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

Μηχανισμοί Μετάβασης από το IPv4 στο IPv6 Πρωτόκολλο Μελέτη του 6to4 Μηχανισμού Σχεδίαση και Υλοποίηση της 6to4 MIB

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Θ. Φιλιππίδης

Επιβλέπων : Μάγκλαρης Βασίλειος
Καθηγητής Ε.Μ.Π

Αθήνα, Μάρτιος 2005

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

Μηχανισμοί Μετάβασης από το IPv4 στο IPv6 Πρωτόκολλο Μελέτη του 6to4 Μηχανισμού Σχεδίαση και Υλοποίηση της 6to4 MIB

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Θ. Φιλιππίδης

Επιβλέπων : Μάγκλαρης Βασίλειος
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Ιουλίου 2005.

.....	.....	.....
Μάγκλαρης Βασίλειος	Συκάς Ευστάθιος	Παπαβασιλείου Συμεών
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Επ. Καθηγητής Ε.Μ.Π

Αθήνα, Μάρτιος 2005

Περίληψη

Σκοπός της διπλωματικής εργασίας ήταν η μελέτη των μηχανισμών μετάβασης από την προηγούμενη έκδοση του δικτυακού πρωτοκόλλου IP, το IPv4, στο IPv6. Μελετήθηκε κυρίως ο μηχανισμός 6to4, για τον οποίο καθορίστηκε η αντίστοιχη MIB (Management Information Base - Βάση Πληροφοριών Διαχείρισης) για διαχείριση μέσω του δημοφιλούς πρωτοκόλλου SNMP (Simple Network Management Protocol). Τέλος, έγινε η υλοποίηση της MIB αυτής σε πρόγραμμα agent.
Συγκεκριμένα, αρχικά γίνεται μια εισαγωγή στο νέο IPv6 Πρωτόκολλο Δικτύου, περιγράφονται τα κύρια χαρακτηριστικά και οι λειτουργίες του και οι λύσεις που δίνει στις αδυναμίες του υπάρχοντος IPv4 πρωτοκόλλου . Εξηγούνται οι λόγοι για τους οποίους καθυστερεί η μετάβαση στο IPv6 και περιγράφονται οι διάφοροι μηχανισμοί που δημιουργήθηκαν για να διευκολύνουν τη διαδικασία μετάβασης. Εκτενέστερα αναλύονται οι μηχανισμοί tunneling, οι οποίοι είναι οι πιο διαδεδομένοι στο παρόν στάδιο μετάβασης και ειδικότερα ο 6to4 μηχανισμός, του οποίου αναλύονται η λειτουργία και τα χαρακτηριστικά. Επίσης, παρουσιάζονται και τα διάφορα θέματα ασφαλείας που τον αφορούν. Ακολούθως, καθορίζονται οι διαχειριστικές ανάγκες του 6to4 και βάσει αυτών καθορίζεται η *sixtofour*-MIB για την διαχείριση, μέσω SNMP, ενός μηχανήματος στο οποίο είναι εγκατεστημένος ο 6to4 μηχανισμός. Η MIB είναι αρκετά ευέλικτη ώστε να καλύπτει μια σειρά περιπτώσεων για το μηχάνημα αυτό: 6to4 router, relay router ή απλός 6to4 host. Τέλος, υλοποιείται σε γλώσσα προγραμματισμού C η συγκεκριμένη MIB με χρήση του NET-SNMP, για λειτουργικό σύστημα Linux.
Μαζί με την εργασία παρατίθεται εγχειρίδιο για την υλοποίηση του 6to4 μηχανισμού σε Linux (6to4 host και 6to4 router) καθώς και εγχειρίδιο λειτουργίας για το NET-SNMP έκδοσης 5.

Λέξεις Κλειδιά

6to4, μηχανισμός 6to4, 6to4 δρομολογητής, ενδιάμεσος (relay) 6to4 δρομολογητής, 6to4 site, μηχανισμοί μετάβασης, Πρωτόκολλο Δικτύου (IP), IPv6, IPv4, SNMP (Simple Network Management Protocol), MIB (Management Information Base), NET-SNMP, tunnel-τούνελ, ενθυλάκωση.

Abstract

The scope of this thesis was the study of the IPv6 Transition Mechanisms. These mechanisms have been proposed in order to allow the smooth transition from IPv4 (Internet Protocol version 4) to IPv6 (Internet Protocol version 6). The research conducted focuses on the 6to4 transition mechanism for which the corresponding MIB (Management Information Base) has been suggested to enable monitoring and management via SNMP. An appropriate MIB agent has also been implemented.

IPv6 supports a number of useful new features. This work surveys the main design and operational elements of the new protocol as well as the solutions it provides for some of the existing IPv4 problems. However, for a number of reasons, which are explained in the text, the transition to full, native IPv6 networking has so far been delayed. The main mechanisms that have been implemented to facilitate a gradual transition are described. Various tunnelling mechanisms, currently the main transition tools, are then analyzed more extensively. One of them, the 6to4 mechanism is the main object of this work and its operational characteristics and main features are further detailed. A number of security considerations are also investigated. After the administrative needs of 6to4 are determined, the sixtofour-MIB is suggested to enable management via SNMP. The proposed MIB is flexible enough to cover a number of different 6to4 configurations (namely, 6to4 router, relay router or simple 6to4 host). Finally, the MIB agent is implemented in the C programming language with the use of the NET-SNMP suite, under the Linux Operating System.

Moreover, at the end of the thesis, a tutorial for the implementation of a 6to4 router in Linux is given along with a tutorial for the version 5 of NET-SNMP.

Keywords

6to4, 6to4 mechanism, 6to4 router, relay router, 6to4 site, transition mechanisms, Internet Protocol (IP), IPv6, IPv4, SNMP (Simple Network Management Protocol), MIB (Management Information Base), NET-SNMP, tunnel, automatic tunnelling, encapsulation.

Περιεχόμενα

Πίνακας Σχημάτων

1. ΕΙΣΑΓΩΓΗ

Η ιλιγγιώδης εξέλιξη της τεχνολογίας στον τομέα των ηλεκτρονικών υπολογιστών, και ιδιαίτερα των δικτύων υπολογιστών, που σιγά-σιγά αντικαθιστούν τις υπόλοιπες παραδοσιακές μορφές επικοινωνίας, φανέρωσε την αδυναμία του υπάρχοντος πρωτοκόλλου δικτύου IPv4, να διαχειριστεί τον ολοένα αυξανόμενο αριθμό των χρηστών και των δικτύων που αποτελούν σήμερα το internet, αλλά και να εκμεταλλευτεί τον μεγάλο όγκο των νέων εφαρμογών που συνεχώς εμφανίζονται με αυξανόμενες απαιτήσεις. Για αυτόν τον λόγο, πριν 10 περίπου χρόνια (1994), η Internet Engineering Task Force (IETF) ανακοίνωσε ένα νέο Πρωτόκολλο Δικτύου (Internet Protocol), το IP νέας γενιάς (**IPng**) ή **IPv6** όπως επικράτησε, για αντικατάσταση του ήδη υπάρχοντος και παγκοσμίως καθιερωμένου IPv4.
Το νέο αυτό πρωτόκολλο δίνει λύση σε πολλά από τα προβλήματα του προκατόχου του, εκμεταλλεύεται σε μεγάλο βαθμό τις νέες εφαρμογές και γενικά προσφέρει μια άλλη δυναμική στο internet και στις δικτυακές επικοινωνίες γενικότερα. Εντούτοις η μετάβαση στο IPv6 είναι εξαιρετικά δύσκολη και σίγουρα δεν μπορεί να συντελεστεί σε μικρό χρονικό διάστημα, αν αναλογιστεί κανείς τον αριθμό των χρηστών και το μέγεθος του internet καθώς και το μικρό ποσοστό της ενημέρωσης και του μικρού βαθμού κατανόησης για το νέο πρωτόκολλο.
Για τον λόγο αυτό υλοποιήθηκαν κάποιοι μηχανισμοί με σκοπό την εύκολη και σταδιακή μετάβαση από το υπάρχον IPv4 πρωτόκολλο στο IPv6. Η πιο διαδεδομένη κατηγορία αυτών των μηχανισμών στο παρόν στάδιο μετάβασης είναι οι μηχανισμοί tunneling, για τους οποίους δίνεται μια εκτενής περιγραφή στην συγκεκριμένη εργασία, και ειδικότερα ο 6to4. Πρόκειται για ένα μηχανισμό αυτόματης δημιουργίας tunnels που δίνει τη δυνατότητα σε απομονωμένα IPv6 sites (νησίδες IPv6 στις οποίες δεν παρέχονται υπηρεσίες διασύνδεσης IPv6 από κάποιο πάροχο ISP) να επικοινωνούν τόσο μεταξύ τους όσο και με κόμβους που ανήκουν στο native IPv6 δίκτυο, την υποδομή δηλαδή που έχει δημιουργηθεί και υποστηρίζει αποκλειστικά το IPv6 πρωτόκολλο. Γίνεται λεπτομερής παρουσίαση του 6to4 μηχανισμού και αναλύονται τα χαρακτηριστικά, η λειτουργία του και τα διάφορα ζητήματα ασφαλείας που τον αφορούν.
Ακολούθως, αφού καθοριστούν οι διαχειριστικές ανάγκες του 6to4, προτείνεται η *sixtofour*-MIB (Management Information Base) για διαχείριση του μηχανισμού μέσω SNMP, η οποία στη συνεχεία υλοποιείται με τη σουίτα εφαρμογών NET-SNMP.

1.1 ΑΝΑΓΚΗ ΓΙΑ ΝΕΟ ΠΡΩΤΟΚΟΛΛΟ ΔΙΚΤΥΟΥ

Το ήδη υπάρχον Πρωτόκολλο Internet IPv4 ("RFC 791", 1981) αποδείχτηκε ένα σταθερό και δυνατό πρωτόκολλο, που κάλυψε σε πολύ μεγάλο βαθμό τις απαιτήσεις για τις οποίες είχε σχεδιαστεί και επικράτησε τελικά στο παγκόσμιο internet. Εντούτοις, εδώ και μερικά είχαν αρχίσει να διαφαίνονται σημαντικά προβλήματα στα οποία το συγκεκριμένο πρωτόκολλο αδυνατούσε να δώσει λύση . Μεγάλες εξελίξεις συντελέστηκαν στον τομέα των δικτυακών επικοινωνιών και νέες εφαρμογές ήρθαν στο προσκήνιο, τις οποίες όμως το IPv4 δεν μπορεί να υποστηρίξει και να εκμεταλλευτεί στον απαιτούμενο βαθμό. Δίνονται στη συνεχεία οι σημαντικότερες απαιτήσεις που κατέστησαν επιτακτική την ανάγκη για ένα νέο πρωτόκολλο δικτύου:

- Ο τεράστιος αριθμός των νέων διασυνδεδεμένων υπολογιστών και δικτύων, και γενικότερα η επέκταση του internet σήμερα, πέρα από κάθε αρχική προσδοκία οδήγησαν σε εξάντληση των IPv4 διευθύνσεων. Το πρόβλημα είχε αρχίσει να διαφαίνεται πριν ακόμη το 1994, οπότε ο αριθμός των δικτύων διπλασιαζόταν κάθε χρόνο. Προτάθηκαν και υλοποιήθηκαν διάφορες λύσεις για την αντιμετώπιση του προβλήματος, αρχικά η δρομολόγηση τύπου Classless (Classless Inter Domain Routing - CIDR), η οποία έδωσε κάποια παράταση ζωής στο IPv4, και αργότερα ο μηχανισμός Network Address Translation (NAT), με δίκτυα ιδιωτικών διευθύνσεων κάτω από μια ή περισσότερες παγκόσμιες διευθύνσεις. Όμως, η τελευταία λύση προκάλεσε αλλά προβλήματα όπως μείωση της απόδοσης, αδυναμία peer-to-peer επικοινωνίας και εκμετάλλευσης των σχετικών εφαρμογών, καθώς και προβλήματα ασφαλείας από άκρο σε άκρο (end-to-end security).

- Η ανάγκη για υποστήριξη εφαρμογών πολυμέσων (multimedia) και real-time μεταδόσεων δεδομένων με επιπλέον απαιτήσεις για ποιότητα υπηρεσίας (Quality of Service – QoS). Αν και θεωρητικά υπάρχει το πεδίο *TOS (Type of Service)* στην IPv4 επικεφαλίδα – αργότερα μετατράπηκε σε *DS (Differentiated Services)* -, πρακτικά δεν υλοποιείται και ουσιαστικά η όλη διαδικασία έλεγχου της ποιότητας υπηρεσίας έγκειται στον διαχωρισμό των UDP από τα TCP πακέτα, με τα προβλήματα που και αυτό επιφέρει (Layer Violation Problem) – σπάταλη υπολογιστικής ισχύος στους δρομολογητές και αδυναμία για την πιο πάνω διάκριση αν χρησιμοποιείται κρυπτογράφηση.

- Η εξάπλωση των ασύρματων δικτύων, τα οποία συνεχώς αυξάνονται με γοργούς ρυθμούς. Όμως, κινούμενοι χρηστές που υποστηρίζουν το IPv4 πρωτόκολλο δεν έχουν τη δυνατότητα να διατηρήσουν την συνοδό και την διεύθυνση τους, όταν περνούν από το ένα υποδίκτυο στο άλλο (πρόβλημα Mobility).

- Η αδυναμία των backbone routers να κρατήσουν στους πίνακες τους τον τεράστιο αριθμό δικτύων που υπάρχουν σήμερα στο internet. Σε πολλούς σήμερα συνοριακούς δρομολογητές ο αριθμός των εγγραφών ξεπερνά τις 80 με 100 000 και προβάλλει επιτακτικά η ανάγκη για καλύτερη διευθυνσιοδότηση και ιεράρχηση.

- Η ανάγκη για απλούστερο configuration. Οι IPv4 διευθύνσεις πρέπει να ανατεθούν είτε με το χέρι (manual) από τον διαχειριστή, είτε να χρησιμοποιηθεί κάποιο stateful address configuration protocol όπως το DHCP (Dynamic Host Configuration Protocol). Και το πρόβλημα γίνεται ολοένα και εντονότερο όσο οι υπολογιστές και οι υπόλοιπες συσκευές που χρησιμοποιούν το IP πρωτόκολλο αυξάνουν και τα δίκτυα μεγαλώνουν.

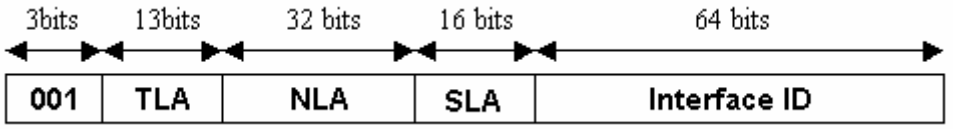
- Οι απαιτήσεις για ασφάλεια σε επίπεδο IP. Αν και υπάρχει ήδη το IPsec (Internet Protocol Security), είναι προαιρετικό και όχι ενσωματωμένο στο IPv4 πρωτόκολλο και υλοποιείται σε πολύ μικρό ποσοστό μηχανημάτων. Όμως, δεδομένης της χρήσης σήμερα του internet (εταιρίες, τράπεζες, διάφορα είδη συναλλαγών, έλεγχος συστημάτων ασφαλείας) είναι αδιαμφισβήτητο πλέον το γεγονός ότι η ασφάλεια πρέπει πλέον να εφαρμόζεται από το στρώμα δικτύου.

1.2 IPv6 ΠΡΩΤΟΚΟΛΛΟ

Λύση στα πιο πάνω προβλήματα έρχεται να δώσει το IPng (IP next generation) ή αλλιώς IPv6 Πρωτόκολλο Δικτύου, που καθορίστηκε το 1994 από την Internet Engineering Task Force (IETF).
Το νέο πρωτόκολλο επιλύει τα μεγάλα προβλήματα του IPv4 και εκμεταλλεύεται και υποστηρίζει τις νέες πιο απαιτητικές εφαρμογές. Συγκεκριμένα:

- Δίνει οριστική πλέον λύση στο πρόβλημα των διευθύνσεων. Το μέγεθος της IPv6 διεύθυνσης είναι 16 bytes (128 bit), το τετραπλάσιο δηλαδή της μέχρι τώρα χρησιμοποιούμενης IPv4 διευθύνσεως, που με ένα – αισιόδοξο- υπολογισμό αντιστοιχίζει περίπου $6x10^{20}$ διευθύνσεις σε κάθε τετραγωνικό μέτρο της επιφάνειας της γης. Έτσι, πλέον ο κάθε χρηστής μπορεί να έχει πολλαπλές IP διευθύνσεις, εκτός από τους υπολογιστές του (κινητούς και σταθερούς), στα PDAs , στο κινητό τηλέφωνο, σε τηλεοράσεις, ραδιόφωνα και γενικά σε οποία ηλεκτρική οικιακή συσκευή επιθυμεί (αν η συσκευή βέβαια παρέχει αυτήν την δυνατότητα).
  Επιπλέον, δεν υπάρχει κανένας πλέον λόγος συνέχισης της χρήσης λύσεων τύπου NAT που χρησιμοποιούνται σε μεγάλο βαθμό σήμερα.

  Η μορφή της unicast IPv6 διευθύνσεως (χαρακτηρίζεται από το format prefix 001) δίνεται πιο κάτω:



Σχήμα 1: Μορφή της unicast IPv6 διευθύνσεως

όπου το TLA ID (Top-Level Aggregation IDentifier) καθορίζεται από κάποιον οργανισμό ο οποίος παρέχει συνδεσιμότητα δικτύου σε μεγάλη κλίμακα (public transit topology), το NLA ID (Next-Level Aggregation IDentifier) το

οποίο καθορίζουν οι οργανισμοί αυτοί για τους ISPs, και τα υπόλοιπα (64+16) bit για ιεράρχηση και διευθυνσιοδότηση μέσα στο site.

- Έτσι, όπως φαίνεται και από το πιο πάνω σχήμα, καθορίζεται μια πιο λειτουργική και ιεραρχική διευθυνσιοδότηση και δρομολόγηση στο IPv6, η οποία στηρίζεται στα πολλαπλά επίπεδα παροχών ISPs και οργανισμών, και που όπως υπολογίζεται θα μειώσει κατά ένα ποσοστό 75% το σημερινό μέγεθος των πινάκων δρομολόγησης των δρομολογητών.

- Καλύτερη υποστήριξη της Ποιότητας Υπηρεσίας (QoS) με τα πεδία *Traffic Class* και *Flow Label* που έχουν προστεθεί στην IPv6 επικεφαλίδα. Το πρώτο πεδίο, καθορίζει την προτεραιότητα για το κάθε πακέτο που δρομολογείται ενώ το δεύτερο πεδίο, *Flow Label*, εισάγει την έννοια της ροής πακέτων, (πακέτα από την ίδια προέλευση προς τον ίδιο προορισμό, που απαιτούν την ίδια επεξεργασία από τον δρομολογητή), και το οποίο αναμένεται να χρησιμοποιηθεί για real-time μεταδόσεις, τόσο audio όσο και video streaming, μειώνοντας σημαντικά την καθυστέρηση (delay) στους ενδιάμεσους δρομολογητές. Αυτό οφείλεται, πρώτον στο ότι πλέον όλα τα δεδομένα που χρειάζεται ο δρομολογητής για την απαιτούμενη επεξεργασία και προώθηση του πακέτου υπάρχουν στην IPv6 επικεφαλίδα, χωρίς να χρειάζεται να 'παραβιάσει' ανώτερα στρωματά άρα και να σπαταλήσει περισσότερη υπολογιστική ισχύ, προσθέτοντας επιπλέον καθυστέρηση στο χρόνο μετάδοσης. Και το δεύτερο και σημαντικότερο, οι περισσότεροι δρομολογητές θα διατηρούν ένα είδος κρυφής μνήμης (cache) στην οποία θα 'κρατούν' τις ενέργειες που απαιτούνται για πακέτα της ίδιας ροής, χωρίς να χρειάζεται πλέον η επεξεργασία του κάθε πακέτου ξεχωριστά, μειώνοντας και με αυτόν τον τρόπο σημαντικά την καθυστέρηση.

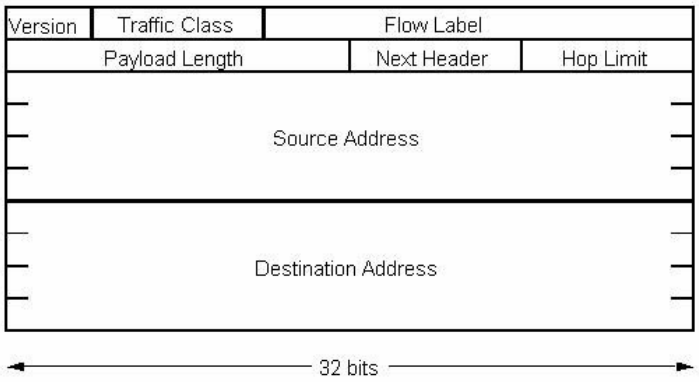
- Έχει εξ' αρχής γίνει σχεδιασμός με βάση την ασφάλεια. Το IPSec πρωτόκολλο ασφάλειας είναι ενσωματωμένο στο νέο πρωτόκολλο και αποτελεί αναπόσπαστο κομμάτι του, παρέχοντας Πιστοποίηση Αυθεντικότητας και Ακεραιότητας και Κρυπτογράφηση.

- IPv6 Mobility, παρέχεται δηλαδή τη δυνατότατα σε κινούμενους χρηστές να διατηρούν την σύνδεση τους, ενώ μετακινούνται από το ένα δίκτυο στο άλλο. Συγκεκριμένα, ο χρηστής διατηρεί την διεύθυνση του τοπικού του δικτύου (home address), και παράλληλα του αναθέτονται και οι διευθύνσεις των εκάστοτε δικτύων στα οποία εισέρχεται. Ένας δρομολογητής στο τοπικό δίκτυο, ο home agent, ενημερώνεται από τον χρηστή για τις εκάστοτε διευθύνσεις του κάθε φορά που ο τελευταίος αλλάξει δίκτυο, και στην περίπτωση που κάποιος άλλος χρηστής προσπαθήσει να επικοινωνήσει με τον κινούμενο χρηστή μέσω της τοπικής του διευθύνσεως, τα πακέτα φτάνουν στον home agent, ο οποίος τα ενθυλακώνει, και μέσω tunnel τα στέλνει στον κινούμενο χρηστή-προορισμό. Ακολούθως, ο ίδιος ο χρηστής ενημερώνει τον κόμβο προελεύσεως (ο κόμβος που του έστειλε τα πακέτα) για την προσωρινή διεύθυνση του και η επικοινωνία γίνεται κατευθείαν μεταξύ των δυο, χωρίς την μεσολάβηση του home agent. Ο κινούμενος χρηστής, κάθε φορά που αλλάζει διεύθυνση, εκτός από τον home agent ενημερώνει και τους υπόλοιπους χρηστές με τους οποίους είχε ή έχει επικοινωνία, αλλά

παράλληλα διατηρεί και τις προηγούμενες του διευθύνσεις για να δέχεται και τα πακέτα που καταφτάνουν σε αυτές.

- Παρέχει την δυνατότητα τόσο για Stateful Address Configuration, με την παρουσία ενός DHCPv6 server, όσο και για Stateless Address Configuration, μόνο από τις διαφημίσεις του δρομολογητή (router advertisements). Συγκεκριμένα, οι διάφοροι κομβόι του δικτύου καθορίζουν αρχικά τις link-local διευθύνσεις τους από το link-local πρόθεμα FE80::/64 και τον interface identifier που καθορίζουν οι ίδιοι, και με ένα μήνυμα Neighbor Discovery, ελέγχουν την μοναδικότητα της διευθύνσεως τους (duplicate address detection). Ακολούθως, στην περίπτωση που δεν έχουν πάρει κάποιο μήνυμα router-advertisement, στέλνουν οι ίδιοι στον router μηνύματα router-solicitations, απαιτώντας router-advertisement. Αν δεν υπάρχει router, χρησιμοποιείται κάποιο stateful address autoconfiguration πρωτόκολλο. Στην αντίθετη περίπτωση (ύπαρξη δρομολογητή), ο δρομολογητής άπαντα με κάποιο router-advertisement και από το πρόθεμα που διαφημίζεται σχηματίζεται η παγκόσμια και η site-local διεύθυνση του κόμβου και τίθενται οι παράμετροι hop limit, reachable time, retransmission time, MTU και καθορίζεται το default route προς τον δρομολογητή που έστειλε τις διαφημίσεις.

- Νέα διαμόρφωση της επικεφαλίδας, όπου χρησιμοποιούνται τα αυστηρώς απαραίτητα πεδία, χωρίς να υπάρχει η επιβάρυνση αχρησιμοποίητων πεδίων, όπως το *options* της IPv4 επικεφαλίδας που δεν χρησιμοποιούνταν στις περισσότερες περιπτώσεις, αλλά ο δρομολογητής έπρεπε να το ελέγχει. Όλα τα επιπλέον πεδία συμπεριλαμβάνονται στις δευτερεύουσες επικεφαλίδες που ακολουθούν σαν προέκταση της πρώτης, και η ύπαρξη τους δηλώνεται στο πεδίο *Next Header*. Έτσι, η επικεφαλίδα του IPv6 πακέτου έχει μόλις το διπλάσιο μέγεθος της IPv4 επικεφαλίδας, αν και το μήκος της IPv6 διευθύνσεως είναι τετραπλάσιο από το μήκος της αντίστοιχης IPv4. Συγκεκριμένα, στην επικεφαλίδα υφίστανται μόνο τα πεδία (1)*version*, όπου καθορίζεται αν πρόκειται για IPv4 ή IPv6 πακέτο, (2) *Traffic Class*, (3) *Flow Label* που εξηγούνται πιο πάνω, (4) *Next Header*, που υποδηλώνει αν κάποιες επιπλέον extension headers υπάρχουν μετά την επικεφαλίδα, (6) *Hop Limit*, που είναι το αντίστοιχο του TTL στο IPv4, δηλαδή μετά από πόσους ενδιάμεσους κόμβους θα απορριφθεί το πακέτο, (7) η διεύθυνση πηγής και (8) η διεύθυνση προορισμού. Πιο κάτω φαίνεται η IPv6 επικεφαλίδα:



Σχήμα 2: IPv6 επικεφαλίδα

- Απλοποίηση της διαχείρισης και του configuration του IPv6 multicast, παρέχοντας επιπλέον και καλύτερη κλιμάκωση (scaling) με χρήση του Rendezvous Point (PR).

- Καλύτερες προϋποθέσεις για multihoming, λόγω των πολλαπλών unicast διευθύνσεων ανά interface, της χρήσης των site-local διευθύνσεων εντός του site και των πλέον καθορισμένων και διαχωρισμένων TLAs για τον κάθε ISP.

2. ΜΗΧΑΝΙΣΜΟΙ ΜΕΤΑΒΑΣΗΣ
(TRANSITION MECHANISMS)

Η μετάβαση από το IPv4 πρωτόκολλο στο IPv6 δεν είναι εύκολη υπόθεση - ιδιαίτερα αν αναλογιστεί κανείς το μέγεθος του internet σήμερα , τον τεράστιο αριθμό των χρηστών και sites. Πολλές εταιρίες και οργανισμοί σήμερα στηρίζονται αποκλειστικά στο internet για την λειτουργία τους και παροχή συνεχών υπηρεσιών, κάτι που καθιστά αδύνατη μια μετάβαση στο IPv6 με κατέβασμα των συστημάτων τους έστω και για λίγο για να γίνει η αναβάθμιση τους στο IPv6 πρωτόκολλο. Επιπλέον, η τεραστία επένδυση που έχει γίνει στο IPv4, και ιδιαίτερα σε μηχανήματα που δεν μπορούν να αναβαθμιστούν όπως κινητά τηλεφωνά και δικτυακοί εκτυπωτές στέκεται εμπόδιο στην άμεση μετάβαση. Τέλος, ίσως ο σημαντικότερος παράγοντας επιβράδυνσης της μετάβασης, είναι η μέχρι στιγμής μικρή κατανόηση του IPv6 πρωτοκόλλου, των δυνατοτήτων του αλλά και των απαιτήσεων του. Η ανάγκη για μετάβασης στο νέο αυτό πρωτόκολλο το οποίο θα λύσει πολλά από τα προβλήματα του προκατόχου του δεν έχει γίνει κατανοητή από όλους. Όλα τα πιο πάνω δείχνουν ότι η μετάβαση δεν μπορεί να γίνει από τη μια στιγμή στην άλλη, ούτε υπάρχει κάποια "μέρα-D" μέχρι την οποία θα έχει γίνει η μετάβαση. Η μόνη λύση για το πέρασμα στο νέο πρωτόκολλο είναι κάποιοι μηχανισμοί για σταδιακή και "ανώδυνη" μετάβαση.

