύπου 1 ή 4 και αναφέρεται σε κάποιον router, τότε η τιμή του πεδίου Link State ID είναι πάντοτε η OSPF Router ID του δρομολογητή ο οποίος περιγράφεται από το LSA. Τέλος, όταν πρόκειται για AS-external-LSA τύπου 5, το οποίο αναφέρεται σε μία default διαδρομή, τότε η τιμή του Link State ID έχει τεθεί στην DefaultDestination (0.0.0.0).

Στον παρακάτω πίνακα συνοψίζονται οι τιμές που φέρει το πεδίο **Link State ID** για κάθε τύπο LSA :

| **LS Type** | **Link State ID** |
|---|---|
| 1 | Η Router ID του δρομολογητή-πηγή. |
| 2 | Η IP διεύθυνση του interface που συνδέεται στον καθορισμένο router του δικτύου. |
| 3 | Η IP διεύθυνση του destination network. |
| 4 | Η Router ID του περιγραφόμενου AS router στα σύνορα του δικτύου. |
| 5 | Η IP διεύθυνση του destination network. |

- **Advertising Router**

Στο πεδίο αυτό καθορίζεται η OSPF Router ID του δρομολογητή ο οποίος αποτελεί την πηγή παραγωγής των LSAs (originator router). Για router-LSAs, η τιμή του πεδίου αυτού είναι ταυτόσημη με την αντίστοιχη τιμή του Link State ID πεδίου. Τα network-LSAs παράγονται από έναν καθορισμένο δρομολογητή του δικτύου (network's Designated Router), ενώ τα summary-LSAs παράγονται από δρομολογητές των συνόρων της περιοχής (area border routers). Τέλος, τα AS-external-LSAs παράγονται από τους boundary routers.

Το πρωτόκολλο OSPF συνοδεύεται από την αντίστοιχη Βάση Πληροφορίας Διαχείρισης ή MIB, από την οποία μπορούν να αντληθούν τα απαραίτητα στοιχεία διαχείρισης με χρήση του πρωτοκόλλου SNMP, σύμφωνα με τη διαδικασία που περιγράφηκε προηγουμένως.

Μία από τις τελευταίες εκδόσεις της OSPF MIB που χρησιμοποιούνται αυτή τη στιγμή και υποστηρίζεται από τη Cisco, είναι η **RFC 1253 OSPF Version 2 MIB**. Στη συνέχεια, θα κάνουμε χρήση των αντικειμένων που περιέχονται στην εν λόγω MIB, προκειμένου να αντλήσουμε τις απαραίτητες πληροφορίες που θα χρειαστούμε στα πλαίσια της εφαρμογής μας.

## 5.2 ΑΝΑΛΥΣΗ ΚΑΙ ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στη συνέχεια, ακολουθεί μία αναλυτική περιγραφή των επιμέρους στοιχείων που συνθέτουν τη λειτουργία του network telescope έτσι όπως αυτό αναπτύχθηκε, καθώς και των διάφορων τεχνικών που χρησιμοποιήθηκαν για την υλοποίησή του.

### 5.2.1 Προσδιορισμός του unallocated address space

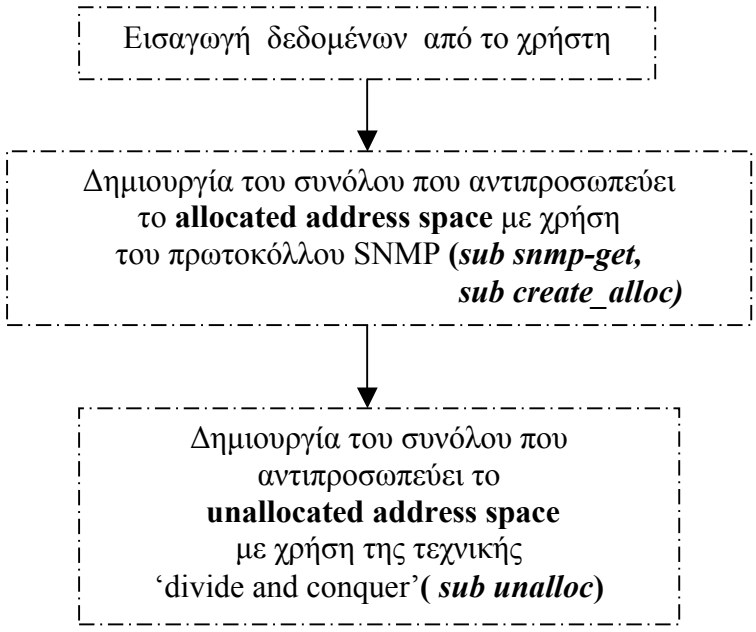
Το πρώτο μέρος της υλοποίησης του network telescope που δημιουργήσαμε βασίζεται στην εφαρμογή ενός κώδικα, ο οποίος υλοποιεί τα δύο πρώτα βήματα έτσι όπως τα περιγράψαμε στην αρχή του κεφαλαίου. Με χρήση του πρωτοκόλλου SNMP, πραγματοποιείται η συλλογή των αποδιδόμενων σε χρήστες διευθύνσεων του δικτύου, που αποτελούν το λεγόμενο **allocated address space** και στη συνέχεια ακολουθεί η εξαγωγή του συνόλου των μη αποδιδόμενων διευθύνσεων ή **unallocated address space** του δικτύου. Παράλληλα δημιουργείται κατάλληλο φίλτρο που χρησιμεύει στη συνέχεια για τον καθορισμό του διαχειριζόμενου address space, προς το οποίο θέλουμε να μελετήσουμε την κίνηση.

Ο συγκεκριμένος κώδικας είναι υλοποιημένος σε γλώσσα *perl* και εκτελείται σαν μία ενιαία εφαρμογή, ζητώντας από το χρήστη να ορίσει αρχικά κάποιες

παραμέτρους, οι οποίες δίνονται από τον ίδιο ως inputs, μέσω του αντίστοιχου configuration file. Τα δεδομένα που προσδιορίζονται από το χρήστη είναι τα εξής :

- η IP διεύθυνση ενός από τους routers του δικτύου (router_ip) από τον οποίο θα εξαχθούν όλες οι χρησιμοποιούμενες IPs του δικτύου, με χρήση του πρωτοκόλλου SNMP και της OSPF MIB (υπό τη μορφή LSAs).
- η IP διεύθυνση ή το σύνολο των IP διευθύνσεων που αντιπροσωπεύουν ολόκληρο το δίκτυο (network_ips) και περιλαμβάνουν όλες τις διαθέσιμες IPs, τόσο αυτές που έχουν αποδοθεί για χρήση, όσο και αυτές που δεν έχουν αποδοθεί (π.χ. 147.102.0.0/16 για το Πολυτεχνείο).
- το όνομα του αρχείου (data_file) από το οποίο μπορούν να εισαχθούν απευθείας τα δεδομένα που αφορούν το allocated address space του δικτύου, παραλείποντας τη λειτουργία εύρεσής τους μέσω του πρωτοκόλλου SNMP. Σε περίπτωση όμως που πραγματοποιείται κανονικά η παραπάνω λειτουργία, η παράμετρος "data_file" θα πρέπει να έχει την default τιμή "data".
- το όνομα του φίλτρου που ο χρήστης επιθυμεί να δημιουργηθεί με την εκτέλεση του προγράμματος, το οποίο αντιστοιχεί στο unallocated address space του δικτύου.

Ύστερα από την εισαγωγή των απαραίτητων δεδομένων, οι ενέργειες που πραγματοποιούνται κατά τη διάρκεια εκτέλεσης του παραπάνω κώδικα, μπορούν να αναπαρασταθούν σχηματικά ως εξής :



Για την εκτέλεση του δεύτερου και του τρίτου βήματος του παραπάνω αλγορίθμου, ο κυρίως κώδικας καλεί αντίστοιχα δύο υπορουτίνες (functions), οι οποίες υλοποιούν τις αντίστοιχες ενέργειες. Στη συνέχεια περιγράφουμε περισσότερο αναλυτικά καθεμία από αυτές.

- Συνάρτηση '**sub snmp-get'**

  Ο κώδικας που εκτελείται με την κλήση της συγκεκριμένης συνάρτησης έχει σκοπό τη συγκέντρωση πληροφοριών που αφορούν το σύνολο των IP διευθύνσεων που έχουν αποδοθεί κανονικά προς χρήση και αποτελούν το **allocated address space** του δικτύου. Η απόκτηση αυτών των δεδομένων πραγματοποιείται με χρήση του πρωτοκόλλου διαχείρισης SNMP, το οποίο ανακτά πληροφορίες από τα διαχειριζόμενα αντικείμενα που βρίσκονται στην RFC 1253 OSPF Version 2 MIB που αναφέραμε προηγουμένως. Ύστερα από εξέταση της εν λόγω MIB, παρατηρήσαμε ότι το κατάλληλο αντικείμενο που μας προσφέρει τις πληροφορίες που χρειαζόμαστε είναι το ospfLsdbAdvertisement με object id : 1.3.6.1.2.1.14.4.1.8 , το οποίο ανήκει με τη σειρά του στο αντικείμενο **OspfLsdbEntry,** που αφορά την OSPF Process's Link State Database, όπως φαίνεται και από τον παρακάτω πίνακα :

Table: **ospfLsdbTable**
The OSPF Process's Link State Database.

| **OspfLsdbEntry** | | | |
|---|---|---|---|
| **Name** | **Syntax** | **Access** | **Registration** |
| ospfLsdbAreaId | OID | read-only | 1.3.6.1.2.1.14.4.1.1 |
| ospfLsdbType | Enumerated | read-only | 1.3.6.1.2.1.14.4.1.2 |
| ospfLsdbLsid | OID | read-only | 1.3.6.1.2.1.14.4.1.3 |
| ospfLsdbRouterId | OID | read-only | 1.3.6.1.2.1.14.4.1.4 |
| ospfLsdbSequence | Integer32 | read-only | 1.3.6.1.2.1.14.4.1.5 |
| ospfLsdbAge | Integer32 | read-only | 1.3.6.1.2.1.14.4.1.6 |
| ospfLsdbChecksum | Integer32 | read-only | 1.3.6.1.2.1.14.4.1.7 |
| **ospfLsdbAdvertisement** | **Octet String** | **read-only** | **1.3.6.1.2.1.14.4.1.8** |

Στη συνέχεια φαίνονται και οι ακριβείς πληροφορίες που παρέχει η MIB για το συγκεκριμένο αντικείμενο :

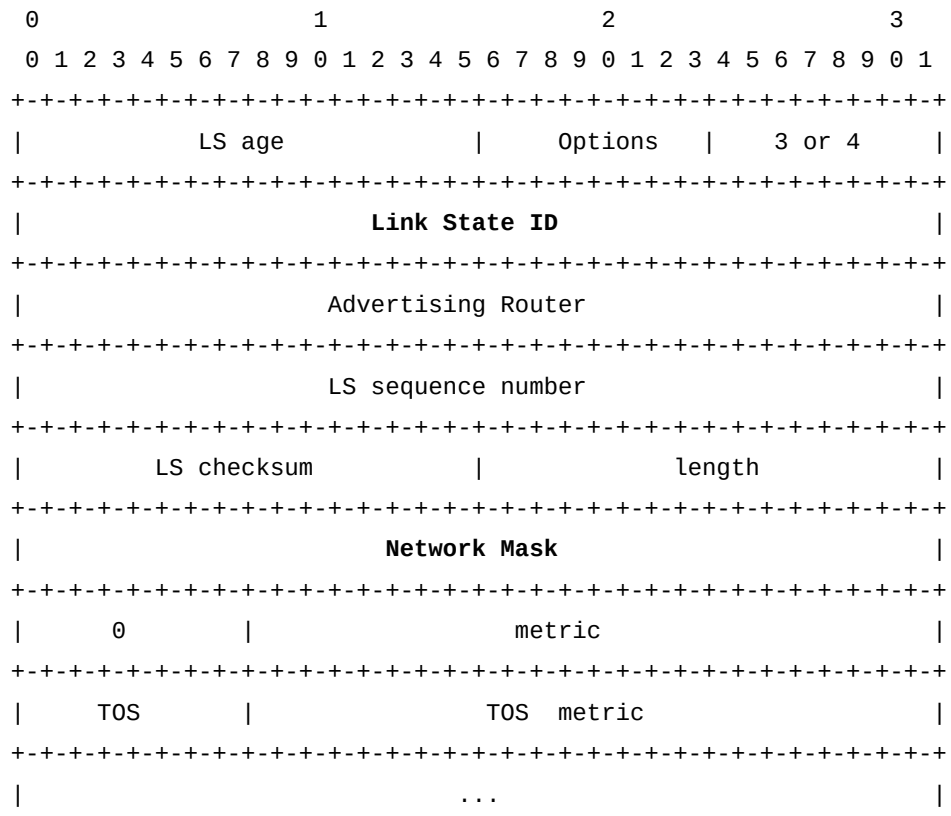
**ospfLsdbAdvertisement** OBJECT-TYPE

```
Registration:  1.3.6.1.2.1.14.4.1.8
Access:        read-only
Status:        mandatory
Description:   The entire Link State Advertisement, including
               its header.
Reference:    "OSPF Version 2, Section 12 Link State Advertisements"
::= { ospfLsdbEntry 8 }
```

Τα δεδομένα που ανακτάμε από το παραπάνω αντικείμενο με κλήση της συνάρτησης **snmpwalk** βρίσκονται σε μορφή LSAs.

Τα περισσότερα είναι τύπου 3, δηλαδή summary-LSAs, που είναι και αυτά που μας ενδιαφέρουν, καθώς περιγράφουν τις IPs των δικτύων και υποδικτύων που είναι συνδεδεμένα με τον Designated Router, μέσω του οποίου εκτελείται και το snmpwalk.

Οι πληροφορίες που ανακτάμε είναι σε μορφή σειράς δεκαεξαδικών ψηφίων και αποτελούνται από bytes, τα οποία ακολουθούν τη γενική μορφή αναπαράστασης των summary-LSAs έτσι όπως αυτή παρουσιάζεται στη συνέχεια :



Όπως φαίνεται από το παραπάνω σχήμα, τα bytes 24-31 αντιστοιχούν στο πεδίο `LS Type` που για τα summary-LSAs έχει την τιμή 3 ή 4, ενώ από τα υπόλοιπα πεδία, αυτά που μας ενδιαφέρουν είναι το `Link State ID` που αντιπροσωπεύει την IP διεύθυνση του δικτύου και το πεδίο `Network Mask` που αντιστοιχεί στη μάσκα της IP διεύθυνσης. Ο συνδυασμός των bytes που αποτελούν τα δύο τελευταία πεδία μπορεί να μας δώσει έπειτα από κατάλληλη επεξεργασία την ολοκληρωμένη CIDR αναπαράσταση της IP διεύθυνσης για κάθε υποδίκτυο, στη δεκαδική της μορφή xxx.xxx.xxx.xxx./mask. Τη λειτουργία αυτή πραγματοποιεί η επόμενη συνάρτηση :

- Συνάρτηση '**sub create_alloc'**

Η συνάρτηση αυτή δέχεται ως είσοδο τα δεδομένα που προέκυψαν από την εκτέλεση της 'snmp_get'και στη συνέχεια αναλαμβάνει την επεξεργασία και την κατάλληλη μορφοποίησή τους, με σκοπό τη δημιουργία συνόλου για το allocated address space του δικτύου, το οποίο θα περιέχει τις IP διευθύνσεις στη δεκαδική CIDR αναπαράσταση. Υπάρχει το ενδεχόμενο, τα παραπάνω δεδομένα να είναι ήδη διαθέσιμα σε κάποιο αρχείο και να μην υπάρχει η ανάγκη εύρεσής τους μέσω της συνάρτησης 'snmp_get'. Στην περίπτωση αυτή, η εκτέλεση της 'snmp_get' παραλείπεται και η 'create_alloc' δέχεται τα δεδομένα εισόδου απευθείας από το εν λόγω αρχείο, το οποίο προσδιορίζεται μέσω του configuration file.

Ένα παράδειγμα εκτέλεσης της **`'snmp_get'`** φαίνεται στη συνέχεια, όπου τα αποτελέσματα που παίρνουμε με την κλήση της βρίσκονται στην αρχική δεκαεξαδική τους μορφή, προτού υποστούν την κατάλληλη επεξεργασία:

```
SNMPv2-SMI::mib-2.14.4.1.8.0.0.0.0.1.147.102.255.2.147.102.255.2 = Hex-
STRING: 00 01 22 01 93 66 FF 02 93 66 FF 02 80 00 0B 16
AA AB 00 6C 03 00 00 07 C2 B1 D0 78 FF FF FF FC
03 00 00 30 93 66 FF 03 FF FF FF FF 03 00 00 01
93 66 FF 02 FF FF FF FF 03 00 00 01 93 66 FF 03
93 66 E0 16 01 00 00 01 93 66 E0 14 FF FF FF FC
03 00 00 01 93 66 FF 05 93 66 E0 0E 01 00 03 E8
93 66 E0 0C FF FF FF FC 03 00 03 E8
SNMPv2-SMI::mib-2.14.4.1.8.0.0.0.0.1.147.102.255.3.147.102.255.3 = Hex-
STRING: 00 01 22 01 93 66 FF 03 93 66 FF 03 80 00 16 6C
DB B7 00 54 02 00 00 05 93 66 FF 03 FF FF FF FF
03 00 00 01 93 66 FF 01 FF FF FF FF 03 00 00 01
```

```
93 66 E0 2D 93 66 E0 2D 02 00 00 01 93 66 FF 02
93 66 E0 15 01 00 00 01 93 66 E0 14 FF FF FF FC
03 00 00 01
SNMPv2-SMI::mib-2.14.4.1.8.0.0.0.0.1.147.102.255.4.147.102.255.4 = Hex-
STRING: 00 01 22 01 93 66 FF 04 93 66 FF 04 80 00 00 9B
BA 98 00 48 02 00 00 04 93 66 FF 04 FF FF FF FF
03 00 00 01 93 66 FF 03 FF FF FF FF 03 00 00 01
93 66 E0 31 93 66 E0 32 02 00 00 01 93 66 FF 06
FF FF FF FF 03 00 00 01
```

Με κλήση της **'create_alloc'** και ύστερα από κατάλληλη επεξεργασία, τα αρχικά δεδομένα μετατρέπονται στην CIDR αναπαράσταση :

```
147.102.224.45/255.255.255.252
147.102.224.49/255.255.255.252
147.102.1.0/255.255.255.0
```

Συμπερασματικά λοιπόν μπορούμε να πούμε, πως αυτό που επιτυγχάνεται με την κλήση των δύο παραπάνω συναρτήσεων, είναι ουσιαστικά η συγκέντρωση των IP διευθύνσεων των υποδικτύων που υπάρχουν στο ευρύτερο δίκτυο που μελετάμε και οι οποίες έχουν αποδοθεί κανονικά σε χρήστες. Αυτό πραγματοποιείται όπως προαναφέραμε, με κλήση της snmpwalk σε καθορισμένο δρομολογητή του δικτύου, ενώ τα στοιχεία που συγκεντρώνουμε, τα επεξεργαζόμαστε κατάλληλα προκειμένου να εξασφαλίσουμε την επιθυμητή δεκαδική μορφή των παραπάνω διευθύνσεων.

Το σύνολο των διευθύνσεων αυτών, που στην ουσία αποτελούν το allocated address space του δικτύου, αποθηκεύεται σε ένα αρχείο με το όνομα 'results' στην προκειμένη περίπτωση. Τέλος, προκειμένου να ελαχιστοποιήσουμε τον αριθμό των παραπάνω διευθύνσεων που έχουμε συγκεντρώσει, αυξάνοντας παράλληλα την απόδοση του αλγορίθμου, εφαρμόζουμε ένα είδος συμπίεσης των δεδομένων με εκτέλεση ενός κατάλληλου script που έχουμε ονομάσει 'comp.pl'. Η εφαρμογή αυτή δέχεται στην είσοδό της τη λίστα με τις διευθύνσεις που περιέχονται στο αρχείο 'results' και επιστρέφει ένα νέο αρχείο με το όνομα 'new_res', το οποίο περιέχει μία νέα λίστα, με τις παραπάνω διευθύνσεις των υποδικτύων να βρίσκονται σε όσο το δυνατόν περισσότερο συμπιεσμένη μορφή.

Το κομμάτι του κώδικα που υλοποιεί τις συναρτήσεις **'snmp_get'** και **'create_alloc'** έτσι όπως τις περιγράψαμε, παρουσιάζεται στη συνέχεια γραμμένο σε γλώσσα *perl*, ενώ ακολουθεί και η εφαρμογή **'comp.pl'** η οποία πραγματοποιεί τη διαδικασία συγχώνευσης των διευθύνσεων :

```
sub snmp_get {

open (DATA, '>data');

$SNMP_GET_CMD = "snmpwalk -v 2c -c public";
$SNMP_TARGET = $_[0];

# collection of all the allocated IP addresses
$snmp_res = `${SNMP_GET_CMD} ${SNMP_TARGET} .1.3.6.1.2.1.14.4.1.8`;
print DATA "$snmp_res\n";
close DATA;
}

sub create_alloc {
open (DATA, $_[0]);

while (<DATA>) {
push @values, <DATA>; }

$values = join(" ",@values);
#give the results the formal CIDR format
@lines = split(/SNMPv2-SMI::/,$values);
$numoflines=$#lines;
my ($i)=0;
open (RESULTS, '>results') || die "cannot create results: $!";
while ($i<=$numoflines)
{
@lineparts=split(/Hex-STRING:/,$lines[$i]);
$linkadv=$lineparts[1];
$linkadv=~s/\n//g;
$linkadv=~s/\s//g;
if ((substr($linkadv,6,2)) ne ("01" && "04"))
{
 $ip=substr($linkadv,8,8);
 $addr=pack("H*",$ip);
 $addr=join(".",unpack("C*",$addr));
 print RESULTS "$addr/";
```

```
 $mask=substr($linkadv,,40,8);
 $mask=pack("H*",$mask);
 $mask=join(".",unpack("C*",$mask));
 print RESULTS "$mask\n";

}
$i=$i+1;
}
#make the subnets' IPs as summarized as possible
system ("cat results |./comp.pl > new_res");
close (RESULTS);
}
```

**comp.pl**

```
#!/usr/bin/perl

use Net::IP;
use NetAddr::IP;
 while (<STDIN>)
 {
  $ip=NetAddr::IP->new($_);
  if ($ip ne (new NetAddr::IP 'default'))
 { push (@addresses, $ip);}

 }
```

- Συνάρτηση **'sub unalloc'**

  Στο κομμάτι του κώδικα που εκτελείται με την κλήση της συνάρτησης αυτής, πραγματοποιείται η διαδικασία εύρεσης του **unallocated address space** του δικτύου, του συνόλου δηλαδή των διευθύνσεων που δεν έχουν αποδοθεί σε χρήστες και την κίνηση προς τις οποίες μας ενδιαφέρει να μελετήσουμε. Στην ουσία, αυτό που έχουμε να κάνουμε είναι η αφαίρεση δύο συνόλων, του συνόλου που αντιπροσωπεύει το 'total address space' του δικτύου και αυτού που δημιουργήσαμε στο προηγούμενο βήμα και αφορά το 'allocated address space'.

  Η τεχνική που εφαρμόζεται στον αλγόριθμο της συνάρτησης ακολουθεί τη λογική του **'divide and conquer'**. Ξεκινώντας από την αρχική IP διεύθυνση που αντιπροσωπεύει ολόκληρο το δίκτυο και η οποία δίνεται ως όρισμα με την κλήση της συνάρτησης, αυξάνουμε διαδοχικά τα bits της μάσκας υποδιαιρώντας το σε μικρότερα υποδίκτυα. Κάθε φορά συγκρίνουμε το κάθε νέο υποδίκτυο που

δημιουργείται με όλες τις IPs των υποδικτύων που περιέχονται στο αρχείο ‘new_res’ και αποτελούν το ‘allocated address space’. Εάν οι δύο IPs ταυτίζονται, τότε θεωρούμε το υποδίκτυο που εξετάζουμε ‘allocated’ και συνεχίζουμε με τον έλεγχο των υπολοίπων. Εάν απλά το πρώτο εμπεριέχει κάποιο από τα υπόλοιπα υποδίκτυα του αρχείου αλλά είναι μεγαλύτερου μεγέθους, τότε συνεχίζουμε την υποδιαίρεσή του σε ακόμη μικρότερα υποδίκτυα στην επόμενη επανάληψη του αλγορίθμου. Τέλος, εάν το υποδίκτυο το οποίο εξετάζουμε δεν βρίσκεται μέσα στο allocated address space, τότε το εκλαμβάνουμε ως ‘unallocated subnetwork’ και καταχωρούμε την IP διεύθυνσή του σε αντίστοιχο αρχείο. Η μέθοδος αυτή εφαρμόζεται επαναληπτικά, μέχρις ότου βρεθούν και καταχωρηθούν όλες οι IPs των μη αποδιδόμενων υποδικτύων.

Πιο αναλυτικά, η λογική του αλγορίθμου που περιγράψαμε παρουσιάζεται στη συνέχεια με τη μορφή ψευδοκώδικα :

```
Αρχικοποιήσεις μεταβλητών.
Άνοιγμα αρχείου 'filter.def' για αποθήκευση των αποτελεσμάτων και
δημιουργία κατάλληλου φίλτρου.
Δημιουργία του array fragm[] για καταχώρηση των IPs των
υποδικτύων που χρειάζονται περαιτέρω διάσπαση σε κάθε επανάληψη.
Αρχικοποίηση του fragm[] με το συνολικό address space του
δικτύου.
Για (masklength=initial_value; masklength<=32; masklength++)
    {
       Δημιουργία του array space[] για αποθήκευση των νέων IPs
       των υποδικτύων που προκύπτουν από την κάθε διάσπαση.
       Για κάθε στοιχείο του fragm[]
       {  Υποδιαίρεση του υποδικτύου σε μικρότερα υποδίκτυα με
          μεγαλύτερη μάσκα, με χρήση της συνάρτησης split.
          Καταχώρηση των νέων υποδικτύων στο array space[].
       }
          Για κάθε στοιχείο του space
          {  Έλεγχος του κάθε νέου υποδικτύου που δημιουργήθηκε
             Για όλες τις allocated IPs του αρχείου new_res
             {  Σύγκριση της κάθε IP με την IP του υποδικτύου που
                εξετάζουμε
                Περίπτωση 1η : το υποδίκτυο περιλαμβάνει την IP και
                               έχουν μάσκα του ίδιου μεγέθους,
```

```
                    δηλαδή ταυτίζονται.
                    Έξοδος από το βρόχο.

        Περίπτωση 2η : το υποδίκτυο περιλαμβάνει την IP αλλά
                    είναι μεγαλύτερου μεγέθους οπότε
                    χρειάζεται περαιτέρω υποδιαίρεση.
                    Αποθήκευση της IP του υποδικτύου στο
                    array fragm για περαιτέρω έλεγχο.
                    Έξοδος από το βρόχο.

        Περίπτωση 3η :το υποδίκτυο δεν περιλαμβάνει την IP
                    Αύξηση κατάλληλου μετρητή
    }

    Εάν το υποδίκτυο που εξετάσαμε δεν περιλαμβάνει
    καμία από τις allocated IPs του αρχείου new_res,
    τότε το θεωρούμε 'unallocated' και καταχωρούμε την IP
    του στο αρχείο filter.def
     Αλλιώς συνεχίζουμε τον έλεγχο με το επόμενο υποδίκτυο
    }
    Επαναλαμβάνουμε το αρχικό loop και συνεχίζουμε την
    υποδιαίρεση σε μικρότερα υποδίκτυα, μέχρις ότου βρεθούν
    όλα τα unallocated υποδίκτυα και δημιουργηθεί το
    αντίστοιχο φίλτρο.
}
```

Όπως φαίνεται, η προσέγγιση 'divide-and-conquer' βασίζεται στην υποδιαίρεση του αρχικού προβλήματος σε άλλα υποπροβλήματα, που είναι παρόμοια με το αρχικό αλλά μικρότερα σε μέγεθος. Η διαδικασία αυτή επαναλαμβάνεται αναδρομικά, μέχρις ότου καταλήξουμε σε προβλήματα των οποίων η λύση είναι απλή ή προφανής. Στη συνέχεια, καθένα από αυτά επιλύεται ξεχωριστά και στο τέλος, οι επιμέρους λύσεις των διαφόρων υποπροβλημάτων συντίθενται, προκειμένου να δημιουργήσουν μια ολοκληρωμένη μορφή λύσης για το αρχικό πρόβλημα.

Η μέθοδος αυτή, που εφαρμόζεται για διαδοχικές συγκρίσεις μεταξύ δύο συνόλων, αποδεικνύεται στην περίπτωσή μας σαφώς πιο αποτελεσματική και λιγότερο χρονοβόρα σε σχέση με τη σύγκριση όλων των στοιχείων των δύο συνόλων ένα προς

ένα. Ενδεικτικά αναφέρουμε, ότι εφαρμογή του αλγορίθμου ‘divide-and-conquer’ στο υποδίκτυο ‘147.102.222.0/29’ απαιτεί μηδαμινό χρόνο εκτέλεσης και 150 συνολικά επαναλήψεις, τη στιγμή που ο αλγόριθμος σύγκρισης ένα προς ένα απαιτεί αντιστοίχως 25.3936 αριθμό επαναλήψεων. Για το συνολικό εύρος του δικτύου, ‘147.102.0.0/16’, χρειάζονται 15.350 επαναλήψεις συνολικά από τον πρώτο αλγόριθμο, οι οποίες ολοκληρώνονται σε 8 sec, ενώ ο δεύτερος απαιτεί πολύ μεγάλο χρόνο και τεράστιο αριθμό επαναλήψεων.

Το αρχείο ‘filter.def’ που επιστρέφει η συνάρτηση “unalloc” περιέχει ουσιαστικά το σύνολο των διευθύνσεων των υποδικτύων που συνθέτουν το unallocated address space και έχει δημιουργηθεί με τέτοιο τρόπο, ώστε να έχει την κατάλληλη μορφή, που θα επιτρέπει στο χρήστη να το χειριστεί στη συνέχεια σαν φίλτρο, κατά τη διαδικασία συλλογής δεδομένων κίνησης του δικτύου με χρήση των flow-tools. Με τον τρόπο αυτό αυτοματοποιείται η διαδικασία εύρεσης του συνόλου των μη αποδιδόμενων διευθύνσεων, την κίνηση των οποίων μας ενδιαφέρει να μελετήσουμε, βασιζόμενοι στη λογική λειτουργίας του network telescope.

Ο κώδικας που υλοποιεί τη διαδικασία που περιγράψαμε παρουσιάζεται στη συνέχεια:

```
sub unalloc {

use Net::IP;
use NetAddr::IP;

# masklength twn subnets pou dimiourgountai me xrisi tis 'split'

# synolo IPs sto arxeio 'results'
 $num=0;

 open (COUNT, "new_res") || die "cannot open new_res: $!";

  for $ip (map { NetAddr::IP->new($_) } <COUNT>)
  { $num = $num+1; }

 close (COUNT);
# print "\n",$num;

# array opou kataxwrountai ta subnets pou xreiazontai epipleon
# fragmentation se kathe loop
 @fragm=();

# arxikopoiisi toy array me to synoliko address space toy diktyou

 $fragm[0]=$_[0];
 print "$fragm[0]\n";
```

```
 if ($_[1] ne '') {
 $fragm[1]=$_[1];
 }

  open (FILTER, '>>filter.def') || die "cannot open filter.def: $!";

  print FILTER "\n\nfilter-primitive test_$_[2]\n";
  print FILTER "type ip-address-prefix\n";


# method 'divide and conquer'

   $start_ip = NetAddr::IP->new($fragm[0]);
   $i=$start_ip->masklen();


  for (($i+1); $i<=32; $i++)
   {
        $j=0;
        $k=0;
        $index=0;


# array gia tin apothikeusi twn new subnets pou dimiourgountai kathe
# fora me ti split
        @space=();

# spasimo twn stoixeiwn tou 'fragm' se mikrotera ypodiktya me
# megalyteri maska ksekinontas apo ti '17'

        while ($index<=$#fragm)
         {

         join("\n", NetAddr::IP->new($fragm[$index])->split($i)), "\n";
         push @space, NetAddr::IP->new($fragm[$index])->split($i);

          $index=$index+1;

         }
      $k=0;
      $j=0;
      @fragm=();

# elegxos olwn twn stoixeiwn toy @space, diladi twn kainourgiwn subnets
# pou dimiourgise i split

 while ($j<=$#space)
  {
     $cnt=0;
     $check=0;

      open (RESULTS, "new_res") || die "cannot create new_res: $!";

# sygkrisi me oles tis allocated IPs pou periexontai sto arxeio
# "results"

    for my $ip (map { NetAddr::IP->new($_) } <RESULTS>)
    {
      $size=$ip->masklen();
```

```
# case 1 : to subnet pou eksetazetai perilamvanei tin IP kai exoun
# maska tou idiou megethous, diladi tautizontai

      if (($space[$j]->contains($ip)) && ($size eq $i))
                                                        {
#                                                  print "$ip allocated!\n";

                                                         close (RESULTS);
                                                         last;

                                                        }

# case 2 : to subnet perilamvanei tin IP alla exei megalytero megethos
# kai xreiazetai epipleon fragmentation

      elsif ($space[$j]->contains($ip))

# apothikeuoume tin IP tou sygkekrimenou subnet sto array 'fragm' gia
# epipleon elegxo sti synexeia

                              {
                                $fragm[$k]=$space[$j];
                                $k=$k+1;
                                close (RESULTS);
                                last;
                              }

# case 3: to subnet den perilamvanei tin IP

      else {
#        print $space[$j],",contains not $ip\n";
            $cnt=$cnt+1;  }

    }

# elegxoume ean to subnet den perilamvanei kamia apo tis IPs tou
# "results" opote to thewroume 'unallocated'

      if ($cnt eq $num) {
#                           print $space[$j]," not allocated!\n";
                          print FILTER "permit $space[$j]\n";
                        }

# synexizoume to spasimo se mikrotera notations mexri na vrethoun ola
# ta unallocated subnets

       $j=$j+1;

   }
 close (RESULTS);
}
print FILTER "default deny\n\n\n";
print FILTER "filter-definition test_$_[2]\n";
print FILTER "match ip-destination-address  $_[2]\n";
print FILTER "or\n";
print FILTER "match ip-source-address test_$_[2]\n";
close (FILTER);
}
```

Οι δύο συναρτήσεις που περιγράψαμε παραπάνω, καλούνται από το κυρίως πρόγραμμα, το οποίο έχουμε συμπεριλάβει στο αρχείο *'Telescope.pl'*. Για την εκτέλεσή του, χρησιμοποιείται ένα αντίστοιχο configuration file (*TelescopeConfig.pm*), όπου καταχωρούνται από το χρήστη τα απαραίτητα δεδομένα για την εκτέλεση του προγράμματος. Τα δεδομένα αυτά διαβάζονται από το κυρίως πρόγραμμα και χρησιμοποιούνται ως παράμετροι στη συνέχεια κατά την κλήση των συναρτήσεων *snmp_get, create_alloc* και *unalloc*.

Ένα παράδειγμα για το configuration file, καθώς και το κυρίως πρόγραμμα, φαίνονται στη συνέχεια :

**TelescopeConfig.pm**

```
%Telescope::Config = (
# Please give the router's IP for use by the SNMP protocol
                        router_ip => '147.102.255.5',
# Please give the network IPs that represent the total address space,
# in CIDR format ('xxx.xxx.xxx.xxx/mask') starting from the IP with the
# bigger mask
                        network_ip1 => '147.102.0.0/17',
                        network_ip2 => '147.102.128.0/17',
# Please give the file-name with the source-data of the allocated address
# space. In other case give the default value 'data'
                        data_file => 'data',
# Please give the name for the filter to be created
                        filter_name => 'telescope_ntua' );
```

**Telescope.pl**

```
#!/usr/local/bin/perl

use TelescopeConfig;

snmp_get ($Telescope::Config{router_ip});
create_alloc ($Telescope::Config{data_file});
unalloc ($Telescope::Config{network_ip1}, $Telescope::Config{network_ip2},
$Telescope::Config{filter_name});

sub snmp_get {
              . . . }

sub create_alloc {
              . . . }

sub unalloc {
              . . . }
```

Από το σημείο αυτό λοιπόν, σκοπός μας είναι η δημιουργία και η χρήση κατάλληλων εργαλείων, με τα οποία θα μπορούμε να καταγράψουμε και να μελετήσουμε την 'ύποπτη' θεωρητικά κίνηση προς το σύνολο των μη χρησιμοποιούμενων διευθύνσεων που έχουμε δημιουργήσει με τον τρόπο που περιγράψαμε. Αυτό είναι και το αντικείμενο του δεύτερου μέρους της εφαρμογής που περιγράφουμε στη συνέχεια.

### 5.2.2 Συλλογή και καταγραφή των δεδομένων κίνησης

#### 5.2.2.1 Τεχνολογία NetFlow

Έχοντας στη διάθεσή μας τα δεδομένα που συγκεντρώσαμε από το προηγούμενο μέρος της εφαρμογής, μπορούμε πλέον να καταγράψουμε και να παρατηρήσουμε στοιχεία που αφορούν την κίνηση σε συγκεκριμένα κομμάτια του δικτύου που μας ενδιαφέρουν, κάνοντας χρήση κατάλληλων εργαλείων, των λεγόμενων flow-tools . Τα εργαλεία αυτά μας προσφέρουν τη δυνατότητα να καταγράψουμε, να φιλτράρουμε, να τυπώσουμε και να αναλύσουμε σειρές από flows (flow logs), τα οποία προέρχονται από σχετικές Netflow εγγραφές (Netflow records). Τι είναι όμως η τεχνολογία Netflow και πώς ορίζεται ένα flow;

Το **NetFlow** είναι μία τεχνική διαχείρισης δικτυακής κίνησης, η οποία αναπτύχθηκε από τους Darren Kerr και Barry Bruins της Cisco Systems, το 1996. Η τεχνική αυτή περιγράφει τον τρόπο με τον οποίο ένας δρομολογητής αντλεί στατιστικά στοιχεία για δρομολογημένες σειρές πακέτων και αποτελεί πλέον ένα αναπόσπαστο χαρακτηριστικό για τους περισσότερους routers της Cisco όπως είναι ο Juniper,ο Extreme και άλλοι. Όταν ο διαχειριστής ενός δικτύου ενεργοποιεί την εξαγωγή Netflow δεδομένων σε ένα interface του δρομολογητή, τα διάφορα στατιστικά στοιχεία κίνησης των πακέτων που λαμβάνονται για το συγκεκριμένο interface, καταμετρούνται σαν ένα flow και καταγράφονται σε μία δυναμική μνήμη (flow cache).

Ένα **flow** ορίζεται ως μία σειρά πακέτων μονής κατεύθυνσης μεταξύ δύο σημείων ενός δικτύου. Αυτό σημαίνει πως σε μία σύνδεση μεταξύ ενός client κι ενός server, αντιστοιχούν δύο flows, ένα προς κάθε κατεύθυνση, από τον client προς τον server κι από τον server προς τον client. Ένα flow χαρακτηρίζεται από εφτά πεδία-κλειδιά, τα οποία είναι τα εξής : source IP address, destination IP address, source port number, destination port number, protocol type, type of services, router input interface. Κάθε στιγμή που ο δρομολογητής λαμβάνει ένα πακέτο, εξετάζει αυτά τα εφτά πεδία και λαμβάνει μία αντίστοιχη απόφαση : εάν το πακέτο ανήκει σε ένα υπάρχον flow, τα στατιστικά στοιχεία που αφορούν το συγκεκριμένο flow θα μεταβληθούν ανάλογα, διαφορετικά, ένα νέο flow θα δημιουργηθεί.

Καθώς η διαδικασία αυτή δημιουργίας νέων flows συνεχίζεται, τα προηγούμενα flows που αποτελούν μία Netflow εγγραφή (Netflow records) αποστέλλονται μέσω UDP πακέτων σε έναν καθορισμένο από το χρήστη σταθμό λήψης, στην περίπτωση που ικανοποιηθεί μία από τις παρακάτω πιθανές συνθήκες :

- Το πρωτόκολλο μεταφοράς υποδεικνύει την ολοκλήρωση της σύνδεσης, συνήθως μέσω ενός πακέτου TCP FIN ή RST, ενώ ακολουθεί μία μικρή καθυστέρηση για την ολοκλήρωση της τριπλής χειραψίας τερματισμού της σύνδεσης.
- Κίνηση που αφορά το συγκεκριμένο flow δεν έχει εμφανιστεί για 15 δευτερόλεπτα.
- Για flows που παραμένουν συνεχώς ενεργά, τα αντίστοιχα flow records αποστέλλονται κάθε 30 λεπτά, επιβεβαιώνοντας μία περιοδική αναφορά των στατιστικών στοιχείων που αφορούν τα ενεργά flows.

Κάθε Netflow record περιλαμβάνει πεδία με στοιχεία που αφορούν τα πακέτα που συνθέτουν το κάθε flow, πέρα από τα εφτά στάνταρ πεδία που αναφέραμε. Αυτά τα πεδία μπορεί να περιέχουν πληροφορίες για τους χρόνους έναρξης και ολοκλήρωσης του κάθε flow (start and end time), τον αριθμό των πακέτων και των bytes που συνθέτουν το flow, τους αριθμούς των Αυτόνομων Συστημάτων πηγής και προορισμού (source and destination AS numbers), τις IP διευθύνσεις πηγής και προορισμού, τους αριθμούς των interfaces εισόδου και εξόδου, τον τύπο πρωτοκόλλου και διάφορα TCP flags (ή λογικά OR των TCP flags). Στην περίπτωση ειδικά του πρωτοκόλλου ICMP (Internet Control Message Protocol), ο τύπος των ICMP μηνυμάτων καταγράφεται επίσης στο πεδίο του destination port των Netflow records.

Παρατηρώντας τα flow records, διαπιστώνουμε ότι δεν υπάρχει πληροφορία ωφέλιμου φορτίου (payload) ανάμεσα στα πεδία του flow. Αυτή είναι και μία βασική διαφορά που εντοπίζουμε συγκρίνοντας το Netflow με τα υπόλοιπα παραδοσιακά Συστήματα Εντοπισμού Επιθέσεων (IDS). Ένα flow record δεν περιλαμβάνει κάποια πληροφορία υψηλού επιπέδου, παρά μόνο στοιχεία που αφορούν την κίνηση του συστήματος (traffic profiles). Κατά συνέπεια, αυτό αφαιρεί από το Netflow τη δυνατότητα να διεισδύσει στα διάφορα πακέτα σε βάθος αναλύοντας το περιεχόμενό

τους, ωστόσο μπορεί να αντλήσει αρκετές άλλες πληροφορίες που οδηγούν σε χρήσιμα συμπεράσματα. Το βασικό πλεονέκτημα της προσέγγισης μέσω Netflow δεδομένων είναι η υψηλή ταχύτητα. Καθώς η παρατήρηση του περιεχομένου των πακέτων παραλείπεται, μειώνεται αυτόματα ο γενικός χρόνος επεξεργασίας, γεγονός το οποίο καθιστά το Netflow ένα εξαιρετικό εργαλείο για υπερφορτωμένα και μεγάλης ταχύτητας δικτυακά περιβάλλοντα. Επιπλέον, το χαρακτηριστικό αυτό αποδεικνύεται πολύ χρήσιμο σε περιπτώσεις εντοπισμού "mutant attacks", όπου τα υπόλοιπα 'signature-based' Συστήματα Εντοπισμού Επιθέσεων θα αποτύγχαναν.

Καθώς τα flow records προέρχονται απευθείας από κάποιον δρομολογητή του συστήματος που αποτελεί ένα βασικό στοιχείο ενός δικτύου, το Netflow δίνει τη δυνατότητα παραγωγής μιας ευρύτερης εικόνας που αφορά το σύνολο της δικτυακής κίνησης και έτσι προλαμβάνει τον εντοπισμό γεγονότων που αφορούν την ασφάλεια του δικτύου.

Ο σταθμός συλλογής των Netflow records θα πρέπει να διαθέτει κάποιον ανάλογο μηχανισμό ανάλυσης και επεξεργασίας των στοιχείων που αντιστοιχούν στα flows που συγκεντρώθηκαν για τη μελέτη τους σε πραγματικό χρόνο (real time). Ο μηχανισμός αυτός μπορεί να είναι είτε ένα ειδικό software ή hardware του εμπορίου, είτε ένας σταθμός με κατάλληλα εργαλεία ανάλυσης και επεξεργασίας (open source tools).

Εάν αναλυθούν κατάλληλα, τα Netflow records μπορεί να αποδειχθούν πολύ χρήσιμα στον έγκαιρο εντοπισμό worms ή οποιασδήποτε άλλης δυσλειτουργίας του δικτύου και των παροχέων υπηρεσίας (service providers). Με βάση τα Netflow δεδομένα, μπορούν να αναπτυχθούν διάφορες μέθοδοι μελέτης και ανάλυσης δικτυακής κίνησης. Μία από αυτές είναι και η μέθοδος που βασίζεται στην αντιστοίχιση με συγκεκριμένα πρότυπα, το λεγόμενο 'pattern matching'.

- Pattern Matching

  Πρόκειται για μία από τις μεθόδους που χρησιμοποιούνται για την αναγνώριση δικτυακών δυσλειτουργιών και βασίζονται στην ανάλυση δεδομένων που προέρχονται από flow records. Σύμφωνα με τη μέθοδο αυτή, τα flows που συγκεντρώνονται, ελέγχονται και εντοπίζονται αυτά που αφορούν κάποιους κόμβους, οι οποίοι θεωρούνται 'ύποπτοι' με βάση συγκεκριμένα κριτήρια. Τα

κριτήρια αυτά μπορεί να σχετίζονται με οποιοδήποτε από τα πεδία που περιέχονται σε ένα flow, αλλά εκείνα που ενδιαφέρουν συνήθως είναι οι IP διευθύνσεις πηγής και προορισμού πακέτων (source and destination IP addresses) ή οι πόρτες πηγής και προορισμού επίσης (source and destination port numbers).

- Port matcing

  Κατά κανόνα, οι περισσότερες επιθέσεις στοχεύουν σε μία συγκεκριμένη πόρτα του συστήματος. Για παράδειγμα, το SQL Slammer worm λειτουργεί στην πόρτα 1434, το Netbus Trojan στην πόρτα 12345 κλπ. Οι διαχειριστές των δικτύων μπορούν να φιλτράρουν τα flow records που έχουν συγκεκριμένο destination port, προκειμένου να εντοπίσουν ανάλογες επιθέσεις. Αυτή η μέθοδος είναι πολύ απλή στην εφαρμογή της και μπορεί να χρησιμοποιηθεί στις περισσότερες περιπτώσεις, παρ'ότι μπορεί να παράγει μερικές φορές λανθασμένα συμπεράσματα.

- IP address matching

  Για μία εταιρία ή έναν πάροχο ISP που εφαρμόζει μία flow-based μέθοδο εντοπισμού δυσλειτουργιών, υπάρχουν συνήθως συγκεκριμένοι κανόνες που ακολουθούνται και μπορεί να βασίζονται για παράδειγμα:

  - στην εξερχόμενη κίνηση

    Για μία εταιρία ή έναν πάροχο ISP, οποιοδήποτε flow record του οποίου η IP source address δεν αποτελεί κομμάτι της επιτρεπόμενης εξερχόμενης κίνησης της περιοχής του δικτύου του, θα πρέπει να θεωρηθεί ως μη αποδεκτό.

  - στην εισερχόμενη κίνηση

    Το ίδιο συμβαίνει και στην περίπτωση όπου η IP source address οποιουδήποτε flow record δεν αποτελεί κομμάτι της επιτρεπόμενης εισερχόμενης κίνησης της περιοχής του δικτύου, οπότε θεωρείται και πάλι μη αποδεκτό.

- σε καθορισμένο σύνολο IP διευθύνσεων

  Κάποια είδη δυσλειτουργιών μπορεί να σχετίζονται με μία ή περισσότερες καθορισμένες IP διευθύνσεις. Για παράδειγμα, όταν το W32/Netsky.c worm εξαπλώνεται, στέλνει μία σειρά DNS πακέτων στους ακόλουθους DNS servers :

  145.253.2.171, 151.189.13.35, 193.141.40.42, 193.189.244.205, 193.193.144.12, 193.193.158.10, 194.25.2.129, 194.25.2.129, 194.25.2.130, 194.25.2.131, 194.25.2.132, 194.25.2.133, 194.25.2.134, 195.185.185.195, 195.20.224.234, 212.185.252.136, 212.185.252.73, 212.185.253.70, 212.44.160.8, 212.7.128.162, 212.7.128.165, 213.191.74.19, 217.5.97.137, 62.155.255.16

  Κατά συνέπεια, οποιοδήποτε flow record του οποίου η destination address αντιστοιχεί σε μία από τις παραπάνω διευθύνσεις και επιπλέον η destination port τυγχάνει να είναι η 53, θα ενεργοποιούσε κάποιο συναγερμό και ένας περαιτέρω έλεγχος θα ήταν απαραίτητος.

#### 5.2.2.2 Τα OSU Flow Tools

Το Πανεπιστήμιο του Ohio State έχει συντάξει μία σειρά από εργαλεία για τη συλλογή, το φιλτράρισμα, την ανάλυση και την εκτύπωση Netflow δεδομένων. Τα εργαλεία αυτά, γνωστά ως **flow-tools**, μπορούν να δοθούν ως pipelined commands σε περιβάλλον UNIX, καθιστώντας έτσι εύκολη την επεξεργασία των δεδομένων χωρίς την ανάγκη δημιουργίας ενδιάμεσων αρχείων. Τα flow-tools διακρίνονται σε εργαλεία συλλογής flows (**capture tools**), γενικής ανάλυσης δεδομένων (**general analysis tools**) και εργαλεία ασφάλειας (**security tools**).

Σκοπός των 'capture tools' είναι η συλλογή των flows που προέρχονται από τα Netflow records. Όπως αναφέραμε και προηγουμένως, για κάθε Netflow router, πρέπει να έχει καθοριστεί ένας συγκεκριμένος σταθμός συλλογής, στον οποίο θα αποστέλλονται τα flows που συγκεντρώνονται. Η διαδικασία αυτή γίνεται με τη βοήθεια UDP πακέτων που αποστέλλονται σε καθορισμένο σταθμό και σε συγκεκριμένο port, όπου υπάρχει κάποιο κατάλληλο πρόγραμμα το οποίο αναλαμβάνει τη συλλογή και την επεξεργασία των δεδομένων. Για παράδειγμα, η εντολή flow-export destination 10.0.0.1 12345 θα προκαλούσε την εξαγωγή Netflow records στον host με IP διεύθυνση 10.0.0.1 και στο UDP port 12345.

Ένα από τα βασικά capture tools είναι το **flow-capture**, το οποίο ακούει σε ένα καθορισμένο UDP port και γράφει τα δεδομένα που συγκεντρώνει σε κατάλληλα αρχεία (log files). Προκειμένου δε να αποφευχθεί η υπερχείλιση λόγω του μεγέθους των δεδομένων, το εργαλείο γράφει περιοδικά σε ένα νέο αρχείο κάθε φορά.

Το εργαλείο **flow-receive** κάνει κάτι αντίστοιχο με το flow-capture, με τη διαφορά ότι εμφανίζει τα αποτελέσματα στο stdout αντί να τα καταχωρεί σε συγκεκριμένα αρχεία. Αυτό το εργαλείο θα χρησιμοποιήσουμε κι εμείς στη συνέχεια για τη συλλογή των δεδομένων μας.

Ένα ακόμη χρήσιμο εργαλείο, το οποίο ανήκει στα 'analysis flow-tools' είναι το **flow-filter**, το οποίο παρέχει τη δυνατότητα συλλογής και επεξεργασίας συγκεκριμένων flow records, κάποια από τα πεδία των οποίων αντιστοιχούν σε συγκεκριμένες τιμές που μπορεί να αφορούν για παράδειγμα ένα από τα παρακάτω :

- τον αριθμό του source ή destination port (-p και –P options)
- τον τύπο του IP πρωτοκόλλου (-r option)
- τον αριθμό του input ή output interface (-i και -I options)
- τις source ή destination IP addresses που περιέχονται σε συγκεκριμένες λίστες, για παράδειγμα Access Control Lists της Cisco (ACLs, -S και –D options)
- τον αριθμό του source ή destination Αυτόνομου Συστήματος (AS number –a και –A options)

Οι περισσότερες από τις επιλογές αυτές δίνουν τη δυνατότητα να καθοριστούν συγκεκριμένες λίστες με κατάλληλες τιμές για αντιστοίχιση.