Το γεγονός αυτό είχε ληφθεί υπόψη από την IETF κατά τον σχεδιασμό του πρωτοκόλλου και έτσι το IPv6 είναι κατά πολύ μεγάλο μέρος συμβατό με το IPv4.Επιπλέον είχε συσταθεί μια ειδική ομάδα ερευνών για να ασχοληθεί αποκλειστικά με τα θέματα της μετάβασης. Η έρευνα που επιτελούνταν στο θέμα της μετάβασης είχε ως τελικό σκοπό κάποιους μηχανισμούς που να προσφέρουν:

- **Δυνατότητα σταδιακής μετάβασης**, έτσι ώστε IPv4 hosts και routers του κάθε δικτύου, να μπορούν να αναβαθμιστούν σε IPv6 ο καθένας ξεχωριστά, χωρίς την απαίτηση να έχουν και οι υπόλοιποι κομβόι του δικτύου εγκαταστήσει το IPv6 πρωτόκολλο.
- **Ελάχιστες απαιτήσεις αναβάθμισης**, και συγκεκριμένα το μόνο που απαιτείται είναι ένας DNS server που να χειρίζεται IPv6 εγγραφές στην περίπτωση των hosts, και καμία απαίτηση στην περίπτωση των routers.
- **Απλότητα διευθυνσιοδότησης**, και συγκεκριμένα δυνατότητα να διατηρηθούν οι IPv4 διευθύνσεις που είχε το μηχάνημα πριν την αναβάθμιση παράλληλα με τις IPv6 διευθύνσεις, χωρίς την ανάγκη επαναπροσδιορισμού νέων διευθύνσεων.
- **Καμία προεργασία εγκατάστασης** να μην χρειάζεται για την αναβάθμιση από IPv4 σε IPv6.

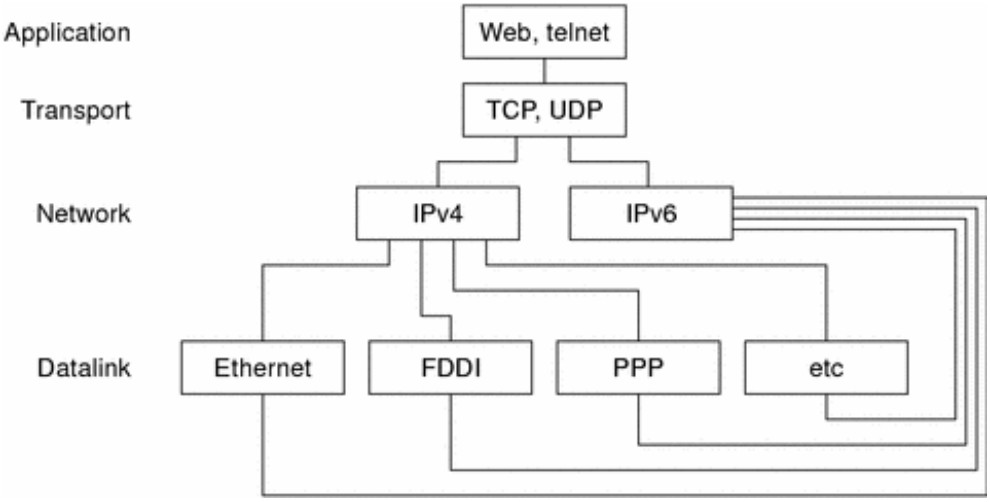
Για αυτούς τους λόγους, υλοποιήθηκε και ενσωματώθηκε στο νέο πρωτόκολλο ένα σύνολο μηχανισμών, το **SIT** (Simple Internet Transition), που περιλαμβάνει κάποιους κανόνες και πρωτόκολλα, για την διευκόλυνση της μετάβασης. Συγκεκριμένα, το SIT παρέχει:

- Μια δομή IPv6 διευθύνσεων που μπορεί να προκύψει από τις IPv4 διευθύνσεις. Οι διευθύνσεις αυτές είναι οι IPv6 IPv4-compatible διευθύνσεις της μορφής ::ww.xx.yy.zz όπου ww.xx.yy.zz η IPv4 διεύθυνση που είχε πριν την αναβάθμιση ο συγκεκριμένος κόμβος.
- Την δυνατότητα λειτουργίας των λειτουργικών συστημάτων με διπλή στοίβα πρωτοκόλλων (dual stack) ταυτόχρονα. Δηλαδή, το ένα πρωτόκολλο δεν επεμβαίνει στην λειτουργία του αλλού, και το κάθε μηχάνημα, συνήθως αναλόγως του αποτελέσματος της αναζήτησης DNS, επιλέγει ποια από τις δυο στοίβες θα χρησιμοποιήσει για επικοινωνία. Τα στρωματά ανωτέρου επιπέδου συνεργάζονται και με τα δυο πρωτόκολλα.
- Ένα μηχανισμό για την ενθυλάκωση των IPv6 πακέτων μέσα σε IPv4 πακέτα (tunneling) για μετάδοση τους πάνω από σύννεφα IPv4. Οι μηχανισμοί αυτοί είναι οι πλέον χρησιμοποιούμενοι σήμερα και θα αναλυθούν εκτενέστερα στις επόμενες ενότητες.
- Προαιρετικά, δυνατότητα μετατροπής του IPv6 πακέτου σε IPv4 και αντίστροφα.

Βάση των πιο πάνω δυνατοτήτων που παρέχει το SIT, προέκυψαν σε αυτό το διάστημα συνύπαρξης των δυο πρωτοκόλλων αρκετοί μηχανισμοί μετάβασης, που μπορούν να ταξινομηθούν στις τρεις πιο κάτω κατηγορίες:

- Μηχανισμοί Dual Stack (διπλής στοίβας)
- Μηχανισμοί Tunneling
- Μηχανισμοί Translation

2.1 Μηχανισμοί Dual Stack



Σχήμα 3: Μηχανισμοί Dual-Stack

Οι μηχανισμοί αυτοί είναι απλοί στην υλοποίηση τους, η οποία έγκειται απλώς στην εγκατάσταση και των δυο πρωτοκόλλων στα λειτουργικά συστήματα των μηχανημάτων του δικτύου.
Έτσι τα μηχανήματα στα οποία υπάρχει εγκατεστημένο και το IPv4 και το IPv6 πρωτόκολλο μπορούν να λάβουν και να προωθήσουν πακέτα και από τα δυο πρωτοκολλά. Η επιλογή της στοίβας η οποία θα χρησιμοποιηθεί γίνεται κυρίως βάση του αποτελέσματος της DNS αναζήτησης. Δηλαδή, αν ο κόμβος με τον οποίο θα γίνει επικοινωνία έχει καταγραμμένη μόνο IPv6 διεύθυνση, χρησιμοποιείται η IPv6 στοίβα, αν υπάρχει μόνο IPv4 διεύθυνση χρησιμοποιείται η IPv4 στοίβα και στην περίπτωση που υπάρχουν εγγραφές τόσο IPv4 όσο και IPv6 διευθύνσεων, η default συμπεριφορά είναι να χρησιμοποιηθούν οι IPv6. Σε αυτό το σημείο πρέπει να τονιστεί ότι για να εγγράψει ο κόμβος την IPv6 σύνδεση του, πρέπει με κάποιο τρόπο να του παρέχεται IPv6 συνδεσιμότητα (είτε native είτε μέσω tunnel), εκτός από το να έχει απλώς εγκαταστήσει το IPv6 πρωτόκολλο. Και αυτή ακριβώς είναι κυρίως η αδυναμία της συγκεκριμένης κατηγορίας μηχανισμών στον παρόν στάδιο μετάβασης. Δηλαδή, απομονωμένοι dual-stack κόμβοι στους οποίους δεν παρέχεται IPv6 σύνδεση, δεν μπορούν να επικοινωνήσουν μεταξύ τους και σε αυτές τις περιπτώσεις είναι απαραίτητη η συνεργασία της συγκεκριμένης κατηγόριας μηχανισμών με κάποιο άλλο μηχανισμό μετάβασης, όπως για παράδειγμα με τους μηχανισμούς tunneling, η οποία είναι μια τεχνική που χρησιμοποιείται κατά κόρον σήμερα. Λογικά όμως, σε μεταγενέστερα σταδία μετάβασης, αυτή η κατηγόρια θα χρησιμοποιηθεί περισσότερο.

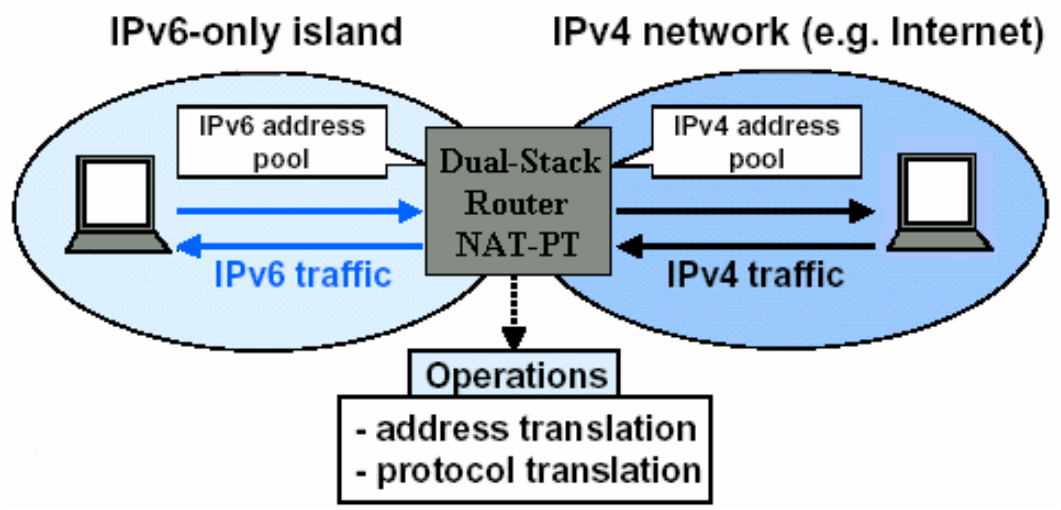
2.2 Μηχανισμοί Translation

Οι μηχανισμοί αυτοί επιτρέπουν την επικοινωνία μεταξύ κόμβων που υποστηρίζουν μόνο IPv4 (IPv4-only) και κόμβων που υποστηρίζουν μόνο IPv6 (IPv6-only), και ουσιαστικά ‘μεταφράζουν’ την κίνηση από το ένα πρωτόκολλο στο άλλο. Αναμένεται ότι θα χρησιμοποιηθούν αρκετά στα τελευταία σταδία μετάβασης, και κυρίως για τις περιπτώσεις μηχανημάτων που δεν μπορούν να αναβαθμιστούν στο IPv6.

Οι κυριότεροι μηχανισμοί αυτής της κατηγορίας είναι ο **NAT-PT** (Network Address Translation – Protocol Translation) ο οποίος χρησιμοποιεί SIIT (Stateless IP ICMP Translation) αλγόριθμους για μετατροπή των IPv6 πακέτων σε IPv4 και αντίστροφα και οι **BIS** (Bump In The Stack) και **BIA** (Bump in the API) οι οποίοι εκτελούν τις ίδιες διαδικασίες, αλλά ο ένας στο επίπεδο μεταφοράς και ο τελευταίος στο επίπεδο εφαρμογής.

2.2.1 NAT-PT

Ο NAT-PT ((Network Address Translation – Protocol Translation) μηχανισμός, ο πλέον διαδεδομένος και καθιερωμένος της κατηγορίας των translation mechanisms , στηρίζεται σε λειτουργία παρόμοια με αυτή των κλασσικών NATs.

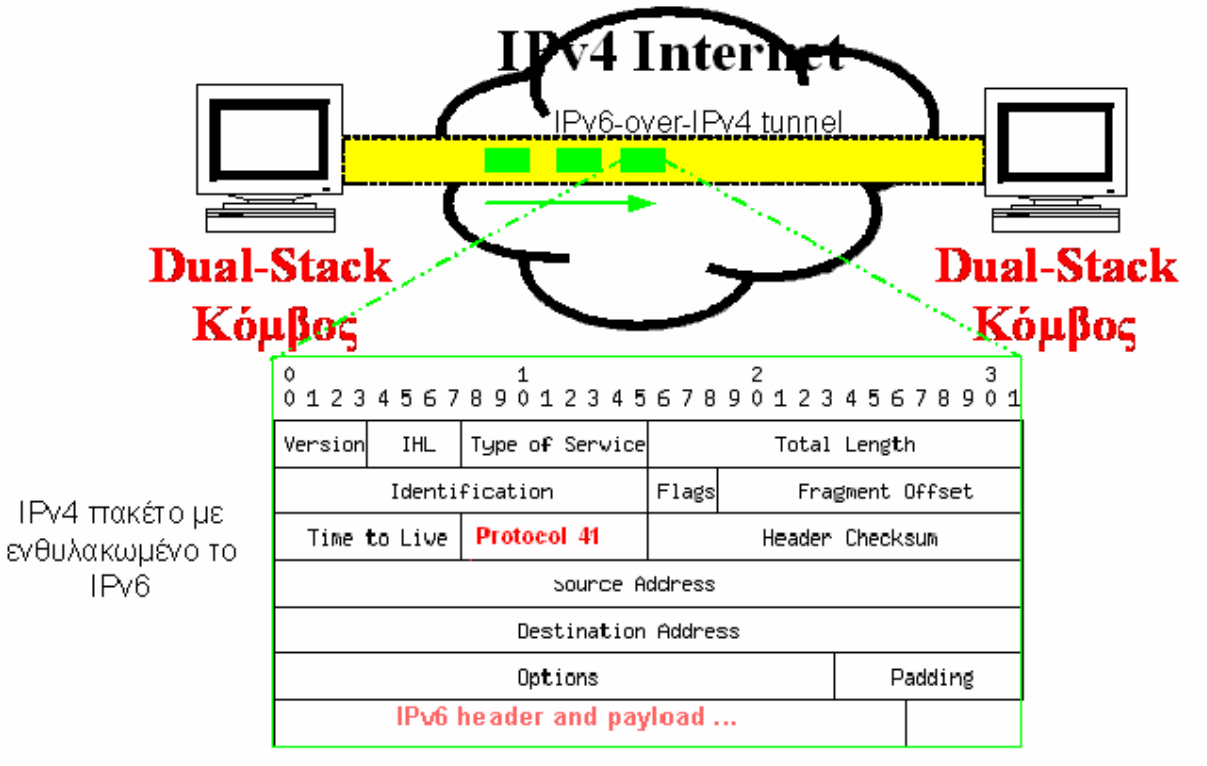


Σχήμα 4: Ο NAT-PT μηχανισμός μετάβασης

Υπάρχει ο δρομολογητής στο σύνορο μεταξύ του IPv6 και του IPv4 δικτύου, στον οποίο εγκαθίσταται το NAT-PT, ο οποίος διατηρεί δυο δεξαμενές (pools) διευθύνσεων, μια με IPv6 διευθύνσεις και μια με IPv4, τις οποίες αντιστοιχεί δυναμικά για να δημιουργηθεί η συνοδός για επικοινωνία μεταξύ του IPv4 host με τον IPv6. Επιπλέον όμως, παράλληλα με την μετάφραση των διευθύνσεων, γίνεται και μετάφραση της IPv6 επικεφαλίδας σε IPv4 και αντίστροφα. Αυτό φυσικά, προσθέτει στο συγκεκριμένο μηχανισμό και το μειονέκτημα ότι κάποια services που προσφέρει το ένα πρωτόκολλο αλλά όχι και το άλλο είναι πιθανό να χαθούν κατά την ‘μετάφραση’ αφού οι επικεφαλίδες των δυο πρωτοκόλλων δεν είναι ίδιες. Επιπλέον, οι μηχανισμοί αυτοί φέρουν και τα μειονεκτήματα των κλασικών NATs, λόγω της ‘μετάφρασης’ διευθύνσεων και πρωτοκόλλου που εκτελείται. Και σε αυτή την περίπτωση ισχύει η “best effort” προσέγγιση, και επιπλέον η απ’ άκρου σ’ άκρου ασφάλεια σε επίπεδο δικτύου είναι αδύνατη. Για αυτόν ακριβός τον λόγο, οι συγκεκριμένοι μηχανισμοί πρέπει να χρησιμοποιούνται μόνο στην περίπτωση που υπάρχει ανάγκη επικοινωνίας IPv4-only και IPv6-only hosts, και όχι σε άλλες περιπτώσεις.

2.3 ΜΗΧΑΝΙΣΜΟΙ TUNNELING

Για όσο διάστημα διευρύνεται η χρήση του IPv6 είναι χρήσιμο αλλά και αναγκαίο, τουλάχιστον στο μεταβατικό στάδιο, η IPv4 υποδομή να παραμείνει λειτουργική και να χρησιμοποιηθεί επίσης στην μετάδοση φορτίου. Η πιο διαδεδομένη τεχνική, με την οποία γίνεται η εκμετάλλευση της υποδομής IPv4 για την μεταφορά IPv6 πακέτων είναι οι μηχανισμοί tunneling, η τρίτη και σημαντικότερη κατηγορία των μηχανισμών μετάβασης. Το tunneling γενικότερα είναι μια τεχνική που χρησιμοποιείται σε μεγάλο βαθμό σήμερα και στηρίζεται στην ενθυλάκωση ενός πρωτοκόλλου ενός δικτύου σε πακέτα αλλού πρωτοκόλλου για μετάδοση τους πάνω από άλλο δίκτυο (παραδείγματα το GRE και το PPTP πρωτόκολλα που χρησιμοποιούνται σήμερα σε μεγάλο βαθμό). Η τεχνική αυτή χρησιμοποιείται και στο μεταβατικό στάδιο από το IPv4 στο IPv6.Πιο συγκεκριμένα, IPv6 hosts και routers έχουν την δυνατότητα να ανταλλάσσουν IPv6 πακέτα, τα οποία ενθυλακώνουν μέσα σε IPv4 πακέτα, μεταδίδοντας τα έτσι πάνω από το ήδη υπάρχον δίκτυο (internet). Με αυτό τον τρόπο οι ενδιάμεσοι δρομολογητές, αν και δεν υποστηρίζουν το IPv6 πρωτόκολλο, προωθούν τα ενθυλακωμένα πακέτα σαν να πρόκειται για κανονικά IPv4 πακέτα. Για την λειτουργία των μηχανισμών tunneling πρέπει οι δυο κομβόι στα άκρα του tunnel - οι κομβόι που κάνουν την ενθυλάκωση και την απενθυλάκωση - να είναι dual-stack, να έχουν δηλαδή εγκατεστημένες και τις δυο στοίβες πρωτοκόλλων IPv4 και IPv6. Πιο κάτω φαίνεται η διάδοση των ενθυλακωμένων IPv6 πακέτων, κάτω από IPv4 επικεφαλίδα, πάνω από την IPv4 υποδομή, καθώς και η μορφή των πακέτων που μεταδίδονται.



Σχήμα 5: Μηχανισμοί tunneling.

2.3.1 Ορολογία:

IPv4-only node: Ένας κόμβος που υλοποιεί μόνο IPv4 και έχει μόνο IPv4 διεύθυνση. Ο κόμβος αυτός δεν υποστηρίζει IPv6.

IPv6-only node: Ένας κόμβος που υποστηρίζει μόνο IPv6 και μπορεί να επικοινωνήσει μόνο με IPv6 κόμβους και εφαρμογές.

IPv6/IPv4 node: Κόμβος που υλοποιεί τόσο IPv6 όσο και IPv4.

IPv4 node: Κόμβος που υλοποιεί IPv4 (στέλνει και λαμβάνει IPv4 πακέτα). Ο κόμβος αυτός μπορεί να είναι IPv4-only ή IPv6/IPv4 κόμβος.

IPv6 node : Κόμβος που υλοποιεί IPv6 (στέλνει και λαμβάνει IPv6 πακέτα). Ο κόμβος αυτός μπορεί να είναι IPv6-only ή IPv6/IPv4 κόμβος.

2.3.2 Είδη μηχανισμών tunneling:

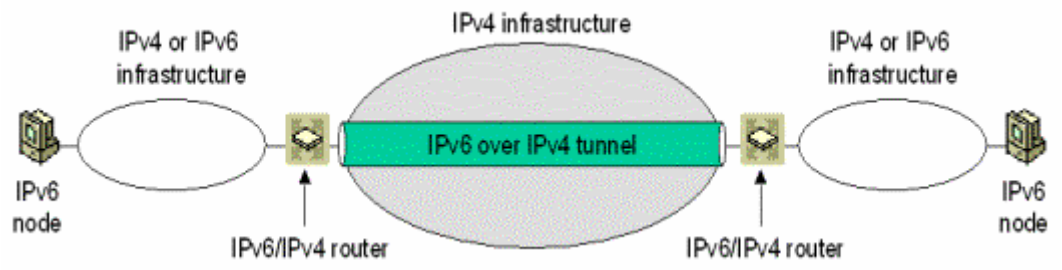
Οι μηχανισμοί tunneling κατηγοριοποιούνται, ως προς τον μηχανισμό με τον οποίο ο κόμβος εισόδου, ο οποίος πραγματοποιεί την ενθυλάκωση, καθορίζει την διεύθυνση του κόμβου εξόδου(άλλο άκρο του τούνελ), σε μηχανισμούς configured tunnelling και automatic tunnelling.

Στο **configured tunneling** η IPv4 διεύθυνση του άκρου του τούνελ, καθορίζεται από τις configuration πληροφορίες που έχουν οριστεί εκ των προτέρων στον κόμβο στον οποίο γίνεται η ενθυλάκωση των πακέτων από τον διαχειριστή του. Για το κάθε τούνελ που υλοποιεί ο συγκεκριμένος κόμβος πρέπει να υπάρχει αποθηκευμένη η διεύθυνση του αλλού άκρου του τούνελ. Όταν ένα πακέτο μεταδίδεται μέσω του τούνελ, η διεύθυνση του άλλου άκρου –άκρο εξόδου-, η οποία έχει γίνει configure για το συγκεκριμένο τούνελ, τίθεται ως η διεύθυνση προορισμού στην IPv4 επικεφαλίδα που ενθυλακώνει το IPv6 πακέτο.

Στο **automatic tunneling** η IPv4 διεύθυνση του άκρου του τούνελ, μπορεί να προκύψει από την IPv6 διεύθυνση προορισμού του πακέτου που πρόκειται να μεταδοθεί μέσω του τούνελ. Αυτή μπορεί να είναι είτε 6to4 διεύθυνση (επόμενη ενότητα "6to4 Μηχανισμός") είτε IPv4-compatible διεύθυνση (0:0:0:0:0:0:IPv4 Address) είτε γενικά διεύθυνση στην οποία η IPv4 εμπεριέχεται με κάποιο τρόπο στην IPv6 διεύθυνση. Στην περίπτωση αυτή οι IPv6/IPv4 κομβόι επικοινωνούν πάνω από την IPv4 υποδομή δρομολόγησης χωρίς να χρειάζεται να έχουν γίνει εκ των προτέρων configured τα τούνελ που θα χρησιμοποιηθούν.

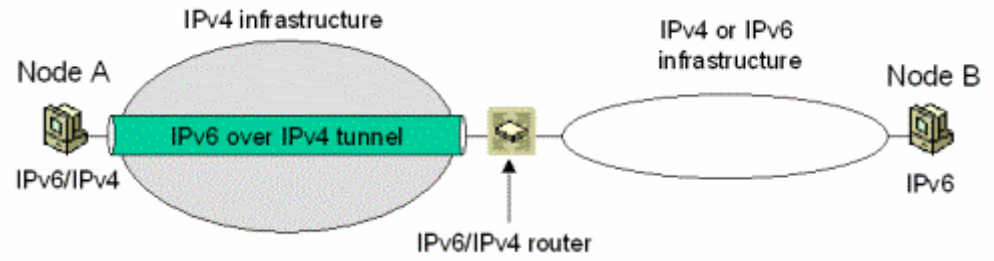
Στην πρώτη κατηγορία tunneling συγκαταλέγονται κυρίως οι περιπτώσεις επικοινωνίας:

i. Router-to-Router



Σχήμα 6: Περίπτωση επικοινωνίας router-to-router

ii. Host-to-Router

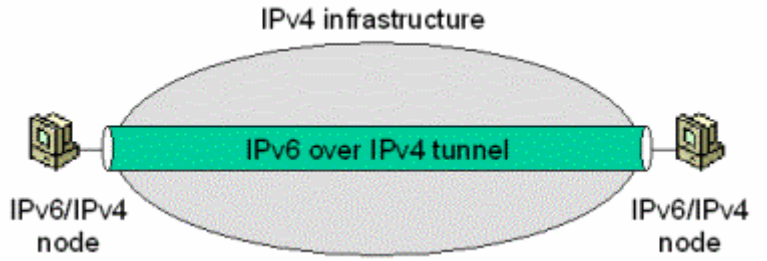


Σχήμα 7: Περίπτωση επικοινωνίας host-to-router

όπου τα πακέτα προωθούνται σε ένα ενδιάμεσο δρομολογητή (router) όπου θα γίνει η απενθυλάκωση τους. Σε αυτή την περίπτωση η διεύθυνση του άκρου του τούνελ είναι διαφορετική από την τελική διεύθυνση προορισμού, άρα απαιτείται η χρήση configured tunneling, αφού η διεύθυνση του router (άκρο του τούνελ) δεν μπορεί να προκύψει από την διεύθυνση προορισμού του πακέτου.
Εξαίρεση σε αυτό παρουσιάζει ο 6to4 μηχανισμός και γενικά οι μηχανισμοί όπου ο μεταβατικός μηχανισμός υλοποιείται στον συνοριακό δρομολογητή του site (από την διεύθυνση του οποίου καθορίζονται οι διευθύνσεις του site), οι οποίοι αν και συγκαταλέγονται στην Router-to-Router περίπτωση επικοινωνίας, μπορεί από την διεύθυνση προορισμού να υπολογιστεί η διεύθυνση του router που θα εκτελέσει την απενθυλάκωση, και έτσι οι μηχανισμοί αυτοί εντάσσονται στην κατηγόρια των automatic tunnelling.

Στην δεύτερη κατηγορία ανήκουν οι περιπτώσεις:

i. Host-to-Host



Σχήμα 8: Περίπτωση επικοινωνίας host-to-host

ii. Router-to-Host

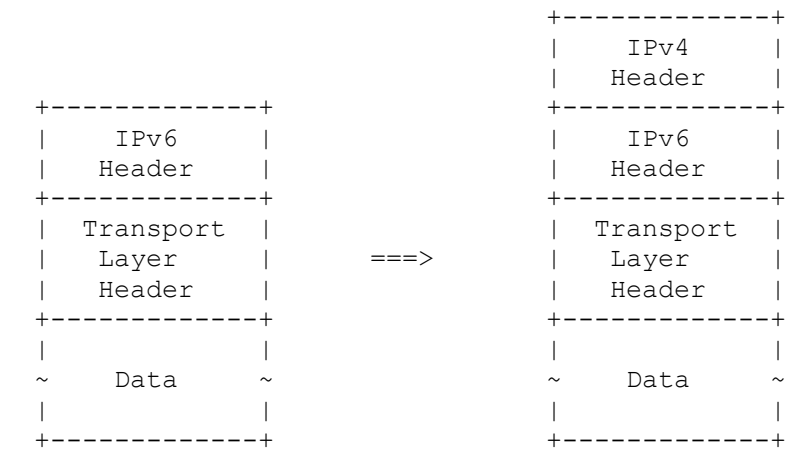
όπου το πακέτο μεταδίδεται μέσω του τούνελ μέχρι τον τελικό προορισμό. Σε αυτή την περίπτωση, η διεύθυνση προορισμού του IPv6 πακέτου περιέχει με κάποιο τρόπο την IPv4 διεύθυνση προορισμού που θα χρησιμοποιηθεί στην εξωτερική IPv4 επικεφαλίδα, έτσι δεν χρειάζεται να έχει καθοριστεί εκ των πρότερων το άλλο άκρο του τούνελ, απλοποιώντας σημαντικά την διαδικασία υλοποίησης του τούνελ.

2.3.3 Λειτουργία των μηχανισμών tunneling :

Τα κύρια χαρακτηριστικά λειτουργίας των μηχανισμών είναι ίδια και για τις δυο κατηγορίες tunnels :

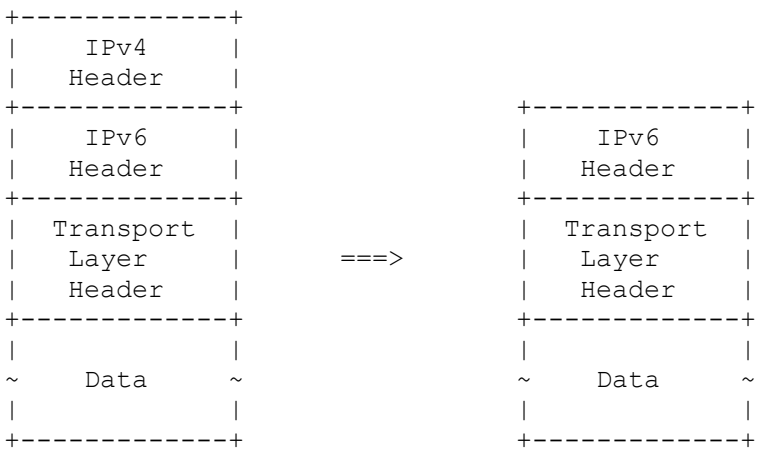
- Ο κόμβος εισόδου (ο κόμβος στον οποίο θα γίνει η ενθυλάκωση) δημιουργεί την IPv4 επικεφαλίδα με τιμή 41 στο πεδίο protocol, κάτω από την οποία ενθυλακώνει το IPv6 πακέτο, και ακολούθως δρομολογεί το πακέτο με κανονική IPv4 δρομολόγηση, θέτοντας ως διεύθυνση προορισμού την IPv4 διεύθυνση του κόμβου εξόδου, όπου θα γίνει η απενθυλάκωση.
- Στο άλλο άκρο του τούνελ, ο κόμβος εξόδου (ο κόμβος στον οποίο θα γίνει η απενθυλάκωση), παραλαμβάνει το ενθυλακωμένο πακέτο, το επανασυναρμολογεί αν προέκυψε κατατεμαχισμός του, και αφού δει την τιμή 41 στο πεδίο protocol αφαιρεί την IPv4 επικεφαλίδα. Ακολούθως, αν η διεύθυνση προορισμού του IPv6 πακέτου είναι διαφορετική από την δική του, προωθεί το πακέτο στον προορισμό με κανονική IPv6 δρομολόγηση. Πρέπει να σημειωθεί ότι κατά την φάση της απενθυλάκωσης δεν μεταβάλλεται η IPv6 επικεφαλίδα και απλώς μειώνεται το hop limit κατά ένα σε περίπτωση περαιτέρω προώθησης του πακέτου.
- Ίσως ο κόμβος εισόδου χρειάζεται να κρατά πληροφορίες για κάποιες παραμέτρους όπως το MTU (Maximum Transfer Unit - το μέγεθος του μεγαλύτερου IPv6 πακέτου που μπορεί να δρομολογηθεί) και το Hop Limit (το αντίστοιχο του TTL στο IPv4 πρωτόκολλο - ο αριθμός των ενδιάμεσων κόμβων από όπου θα περάσει το πακέτο μέχρι να απορριφθεί αν δεν έχει φτάσει στον προορισμό του), ούτως ώστε να προωθήσει τα IPv6 πακέτα μέσα στο tunnel.

Πιο κάτω φαίνεται η ενθυλάκωση ενός IPv6 σε ένα IPv4 πακέτο που πραγματοποιείται στον κόμβο εισόδου:



Σχήμα 9: Ενθυλάκωση IPv6 σε IPv4 πακέτο

και η **απενθυλάκωση** του πακέτου στο άλλο άκρο του τούνελ:



Σχήμα 10: Απενθυλάκωση IPv6 πακέτου

2.3.3.1 Maximum Transfer Unit (MTU) του tunnel και Κατατεμαχισμός των πακέτων

Αν και όχι καταστροφικός, η κατατεμαχισμός των IPv6 πακέτων για την μετάδοση τους μέσω του τούνελ είναι ανεπιθύμητος. Ο κατατεμαχισμός αυτός μπορεί να μειωθεί στο ελάχιστο έχοντας τον κόμβο εισόδου να ανιχνεύει το IPv4 Path MTU κατά μήκος του tunnel, χρησιμοποιώντας το IPv4 Path MTU Discovery Protocol (RFC 2185) και να καταγράφει το path MTU το οποίο έχει υπολογίσει. Έτσι, το IPv6 στρώμα(network layer) βλέπει το τούνελ σαν να πρόκειται για στρώμα ζεύξης (link layer), με MTU ίσο με το IPv4 MTU path που έχει καταγράψει μείον το μέγεθος της IPv4 επικεφαλίδας (20 bytes).
Όμως υπάρχει και η περίπτωση που ο κατατεμαχισμός των IPv6 πακέτων είναι αναπόφευκτος. Αυτό συμβαίνει στην περίπτωση που IPv4 path MTU υπολογίζει MTU μικρότερο από 1280 bytes, όπου 1280 είναι το μικρότερο μέγεθος που μπορεί να έχει ένα IPv6 πακέτο. Σε αυτή την περίπτωση, το στρώμα IPv6 (network layer)

πρέπει να ‘δει’ ένα στρώμα ζεύξης (link layer) με MTU 1280 bytes και ο κόμβος εισόδου να χρησιμοποιήσει IPv4 κατατεμαχισμό, κατά την προώθηση των πακέτων. Σε αυτή την περίπτωση, το Don't Fragment bit δεν πρέπει να τίθεται στην IPv4 επικεφαλίδα.

Στην περίπτωση όμως που οι κομβόι εισόδου έχουν να υλοποιήσουν μεγάλο αριθμό tunnels, αποφεύγεται ο υπολογισμός του IPv4 Path MTU, και αντί αυτού χρησιμοποιείται η τιμή του MTU του στρώματος ζεύξης(link layer) που είναι κάτω από το στρώμα δικτύου. Και σε αυτή την περίπτωση μπορεί το MTU του στρώματος ζεύξης να είναι μικρότερο από 1280, και όπως και πριν, το MTU για το tunnel πρέπει να τεθεί σε 1280 και το Don't Fragment bit στην τιμή 0 (να επιτρέπεται δηλαδή ο τεμαχισμός)

2.3.3.2 Hop Limit

Τα IPv6-πανω από-IPv4 tunnels ανήκουν στην κατηγορία "single-hop", δηλαδή το IPv6 hop limit μειώνεται κατά ένα όταν τα IPv6 πακέτα διασχίζουν το τούνελ. Με αυτό τον τρόπο κρύβεται η παρουσία του τούνελ και η ύπαρξη του είναι άγνωστη για τον χρηστή του δικτύου που δεν μπορεί να τα εντοπίσει με διαγνωστικά εργαλεία όπως το traceroute.
Για την υλοποίηση του single hop μοντέλου, οι δυο κόμβοι στα άκρα του tunnel, μεταχειρίζονται το πεδίο IPv6 hop limit όπως στην περίπτωση προώθησης ενός πακέτου σε ένα άλλο κόμβο, δηλαδή μειώνοντας το hop limit κατά 1 όταν προωθούν ένα IPv6 πακέτο, σαν να μην υπήρχαν οι ενδιάμεσοι IPv4 κομβόι (ο κόμβος προορισμού δεν μειώνει το hop limit).

Όσον αφορά το πεδίο TTL της εξωτερικής IPv4 επικεφαλίδας που χρησιμοποιείται για την ενθυλάκωση, επαφίεται στον εκάστοτε διαχειριστή να το θέσει στην τιμή μου επιθυμεί.

2.3.3.3. Κατασκευή της IPv4 Επικεφαλίδας

Κατά την ενθυλάκωση του IPv6 σε ένα IPv4 πακέτο τα πεδία της IPv4 επικεφαλίδας είναι τα εξής:

- Version: 4
- IP Header Length in 32-bit words: 5
- Type of Service: 0
- Total Length:

  Το μέγεθος του IPv6 φορτίου, συν τις IPv6 και IPv4 επικεφαλίδες.

- Identification: Όπως κάθε IPv4 πακέτο

- Flags:

  Η Don't Fragment (DF) flag όπως περιγράφεται πιο πάνω. Στην περίπτωση τεμαχισμού να τεθεί σε ένα το More Fragments (MF) bit.

- Fragment offset:

  Τίθεται όπως ορίζεται κατά τον τεμαχισμό.

- Time to Live:

  Καθορίζεται από τον διαχειριστή του κόμβου που κάνει την ενθυλάκωση.

- Protocol:

  41 (Καθορισμένος αριθμός για IPv6 φορτίο)

- Header Checksum:

  Υπολογίζεται το checksum της IPv4 επικεφαλίδας.

- Source Address:

  Η IPv4 διεύθυνση του εξωτερικού interface του κόμβου που πραγματοποιεί την ενθυλάκωση.

- Destination Address:

  Η IPv4 διεύθυνση του αλλού άκρου (κόμβος εξόδου) του tunnel.

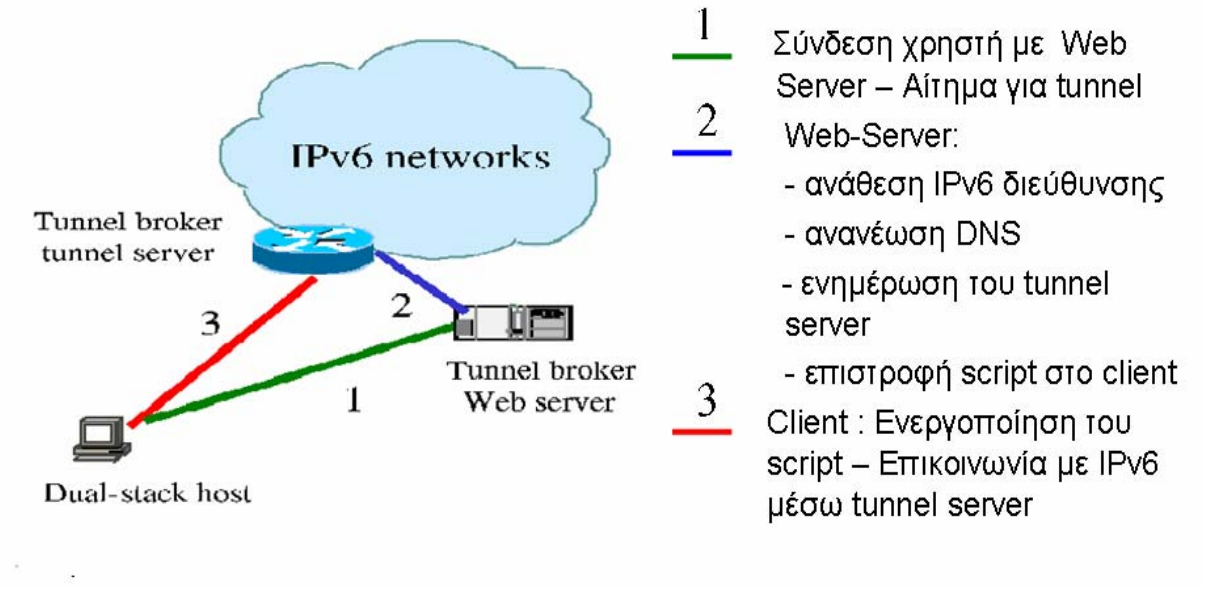
2.3.4 ΜΗΧΑΝΙΣΜΟΙ AUTOMATIC TUNNELING

Στην συνέχεια περιγράφονται συνοπτικά οι σημαντικότεροι μηχανισμοί automatic tunneling που έχουν χρησιμοποιηθεί και χρησιμοποιούνται μέχρι σήμερα. Στους μηχανισμούς αυτούς περιλαμβάνεται και ο 6to4 , ο οποίος θα περιγράφει εκτενέστερα σε επόμενες ενότητες. Οι υπόλοιποι μηχανισμοί δίνονται πιο κάτω:

2.3.4.1 Tunnel Broker

Ο συγκεκριμένος μηχανισμός παρέχει τον απλούστερο τρόπο σε ένα dual-stack host που βρίσκεται σε IPv4 δίκτυο να επικοινωνήσει με IPv6 κόμβους. Ο χρηστής που επιθυμεί IPv6 σύνδεση, συνδέεται σε ένα Web Server και αιτείται IPv6 σύνδεση. Κατόπιν, ο Web Server του αναθέτει μια IPv6 διεύθυνση, ανανεώνει αυτόματα το DNS, ενημερώνει τον Tunnel Broker Tunnel Server, μέσω του οποίου θα γίνει η σύνδεση και τέλος αποστέλλει ένα script στον χρηστή. Όταν ο χρήστης το τρέξει εγκαθίσταται ένα IPv6-over-IPv4 tunnel με τον broker server και η επικοινωνία με το IPv6 host γίνεται μέσω του broker server.

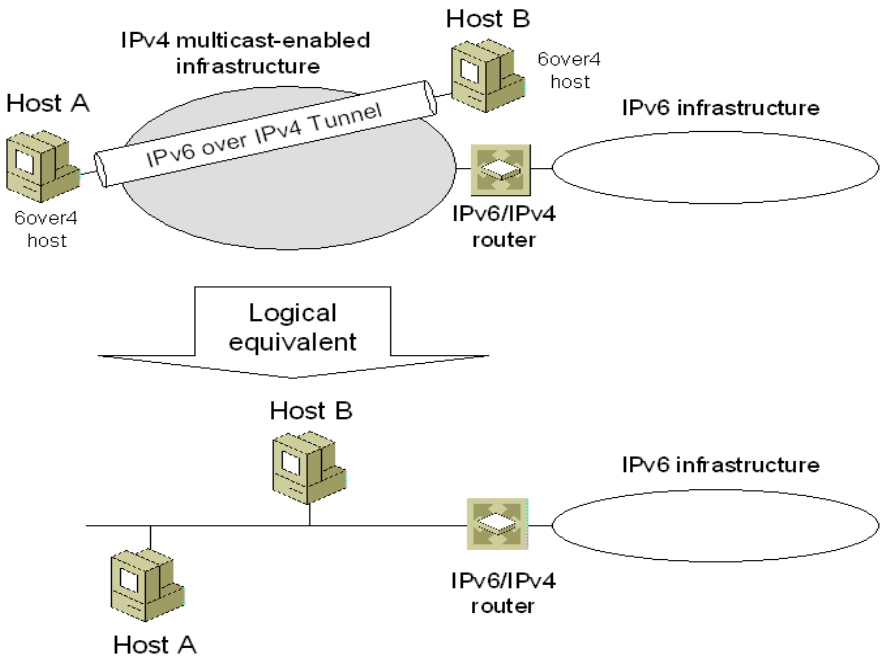
Ο tunnel broker μηχανισμός ,όπως αναφέρθηκε, είναι ο απλούστερος τρόπος για τους απλούς dial-up χρηστές να ενωθούν στο IPv6 δίκτυο και σήμερα προσφέρεται δωρεάν στις περισσότερες περιπτώσεις. Παρόλα αυτά υπάρχουν ακόμη κάποια προβλήματα όσον αφορά την ανακατανομή των δυναμικών IPv4 διευθύνσεων καθώς και κάποια ζητήματα ασφαλείας. Επίσης, ένα άλλο πρόβλημα που συχνά παρουσιάζεται είναι ότι οι διαχειριστές δεν γνωρίζουν για την ύπαρξη των μηχανισμών και τα περισσότερα firewalls αποκόπτουν τα πακέτα με τιμή του πεδίου protocol 41.



Σχήμα 11: Μηχανισμός Tunnel Broker

2.3.4.2 6over4
(ή αλλιώς multicast tunnelling)

Ο μηχανισμός αυτός επιτρέπει την επικοινωνία αποκομμένων IPv6 host (6over4) βρίσκονται σε ένα IPv4 domain που υποστηρίζει IPv4 multicast να επικοινωνούν μέσω αυτόματων τούνελ . Οι ίδιοι οι hosts σε αυτή την περίπτωση ενθυλακώνουν τα πακέτα και επικοινωνούν μεταξύ τους μέσω τούνελ μέσα στο IPv4 δίκτυο ή στέλνουν τα πακέτα σε ένα δρομολογητή με IPv6 σύνδεση που καταλαβαίνει 6over4 (συνήθως ένα 6to4 router) για επικοινωνία με native IPv6 hosts ή άλλους 6over4 που ανήκουν σε άλλο δίκτυο.
Ο μηχανισμός αυτός είναι γνωστός και ως multicast tunnelling, αφού χρησιμοποιεί την IPv4 υποδομή σαν να επρόκειτο για ένα τμήμα φυσικής ζεύξης, δηλαδή ένα LAN (Local Area Network), όπως φαίνεται στο πιο κάτω σχήμα.



Σχήμα 12: Μηχανισμός 6over4

Οι κομβόι στους οποίους έχει εγκατασταθεί 6over4 κάνουν αυτόματα μόνοι τους configure την link-local διεύθυνση του FE80::*WWXX*:*YYZZ* (όπου ww.xx.yy.zz η IPv4 διεύθυνση του – παγκόσμια ή ιδιωτική)και χρησιμοποιούν τις διεργασίες του Neighbour Discovery (address resolution και router discovery) όπως γίνεται σε μια φυσική ζεύξη με δυνατότητες multicast . Συγκεκριμένα, ο μηχανισμός παρέχει ένα σύνολο συσχετισμών των IPv4 multicast διευθύνσεων με τις αντίστοιχες IPv6, έτσι ώστε να επιτευχθούν οι λειτουργίες του IPv6 multicast πάνω από το IPv4 multicast, και οι 6over4 κομβόι στο IPv4 δίκτυο να επικοινωνούν μέσω των link-local διευθύνσεων τους.

Ο συγκεκριμένος μηχανισμός δεν χρησιμοποιήθηκε σε μεγάλο βαθμό λόγω ακριβώς της απαίτησης του για IPv4 multicast, κάτι που οι περισσότεροι παρόχοι ISP και

διαχειριστές δεν το παρέχουν . Έτσι, η χρήση του μηχανισμού περιορίστηκε μόνο σε πανεπιστημιακά δίκτυα.

2.3.4.3 ISATAP

Το ISATAP (Intra-Site Automatic Tunnel Addressing Protocol) είναι η εναλλακτική λύση στο πρόβλημα του 6over4, αφού και πάλι χρησιμοποιεί την IPv4 υποδομή σαν ένα ιδεατό link χωρίς όμως να απαιτείται IPv4 multicast. Επιπλέον, και σε αυτή την περίπτωση ο μηχανισμός δουλεύει και κάτω από NAT (Network Address Translation).

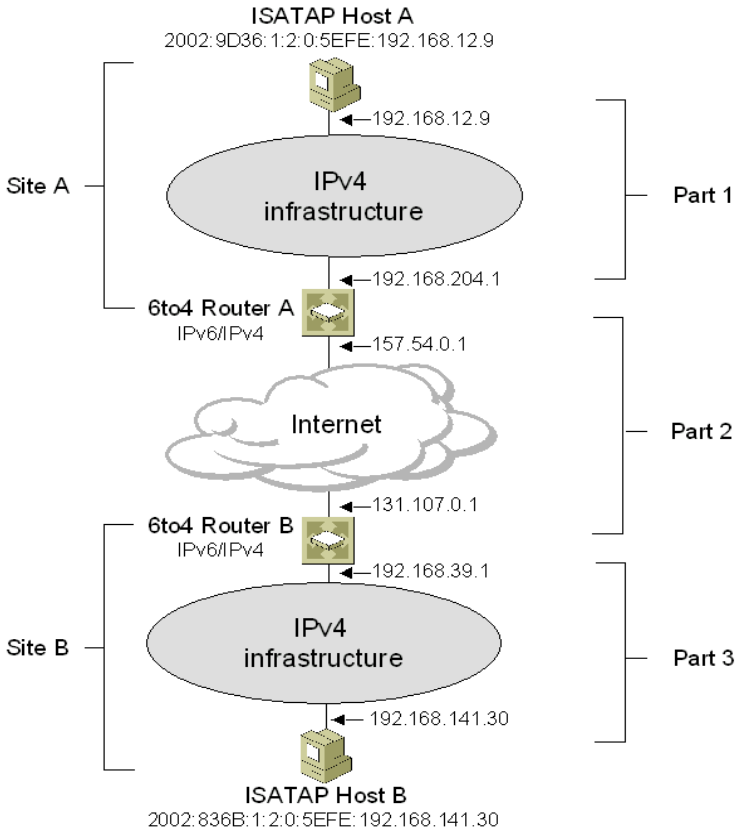
Χρησιμοποιώντας το link-local πρόθεμα fe80::/64 και τον interface identifier 0:5EFE:w.x.y.z (οπου w.x.y.z η IPv4 διεύθυνση του interface – παγκόσμια ή ιδιωτική) σχηματίζεται η link-local διεύθυνση για επικοινωνία με τον ISATAP router. Μόλις αρχικοποιηθεί ένας ISATAP κόμβος ψάχνει μέσω DNS για το όνομα "ISATAP" λαμβάνοντας έτσι τις διευθύνσεις όλων των ISATAP routers και φτιάχνει την Potential Router List (PRL). Ακολούθως, ο κόμβος στελνει μήνυμα Router Solicitation για να λάβει το πρόθεμα από τον router και να φτιάξει την παγκόσμια IPv6 ISATAP διεύθυνση του.
Έτσι, ο router στέλνει μήνυμα router-advertisement στον κόμβο, όπου μέσα περιέχεται το site-prefix έτσι ώστε ο κόμβος να φτιάξει την δική του παγκόσμια μοναδική διεύθυνση. Στην περίπτωση επικοινωνίας μέσα στο ίδιο site, ενθυλακώνονται τα πακέτα από τον ISATAP κόμβο και χρησιμοποιείται σαν IPv4 διεύθυνση προορισμού η διεύθυνση που προκύπτει από τον Interface Identifier της διεύθυνσης προορισμού (τα τελευταία 4 bytes). Στην περίπτωση που η επικοινωνία είναι με ISATAP ή native IPv6 κόμβο αλλού site, τα πακέτα ενθυλακώνονται και στέλνονται στον ISATAP router (που και σε αυτή την περίπτωση είναι συνήθως 4νας 6to4 router).

Ο μηχανισμός αυτός παρέχει εύκολη υλοποίηση του IPv6 σε διασκορπισμένους κόμβους μέσα σε IPv4 domain και υποστηρίζεται στα περισσότερα λειτουργικά συστήματα. Επιπλέον δουλεύει και κάτω από NAT.
Όσον αφορά την ασφάλεια, υπαρχουν και σε αυτή την περίπτωση κάποια κενά κυρίως ως προς τους ISATAP routers. Γενικά όμως, το μοντέλο αυτό υπερτερεί των υπόλοιπων μηχανισμού αυτομάτου tunneling, τόσο στο ότι παρέχει IPv4 ingress filtering και ip-protocol-41 filtering στους συνοριακούς δρομολογητές του site όσο και στο γεγονός ότι πιο δύσκολα ένας κακόβουλος κόμβος θα 'υποδυθεί' τον ISATAP router, αφού ο τελευταίος δέχεται πακέτα μόνο από τους δρομολογητές που είναι καταγραμμένοι στην Potential Router List (PRL).

Τα μειονεκτήματα του μηχανισμού είναι η δυσκολότερη υλοποίηση του από τους άλλους μηχανισμούς, το γεγονός ότι ακόμα δεν έχει γίνει standard αν στην πραγματικότητα όμως χρησιμοποιείται ήδη στις περισσοτερες πλατφόρμες .

Συνήθως, ο ISATAP μηχανισμός συνδυάζεται με το 6to4, κυρίως σε περιπτώσεις που η υλοποίηση γίνεται κάτω από NAT. Έτσι, εσωτερικά στο site χρησιμοποιείται ο ISATAP, και η σύνδεση με το IPv6 γίνεται μέσω του 6to4 router, ο οποίος διαφημίζει εσωτερικά το 6to4 πρόθεμα 2002::V4ADDR::/64 έτσι ώστε να σχηματιστούν οι global διευθύνσεις(2002:V4ADDR:1:5EFE:x.y.z.w οπου x.y.z.w η ιδιωτικη ή

παγκοσμια IPv4 διεύθυνση του host) για επικοινωνία εκτός του site. Το σενάριο αυτό δίνεται στο πιο κάτω σχήμα:



Σχήμα 13: Συνδυασμός 6to4 και ISATAP

2.3.4.4 Teredo

Επιτρέπει σε host που βρίσκονται πίσω από ένα ή περισσότερα επίπεδα NAT να επικοινωνούν μέσω τούνελ με IPv6 κόμβους. Η ενθυλάκωση των πακέτων γίνεται σε IPv4 UDP πακέτα τα οποία προωθούνται στο teredo relay- που είναι συνδεδεμένος με το IPv6 δίκτυο, για απενθυλάκωση- στην περίπτωση που η επικοινωνία είναι με native IPv6.

Σχήμα 14: Μηχανισμός Teredo. Ενθυλάκωση σε IPv4 UDP πακέτο

Στο σημείο αυτό έγκειται και η διάφορα του Teredo από τον 6to4, στο ότι δηλαδή η ενθυλάκωση γίνεται από τον κόμβο, και όχι στον συνοριακό δρομολογητή. Επίσης, λόγω του ότι τα περισσότερα NATs δεν επιτρέπουν πακέτα protocol 41 να διαπεράσουν το NAT, η ενθυλάκωση γίνεται σε IPv4 UDP πακέτα (δηλαδή στο πεδίο

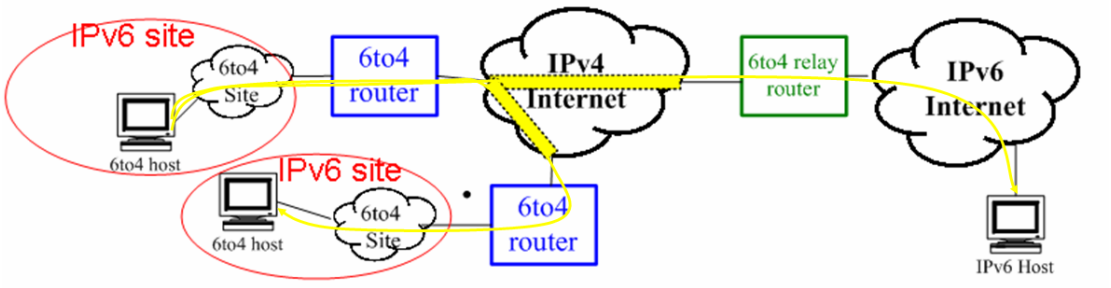
protocol τίθεται η τιμή του πρωτοκόλλου UDP) , όπου τα UDP συνήθως περνάνε από τα διάφορα επίπεδα NAT.

Ο μηχανισμός αυτός αποτελεί την τελευταία λύση όταν δεν υπάρχει δυνατότητα υλοποίησης των υπόλοιπων μηχανισμών. Η υλοποίηση του είναι περίπλοκη και μπορεί να λειτουργήσει μόνο κάτω από συγκεκριμένα είδη NAT, η διαχείριση του δύσκολη και σήμερα δεν υπάρχουν ακόμη αρκετοί teredo relays για σύνδεση με native IPv6 hosts.

3. ΜΗΧΑΝΙΣΜΟΣ 6TO4

3.1 Εισαγωγή:

Το 6to4 είναι ένας μηχανισμός αυτόματου tunneling που παρέχει την δυνατότητα σε απομονωμένα IPv6 sites, στα οποία δεν υπάρχει ευρύτερη IPv6 συνδεσιμότητα και υπηρεσίες από κάποιο πάροχο ISP, να επικοινωνούν μεταξύ τους πάνω από το ήδη υπάρχον IPv4 δίκτυο. Επιπλέον δίνεται η δυνατότητα να επικοινωνούν με sites που ανήκουν στο native IPv6 δίκτυο μέσω relay router. Ο μηχανισμός αυτός σχεδιάστηκε για το μεταβατικό στάδιο από τις IPv4 στις IPv6 διευθύνσεις ,κατά την περίοδο δηλαδή που οι δυο διευθύνσεις θα συνυπάρχουν. Σκοπός του μηχανισμού είναι απομονωμένες νησίδες IPv6 που έχουν πρόσβαση στο IPv4 wide area network , το οποίο δεν υποστηρίζει IPv6, να επικοινωνούν μεταξύ τους με εύκολο και αυτόματο τρόπο.
Με τον 6to4 μηχανισμό κάθε site που έχει μια **παγκόσμια μοναδική unicast IPv4 διεύθυνση** μπορεί χρησιμοποιώντας ενθυλάκωση με automatic tunneling να μεταδώσει πακέτα πάνω από το παγκόσμιο IPv4 δίκτυο χρησιμοποιώντας ένα μοναδικό πρόθεμα (prefix) 2002::/16. Εκτός από τον 6to4 router, οι υπόλοιποι τοπικοί hosts και routers κάτω από αυτόν δεν χρειάζεται να υλοποιούν τον 6to4 μηχανισμό και οι διάφορες λειτουργίες του εσωτερικού δικτύου γίνονται όπως ορίζει το IPv6 πρωτόκολλο.



Σχήμα 15: 6to4 μηχανισμός

Ο 6to4 μηχανισμός, επικράτησε των υπόλοιπων μηχανισμών μετάβασης και αντικατέστησε γρήγορα τα configured tunnels που αρχικά χρησιμοποιούνταν, τόσο λόγω της ευκολίας υλοποίησης του αλλά και των ελάχιστων απαιτήσεων , οι οποίες είναι:

- Ο συνοριακός δρομολογητής του site, ο οποίος είναι ο μόνος που υλοποιεί τον 6to4 μηχανισμό, πρέπει να είναι dual-stack.
- Το site πρέπει να διαθέτει μια παγκόσμια unicast IPv4 διεύθυνση για την επικοινωνία πάνω από την IPv4 υποδομή.
- Ένας 'διαθέσιμος' relay router, για την περίπτωση επικοινωνίας του 6to4 site με το native IPv6 δίκτυο. (δηλαδή με διευθύνσεις προορισμού με παγκόσμιο unicast TLA 2001 ή 3ffe)

Έτσι, απλά υλοποιώντας τον 6to4 μηχανισμό στον συνοριακό δρομολογητή του 6to4 site, το οποίο είναι στην ουσία ένα κανονικό IPv6 site, προσφέρεται σε όλους τους hosts IPv6 συνδεσιμότητα, χωρίς οι ίδιοι να υλοποιούν 6to4, και χωρίς να χρειάζεται

να κάνουν κάποια ειδική διαδικασία, εκτός του ότι καθορίζουν στο default route τους, next hop τη διεύθυνση του 6to4 router. Επιπλέον, ο 6to4 μηχανισμός δεν επιβαρύνει με επιπρόσθετες έγγραφες τους IPv4 πίνακες δρομολόγησης και δεν απαιτεί κάποιο εξωτερικό IPv6 πρωτόκολλο δρομολόγησης αφού το αντίστοιχο IPv4 εκτελεί την απαιτούμενη διαδικασία δρομολόγησης.

.

3.2 Ορολογία:

6to4 pseudo-interface: Αντιστοιχεί σε ένα IPv6 interface και είναι το σημείο στο οποίο γίνεται η ενθυλάκωση του IPv6 πακέτου στο IPv4. Το 6to4 pseudo-interface είναι το end-point του τούνελ.

6to4 prefix: Το IPv6 πρόθεμα που φτιάχνεται όπως περιγράφεται στην επόμενη ενότητα.

6to4 address: Η 6to4 διεύθυνση που κατασκευάζεται βάσει του 6to4 προθέματος.

Native IPv6 address:IPv6 διεύθυνση που κατασκευάζεται χρησιμοποιώντας διαφορετικό πρόθεμα από το 6to4 prefix.