Για παράδειγμα, η εντολή `flow-filter –f flow.acl –S attackers –D victims`
θα διάβαζε τις λίστες που θα περιείχε το αρχείο flow.acl και θα εντόπιζε τα flow records των οποίων η τιμή του πεδίου της source IP address θα ταίριαζε με κάποιο από τα στοιχεία της ACL "attackers" και αντίστοιχα η τιμή του πεδίου της destination IP address θα ταίριαζε με κάποιο από τα στοιχεία της ACL "victims".

Στα analysis flow tools ανήκουν επίσης δύο ακόμη εργαλεία, το **flow-cat** και το **flow-print**. Το πρώτο από αυτά διαβάζει μια σειρά από flow logs είτε από την οθόνη (stdin) είτε από κάποιο αρχείο, και με τη σειρά του τα αποτυπώνει και πάλι στην οθόνη ή σε κάποιο άλλο καθορισμένο αρχείο με τη μορφή binary records που δεν είναι όμως άμεσα αναγνώσιμη. Εδώ λοιπόν βρίσκει εφαρμογή το flow-print, το οποίο

διαβάζει τα binary records από ένα flow log και τυπώνει το περιεχόμενό του σε μία από τι αρκετές υπάρχουσες, αναγνώσιμες μορφές απεικόνισης. Για παράδειγμα, η εντολή flow-cat *|flow-print –f 5 θα τύπωνε τις αντίστοιχες πληροφορίες για όλα τα flow logs που ανήκουν στο τρέχον αρχείο, χρησιμοποιώντας απεικόνιση τύπου 5. Η μορφή αυτή περιλαμβάνει τα παρακάτω πεδία για κάθε flow : start and end time, source and destination IP address, TCP or UDP ports, IP protocol type, source and destination interface numbers, TCP flags, και έναν μετρητή των bytes και των packets του κάθε flow.

Κάνοντας χρήση των παραπάνω εργαλείων, μπορούμε να εφαρμόσουμε κι εμείς κάτι ανάλογο προκειμένου να συγκεντρώσουμε στατιστικά στοιχεία για τα flows που αφορούν το σύνολο του 'non-allocated address space', το οποίο μας ενδιαφέρει να μελετήσουμε. Η λίστα των διευθύνσεων που αντιπροσωπεύουν το σύνολο αυτό, βρίσκεται ήδη καταχωρημένη στο αρχείο `'filter.def'`, συνθέτοντας το φίλτρο με την ονομασία `'telescope'`, του οποίου οι κανόνες ορίζουν ότι επιτρέπεται μόνο η κίνηση με προορισμό (destination IP) ή πηγή (source IP), μία από τις διευθύνσεις του συνόλου των unallocated addresses.

Εκτελώντας στη συνέχεια για παράδειγμα την παρακάτω εντολή, σε μορφή pipelined command :

```
> flow-cat < netflow-sample | flow-send 0/147.102.13.28/6666 | flow-receive -f
  filter.def -F telescope 147.102.13.28/0/6666 | flow-print
```

παίρνουμε στην οθόνη αποτελέσματα της παρακάτω μορφής, τα οποία περιέχουν πληροφορίες μόνο για τα flows των οποίων το source ή destination IP ανήκει στη λίστα που ορίζεται από το φίλτρο 'telescope'.

```
srcIP                  dstIP                  router_sc          prot    srcPort     dstPort
octets      packets
flow-receive:   New   exporter:   time=1128044188   src_ip=147.102.13.28   dst_ip=147.102.13.28
d_version=7
147.102.28.116/24     147.102.171.55/16     147.102.220.200   6         2754                139
96          2
147.102.28.116/24     147.102.203.185/16   147.102.220.200   6         2559                139
48          1
147.102.23.13/24      147.102.227/16         147.102.220.200   6         1527                445
96          2
147.102.192.220/24   147.102.94.237/16     147.102.220.200   6         1923                445
96          2
```

```
147.102.28.116/24     147.102.75.211/16     147.102.220.200   6         2557                 80
48          1
147.102.28.116/24     147.102.242.102/16    147.102.220.200   6         2693                 139
96          2
147.102.163.23/24     147.102.113.185/16    147.102.220.200   6         3300                 445
96          2
147.102.23.13/24      147.102.17.130/16     147.102.220.200   6         1486                 445
96          2
147.102.28.116/24     147.102.198.24/16     147.102.220.200   6         2656                 139
96          2
147.102.28.116/24     147.102.142.103/16    147.102.220.200   6         2691                 80
96          2
147.102.163.23/24     147.102.120.164/16    147.102.220.200   6         3361                 445
96          2
```

Ωστόσο, σκοπός μας είναι να δημιουργήσουμε κάποια άλλα πιο εύχρηστα εργαλεία, τα οποία θα μπορούν να μας δώσουν πιο πολλές και περισσότερο ομαδοποιημένες πληροφορίες για τα flows των IPs που μας ενδιαφέρουν. Για το λόγο αυτό, θα χρησιμοποιήσουμε μία παραλλαγή του εργαλείου flow-receive. Είναι το λεγόμενο flow-rrd-receive, το οποίο παρέχει ομαδοποιημένα στατιστικά αποτελέσματα και γραφικές παραστάσεις αυτών, κάνοντας χρήση του εργαλείου RRDTool.

### 5.2.3 Αποθήκευση και παρουσίαση των αποτελεσμάτων

#### 5.2.3.1 Εισαγωγή στην Τεχνολογία του RRDTool

Είναι σίγουρα πολύ χρήσιμο για κάποιον να μπορεί να συγκεντρώνει πληροφορίες και στατιστικά στοιχεία κάθε είδους, από τη θερμοκρασία ενός δωματίου, μέχρι τον αριθμό των bits που πέρασαν από ένα συγκεκριμένο interface ενός router. Ωστόσο, είναι αρκετά πολύπλοκο να καταχωρηθούν και να αναπαρασταθούν αυτά τα δεδομένα σύμφωνα με έναν συστηματικό και αποτελεσματικό τρόπο. Στο σημείο αυτό είναι που βρίσκει εφαρμογή το RRDtool, το οποίο επιτρέπει τη συλλογή, την ανάλυση και την αναπαράσταση στοιχείων, που μπορεί να προέρχονται από κάθε είδους πηγές δεδομένων (data sources). Το κομμάτι του RRDtool που αφορά την ανάλυση δεδομένων, βασίζεται στη δυνατότητα της γρήγορης παραγωγής γραφικών παραστάσεων για τις τιμές που συγκεντρώθηκαν σε μία καθορισμένη χρονική περίοδο.

Τι είναι το RRDtool ;

Η ονομασία RRD προέρχεται από τα αρχικά για το Round Robin Database. Το RRD είναι ένα σύστημα για την αποθήκευση και την αναπαράσταση δεδομένων κατά τη διάρκεια μιας χρονικής περιόδου. Τα δεδομένα αυτά αποθηκεύονται σε συμπιεσμένη μορφή χωρίς να υπόκεινται χρονική υπερχείλιση και με βάση αυτά μπορούν να παραχθούν ανάλογες γραφικές παραστάσεις.

Το RRDtool είναι ένα εργαλείο το οποίο βασίζεται στην τεχνική του Round Robin. Η τεχνική αυτή λειτουργεί με βάση ένα καθορισμένο σύνολο δεδομένων και έναν δείκτη που δείχνει κάθε φορά στο τρέχον αντικείμενο. Θα μπορούσαμε να φανταστούμε έναν κύκλο με ένα σύνολο από σημεία κατά μήκος της περιφέρειάς του (τα οποία αντιστοιχούν στους χώρους όπου αποθηκεύονται τα δεδομένα). Υποθέτουμε επίσης πως υπάρχει ένα βέλος - δείκτης με αφετηρία το κέντρο του κύκλου, το οποίο δείχνει σε ένα από τα σημεία αυτά. Όταν το συγκεκριμένο δεδομένο προσπελαστεί, ο δείκτης μετακινείται στο επόμενο αντικείμενο. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς καθώς βρισκόμαστε πάνω σε έναν κύκλο χωρίς αρχή και τέλος. Ύστερα από κάποιο χρονικό διάστημα, όλα τα διαθέσιμα σημεία θα έχουν χρησιμοποιηθεί και αυτόματα η διαδικασία επαναχρησιμοποιεί τα προηγούμενα ίδια σημεία. Κατ'αυτόν τον τρόπο, το σύνολο των δεδομένων δε μεγαλώνει σε μέγεθος καθώς χρησιμοποιούνται συνεχώς οι ίδιοι χώροι αποθήκευσης. Το RRDtool δουλεύει με αντίστοιχες βάσεις δεδομένων, τις Round Robin Databases (RRDs), στις οποίες έχει τη δυνατότητα καταχώρησης και προσπέλασης δεδομένων. Τι ακριβώς όμως μπορεί να κάνει ένας χρήστης με τη βοήθεια του εργαλείου αυτού;

Το RRDtool προήλθε από το εργαλείο MRTG (Multi Router Traffic Grapher). Το MRTG ξεκίνησε σαν ένα μικρό script για τη γραφική αναπαράσταση της χρήσης του Internet από ένα πανεπιστήμιο. Στη συνέχεια επεκτάθηκε και στην απεικόνιση άλλων δεδομένων, μεταξύ των οποίων η θερμοκρασία, η ταχύτητα, η τάση κ.α. Συνήθως το RRDtool χρησιμοποιείται για την καταγραφή και την επεξεργασία δεδομένων τα οποία συλλέγονται με χρήση του πρωτοκόλλου SNMP. Τα δεδομένα αυτά είναι συνήθως bytes που μεταφέρονται προς ή από ένα δίκτυο ή έναν υπολογιστή. Αλλά μπορεί να παριστάνουν επίσης και επίπεδα τηλεπικοινωνιακού θορύβου, κατανάλωση ισχύος, ηλιακή ακτινοβολία, θερμοκρασία ενός τόπου και πολλά άλλα.

Το μόνο που χρειάζεται είναι ένας κατάλληλος αισθητήρας για τη μέτρηση αυτών των δεδομένων και την τροφοδοσία τους στο RRDtool για περαιτέρω επεξεργασία.

Το RRDtool επιτρέπει στη συνέχεια τη δημιουργία βάσεων δεδομένων (databases), την καταχώρηση δεδομένων στις βάσεις αυτές, τη μετέπειτα εξαγωγή τους και τη δημιουργία γραφικών παραστάσεων σε μορφή τύπου PNG για απεικόνιση από έναν web browser. Αυτές οι PNG απεικονίσεις εξαρτώνται από τα συλλεγόμενα δεδομένα, τα οποία μπορεί να αντιπροσωπεύουν για παράδειγμα τη γενική εικόνα κίνησης ενός δικτύου. Τι ακριβώς όμως είναι αυτό που κάνει το RRDtool τόσο ξεχωριστό;

Παρά το γεγονός ότι το RRDtool στηρίζεται σε βάσεις δεδομένων (databases), υπάρχουν κάποιες διακριτές διαφορές μεταξύ αυτού και άλλων βάσεων δεδομένων :

- Το RRDtool έχει τη δυνατότητα αποθήκευσης δεδομένων στις databases αλλά και τη δημιουργία γραφικών παραστάσεων με βάση τα δεδομένα αυτά, σε αντίθεση με άλλες databases που απλά αποθηκεύουν δεδομένα.
- Στην περίπτωση γραμμικών databases, τα νέα δεδομένα εισάγονται στο τέλος του αντίστοιχου πίνακα της βάσης. Έτσι το μέγεθός της αυξάνεται διαρκώς, σε αντίθεση με μία RRDtool database, της οποίας το μέγεθος είναι καθορισμένο εξαρχής και μπορεί να θεωρηθεί σαν την περίμετρο ενός κύκλου όπως αναφέραμε παραπάνω, όπου τα νέα δεδομένα εισάγονται πάνω στην περίμετρο αυτή. Όταν συμπληρωθούν οι θέσεις αποθήκευσης δεδομένων της περιμέτρου, τα νέα δεδομένα επικαλύπτουν τα προηγούμενα. Με αυτόν τον τρόπο, το μέγεθος της RRDtool database παραμένει πάντα σταθερό. Από την τεχνική αυτή προέρχεται και το όνομα “Round Robin”.
- Άλλες databases αποθηκεύουν τις τιμές των δεδομένων όπως ακριβώς τις λαμβάνουν. Το RRDtool έχει τη δυνατότητα υπολογισμού του επιπέδου διαφοράς της κάθε τιμής από την προηγούμενή της και στη συνέχεια μπορεί να αποθηκεύσει την πληροφορία αυτή.
- Άλλες databases ανανεώνονται κάθε φορά που τροφοδοτούνται με νέες τιμές. Μία RRDtool database είναι δομημένη με τέτοιο τρόπο, ώστε να απαιτεί την εισαγωγή δεδομένων σε καθορισμένες χρονικές στιγμές. Εάν δεν υπάρχουν νέα

δεδομένα για εισαγωγή, καταχωρεί μία άγνωστη (unknown) τιμή για τη στιγμή αυτή. Συνεπώς, όταν χρησιμοποιούμε το RRDtool, είναι υποχρεωτικό να διαθέτουμε scripts τα οποία τρέχουν σε καθορισμένα χρονικά διαστήματα, ώστε να εξασφαλίσουμε μία σταθερή ροή δεδομένων που ανανεώνει την RRDtool database.

Η δομή μιας RRD database είναι διαφορετική από άλλες γραμμικές databases, οι οποίες περιέχουν πίνακες με στήλες και πολλές άλλες παραμέτρους. Αυτές οι παράμετροι είναι μερικές φορές αρκετά πολύπλοκες, ειδικά όταν πρόκειται για μεγάλου μεγέθους databases. Οι RRDtool databases χρησιμοποιούνται κυρίως για λειτουργίες διαχείρισης και για το λόγο αυτό η δομή τους είναι απλή. Οι παράμετροι που χρειάζεται να καθοριστούν είναι μεταβλητές που παίρνουν διάφορες τιμές και αρχεία με τις τιμές αυτές. Καθώς ο χρόνος αποτελεί σημαντικό στοιχείο, υπάρχουν επίσης μερικές μεταβλητές σχετικές με το χρόνο που καθορίζονται επίσης. Λόγω της δομής της, ο ορισμός μιας RRDtool database περιλαμβάνει επιπλέον και κάποιες προβλέψεις που καθορίζουν συγκεκριμένες ενέργειες στην περίπτωση απουσίας νέων τιμών των μεταβλητών κατά την ανανέωση της database. Data Source (**DS**), heartbeat, Date Source Type (**DST**), Round Robin Archive (**RRA**), και Consolidation Function (**CF**) είναι κάποιες από τις ορολογίες που σχετίζονται με τις RRDtool databases.

Για την εκτέλεση των διαφόρων λειτουργιών, το RRDtool χρησιμοποιεί ένα πλήθος συναρτήσεων. Αναφέρουμε τις πιο χαρακτηριστικές από αυτές :

Δημιουργώντας μία νέα βάση δεδομένων – RRDtool database

- **rrdtool create** : με την εκτέλεση της συνάρτησης αυτής δημιουργείται μία νέα RRDtool database. Το επόμενο παράδειγμα δίνει μία εικόνα του τρόπου εκτέλεσής της :

```
rrdtool create target.rrd \
       --start 1023654125 \
       --step 300 \
       DS:mem:GAUGE:600:0:671744 \
       RRA:AVERAGE:0.5:12:24 \
       RRA:AVERAGE:0.5:288:31
```

Σε αυτό το παράδειγμα δημιουργείται μία νέα database με την ονομασία *target.rrd*. Ο χρόνος έναρξης (1'023'654'125) καθορίζεται με βάση το συνολικό αριθμό seconds since epoch (time in seconds since 01-01-1970). Ορίζεται επίσης ο χρόνος ανανέωσης των δεδομένων της βάσης (update time), ο οποίος πρέπει να είναι μεγαλύτερος από το χρόνο έναρξης και να εκφράζεται επίσης σε seconds since epoch. Το βήμα των 300 δευτερολέπτων (step 300) ορίζει ότι η database περιμένει νέες τιμές κάθε 300 δευτερόλεπτα, δηλαδή το script με τον αντίστοιχο κώδικα θα πρέπει να τρέχει κάθε **step** δευτερόλεπτα, έτσι ώστε η database να ανανεώνεται σε κάθε περίοδο των **step** δευτερολέπτων.

Data Acquisition

Ο όρος DS (Data Source) αναφέρεται στη μεταβλητή που σχετίζεται με την διαχειριζόμενη παράμετρο του συστήματος. Η σύνταξή της έχει ως εξής :

```
DS:variable_name:DST:heartbeat:min:max
```

Το **DS** είναι μια λέξη-κλειδί. Το `variable_name` αναφέρεται στο όνομα της μεταβλητής στην οποία αποθηκεύεται η διαχειριζόμενη παράμετρος μέσα στην database. Είναι δυνατό να υπάρχουν τόσες DSs μέσα σε μία database όσες ακριβώς χρειάζονται. Μετά από κάθε χρονικό διάστημα που ορίζεται από το step, η DS αποκτά μία νέα τιμή προκείμένου να γίνει η ανανέωση της database. Αυτή η τιμή ονομάζεται επίσης Primary Data Point **(PDP)**. Στο παράδειγμά μας, μία νέα PDP παράγεται κάθε 300 δευτερόλεπτα. Ωστόσο, το RRDtool μπορεί να διαχειρίζεται σωστά τα νέα δεδομένα, ακόμα και αν αυτά δεν παράγονται ακριβώς κάθε 300 δευτερόλεπτα.

Το **DST** (Data Source Type) καθορίζει τον τύπο του DS, ο οποίος μπορεί να είναι COUNTER, DERIVE, ABSOLUTE, GAUGE. Στο παρακάτω σχήμα διακρίνεται η διαφορά μεταξύ των τύπων αυτών :

```
Values       = 300, 600, 900, 1200
Step         = 300 seconds
COUNTER DS   =    1,  1,   1,    1
DERIVE DS    =    1,  1,   1,    1
ABSOLUTE DS  =    1,  2,   3,    4
GAUGE DS     = 300, 600, 900, 1200
```

Η επόμενη παράμετρος είναι το **heartbeat**. Στο παράδειγμά μας, η τιμή του είναι 600 δευτερόλεπτα. Εάν η database δεν πάρει κάποιο νέο PDP μέσα στα 300 δευτερόλεπτα για μία μεταβλητή, θα περιμένει για ακόμη 300 δευτερόλεπτα, σύνολο 600 δευτερόλεπτα. Εάν και πάλι δεν λάβει κάποιο νέο PDP, θα καταχωρήσει μία UNKNOWN τιμή στη μεταβλητή. Αυτή η UNKNOWN τιμή είναι ένα ιδιαίτερο χαρακτηριστικό του RRDtool και είναι προτιμότερη από κάποια άλλη τιμή, για παράδειγμα το 0, που θα μπορούσε να είναι επίσης κάποιο έγκυρο δεδομένο.

Οι επόμενες δύο παράμετροι (**min, max**) είναι η ελάχιστη και η μέγιστη τιμή αντίστοιχα. Εάν η μεταβλητή που πρόκειται να καταχωρηθεί έχει κάποια προβλέψιμη μέγιστη και ελάχιστη τιμή, αυτές θα πρέπει να καθοριστούν. Στη συνέχεια, οποιαδήποτε τιμή που θα ληφθεί και θα βρίσκεται έξω από το εύρος αυτό, θα καταχωρείται σαν UNKNOWN τιμή.

Consolidation – Round Robin Archives (RRAs)

Η επόμενη γραμμή στο παράδειγμά μας ορίζει ένα round robin αρχείο (round robin archive - RRA). Η σύνταξη για τον ορισμό ενός RRA έχει ως εξής :

```
RRA:CF:xff:step:rows
```

Το **RRA** είναι η λέξη-κλειδί για τη δήλωση RRAs. Η συνάρτηση συγχώνευσης (consolidation function - CF) μπορεί να είναι `AVERAGE`, `MINIMUM`, `MAXIMUM`, and `LAST`. Η έννοια του σημείου συγχώνευσης (consolidated data point - CDP) εμφανίζεται στο σημείο αυτό. Ένα CDP προκύπτει από το μέσο όρο (average), τη μέγιστη/ελάχιστη (min/max) τιμή ή την τελευταία τιμή από ένα σύνολο *`step`* PDPs. Ένα RRA κρατάει σειρές (rows) από CDPs.

Αναφερόμενοι και πάλι στο παραπάνω παράδειγμα, παρατηρούμε ότι για το πρώτο RRA, 12 (steps) PDPs (DS variables) είναι AVERAGed (CF)για να σχηματίσουν ένα CDP. 24 σειρές (rows) από αυτά τα CDPs καταχωρούνται στο αρχείο. Ένα νέο PDP προκύπτει κάθε 300 δευτερόλεπτα. 12 PDPs αντιστοιχούν σε 12 επαναλήψεις 300 δευτερολέπτων, δηλαδή σε 1 ώρα. Αυτό σημαίνει ότι 1 CDP που αντιστοιχεί σε 12 PDPs, αντιπροσωπεύει δεδομένα διάρκειας 1 ώρας, δηλαδή 24 τέτοια CDPs αντιπροσωπεύουν 1 μέρα. Συνεπώς, το RRA αυτό είναι ένα αρχείο με δεδομένα μίας ημέρας. Ύστερα από 24 CDPs, το CDP νούμερο 25 θα αντικαταστήσει το πρώτο

CDP. Το δεύτερο RRA αποθηκεύει 31 CDPs, εκ των οποίων καθένα αντιπροσωπεύει μία μέση τιμή για δεδομένα μίας ημέρας (288 PDPs, each covering 300 seconds = 24 hours). Συνεπώς, αυτό το RRA είναι ένα αρχείο με δεδομένα ενός μήνα.
Μία database μπορεί να περιέχει πολλά RRAs. Εάν υπάρχουν και πολλά DSs, κάθε ξεχωριστό RRA αποθηκεύει δεδομένα για όλα τα DSs της database. Για παράδειγμα, εάν μία database έχει 3 DSs και ορίζονται ημερήσια, εβδομαδιαία, μηνιαία και ετήσια RRAs, τότε το κάθε RRA θα κρατάει δεδομένα και από τις 3 πηγές δεδομένων (DSs).

Προσθέτοντας νέα δεδομένα σε μια RRDtool database

Έχοντας δημιουργήσει τη βάσή δεδομένων RRDtool database όπου καταχωρούνται τα δεδομένα μας, στη συνέχεια μας ενδιαφέρει να ανανεώνουμε την database, εισάγοντας νέα δεδομένα ανά τακτά χρονικά διαστήματα. Τη λειτουργία αυτή του update υλοποιεί η επόμενη συνάρτηση :
- **rrdtool update** : χρησιμοποιείται για την καταχώρηση νέων δεδομένων σε μία RRDtool database. Για παράδειγμα, με την εκτέλεση της εντολής

```
$ rrdtool update target.rrd time_t:value1:value2:…
```

ανανεώνονται τα δεδομένα της database που αντιστοιχεί στο αρχείο *target.rrd*.
Οι τιμές θα πρέπει πάντα να δίνονται με τη χρονολογική σειρά δημιουργίας τους.
Το `time_t` είναι το standard unix timestamp, ο αριθμός δηλαδή των δευτερολέπτων από την 1$^{η}$ Ιανουαρίου του 1970. Χρησιμοποιώντας το χαρακτήρα 'N' από το 'Now', μπορούμε να αναφερθούμε στην τρέχουσα χρονική στιγμή. Για παράδειγμα, το `$ rrdtool update yourfile.rrd N:12322` ανανεώνει την πρώτη μεταβλητή του rrd αρχείου κι αυτό γίνεται φανερό από τη στιγμή που έχουμε εισάγει μόνο μία.

Αναπαριστώντας τα δεδομένα μιας RRDtool database με γραφικές παραστάσεις

Έχοντας δημιουργήσει τη βάση δεδομένων μας και έχοντας ξεκινήσει να προσθέτουμε νέα δεδομένα σε αυτή, μπορούμε στη συνέχεια να εξάγουμε τα δεδομένα αυτά με τη μορφή γραφικών παραστάσεων. Το RRDtool, όπως και το MRTG έχει τη δυνατότητα παραγωγής γραφικών παραστάσεων σε μορφή GIF, PNG ή GD. Ωστόσο, η πιο εύχρηστη μορφή είναι η PNG (Portable Network Graphics). Η

αντίστοιχη συνάρτηση για τη δημιουργία γραφικών παραστάσεων είναι η rrdtool graph :

- **rrdtool graph** : παράγει γραφικές παραστάσεις με βάση δεδομένα που περιέχονται σε ένα ή περισσότερα rrds. Το επόμενο παράδειγμα είναι χαρακτηριστικό μιας τέτοιας λειτουργίας :

```
$ rrdtool graph graph.png DEF:pkt=datafile.rrd:packets:AVERAGE \
LINE1:pkt#ff0000:Packets
```

Εξορισμού, το RRDtool παράγει PNG αρχεία με δεδομένα για χρονική διάρκεια μίας ημέρας. Με την εκτέλεση της εντολής, καθορίζουμε συγκεκριμένες παραμέτρους, όπως :

- **DEF** : η παράμετρος αυτή καθορίζει ποια δεδομένα θα εξαχθούν από την RRDtool database. Στο παράδειγμά μας, το pkt είναι το όνομα της μεταβλητής που θα χρησιμοποιηθεί για την απεικόνιση των δεδομένων στη γραφική παράσταση. Το data.rrd:packets αναφέρεται στο rrd αρχείο-πηγή και στα δεδομένα με όνομα packets, το οποίο προσδιορίστηκε κατά τη δημιουργία του rrd αρχείου. Τέλος, το AVERAGE είναι το είδος της μεθόδου συγχώνευσης (consolidation method) που θα χρησιμοποιηθεί και μπορεί να επιλεγεί μεταξύ των MIN, MAX, AVERAGE, LAST.
  Αυτό που συμβαίνει λοιπόν είναι το εξής : αφού επιλεγεί μία μεταβλητή από ένα rrd αρχείο, της δίνεται ένα νέο όνομα για την απεικόνισή της στη γραφική παράσταση, με το οποίο αναφέρεται και στη συνέχεια. Αυτό αποδεικνύεται χρήσιμο στην περίπτωση που θέλουμε να αναφερθούμε σε δεδομένα από διαφορετικά rrds που μπορεί να έχουν ταυτόσημες ονομασίες μέσα στα αρχεία. Δίνοντάς τους λοιπόν διαφορετικά ονόματα, μπορούμε να επεξεργαστούμε το καθένα ξεχωριστά.
- **LINE** : η παράμετρος αυτή καθορίζει τον τύπο του γραφήματος που θα σχεδιαστεί. Μπορεί να επιλεγεί μεταξύ των LINE1, LINE2, LINE3, AREA,…Το 'pkt' είναι και πάλι το όνομα της μεταβλητής του γραφήματος που θα σχεδιαστεί (όπως καθορίστηκε και από την παράμετρο DEF), ενώ το '#ff0000' καθορίζει το χρώμα της γραμμής του γραφήματος (κόκκινο στην

προκειμένη περίπτωση). Το ‘Packets’ αναφέρεται στο όνομα που θα έχει η γραμμή στη λεζάντα.

#### 5.2.3.2 Γενική δομή και λειτουργία του ‘*flow-rrd-receive’*

Έχοντας υπόψη τα παραπάνω χαρακτηριστικά λειτουργίας του RRDTool, θα περιγράψουμε στη συνέχεια τη γενική δομή εκτέλεσης του προγράμματος flow-rrd-receive, καθώς και τις πληροφορίες που μπορούμε να πάρουμε με την εκτέλεσή του.

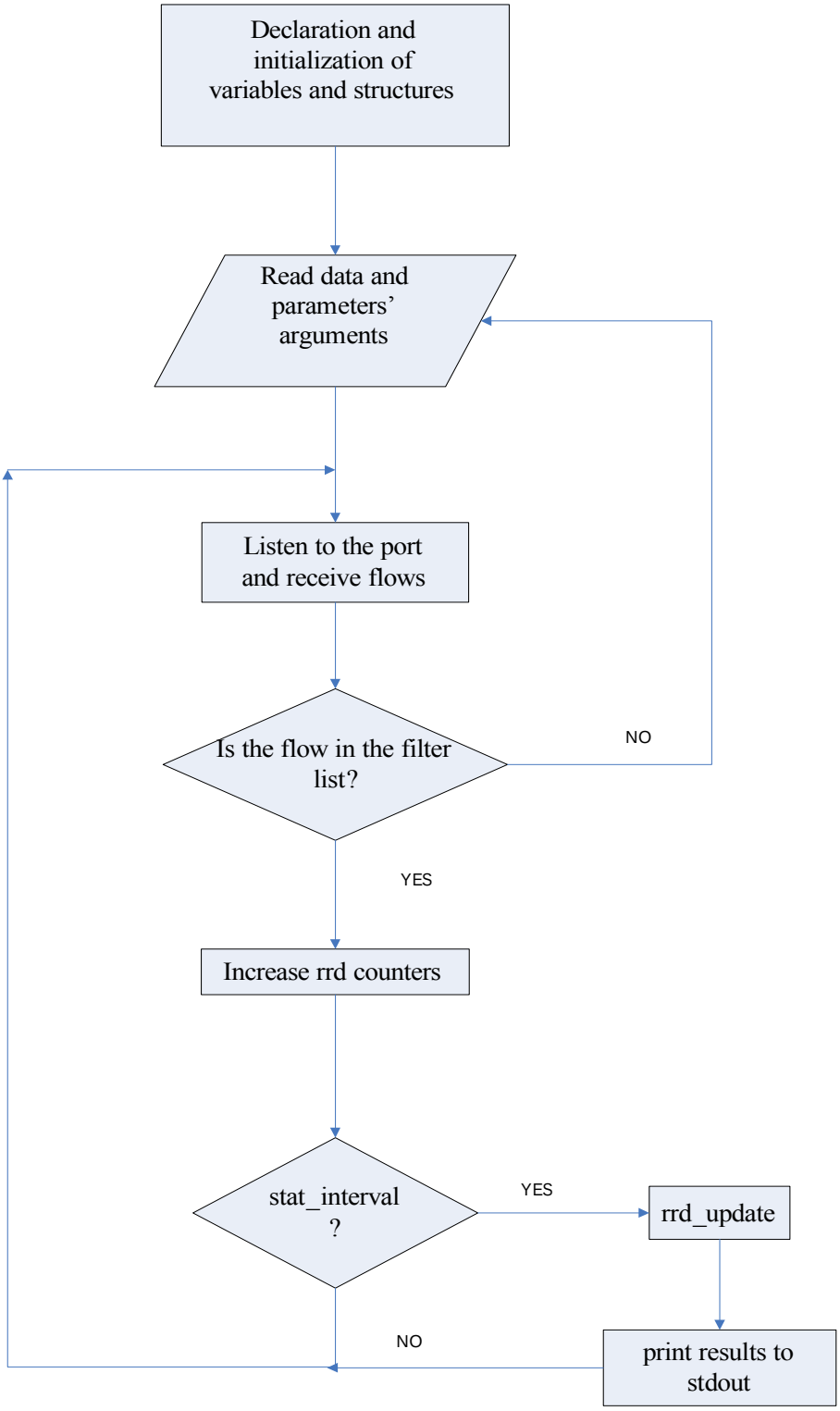
Η γενική λογική του έχει ως εξής : όπως και το απλό flow-receive, έτσι και το flow-rrd-receive, “ακούει” σε ένα προκαθορισμένο port και καταγράφει τα flows των πακέτων που διέρχονται από το σημείο αυτό του δικτύου (με ρυθμό δειγματοληψίας **1/100** πακέτα ). Στη συνέχεια, επεξεργάζεται τις πληροφορίες που περιέχονται στα διάφορα πεδία των flows που λαμβάνει και μεταβάλλει ανάλογα τους αντίστοιχους μετρητές, οι οποίοι έχουν οριστεί αρχικά και αφορούν διάφορες παραμέτρους της διερχόμενης κίνησης των πακέτων, όπως για παράδειγμα το μέγεθος των πακέτων σε bytes (**packet size**), το συνολικό αριθμό των πακέτων που περιέχονται σε ένα flow (**packets per flow**), το μέσο αριθμό flows ανά δευτερόλεπτο (**flows per second**), το μέσο αριθμό Kbits ανά δευτερόλεπτο (**Kbits per second**) κλπ. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς και κάθε φορά που ολοκληρώνεται ένα καθορισμένο χρονικό διάστημα (stat_interval), τα παραπάνω δεδομένα καταχωρούνται στο rrd αρχείο (ft_online.rrd) που έχουμε αρχικά δημιουργήσει. Στο σημείο αυτό δηλαδή πραγματοποιείται η διαδικασία του rrd _update, έτσι όπως την περιγράψαμε προφγουμένως. Ακολουθεί και πάλι η ανάγνωση των flows, η αύξηση των κατάλληλων μετρητών και η ενημέρωση του rrd file με την ολοκλήρωση κάθε χρονικής περιόδου. Ύστερα από την πάροδο κάποιου χρονικού διαστήματος από την έναρξη εκτέλεσης του προγράμματος, στο rrd file περιέχονται οι διάφορες πληροφορίες που μας ενδιαφέρουν, τις οποίες μπορούμε στη συνέχεια να μελετήσουμε και να τις χρησιμοποιήσουμε για τη σχεδίαση αντίστοιχων γραφημάτων.

Σημαντικό στοιχείο του κώδικα αποτελεί επίσης και η δυνατότητα καθορισμού διαφόρων επιλογών, μέσω αντίστοιχων παραμέτρων που δίνονται κατά την κλήση της εντολής. Μία από αυτές που ενδιαφέρει ιδιαίτερα είναι η παράμετρος **–f** στην

οποία ορίζεται το όνομα ενός αρχείου (π.χ. filter.def), το οποίο περιέχει κάποιο φίλτρο που επιθυμούμε να χρησιμοποιηθεί κατά την εκτέλεση του κώδικα. Στο αρχείο αυτό περιέχεται η λίστα των IP διευθύνσεων των οποίων μας ενδιαφέρει η εισερχόμενη ή εξερχόμενη κίνηση και για το λόγο αυτό, τα flows που λαμβάνονται, φιλτράρονται με βάση τη λίστα αυτή.

Με την παράμετρο **–F** ορίζουμε επίσης το συγκεκριμένο όνομα του φίλτρου το οποίο θέλουμε να χρησιμοποιηθεί και περιέχεται στο παραπάνω αρχείο (π.χ. telescope).

Η γενικότερη δομή λειτουργίας του flow-rrd-receive με χρήση φίλτρου, έτσι όπως την περιγράψαμε παραπάνω, μπορεί να παρασταθεί και βάσει του παρακάτω διαγράμματος :


Declaration and initialization of variables and structures
Read data and parameters' arguments
Listen to the port and receive flows
Is the flow in the filter list?
NO
YES
Increase rrd counters
stat_interval ?
YES
rrd_update
NO
print results to stdout

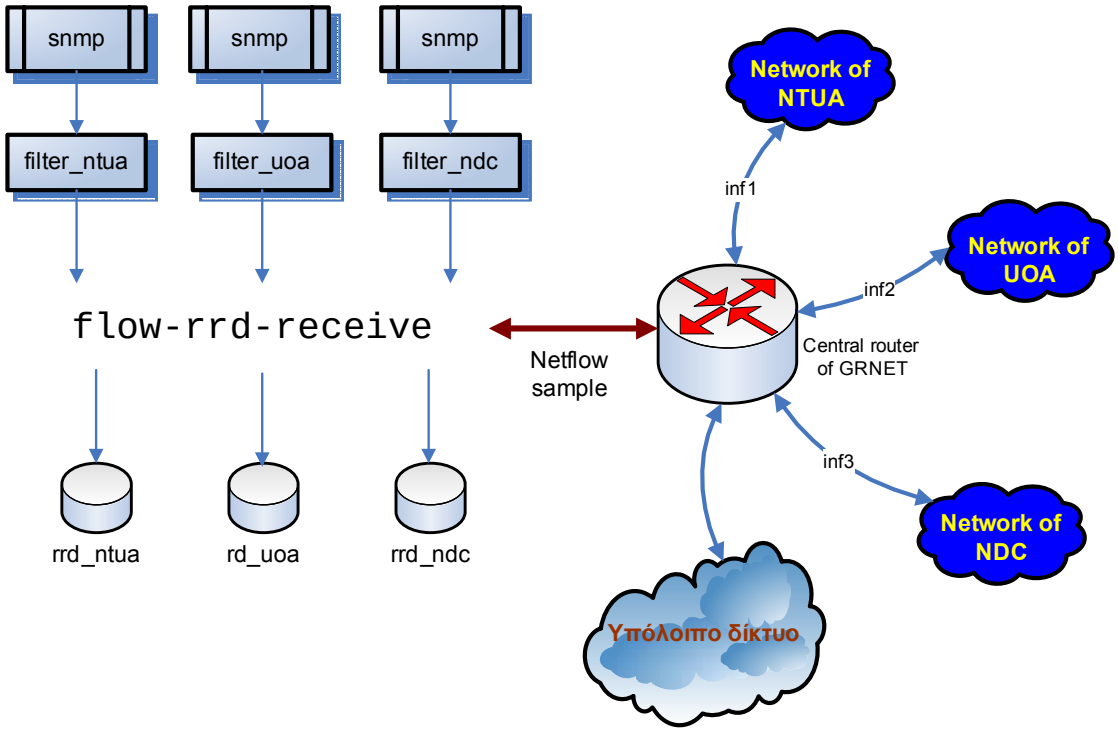
#### 5.2.3.3 Διεύρυνση της λειτουργίας του flow-rrd-receive για ταυτόχρονη εφαρμογή σε περισσότερα του ενός δίκτυα

Η χρήση του παραπάνω εργαλείου θα μπορούσε να γίνει περισσότερο αποδοτική, εάν τα αποτελέσματα που παίρναμε από αυτό μπορούσαν να μας δώσουν πληροφορίες για την κίνηση που θα αφορούσε συγχρόνως ξεχωριστά δίκτυα. Μάλιστα, θα ήταν ακόμη πιο βολικό, εάν για κάθε δίκτυο υπήρχε η δυνατότητα να εφαρμοστεί ένα ξεχωριστό φίλτρο, ανάλογα με το σύνολο των διευθύνσεων για τις οποίες θα ενδιέφερε η μελέτη της κίνησης. Το σύνολο αυτό θα μπορούσε να είναι και πάλι το "unallocated address space" του κάθε δικτύου και έτσι θα γινόταν ταυτόχρονη μελέτη της θεωρούμενης "ύποπτης" κίνησης για όλα τα δίκτυα που θα επιλέγαμε.

Ένας πιθανός τρόπος για να το πετύχουμε αυτό, θα ήταν να καλούμε επανειλημμένως τον ίδιο κώδικα του flow-rrd-receive για κάθε δίκτυο ξεχωριστά. Αυτό όμως δεν αποτελεί μία εύκολη και αποτελεσματική λύση, καθώς για την εκτέλεση του flow-rrd-receive σε κάθε δίκτυο, θα έπρεπε να διαθέτουμε πρόσβαση τουλάχιστον σε έναν από τους δρομολογητές ή hosts του δικτύου αυτού, από τον οποίο θα αντλούσαμε τα αντίστοιχα flows. Κάτι τέτοιο όμως δεν θεωρείται εφικτό.

Με τη λύση που προτείνουμε, μας δίνεται η δυνατότητα λήψης των δεδομένων κίνησης του κάθε δικτύου μέσω ενός κεντρικού router, ο οποίος συνδέεται ταυτόχρονα με τα υπόλοιπα δίκτυα, μέσω ξεχωριστών interfaces.

Το επόμενο σχήμα απεικονίζει την παραπάνω λογική για 3 ξεχωριστά δίκτυα. Υποθέτουμε για παράδειγμα ότι βρισκόμαστε στο δίκτυο του ΕΔΕΤ και θέλουμε να εξάγουμε δεδομένα για το δίκτυο του Ε.Μ.Π, του Πανεπιστημίου Αθηνών και του Εθνικού Κέντρου Τυποποίησης (ΕΚΤ) , τα οποία συνδέονται με έναν κεντρικό router του ΕΔΕΤ, μέσω 3 ξεχωριστών interfaces, inf1, inf2 και inf3.




***Εικόνα 12 :*** *Αναπαράσταση της δομής του μηχανισμού για συλλόγή δεδομένων κίνησης από το δίκτυο του ΕΔΕΤ, με χρήση του νέου εργαλείου flow-rrd-receive*

Το νέο, τροποποιημένο εργαλείο *flow-rrd-receive* θα μας δώσει δεδομένα για καθένα από τα 3 αυτά δίκτυα ξεχωριστά, τα οποία θα καταχωρηθούν σε 3 αντίστοιχα rrd αρχεία, ένα για το καθένα. Τα αρχεία αυτά θα πρέπει να έχουν δημιουργηθεί από πριν με τη βοήθεια του rrdtool create και η καταχώρηση των αντίστοιχων δεδομένων σε καθένα από αυτά, θα γίνει αυτόματα κατά την εκτέλεση του *flow-rrd-receive*. Για κάθε δίκτυο, θα χρησιμοποιηθεί επίσης ένα ξεχωριστό φίλτρο για το 'unallocated address space' του, το οποίο θα έχει δημιουργηθεί με τη βοήθεια του SNMP και των υπόλοιπων ενεργειών που περιγράψαμε στο πρώτο μέρος.

#### 5.2.3.4 Διασχίζοντας τον κώδικα

Στη συνέχεια, θα παρουσιάσουμε τις απαραίτητες μετατροπές που πραγματοποιήσαμε στον αρχικό κώδικα του flow-rrd-receive, προκειμένου να επεκτείνουμε την αρχική λειτουργία του σε μία ευρύτερη εφαρμογή, έτσι όπως την περιγράψαμε παραπάνω.

Οι βασικές μετατροπές και εισαγωγές νέων στοιχείων που πραγματοποιήσαμε στον κώδικα αφορούν στα εξής :

- Καθορισμός των δικτύων για τα οποία θα πραγματοποιηθεί μελέτη της κίνησης με χρήση συγκεκριμένων interfaces.

  Όπως είπαμε, μας ενδιαφέρει να καταγράψουμε ταυτόχρονα την κίνηση που αφορά διαφορετικά δίκτυα, τα οποία συνδέονται σε έναν κεντρικό δρομολογητή μέσω συγκεκριμένων interfaces. Ορίζοντας λοιπόν μία νέα παράμετρο που θα δέχεται ως ορίσματα τους αριθμούς αυτούς των interfaces, μπορούμε να προσδιορίσουμε τα δίκτυα που μας ενδιαφέρουν, καθένα από τα οποία θα αντιστοιχεί σε ένα συγκεκριμένο interface.

  Το flow-rrd-receive διαθέτει ήδη ένα σύνολο παραμέτρων, μέσω των οποίων μπορούν να δοθούν διάφορες επιλογές κατά την κλήση του και είναι οι εξής :

  [-A] : AS substitution, [-b] : output byte order, [-C] : comment field, [-d] : debug,
  [-f] : filter name, [-F] : filter active, [-h]: help, [-s], [-m] : privacy mask,
  [-n] : network ip, [-N] : network mask, [-o] : output filename, [-S] : stat interval ,
  [-t] : tag filename, [-T] : active tag, [-V] : PDU version, [-z] : compress level

  Πώς μπορούμε εμείς να ορίσουμε μία νέα παράμετρο και τα ορίσματα που αυτή θα μπορεί να δέχεται; Αυτό το επιτυγχάνουμε με χρήση της συνάρτησης **getopt.**

  | Function: int **getopt** (*int argc, char **argv, const char *options*) |
  |---|

  Η `getopt` function διαβάζει μία λίστα από ορίσματα που προσδιορίζονται από τα *argv* και *argc*. Οι τιμές των μεταβλητών αυτών αντιστοιχούν στις επιλογές που έχει δώσει ο χρήστης κατά την κλήση του προγράμματος (π.χ -f, -s κλπ.) και λαμβάνονται συνήθως απευθείας από την κυρίως συνάρτηση *main*.

  Το όρισμα *options* είναι ένα string το οποίο περιέχει τη σειρά των χαρακτήρων που μπορεί να δοθούν σαν επιλογές από το χρήστη για το συγκεκριμένο

πρόγραμμα. Εάν η getopt διαβάσει κάποιον από τους χαρακτήρες αυτούς, τον επιστρέφει και στη συνέχεια εκτελείται η αντίστοιχη λειτουργία.

Εάν ο χαρακτήρας αυτός παίρνει επιπλέον και κάποιο όρισμα (π.χ –f filter_name), η getopt επιστρέφει το όρισμα αυτό καταχωρώντας το στη μεταβλητή *optarg*.

Εμείς ορίζουμε δύο νέους χαρακτήρες-παραμέτρους που μπορούν να δοθούν σαν επιλογές από ο χρήστη, για τον καθορισμό συγκεκριμένων interfaces :

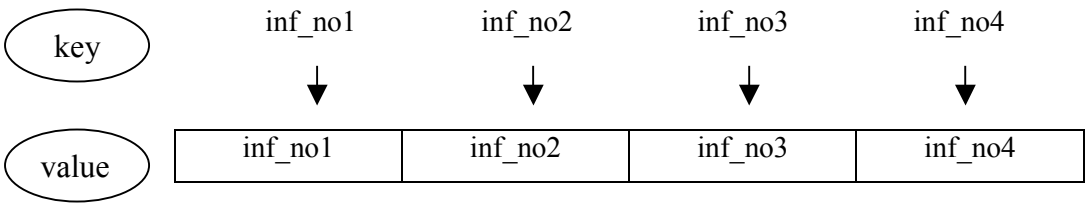
Η παράμετρος **–i** ενεργοποιεί τη λειτουργία εκτέλεσης του προγράμματος για ξεχωριστά interfaces και δεν χρειάζεται κάποιο όρισμα.

Η παράμετρος **–I** παίρνει ως ορίσματα τους συγκεκριμένους αριθμούς των interfaces που θέλουμε να μελετήσουμε, χωρισμένους με κόμμα
(π.χ. –I 158, 106, 0).

Εάν ο χρήστης δώσει τις παραπάνω επιλογές, τότε εκτελούνται αυτόματα οι αντίστοιχες ενέργειες, οι οποίες πραγματοποιούν τα εξής :

Τα ορίσματα της παραμέτρου -I διαβάζονται με τη βοήθεια της 'getopt' και αποθηκεύονται σαν ένα ενιαίο string. Στη συνέχεια, το string αυτό διασπάται στα επιμέρους κομμάτια που αντιστοιχούν στους αριθμούς των interfaces, τα οποία είναι τύπου character. Αφού πραγματοποιηθεί η μετατροπή τους σε integers ώστε να ελαχιστοποιηθεί ο χώρος που καταλαμβάνουν στη μνήμη, καταχωρούνται διαδοχικά σε ένα hash table (**i_hash**) με τη μορφή ζεύγους
(key,value) = (inf_no, inf_no). Το hash table αυτό θα χρησιμοποιηθεί στη συνέχεια για την προσπέλαση των συγκεκριμένων interfaces.

Ο διαχωρισμός του string στα επιμέρους κομμάτια, έγινε με τη βοήθεια της
**strtok function,** η οποία αποδεικνύεται χρήσιμη στην περίπτωση που δίνονται παραπάνω από ένα ορίσματα, τα οποία πρέπει να διαχειριστούν σαν ξεχωριστά δεδομένα.




**i_hash**

Το κομμάτι του κώδικα που εκτελεί τις παραπάνω ενέργειες παρουσιάζεται στη συνέχεια. Για διευκόλυνση παραθέτουμε και τις αρχικοποιήσεις των αντίστοιχων μεταβλητών που έχουν γίνει σε προηγούμενο σημείο του προγράμματος :

```
  const char delimiters[] = ",";
  char *token;
  int tok;
  int *p_tok;
  char *inf_active;
  inf_active = (char*)0L;
  ght_hash_table_t *i_hash;
  i_hash = ght_create(10) ;

while ((i = getopt(argc, argv, "A:b:C:d:f:F:h?m:o:siI:S:t:T:V:z:n:N:"))!=-1)
switch (i) {
……
case 'i': /* check flows for specific interfaces */
          enable_per_inf=1;
          break;

case 'I': /* numbers of interfaces specified */
       inf_active = optarg;
                       token = strtok (inf_active, delimiters);
                  tok = atoi (token);

         /* fill in the 'i_hash' with the interfaces' identical numbers */
                while (1) {

                    p_tok = malloc(sizeof(int));
                    *p_tok = tok;

                  ght_insert (i_hash, p_tok, sizeof(char)*strlen(token), token);

                    token = strtok (NULL, delimiters);
                    if (token == NULL) {
                     break;
                    }
                    tok =atoi (token);
                  }

              break;
              ……

          }
```

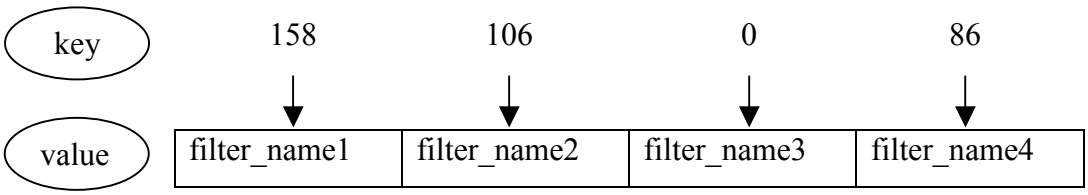
- Καθορισμός φίλτρων για εφαρμογή στα δεδομένα κίνησης του κάθε interface

  Όπως είπαμε, μας ενδιαφέρει να μπορούμε να καθορίσουμε ένα ξεχωριστό φίλτρο με διαφορετικούς κανόνες για το κάθε interface, το οποίο θα αντιστοιχίζεται σε αυτό και θα εφαρμόζεται κάθε φορά που λαμβάνονται δεδομένα για το συγκεκριμένο interface.

  Για το σκοπό αυτό, μετατρέπουμε τις λειτουργίες που εκτελούνται με την εισαγωγή της παραμέτρου –F, η οποία μπορεί πλέον να δέχεται παραπάνω του ενός ορίσματα, τα οποία καθορίζουν τα ονόματα των επιμέρους φίλτρων, καθένα από τα οποία θα χρησιμοποιηθεί για ένα ξεχωριστό interface. Θεωρούμε ότι τα filter names διαχωρίζονται με τα αντίστοιχα interfaces τους με μία παύλα
  (π.χ –F filter_name1-158, filter_name2-106, filter_name3-0,…).
  Στο σημείο αυτό κάνουμε την παραδοχή ότι όλα τα φίλτρα περιλαμβάνονται στο ίδιο αρχείο, το όνομα του οποίου δίνεται ως όρισμα της παραμέτρου –f
  (π.χ. -f filter.def)

  Το όνομα του κάθε φίλτρου, καταχωρείται και πάλι σε ένα hash table (**f_hash**), έχοντας ως *key* τον αριθμό του interface και *value* το όνομα του αντίστοιχου φίλτρου. Αυτό χρησιμεύει στη συνέχεια, όταν γνωρίζοντας τον αριθμό του κάθε interface, μπορούμε εύκολα να βρούμε το αντίστοιχο φίλτρο που θα εφαρμοστεί στα flows τα οποία προέρχονται ή προορίζονται προς αυτό.

  Ο διαχωρισμός των ορισμάτων που εισάγονται κατά την κλήση της εντολής, γίνεται και πάλι με τη βοήθεια της συνάρτησης **strtok**.