6to4 router :Ένας IPv6 router(δρομολογητής) ο οποίος υποστηρίζει το IPv6 pseudo-interface. Είναι ο συνοριακός δρομολογητής μεταξύ του IPv6 site και του wide-area IPv4 network. Δρομολογεί το διακινούμενο φορτίο μεταξύ των 6to4 hosts μέσα στο ίδιο το site και μεταξύ άλλων 6to4 routers και relay routers πάνω από ένα IPv4 inter-network όπως το Internet. Ο 6to4 router χρειάζεται επιπλέον manual configuration για το 6to4 pseudo-interface όπου θα γίνεται η ενθυλάκωση και η απενθυλάκωση των IPv6 πακέτων.

6to4 host: Ένας IPv6 host που έχει τουλάχιστον μια 6to4 διεύθυνση. Κατά τα αλλά είναι ένας κανονικός IPv6 host.Οι 6to4 hosts δεν χρειάζονται καθόλου manual configuration και δημιουργούν από μόνοι τους τις 6to4 διευθύνσεις τους χρησιμοποιώντας τους standard μηχανισμούς address autoconfiguration.

6to4 site: Ένα site που εσωτερικά χρησιμοποιεί το IPv6 πρωτόκολλο και περιέχει τουλάχιστον ένα 6to4 host και τουλάχιστον ένα 6to4 router.

Relay router: Ένας 6to4 router ο οποίος έχει συντονισθεί έτσι ώστε να υποστηρίζει δρομολόγηση μεταξύ 6to4 διευθύνσεων και native IPv6 διευθύνσεων. Ουσιαστικά είναι ένας κανονικός IPv6 router με τουλάχιστον ένα 6to4 pseudo-interface και τουλάχιστον ένα IPv6 interface.

IPv4-only node: Ένας κόμβος που υλοποιεί μόνο IPv4 και έχει μόνο IPv4 διεύθυνση. Ο κόμβος αυτός δεν υποστηρίζει IPv6.

IPv6-only node: Ένας κόμβος που υποστηρίζει μόνο IPv6 και μπορεί να επικοινωνήσει μόνο με IPv6 κόμβους και εφαρμογές.

IPv6/IPv4 node: Κόμβος που υλοποιεί τόσο IPv6 όσο και IPv4.

IPv4 node: Κόμβος που υλοποιεί IPv4 (στέλνει και λαμβάνει IPv4 πακέτα). Ο κόμβος αυτός μπορεί να είναι IPv4-only ή IPv6/IPv4 κόμβος.

IPv6 node : Κόμβος που υλοποιεί IPv6 (στέλνει και λαμβάνει IPv6 πακέτα). Ο κόμβος αυτός μπορεί να είναι IPv6-only ή IPv6/IPv4 κόμβος.

3.3 Διευθύνσεις 6to4

3.3.1 Προσδιορισμός του προθέματος 6to4:

Ο μηχανισμός 6to4 εφαρμόζεται σε ένα site που έχει τουλάχιστον μια παγκόσμια μοναδική IPv4 διεύθυνση(αναφέρεται ως V4ADDR από τώρα και στο εξής). Η IANA έχει ορίσει το unicast 6to4 πρόθεμα (TLA) **2002::/16** για αποκλειστική χρήση από τον 6to4 μηχανισμό. Το 6to4 πρόθεμα μαζί με την IPv4 διεύθυνση σχηματίζουν το /48 πρόθεμα του site (site prefix) 2002::V4ADDR/48, το οποίο είναι και αυτό μοναδικό λόγω του ότι προκύπτει από τη μοναδική V4ADDR. Πιο κάτω δίνεται ένα παράδειγμα του σχηματισμού του 6to4 προθέματος του site από την IPv4 διεύθυνση:

Σχήμα 16: Σχηματισμός του 6to4 site prefix

Το πρόθεμα που αυτό είναι ένα κανονικό IPv6 πρόθεμα, επιτρέποντας έτσι στο IPv6 domain (το 6to4 site στην περίπτωση μας) να το χρησιμοποιήσει όπως κάθε έγκυρο IPv6 πρόθεμα (αυτόματη ανάθεση διευθύνσεων και discovery καθώς και native IPv6 routing). Επιπλέον, με αυτό τον τρόπο σχηματίζονται αμέσως 2^{16} υποδίκτυα με 2^{64} κόμβους το καθένα, κάτω από μια και μοναδική IPv4 διεύθυνση, δίνοντας έτσι και ο ίδιος ο μηχανισμός μια λύση στο πρόβλημα της έλλειψης των IPv4 διευθύνσεων.

Αφότου καθοριστεί το 6to4 πρόθεμα του site /48, καθορίζονται τα διάφορα υποδίκτυα του site, παίρνοντας το καθένα μοναδική τιμή για το SLA ID, έτσι που τελικά σχηματίζεται το μοναδικό 2002:V4ADDR:SLA ID::/64 πρόθεμα του υποδικτύου (subnet prefix). Το πρόθεμα αυτό διαφημίζεται από τους 6to4 routers με μηνύματα Router Advertisements όπως ορίζει το Neighbor Discovery (ND) protocol έτσι ώστε

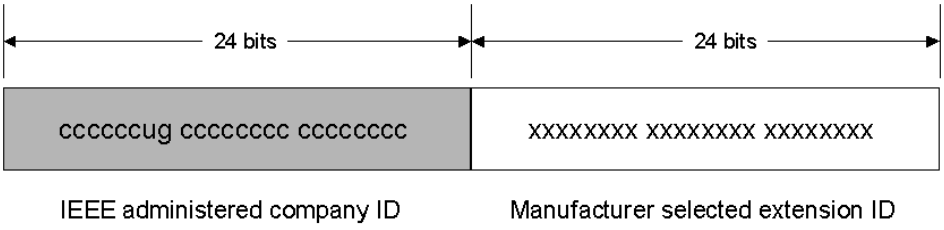
οι hosts να σχηματίσουν από μόνοι τους την 6to4 παγκόσμια διεύθυνση τους με Stateless Address Autoconfiguration. Συγκεκριμένα, οι 6to4 hosts υπολογίζουν από μόνοι τους τα τελευταία 64 bit, τα οποία είναι ο μοναδικός για τον κάθε κόμβο interface identifier, και σε συνδυασμό με τα 64 bit του προθέματος υποδικτύου που περιέχονται στις διαφημίσεις του 6to4 router, σχηματίζεται η παγκόσμια IPv6 (6to4) διεύθυνση τους (σχήμα πιο κάτω),με την οποία θα εγγράφουν στο DNS καθιστώντας έτσι δυνατή την ανάγνωση της από άλλους IPv6 κόμβους, δίνοντας τους έτσι την δυνατότητα επικοινωνίας τόσο με άλλους 6to4 κόμβους όσο και με native IPv6 κόμβους.

Bits 0-16	17-48	49-64	65-128
2002	IPv4 Address	SLA ID	Interface ID (probably the EUI-64 ID)

Σχήμα 17: 6to4 διεύθυνση

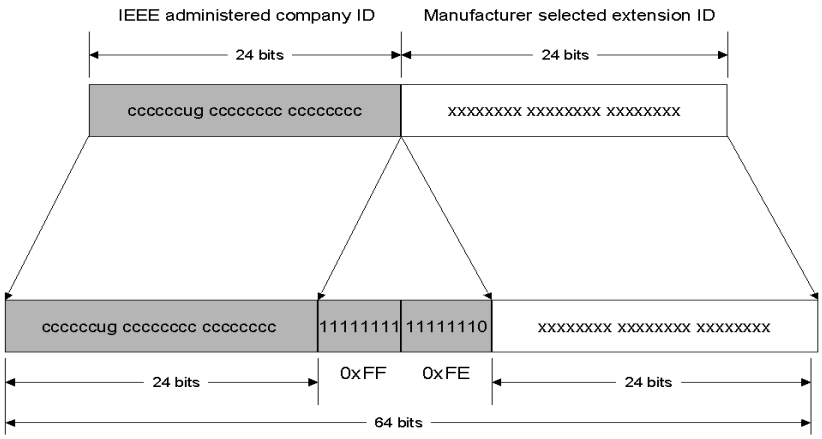
3.3.2 Interface Identifier

Μέσα σε ένα site ο κάθε host έχει να σχηματίσει τα τελευταία 64-bit που είναι γνωστά σαν ο Interface Identifier, και είναι μοναδικά για τον κάθε κόμβο. Έχουν προταθεί και έχουν υλοποιηθεί διάφοροι τρόποι για τον προσδιορισμό του Interface ID. Στην συγκεκριμένη αναφορά περιγράφεται ο EUI-64, ο οποίος είναι ο πιο καθιερωμένος και χρησιμοποιείται σήμερα από τα περισσότερα συστήματα. Ο Extended Unique Identifier (EUI)-64 address-based interface identifier ορίστηκε από το Institute of Electrical and Electronic Engineers (IEEE) και χρησιμοποιεί για τον υπολογισμό του Interface ID την 48-bit μοναδική για κάθε κατασκευαστή διεύθυνση MAC, ορίζοντας έτσι μοναδικά τον Interface ID. Πιο κάτω φαίνονται τα 48-bit της MAC διεύθυνσης:



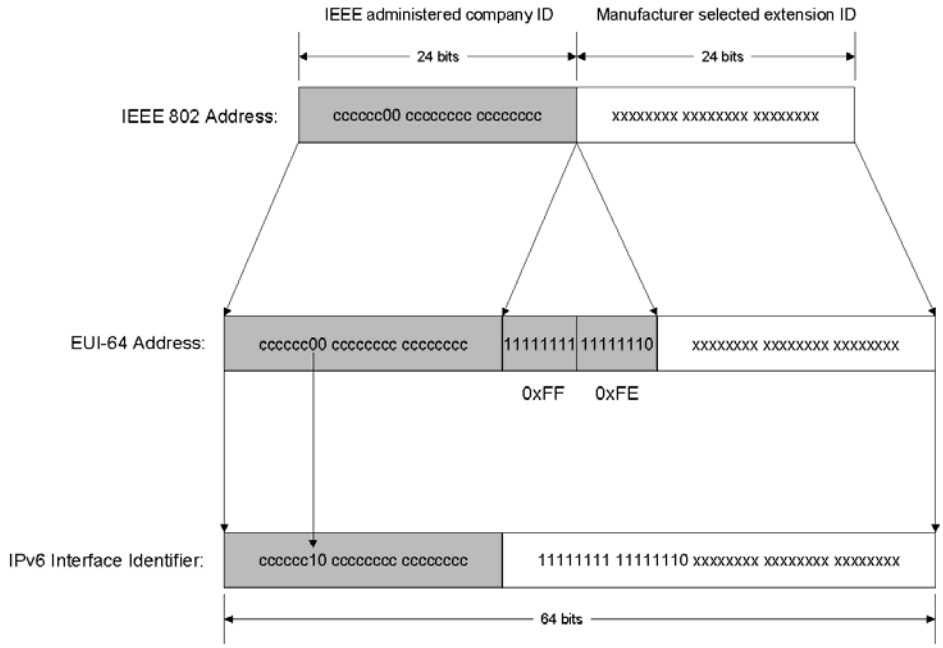
Σχήμα 18: EUI – 64. Σχηματισμός του Interface Identifier. (1/3)

Όπου με το γράμμα u συμβολίζεται το Universal/Local (U/L) bit το οποίο είναι το διπλανό του low order bit του πρώτου byte.
Η διαδικασία προσδιορισμού του EUI-64 έχει ως εξής:
Επειδή ο Interface ID έχει μήκος 64-bit και η MAC διεύθυνση 48-bit, 16 επιπλέον bits με τιμή 11111111 11111110 (0xFFFE) εισέρχονται στη MAC διεύθυνση μεταξύ του company ID και του extension ID, όπως φαίνεται στο πιο κάτω σχήμα:



Σχήμα 19: EUI – 64. Σχηματισμός του Interface Identifier. (2/3)

Και τέλος παίρνεται το συμπλήρωμα του U/L (αν είναι 1 τίθεται σε 0 και αν είναι 0 σε 1) σχηματίζοντας έτσι τον μοναδικό Interface ID. Ολόκληρη η διαδικασία φαίνεται στο πιο κάτω σχήμα:



Σχήμα 20: EUI – 64. Σχηματισμός του Interface Identifier. (3/3)

3.4 Επιλογή Κατάλληλης Διεύθυνσης

(RFC 3484)

Για να διασφαλιστεί η σωστή λειτουργία του 6to4 μηχανισμού πρέπει να υλοποιηθεί κατάλληλα η επιλογή της διεύθυνσης πηγής και της διεύθυνσης προορισμού. Για αυτό έχουν προταθεί οι παρακάτω κανόνες που επιτρέπουν την σωστή λειτουργία του μηχανισμού:

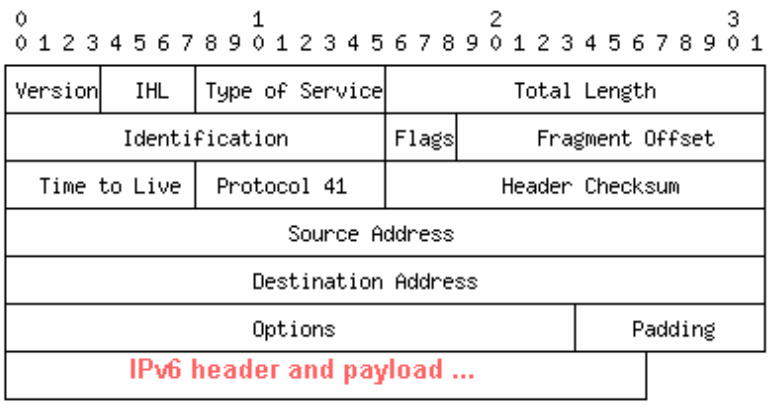
- Αν ένας host έχει μόνο 6to4 διεύθυνση και ο άλλος έχει και 6to4 και native IPv6 διεύθυνση, τότε και οι δυο θα χρησιμοποιήσουν την 6to4 διεύθυνση.

- Αν και οι δυο έχουν και 6to4 διεύθυνση και native IPv6 διεύθυνση τότε η native IPv6 διεύθυνση θα χρησιμοποιηθεί και από τους δυο.

Ενθυλάκωση(Encapsulation) /Απενθυλάκωση (Decapsulation) των 6to4 πακέτων – Μηχανισμοί Tunneling:

Τα IPv6 πακέτα ενθυλακώνονται σε IPv4 πακέτα όταν φτάσουν στον 6to4 router στο σύνορο του 6to4 site, έτσι ώστε να μεταδοθούν πάνω στο ήδη υπάρχον IPv4 δίκτυο μεσώ τούνελ. Συγκεκριμένα, όσα IPv6 πακέτα φτάσουν στον 6to4 router με διεύθυνση προορισμού που δεν ανήκει στο συγκεκριμένο site, προωθούνται στο 6to4 interface για να μεταδοθούν μεσώ τούνελ είτε στον προορισμό κατευθείαν αν πρόκειται για κάποιον 6to4 host, είτε στον configured relay router αν προορίζονται για κάποιον native IPv6 host. Τα IPv6 πακέτα ταξιδεύουν μέσα στα IPv4 πακέτα με τιμή 41 στο πεδίο protocol της IPv4 επικεφαλίδας, έτσι ώστε ο δρομολογητής-παραλήπτης, να αντιλαμβάνεται ότι έχει γίνει ενθυλάκωση, και να ξεφλουδίζει τα πακέτα για να τα προωθεί στον τελικό προορισμό τους. Επιπλέον, στην IPv4 επικεφαλίδα περιέχονται οι IPv4 διευθύνσεις πηγής και προορισμού. Η μια (ή και οι δυο αν ο προορισμός είναι επίσης 6to4 κόμβος) είναι η V4ADDR που έχει οριστεί πιο πάνω, η μοναδική δηλαδή παγκόσμια IPv4 διεύθυνση του εκάστοτε 6to4 site. Στην άλλη περίπτωση (native IPv6 site) η διεύθυνση προορισμού στην εξωτερική IPv4 επικεφαλίδα είναι η IPv4 διεύθυνση του Relay Router που έχει καθοριστεί από τον διαχειριστή στον πίνακα δρομολόγησης του 6to4 router.

Συγκεκριμένα, αφού προωθηθούν τα πακέτα στο 6to4 interface, γράφεται στο πεδίο source address της IPv4 επικεφαλίδας η IPv4 διεύθυνση του site. Ακολούθως, αν τα πρώτα 16-bits της διεύθυνση προορισμού αντιστοιχούν στην στατική έγγραφη του routing table 2002::/16 (άρα πρόκειται για 6to4 διεύθυνση προορισμού) και το πρόθεμα /48 είναι διάφορο του τοπικού site, τότε αποσπάται η IPv4 (V4ADDR) διεύθυνση του 6to4 site και γράφεται στο πεδίο destination address της IPv4 επικεφαλίδας. Αν όμως ο η διεύθυνση προορισμού δεν αντιστοιχεί σε 6to4 διεύθυνση (δηλ. η διεύθυνση προορισμού δεν ταιριάζει στο πρόθεμα 2002::/16) αλλά σε άλλη IPv6 unicast διεύθυνση, τίθεται σαν διεύθυνση προορισμού στην IPv4 επικεφαλίδα η διεύθυνση του relay router που έχει καθοριστεί για το συγκεκριμένο route, κάποια συγκεκριμένη unicast IPv4 διεύθυνση του Relay Router ή εναλλακτικά η anycast διεύθυνση 192.88.99.0/24 (RFC 3068 - An Anycast Prefix for 6to4 Relay Routers) για τον πλησιέστερο Relay Router. Κατόπιν, αφού έχει γίνει η ενθυλάκωση προωθούνται μέσω τούνελ τα IPv4 πακέτα (που περιέχουν την επικεφαλίδα IPv6 και τα δεδομένα) με IPv4 δρομολόγηση πάνω από το IPv4 δίκτυο. Φαίνεται πιο κάτω το τελικό IPv4 πακέτο που μεταδίδεται:



Σχήμα 21: Το IPv4 πακέτο όπως μεταδίδεται μέσα από το tunnel

Τέλος, στην άλλη πλευρά του τούνελ ο 6to4 router του site στο οποίο ανήκει η διεύθυνση προορισμού, αν δει τιμή 41 στο πεδίο protocol της επικεφαλίδας, προωθεί τα πακέτα στο δικό του 6to4 interface, όπου αφού κάνει κάποιους έλεγχους ασφάλειας, ¨ξεφλουδίζει¨ με την ίδια διαδικασία την IPv4 επικεφαλίδα, βλέπει την 6to4 διεύθυνση προορισμού και με κανονική IPv6 δρομολόγηση στέλνει τα πακέτα στον προορισμό τους. Ομοίως, και ο Relay Router, αφού πάρει τα πακέτα μέσα από το τούνελ και διαπιστώσει ότι προέρχονται από 6to4 site, αφαιρεί την IPv4 επικεφαλίδα και προωθεί τα πακέτα με IPv6 δρομολόγηση μέσα στο native IPv6 site.

Κατά την ενθυλάκωση, τα 8-bits του πεδίου TimetoLive της IPv4 επικεφαλίδας τίθενται όπως τα 8-bits του πεδίου Hop Limit της επικεφαλίδας IPv6 (αφού ουσιαστικά δεν υπάρχει διάφορα στην υλοποίηση τους από τα upper-layer protocols που βασίζονται στο στρώμα δικτύου IP), εκτός αν οριστεί διαφορετικά από τον διαχειριστή του συστήματος.

Σημαντικό να αναφερθεί είναι το ότι ο 6to4 μηχανισμός δεν πρέπει να αποστέλλει πακέτα σε broadcast ή multicast IPv4 προορισμούς. Γενικά πρέπει να απορρίπτει όλα τα πακέτα που οι IPv4 και IPv6 διευθύνσεις δεν είναι παγκόσμιες unicast. (Αναλυτικότερα στην παράγραφο ¨Θέματα Ασφαλείας¨).

3.5.1 Maximum Transmission Unit

Σε ότι αφορά το μέγιστο μέγεθος μετάδοσης του πακέτου, ισχύουν τα όσα αναφέρθηκαν στην αντίστοιχη παράγραφο για το MTU για τους μηχανισμούς tunneling. Αν το μέγεθος της MTU είναι πολύ μεγάλο για κάποια ενδιάμεσα site, θα πραγματοποιηθεί IPv4 κατατεμαχισμός(fragmentation) των πακέτων. Αν και όχι επιθυμητός δεν είναι καταστροφικός εκτός από την περίπτωση της anycast διεύθυνσης για relay routers, που μπορεί τα τεμάχια να καταλήξουν σε διαφορετικούς IPv4 προορισμούς. Γενικά, το IPv4 "do not fragment" bit δεν πρέπει να τεθεί ¨1¨ στην επικεφαλίδα του ενθυλακωμένου πακέτου.

3.5.2 Link-Local Address:

Δεν χρειάζεται να οριστεί link-local διεύθυνση για τον 6to4 interface και δεν έχει αναφερθεί μέχρι τώρα κάποια περίπτωση χρήσης της. Σε περίπτωση που επιθυμείται καθορισμός της χρησιμοποιείται το πρόθεμα FE80::///64 και τίθεται στα 32 πρώτα bits του Interface ID η IPv4 διεύθυνση και στα υπόλοιπα 32 το 0.

3.5.3 Neighbor Discovery:

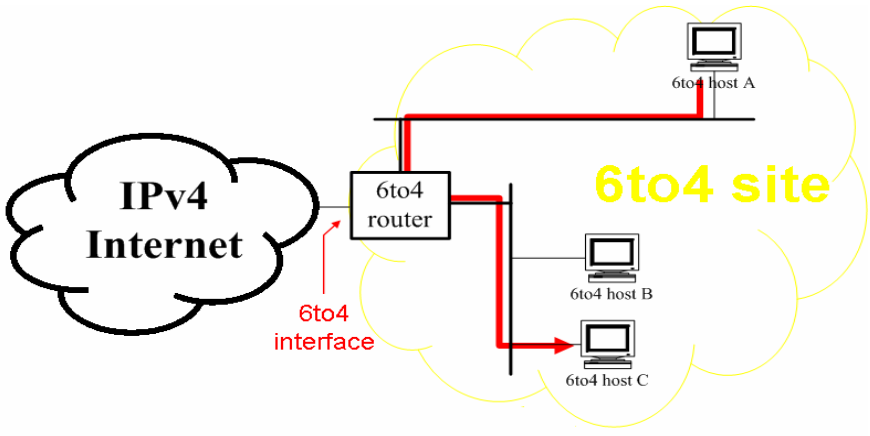
Η μόνη λειτουργία του ND που μπορεί να χρησιμοποιηθεί είναι η Stateless Address Autoconfiguration για τον σχηματισμό της link-layer διεύθυνσης, που όπως αναφέρθηκε στην προηγούμενη παράγραφο δεν είναι αναγκαία για το 6to4 interface.

3.6 ΣΕΝΑΡΙΑ ΧΡΗΣΗΣ ΤΟΥ 6TO4

3.6.1 Επικοινωνία 6to4 host με 6to4 host μέσα στο ίδιο site

Σε αυτή την περίπτωση χρησιμοποιείται μόνο IPv6 δρομολόγηση μέσα στο ίδιο το site και οι hosts συμπεριφέρονται σαν δυο κανονικοί IPv6 κόμβοι. Γενικά, μέσα στο 6to4 site χρησιμοποιείται κάποιο πρωτόκολλο εσωτερικής δρομολόγησης όπως το RIPng (Routing Information Protocol Next Generation – το αντίστοιχο του RIP για IPv6) και η μόνη διάφορα από τα αλλά native IPv6 sites είναι μια εγγραφή στους πίνακες δρομολόγησης του προθέματος 2002::/16 με next-hop την διεύθυνση του 6to4 router, ο οποίος έστειλε τις διαφημίσεις από τις οποίες προέκυψαν οι IPv6 διευθύνσεις των κόμβων και η οποία χρησιμοποιείται μόνο στην περίπτωση επικοινωνίας με κάποιο IPv6 (6to4 ή native) κόμβο εκτός του site. Το 6to4 interface δεν χρησιμοποιείται σε αυτό το σενάριο επικοινωνίας.

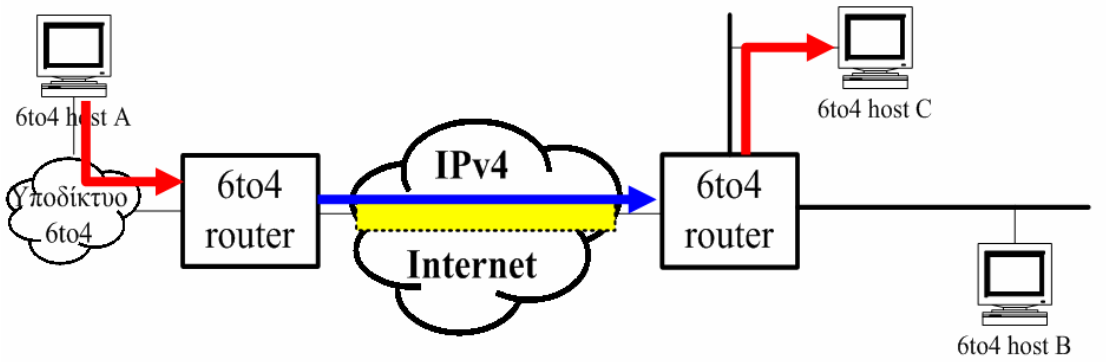
Η περίπτωση του πρώτου σεναρίου επικοινωνίας φαίνεται στο πιο κάτω σχήμα μεταξύ του host A και του host C.



Σχήμα 22: Περίπτωση επικοινωνίας 6to4 hosts στο ίδιο 6to4 site

3.6.2 Επικοινωνία μεταξύ δυο διαφορετικών 6to4 sites

Για αυτή την περίπτωση είναι αναγκαίο τα sites που πρόκειται να επικοινωνήσουν να διαθέτουν από τουλάχιστον μια παγκόσμια μοναδική IPv4 διεύθυνση. Αυτό το σενάριο υλοποιείται για τα sites που αρχίζουν να χρησιμοποιούν IPv6 αλλά ακόμα δεν μπορούν να έχουν πρόσβαση σε native ISP services. Μέσα στο site όλοι οι host υλοποιούν το IPv6 πρωτόκολλο , οι διευθύνσεις ορίζονται μεσώ των διαφημίσεων του router (router advertisement→ stateless address autoconfiguration) και οι hosts έχουν τις DNS name/address έγγραφες τους , έτσι ώστε να είναι δυνατό ένας host από κάποιο άλλο site να αποκτήσει την διεύθυνση τους.(Σε επόμενη παράγραφο περιγράφεται το DNS στο IPv6). Στο σενάριο αυτό το tunnel εγκαθίσταται μεταξύ των συνοριακών δρομολογητών των δυο sites στα οποία ανήκουν οι 6to4 hosts που πρόκειται να επικοινωνήσουν και τα πακέτα δρομολογούνται κατευθείαν από τον ένα 6to4 router στον άλλο χωρίς την παρουσία κάποιου ενδιάμεσου relay. Η περίπτωση του δεύτερου σεναρίου φαίνεται πιο κάτω μεταξύ του host A και του host C.



Σχήμα 23: Περίπτωση επικοινωνίας 6to4 host με 6to4 host που ανήκει σε άλλο site

Διαδικασία:

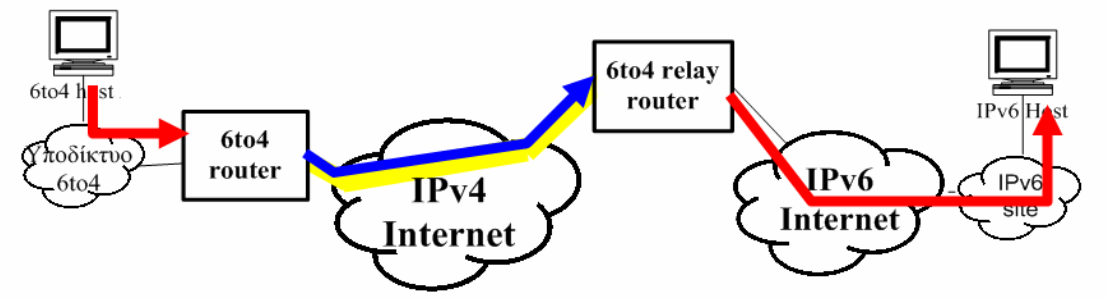
Ο 6to4 host στο ένα site(ουσιαστικά ένας IPv6 host) μαθαίνει μεσώ DNS αναζήτησης την διεύθυνση του αλλού 6to4 host. Μέσα στο site εκτός από τις 2002:V4ADDR::/48 διευθύνσεις, οι υπόλοιπες χειρίζονται σαν οποιεσδήποτε μη τοπικές διευθύνσεις και στέλνονται μεσώ του default route στο 6to4 router. Όταν ο router δει το 2002::/16 και επιπλέον ότι ο προορισμός δεν ανήκει στο τοπικό δίκτυο, προωθεί τα πακέτα στο 6to4 pseudo-interface, γίνονται οι απαραίτητοι έλεγχοι ασφάλειας και ακολούθως γίνεται η ενθυλάκωση σε IPv4 πακέτα. Το πεδίο protocol της εξωτερικής IPv4 επικεφαλίδας τίθεται σε 41 και κατόπιν τα πακέτα στέλνονται με κανονική IPv4 δρομολόγηση στον προορισμό τους με διεύθυνση προορισμού την IPv4 διεύθυνση που προκύπτει από το 2002::V4ADDR::/48 της IPv6 διεύθυνσης προορισμού, αν αφαιρεθούν τα πρώτα 2002::/16 bit ,και διεύθυνση προελεύσεως την IPv4 διεύθυνση του site που είναι και η διεύθυνση του εξωτερικού interface προς το Internet, από όπου θα σταλούν τα πακέτα.
Όταν ο 6to4 router του site προορισμού πάρει το πακέτο και δει στο IPv4 protocol 41 ‘ξεφλουδίζει’ την IPv4 επικεφαλίδα αφού γίνουν οι απαραίτητοι έλεγχοι

ασφάλειας και προωθεί το πακέτο σαν με κανονική IPv6 δρομολόγηση μέσα στο 6to4 site προορισμού.