**f_hash**

Το κομμάτι του κώδικα που ακολουθεί, παρουσιάζει αναλυτικά τις εντολές που εκτελούνται με την εισαγωγή της παραμέτρου –f και –F και των αντίστοιχων ορισμάτων τους :

```
char *filter;
char  *filter_list, *f_name, *inf_no;
int i_no;
int *p_i_no, *f_key;
char *filter_fname, *filter_active;
ght_hash_table_t *f_hash;

filter_fname = FT_PATH_CFG_FILTER;
filter_active = (char*)0L;
f_hash = ght_create(100) ;
.
.
.
case 'f': /* filter fname */
      filter_fname = optarg;
      break;

case 'F': /* filter_name */

      filter_active = 1;

      filter_list=optarg;
      f_name = strtok (filter_list, "-");


      while (1) {

                  inf_no = strtok (NULL, ",");
/*case of one single filter - variable 'filter_name' contains the name of the
filter for use */

                           if (inf_no ==NULL) {
                                        filter_name = f_name;
                                        break;
                                       }
 /* case of more filters - fill in the 'f_hash' with the filter-names */

                         i_no = atoi(inf_no);
                         p_i_no = malloc(sizeof(int));
                         *p_i_no = i_no;
                         ght_insert(f_hash, f_name, sizeof(p_i_no),p_i_no);
                         f_name = strtok (NULL, "-");
                         if (f_name ==NULL) { break;}

                    }

      break;


/* load filters */

if (filter_active) {

     if (ftfil_load(&ftfil, filter_fname))
      fterr_errx(1, "ftfil_load(%s): failed", filter_fname);

     if (!enable_per_inf) {
      if (!(ftfd = ftfil_def_find(&ftfil, filter_name)))
      fterr_errx(1, "ftfil_def_find(%s): failed", filter_name);
     }
  }
```

- Επιλογή των flows κάθε δικτύου, καταχώρησή τους και μεταβολή των αντίστοιχων μετρητών

Τι συμβαίνει στην περίπτωση που επιλέγουμε να μελετήσουμε ξεχωριστά τα δεδομένα κίνησης που αφορούν διαφορετικά δίκτυα; Πώς μπορούμε να εντοπίσουμε και στη συνέχεια να απομονώσουμε τα εισερχόμενα ή εξερχόμενα flows του κάθε δικτυου; Η γενική λογική στην οποία στηριζόμαστε είναι η εξής:

Για κάθε flow το οποίο διέρχεται από το σημείο του δικτύου όπου 'ακούει' το flow-rrd-receive (είτε εισερχόμενο είτε εξερχόμενο από το δίκτυό μας), πραγματοποιείται μία σύγκριση μεταξύ αυτού (του αντίστοιχου interface) και των αριθμών των interfaces που έχουν οριστεί από το χρήστη. Εάν το interface του flow μέσω του οποίου εισέρχεται ή εξέρχεται, αντιστοιχεί σε κάποιο από τα interfaces που έχουν δοθεί ως πιθανές επιλογές, τότε το flow γίνεται αποδεκτό και οι πληροφορίες που μεταφέρει καταχωρούνται στο hash table με το όνομα **'inf_hash'**. Για την καταχώρηση χρησιμοποιείται ως *key* ο αριθμός του interface και ως *value,* θεωρούνται οι πληροφορίες που μεταφέρει το flow. Ακολουθεί η αύξηση των κατάλληλων μετρητών. Διαφορετικά, το flow απορρίπτεται και εξετάζεται το επόμενο.

Εάν επιπλέον έχει οριστεί κάποιο φίλτρο για εφαρμογή στα δεδομένα ενός συγκεκριμένου interface, τότε πριν την αύξηση των μετρητών, γίνεται ένας ακόμη έλεγχος για να εξεταστεί εάν ικανοποιούνται οι κανόνες του φίλτρου από τα στοιχεία του flow ή όχι. Το όνομα του αρχείου με το κατάλληλο φίλτρο που θα εφαρμοστεί στο interface, ανασύρεται από το f-hash, με το *key* να έχει την τιμή του αντίστοιχου interface number.

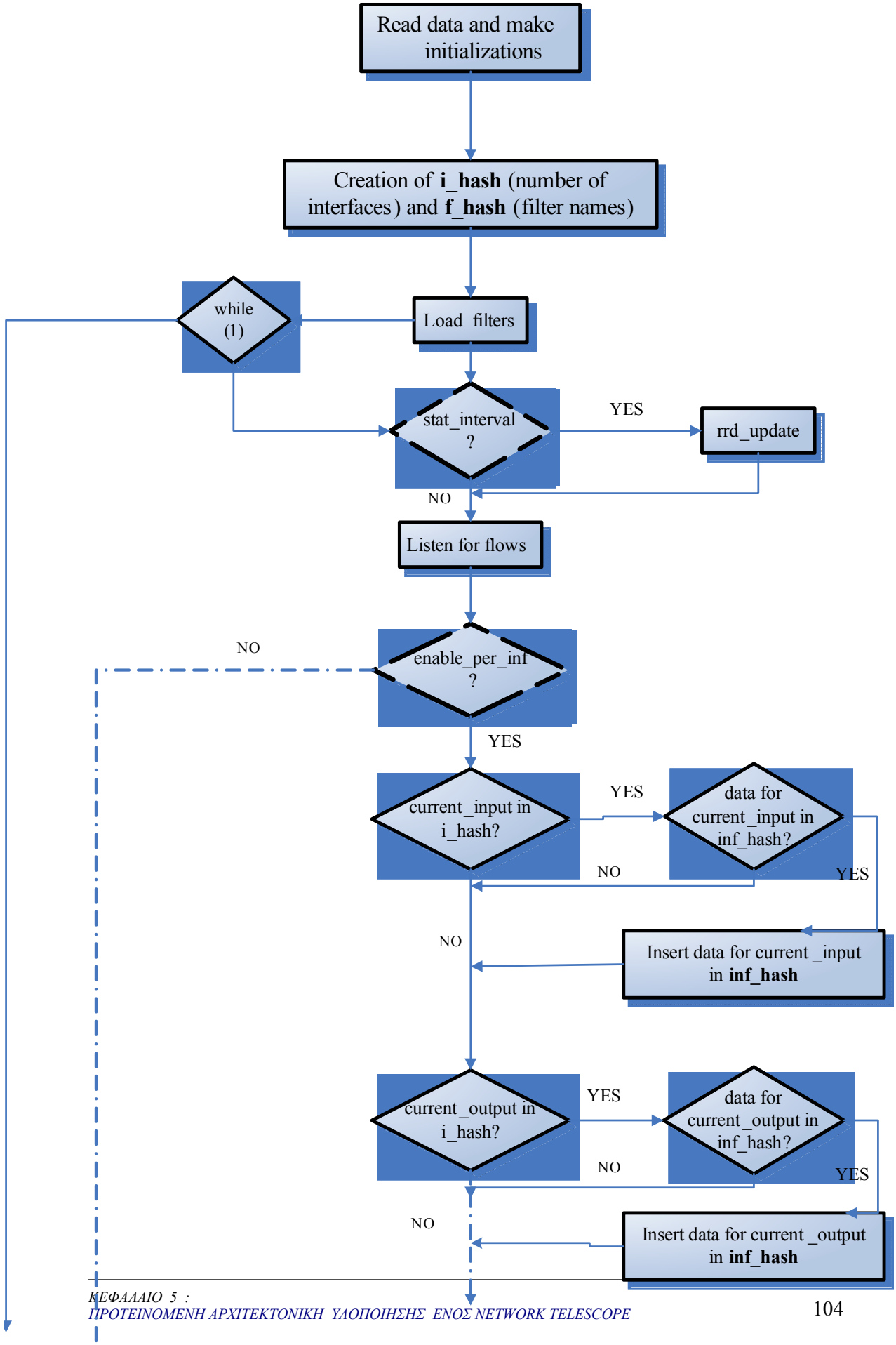
Στην περίπτωση που το flow γίνει τελικά αποδεκτό, αφού περάσει από τους παραπάνω ελέγχους, ακολουθεί κατάλληλη τροποποίηση των rrd μετρητών, σύμφωνα με τις πληροφορίες που αυτό μεταφέρει. Εδώ πραγματοποιείται ένας διαχωρισμός των flows που θεωρούνται εισερχόμενα ή εξερχόμενα για το interface που μελετάμε και με βάση αυτόν μεταβάλλονται ανάλογα οι κατάλληλοι μετρητές, οι οποίοι είναι διαφορετικοί για την εισερχόμενη και για την εξερχόμενη κίνηση.

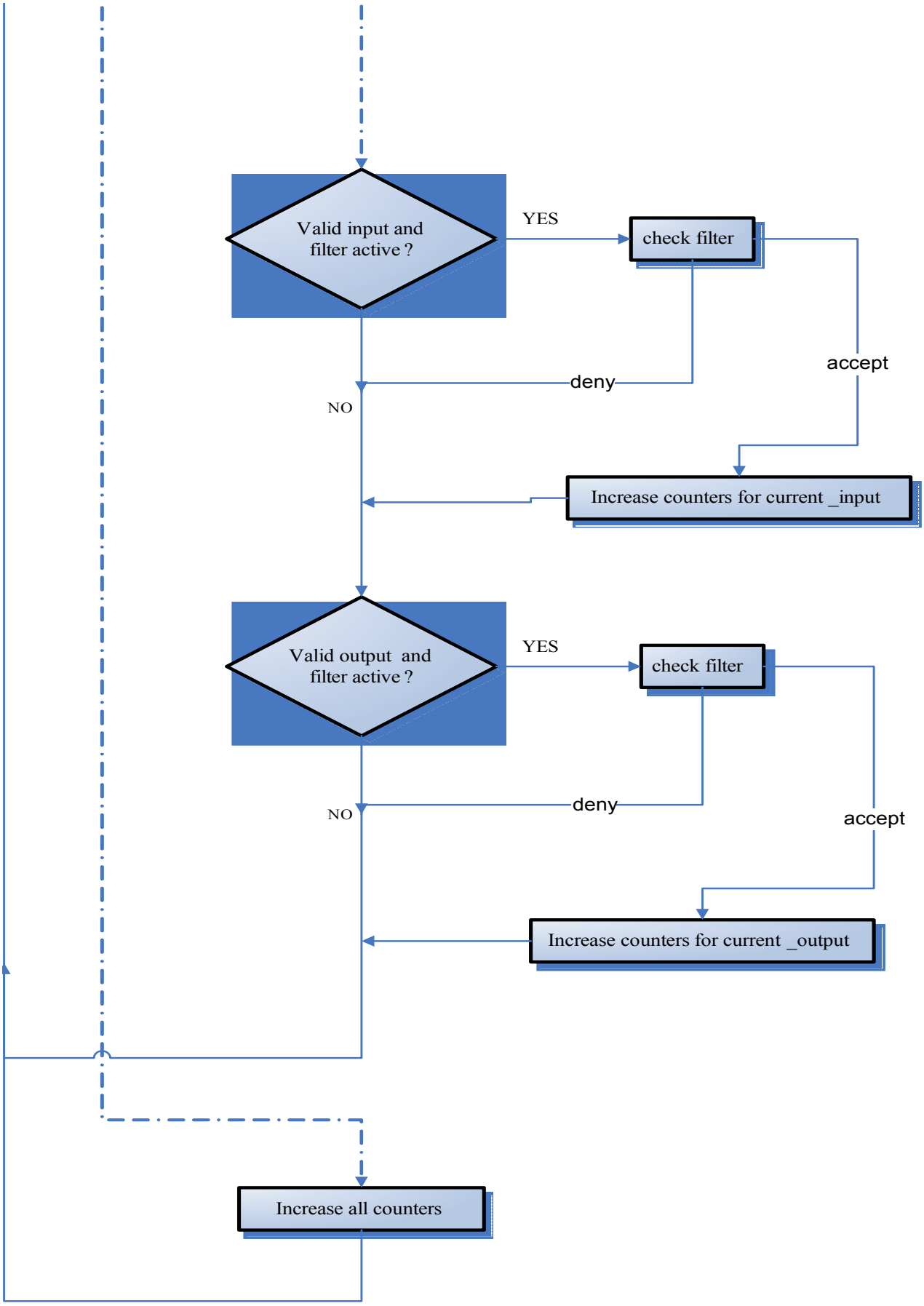
Πιο αναλυτικά, οι ενέργειες που εκτελούνται για την υλοποίηση των παραπάνω, φαίνονται στη συνέχεια με τη μορφή ψευδοκώδικα και λογικού διαγράμματος:

```
/* Εάν η ΄λειτουργία ανά interface’ έχει ενεργοποιηθεί */
   Εάν (enable_per_inf_)
   {
      Για κάθε στοιχείο του i_hash (no_of_interace)

    {    /* Σύγκρινε το no_of_interface με το current_input */

           /* valid current_input */
           Εάν (no_of_interace = current_input)
           {  /*Ενημέρωση κατάλληλης μεταβλητής */
                valid_inp=1;
           }
           Αλλιώς valid_inp = 0;  /* invalid current_input */

           Εάν (valid_inp =1)
           {
             Καταχώρησε τις πληροφορίες του flow για το συγκεκριμένο
             interface στο inf_hash (εάν δεν είναι ήδη
   καταχωρημένες)

           }

       /* Σύγκρινε το no_of_interace με το current_output */
           Εάν (no_of_interface = current_output)

           /* valid current_output */
           {  /*Ενημέρωση κατάλληλης μεταβλητής */
                valid_out=1; }

           Αλλιώς valid_out =0;  /* invalid current_output */

           Εάν (valid_out =1)
           {
             Καταχώρησε τις πληροφορίες του flow για το συγκεκριμένο
             interface στο inf_hash (εάν δεν είναι ήδη
             καταχωρημένες)
           }
         }
       }
```

```
/* Εύρεση κατάλληλου φίλτρου για το current _input */
Εάν (enable_per_inf_) /* λειτουργία ανά interface */
 {
   Εάν (filter_active AND valid current _input)
   {
       Εντόπισε το αντίστοιχο φίλτρο (filter_fname) του
       interface από το f_hash

       Εάν τα στοιχεία του flow ικανοποιούν το φίλτρο
       { fi_deny=1;}
        Αλλιώς { fi_deny=0;}
    }
   Εάν δεν έχει προσδιοριστεί φίλτρο για το συγκεκριμένο interface
   τότε (fi_deny=1;)

/* Αύξηση rrd μετρητών σε περίπτωση έγκυρων δεδομένων */
   Εάν (fi_deny=1) /* οι κανόνες του φίλτρου ικανοποιούνται */
   {
      Αύξησε τους αντίστοιχους μετρητές για το current_input
   }

/* Εύρεση κατάλληλου φίλτρου για το current _output */
   Εάν (filter_active AND valid current _output)
   {
       Εντόπισε το αντίστοιχο φίλτρο (filter_fname) του
       interface από το f_hash

       Εάν τα στοιχεία του flow ικανοποιούν το φίλτρο
       { fi_deny=1;}
        Αλλιώς { fi_deny=0;}
    }
   Εάν δεν έχει προσδιοριστεί φίλτρο για το συγκεκριμένο interface
   τότε (fi_deny=1;)

/* Αύξηση rrd μετρητών σε περίπτωση έγκυρων δεδομένων */
   Εάν (fi_deny=1)
   {
      Αύξησε τους αντίστοιχους μετρητές για το current_output
   }
```

```
}
Αλλιώς /* απλή περίπτωση – όλα τα interfaces */
{
    Έλεγχος του κατάλληλου φίλτρου (εάν έχει δοθεί)

   Αν το φίλτρο επαληθεύεται από τα στοιχεία του flow
    {  Αύξηση των rrd μετρητών }

  Αλλιώς
     { Επιστροφή σε προηγούμενο βήμα για ανάγνωση του επόμενου flow
}

}
```


Read data and make initializations
Creation of **i_hash** (number of interfaces) and **f_hash** (filter names)
while (1)
Load filters
stat_interval ?
YES
rrd_update
NO
Listen for flows
NO
enable_per_inf ?
YES
current_input in i_hash?
YES
data for current_input in inf_hash?
NO
YES
NO
Insert data for current _input in **inf_hash**
current_output in i_hash?
YES
data for current_output in inf_hash?
NO
YES
NO
Insert data for current _output in **inf_hash**


Valid input and filter active ?
YES
check filter
accept
deny
NO
Increase counters for current _input
Valid output and filter active ?
YES
check filter
deny
NO
accept
Increase counters for current _output
Increase all counters

```
/* accept flows for specific interfaces */
if (enable_per_inf) {

                char hashed_i[100];
                ght_iterator_t iterator;
                ght_hash_table_t *inf_hash;
                char* inf_key;
                int *no_of_inf, *filter_inp_key;
                int cnt, cnt2, valid_inp, valid_out;
                inf_hash = ght_create(100) ;
                struct stat0* testp1, testp;

                cnt=0;
                cnt2=0;

        for (no_of_inf = ght_first(i_hash, &iterator, &p_key); no_of_inf;
               no_of_inf = ght_next(i_hash, &iterator, &p_key))
             {

                 /* check cur.input */
                sprintf(hashed_i,"%d",cur.input);

                valid_inp=0;

                if (cur.input == *no_of_inf) { valid_inp=1;}
                else  {valid_inp=0;}

                if (valid_inp == 1) /* valid cur.input */
                {
                   /* case of interface 0 */
                  if (cur.input==0) {
                      sprintf(hashed_i,"i0");
                      }
                if ((testp1=(struct stat0*) ght_get(inf_hash,
        sizeof(hashed_i),hashed_i)))
                 {
                        //found
                        } else
                        {
                       //not found then insert data for current input in 'inf_hash'
                        testp1 =malloc(sizeof(struct stat0));
                        bzero(testp1,sizeof(struct stat0));
                        ght_insert(inf_hash,testp1,sizeof(hashed_i),hashed_i);
                        printf("NEW ENTRY %s into hash \n",hashed_i);
```

```
                testp1->start = 0xFFFFFFFF;
                testp1->end= 0;
                testp1->time_start= 0xFFFFFFFF;
                testp1->time_end= 0;
        }

    filter_inp_key = no_of_inf;

    cnt=1; /* valid input found */
}

/*check cur.output*/

  sprintf(hashed_i,"%d",cur.output);
  valid_out = 0;

  if (cur.output == *no_of_inf)  { valid_out=1;}  else {valid_out=0;}

     if (valid_out == 1) /* valid cur.output */
      {
        /* case of interface 0 */
         if (cur.output==0) {
         sprintf(hashed_i,"i0");
         }
         if ((testp=(struct stat0*) ght_get(inf_hash,  sizeof(hashed_i),hashed_i)))
          {
          //found
           } else
           {
           //not found then insert data for current output in 'inf_hash'
           testp =malloc(sizeof(struct stat0));
           bzero(testp,sizeof(struct stat0));
           ght_insert(inf_hash,testp,sizeof(hashed_i),hashed_i);
           printf("NEW ENTRY2 %s into hash \n",hashed_i);
           testp->start = 0xFFFFFFFF;
           testp->end= 0;
           testp->time_start= 0xFFFFFFFF;
           testp->time_end= 0;
          }

     filter_out_key = no_of_inf;
     cnt2=1;  /* valid output found */
     }
  }
```

```
  }

/* look for filter for valid current input */
if (filter_active && cnt) {

/* find the appropriate filter name */
    filter_name = ght_get(f_hash, sizeof(int), filter_inp_key);
        if (filter_name != NULL) {

   if (!(ftfd = ftfil_def_find(&ftfil, filter_name)))
   fterr_errx(1, "ftfil_def_find(%s): failed", filter_name);


/* if a filter is specified check its rules for current input */
   if (ftfd) {
    if (ftfil_def_eval(ftfd, out_rec, &fo) == FT_FIL_MODE_DENY) {
        ++filtered_flows;
        fi_permit = 0; /* deny flows */
        }
       else { fi_permit = 1; } /* permit flows */
      }
    }

 /* no filter - permit all flows */
    else  { fi_permit = 1; }
  }

if (enable_per_inf) {

/* increase counters for current input if permitted by the filter */
 if (cnt && fi_permit )
 COUNT_ACTION;


/* look for filter for valid current output */

        if (filter_active && cnt2) {
/* find the appropriate filter name */
    filter_name = ght_get(f_hash, sizeof(filter_out_key), filter_out_key);
        if (filter_name != NULL) {


      if (!(ftfd = ftfil_def_find(&ftfil, filter_name)))
      fterr_errx(1, "ftfil_def_find(%s): failed", filter_name);
```

```
   /* if a  filter is specified check its rules for current output */
   if (ftfd) {
    if (ftfil_def_eval(ftfd, out_rec, &fo) == FT_FIL_MODE_DENY) {
         ++filtered_flows;
         fo_permit = 0;  /* deny flows */
       }
       else  {  fo_permit = 1; }  /* permit flows */
      }
   }

/* no filter - permit all flows */
   else  { fo_permit = 1; }
 }


/* increase counters for current output if permitted by the filter*/
if (cnt2 && fo_permit)
COUNT_ACTION;


/*accept all flows*/
else {

     /* filter? */
        if (ftfd)
          if (ftfil_def_eval(ftfd, out_rec, &fo) == FT_FIL_MODE_DENY) {
            ++filtered_flows;
            continue;
          }
```

- Καταγραφή δεδομένων στο rrd αρχείο – rrdupdate

Σκοπός της συγκεκριμένης εφαρμογής, όπως είπαμε, είναι η δημιουργία rrd αρχείου, όπου θα περιέχονται οι πληροφορίες των flows που συγκεντρώθηκαν μέσω των αντίστοιχων μετρητών και η συνεχής ανανέωσή του με τα νέα δεδομένα, ύστερα από την πάροδο κατάλληλου χρονικού διαστήματος. Η δημιουργία του αρχείου αυτού που στην ουσία αποτελεί μία *rrd database*, γίνεται αρχικά από το χρήστη, με τη βοήθεια της συνάρτησης *rrdtool-create*, όπου ταυτόχρονα ορίζονται και οι διάφορες παράμετροι που έχουμε περιγράψει παραπάνω (όπως ο χρόνος ανανέωσης των δεδομένων της βάσης – update time).

Αφού έχει δημιουργηθεί το rrd file με την κατάλληλη ονομασία, μπορεί πλέον να εκτελεστεί το flow-rrd-receive και να αρχίσει η διαδικασία καταχώρησης και ανανέωσης δεδομένων στην rrd database. Αυτό πραγματοποιείται σε καθορισμένα χρονικά διαστήματα, με τη βοήθεια της συνάρτησης *rrdtool update*. Ταυτόχρονα, τυπώνονται και στην οθόνη κάποια από τα παραπάνω δεδομένα.

Στην περίπτωση που έχουμε επιλέξει τη ‘λειτουργία ανά interface’, τότε θα πρέπει να δημιουργηθούν διαφορετικά rrd αρχεία, καθένα από τα οποία θα αντιστοιχεί σε ένα και μόνο interface και θα περιλαμβάνει τις πληροφορίες που αφορούν μόνο αυτό. Το update θα γίνεται κάθε φορά για όλα τα αρχεία αυτά.

Στη συνέχεια, παραθέτουμε το κομμάτι του κώδικα που εκτελεί τη διαδικασία του *rrd-create* και του *rrd-update*. Αυτό γίνεται με τη βοήθεια της STAT_ACTION που ορίζεται αρχικά, μέσα από την οποία πραγματοποιούνται οι αρχικοποιήσεις κατάλληλων μεταβλητών και η γίνεται η κλήση της *rrdtool update* :

**rrd - create**

```
/usr/local/bin/rrdtool create ft_online.rrd --step $1 \
DS:kbits_per_sec:GAUGE:800:U:U \
DS:flows_per_sec:GAUGE:800:U:U \
DS:packets_per_flow:GAUGE:800:U:U \
DS:packets_size:GAUGE:800:U:U \
DS:flow_size:GAUGE:800:U:U \
DS:ip_pkt_size32:GAUGE:800:U:U \
DS:ip_pkt_size64:GAUGE:800:U:U \
DS:ip_pkt_size96:GAUGE:800:U:U \
DS:ip_pkt_size128:GAUGE:800:U:U \
DS:ip_pkt_size160:GAUGE:800:U:U \
DS:ip_pkt_size192:GAUGE:800:U:U \
DS:ip_pkt_size224:GAUGE:800:U:U \
DS:ip_pkt_size256:GAUGE:800:U:U \
DS:ip_pkt_size288:GAUGE:800:U:U \
DS:ip_pkt_size320:GAUGE:800:U:U \
DS:pkts_per_flow1:GAUGE:800:U:U \
DS:pkts_per_flow2:GAUGE:800:U:U \
DS:pkts_per_flow4:GAUGE:800:U:U \
DS:pkts_per_flow8:GAUGE:800:U:U \
DS:pkts_per_flow12:GAUGE:800:U:U \
DS:pkts_per_flow16:GAUGE:800:U:U \
DS:pkts_per_flow20:GAUGE:800:U:U \
DS:pkts_per_flow24:GAUGE:800:U:U \
DS:pkts_per_flow28:GAUGE:800:U:U \
DS:pkts_per_flow32:GAUGE:800:U:U \
DS:octs_per_flow32:GAUGE:800:U:U \
DS:octs_per_flow64:GAUGE:800:U:U \
DS:octs_per_flow128:GAUGE:800:U:U \
DS:octs_per_flow256:GAUGE:800:U:U \
DS:octs_per_flow512:GAUGE:800:U:U \
DS:octs_per_flow1280:GAUGE:800:U:U \
DS:octs_per_flow2048:GAUGE:800:U:U \
DS:octs_per_flow2816:GAUGE:800:U:U \
DS:octs_per_flow3584:GAUGE:800:U:U \
DS:octs_per_flow4352:GAUGE:800:U:U \
DS:octs_per_flow15872g:GAUGE:800:U:U \
DS:flow_time_dst10:GAUGE:800:U:U \
DS:flow_time_dst50:GAUGE:800:U:U \
DS:flow_time_dst100:GAUGE:800:U:U \
DS:flow_time_dst200:GAUGE:800:U:U \
DS:flow_time_dst500:GAUGE:800:U:U \
DS:flow_time_dst1000:GAUGE:800:U:U \
```

```
DS:flow_time_dst2000:GAUGE:800:U:U \
DS:flow_time_dst3000:GAUGE:800:U:U \
DS:flow_time_dst4000:GAUGE:800:U:U \
DS:syn_only_nflows:GAUGE:800:U:U \
DS:in_udp_noctets:GAUGE:800:U:U \
DS:in_udp_npackets:GAUGE:800:U:U \
DS:out_udp_noctets:GAUGE:800:U:U \
DS:out_udp_npackets:GAUGE:800:U:U \
DS:in_icmp_noctets:GAUGE:800:U:U \
DS:in_icmp_npackets:GAUGE:800:U:U \
DS:out_icmp_noctets:GAUGE:800:U:U \
DS:out_icmp_npackets:GAUGE:800:U:U \
DS:nv_only_nflows:GAUGE:800:U:U \
DS:in_tcp_noctets:GAUGE:800:U:U \
DS:in_tcp_npackets:GAUGE:800:U:U \
DS:out_tcp_noctets:GAUGE:800:U:U \
DS:out_tcp_npackets:GAUGE:800:U:U \
RRA:AVERAGE:0.5:1:28800 \
RRA:AVERAGE:0.5:5:28800 \
RRA:MAX:0.5:1:28800 \
RRA:MAX:0.5:5:28800 \
RRA:HWPREDICT:1440:0.1:0.0035:288
```

**rrd - update**

```
#define STAT_ACTION\
         fs0.aflowtime = fs0.time / fs0.nflows;\
        fs0.aps = fs0.noctets / fs0.npackets;\
        fs0.afs = fs0.noctets / fs0.nflows;\
        fs0.apf = fs0.npackets / fs0.nflows;\
        fs0.fps = (float)fs0.nflows / ((fs0.end - fs0.start) / 1000);\
        fs0.aos = ((float)(fs0.noctets*8) / 1000) / ((fs0.end -
fs0.start) / 1000);\
        fs0.time_real = fs0.time_end - fs0.time_start;\
        fs0.fps_real = (float)fs0.nflows / (float)fs0.time_real;\
        fs0.aos_real = ((float)(fs0.noctets*8) / 1000) /
(fs0.time_real);\
        strcpy(fmt_buf, "Total Flows                    : ");\
        fmt_uint64(fmt_buf+34, fs0.nflows, FMT_JUST_LEFT);\
        puts(fmt_buf);\
        strcpy(fmt_buf, "Total Octets                   : ");\
        fmt_uint64(fmt_buf+34, fs0.noctets, FMT_JUST_LEFT);\
        puts(fmt_buf);\
        strcpy(fmt_buf, "Total Packets                  : ");\
```

```
        fmt_uint64(fmt_buf+34, fs0.npackets, FMT_JUST_LEFT);\
        puts(fmt_buf);\
        strcpy(fmt_buf, "Total Time (1/1000 secs) (flows): ");\
        fmt_uint64(fmt_buf+34, fs0.time, FMT_JUST_LEFT);\
        puts(fmt_buf);\
        strcpy(fmt_buf, "Duration of data  (realtime)    : ");\
        fmt_uint32(fmt_buf+34, fs0.time_real, FMT_JUST_LEFT);\
        puts(fmt_buf);\
        strcpy(fmt_buf, "Duration of data (1/1000 secs)  : ");\
        fmt_uint64(fmt_buf+34, (fs0.end - fs0.start), FMT_JUST_LEFT);\
        puts(fmt_buf);\
        printf("Average flow time (1/1000 secs) : %4.4f\n",
fs0.aflowtime);\
        printf("Average packet size (octets)    : %4.4f\n", fs0.aps);\
        printf("Average flow size (octets)      : %4.4f\n", fs0.afs);\
        printf("Average packets per flow        : %4.4f\n", fs0.apf);\
        printf("Average flows / second (flow)   : %4.4f\n", fs0.fps);\
        printf("Average flows / second (real)   : %4.4f\n",
fs0.fps_real);\
        printf("Average Kbits / second (flow)   : %4.4f\n", fs0.aos);\
        printf("Average Kbits / second (real)   : %4.4f\n",
fs0.aos_real);\
        printf("\n\n");\


sprintf(rrd_update1_arg2,"%lu:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%
f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f
:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f:%f",now,\

fs0.aos_real,fs0.fps_real,fs0.apf,fs0.aps,fs0.afs,(float)fs0.psize32
,(float)fs0.psize64 ,\
            (float)fs0.psize96 ,(float)fs0.psize128 ,(float)fs0.psize160
,\
            (float)fs0.psize192 ,(float)fs0.psize224 ,(float)fs0.psize256
,\
            (float)fs0.psize288 ,(float)fs0.psize320 ,\
            (float)fs0.fpsize1 ,(float)fs0.fpsize2 ,(float)fs0.fpsize4 ,\
            (float)fs0.fpsize8 ,(float)fs0.fpsize12 ,(float)fs0.fpsize16
,\
            (float)fs0.fpsize20 ,(float)fs0.fpsize24 ,(float)fs0.fpsize28
,(float)fs0.fpsize32 ,\
            (float)fs0.fosize32 ,(float)fs0.fosize64
,(float)fs0.fosize128 ,(float)fs0.fosize256 ,\
            (float)fs0.fosize512 ,(float)fs0.fosize1280
```

```
,(float)fs0.fosize2048 ,\
            (float)fs0.fosize2816 ,(float)fs0.fosize3584
,(float)fs0.fosize4352 ,(float)fs0.fosize_other ,\
            (float)fs0.ftime10 ,(float)fs0.ftime50 ,(float)fs0.ftime100
,(float)fs0.ftime200 ,\
            (float)fs0.ftime500 ,(float)fs0.ftime1000
,(float)fs0.ftime2000 ,(float)fs0.ftime3000 ,\
            (float)fs0.ftime4000 , (float) fs0.syn_only_nflows , (float)
fs0.in_udp_noctets , (float)    fs0.in_udp_npackets ,\
            (float)fs0.out_udp_noctets ,(float)
fs0.out_udp_npackets,(float) fs0.in_icmp_noctets,(float)
fs0.in_icmp_npackets,\
            (float)fs0.out_icmp_noctets,(float) fs0.out_icmp_npackets ,
(float) fs0.nv_only_nflows ,(float) fs0.in_tcp_noctets,\
            (float) fs0.in_tcp_npackets,
(float)fs0.out_tcp_noctets,(float) fs0.out_tcp_npackets );\
        optind = 0; opterr = 0;\
        rrd_update(3,rrdargv1);\
        if (rrd_test_error()) {\
        printf("MYFILE:%s \n MYARG2:%s \n RRD:%s\n",
rrdargv1[1],rrdargv1[2], rrd_get_error());\
        rrd_clear_error();\
        }\
        printf("rrdupdate: %lu\n",now);\



if(stat_interval)
   { . . .

if (enable_per_inf) {

ght_iterator_t iterator;
char* p_key;

for (testp_rrd = (struct stat0*)ght_first(inf_hash, &iterator, &p_key);
testp_rrd;
       testp_rrd= (struct stat0*)ght_next(inf_hash, &iterator, &p_key))
        {
        //Write to different rrd per Interface
        sprintf(rrd_update1_arg1,"ft_online%s.rrd",p_key);
          rrd_update1_arg1[strlen(rrd_update1_arg1)]='\0';

        fs0=*testp_rrd;
```

```
        if (fs0.nflows==0) { printf("nflows=0\n");
        continue; }
        printf("Interface %s \n",p_key);
        STAT_ACTION;
        flag=0;
        bzero(&fs0, sizeof fs0);
        fs0.start = 0xFFFFFFFF;
        fs0.end = 0;
        fs0.time_start = 0xFFFFFFFF;
        fs0.time_end = 0;

        cur.flows = 1;
        *testp_rrd=fs0;

        } //end of for loop
        } //end of  if
        else
        {
        STAT_ACTION;
        bzero(&fs0, sizeof fs0);
        fs0.start = 0xFFFFFFFF;
        fs0.end = 0;
        fs0.time_start = 0xFFFFFFFF;
        fs0.time_end = 0;
        cur.flows = 1;
        }
    }
}
```

Στη συνέχεια παρουσιάζονται αναλυτικά τα είδη των μετρήσεων που παίρνουμε με την εκτέλεση του flow-rrd-receive. Στον πρώτο πίνακα φαίνονται τα αποτελέσματα που εμφανίζονται περιοδικά στην οθόνη με την εκτέλεση του προγράμματος, ενώ στον δεύτερο πίνακα παρουσιάζονται οι μετρητές των οποίων οι τιμές αποθηκεύονται στο rrd αρχείο που δημιουργείται και οι οποίες μεταβάλλονται δυναμικά κατά την εκτέλεση της εφαρμογής, κάθε φορά που γίνεται το update των δεδομένων :

**ΠΙΝΑΚΑΣ Ι**

| | |
|---|---|
| **Total flows** | Συνολικός αριθμός flows |
| **Total Octets** | Συνολικός αριθμός bytes |
| **Total Packets** | Συνολικός αριθμός πακέτων |
| **Duration of data (realtime)** | Πραγματική διάρκεια δεδομένων |
| **Average packet size (octets)** | Μέσο μέγεθος πακέτων (bytes) |
| **Average flow size (octets)** | Μέσο μέγεθος flows (bytes) |
| **Average packets per flow** | Μέσος αριθμός πακέτων / flow |
| **Average flows / second (real)** | Μέσος αριθμός flows / δευτερόλεπτο |
| **Average Kbits / second (real)** | Μέσος αριθμός Kbits / δευτερόλεπτο |

**ΠΙΝΑΚΑΣ ΙΙ**

| | |
|---|---|
| kbits_per_sec | flows_per_sec |
| packets_per_flow | packets_size |
| flow_size | ip_pkt_size |
| octs_per_flow | flow_time_dst |
| in_udp_noctets | in_udp_npackets |
| out_udp_noctets | out_udp_npackets |
| in_icmp_noctets | in_icmp_npackets |
| out_icmp_noctets | out_icmp_npackets |
| in_tcp_noctets | in_tcp_npackets |
| out_tcp_noctets | out_tcp_npackets |
| syn_only_nflows | nv_only_nflows |

# ΚΕΦΑΛΑΙΟ 6 : ΠΕΙΡΑΜΑΤΙΚΗ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ NETWORK TELESCOPE ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΔΙΚΤΥΟ

Στο κεφάλαιο αυτό, θα γίνει μία παρουσίαση της λειτουργίας του network telescope έτσι όπως αυτή περιγράφηκε παραπάνω, μέσα από την εφαρμογή του σε πραγματικό δίκτυο και τη συγκέντρωση δεδομένων που αφορούν την κίνηση του δικτύου αυτού. Για το σκοπό αυτό, επιλέγουμε το δίκτυο του Ε.Μ.Π. για το οποίο θα συλλέξουμε και θα παρουσιάσουμε κάποια ενδεικτικά στοιχεία κίνησης, έτσι όπως θα προκύψουν ύστερα από την εφαρμογή του telescope. Συγκριτικά, θα αναφέρουμε και αντίστοιχα στοιχεία που προκύπτουν για το δίκτυο του Πανεπιστημίου Αθηνών.

## 6.1 ΕΝΔΕΙΚΤΙΚΗ ΕΦΑΡΜΟΓΗ ΤΟΥ TELESCOPE ΣΤΟ ΔΙΚΤΥΟ ΤΟΥ Ε.Μ.Π.

Η συγκέντρωση δεδομένων κίνησης θα γίνει για το σύνολο των μη χρησιμοποιούμενων διευθύνσεων του δικτύου του Ε.Μ.Π. και είναι αυτή που έχουμε αναφέρει ως ‘κακόβουλη’ κίνηση. Πρώτο βήμα λοιπόν είναι η δημιουργία του ‘unallocated address space’ του δικτύου, η οποία πραγματοποιείται με την εφαρμογή του `Telescope.pl`, έχοντας προσδιορίσει τα κατάλληλα δεδομένα εισόδου μέσα από το configuration file `TelescopeConfig.pm`. Τα δεδομένα αυτά αφορούν την IP διεύθυνση του κατάλληλου router μέσω του οποίου θα εξαχθούν όλες οι χρησιμοποιούμενες διευθύνσεις του δικτύου (allocated address space) και την IP διεύθυνση η οποία αντιπροσωπεύει το διαθέσιμο πλήθος διευθύνσεων για το σύνολο του δικτύου (total address space).

Συγκεκριμένα, για το δίκτυο του Ε.Μ.Π., προσδιορίζουμε τα εξής δεδομένα :

**TelescopeConfig.pm**

```
%Telescope::Config = (
# Please give the router's IP for use by the SNMP protocol
                        router_ip => '147.102.255.5',
# Please give the network IPs that represent the total address space,
# in CIDR format ('xxx.xxx.xxx.xxx/mask') starting from the IP with the
# bigger mask
                        network_ip1 => '147.102.0.0/17',
                        network_ip2 => '147.102.128.0/17',
# Please give the file-name with the source-data of the allocated address
# space. In other case give the default value 'data'
                        data_file => 'data',
# Please give the name for the filter to be created
                        filter_name => 'telescope_ntua' );
                       );
```

Ύστερα από την εφαρμογή του `Telescope.pl`, το ζητούμενο unallocated address space έχει δημιουργηθεί στο αρχείο `'filter.def'`, έχοντας τη μορφή κατάλληλου φίλτρου, το οποίο ορίζεται με την ονομασία `'telescope_ntua'`. Το φίλτρο αυτό θα χρησιμοποιηθεί στη συνέχεια από τα flow-tools για τη συγκέντρωση δεδομένων κίνησης.

Τα περιεχόμενα του `'filter.def'` με τις unallocated IP addresses του δικτύου του ΕΜΠ, καθώς και τα περιεχόμενα του αρχείου `'ntua_res'` με τις allocated IP addresses το οποίο δημιουργείται αρχικά από την εφαρμογή, παρουσιάζονται στη συνέχεια. Όπως παρατηρούμε, οι IP διευθύνσεις αντιστοιχούν σε υποδίκτυα και παριστάνονται με την κλασική CIDR μορφή, δηλαδή xxx.xxx.xxx.xxx / mask. Είναι επίσης φανερό ότι το σύνολο των διευθύνσεων του ενός αρχείου αποτελεί ουσιαστικά το συμπλήρωμα του συνόλου που περιέχεται στο δεύτερο αρχείο, ενώ και τα δύο μαζί συνθέτουν το ευρύτερο σύνολο που αντιστοιχεί στο total address space του δικτύου.

**‘ntua_res’ (allocated address space)**

147.102.2.0/23
147.102.5.0/24
147.102.6.0/23
147.102.8.0/24
147.102.10.0/23
147.102.12.0/22
147.102.16.0/24
147.102.18.0/23
147.102.20.0/23
147.102.23.0/24
147.102.24.0/21
147.102.32.0/21
147.102.40.0/22
147.102.46.0/23
147.102.48.0/22
147.102.52.0/24
147.102.55.0/24
147.102.58.0/24
147.102.71.0/24
147.102.72.0/24
147.102.81.0/24
147.102.82.0/23
147.102.84.0/23
147.102.88.0/24
147.102.92.0/24
147.102.98.0/24
147.102.100.0/23
147.102.102.0/24
147.102.106.0/23
147.102.109.0/24
147.102.110.0/23
147.102.112.0/24
147.102.116.0/24
147.102.121.0/24
147.102.122.0/23
147.102.124.0/24
147.102.128.0/24
147.102.150.0/23
147.102.152.0/21
147.102.160.0/22
147.102.172.0/22
147.102.176.0/23
147.102.178.0/24
147.102.190.0/23
147.102.192.0/23
147.102.194.0/24
147.102.205.0/24
147.102.206.0/24
147.102.208.0/24
147.102.210.0/23
147.102.212.0/22
147.102.219.0/24
147.102.220.0/24
147.102.221.128/25
147.102.222.0/23
147.102.224.8/29
147.102.224.16/28
147.102.224.32/30
147.102.224.44/30
147.102.224.48/30
147.102.224.64/27
147.102.224.96/29
147.102.224.196/30
147.102.225.0/24
147.102.228.0/23
147.102.230.0/24
147.102.232.0/23
147.102.240.0/24
147.102.243.0/24
147.102.245.0/24
147.102.246.0/23
147.102.254.0/24
147.102.255.4/30
147.102.255.2/31
147.102.255.1/32

**‘filter.def’ (unallocated address space)**

```
filter-primitive test-prefix
type ip-address-prefix
permit 147.102.136.0/21
permit 147.102.60.0/22
permit 147.102.64.0/22
permit 147.102.76.0/22
permit 147.102.132.0/22
permit 147.102.144.0/22
permit 147.102.164.0/22
permit 147.102.168.0/22
permit 147.102.180.0/22
permit 147.102.184.0/22
permit 147.102.196.0/22
permit 147.102.200.0/22
permit 147.102.236.0/22
permit 147.102.248.0/22
permit 147.102.44.0/23
permit 147.102.56.0/23
permit 147.102.68.0/23
permit 147.102.74.0/23
permit 147.102.86.0/23
permit 147.102.90.0/23
permit 147.102.94.0/23
permit 147.102.96.0/23
permit 147.102.104.0/23
permit 147.102.114.0/23
permit 147.102.118.0/23
permit 147.102.126.0/23
permit 147.102.130.0/23
permit 147.102.148.0/23
permit 147.102.188.0/23
permit 147.102.216.0/23
permit 147.102.226.0/23
permit 147.102.234.0/23
permit 147.102.252.0/23
permit 147.102.0.0/24
permit 147.102.4.0/24
permit 147.102.9.0/24
permit 147.102.17.0/24
permit 147.102.22.0/24
permit 147.102.53.0/24
permit 147.102.54.0/24
permit 147.102.59.0/24
permit 147.102.70.0/24
permit 147.102.73.0/24
permit 147.102.80.0/24
permit 147.102.89.0/24
permit 147.102.93.0/24
permit 147.102.99.0/24
permit 147.102.103.0/24
permit 147.102.108.0/24
permit 147.102.113.0/24
permit 147.102.117.0/24
permit 147.102.120.0/24
permit 147.102.125.0/24
permit 147.102.129.0/24
permit 147.102.179.0/24
permit 147.102.195.0/24
permit 147.102.204.0/24
permit 147.102.207.0/24
permit 147.102.209.0/24
permit 147.102.218.0/24
permit 147.102.231.0/24
permit 147.102.241.0/24
permit 147.102.242.0/24
permit 147.102.244.0/24
permit 147.102.221.0/25
permit 147.102.255.128/25
permit 147.102.224.128/26
permit 147.102.255.64/26
permit 147.102.224.224/27
permit 147.102.255.32/27
permit 147.102.224.112/28
permit 147.102.224.208/28
permit 147.102.255.16/28
permit 147.102.224.0/29
permit 147.102.224.56/29
permit 147.102.224.104/29
permit 147.102.224.200/29
permit 147.102.255.8/29
permit 147.102.224.52/30
permit 147.102.224.192/30
permit 147.102.255.0/32
default deny
```

```
filter-definition telescope_ntua
match ip-destination-address  test-prefix
or
match ip-source-address test-prefix
```

Το μέγεθος του "unallocated address space" που προκύπτει για το δίκτυο του NTUA είναι **33.435 IPs.** Η τιμή αυτή χαρακτηρίζει επίσης και το μέγεθος του telescope.

## 6.2 ΕΦΑΡΜΟΓΗ ΤΟΥ TELESCOPE ΣΤΟ ΔΙΚΤΥΟ ΤΟΥ Ε.Κ.Π.Α

Με ανάλογη εφαρμογή του `Telescope.pl` όπως και προηγουμένως, δημιουργούμε στο αρχείο `'filter.def'` ένα επιπλέον φίλτρο με όνομα `'telescope_uoa'`, το οποίο αφορά τις unallocated addresses για το δίκτυο του UOA, όπως φαίνεται στη συνέχεια. Το total address space του δικτύου αντιπροσωπεύεται από τη διεύθυνση **`'195.134.64.0/18'`** .

```
filter-primitive test-prefix2
type ip-address-prefix
permit 195.134.104.0/22
permit 195.134.72.0/23
permit 195.134.77.0/24
permit 195.134.78.0/24
permit 195.134.103.0/24
permit 195.134.108.0/24
permit 195.134.116.0/24
permit 195.134.74.128/25
permit 195.134.92.128/25
permit 195.134.109.128/25
permit 195.134.110.128/25
permit 195.134.118.0/25
permit 195.134.124.0/25
permit 195.134.68.128/26
permit 195.134.89.192/26
permit 195.134.98.192/26
permit 195.134.101.192/26
permit 195.134.102.192/26
permit 195.134.109.64/26
permit 195.134.111.128/26
permit 195.134.68.224/27
permit 195.134.64.164/30
permit 195.134.90.0/27
permit 195.134.90.160/27
permit 195.134.90.192/27
permit 195.134.101.96/27
permit 195.134.110.32/27
permit 195.134.113.192/27
permit 195.134.102.0/28
permit 195.134.110.16/28
permit 195.134.113.224/28
permit 195.134.64.176/29
permit 195.134.64.248/29
permit 195.134.98.0/29
permit 195.134.98.88/29
permit 195.134.102.80/29
permit 195.134.102.184/29
permit 195.134.109.8/29
permit 195.134.64.0/30
permit 195.134.64.48/30
permit 195.134.64.84/30
permit 195.134.64.92/30
permit 195.134.64.108/30
permit 195.134.64.116/30
```

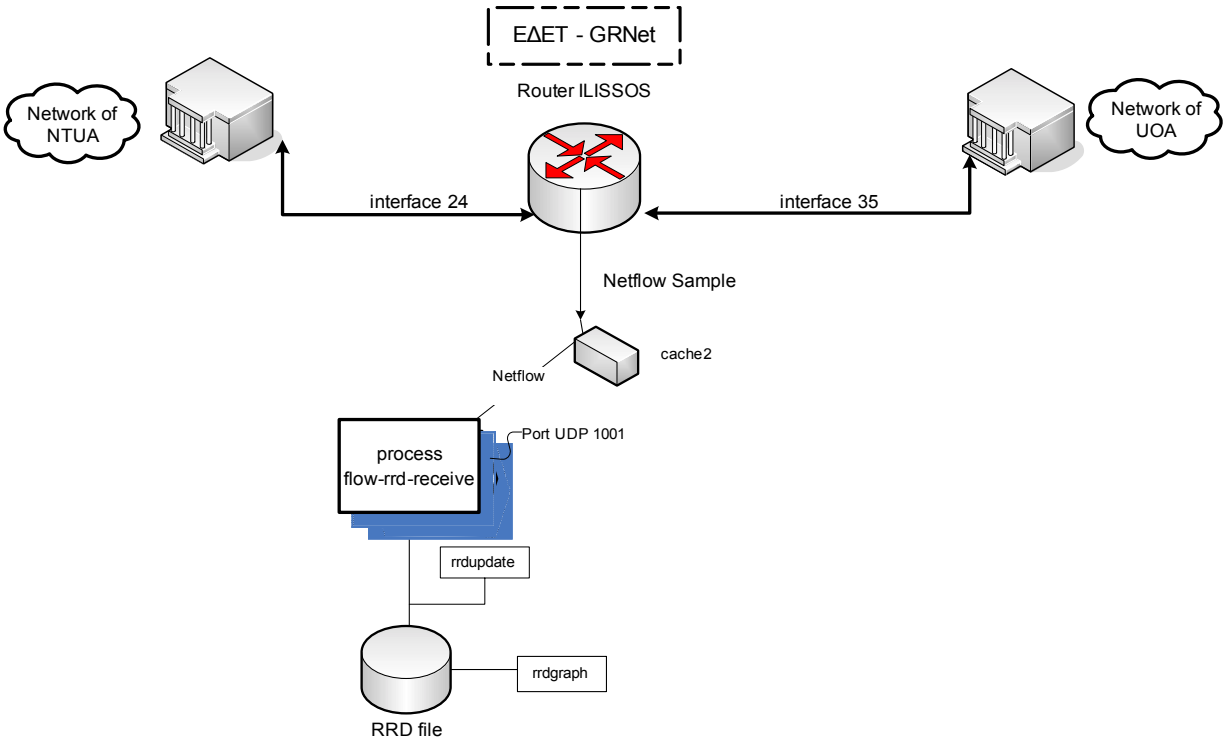
```
permit 195.134.64.204/30
permit 195.134.102.148/30
permit 195.134.64.30/31
permit 195.134.64.192/31
permit 195.134.109.0/31
permit 195.134.109.6/31
permit 195.134.64.155/32
permit 195.134.64.159/32
permit 195.134.109.2/32
permit 195.134.109.5/32
default deny

filter-definition telescope_uoa
match ip-destination-address  test-prefix2
or
match ip-source-address test-prefix2
```

Το μέγεθος του "unallocated address space" που προκύπτει για το δίκτυο του UOA είναι **4.314 IPs.**

## 6.3 ΣΥΓΚΕΝΤΡΩΣΗ ΚΑΙ ΚΑΤΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ 'ΥΠΟΠΤΗΣ' ΚΙΝΗΣΗΣ ΓΙΑ ΤΑ ΔΙΚΤΥΑ ΤΩΝ Ε.Μ.Π ΚΑΙ Ε.Κ.Π.Α

Έχοντας πλέον στη διάθεσή μας τα unallocated address spaces των δύο δικτύων, τόσο του NTUA όσο και του UOA, μπορούμε να ξεκινήσουμε να συγκεντρώνουμε παράλληλα στοιχεία για την κίνηση που δρομολογείται προς τα κομμάτια αυτά, κάνοντας χρήση των flow-tools. Για το σκοπό αυτό χρησιμοποιούμε ένα ενδεικτικό σύνολο δεδομένων κίνησης (netflow sample) που αφορούν ολόκληρο το δίκτυο του Ε.Μ.Π., τα οποία αντλούμε από έναν κεντρικό router (ilissos) του ευρύτερου δικτύου του ΕΔΕΤ.