Έτσι αν γίνουν οι κατάλληλες έγγραφες DNS , ο οποιαδήποτε αριθμός sites μπορεί να επικοινωνήσουν μεταξύ τους, χωρίς την χρήση εξωτερικού πρωτοκόλλου δρομολόγησης (π.χ. BGP4+)αφού το ήδη υπάρχον IPv4 πρωτόκολλο εκτελεί την απαραίτητη διαδικασία.

3.6.2 Επικοινωνία 6to4 site με site που είναι συνδεδεμένο στο native IPv6 δίκτυο

Για αυτή την περίπτωση επικοινωνίας χρειάζεται τουλάχιστον ένας relay router, ο οποίος είναι στην ουσία ένας κανονικός IPv6 router με επιπλέον ένα 6to4 pseudo-interface όπου εκεί εκτελούνται η ενθυλάκωση και απενθυλάκωση των IPv6 πακέτων. Στη συγκεκριμένη αναφορά αναλύεται μόνο η περίπτωση που δεν χρησιμοποιείται κάποιο εξωτερικό πρωτόκολλο δρομολόγησης αλλά απλά ο 6to4 router του site που θέλει να στείλει σε site που ανήκει στο native IPv6 δίκτυο έχει καταγραμμένη στο routing table του την διεύθυνση του relay router που θα χρησιμοποιηθεί. Η διεύθυνση αυτή μπορεί να είναι είτε η unicast IPv4 διεύθυνση του relay είτε η anycast διεύθυνση 192.88.99.1 που καθορίζεται από το σχετικό RFC 3068 “An Anycast Prefix for 6to4 Relay Routers” για προώθηση των πακέτων προς τον πλησιέστερο για το 6to4 site relay router. Το σενάριο αυτό έχει επικρατήσει και χρησιμοποιείται αποκλειστικά σήμερα για αυτή την περίπτωση επικοινωνίας. Το σενάριο αυτό φαίνεται στο πιο κάτω σχήμα.



Σχήμα 24: Περίπτωση επικοινωνίας 6to4 host με host που ανήκει στο native IPv6 δίκτυο

Διαδικασία:

- **Αποστόλη πακέτων από host του 6to4 site σε host που ανήκει σε native IPv6 domain:**

  Αφού βρεθεί μεσώ DNS η διεύθυνση του host όπου θα αποσταλεί το πακέτο, ο 6to4 host στέλνει το πακέτο στον 6to4 router του site μέσω το default route. Ο 6to4 router βλέπει την διεύθυνση προορισμού και προωθεί το πακέτο στο 6to4 interface, όπου αφού γίνουν οι απαραίτητοι ελέγχοι ασφαλείας, ενθυλακώνεται

το IPv6 πακέτο κάτω από την IPv4 επικεφαλίδα. Διεύθυνση προορισμού τίθεται η διεύθυνση του relay router που υπάρχει σαν next-hop στον πίνακα δρομολόγησης για τον συγκεκριμένο προορισμό. Αυτή μπορεί να είναι είτε η unicast διεύθυνση του relay είτε η anycast διεύθυνση, για τον πλησιέστερο προς το 6to4 site relay. Διεύθυνση προελεύσεως τίθεται η παγκόσμια IPv4 διεύθυνση του site, το πεδίο protocol στην τιμή 41 και προωθούνται τα πακέτα με κανονική IPv4 δρομολόγηση.
Ακολούθως, αφού ο relay router πάρει τα πακέτα και δει το IPv4 protocol 41, κάνει τους απαραίτητους έλεγχους ασφάλειας, ¨ξεφλουδίζει¨ την IPv4 επικεφαλίδα και προωθεί κανονικά με IPv6 δρομολόγηση το πακέτο στον προορισμό του.

- **Αποστόλη πακέτων από το native IPv6 site στο 6to4 site:**

Ο 6to4 relay router διαφημίζει στο το πρόθεμα 2002::/16 μέσα στο native IPv6 δίκτυο, έτσι ώστε ο κάθε κόμβος να έχει ένα route 2002::/16 με next-hop την διεύθυνση του relay router που έστειλε τις διαφημίσεις.(Είναι σημαντικό οι υπόλοιποι κομβόι του native IPv6 να φιλτράρουν και να αποβάλλουν διαφημίσεις με προθέματα μεγαλύτερα από 16-bits, έτσι ώστε να μην διογκώνονται οι πίνακες δρομολόγησης). Αν ο IPv6-only node θέλει να στείλει σε ένα 6to4 host, μεσώ του route prefix 2002::/16 που έχει στο table του από τις διαφημίσεις του relay router στέλνει τα πακέτα στον router, όπου ακολούθως αφού η next-hop IPv6 διεύθυνση ταιριάζει στο πρόθεμα 2002::/16, γίνεται η προώθηση τους στο 6to4 pseudo-interface όπως στα προηγούμενα και η ενθυλάκωση των πακέτων, για να αποσταλούν με IPv4 στον 6to4 router του site προορισμού, με IPv4 διεύθυνση προελεύσεως την διεύθυνση του εξωτερικού interface του relay router.

3.7 ΘΕΜΑΤΑ ΑΣΦΑΛΕΙΑΣ ΤΟΥ 6TO4 ΜΗΧΑΝΙΣΜΟΥ

3.7.1 ΑΠΕΙΛΕΣ – ΕΙΔΗ ΕΠΙΘΕΣΕΩΝ

Όπως περιγράφτηκε πιο πάνω ο 6to4 μηχανισμός περιλαμβάνει 6to4 routers και 6to4 relay routers που δέχονται και ¨ξεφλουδίζουν¨ IPv4 protocol 41 πακέτα από οποιονδήποτε κόμβο μέσα στο Internet καθώς και πακέτα IPv6 (οι relay routers) από οποιονδήποτε IPv6 κόμβο. Αυτά κυρίως τα χαρακτηριστικά καθιστούν τον μηχανισμό - όπως και γενικά όλους τους μηχανισμούς automatic tunneling, οι οποίοι δεν γνωρίζουν εκ των πρότερων τον κόμβο προελεύσεως των πακέτων- ευάλωτο σε κακόβουλες επιθέσεις κυρίως Denial of Service (DoS) και διευκολύνουν το spoofing των IPv6 διευθύνσεων από κακόβουλους κόμβους. Και το πρόβλημα δεν είναι τόσο η ζημία που οι επιθέσεις αυτές μπορεί να προκαλέσουν στο μηχάνημα που υλοποιεί τον

6to4 μηχανισμό, όσο το γεγονός ότι ο μηχανισμός χρησιμοποιείται για την απόκρυψη των ιχνών του επιτιθέμενου.

Για την σωστή και ασφαλή υλοποίηση του 6to4 μηχανισμού πρέπει να λαμβάνονται σοβαρά υπόψη τα διάφορα είδη επιθέσεων στα οποία ο 6to4 μηχανισμός είναι ευάλωτος.
Συγκεκριμένα οι 6to4 μηχανισμοί μπορούν να υποστούν τις πιο κάτω απειλές:

- Επιθέσεις Άρνησης Υπηρεσίας (Denial-of-Service - DoS) κατά τις οποίες ο κακόβουλος κόμβος εμποδίζει τον κόμβο που δέχεται την επίθεση να επικοινωνήσει με άλλους κόμβους.
- Επιθέσεις Άρνησης Υπηρεσίας με 'ανάκλαση'(reflected Denial-of-Service) όπου ο κακόβουλος κόμβος αντανακλά το διακινούμενο φορτίο από ανυποψίαστους κόμβους σε ένα κόμβο (ο οποίος δέχεται την επίθεση), εμποδίζοντας έτσι την επικοινωνία του με άλλους κόμβους.
- 'Υποκλοπή υπηρεσιών' (service theft)κατά την οποία ο κακόβουλος κόμβος/site/χρηστής μπορεί να χρησιμοποιήσει υπηρεσίες του relay router χωρίς αρμοδιότητα. Συγκεκριμένα, ενώ ο διαχειριστής του relay router θέτει περιορισμούς έτσι ώστε μόνο περιορισμένα 6to4 sites και συγκεκριμένα IPv6 sites να χρησιμοποιούν τον relay ,οι χρηστές είτε με χρήση την IPv4 διεύθυνσης του relay (αντί της anycast στην οποία εφαρμόζονται οι περιορισμοί) είτε χρησιμοποιώντας άλλες μεθόδους αποκτούν πρόσβαση στον relay.
- Επιθέσεις με μηνύματα ND
- Τοπικές IPv4 Broadcast Επιθέσεις

Οι δυο τελευταίες επιθέσεις αντιμετωπίζονται εξ' ολοκλήρου με σωστή υλοποίηση των Sanity Checks, τους οποίους πραγματοποιούν οι περισσότερες σήμερα tunnel devices, και ουσιαστικά ο κίνδυνος τους πλέον δεν υφίσταται.

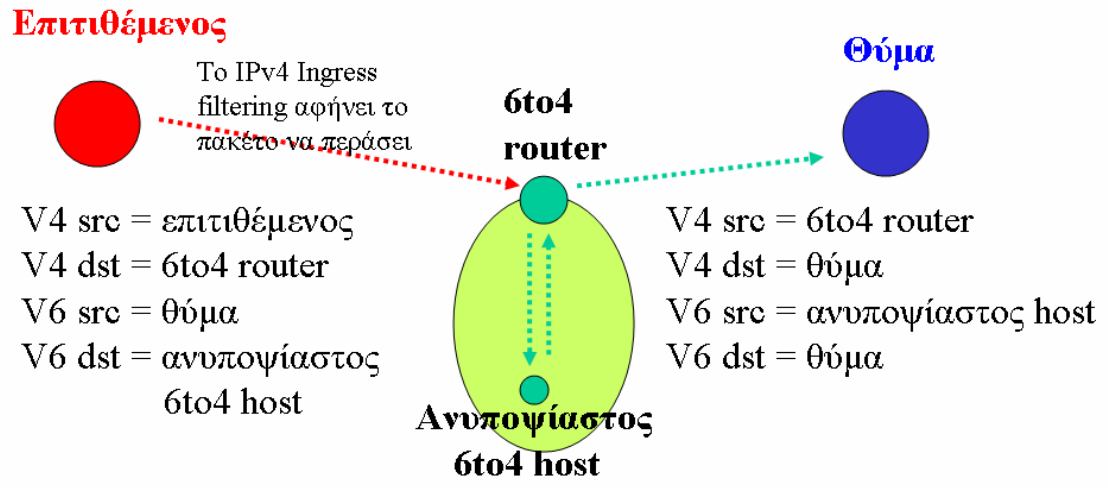
3.7.2 ΛΟΓΟΙ ΠΟΥ ΚΑΘΙΣΤΟΥΝ ΔΥΝΑΤΕΣ ΤΙΣ ΕΠΙΘΕΣΕΙΣ

- Οι 6to4 routers πρέπει να δέχονται και να απενθυλακώνουν IPv4 πακέτα από όλους τους relay routers. Και δεν είναι δυνατό να γνωρίζουν αν ο relay από τον οποίο προέρχονται τα πακέτα είναι έμπιστος και για την ακρίβεια ούτε καν αν είναι υπαρκτός. Αυτό είναι κυρίως το τρωτό σημείο του 6to4 μηχανισμού και για το οποίο έχει γίνει και συνεχίζει να γίνεται η περισσότερη ερευνά. Ο 6to4 router είναι αδύνατο να γνωρίζει την πραγματική διεύθυνση του relay router (που μπορεί να είναι οποιαδήποτε unicast διεύθυνση) που αναγράφεται στην εξωτερική IPv4 επικεφαλίδα ως η διεύθυνση προελεύσεως. Έτσι απλούστατα οποιοσδήποτε IPv4 ή IPv6 χρηστής μπορεί να στείλει πακέτα σε κάποιο 6to4 χρηστή, με οποιαδήποτε IPv6 διεύθυνση προελεύσεως αφού στην περίπτωση επικοινωνίας μέσω relay δεν μπορεί να γίνει καμία αντιστοίχηση.

- Οι relay routers πρέπει να δέχονται πακέτα από όλους τους 6to4 routers και επιπλέον από όλους τους native IPv6 κόμβους, τα οποία ανάλογα ενθυλακώνουν ή απενθυλακώνουν χωρίς να γίνεται κάποιος έλεγχος στα δεδομένα των πακέτων.

- Η ελλιπής, και πολλές φορές λανθασμένη υλοποίηση και διαχείριση των 6to4 και των Relay Routers, που αφήνει σημαντικά κενά ασφάλειας. Επιπλέον, Οι λανθασμένοι ή όχι πλήρεις έλεγχοι ασφάλειας που γίνονται στους δυο πιο πάνω δρομολογητές.

Πιο κάτω δίνεται ένα κλασικό παράδειγμα όπου ο επιτιθέμενος, ένας IPv4 χρηστής χρησιμοποιεί τον 6to4 μηχανισμό για να εξαφανίσει τα ίχνη του.



Σχήμα 25: Χρήση του 6to4 μηχανισμού σε επίθεση Reflected DoS

Για τον πρώτο θέμα ασφαλείας, που είναι και το μείζον πρόβλημα του μηχανισμού δεν έχει βρεθεί ακόμη κάποια οριστική λύση και η έρευνα συνεχίζεται. Στο RFC 3964 ("Security Considerations for 6to4") προτείνεται σαν η καλύτερη προς το παρών λύση όλοι οι relay routers να ρυθμίζονται και να διαφημίζουν το anycast prefix 192.88.99.0/24, καθιστώντας έτσι δυσκολότερο το spoofing των διευθύνσεων. Επιπλέον να εκτελούν ingress filtering στα πακέτα που προέρχονται από το IPv6 δίκτυο και να κάνουν τους έλεγχους ασφαλείας (Sanity Checks) που δίνονται πιο κάτω.

Όμως, η μεγαλύτερη σημασία για την αντιμετώπιση των ενδεχόμενων κενών ασφαλείας του μηχανισμού πρέπει να δοθεί στο στάδιο της υλοποίησης του και στην κατόπιν διαχείριση του. Σωστή υλοποίηση, διαχείριση και εφαρμογή των απαραίτητων έλεγχων θα ελαττώσει τους πιθανούς κίνδυνους και ουσιαστικά όπως έχει αποδειχτεί ένα σύστημα με σωστά και ολοκληρωμένα υλοποιημένο τον 6to4 μηχανισμό είναι τόσο ευάλωτο όσο ένα οποιοδήποτε άλλο σύστημα χωρίς 6to4.
Οι πιο κάτω έλεγχοι είναι απαραίτητοι για την ασφαλή λειτουργία:

3.7.3 Ασφάλεια 6to4 Router

Πρέπει να εφαρμόζει τους έλεγχους ασφάλειας καθώς και τα Sanity Checks στο διακινούμενο φορτίο που λαμβάνει τόσο από το τοπικό 6to4 site όσο και από τους

άλλους 6to4 routers και τους relay routers. Οι ελέγχοι αυτοί εκτός από την προστασία που παρέχουν ενάντια στα διάφορα είδη επιθέσεων, εγγυούνται και την καλή λειτουργία του μηχανισμού, ότι δηλαδή τα πακέτα από το 6to4 site που διαχειρίζεται ο εκάστοτε διαχειριστής δεν θα απορριφθούν από άλλους 6to4 ή relay routers που εκτελούν τους συγκεκριμένους ελέγχους. Στους έλεγχους πρέπει να περιλαμβάνονται τα πιο κάτω:

- Τόσο στη φάση της εισόδου, όσο και της εξόδου των πακέτων από το 6to4 interface όπου γίνεται η ενθυλάκωση/απενθυλάκωση, να ελέγχεται αν η IPv4 διεύθυνση (V4ADDR) που προκύπτει αν αφαιρεθεί το 2002::/16 από την 6to4 διεύθυνση (2002:V4ADDR:) ταιριάζει με την IPv4 διεύθυνση του εξωτερικού πακέτου (εκτός φυσικά όταν τα πακέτα προέρχονται από κάποιο relay router όπου αυτό δεν είναι δυνατό).

- **"Λογικοί έλεγχοι" ("Sanity Checks") IPv4**.
  Να μην επιτρέπονται οι διευθύνσεις οι οποίες είναι ιδιωτικές,broadcast ή δεσμευμένες. Συγκεκριμένα δεν θα επιτρέπεται η ενθυλάκωση και απενθυλάκωση των πιο κάτω IPv4 διευθύνσεων:

  - 0.0.0.0/8 (not assigned yet)
  - 10.0.0.0/8 (private)
  - 127.0.0.0/8 (loopback)
  - 172.16.0.0/12 (private)
  - 192.168.0.0/16 (private)
  - 169.254.0.0/16 (IANA Assigned DHCP link-local)
  - 224.0.0.0/4 (multicast)
  - 240.0.0.0/4 (reserved and broadcast)
  - Τοπικές broadcast διευθύνσεις

- **"Λογικοί έλεγχοι" ("Sanity Checks") IPv6**
  Να μην επιτρέπει την διακίνηση πακέτων όταν η IPv6 διεύθυνση προορισμού δεν είναι παγκόσμια unicast διεύθυνση. Συγκεκριμένα απορρίπτονται οι πιο κάτω διευθύνσεις:

  - 0::/16 (compatible, mapped addresses, loopback, unspecified, ...)
  - fe80::/10 (link-local)
  - fec0::/10 (site-local)
  - ff00::/8 (any multicast)

- Να δίνεται ιδιαίτερη προσοχή κατά τον καθορισμό των routes στον πίνακα δρομολόγησης έτσι ώστε πακέτα προς αλλά 6to4 sites να μην δρομολογούνται μέσω relay router.

- Να απορρίπτουν πακέτα που προέρχονται από άλλο 6to4 domain μεσώ relay router.

3.7.4 Ασφάλεια Relay Router

Πρέπει να εφαρμόζει έλεγχους ασφάλειας και τα Sanity Checks στο διακινούμενο φορτίο που λαμβάνει τόσο από το native IPv6 site όσο και από τους 6to4 routers. Στους έλεγχους πρέπει να περιλαμβάνονται τα πιο κάτω:

- Να μην επιτρέπει διακίνηση πακέτων που έχουν ιδιωτικές,broadcast ή δεσμευμένες IPv4 διευθύνσεις στο τούνελ ή στο 6to4 πρόθεμα τους.(οι συγκεκριμένες διευθύνσεις δίνονται πιο πάνω – IPv4 Sanity Checks)

- Να μην επιτρέπει διακίνηση πακέτων από 6to4 routers όταν η IPv4 διεύθυνση πηγής του τούνελ δεν ταιριάζει με την V4ADDR της 6to4 διεύθυνση(2002:V4ADDR) του ενθυλακωμένου πακέτου.

- Να εμποδίζει τα πακέτα με διεύθυνση προορισμού μη παγκόσμια IPv6 διεύθυνση (IPv6 Sanity Checks).

- Αποβάλλει πακέτα που προέρχονται από 6to4 routers και έχουν στη διεύθυνση προορισμού το πρόθεμα 6to4.

- Να φιλτράρει και να αποβάλλει τα πακέτα με protocol 41 τα οποία δεν έχουν την anycast διεύθυνση προορισμού 192.88.99.1 . (περίπτωση που ρυθμίζεται στην anycast διεύθυνση)

- Αν εξυπηρετεί μικρό αριθμό δικτύων, να χρησιμοποιεί access lists για έλεγχο της πρόσβασης.

- Χρήση Ingress Filtering για τα πακέτα από το IPv6 site.

3.7.5 ΓΕΝΙΚΕΥΣΗ ΚΑΝΟΝΩΝ ΑΣΦΑΛΕΙΑΣ

Δίνεται πιο κάτω (σε ψευδοκώδικα) η απαιτούμενη διαδικασία με τους έλεγχους ασφαλείας που πρέπει να εκτελούνται κατά την ενθυλάκωση και απενθυλάκωση των πακέτων:

Ενθυλάκωση IPv6 σε IPv4:

```
src_v6 and dst_v6 MUST pass ipv6-sanity checks /*έλεγχος αν δεν
                    είναι οι συγκεκριμένες IPv6 διευθύνσεις που
                    έχουν οριστεί πιο πάνω ότι απαγορεύονται */
else   drop
if prefix (src_v6) == 2002::/16
  ipv4 address embedded in src_v6 MUST match src_v4
else if prefix (dst_v6) == 2002::/16
      dst_v4 SHOULD NOT be assigned to the router
else
  drop
      /* πρόκειται για περίπτωση native-native ipv6 packet */
fi
```

```
accept
```

Απενθυλάκωση IPv4 σε IPv6:

```
src_v4 and dst_v4 MUST pass ipv4-sanity checks  /*έλεγχος αν
              δεν   είναι οι συγκεκριμένες IPv4 διευθύνσεις
              που έχουν οριστεί πιο πάνω ότι απαγορεύονται */
else drop

src_v6 and dst_v6 MUST pass ipv6-sanity checks, /*έλεγχος αν
               δεν   είναι οι συγκεκριμένες IPv4 διευθύνσεις
               που έχουν οριστεί πιο πάνω ότι απαγορεύονται */

if prefix (dst_v6) == 2002::/16
  ipv4 address embedded in dst_v6 MUST match dst_v4
      if prefix (src_v6) == 2002::/16
          ipv4 address embedded in src_v6 MUST match src_v4
          dst_v4 SHOULD be assigned to the router
      fi
elif prefix (src_v6) == 2002::/16
  ipv4 address embedded in src_v6 MUST match src_v4
  dst_v4 SHOULD be assigned to the router
else
  drop
      /* πρόκειται για περίπτωση native-native ipv6 packet */
fi
 accept
```

Όπου src_v4, src_v6 και dst_v4, dst_v6 οι διευθύνσεις πηγής και προορισμού για τα IPv4 και τα IPv6 πακέτα αντίστοιχα.

3.7.6 Επιπλέον Θέματα Ασφάλειας

- Για τη αποφυγή του spoofing του IPv6 φορτίου, συνίσταται χρήση IPv4 IPsec στην περίπτωση που δεν χρησιμοποιείται IPv6 IPSec. Έτσι, χρησιμοποιώντας IPsec authentication, εμποδίζεται η παραποίηση του ενθυλακωμένου(IPv6) φορτίου.

- Ο 6to4 μηχανισμός δεν επηρεάζεται από την χρήση firewall μπροστά από τον 6to4 router.

3.8 6to4 ΚΑΙ DNS

Οι 6to4 διευθύνσεις, όπως και όλες οι IPv6 διευθύνσεις, χρησιμοποιούν τον τύπο έγγραφης AAAA. Ο τύπος αυτός παρέχει ένα πλήρες domain name σε μια IPv6 διεύθυνση. Η δομή του τύπου έγγραφης AAAA φαίνεται πιο κάτω:

```
Name    IN    AAAA    Address
```

Ένας κόμβος μπορεί να έχει περισσότερες από μια AAAA έγγραφες και η επιλογή το ποιας θα χρησιμοποιηθεί γίνεται όπως περιγράφεται στην παράγραφο ¨Επιλογή κατάλληλης διευθύνσεως¨.

3.9 ΣΥΝΟΨΗ ΤΩΝ ΛΕΙΤΟΥΡΓΙΩΝ ΤΟΥ 6TO4 ΜΗΧΑΝΙΣΜΟΥ

3.9.1 Κανόνες Αποστόλης και Απενθυλάκωσης :

Συνοψίζοντας την λειτουργία του 6to4 μηχανισμού, η μόνη διάφορα από την standard προώθηση IPv6 πακέτων είναι οι παρακάτω κανόνες που υλοποιούνται στους 6to4 routers και καλύπτουν τις τρεις προαναφερθείσες περιπτώσεις:
Το "next-hop" αναφέρεται γενικά στους πίνακες δρομολόγησης σαν ο επόμενος IPv6 κόμβος που θα μεταδοθεί το πακέτο, όχι κατά ανάγκη ο κόμβος προορισμού αλλά ο επόμενος IPv6 γείτονας όπως έχει υπολογιστεί από τους κανονικούς IPv6 μηχανισμούς δρομολόγησης. Στην περίπτωση που ο τελικός προορισμός είναι 6to4 διεύθυνση, αυτή θα θεωρείται ο next-hop. Αν η τελική διεύθυνση δεν είναι 6to4 και επιπλέον δεν είναι τοπική, το next-hop θα είναι η διεύθυνση του relay router.

Η συνολική διαδικασία της ενθυλάκωση δίνεται (σε ψευδοκώδικα) πιο κάτω:

```
if the next hop IPv6 address for an IPv6 packet
does match the prefix 2002::/16, and
does not match any prefix of the local site
     then
          apply any security checks (δες επόμενη παράγραφο);
           encapsulate the packet in IPv4 (όπως περιγράφηκε σε
       προηγούμενη παράγραφο)
          with IPv4 destination address = the NLA value V4ADDR
          extracted from the next hop IPv6 address;
          queue the packet for IPv4 forwarding.
```

Και κατόπιν για την απενθυλάκωση των πακέτων:

```
apply any security checks (δες επόμενη παράγραφο);
    remove the IPv4 header;
    submit the packet to local IPv6 routing.
```

3.9.2 6TO4 ROUTER

Ο ***6to4 Router*** :

- Παρέχει native IPv6 σύνδεση στους τοπικούς hosts και routers.

- Αν τα πακέτα στέλνονται σε εκτός του site 6to4 διευθύνσεις, τα προωθεί μέσω τούνελ στον 6to4 router που αντιστοιχεί στην διεύθυνση προορισμού, χρησιμοποιώντας IPv4.
- Αν τα πακέτα έχουν διεύθυνση προορισμού κάποια τοπική 6to4 διεύθυνση, τα προωθεί κανονικά όπως ορίζει το IPv6 πρωτόκολλο επικοινωνίας.
- Αν τα πακέτα στέλνονται προς κάποια IPv6 διεύθυνση (όχι 6to4), τότε στέλνονται και πάλι μεσώ τούνελ στον Relay Router που έχει οριστεί ή στον πλησιέστερο όπως ορίζει η anycast διεύθυνση, ο οποίος προωθεί τα πακέτα στον προορισμό τους.
- Αν λάβει πακέτα από 6to4 διευθύνσεις , εκτελείτε στο 6to4 interface η απενθυλάκωση τους αν προέρχονται από 6to4 routers και έχουν περάσει τους έλεγχους ασφάλειας..
- Αν λάβει πακέτα από relay router, εκτελείτε στο 6to4 interface η απενθυλάκωση τους αν προέρχονται από τον πλησιέστερο relay router και έχουν περάσει τους έλεγχους ασφάλειας..

3.9.3 RELAY ROUTER

Ο ***Relay Router*** :

- Διαφημίζει το πρόθεμα 2002::/16 στο IPv6 routing domain, έτσι ώστε να στέλνονται σε αυτόν όλη κίνηση από τους native IPv6 κόμβους προς τα διάφορα 6to4 sites.
- (Αν υλοποιεί το RFC 3068) Διαφημίζει στο IPv4 routing domain το '6to4 Relay anycast prefix' (192.88.99.0/24), έτσι ώστε να δέχεται και να προωθεί την κίνηση μεσώ τούνελ από τους πλησιέστερους 6to4 routers, προς τους native IPv6 κόμβους.
- Αν τα πακέτα προέρχονται από 6to4 router, και έχουν περάσει τους κανόνες ασφάλειας, τα απενθυλακώνει και τα προωθεί με κανονική IPv6 δρομολόγηση.
- Αν τα πακέτα προέρχονται από native IPv6 διευθύνσεις και προορίζονται για 2002::/16 , τα προωθεί μεσώ τούνελ στον 6to4 router.

3.10 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ

Ο 6to4 μηχανισμός είναι ο πιο διαδεδομένος μηχανισμός μετάβασης στο παρόν στάδιο, και αυτό είναι φυσικό, αφού έχει να παραθέσει αρκετά πλεονεκτήματα έναντι των υπόλοιπων μηχανισμών. Διαθέτοντας έστω και μια unicast IPv4 διεύθυνση, και με ελάχιστο manual configuration, δίνεται η δυνατότητα σε ένα ολόκληρο site να αποκτήσει IPv6 σύνδεση. Και εκτός από αυτό, ο συγκεκριμένος μηχανισμός δεν χρειάζεται να υλοποιηθεί σε κάθε κόμβο του site, κάτι που θα ήταν δύσκολο και κοπιαστικό για τον διαχειριστή, αλλά περιορίζεται στην υλοποίηση του μόνο στον συνοριακό δρομολογητή και μέσω των διαφημίσεων που στέλνει και χρησιμοποιώντας τις διεργασίες του Stateless Address Autoconfiguration, οι hosts σχηματίζουν τις παγκόσμιες IPv6 6to4 διευθύνσεις. Και επιπλέον, με αυτόν τον τρόπο σχηματίζονται αμέσως κάτω από μια IPv4 διεύθυνση 2^{16} υποδίκτυα με 2^{64} κόμβους το καθένα, δίνοντας έτσι και ο ίδιος μηχανισμός μια λύση και στο πρόβλημα της

εξάντλησης των διευθύνσεων. Ακολούθως, ο 6to4 είναι ένας μηχανισμός που έχει γίνει standard (RFC 3056 – "Connection of IPv6 Domains via IPv4 Clouds"), έχει καθιερωθεί και υποστηρίζεται σήμερα από τα περισσότερα λειτουργικά συστήματα. Και τέλος, σήμερα υπάρχει μια πληθώρα relay routers, 'πρόθυμων' να εξυπηρετήσουν οποιονδήποτε 6to4 host, χωρίς κανένα κόστος.