Το επόμενο σχήμα παριστάνει τη συνδεσμολογία των παραπάνω δικτύων και τον τρόπο με τον οποίο συλλέγουμε τα δεδομένα κίνησης :




***Εικόνα 13 :*** *Συνδεσμολογία των δικτύων του ΕΔΕΤ και αναπάράσταση της δομής για τη συλλογή δεδομένων με χρήση του flow-rrd-receive*

### 6.3.1 Συλλογή δεδομένων δικτυακής κίνησης με χρήση των flow-tools

Με εφαρμογή των λειτουργιών που μας προσφέρουν τα εργαλεία όπως είναι τα flow-tools και με παράλληλη χρήση του φίλτρου `'telescope_ntua'` που έχουμε ήδη δημιουργήσει, απομονώνουμε και καταγράφουμε τα flows που σχετίζονται με το unallocated address space του Ε.Μ.Π. Τα δεδομένα που συλλέξαμε για διάρκεια μίας ημέρας, στις **4-10-2005**, φαίνονται στη συνέχεια :

```
> flow-cat ilissos-nf | flow-nfilter -f filter.def -F telescope_ntua
   | flow-stat -f 0

#  --- ---- ---- Report Information --- --- ---
#
# Fields:     Total
# Symbols:    Disabled
# Sorting:    None
# Name:       Overall Summary
#
# Args:       flow-stat -f 0
#
```

```
Total Flows                      : 28811
Total Octets                     : 9814518
Total Packets                    : 28877
Total Time (1/1000 secs) (flows): 313024
Duration of data  (realtime)     : 86380
Average flow time (1/1000 secs) : 10.0000
Average packet size (octets)     : 339.0000
Average flow size (octets)       : 340.0000
Average packets per flow         : 1.0000
Average flows / second (real)    : 0.3335
Average Kbits / second (flow)    : 0.9088
Average Kbits / second (real)    : 0.9090
```

Το εργαλείο **flow-stat** μας δίνει κάποιες γενικές πληροφορίες για το σύνολο των flows που περιέχονται στο netflow-sample (ilissos-nf) και προορίζονται προς τις unallocated addresses του δικτύου. Μπορούμε να παρατηρήσουμε πιο αναλυτικά τα στοιχεία που αφορούν τα παραπάνω flows με τη βοήθεια του **flow-print**. Ενδεικτικά παραθέτουμε κάποια από αυτά :

```
> flow-cat ilissos-nf | flow-nfilter -f filter.def -F telescope_ntua
  | flow-print
```

| Sif | SrcIPaddress | Dif | DstIPaddress | Pr | SrcP | DstP | Pkts | Octets |
|---|---|---|---|---|---|---|---|---|
| 0007 | 81.1.121.85 | 0018 | 147.102.165.27 | 06 | f8e | 3b0e | 1 | 48 |
| 0007 | 202.97.35.182 | 0018 | 147.102.118.100 | 01 | 0 | b00 | 1 | 56 |
| 0007 | 208.53.81.141 | 0018 | 147.102.227.149 | 06 | 5c2 | 3b0e | 1 | 48 |
| 0007 | 201.250.103.189 | 0018 | 147.102.9.180 | 11 | 10a0 | 229c | 1 | 38 |
| 0007 | 205.200.1.229 | 0018 | 147.102.249.27 | 06 | dc2 | 3b0e | 1 | 48 |
| 0007 | 128.119.108.149 | 0018 | 147.102.9.180 | 11 | b3d | 229c | 1 | 38 |
| 0007 | 65.151.148.210 | 0018 | 147.102.97.30 | 06 | d19 | 3b0e | 1 | 48 |
| 0007 | 208.53.81.141 | 0018 | 147.102.227.153 | 06 | 4ae | 3b0e | 1 | 48 |
| 0007 | 213.240.2.126 | 0018 | 147.102.114.165 | 06 | d3a | 3b0e | 1 | 48 |
| 0007 | 61.133.3.47 | 0018 | 147.102.186.0 | 06 | 50 | d1aa | 1 | 40 |
| 0007 | 61.152.90.160 | 0018 | 147.102.224.151 | 06 | 50 | d751 | 1 | 48 |
| 0007 | 84.13.132.102 | 0018 | 147.102.89.166 | 06 | e80 | 3b0e | 1 | 48 |
| 0007 | 62.101.185.164 | 0018 | 147.102.78.130 | 06 | 905 | 2df8 | 1 | 48 |
| 0007 | 61.141.32.90 | 0018 | 147.102.133.198 | 06 | 50 | d78d | 1 | 44 |
| 0007 | 71.137.213.144 | 0018 | 147.102.63.201 | 06 | 7dc | 3b0e | 1 | 48 |
| 0007 | 61.168.255.97 | 0018 | 147.102.115.134 | 01 | 0 | b00 | 1 | 168 |
| 0007 | 218.27.127.77 | 0018 | 147.102.126.88 | 01 | 0 | b00 | 1 | 56 |
| 0007 | 71.137.213.144 | 0018 | 147.102.63.213 | 06 | 1311 | 3b0e | 1 | 48 |
| 0007 | 205.200.1.229 | 0018 | 147.102.249.52 | 06 | e3f | 3b0e | 1 | 48 |
| 0007 | 61.129.15.84 | 0018 | 147.102.231.172 | 06 | 50 | 864 | 1 | 48 |
| 0007 | 66.24.117.160 | 0018 | 147.102.9.180 | 11 | c63 | b946 | 1 | 38 |
| 0007 | 82.2.121.249 | 0018 | 147.102.70.10 | 06 | 1069 | 3b0e | 1 | 48 |
| 0007 | 68.190.63.192 | 0018 | 147.102.115.150 | 06 | 6e2 | 3b0e | 1 | 48 |

```
0007 61.141.32.90     0018 147.102.68.143    06 50   6f7f  1          44
0007 68.190.63.192    0018 147.102.115.155   06 1126 3b0e  1          48
0007 80.97.67.144     0018 147.102.179.112   06 be3  3b0e  1          48
0007 193.203.141.85   0018 147.102.79.189    06 806  3b0e  1          48
```

Όπως παρατηρούμε, το `destination interface` για όλα τα παραπάνω flows είναι το `0018(Hex) = 24(dec)`, το οποίο αντιστοιχεί στον αριθμό του interface που συνδέει τον κεντρικό router ilisso του ΕΔΕΤ με το δίκτυο του Ε.Μ.Π. Επιπλέον, είναι φανερό ότι όλες οι `destination IPs` περιλαμβάνονται στο σύνολο του unallocated address space του δικτύου, όπως ήταν αναμενόμενο. Το πλήθος των flows που καταγράφηκαν για τη συγκεκριμένη μέρα που έγινε η μέτρηση, προέκυψε ίσο με **28.811 flows** τα οποία αναλογούν σε **33.435 IPs** συνολικά.

Με χρήση και πάλι των flow-tools και του φίλτρου `telescope_uoa` συγκεντρώνουμε τα αντίστοιχα στοιχεία για τα flows που αφορούν το δίκτυο του UOA. Για τη διάρκεια της ίδιας μέρας και πάλι (**4-10-2005)**, συγκεντρώθηκαν τα παρακάτω στοιχεία κίνησης :

***Συνολικά στοιχεία κίνησης :***

```
> flow-cat ilissos-nf | flow-nfilter -f filter2.def -F telescope_uoa
  |  flow- stat -f 0
```

```
#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Disabled
# Sorting:   None
# Name:      Overall Summary
#
# Args:      flow-stat -f 0
#
Total Flows                     : 2976
Total Octets                    : 289585
Total Packets                   : 2985
Total Time (1/1000 secs) (flows): 71644
Duration of data  (realtime)    : 86313
Duration of data (1/1000 secs)  : 86312256
Average flow time (1/1000 secs) : 24.0000
Average packet size (octets)    : 97.0000
Average flow size (octets)      : 97.0000
Average packets per flow        : 1.0000
Average flows / second (flow)   : 0.0345
Average flows / second (real)   : 0.0345
Average Kbits / second (flow)   : 0.0268
Average Kbits / second (real)   : 0.0268
```

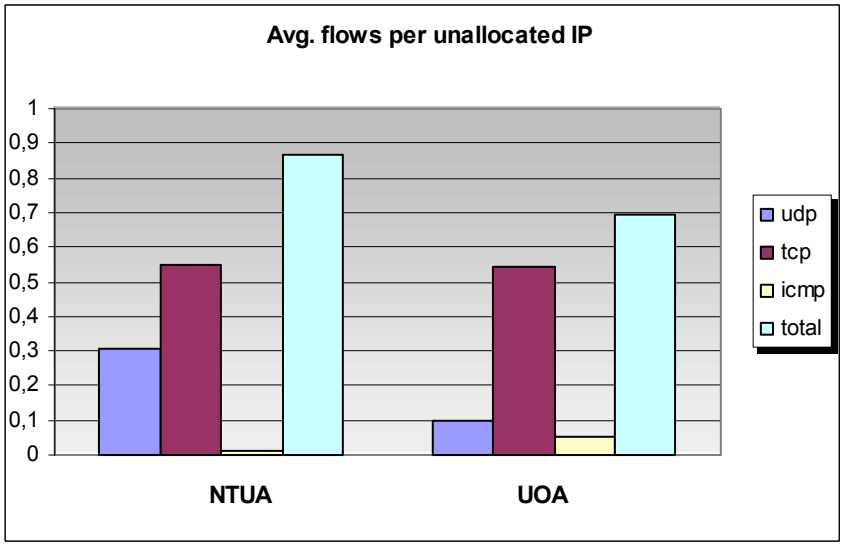
***Πληροφορίες αναφορικά με το κάθε πρωτόκολλο :***

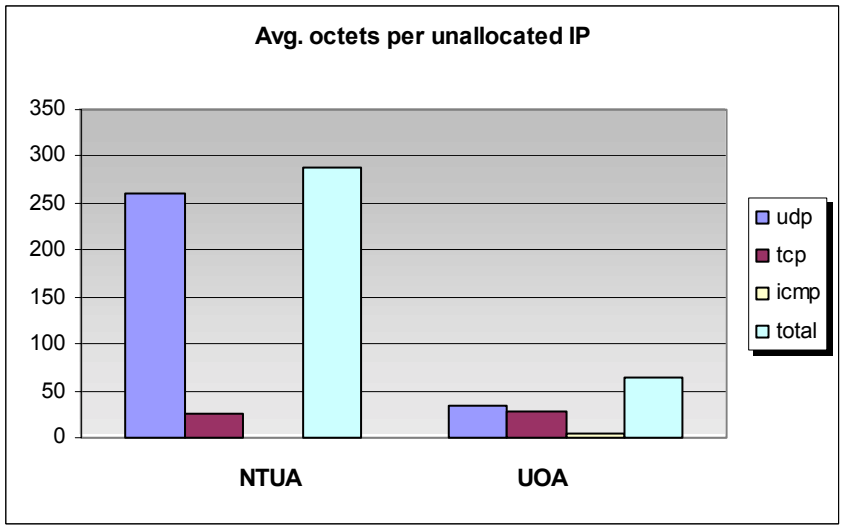
```
> flow-cat ilissos-nf | flow-nfilter -f filter2.def -F telescope_uoa
  |  flow- stat -f 12


#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Disabled
# Sorting:   None
# Name:      IP protocol
#
# Args:      flow-stat -f 12
#
#
# protocol  flows                octets                packets
#
    17      452                  154537                452
     6      2229                 110776                2234
     1      295                  24272                 299
```

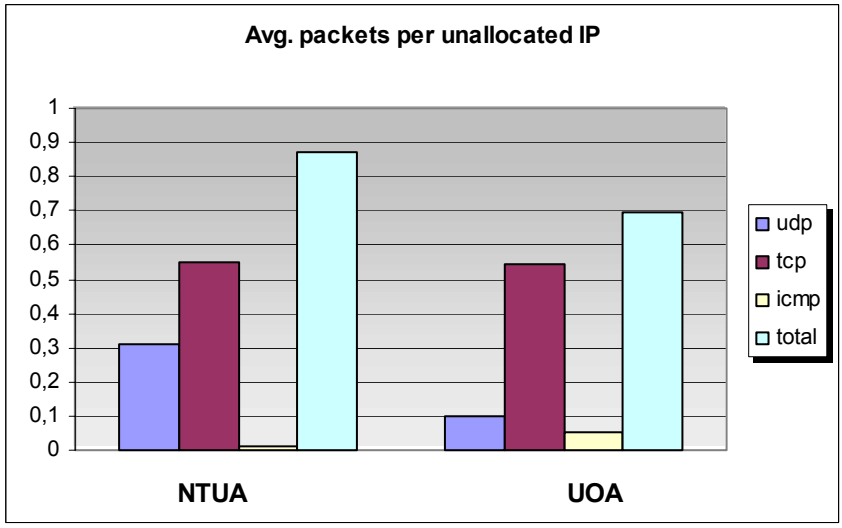
Στην προκειμένη περίπτωση, τα flows με προορισμό διευθύνσεις από το unallocated address space του UOA προέκυψαν **2.976** το πλήθος. Ένα τέτοιο νούμερο όμως φαίνεται φυσιολογικό αν το συγκρίνουμε με το μέγεθος του unallocated address space που προέκυψε για το δίκτυο του UOA, το οποίο περιέχει **4.314 IPs**.

Και στις δύο περιπτώσεις παρατηρούμε μία αναλογία της τάξεως του 0,7 – 0,8 όσον αφορά τον όγκο της "ανεπιθύμητης" κίνησης που αντιστοιχεί σε συγκεκριμένο πλήθος IP διευθύνσεων, από τα δεδομένα που συλλέξαμε για τη συγκεκριμένη μέρα.

Συλλέγοντας παρόμοια στοιχεία για την κίνηση του κάθε δικτύου ξεχωριστά, για χρονική διάρκεια τριών ημερών (3, 4 και 5/10 ), μπορούμε να δημιουργήσουμε μία σχετικά αντιπροσωπευτική εικόνα της "ανεπιθύμητης" κίνησης που δρομολογείται προς το καθένα. Από τα flags των flows που συγκεντρώνουμε, εξάγουμε πληροφορίες σχετικά με τον τύπο των πακέτων. Τα επόμενα διαγράμματα αναπαριστούν τις μέσες τιμές των δεδομένων που συλλέξαμε για τα δίκτυα των NTUA και UOA, όπου φαίνεται αναλυτικά και ο όγκος της κίνησης που αντιστοιχεί σε κάθε πρωτόκολλο μεταφοράς. Οι τιμές στους κάθετους άξονες είναι κανονικοποιημένες ως προς το συνολικό πλήθος των unallocated addresses στο οποίο αντιστοιχούν οι τιμές της μετρούμενης κίνησης. Το γεγονός αυτό διευκολύνει τη σύγκριση μεταξύ των δεδομένων που αφορούν διαφορετικά δίκτυα.


Avg. flows per unallocated IP
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
NTUA
UOA
udp
tcp
icmp
total


Avg. octets per unallocated IP
350
300
250
200
150
100
50
0
NTUA
UOA
udp
tcp
icmp
total


Avg. packets per unallocated IP
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
NTUA
UOA
udp
tcp
icmp
total

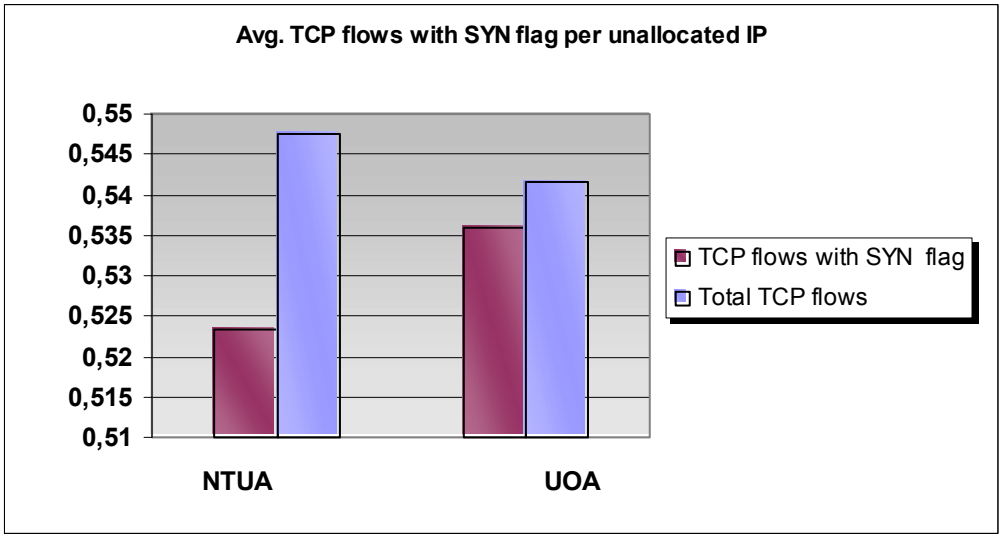
Παρατηρώντας τα παραπάνω διαγράμματα, μπορούμε να διαπιστώσουμε ότι υπάρχει μία σχετική αναλογία και σταθερότητα στο πλήθος των TCP πακέτων, που αποτελούν και τον μεγαλύτερο όγκο της κίνησης που αποστέλλεται προς τα δύο δίκτυα . Στα ICMP πακέτα παρατηρείται μία μικρή υπεροχή για το δίκτυο του UOA, ενώ όσον αφορά την κίνηση που οφείλεται σε UDP πακέτα, αυτή φαίνεται να είναι για κάποιο λόγο κατά πολύ μεγαλύτερη για το δίκτυο του NTUA, σε σχέση με αυτή που δρομολογείται προς το δίκτυο του UOA. Το γεγονός αυτό ίσως να οφείλεται σε περισσότερες απόπειρες port scanning από μηχανισμούς εξάπλωσης worms, ή αιτήσεις για εγκατάσταση σύνδεσης μέσω UDP πακέτων που επιχειρούνται προς το δίκτυο του Ε.Μ.Π. και θα μπορούσε να εξηγηθεί, αν λάβουμε υπόψη μας το μέγεθος του αντίστοιχου total address space του δικτύου, καθώς ένα μεγαλύτερο πλήθος διευθύνσεων, θα μπορούσε πιο εύκολα να προσελκύσει επιθετική κίνηση.

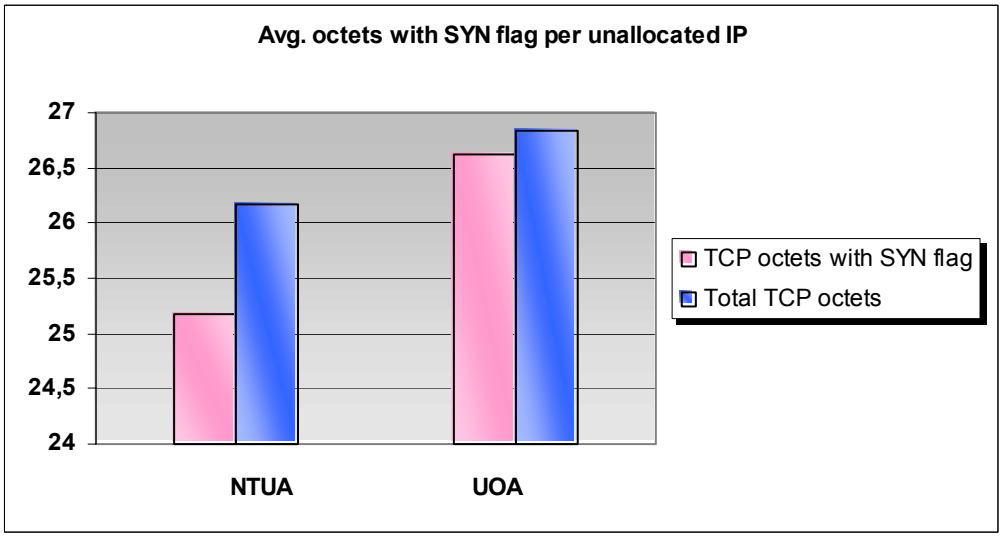
Ο επόμενος πίνακας συνοψίζει τα παραπάνω αποτελέσματα, τα οποία εμφανίζονται ως ποσοστά της κίνησης που αναλογεί σε κάθε πρωτόκολλο, αναφορικά με τη συνολική κίνηση που δρομολογείται προς κάθε δίκτυο :

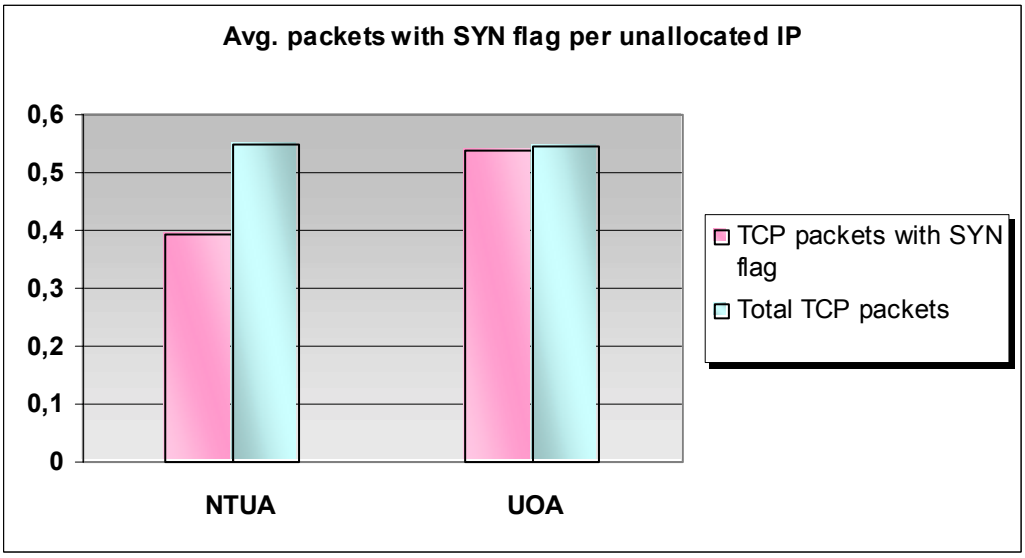
| Protocol | NTUA | UOA |
|---|---|---|
| TCP | 66,5 % | **81 %** |
| UDP | 32,1 % | **12 %** |
| **ICMP** | **1,3 %** | **7 %** |

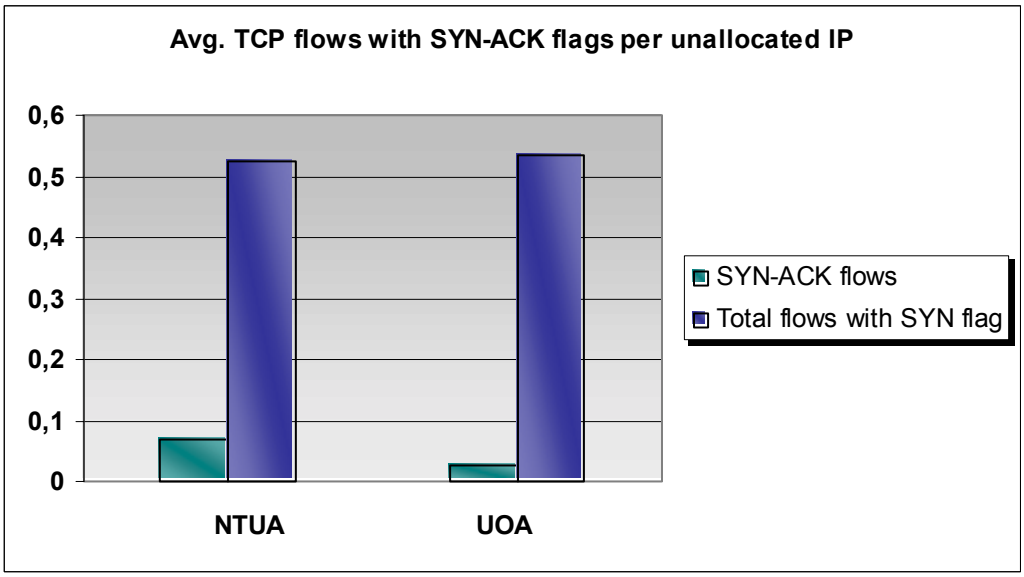
Προχωρώντας λίγο παραπέρα στην ανάλυση της συλλεγόμενης κίνησης, απομονώνουμε με τη βοήθεια κατάλληλων φίλτρων, τα TCP flows που περιέχουν SYN flags. Από τα flows αυτά μπορούμε να αντλήσουμε πληροφορίες για την κίνηση εκείνη που πιθανόν να οφείλεται είτε σε απόπειρες port scanning που έγιναν με σκοπό τον εντοπισμό ενεργών ports του δικτύου, είτε σε address spoofing λόγω DDoS επιθέσεων. Στην πρώτη περίπτωση, τα πακέτα που λαμβάνονται είναι τύπου SYN και προέρχονται από αιτήσεις που έγιναν απευθείας προς κάποιες από τις unallocated IPs προκειμένου να επιτευχθεί κάποια σύνδεση. Στη δεύτερη περίπτωση, τα πακέτα που λαμβάνονται είναι τύπου SYN-ACK και προέρχονται από κάποιο πιθανό θύμα, το οποίο αποκρίθηκε στις αιτήσεις SYN που έλαβε, με αποστολή SYN-ACK πακέτων προς τις υποτιθέμενες 'spoofed' διευθύνσεις πηγής.