Από την άλλη πλευρά, όσον αφορά τα 'αδύνατα' σημεία του 6to4 μηχανισμού, πρώτο είναι το θέμα της ασφάλειας και κυρίως η χρήση του ως εργαλείο στα χέρια κακόβουλων χρηστών για απόκρυψη των ιχνών τους. Αλλά, όπως αναλύθηκε πιο πάνω, στο κομμάτι για την ασφάλεια του 6to4, ο μηχανισμός με σωστή υλοποίηση και διαχείριση είναι τόσο ευάλωτος όσο και οποιαδήποτε άλλη συσκευή βρίσκεται στο διαδίκτυο.
Η άλλη αδυναμία του 6to4 είναι ότι ο μηχανισμός δεν μπορεί να υλοποιηθεί πίσω από NAT λόγω της απαίτησης του για μοναδική παγκόσμια διεύθυνση. Αν και μπορεί να υλοποιηθεί μέσα στο NAT box, εξωτερικά δηλαδή από το NAT και να παρέχει IPv6 σύνδεση στους κόμβους που βρίσκονται κάτω από το NAT, χωρίς μάλιστα να απαιτείται και 'μετάφραση' διευθύνσεων (address translation), το σενάριο αυτό δεν λειτουργεί κάτω από πολλαπλά επίπεδα NAT. Σε τέτοιες περιπτώσεις ο 6to4 μηχανισμός συνδυάζεται με κάποιον άλλο μηχανισμό μετάβασης, συνήθως ISATAP, παρέχοντας την εξωτερική IPv6 σύνδεση.

4. ΑΝΑΓΚΕΣ ΔΙΑΧΕΙΡΙΣΗΣ 6TO4 ΜΗΧΑΝΙΣΜΟΥ

Στα προηγούμενα παρουσιάστηκε ο μηχανισμός 6to4 και καθορίστηκαν οι διάφορες παράμετροι για την υλοποίηση και λειτουργία του. Ακολούθως καθορίζονται οι διαχειριστικές ανάγκες, που πρέπει να ληφθούν υπόψη από ένα διαχειριστή για την σωστή και ασφαλή λειτουργία τόσο του ίδιου του μηχανισμού όσο και του συστήματος στο οποίο λειτουργεί το 6to4 ειδικότερα. Συγκεκριμένα, στην παρούσα ενότητα περιγράφονται όλα όσα πρέπει να ληφθούν υπόψη από το διαχειριστή ενός μηχανήματος στο οποίο έχει ρυθμιστεί το 6to4 interface, τόσο στην περίπτωση που το μηχάνημα είναι ένας απλός host όσο και, στη σημαντικότερη περίπτωση που το μηχάνημα είναι ο 6to4 router του site ή κάποιος relay router.

Το σύνολο των διαχειριστικών αναγκών ενός 6to4 μηχανήματος μπορούν να χωριστούν σε τρεις κυρίως κατηγόριες: (α) οι ειδικές παράμετροι ρύθμισης με τις οποίες δημιουργείται και ενεργοποιείται το/τα 6to4 interface/s του συστήματος και (β) οι παράμετροι εκείνες που πρέπει να ελέγχονται από το διαχειριστή του 6to4. Οι παράμετροι της τελευταίας κατηγορίας χρησιμοποιούνται για να διαφυλαχθεί η δική του ασφάλεια αλλά και να εμποδιστεί η χρήση του μηχανισμού από άλλους (πιθανά κακόβουλους) χρηστές, αφορούν όμως και έλεγχους οι οποίοι θα αποτρέψουν την απόρριψη των πακέτων με προέλευση το δικό του μηχάνημα και προορισμό κάποιον άλλο 6to4 ή relay router. Τέλος, (γ) η τελευταία ομάδα διαχειριστικών αναγκών έχει να κάνει με την γενική διαχείριση του μηχανήματος στο οποίο έχει εγκατασταθεί ο μηχανισμός. Εδώ, καλύπτονται τα διάφορα ζητήματα που αφορούν στο αν το μηχάνημα είναι 6to4 host ή router ή relay, τα διάφορα ζητήματα των διεργασιών Neighbor Discovery και Router Discovery, γενικές πληροφορίες για το interface και στατιστικά στοιχεία. Η τελευταία αυτή ομάδα αναγκών δεν αφορά συγκεκριμένα τον 6to4 μηχανισμό, και απλά αναφέρονται συνοπτικά οι διάφορες παράμετροι που πρέπει να ληφθούν υπόψη από τον διαχειριστή του μηχανήματος στο οποίο είναι εγκατεστημένος ο 6to4, ειδικά στην περίπτωση που αυτό είναι δρομολογητής. Παρατίθενται στο σημείο αυτό για λόγους ολοκληρωμένης διαχείρισης.

4.1 Παράμετροι Ρύθμισης (Configuration Parameters)

Πρώτα από όλα, πρέπει να δίνεται η δυνατότητα στο διαχειριστή να ελέγχει και να τροποποιεί αν κρίνεται απαραίτητο τις παραμέτρους με τις οποίες ρυθμίστηκε (configure) το 6to4 pseudo-interface. Συγκεκριμένα, απαιτείται πρόσβαση και δυνατότητα μετατροπής των πιο κάτω:

- Έλεγχος κατά πόσο υπάρχει κάποιο 6to4 interface στο σύστημα.
- Αν όχι, να μπορεί να ελέγξει αν στο σύστημα υπάρχει εγκατεστημένη κατάλληλη IPv6-IPv4 tunnel device για δημιουργία τούνελ μεταξύ δυο IPv6 κόμβων και επικοινωνία πάνω από το IPv4 δίκτυο, έτσι ώστε να μπορεί είτε μεσώ SNMP είτε με manual configuration να δημιουργήσει ένα 6to4 interface, παρέχοντας τη δυνατότητα για επικοινωνία με το native IPv6 δίκτυο ή με αλλά 6to4 sites. Ουσιαστικά, ύπαρξη της tunnel device φανερώνει ότι ο κόμβος είναι dual-stack, δηλαδή έχει εγκατεστημένες και τις δυο στοίβες πρωτοκόλλων, που είναι απαραίτητη προϋπόθεση για να είναι δυνατή η

λειτουργία γενικά των μηχανισμών IPv6-over-IPv4 tunneling, στους οποίους συγκαταλέγεται και ο 6to4.

- Στην περίπτωση που υπάρχει το 6to4 interface, να δίνεται η κατάσταση του, αν δηλαδή το εν λόγω interface είναι ‘ανεβασμένο’ και διακινεί πακέτα ή ‘κατεβασμένο’, καθώς και η δυνατότητα να μεταβάλει την κατάσταση του. Επιπλέον, στην περίπτωση που έχουν εγκατασταθεί και ρυθμιστεί στο σύστημα περισσότερα από ένα 6to4 interfaces, να επιστρέφει τον αριθμό τους καθώς και την κατάσταση λειτουργίας για το κάθε interface. Γενικά, (από το RFC 3056 για το 6to4)δεν συνίσταται η υλοποίηση περισσότερων του ενός 6to4 interfaces, κάτι το οποίο μόνο προβλήματα έχει προκαλέσει σε όσες περιπτώσεις γίνεται (απόρριψη πακέτων λόγω αδυναμίας ταύτισης της IPv4 διευθύνσεως του interface με την V4ADDR που περιεχόταν στην IPv6 διεύθυνση λόγω των πολλαπλών IPv4 διευθύνσεων που χρησιμοποιούνταν από τα interfaces). Παρόλα αυτά ο διαχειριστής μπορεί να εγκαταστήσει τόσα 6to4 interfaces όσες και οι παγκόσμιες unicast διευθύνσεις IPv4 που διαθέτει στο συγκεκριμένο μηχάνημα.
- Την IPv4 διεύθυνση του site, η οποία θα είναι η εξωτερική διεύθυνση του interface για την επικοινωνία του, πάνω από την IPv4 υποδομή δρομολόγησης, με τα υπόλοιπα 6to4 και native IPv6 sites. Αυτή είναι η διεύθυνση στην οποία γίνεται configure το 6to4 interface και απαραίτητη προϋπόθεση για να λειτουργήσει ο 6to4 μηχανισμός είναι η διεύθυνση αυτή να είναι παγκόσμια(global), μοναδική και unicast. Επίσης, πρέπει να ληφθεί υπόψη ότι σε κάθε IPv4 διεύθυνση αντιστοιχεί ένα μοναδικό 6to4 interface, και γενικά ένα μόνο tunnel interface ανά IPv4 διεύθυνση, γιατί στην αντίθετη περίπτωση τα πακέτα απορρίπτονται.
- Την 6to4 διεύθυνση του site. Πρόκειται για μια IPv6 διεύθυνση, που σχηματίζεται από το 6to4 πρόθεμα 2002::/16 και την IPv4 διεύθυνση που έχει ανατεθεί στο 6to4 interface, της μορφής 2002:V4ADDR::/48. Στην περίπτωση 6to4 site, η διεύθυνση αυτή είναι το παγκόσμια μοναδικό πρόθεμα του site, με το οποίο ο ‘έξω κόσμος’ γνωρίζει το site, και είναι αυτό που διαφημίζεται από τους 6to4 routers. Στην περίπτωση ενός μοναδικού 6to4 host, ο οποίος έχει εγκατεστημένο τον 6to4 μηχανισμό η διεύθυνση είναι συνήθως της μορφής 2002:V4ADDR::1/128, αν και κάποια λειτουργικά ακολουθούν διαφορετική υλοποίηση (π.χ. 2002:V4ADDR:: V4ADDR/128)
- Τον/τους Relay Routers που θα χρησιμοποιηθούν σε περίπτωση που ο κόμβος προορισμού ανήκει σε native IPv6 domain. Σήμερα υπάρχει διαθέσιμη πληθώρα Relay Routers που μπορούν να χρησιμοποιηθούν. Εναλλακτικά μπορεί να χρησιμοποιηθεί και η anycast διεύθυνση 192.88.99.0/24 (RFC 3068) , για επιλογή του πλησιέστερου Relay για την μετάδοση των πακέτων. Ο διαχειριστής πρέπει να έχει την δυνατότητα να διαχειρίζεται τα διάφορα routes προς τους relay routers, να αλλάζει το δίκτυο προορισμού και τις διευθύνσεις των relays και γενικά να μπορεί είτε να προσθέσει είτε να αφαιρέσει routes για τα πακέτα προς κάποια native IPv6 sites, που η δρομολόγηση σε αυτά επιβάλλεται να γίνεται μέσω κάποιου relay.
- Επιπλέον, ο διαχειριστής πρέπει να θέσει το TTL(Time To Live) της εξωτερικής IPv4 επικεφαλίδας με τιμή πεδίου protocol 41, κάτω από την οποία ενθυλακώνεται το 6to4 πακέτο για την μετάδοση του. Γενικά, τα tunnels κατατάσσονται στην κατηγόρια των ‘single hop’, δηλαδή το Hop Limit της IPv6 επικεφαλίδας μειώνεται κατά ένα κατά την διάβαση από το ένα άκρο του tunnel στο άλλο, οποίος και αν είναι ο αριθμός των ενδιάμεσων

κόμβων μέσα στο tunnel. Η τιμή του TTL του εξωτερικού πακέτου είναι ευθύνη του διαχειριστή, αλλά και στην περίπτωση που δεν δηλωθεί στα περισσότερα συστήματα παίρνει αυτόματα την τιμή του Hop Limit της IPv6 επικεφαλίδας.

- Τέλος, το MTU που θα χρησιμοποιηθεί κατά μήκος του tunnel, είναι ένας σημαντικός παράγοντας για τον 6to4 μηχανισμό και για τα tunnels γενικότερα. Γενικά στα tunnels, ο τεμαχισμός των πακέτων είναι ανεπιθύμητος, αλλά και ειδικότερα στην περίπτωση του 6to4 (και συγκεκριμένα όταν χρησιμοποιείται η anycast διεύθυνση για την επικοινωνία με τον πλησιέστερο relay router) υπάρχει το ενδεχόμενο τα πακέτα να προωθηθούν από διαφορετικούς relays, και να μην είναι δυνατή η επανασυναρμολόγηση τους στον προορισμό και να τελικά να απορριφθούν. Για αυτό ο διαχειριστής του μηχανισμού πρέπει είτε να καθορίσει το MTU στο αντίστοιχο του επιπέδου ζεύξης που βρίσκεται κάτω από το στρώμα δικτύου είτε να χρησιμοποιήσει το IPv4 Path MTU Discovery Protocol, για να υπολογίζει κάθε φορά το ιδανικό MTU που θα χρησιμοποιήσει. Φυσικά, το τελευταίο είναι δύσκολο στην περίπτωση που έχουν να υλοποιηθούν πολλά tunnels, και για αυτό σε τέτοιες περιπτώσεις χρησιμοποιείται η πρώτη λύση, με τον καθορισμό του MTU του στρώματος ζεύξης. Όμως, και για τις δυο περιπτώσεις, αν το MTU που επιστρέφεται είναι μικρότερο από 1280, που είναι το μικρότερο μέγεθος πακέτου για το IPv6, η τιμή του MTU πρέπει να τεθεί σε 1280 και το "Do Not Fragment" bit να μην τίθεται σε '1'.

4.2 Ανάγκες Ασφαλείας και Σωστής Λειτουργίας του Μηχανισμού

Δεύτερος σημαντικός τομέας στον οποίο διαχειριστικά πρέπει να δοθεί ιδιαίτερη σημασία είναι η ασφάλεια του 6to4 μηχανισμού και η σωστή λειτουργία του. Με τον όρο σωστή λειτουργία αναφερόμαστε σε όλους εκείνους τους έλεγχους που θα διασφαλίσουν ότι τα πακέτα που θα προωθηθούν από το δικό μας μηχάνημα δεν θα απορριφθούν από κάποιο 6to4 ή relay router προορισμού που εκτελεί σωστά όλους τους απαραίτητους έλεγχους(Sanity Checks). Και επιπλέον, χρήσιμο στην περίπτωση του 6to4 μηχανισμού, που πρόκειται για ένα μηχανισμό automatic tunneling, όπου τα tunnel εγκαθίστανται και αποδεσμεύονται δυναμικά χωρίς να φυλάσσεται κάποια πληροφορία, είναι η καταγραφή κάποιων στατιστικών στοιχείων για τα tunnels που είχαν εγκατασταθεί στο σύστημα, πράγμα το οποίο θα βοηθούσε στον εντοπισμό των ιχνών του επιτιθεμένου και θα αναιρούσε τις αδυναμίες του 6to4 μηχανισμού στο συγκεκριμένο ζήτημα.
Λόγω της φύσης του μηχανισμού, να δέχεται δηλαδή πακέτα πρωτοκόλλου 41 από όλους τους κόμβους, είναι ευάλωτος σε κακόβουλους χρηστές, οι οποίοι παραποιώντας το IPv6 φορτίο πραγματοποιούν επιθέσεις DoS και reflected DoS. Οι συγκεκριμένες επιθέσεις επηρεάζουν το ίδιο το σύστημα 6to4 σε μικρότερο βαθμό αφού η ίδια η λειτουργία του μηχανισμού, λόγω της επιβάρυνσης στην απόδοση που προκαλεί η ενθυλάκωση/απενθυλάκωση, τις περιορίζει. Το σημαντικό πρόβλημα είναι ότι οι επιτιθέμενοι χρησιμοποιούν τον 6to4 μηχανισμό για να επιτεθούν σε άλλους κόμβους αποκρύπτοντας τα ίχνη τους.
Επιπλέον, μη υλοποίηση των διαφόρων ελέγχων (Sanity Checks) καθιστούν το 6to4 μηχανισμό ευάλωτο σε αλλού είδους επιθέσεις όπως Επιθέσεις με ND (Neighbor Discovery) μηνύματα ή επιθέσεις Local IPv4 Broadcast.

Παρακάτω δίνονται οι δυο μεγαλύτερες πηγές κινδύνου για τον 6to4 μηχανισμό, καθώς και οι ενέργειες του διαχειριστή του συστήματος για περιορισμό των διαφόρων κενών ασφαλείας που προκύπτουν:

- Το κυριότερο πρόβλημα είναι η αδυναμία του 6to4 router (και γενικά του κάθε 6to4 host) να γνωρίζει αν ο relay router που του στέλνει τα πακέτα είναι έμπιστος. Δεν είναι δυνατόν για ένα 6to4 router να ταιριάξει την IPv4 διεύθυνση του Relay Router με την IPv6 διεύθυνση της πηγής, αφού ο relay router μπορεί να έχει οποιαδήποτε IPv4 διεύθυνση. Άρα μπορεί απλούστατα η IPv6 διεύθυνση προελεύσεως να υποστεί 'spoofing'. Υπάρχει δηλαδή το ενδεχόμενο να παραποιηθεί και να μετατραπεί στη διεύθυνση του υποψήφιου θύματος ή σε οποιανδήποτε άλλη IPv6 διεύθυνση, καθιστώντας αδύνατο τον εντοπισμό του κόμβου που αρχικά προκάλεσε κάποια επίθεση. Και αν η συγκεκριμένη διεύθυνση αποσταλεί σε πολλούς 6to4 routers, προκύπτει αμέσως μια επίθεση άρνησης υπηρεσίας "με ανάκλαση" (reflected DoS) στο συγκεκριμένο θύμα. Επιπλέον, μέχρι σήμερα δεν απαιτούνταν μηχανισμοί εμπιστοσύνης ανάμεσα σε ένα 6to4 και ένα relay router, καθιστώντας έτσι αδύνατο για τον πρώτο να γνωρίζει αν πρόκειται για έμπιστο ή παράνομο (ή ακόμα και ανύπαρκτο) relay router. Η αντιμετώπιση του συγκεκριμένου προβλήματος είναι ακόμη σε ερευνητικό στάδιο και έχουν προταθεί κάποιες προσωρινές (και όχι εντελώς αποτελεσματικές) λύσεις, όπως το να γίνονται configure όλοι οι relay routers με την anycast διεύθυνση 192:88:99:0/24 καθιστώντας έτσι δυσκολότερο το 'spoofing' (RFC 3964). Παρόλα αυτά, ο διαχειριστής του 6to4 μηχανισμού, πρέπει να επιτρέπει την υποστήριξη συγκεκριμένων relay routers μόνο στις περιπτώσεις που η επικοινωνία γίνεται με έμπιστα native IPv6 sites και συγκεκριμένους IPv6 hosts και πάντα αφότου γίνει το απαραίτητο φιλτράρισμα εισόδου (ingress filtering) και αποκοπή των διευθύνσεων που ορίζονται στην επόμενη παράγραφο. Επίσης, τα πακέτα που λαμβάνονται από τους relay routers πρέπει να ελέγχονται έτσι ώστε να εξακριβώνεται ότι δεν προέρχονται από κάποιο άλλο 6to4 site. Επιπρόσθετα, κατά την υλοποίηση του 6to4 interface και τον καθορισμό των routes, πρέπει να καθοριστεί με σαφήνεια ότι ο 6to4 router θα προωθεί προς τους relay routers μόνο τα πακέτα που έχουν σαν διεύθυνση προορισμού native IPv6 domains και όχι αλλά 6to4 sites (2002::/16).

- Το δεύτερο πρόβλημα, του οποίου η επίλυση επαφίεται καθαρά στον διαχειριστή, αφορά στην λανθασμένη ή μη ολοκληρωμένη υλοποίηση του μηχανισμού, αφήνοντας ανοικτές 'πόρτες' σε κακόβουλους χρηστές για πρόκληση ζημιών στο σύστημα. Συγκεκριμένα, τόσο στη φάση της εισόδου, όσο και της εξόδου των πακέτων από το 6to4 interface όπου γίνεται η ενθυλάκωση/απενθυλάκωση, να ελέγχεται αν η IPv4 διεύθυνση (V4ADDR) που προκύπτει αν αφαιρεθεί το 2002::/16 από την 6to4 διεύθυνση (2002:V4ADDR:) ταιριάζει με την IPv4 διεύθυνση του εξωτερικού πακέτου (εκτός φυσικά όταν τα πακέτα προέρχονται από κάποιο relay router όπου αυτό δεν είναι δυνατό). Επιπλέον κατά το στάδιο της ενθυλάκωσης και της απενθυλάκωσης των πακέτων στο 6to4 interface, πρέπει να ελέγχονται και να απορρίπτονται πακέτα των οποίων η IPv4 διεύθυνση δεν είναι παγκόσμια unicast και συγκεκριμένα να απορρίπτονται οι πιο κάτω περιπτώσεις:

IPv4 Sanity Checks

- `0.0.0.0/8` (the system has no address assigned yet)
- `10.0.0.0/8` (private)
- `127.0.0.0/8` (loopback)
- `172.16.0.0/12` (private)
- `192.168.0.0/16` (private)
- `169.254.0.0/16` (IANA Assigned DHCP link-local)
- `224.0.0.0/4` (multicast)
- `255.0.0.0/8` (broadcast)

Και οι πιο κάτω IPv6 διευθύνσεις:

IPv6 Sanity Checks

- `0::/16 (compatible, mapped addresses, loopback, unspecified, ...)`
- `fe80::/10 (link-local)`
- `fec0::/10 (site-local)`
- `ff00::/8 (any multicast)`

Οι περισσότεροι από τους έλεγχους υλοποιούνται στους μηχανισμούς tunneling του εκάστοτε λειτουργικού. Για τους υπόλοιπους προτείνεται η εφαρμογή Access Control Lists (ACLs) για αποκοπή των συγκεκριμένων διευθύνσεων. Επιπλέον, εκτός από τους πιο πάνω έλεγχους (Sanity Checks), οι ACLs πρέπει να χρησιμοποιούνται και για αποκοπή ανεπιθύμητων διευθύνσεων και δικτύων από τα οποία έχουν στο παρελθόν προκληθεί επιθέσεις.

Επιπλέον, μια καλή λύση για προστασία του IPv6 φορτίου και αποφυγής του 'spoofing' είναι η χρήση IPv4 IPsec κατά μετάδοση των πακέτων πάνω από την IPv4 υποδομή - στην περίπτωση φυσικά που δεν χρησιμοποιείται IPv6 IPsec - αφού σε αυτή την περίπτωση το IPv4 IPSec δεν προσφέρει κάποια ουσιαστική προστασία παρά μάλλον επιπλέον επιβάρυνση -. Έτσι χρησιμοποιώντας IPsec επιβεβαίωση ταυτότητας (authentication) και κρυπτογράφηση εμποδίζεται η παραποίηση του IPv6 φορτίου.

Τέλος, ιδιαίτερα χρήσιμο για την περίπτωση του 6to4 μηχανισμού, και γενικά όταν υπάρχει θέμα διαχείρισης μηχανισμών automatic tunneling, είναι η διατήρηση κάποιων στατιστικών στοιχείων για τα tunnels που είχαν εγκατασταθεί μεταξύ του δικού μας συστήματος και του αλλού άκρου που είναι είτε κάποιος 6to4 είτε relay router. Έτσι, με αυτό τον τρόπο αίρεται μια μεγάλη αδυναμία των συγκεκριμένων μηχανισμών και καθίσταται σε μεγάλο βαθμό δυνατός ο εντοπισμός των ιχνών του επιτιθεμένου κακόβουλου κόμβου. Τα στοιχεία αυτά θα κρατούνται για ένα ορισμένο χρονικό διάστημα το οποίο θα ορίζει ο διαχειριστής.
Συγκεκριμένα, ο διαχειριστής του 6to4 site πρέπει να διατηρεί για το κάθε τούνελ που εγκαταστάθηκε από το σύστημα τα πιο κάτω στοιχεία:

- Την διεύθυνση του απομακρυσμένου χρηστή με τον οποίο έγινε η επικοινωνία. Η συγκεκριμένη διεύθυνση θα είναι είτε 6to4 (2002::/16) διεύθυνση στην περίπτωση 6to4 host, είτε κάποια διεύθυνση με παγκόσμιο unicast TLA (Top

Level Aggretator) όπως 2001::/16 ή 3ffe::/16, στην περίπτωση κάποιου host από το native IPv6 domain.

- Την διεύθυνση του 6to4 χρηστή, που ανήκει στο τοπικό 6to4 site, ο οποίος χρησιμοποίησε το tunnel. Η διεύθυνση χρειάζεται για τον εντοπισμό κακόβουλων χρηστών που προέρχονται μέσα από το τοπικό 6to4 site, (κάτι που δεν χρειάζεται στην περίπτωση που ο 6to4 router εκτελεί το ingress filtering και επιτρέπει μόνο σε έγκυρα πακέτα να εξέλθουν το site) αλλά κυρίως για σύγκριση και αντιστοίχηση των συγκεκριμένων στατιστικών στοιχείων με αυτά που έχουν καταγράφει για κάποια επίθεση.
- Την IPv4 διεύθυνση του ενδιάμεσου relay router, στην περίπτωση που η επικοινωνία έγινε με κάποιο χρηστή που ανήκει στο native IPv6 domain. Τα συγκεκριμένα στοιχεία είναι ιδιαίτερα χρήσιμα τόσο για τον εντοπισμό κακόβουλων ή ελλιπώς υλοποιημένων relay routers (ώστε μέσω των ACLs να απαγορευτεί η επικοινωνία με αυτούς) όσο και για τον εντοπισμό κακόβουλων IPv4 χρηστών, οι οποίοι 'παριστάνουν' τους relays, ενθυλακώνουν ένα πακέτο με 'ψεύτικη' IPv6 διεύθυνση και το προωθούν σε κάποιο/ους 6to4 routers.
- Τέλος, είναι σημαντικό να κρατούνται οι χρονικές στιγμές κατά τις οποίες έγινε η εγκατάσταση του tunnel και ολοκληρώθηκε η επικοινωνία, έτσι ώστε να είναι δυνατός ο εντοπισμός και η ταύτιση των έγγραφων με τις περιπτώσεις επίθεσης.

4.3 Γενικές Παράμετροι Διαχείρισης 6to4 router/host

Η τρίτη κατηγορία αναγκών διαχείρισης δεν έχει να κάνει με καθ' αυτό τον 6to4 μηχανισμό, δηλαδή την καθαυτή λειτουργία του 6to4, αλλά με τις επιπλέον διαχειριστικές ανάγκες ενός 6to4 router, ενός 6to4 relay router ή ενός απλού 6to4 host (ο οποίος έχει εγκατεστημένο τον 6to4 μηχανισμό και δεν ανήκει σε κάποιο 6to4 site) γενικότερα. Τα στοιχεία αυτά είναι σημαντικά για τον διαχειριστή του μηχανήματος το οποίο υποστηρίζει 6to4, και ιδιαίτερα στην περίπτωση διαχείρισης ενός 6to4 ή relay router.
Όπως αναφέρθηκε στις προηγούμενες ενότητες αυτής της εργασίας, στο στάδιο της μελέτης του 6to4 μηχανισμού, το 6to4 site είναι καθ' όλα ένα IPv6 site και η μόνη ιδιαιτερότητα είναι ότι τα πακέτα στέλνονται στον συνοριακό δρομολογητή για ενθυλάκωση/απενθυλάκωση. Ο συνοριακός δρομολογητής, ο 6to4 router, είναι και αυτός στην ουσία ένας IPv6 δρομολογητής χωρίς όμως native IPv6 συνδεσιμότητα. Η μόνη του διαφορά είναι το 6to4 pseudo-interface στο οποίο στέλνονται τα πακέτα (εξερχόμενα) για να ενθυλακωθούν κάτω από την IPv4 επικεφαλίδα και να προωθηθούν στον προορισμό τους, είτε (εισερχόμενα) για να 'ξεφλουδιστούν' και να σταλούν στο 6to4 site. Άρα, εκτός από την ειδική διαχείριση του 6to4 interface, η οποία καλύπτεται από τις δυο προηγούμενες κατηγορίες που εξετάστηκαν , ο επιπλέον χειρισμός του 6to4 μηχανήματος γίνεται όπως ακριβώς για μηχανήματα που υποστηρίζουν το πρωτόκολλο IPv6. Οι πληροφορίες που χρειάζονται για την συγκεκριμένη κατηγορία διαχείρισης παρέχονται από την **IPv6-MIB**, η οποία συμπληρώνει την IP-MIB και τοποθετήθηκε κάτω από την MIB-II. Τα αντικείμενα και οι πίνακες που αναφέρονται στην συνέχεια ανήκουν στην IPv6-MIB.

Η πρώτη σημαντική πληροφορία παρέχεται από το αντικείμενο *ip6Forwarding* το οποίο καθορίζει αν το συγκεκριμένο μηχάνημα είναι 6to4 ή relay router ή κάποιος host ο οποίος έχει εγκαταστήσει το 6to4 για να αποκτήσει IPv6 συνδεσιμότητα. Ακολούθως, γίνεται ο έλεγχος στο *ipAddressTable* αν στο μηχάνημα εκτός από την 6to4 διεύθυνση έχει ανατεθεί και κάποια native IPv6 διεύθυνση (2001::/16) που σημαίνει ότι πρόκειται στην ουσία για κάποιο relay router.

Ιδιαίτερα σημαντικά για την περίπτωση που το μηχάνημα είναι ένας 6to4 router είναι:

- η διαχείριση των περιοδικών διαφημίσεων του router
- του διαστήματος επανάληψης της αποστολής τους (Advertisements Intervals)
- το διάστημα που θα μεσολαβήσει για την έναρξη των αποστολών αφότου σηκωθεί το 6to4 interface (Advertisements Delay)
- οι διάφορες τιμές που στέλνει στους hosts για να καθορίσουν παραμέτρους όπως :
    - το MTU,
    - το Hop Limit,
    - το address prefix για σχηματισμό της παγκόσμιας διεύθυνσης τους,
    - το κατά πόσο θα χρησιμοποιήσουν stateless ή stateful address autoconfiguration,
    - το routing preference (στην περίπτωση του 6to4 router, το αντίστοιχο route πρέπει να τεθεί σε ‘προτίμηση’ 1)
    - κατά πόσο θα χρησιμοποιεί "DAD-Transmits" (Duplicate Address Detection) για πιστοποίηση της μοναδικότητας της link-local διευθύνσεως.

Οι πληροφορίες αυτές παρέχονται από τους πίνακες *ipv6AddrPrefixTable, ipv6AddrTable, ipv6RouterAdvertTable και ipDefaultRouterTable*.

Επίσης, στατιστικά στοιχεία που θα ήταν χρήσιμα για τον διαχειριστή και που αφορούν το 6to4 interface, καλύπτονται από τον πίνακα *ipv6IfStatsTable*. Συγκεκριμένα, μέσα από τις έγγραφες του *ipv6IfStatsTable* μπορούν να ληφθούν στοιχεία για:

- το ποσά πακέτα έφτασαν στο 6to4 interface
- ποσά πακέτα τελικά απορρίφθηκαν λόγω διεύθυνσης προορισμού ή πηγής που δεν τηρούσε τους κανόνες ασφαλείας
- ποσά πακέτα απορρίφθηκαν γιατί δεν είχαν την τιμή 41 στο πεδίο protocol της IPv4 επικεφαλίδας
- ποσά πακέτα τελικά ενθυλακώθηκαν ή απενθυλακώθηκαν και αναλόγως, αν προορίζονταν για άλλο κόμβο, προωθήθηκαν στον προορισμό τους.

Τέλος, στοιχεία που αφορούν τα ICMP μηνύματα που προωθήθηκαν από ή έφτασαν στο 6to4 interface καταγράφονται στον πίνακα *inetIcmpMsgTable*.

5. ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ 6TO4 MIB

Αφού καθορίστηκαν οι διαχειριστικές ανάγκες του 6to4 μηχανισμού, ακολουθεί η εύρεση των αντικείμενων εκείνων τα οποία θα συγκροτήσουν την 6to4 MIB. Μέσω αυτών ο διαχειριστής του συστήματος θα μπορεί να διαχειρίζεται αλλά και να κάνει configure 6to4 interfaces και τα επιμέρους χαρακτηριστικά τους με τη χρήση του πρωτοκόλλου διαχείρισης SNMP. Η υλοποίηση της MIB έγινε βάση του πρωτοκόλλου SNMPv2 και τα αντικείμενα της τρίτης κατηγορίας των γενικών διαχειριστικών αναγκών του μηχανήματος στο οποίο βρίσκεται εγκατεστημένος ο 6to4 δεν χρησιμοποιήθηκαν στη συγκεκριμένη MIB αφού καλύπτονται από άλλες MIBs και κυρίως την IPv6-MIB.

Η MIB αποτελείται ουσιαστικά από δυο μέρη. Το πρώτο είναι οι configuration παράμετροι βάσει των οποίων δημιουργήθηκε και 'λειτουργεί' το/τα 6to4 interface/s και καθορίζοντας τις ο διαχειριστής μπορεί να δημιουργήσει ή να διαγράψει ένα ή περισσότερα 6to4 interfaces μέσω SNMP. Το κομμάτι αυτό της MIB αποτελείται από δυο πίνακες:

- τον ***sixtofourIfTable***, στον οποίο καθορίζονται τα στοιχεία του κάθε 6to4 interface, με δυνατότητα μετατροπής τους καθώς και προσθήκης/διαγραφής interfaces

- τον πίνακα ***sixtofourRelayTable***, όπου γίνεται η διαχείριση των διάφορων routes προς τους relay routers. Τα routes αυτά είτε manual είτε μέσω SNMP έχουν καθοριστεί από τον διαχειριστή στον πίνακα δρομολόγησης του μηχανήματος που υλοποιεί το 6to4.

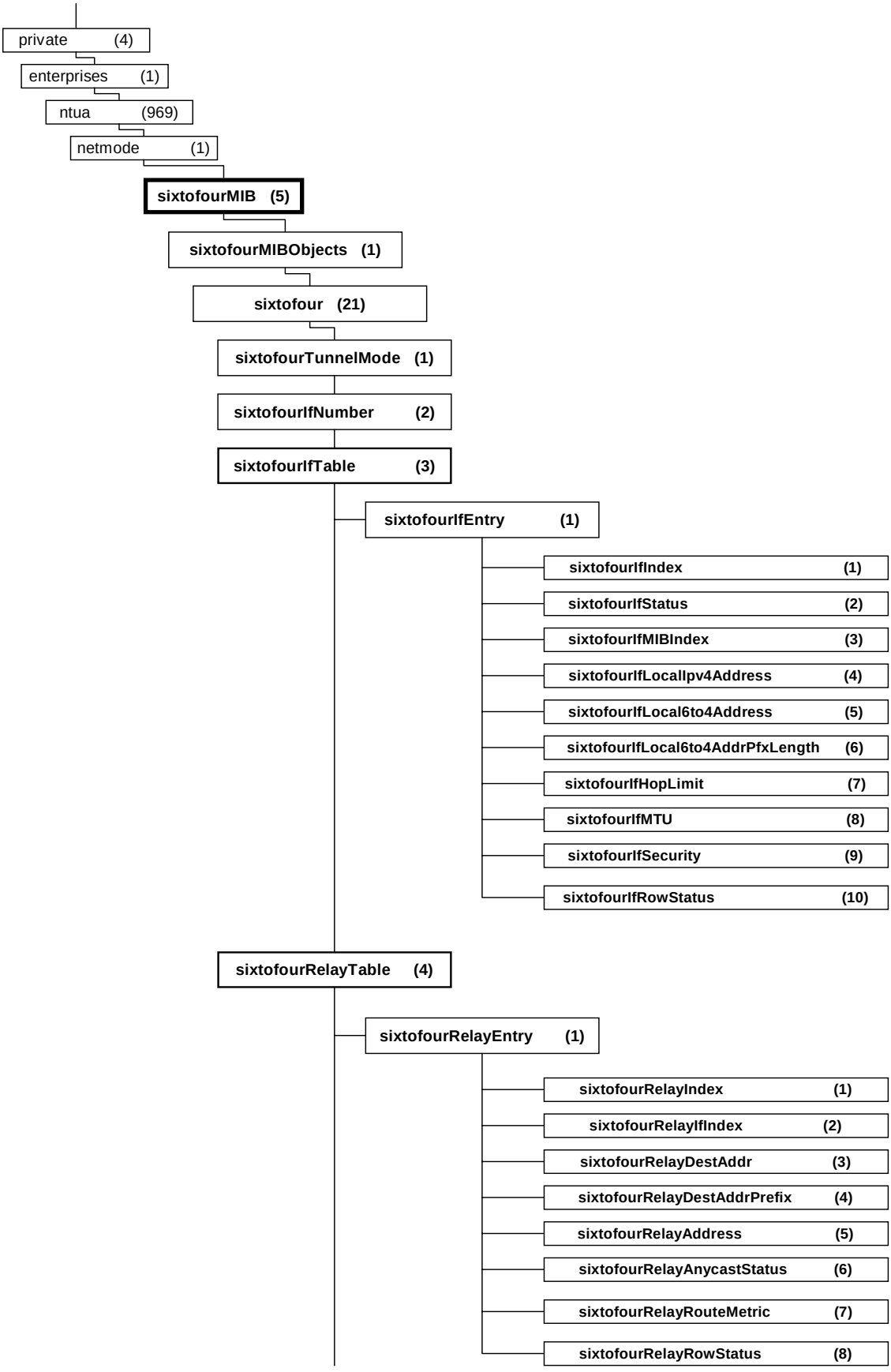
Επιπλέον, στο πρώτο κομμάτι, πριν τους δυο πίνακες, ορίζονται δυο ανεξάρτητα scalar αντικείμενα, το πρώτο καταδεικνύει το κατά πόσο υπάρχει η δυνατότητα υλοποίησης 6to4 interfaces, και το δεύτερο το αριθμό των interfaces που έχουν γίνει configure στο σύστημα.

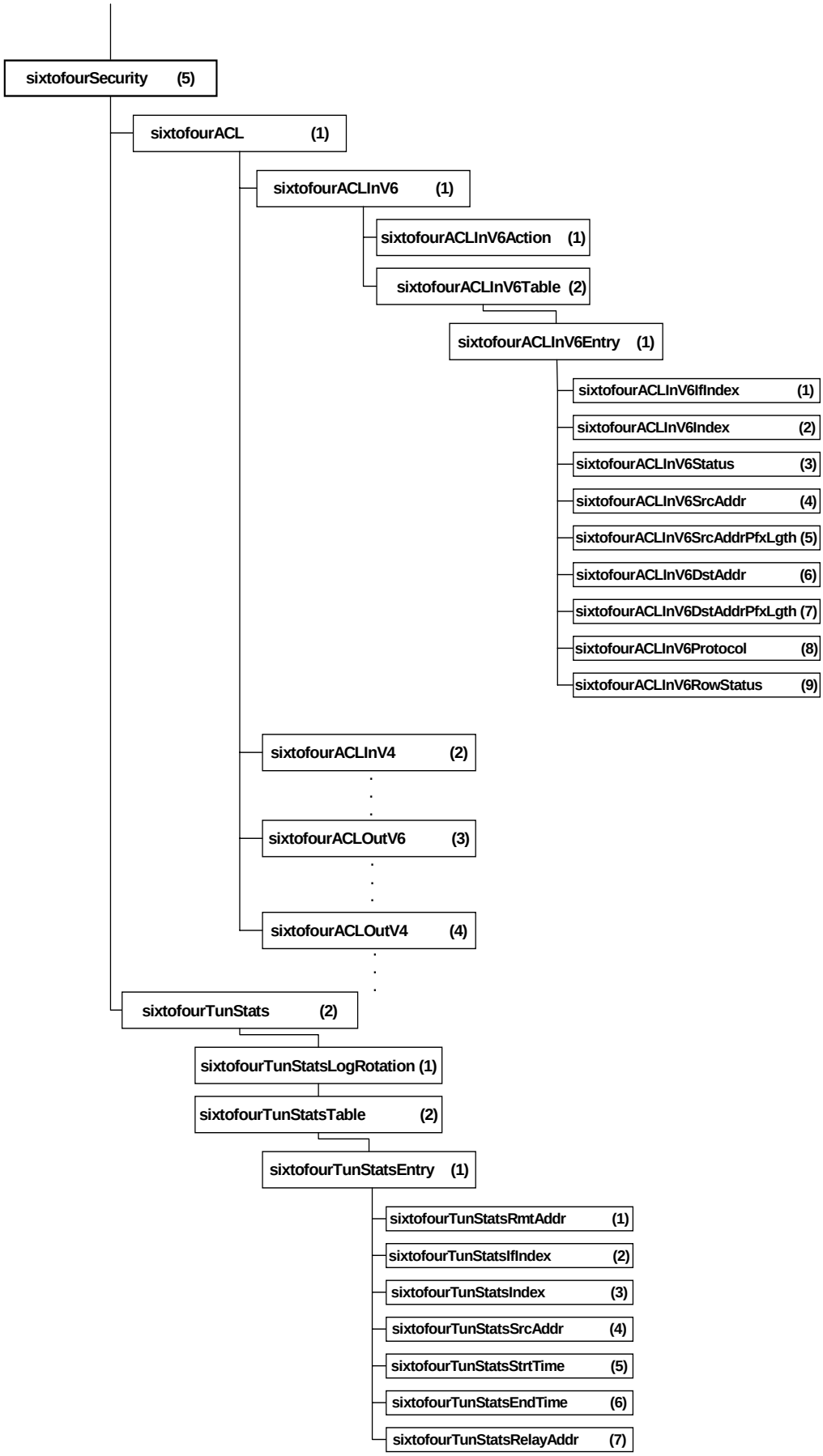
Το δεύτερο μέρος της MIB, που βρίσκεται κάτω από το ***sixtofourSecurity,*** περιέχει όλα τα στοιχεία που ο διαχειριστής πρέπει να ελέγχει για να εξασφαλίζεται η σωστή και ασφαλής λειτουργία του μηχανισμού. Το τμήμα αυτό της MIB χωρίζεται επίσης σε δυο μέρη:

- ένα με τις ACLs (Access Control Lists) που πρέπει να 'σηκωθούν' για να γίνονται οι κατάλληλοι ελέγχοι και βρίσκεται κάτω από το *sixtofourACL*Κάτω από το *sixtofourACL* ορίζονται τέσσερις πίνακες, ένας για την κάθε περίπτωση πακέτων που φτάνουν στον 6to4 router.
- το δεύτερο, με τα στατιστικά που πρέπει να κρατούνται για κάθε τούνελ που εγκαταστάθηκε στο σύστημα, για να είναι κατόπιν δυνατός ο εντοπισμός κακόβουλων κόμβων, που είτε έβλαψαν τον 6to4 μηχανισμό, είτε έκαναν χρήση του για να αποκρύψουν τα ίχνη τους και βρίσκεται κάτω από το *sixtofourTunStats*. Σε αυτή την περίπτωση ορίζεται μόνο ο πίνακας *sixtofourTunStatsTable* που κρατά τα στατιστικά για το κάθε tunnel.

Η δομή της sixtofour-MIB δίνεται στο πιο κάτω σχήμα, και ακολούθως περιγράφονται ένα-ένα τα αντικείμενα της MIB.

5.1 ΔΙΑΓΡΑΜΜΑ ΤΗΣ MIB





Σχήμα 26: Διάγραμμα της sixtofour-MIB

5.2 ΑΝΤΙΚΕΙΜΕΝΑ ΤΗΣ MIB

1. sixtofourTunnelMode :

Το αντικείμενο αυτό υποδηλώνει αν στο προς διαχείριση σύστημα υπάρχει δυνατότητα υλοποίησης του 6to4 interface.
Συγκεκριμένα, χρησιμεύει για τον έλεγχο του αν στο σύστημα στο οποίο θα υλοποιηθεί το 6to4 interface είναι εγκατεστημένη η IPv6 – IPv4 tunnel device, η οποία θα χρησιμοποιηθεί για την ενθυλάκωση των IPv6 πακέτων κάτω από την IPv4 επικεφαλίδα για την μετάδοση τους πάνω από την υπάρχουσα IPv4 υποδομή. Χωρίς την ύπαρξη του μηχανισμού αυτού είναι αδύνατη η υλοποίηση του 6to4 και ο διαχειριστής έχει την δυνατότητα να δημιουργήσει τον πίνακα με τα 6to4 interfaces μόνο εάν στο σύστημα υπάρχει ο συγκεκριμένος μηχανισμός. Το μέγιστο επίπεδο πρόσβασης είναι read-only και είναι αντικείμενο τύπου INTEGER.
Στην περίπτωση που ο μηχανισμός tunneling είναι εγκατεστημένος, επιστρέφει (1) sit. Το sit (Simple Internet Transition) όπου καθορίζεται στο RFC 2893 - Transition Mechanisms for IPv6 Hosts and Routers, είναι το σύνολο των μηχανισμών που ενσωματώθηκαν στο νέο πρωτόκολλο (IPv6) για να διευκολύνουν την μετάβαση από το IPv4. Σε αυτούς τους μηχανισμούς συγκαταλέγονται και οι μηχανισμοί tunneling που χρησιμοποιούντα για την αποστολή IPv6 πακέτων μεσώ τούνελ, ενθυλακωμένων κάτω από την IPv4 επικεφαλίδα, με τιμή του πεδίου protocol 41. Αν ο μηχανισμός tunneling δεν είναι εγκατεστημένος, επιστρέφεται (2) noipv6-ipv4tunnel, υποδηλώνοντας ότι δεν είναι δυνατή η δημιουργία 6to4 τούνελ.

2. sixtofourIfNumber

Ο αριθμός των 6to4 interfaces που υπάρχουν στο σύστημα και για τα οποία έχει δημιουργηθεί ήδη έγγραφη στο ifTable (MIB II) .
Ο αριθμός που επιστέφεται είναι ανεξάρτητος από την κατάσταση λειτουργίας στην οποία βρίσκονται τα interfaces. (‘πάνω’ ή ‘κάτω’)
Και αυτό το αντικείμενο έχει μέγιστο επίπεδο πρόσβασης read-only και ο τύπος του είναι Integer32, έτσι όπως ορίζει τον SNMPv2 πρωτόκολλο, και συγκεκριμένα όπως ορίζεται στην SNMPv2-SMI.

3. sixtofourIfTable

Είναι ο πίνακας με τα 6to4 interfaces που υπάρχουν στο σύστημα και παρέχει στον διαχειριστή τις απαραίτητες πληροφορίες για την διαχείριση του συγκεκριμένου είδους interfaces, που είναι η κατάσταση λειτουργίας τους, η IPv4 και 6to4(IPv6) διεύθυνση τους, το πρόθεμα που διαφημίζουν, το Hop Limit και το MTU που τίθενται στην IPv4 επικεφαλίδα που ενθυλακώνει το

6to4 πακέτο για την μετάδοση μέσα στο τούνελ και επιπλέον ελέγχεται η χρήση IPv4 IPsec για προστασία των πακέτων που μεταδίδονται στο τούνελ. Παρέχει επίσης την δυνατότητα για προσθήκη και διαγραφή έγγραφων μεσώ του αντικειμένου *sixtofourIfRowStatus* αλλά και μεταβολή των τιμών των εγγράψιμων αντικειμένων (επίπεδο πρόσβασης read-create), ακόμα και όταν η κατάσταση της έγγραφης είναι 'active', χωρίς να χρειάζεται δηλαδή να θέσει εκτός λειτουργίας την συγκεκριμένη έγγραφη (κατάσταση 'notInService').
Για την δημιουργία του συγκεκριμένου πίνακα το αντικείμενο *sixtofourTunnelMode* πρέπει να είναι στην τιμή sit(1).
Ο αριθμός των εγγραφών δίνεται από το *sixtofourIfNumber*.

3.1 sixtofourIfIndex

Ο αριθμός που μοναδικά χαρακτηρίζει το κάθε 6to4 interface που υπάρχει στο σύστημα. Είναι ο αριθμός με τον οποίο ο διαχειριστής θα αναζητήσει το εκάστοτε αντικείμενο , είτε για να δει είτε για να θέσει τις τιμές του.
Το επίπεδο πρόσβασης του είναι read-only και είναι τύπου Integer32.

3.2 sixtofourIfStatus

Το αντικείμενο αυτό επιστρέφει στον διαχειριστή την κατάσταση του 6to4 interface, παρέχοντας του παράλληλα την δυνατότητα να μεταβάλει την κατάσταση ανάλογα με το πως επιθυμεί.
Το επίπεδο πρόσβασης του αντικειμένου είναι read-write και είναι τύπου INTEGER.
Επιστρέφει up(1) στην περίπτωση που το 6to4 interface είναι 'ανεβασμένο' και μπορεί να μεταδώσει πακέτα. Στην κατάσταση αυτή τίθεται είτε μέσω SNMP είτε μέσω manual configuration από τον διαχειριστή.
Επιστρέφει down(2) στην περίπτωση που υπάρχει το 6to4 interface αλλά είναι κατεβασμένο', δηλαδή δεν μπορεί να μεταδώσει και να παραλάβει πακέτα.

3.3 sixtofourIfMIBIndex

Το αντικείμενο αυτό είναι η τιμή του *ifIndex* που αντιστοιχεί στο συγκεκριμένο 6to4 interface στον πίνακα *ifTable* (MIB II) . Μια έγγραφη του *sixtofourIfTable* είναι ενεργή μόνο αφότου το συγκεκριμένο 6to4 interface που της αντιστοιχεί έχει ανατεθεί και στον πίνακα *ifTable*.
Το μέγιστο επίπεδο πρόσβασης του αντικειμένου είναι read-only και ο τύπος του είναι Integer32.

3.4 sixtofourIfLocalIpv4Address

Η παγκόσμια μοναδική IPv4 διεύθυνση στην οποία γίνεται configure το συγκεκριμένο 6to4 interface. Είναι η εξωτερική διεύθυνση του interface που θα χρησιμοποιηθεί για την IPv4 δρομολόγηση των 6to4 πακέτων. Επιπλέον, είναι η διεύθυνση βάση της οποίας προσθέτοντας το 6to4 πρόθεμα 2002::/16 σχηματίζεται το 6to4 πρόθεμα του site, το οποίο θα χρησιμοποιήσει για να

διαφημιστεί τόσο εξωτερικά και να το γνωρίζουν οι διάφοροι 6to4 και relay routers, όσο και εσωτερικά, για τον καθορισμό των 6to4 διευθύνσεων των τοπικών κόμβων.
Για την δημιουργία νέων εγγραφών είναι απαραίτητο να ανατεθεί πρώτα σε αυτό το αντικείμενο η IPv4 διεύθυνση του προς δημιουργία interface, η οποία δεν θα αντιστοιχεί σε άλλο 6to4 interface.
Το μέγιστο επίπεδο πρόσβασης του αντικειμένου είναι read-only, έτσι ώστε να μην επιτρέπεται η αλλαγή της IPv4 διεύθυνση που έχει ανατεθεί κατά την δημιουργία του interface, το οποίο μπορεί να προκαλέσει την λανθασμένη λειτουργία ή και κατάρρευση του μηχανισμού. Σε περίπτωση που επιθυμείται ανάθεση 6to4 interface σε νέα IPv4 διεύθυνση, πρέπει να δημιουργηθεί νέα εγγραφή στον πίνακα, έτσι ώστε να ελεγχθεί η μοναδικότητα, η παγκόσμια της εμβέλεια και η μη χρήση της από άλλο 6to4 interface.
Ο τύπος του αντικειμένου είναι IpAddress έτσι όπως τον ορίζει το SNMPv2 πρωτόκολλο, και συγκεκριμένα όπως ορίζεται στην SNMPv2-SMI (πρέπει να επιστρέφεται στον agent σε network byte order).

3.5 *sixtofourIfLocal6to4Address*

Η 6to4 διεύθυνση του site, έτσι όπως προκύπτει από την IPv4 διεύθυνση που ορίστηκε στο προηγούμενο αντικείμενο και το 6to4 πρόθεμα 2002::/16, και είναι η διεύθυνση την οποία ο 'υπόλοιπος κόσμος' γνωρίζει το site. Το μήκος του προθέματος που διαφημίζει το συγκεκριμένο 6to4 site δίνεται στο επόμενο αντικείμενο. Η 6to4 διεύθυνση καθορίζεται αυτόματα σε μερικά συστήματα (με διαφορετικό τρόπο μάλιστα για τα διάφορα συστήματα) ενώ σε αλλά πρέπει να καθοριστεί από τον διαχειριστή. Λανθασμένο configuration της 6to4 διευθύνσεως έχει σαν αποτέλεσμα την αδυναμία λειτουργίας του 6to4 μηχανισμού.
Το επίπεδο πρόσβασης του αντικειμένου είναι read-create και είναι τύπου InetAddressIPv6 όπως αυτό ορίζεται στην INET-ADDRESS-MIB. (και σε αυτό το textual convention πρέπει να δίνεται η διεύθυνση σε network byte order)

3.6 *sixtofourIfLocal6to4AddrPfxLength*

Το μήκος του προθέματος το οποίο θα διαφημίζει ο 6to4 router έξω από το site. Το μήκος αυτό είναι συνήθως διαφορετικό από το μήκος του προθέματος που διαφημίζεται εντός του site. Το μέγεθος αυτό είναι συνήθως 48 για την περίπτωση που το προς διαχείριση μηχάνημα είναι 6to4 router.
Το μέγιστο επίπεδο πρόσβασης του αντικειμένου είναι read-create και είναι τύπου Integer32.

3.7 *sixtofourIfHopLimit*

Η τιμή του πεδίου της IPv4 επικεφαλίδας TTL, που καθορίζει τον αριθμό των ενδιάμεσων κόμβων από όπου θα περάσουν τα ενθυλακωμένα πακέτα μέχρι

να απορριφθούν. Αν το συγκεκριμένο αντικείμενο δεν οριστεί κατά την ενθυλάκωση των πακέτων από τον διαχειριστή, το πεδίο TTL παίρνει την τιμή του αντίστοιχου πεδίου Hop Limit στην IPv6 επικεφαλίδα. Στην περίπτωση αυτή το αντικείμενο επιστρέφει την τιμή 0.
Το μέγιστο επίπεδο πρόσβασης είναι read-write έτσι ώστε ο διαχειριστής να μπορεί να καθορίσει τον επιθυμητό αριθμό κόμβων που θα διανύσουν τα πακέτα μέσα στο τούνελ μέχρι να απορριφθούν, αν δεν έχουν φτάσει στον προορισμό τους ή να θέσει την τιμή 0 έτσι ώστε να τίθεται η τιμή πεδίου στην IPv4 από την IPv6 επικεφαλίδα.
Ο τύπος του αντικειμένου είναι Integer32.

3.8 sixtofourIfMtu

Το αντικείμενο αυτό δίνει την δυνατότητα στον διαχειριστή να καθορίσει το μέγιστο μέγεθος του πακέτου (MTU –Maximum Transfer Unit) που θα χρησιμοποιείται κατά την IPv4 δρομολόγηση. Ο καθορισμός του MTU είναι σημαντικός γιατί γενικά στους μηχανισμούς tunneling ο κατατεμαχισμός των πακέτων είναι ανεπιθύμητος, και ειδικά στο 6to4, στην περίπτωση της anycast διευθύνσεως καταστροφικός. Τιμή ίση με 0 υποδηλώνει ότι το σύστημα κάνει χρήση του IPv4 Path MTU Discovery Protocol για τον υπολογισμό του ιδανικού MTU που θα χρησιμοποιήσει. Στην αντίθετη περίπτωση, ο διαχειριστής πρέπει να θέσει το μέγεθος ίσο με το MTU του link layer κάτω από το IP layer (μείον 20 που είναι το μέγεθος της επικεφαλίδας IPv4)

3.9 sixtofourIfIpSec

Το αντικείμενο παρέχει την πληροφορία του εάν γίνεται η χρήση IPv4 IPsec πιστοποίησης (authentication) ή κρυπτογράφησης (encryption) μεταξύ των δυο άκρων του τούνελ, έτσι ώστε να προστατευτεί το IPv6 φορτίο(μαζί με την επικεφαλίδα του) από παραποίηση των δεδομένων του.
Το επίπεδο πρόσβασης του αντικειμένου είναι read-only και ο τύπος του είναι INTEGER,επιστρέφοντας yes(1) στην περίπτωση που γίνεται χρήση IPsec πρωτοκόλλου, και no(2) στην αντίθετη περίπτωση.

3.10 sixtofourIfRowStatus

Η κατάσταση της συγκεκριμένης γραμμής(έγγραφης) του πίνακα *sixtofourIfTable*. Μεσώ του αντικειμένου αυτού μπορούν να προστεθούν νέες έγγραφες ή να αφαιρεθούν παλιές από τον πίνακα.
Όπως έχει αναφερθεί και πιο πάνω, η δημιουργία νέας έγγραφης είναι δυνατή μόνο αν στο προς δημιουργία 6to4 interface, έχει ανατεθεί μοναδική παγκόσμια IPv4 διεύθυνση, που δεν χρησιμοποιείται από άλλο 6to4 interface. Επίσης, στην περίπτωση που ο πίνακας είναι άδειος, απαραίτητη προϋπόθεση για δημιουργία έγγραφης είναι το αντικείμενο *sixtofourTunnelMode* να έχει την τιμή sit(1).

Για να τεθεί το αντικείμενο στην κατάσταση ‘createAndWait’ ή ‘createAndGo’, πρέπει να έχουν τεθεί όλα τα αντικείμενα εκτός των *sixtofourIfMTU, sixtofourIfHopLimit* και *sixtofourIpSec*.
Δημιουργώντας μια νέα εγγραφή στον πίνακα, της ανατίθεται ένας δείκτης (*sixtofourIfIndex*) και δημιουργείται παράλληλα μια νέα εγγραφή στον πίνακα *ifTable* (MIB II).
Η κατάσταση του αντικειμένου μπορεί να γίνει ‘active’, μόνο αφότου ο agent αναθέσει τιμή στον *ifIndex* (MIB II) για το συγκεκριμένο 6to4 interface, ανεξάρτητα από την κατάσταση που βρίσκεται το interface στην συγκεκριμένη στιγμή.
Αντίθετα, διαγράφοντας μια έγγραφη (γραμμή) από τον πίνακα, διαγράφεται και η αντίστοιχη εγγραφή από το *ifTable*.
Στην περίπτωση που υπάρχουν εγγραφές στο *sixtofourRelayTable* που σχετίζονται με το συγκεκριμένο interface που πρόκειται να σβηστεί, η κατάσταση του αντικειμένου *sixtofourRelayRowStatus* δεν πρέπει να τεθεί σε 'destroy' ή 'notInService'.