Τα επόμενα διαγράμματα αντιπροσωπεύουν τον όγκο της παραπάνω κίνησης με SYN και SYN-ACK flags που περιέχεται στα δεδομένα που συγκεντρώθηκαν για τη διάρκεια των τριών ημερών (3, 4 και 5/10 ). Οι τιμές που αφορούν τον όγκο της κίνησης είναι και πάλι κανονικοποιημένες ως προς το πλήθος των IPs του unallocated address space :



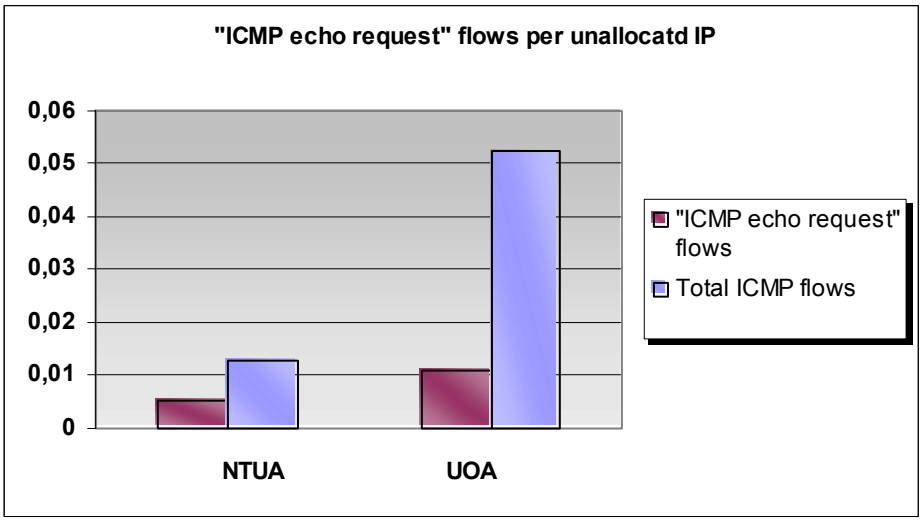



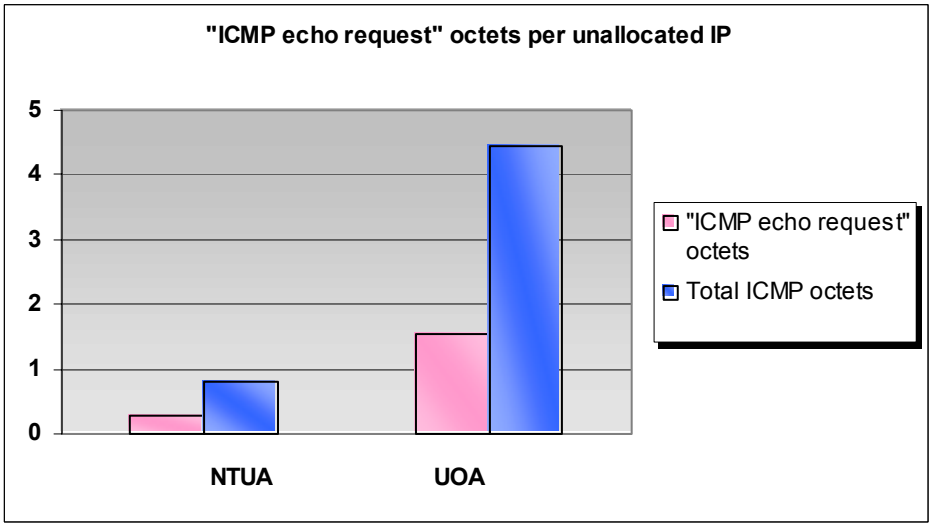


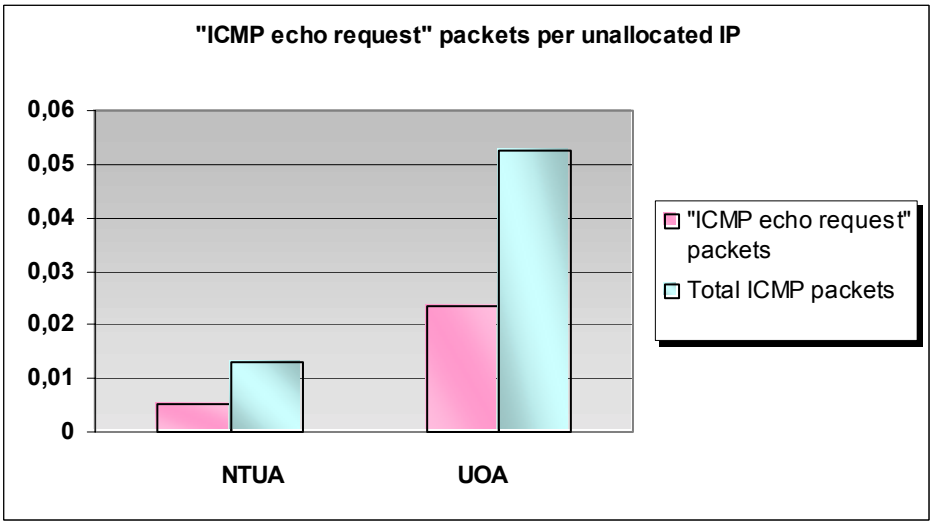




Η παρατήρηση των παραπάνω γραφημάτων μας οδηγεί στο συμπέρασμα, ότι το ποσοστό των TCP πακέτων με SYN flags που εμπεριέχονται στη συνολική TCP κίνηση, είναι λίγο μεγαλύτερο για το δίκτυο του UOA, γεγονός που σημαίνει ότι έγιναν ίσως περισσότερες απόπειρες σύνδεσης μέσω TCP πακέτων ή port scanning για τους κόμβους του δικτύου αυτού. Ωστόσο, τα πακέτα με SYN-ACK flags υπερέχουν για το δίκτυο του NTUA και μπορούν να θεωρηθούν αποτέλεσμα "address spoofing", προερχόμενου από DDoS επιθέσεις. Το γεγονός αυτό θα μπορούσε ίσως να αποδοθεί στο μεγαλύτερο εύρος διευθύνσεων που αναλογούν στο συγκεκριμένο δίκτυο, το οποίο το καθιστά πιο ελκυστικό απέναντι στην επιθετική κίνηση.

Σε παρόμοια συμπεράσματα καταλήγουμε εάν απομονώσουμε και εξετάσουμε τον αριθμό των “ICMP echo request” πακέτων που εστάλησαν προς τα δύο δίκτυα. Στην περίπτωση αυτή παρατηρούμε, πως οι προσπάθειες για εγκατάσταση σύνδεσης, port scanning ή probing προς το δίκτυο του UOA μέσω των πακέτων αυτών είναι περισσότερες, όπως φαίνεται από τα παρακάτω γραφήματα :






### 6.3.2 Παράλληλη συλλογή δεδομένων κίνησης για τα δύο δίκτυα με χρήση του νέου εργαλείου *flow-rrd-receive*

Το εργαλείο αυτό όπως έχουμε εξηγήσει, έχει διαμορφωθεί έτσι, ώστε να καθιστά δυνατή την ταυτόχρονη συλλογή και καταγραφή της δικτυακής κίνησης που προέρχεται από ξεχωριστά δίκτυα. Οι πληροφορίες σχετικά με τα flows των δικτύων αντλούνται δυναμικά και με περιοδικό τρόπο και στη συνέχεια καταγράφονται σε αντίστοιχα rrd files, τα οποία θα πρέπει να έχουμε ήδη δημιουργήσει πριν από την εκτέλεσή του.

Η δυνατότητα επιλογής των συγκεκριμένων δικτύων που μας ενδιαφέρει να μελετήσουμε δίνεται μέσω των παραμέτρων -i και –I. Τα ορίσματα των παραμέτρων αυτών καθορίζουν τους αριθμούς των interfaces που μεσολαβούν στη σύνδεση των δικτύων με τον κεντρικό δρομολογητή του ευρύτερου δικτύου, απ'όπου προέρχονται και τα δεδομένα μας. Επιπλέον, στα δεδομένα που αφορούν το κάθε interface μπορεί να εφαρμοστεί ένα ξεχωριστό φίλτρο, το όνομα του οποίου, όπως και αυτό του αρχείου στο οποίο περιέχεται, καθορίζεται μέσω των παραμέτρων -f και –F.

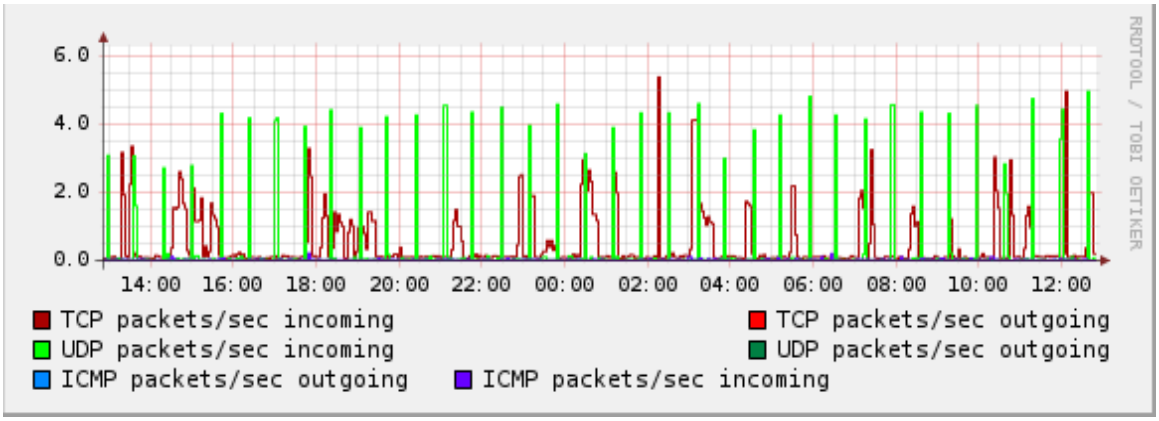
Στη συνέχεια, φαίνεται ένα παράδειγμα εκτέλεσης του **`flow-rrd-receive`**, όπου το interface 24 αντιστοιχεί στο δίκτυο του NTUA, ενώ το interface 35 αντιστοιχεί στο δίκτυο του UOA. Για το πρώτο χρησιμοποιείται το φίλτρο `'telescope_ntua'`, ενώ για το δεύτερο το `'telescope_uoa'` αντίστοιχα :

```
> flow-cat < ilissos-nf | flow-send  0/147.102.13.28/6666 |
  flow-rrd-receive/flow-rrd-receive2 –f filter.def
  -F telescope_ntua-24, telescope_uoa-35   -i -I 24,35
  -o /dev/null -s -S 30 147.102.13.28/0/6666
```

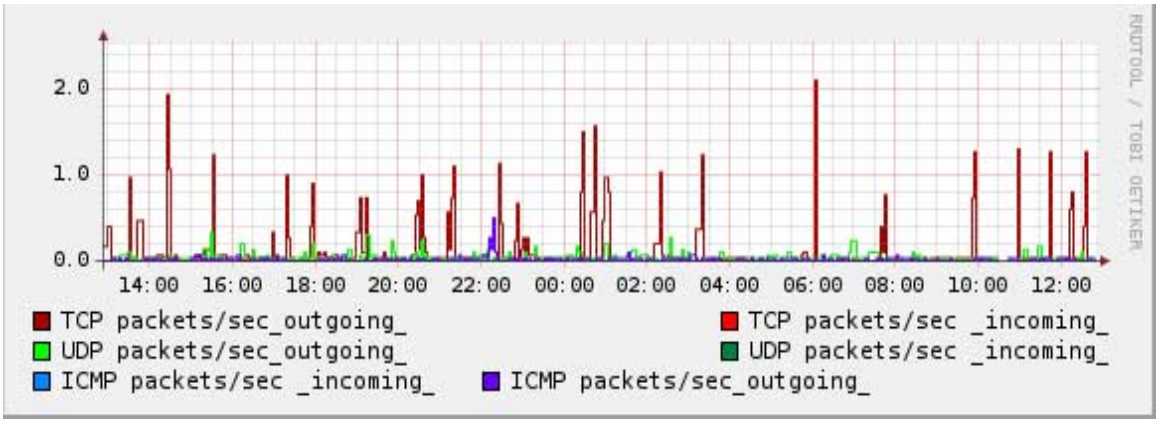
Με την εκτέλεση της παραπάνω εντολής, παίρνουμε στην οθόνη αποτελέσματα της παρακάτω μορφής που ανανεώνονται κάθε 30 δευτερόλεπτα, ενώ ταυτόχρονα οι αναλυτικές μετρήσεις καταχωρούνται στα αρχεία 'ft_online24.rrd' και 'ft_online35.rrd' για τα interfaces 24 και 35 αντίστοιχα.

```
Interface 35
Total Flows                     : 4
Total Octets                    : 192
Total Packets                   : 4
Total Time (1/1000 secs) (flows): 0
Duration of data  (realtime)    : 24
Duration of data (1/1000 secs)  : 23308
Average flow time (1/1000 secs) : 0.0000
Average packet size (octets)    : 48.0000
Average flow size (octets)      : 48.0000
Average packets per flow        : 1.0000
Average flows / second (flow)   : 0.1739
Average flows / second (real)   : 0.1667
Average Kbits / second (flow)   : 0.0668
Average Kbits / second (real)   : 0.0640


rrdupdate: 1128794940

Interface 24
Total Flows                     : 11
Total Octets                    : 1014
Total Packets                   : 11
Total Time (1/1000 secs) (flows): 0
Duration of data  (realtime)    : 71
Duration of data (1/1000 secs)  : 71788
Average flow time (1/1000 secs) : 0.0000
Average packet size (octets)    : 92.0000
Average flow size (octets)      : 92.0000
Average packets per flow        : 1.0000
Average flows / second (flow)   : 0.1549
Average flows / second (real)   : 0.1549
Average Kbits / second (flow)   : 0.1143
Average Kbits / second (real)   : 0.1143

rrdupdate: 1128794880
```

Συγκεντρώνοντας παρόμοια δεδομένα για μεγάλη χρονική διάρκεια, μπορούμε να εξάγουμε αρκετά αντιπροσωπευτικά αποτελέσματα, τα οποία αφορούν την "ανεπιθύμητη" κίνηση προς το δίκτυο που μελετάμε. Στη συνέχεια φαίνεται ένα παράδειγμα με δεδομένα κίνησης που συλλέχθηκαν για τα δίκτυα των NTUA και UOA , με χρήση του *flow-rrd-receive*. Τα γραφήματα έχουν εξαχθεί με τη βοήθεια του εργαλείου *rrdtool graph*, και δίνουν μια γενική εικόνα για την κατανομή της κίνησης ανάλογα με το είδος του πρωτοκόλλου, έτσι όπως αυτή προέκυψε για χρονική διάρκεια 1 ημέρας :




***Εικόνα 14 :*** *Κατανομή "ανεπιθύμητης" κίνησης για το δίκτυο του NTUA, για χρονική διάρκεια μίας ημέρας.*




***Εικόνα 15 :*** *Κατανομή "ανεπιθύμητης" κίνησης για το δίκτυο του UOA , για χρονική διάρκεια μίας ημέρας.*

Η παρατήρηση των παραπάνω γραφημάτων μας οδηγεί στο συμπέρασμα, ότι η θεωρητικά "κακόβουλη" κίνηση δεν εμφανίζει κάποια περιοδικότητα, αλλά έχει μία τυχαία κατανομή στο χρόνο και παρουσιάζει ανομοιομορφία. Οι διάφορες αιχμές που εμφανίζονται, χαρακτηρίζουν προφανώς τα κάθε είδους "επιθετικά" περιστατικά, που είναι συνήθως μικρής διάρκειας και εμφανίζονται στιγμιαία.

Είναι επίσης προφανές ότι ο όγκος της κίνησης είναι σαφώς μεγαλύτερος για το δίκτυο του NTUA, γεγονός το οποίο οφείλεται στο μέγεθος του unallocated address space του δικτύου, αλλά και του total address space, το οποίο περιέχει κατά πολύ περισσότερες IPs σε σχέση με το αντίστοιχο address space για το δίκτυο του UOA.

Τέλος, παρατηρούμε ότι επαληθεύονται τα συμπεράσματα που προέκυψαν από τα προηγούμενα γραφήματα σχετικά με το πλήθος των UDP πακέτων, τα οποία είναι εμφανώς περισσότερα για το δίκτυο του NTUA. Όσον αφορά επίσης την κίνηση που οφείλεται σε TCP πακέτα, παρατηρείται μία σχετική αναλογία ανάμεσα στα δύο δίκτυα και μία σταθερότητα που χαρακτηρίζει το είδος της κίνησης αυτής. Τα ICMP πακέτα εμφανίζουν και πάλι μία υπεροχή όσον αφορά το δίκτυο του UOA.

Με παρόμοιο τρόπο, μπορούμε να εφαρμόσουμε το εργαλείο χωρίς τη χρήση φίλτρων, προκειμένου να συλλέξουμε δεδομένα που αφορούν τη συνολική κίνηση που εξέρχεται ή εισέρχεται προς κάθε interface. Συγκρίνοντας μεταξύ τους τα αποτελέσματα που προκύπτουν, μπορούμε να εξάγουμε συμπεράσματα για το ποσοστό της συνολικής κίνησης που κατευθύνεται προς το unallocated address space του κάθε δικτύου, για το είδος της αλλά και για την πηγή προέλευσής της. Στη διαδικασία αυτή μας βοηθούν τιμές των μετρητών που έχουν καταχωρηθεί στα rrd files, όπως και τα διάφορα flags των flows που παρουσιάσαμε παραπάνω

## 6.4 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

Παρατηρώντας τα αποτελέσματα της πειραματικής λειτουργίας του network telescope που υλοποιήσαμε, μπορούμε να καταλήξουμε σε κάποια χρήσιμα συμπεράσματα που αφορούν τον όγκο της "επιθετικής" κίνησης που καταγράφεται από το telescope και είναι τα εξής :

- Η "ανεπιθύμητη κίνηση" που δρομολογείται προς το unallocated address space ενός δικτύου, εξαρτάται πάντα, τόσο από το μέγεθος του συνόλου αυτού, όσο και από το total address space του δικτύου, καθώς ένα μεγαλύτερο εύρος IP διευθύνσεων μπορεί θεωρητικά να προσελκύσει μεγαλύτερο όγκο κακόβουλης κίνησης.

- Η κατανομή της επιθετικής κίνησης δεν εμφανίζει κάποιου είδους περιοδικότητα στην πορεία του χρόνου, αλλά περιέχει συνήθως αιχμές που αντιστοιχούν σε διάφορα επιθετικά γεγονότα. Επίσης, μεταξύ address spaces που ανήκουν σε διαφορετικά δίκτυα, ακόμη και στην περίπτωση που αυτά έχουν το ίδιο μέγεθος, η κακόβουλη κίνηση μπορεί να εμφανίσει και πάλι τελείως διαφορετική μορφή ή χρονική κατανομή.

- Η μορφή που εμφανίζει η επιθετική κίνηση για ένα συγκεκριμένο δίκτυο, μπορεί να μεταβάλλεται δυναμικά στην πορεία του χρόνου, καθώς ταυτόχρονα μεταβάλλονται και οι διάφορες χρησιμοποιούμενες επιθετικές τεχνικές, αλλά και οι τεχνολογίες των συστημάτων ανίχνευσης και παρεμπόδισης των δικτυακών επιθέσεων που ολοένα και αναπτύσσονται.

Παράλληλα, κατά τη μελέτη των αποτελεσμάτων και προτού γενικεύσουμε τα συμπεράσματά μας, θα πρέπει να λαμβάνουμε υπόψη τους εξής παράγοντες που μπορούν να επηρεάσουν επίσης σημαντικά τα συλλεγόμενα στοιχεία κίνησης :

- Η λειτουργία ενός firewall σε ένα κεντρικό σημείο του δικτύου μπορεί να εμποδίσει κάποια από τα "ανεπιθύμητα" πακέτα να φτάσουν στον προορισμό τους, δηλαδή σε κάποια από τις διευθύνσεις τις οποίες παρατηρούμε. Έτσι, εάν ένα είδος φίλτρου μεσολαβεί στη διαδρομή από την πηγή στον προορισμό, ένα μέρος της "ύποπτης" κίνησης μπορεί να αποκλειστεί από αυτό και να μην παρατηρηθεί τελικά από το telescope.

- Είναι γεγονός ότι οι αλγόριθμοι εντοπισμού πιθανών στόχων που εφαρμόζονται από τα worms, επιλέγουν συνήθως διευθύνσεις γειτονικές των αρχικών τους στόχων, καθώς αυτό τους επιτρέπει να επιταχύνουν τη μολυσματική τους δραστηριότητα λόγω των κοντινών αποστάσεων. Οι γειτονικές λοιπόν διευθύνσεις των ευπρόσβλητων στόχων, θεωρούνται και αυτές πολύ πιθανοί στόχοι. Η παραπάνω ιδιότητα αναφέρεται ως "local preference" και μπορεί να επηρεάσει την παρατηρούμενη "κακόβουλη" κίνηση ανάλογα με τη θέση των διαχειριζόμενων διευθύνσεων.

- Η έλλειψη εύκολης πρόσβασης (reachability) ή σταθερότητας (instability) των διαδρομών που οδηγούν προς τις παρατηρούμενες διευθύνσεις, μπορεί επίσης να οδηγήσει σε λιγότερο ακριβή αποτελέσματα.

- Η διαθεσιμότητα και η λειτουργικότητα των μονοπατιών που διασυνδέουν τα επιμέρους στοιχεία του δικτύου, αποτελούν έναν ακόμη σημαντικό παράγοντα που μπορεί να παρεμποδίσει την πρόσβαση σε κάποιους προορισμούς. Η ύπαρξη μίας διαδρομής σε κάποιον πίνακα δρομολόγησης, δεν εξασφαλίζει απαραίτητα και τη διαθεσιμότητά της. Για παράδειγμα, κατά τη διάρκεια εξαπόλυσης μίας DDoS επίθεσης, τα κανάλια του δικτύου κατακλύζονται από έναν αυξανόμενο όγκο κίνησης, με αποτέλεσμα πολλά από αυτά να αποκλειστούν. Κατά συνέπεια, ο ρυθμός απόρριψης των πακέτων που καταφθάνουν στο κομμάτι αυτό του δικτύου από άλλες πηγές αυξάνει παράλληλα και έτσι οι παρατηρήσεις μπορεί να επηρεαστούν και πάλι σημαντικά.

- Τέλος, ένας ακόμη παράγοντας που πρέπει να λάβουμε υπόψη μας, είναι η τυχαιότητα του δείγματος της κίνησης που χρησιμοποιούμε για την καταγραφή δεδομένων, τα διάφορα στατιστικά σφάλματα κατά τη σύγκριση των αποτελεσμάτων, καθώς και τυχόν ανομοιομορφίες στις τεχνικές ανίχνευσης που χρησιμοποιούνται.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Matthew Todd **, "Worms as Attack Vectors: Theory, Threats, and Defenses"** A Practical Assignment, submitted in partial requirement for GSEC certification (GIAC Security Essentials Certification [GSEC], Version 1.4b, Option 1), Ph.D., January 31, 2003

[2] Nicholas Weaver, Vern Paxson, Stuart Staniford, Robert Cunningham, **"A Taxonomy of Computer Worms"**, WORM '03 October 27, 2003, Washington, DC, USA.

[3] Kevin J. Houle, George M. Weaver, in collaboration with : Neil Long, Rob Thomas,**"Trends in Denial of Service Attack Technology"**,CERT® Coordination Center, October 2001

[4] Jelena Mirkovic, Janice Martin and Peter Reiher, **"A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms"**, Computer Science Department - University of California, Los Angeles, Technical report #020018

[5] Jelena Mirkovic, Gregory Prier, Peter Reiher, **"Attacking DDoS at the Source"**, University of California Los Angeles, Computer Science Department, Los Angeles, USA, Proceedings of the 10 th IEEE International Conference on Network Protocols (ICNP'02) © 2002 IEEE

[6] Rocky K. C. Chang, **"Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial"**, The Hong Kong Polytechnic University, IEEE Communications Magazine, October 2002

[7] Anita K.Jones and Robert S.Sielken, **"Computer System Intrusion Detection: A Survey"**, Department of Computer Science, University of Virginia, Thornton Hall, Charlottesville, September 2, 2000

[8] Joseph C. Freeland (University of Notre Dame), Sally L. Goldberg (Saint Mary's College), **"IDS: What is it good for?"**, Sans 2001 University Workshop, May 4, 2001

[9] Evan Cooke, Michael Bailey, Z.Morley Mao, Danny McPherson, David Watson, Farnam Jahanian, **"Toward Understanding Distributed Blackhole Placement"**, University of Michigan, Arbor Networks, WORM '04, October 29, 2004, Washington, DC, USA,

[10] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, Larry Peterson,**"Characteristics of Internet Background Radiation"**, *IMC'04,* October 25.27, 2004, Taormina, Sicily, Italy.

[11] Vinod Yegneswaran, Paul Barford, Dave Plonka, **"On the Design and Use of Internet Sinks for Network Abuse Monitoring"**, Dept. of Computer Science, Dept. of Information Technology, University of Wisconsin, Madison

[12] David Moore, Colleen Shannon, Geoffrey M.Voelker, Stefan Savage, **" Network Telescopes : Technical Report"**, CAIDA, San Diego Supercomputer Center, Computer Science and Engineering Department, University of California, San Diego

[13] David Moore , **"Network Telescopes"**, DIMACS Large-scale Internet Attacks Workshop, September 23rd, 2003, *www.caida.org* , *www.cs.ucsd.edu*

[14] Efi Arazi, **"The IUCC/IDC Internet Telescope"**, IDC Herzliya, School of Computer Science

[15] Stefan Savage, David Moore, Geoff Voelker, and Colleen Shannon, **"The UCSD Network Telescope : A Real-time Monitoring System for Tracking Internet Attacks"**, Department of Computer Science and Engineering & Cooperative Association for Internet Data Analysis (at SDSC),University of California, San Diego

[16] David Moore, **"Network Telescopes: Observing Small or Distant Security Events",** USENIX Security - August 8th, 2002

[17] Β.Μάγκλαρης, Τ.Χιώτης, Θ.Καρούνος, Φ.Σταματελόπουλος, **"Διαχείριση Δικτύων Υπολογιστών"**, Ε.Μ.Π., ΑΘΗΝΑ 1994

[18] Yiming Gong, **"Detecting Worms and Abnormal Activities with NetFlow"**, August 16, 2004

[19] Mark Fullmer (OARnet), Steve Romig (The Ohio State University), **"The OSU Flow-tools Package and Cisco NetFlow Logs"**, USENIX Association Proceedings of the 14th Systems Administration Conference(LISA 2000), New Orleans, Louisiana, USA, December 3– 8, 2000, *http://www.usenix.org*

[20] OSPF routing protocol, DATA Connection : *http://www.dataconnection.com/iprouting/ospf.htm*

[21] OSPF Configuration Management with SNMP, Cisco Systems: *http://www.cisco.com/en/US/tech*

[22] F.Baker (ACC), R.Coltun (Computer Science Center), **"OSPF Version 2 Management Information Base – RFC 1252"**, August 1991: *http://search.cpan.org*

[23] Alex van den Bogaerdt, **"rrdtutorial"**, July 25, 2005

[24] Ketan Patel,**"rrd-beginners",** June 27, 2005

[25] Avleen Vig, **"Using SNMP and RRDTool on FreeBSD: A guide to generating server statistics"**

[26] Tobias Oetiker “RRDTool Logging and Graphing”, April 26, 2005, *http://www.rrdtool.org*

[27] National Technical University of Athens, Network Management Center , Whois Query Service : *http://www.ntua.gr/nmc/nettest/whois.html*