4. sixtofourRelayTable

Ο πίνακας για την διαχείριση των δρομολογίων (routes) προς τους Relay Routers που θα χρησιμοποιεί το συγκεκριμένο 6to4 interface για ανταλλαγή πακέτων με άλλους hosts που ανήκουν σε κάποιο native IPv6 domain. Η υλοποίηση των routes σε ξεχωριστό πίνακα ήταν αναγκαία αφού κάθε 6to4 interface μπορεί να κάνει configure περισσότερους από ένα relay.
Ο διαχειριστής έχει την δυνατότητα να εγγράψει ή να διαγράψει όσους Relay Routers επιθυμεί, και επιπλέον να μεταβάλει τα διάφορα χαρακτηριστικά του κάθε route, και συγκεκριμένα το δίκτυο και το πρόθεμα (μάσκα) προορισμού, την διεύθυνση του relay router καθώς και το κατά πόσο χρησιμοποιείται η anycast διεύθυνση και το metric του κάθε route..
Στην περίπτωση που δεν υπάρχει υλοποιημένο 6to4 interface στο σύστημα δεν μπορεί να δημιουργηθεί ο *sixtofourRelayTable* πίνακας. Συγκεκριμένα, για δημιουργία νέας εγγραφής για τον πίνακα απαραίτητη προϋπόθεση είναι η ύπαρξη της εγγραφής στον πίνακα *sixtofourIfTable* για το 6to4 interface στο οποίο αντιστοιχεί το route προς τον καθορισμένο relay router. Η τιμή του *sixtofourIfIndex* του αντιστοίχου 6to4 interface δίνεται από το αντικείμενο *sixtofourRelayIfIndex*.
Για όσα αντικείμενα του πίνακα είναι εγγράψιμα, ο διαχειριστής μπορεί να μεταβάλει τις τιμές τους ανεξάρτητα από την κατάσταση της εγγραφής (*sixtofourRelayRowStatus*) την συγκεκριμένη στιγμή.

4.1 sixtofourRelayIndex

Ο δείκτης που μοναδικά υποδηλώνει το συγκεκριμένο route προς τον relay router που θα χρησιμοποιηθεί από το 6to4 interface για αποστολή πακέτων μεσώ τούνελ σε κάποιο native IPv6 site.

Το επίπεδο πρόσβασης στο συγκεκριμένο αντικείμενο είναι read-only και είναι τύπου Integer32.

4.2 sixtofourRelayIfIndex

Ο δείκτης που μοναδικά υποδηλώνει το συγκεκριμένο interface για το οποίο έχει γίνει configure ο συγκεκριμένος relay router. Το αντικείμενο αυτό συσχετίζει την έγγραφη του πίνακα *sixtofourRelayTable* με το interface του πίνακα *sixtofourIfTable*, στο οποίο αντιστοιχεί ο *sixtofourIfIndex*, με τιμή τον αριθμό που επιστρέφει το συγκεκριμένο αντικείμενο.
Το επίπεδο πρόσβασης του αντικειμένου είναι read-only και είναι τύπου Integer32.

4.3 sixtofourRelayDestAddr

Η διεύθυνση του native IPv6 δικτύου προορισμού, το πρόθεμα του οποίου δίνεται στο επόμενο αντικείμενο. Πακέτα προς IPv6 διευθύνσεις που αντιστοιχούν στο συγκεκριμένο δίκτυο/διεύθυνση αποστέλνονται μέσω του Relay Router που αντιστοιχεί στη ίδια εγγραφή του πίνακα. Η εκάστοτε υλοποίηση του SNMP agent πρέπει να ελέγχει έτσι ώστε πακέτα με διευθύνσεις/δίκτυα προορισμού που δεν ανήκουν στο native IPv6 domain να μην δρομολογούνται μέσω κάποιου relay router.
Το επίπεδο πρόσβασης είναι read-create παρέχοντας στον διαχειριστή την δυνατότητα τροποποίησης των διαφόρων δρομολογίων.
Ο τύπος του είναι InetAddressIPv6 όπως ορίζεται στην INET-ADDRESS-MIB.

4.4 sixtofourRelayDestAddrPrefix

Το πρόθεμα (μάσκα υποδικτύου) που χρησιμοποιείται σε συνδυασμό με την πιο πάνω διεύθυνση προορισμού για την εύρεση του Relay Router που θα χρησιμοποιηθεί στην κάθε περίπτωση. Είναι ουσιαστικά όπως στην περίπτωση του IPv4 η μάσκα που εφαρμόζεται στην αναγραφόμενη στην επικεφαλίδα του πακέτου διεύθυνση προορισμού, έτσι ώστε από το routing table του 6to4 router, να καθοριστεί ο επόμενος προορισμός του πακέτου (next hop) που σε αυτή την περίπτωση είναι ο προκαθορισμένος relay router στον οποίο θα σταλούν μεσώ τούνελ τα πακέτα από το 6to4 site.
Το επίπεδο πρόσβασης του αντικειμένου είναι read-write και ο τύπος του είναι InetAddressIPv6 όπως ορίζεται στην INET-ADDRESS-MIB, όπου στην συγκεκριμένη περίπτωση τα πρώτα, ίσα με μήκος του prefix, bit τίθενται σε 1 και τα υπόλοιπα 0.

4.5 sixtofourRelayAddress

Η διεύθυνση του Relay Router που θα σταλούν τα πακέτα, αν ο προορισμός είναι ένας native IPv6 host, για την συγκεκριμένη έγγραφη. Αν έχει γίνει η ταυτοποίηση της διεύθυνσης/prefix(μάσκα) των δυο προηγούμενων αντικειμένων με την IPv6 διεύθυνση προορισμού που αναγράφεται στο 6to4 πακέτο, τότε τα πακέτα ενθυλακώνονται σε IPv4 επικεφαλίδα με διεύθυνση προορισμού την συγκεκριμένη διεύθυνση του relay router που έχει οριστεί από τον διαχειριστή. Η διεύθυνση αυτή μπορεί να είναι είτε η anycast διεύθυνση για τον πλησιέστερο relay (192.88.99.0/24 ή η αντίστοιχη 6to4 2002:c058:6301::) είτε κάποια configured από τον διαχειριστή unicast IPv4διεύθυνση του relay. Το είδος της διεύθυνσης (αν είναι δηλαδή configured ή anycast) για την κάθε εγγραφή φαίνεται από το επόμενο αντικείμενο.
Το επίπεδο πρόσβασης είναι read-write, έτσι ώστε να μπορεί ο διαχειριστής να αλλάξει κάποια routes προς κάποιους relay routers στα οποία υπάρχει αδυναμία επικοινωνίας, είτε οι αντίστοιχοι relays είναι εκτός λειτουργίας, είτε έχουν καταγράφει κάποιες επιθέσεις από τις συγκεκριμένες διευθύνσεις των relay.
Το αντικείμενο είναι τύπου InetAddressIPv6 όπως ορίζεται στην INET-ADDRESS-MIB.

4.6 sixtofourRelayAnycastStatus

Το συγκεκριμένο αντικείμενο καταδεικνύει αν η συγκεκριμένη διεύθυνση του Relay Router (*sixtofourRelayAddress*)είναι η anycast διεύθυνση που καθορίζεται στο RFC 3068 – "An Anycast Prefix for 6to4 Relay Routers".
Το επίπεδο πρόσβασης σε αυτό το αντικείμενο είναι read-only.
Το αντικείμενο είναι τύπου INTEGER, όπου παίρνει την τιμή anycastAddr(1) στην περίπτωση που η διεύθυνση του relay router της εγγραφής είναι η anycast διεύθυνση, και την τιμή configuredAddr(2) στην αντίθετη περίπτωση κατά την οποία από τον διαχειριστή.

4.7 sixtofourRelayRouteMetric

Τέλος, ορίζεται το metric του κάθε route, που καθορίζει την σειρά προτεραιότητας του συγκεκριμένου route στην περίπτωση που υπάρχουν περισσότερες από μια διαδρομές για κάποιο συγκεκριμένο προορισμό. Η μεγαλύτερη προτεραιότητα ανήκει στο route με το μικρότερο metric.
Το επίπεδο πρόσβασης του αντικειμένου είναι read-write, ούτως ώστε ο διαχειριστής να μπορεί να καθορίζει και να τροποποιεί τα metric των διάφορων διαδρόμων.
Ο τύπος του αντικειμένου είναι Integer32.

4.8 sixtofourRelayRowStatus

Η κατάσταση της συγκεκριμένης έγγραφης (γραμμής) του πίνακα *sixtofourRelayTable*. Χρησιμοποιείται για την δημιουργία και την διαγραφή έγγραφων από τον πίνακα.
Για να είναι δυνατή η δημιουργία νέας έγγραφης κάποιου route για Relay Router, απαραίτητη προϋπόθεση είναι να υφίσταται και να έχει ανατεθεί στο *sixtofourRelayIfIndex* η τιμή του αντικειμένου *sixtofourIfIndex*, που αντιστοιχεί στο συγκεκριμένο interface στο οποίο θα ανατεθεί το route προς τον relay router.
Για να τεθεί στην κατάσταση 'active', πρέπει όλα τα αντικείμενα του πίνακα, εκτός το *sixtofourRelayRouteMetric* να έχουν ανατεθεί.
Αλλαγή ή διαγραφή κάποιας έγγραφης, μπορεί να γίνει οποία και αν είναι η κατάσταση της.

5. sixtofourSecurity

Το δεύτερο αυτό τμήμα της sixtofour-MIB καλύπτει όλες εκείνες τις ανάγκες για ασφαλή λειτουργία του μηχανισμού, καθώς και τους απαραίτητους ελέγχους που ο διαχειριστής του 6to4 site πρέπει να εκτελέσει στον συνοριακό του 6to4 δρομολογητή έτσι ώστε τα πακέτα που προέρχονται μέσα από το τοπικό του site να μην απορριφθούν από τους αντιστοίχους έλεγχους (Sanity Checks) που εκτελούν οι άλλοι 6to4 ή relay routers προορισμού. Το *sixtofourSecurity* χωρίζεται σε δυο επιμέρους τμήματα, τις ACLs που επιτρέπουν ή απαγορεύουν σε συγκεκριμένα πακέτα να εισέλθουν ή να εξέλθουν του site, και τον πίνακα με τα στατιστικά στοιχεία για κάθε τούνελ που εγκαταστάθηκε στο σύστημα.

5.1 sixtofourACL

Στο τμήμα αυτό καθορίζονται οι Access Control Lists (ACLs) για τις τέσσερις περιπτώσεις πακέτων που διακινούνται από και προς τον 6to4 μηχανισμό. Οι περιπτώσεις αυτές φαίνονται στο πιο κάτω σχήμα:

Σχήμα 27: Κίνηση από και προς τον 6to4 router.

Οι τέσσερις αυτές περιπτώσεις καθορίζονται ξεχωριστά η κάθε μια, τόσο για ευκολία διαχείρισης, όσο και λόγω των διαφορετικών πολιτικών που πρέπει να ακολουθούνται στην κάθε περίπτωση. Για την κάθε περίπτωση, εκτός από τις

διευθύνσεις που ο ίδιος ο διαχειριστής για κάποιους συγκεκριμένους λόγους (για παράδειγμα δίκτυα από τα οποία είχαν γίνει επιθέσεις στο παρελθόν) θέλει να απαγορεύσει, υπάρχουν κάποιες ορισμένες διευθύνσεις που καθορίζονται στα Sanity Checks τα οποία, στην περίπτωση που δεν τα υλοποιεί η tunnel device του συστήματος πρέπει να τα ελέγξει μέσω των ACLs ο ίδιος ο διαχειριστής. Συγκεκριμένα, για την κάθε περίπτωση πρέπει να ελεγχθούν οι πιο κάτω διευθύνσεις:

Περίπτωση 1η : IPv6 πακέτα από το τοπικό 6to4 site καταφθάνουν στο 6to4 interface για να ενθυλακωθούν σε IPv4 και να μεταδοθούν πάνω από την IPv4 υποδομή. Τα αντικείμενα για αυτή την περίπτωση ορίζονται κάτω από το ***sixtofourACLInV6.***

- **IPv6 Sanity Checks**
  Απορρίπτονται οι πιο κάτω IPv6 διευθύνσεις προορισμού :

  → 0::/16 (compatible, mapped addresses, loopback, unspecified, ...)
  → fe80::/10 (link-local)
  → fec0::/10 (site-local)
  → ff00::/8 (any multicast)

- Απορρίπτονται πακέτα στα οποία η IPv6 διεύθυνση προελεύσεως δεν αντιστοιχεί στην IPv4 διεύθυνση του site. Και συγκεκριμένα αν η V4ADDR που περιέχεται στο 6to4 πρόθεμα του site 2002::V4ADDR/48 της IPv6 διευθύνσεως είναι διαφορετική από την IPv4 διεύθυνση του site.

Περίπτωση 2η : IPv4 πακέτα που προέρχονται μέσα από το tunnel (και έχουν ενθυλακωμένα τα IPv6) καταφθάνουν στον 6to4 router, και κατόπιν στο 6to4 interface όπου θα γίνει η απενθυλάκωση τους. Τα αντικείμενα για αυτή την περίπτωση ορίζονται κάτω από το **sixtofourACLInV4.**

- **IPv4 Sanity Checks**
  Απορρίπτονται πακέτα με τις πιο κάτω IPv4 διευθύνσεις πηγής και προορισμου:

  → 0.0.0.0/8 (not assigned yet)
  → 10.0.0.0/8 (private)
  → 127.0.0.0/8 (loopback)
  → 172.16.0.0/12 (private)
  → 192.168.0.0/16 (private)
  → 169.254.0.0/16 (IANA Assigned DHCP link-local)
  → 224.0.0.0/4 (multicast)
  → 240.0.0.0/4 (reserved and broadcast)

Περίπτωση 3η : IPv6 πακέτα τα οποία έχουν προκύψει από την αφαίρεση της IPv4 επικεφαλίδας των IPv4 πακέτων που κατέφθασαν στο τοπικό

6to4 site, και προωθούνται προς κάποιο 6to4 host εντός του site. Τα αντικείμενα για αυτήν την περίπτωση ορίζονται κάτω από το **sixtofourACLOutV6.**

- **IPv6 Sanity Checks**
  Απορρίπτονται οι πιο IPv6 διευθύνσεις προελεύσεως όπως έχουν οριστεί στη $1^{η}$ περίπτωση.

- Απορρίπτονται τα πακέτα με IPv6 διεύθυνση προορισμού διαφορετική από διεύθυνση που αντιστοιχεί στο συγκεκριμένο 6to4 site. Δηλαδή, η V4ADDR που περιέχεται στην διεύθυνση προορισμού είναι διαφορετική από την IPv4 διεύθυνση του τοπικού 6to4 site.

- Απορρίπτονται τα πακέτα που προέρχονται από 6to4 sites (δηλαδή IPv6 διευθύνσεις πηγής 2002::/16) και η IPv4 διεύθυνση πηγής είναι διαφορετική από την V4ADDR που περιέχεται στην διεύθυνση προελεύσεως.

Περίπτωση $4^{η}$: IPv4 πακέτα (όπου από κάτω βρίσκονται ενθυλακωμένα τα IPv6) τα οποία αφού έχει γίνει η ενθυλάκωση στο 6to4 interface φεύγουν από τον 6to4 router προς τον προορισμό τους. Τα αντικείμενα για αυτή την περίπτωση ορίζονται κάτω από το **sixtofourACLOutV4.**

- **IPv4 Sanity Checks**
  Απορρίπτονται IPv4 πακέτα με διεύθυνση είτε προελεύσεως είτε προορισμού :

  → 0.0.0.0/8 (not assigned yet)
  → 10.0.0.0/8 (private)
  → 127.0.0.0/8 (loopback)
  → 172.16.0.0/12 (private)
  → 192.168.0.0/16 (private)
  → 169.254.0.0/16 (IANA Assigned DHCP link-local)
  → 224.0.0.0/4 (multicast)
  → 240.0.0.0/4 (reserved and broadcast)

Έτσι, ορίζονται τέσσερις συνολικά πίνακες, ένας για κάθε μια από τις πιο πάνω περιπτώσεις, και πριν τον κάθε πίνακα ένα αντικείμενο το οποίο καθορίζει την πολιτική της κάθε λίστας. Επαναλαμβάνεται στο σημείο αυτό ότι οι πρέπει να εφαρμοστούν μόνο οι συγκεκριμένοι από τους πιο πάνω ελέγχους οι οποίοι δεν εκτελούνται από την tunnel device του συστήματος. Επιπλέον, ο διαχειριστής μπορεί με αυτό τον τρόπο είτε να επιτρέψει είτε να απαγορεύσει την διακίνηση πακέτων από και προς όποιες διευθύνσεις ή δίκτυα δεν θεωρεί έμπιστα ή τουλάχιστον ακίνδυνα. Η ιδανική φυσικά κατάσταση θα ήταν να δέχεται και να στέλνει πακέτα μόνο σε προορισμούς με τους οποίους υπάρχει σχέση εμπιστοσύνης.

Οι τέσσερις πίνακες που ορίζονται είναι οι *sixtofourACLInV6Table*, *sixtofourACLInV4Table*, *sixtofourACLOutV6Table* και *sixtofourACLOutV4Table*, όπου ο καθένας αντίστοιχα περιέχει τους κανόνες για τις τέσσερις πιο πάνω περιπτώσεις. Επειδή τα αντικείμενα στους τέσσερις πίνακες έχουν την ίδια σημασία και χρησιμότητα και η διάφορα τους έγκειται ουσιαστικά μόνο στον τύπο της διευθύνσεως, παρατίθεται στη συνέχεια μια γενική περιγραφή τους που καλύπτει και τους τέσσερις πίνακες. Στην κάθε περίπτωση το *sixtofourACL**xxxx**yyyy* αντιστοιχεί με τα sixtofourACL**InV6**yyyy, sixtofourACL**InV4**yyyy, sixtofourACL**OutV6**yyyy, sixtofourACL**OutV4**yyyy για την καθεμία εκ των τεσσάρων περιπτώσεων.

5.1.x.1 sixtofourACLxxxxAction

Το αντικείμενο αυτό ορίζεται πριν τον κάθε πίνακα με τους κανόνες της ACL και καθορίζει την πολιτική της συγκεκριμένης λίστας. Είναι αντικείμενο τύπου INTEGER και παίρνει τις τιμές deny(1) και allow(2). Στην περίπτωση που πάρει την τιμή deny(1) όλα τα πακέτα στα οποία υπάρχει ταύτιση με τους κανόνες του πίνακα απορρίπτονται, ενώ σε όλα τα υπόλοιπα επιτρέπεται η διέλευση. Στην αντίθετη περίπτωση, όπου δηλαδή το αντικείμενο είναι allow(2) επιτρέπεται η διέλευση μόνο στα πακέτα στα οποία αντιστοιχίζονται με τους κανόνες του πίνακα, και απορρίπτονται όλα τα υπόλοιπα.
Το αντικείμενο αυτό είναι τύπου read-write έτσι ώστε ο διαχειριστής να μπορεί να καθορίζει την πολιτική που θα ακολουθεί η συγκεκριμένη ACL.

5.1.x.2 sixtofourACLxxxxTable

Ο πίνακας με τους κανόνες κάθε λίστας. Ο πίνακας είναι τύπου read-create έτσι ώστε να δίνεται το δικαίωμα στον διαχειριστή να εγγράφει ή να διαγράφει κανόνες μέσω SNMP. Στον πίνακα αυτό ορίζονται να γίνονται όσοι από τους έλεγχους που έχουν καταγράφει πιο πάνω δεν εκτελεί η tunnel device του συστήματος για την κάθε περίπτωση. Επιπλέον, ο εκάστοτε διαχειριστής μπορεί να εγγράψει δικούς του κανόνες για να απαγορεύσει ή να επιτρέψει (ανάλογα με την τιμή που ορίζει στο αντικείμενο *sixtofourACLxxxxAction*)την διέλευση πακέτων από συγκεκριμένα sites ή πακέτα συγκεκριμένων πρωτοκόλλων. Επιπλέον, μέσω του αντικειμένου *sixtofourACLxxxxIfIndex* συσχετίζεται ο κάθε κανόνας της συγκεκριμένης λίστας με το αντίστοιχο 6to4 interface στο οποίο θα εφαρμοστεί.
Ο πίνακας αυτός έχει δυο δείκτες, τους *sixtofourACLxxxxIfIndex* και *sixtofourACLxxxxIndex* για την διάκριση του κάθε κανόνα. Ο πρώτος δείχνει το 6to4 interface στο οποίο εφαρμόζεται ο κάθε κανόνας και ο δεύτερος μια μοναδική τιμή για τον κάθε κανόνα του κάθε interface.

5.1.x.2.1 sixtofourACLxxxxIfIndex

Το αντικείμενο αυτό καθορίζει το 6to4 interface στο οποίο εφαρμόζεται ο συγκεκριμένος κανόνας. Η τιμή που επιστρέφει είναι η τιμή του

sixtofourIfIndex για το αντίστοιχο interface. Μαζί με το επόμενο αντικείμενο *sixtofourACLxxxxIndex* αποτελούν τους δείκτες τους πίνακα, συνδυασμός των οποίων δίνει μια μοναδική τιμή στον κάθε κανόνα.
Το αντικείμενο αυτό είναι τύπου Integer32.

5.1.x.2.2 sixtofourACLxxxxIndex

Ο δεύτερος δείκτης του πίνακα, που σε συνδυασμό με το προηγούμενο αντικείμενο καθορίζουν μοναδικά τον κάθε κανόνα του πίνακα. Ο καθορισμός της τιμής που θα λαμβάνει στην κάθε περίπτωση το συγκεκριμένο αντικείμενο επαφίεται στον εκάστοτε διαχειριστή με μόνη απαίτηση η τιμή που προκύπτει από τον συνδυασμό του με το *sixtofourACLxxxxIfIndex* να είναι μοναδική.
Και αυτό το αντικείμενο είναι τύπου Integer32.

5.1.x.2.3 sixtofourACLxxxxStatus

Το αντικείμενο αυτό καθορίζει το αν θα χρησιμοποιείται ο συγκεκριμένος κανόνας ή όχι. Είναι αντικείμενο τύπου INTEGER και παίρνει την τιμή on(1) στην περίπτωση που ο διαχειριστής επιθυμεί να γίνεται ο συγκεκριμένος έλεγχος που καθορίζει ο κανόνας. Στην περίπτωση που τεθεί στην τιμή off(2) ο συγκεκριμένος κανόνας, αν και υπάρχει, δεν εκτελείτε ο έλεγχος τον οποίο καθορίζει.
Το *sixtofourACLInV6Status* είναι το μόνο αντικείμενο (εκτός του αντικειμένου τύπου RowStatus) με μέγιστο επίπεδο πρόσβαση read-create, έτσι ώστε ο διαχειριστής να μπορεί μέσω SNMP να ενεργοποιεί ή να απενεργοποιεί τον συγκεκριμένο κανόνα.

5.1.x.2.4 sixtofourACLxxxxSrcAddr

Η διεύθυνση η οποία θα συγκριθεί με τη διεύθυνση προελεύσεως του πακέτου αφότου της εφαρμοστεί η μάσκα με μήκος που ορίζεται από το επόμενο αντικείμενο *sixtofourACLxxxxSrcAddrPfxLgth*. Στην περίπτωση που οι δυο διευθύνσεις είναι ίδιες εφαρμόζεται ότι ορίζει το αντικείμενο *sixtofourACLxxxxAction* της συγκεκριμένης ACL , αν δηλαδή το πακέτο θα απορριφθεί ή θα του επιτραπεί η διέλευση.
Το αντικείμενο αυτό είναι τύπου InetAddressIPv6 στην $1^{η}$ και $3^{η}$ περίπτωση, όπου ελέγχονται τα IPv6 πακέτα και τύπου IpAddress στην $2^{η}$ και $4^{η}$ περίπτωση όπου τα πακέτα είναι IPv4.

5.1.x.2.5 sixtofourACLxxxxSrcAddrPfxLgth

Το μήκος της μάσκας που θα εφαρμοστεί στην διεύθυνση προελεύσεως του πακέτου έτσι ώστε να γίνει η σύγκριση της με την πιο πάνω διεύθυνση *sixtofourACLxxxxSrcAddr*. Στην περίπτωση που η πιο πάνω διεύθυνση είναι IPv4, το μήκος της μάσκας καθορίζει ποσά bit από τα 32 της μάσκας θα τεθούν σε 1. Τα υπόλοιπα τίθενται σε 0 και εφαρμόζεται ακολούθως η μάσκα στην διεύθυνση προελεύσεως.

Στην περίπτωση που η διεύθυνση *sixtofourACLxxxxSrcAddr* είναι IPv6 η μάσκα μπορεί να είναι μέχρι 128 bit, και το μήκος της που επιστρέφει το συγκεκριμένο αντικείμενο είναι όπως και πριν ο αριθμός των 1, ξεκινώντας από το MSF(most significant bit).
Το αντικείμενο αυτό είναι και για τις δυο περιπτώσεις Integer32, και παίρνει τιμές 0 έως 32 στην πρώτη περίπτωση και τιμές 0 έως 128 στην δεύτερη.

5.1.x.2.6 sixtofourACLxxxxDstAddr

Αντίστοιχα με την διεύθυνση προελεύσεως, το αντικείμενο αυτό είναι η διεύθυνση η οποία θα συγκριθεί με την διεύθυνση προελεύσεως του πακέτου αφότου της εφαρμοστεί η μάσκα με μήκος που ορίζεται από το επόμενο αντικείμενο *sixtofourACLxxxxDstAddrPfxLgth*. Στην περίπτωση που γίνεται αντιστοίχηση των δυο διευθύνσεων εφαρμόζεται η ενέργεια που ορίζει το αντικείμενο *sixtofourACLxxxxAction*, αν δηλαδή το πακέτο θα απορριφθεί ή θα του επιτραπεί η διέλευση.
Το αντικείμενο αυτό είναι τύπου InetAddressIPv6 στην $1^{η}$ και $3^{η}$ περίπτωση, όπου ελέγχονται τα IPv6 πακέτα και τύπου IpAddress στην $2^{η}$ και $4^{η}$ περίπτωση όπου τα πακέτα είναι IPv4.

5.1.x.2.7 sixtofourACLxxxxDstAddrPfxLgth

Αντίστοιχα, και το αντικείμενο αυτό, με το s*ixtofourACLxxxxSrcAddrPfxLgth*, καθορίζει το μήκος της μάσκας που θα εφαρμοστεί στην διεύθυνση προορισμού του πακέτου έτσι ώστε να γίνει η σύγκριση της με την πιο πάνω διεύθυνση *sixtofourACLxxxxDstAddr*. Στην περίπτωση που η πιο πάνω διεύθυνση είναι IPv4, το μήκος της μάσκας καθορίζει ποσά bit από τα 32 της μάσκας θα τεθούν σε 1. Τα υπόλοιπα τίθενται σε 0 και εφαρμόζεται ακολούθως η μάσκα στην διεύθυνση προορισμού.
Στην περίπτωση που η διεύθυνση *sixtofourACLxxxxDstAddr* είναι IPv6 η μάσκα μπορεί να είναι μέχρι 128 bit, και το μήκος της που επιστρέφει το συγκεκριμένο αντικείμενο είναι όπως και πριν ο αριθμός των 1, ξεκινώντας από το MSF(most significant bit).
Το αντικείμενο αυτό είναι και για τις δυο περιπτώσεις Integer32, και παίρνει τιμές 0 έως 32 στην πρώτη περίπτωση και τιμές 0 έως 128 στην δεύτερη.

5.1.x.2.8 sixtofourACLxxxxProtocol

Το πρωτόκολλο ανωτέρου επιπέδου που δηλώνεται στην επικεφαλίδα του πακέτου. Μέσω του αντικειμένου αυτού, ο διαχειριστής έχει την δυνατότητα είτε να απαγορεύει είτε να επιτρέπει την διέλευση πακέτων ενός συγκεκριμένου πρωτοκόλλου. Το αντικείμενο αυτό είναι τύπου INTEGER, και στην περίπτωση που ο κανόνας επιτρέπει/απαγορεύει πακέτα TCP πρωτοκόλλου επιστρέφει tcp(1), στην περίπτωση UDP πακέτων επιστρέφει udp(2), στην περίπτωση

ICMP πακέτων επιστρέφει icmpv6(3) και στην περίπτωση αλλού πρωτοκόλλου επιστρέφει other(4).

5.1.x.2.9 sixtofourACLxxxxRowStatus

Η κατάσταση της συγκεκριμένης έγγραφης (κανόνα) του πίνακα.
Χρησιμοποιείται για την δημιουργία και την διαγραφή έγγραφων από τον πίνακα.
Αλλαγή ή διαγραφή κάποιας έγγραφης, μπορεί να γίνει οποία και αν είναι η κατάσταση της.

5.2 sixtofourTunStats

Στο δεύτερο τμήμα που βρίσκεται κάτω από το *sixtofourSecurity*, το *sixtofourTunStats*, κρατούνται διάφορα στατιστικά για τα tunnels που εγκαταστάθηκαν στο σύστημα. Γενικά, στους μηχανισμούς automatic tunneling, τα τούνελ δημιουργούνται δυναμικά, παραμένουν ενεργά για όσο χρόνο κρατά η επικοινωνία και κατόπιν αποδεσμεύονται χωρίς να κρατείται κάποιο στοιχείο. Αυτό είναι και ένα από τα πλεονεκτήματα τους όσον αφορά κυρίως στην απλότητα υλοποίησης και διαχείρισης τους αλλά ταυτόχρονα είναι και ο κύριος λόγος που τελικά γίνονται εργαλείο στα χέρια κακόβουλων χρηστών και χρησιμοποιούνται για την απόκρυψη των ιχνών των επιτιθεμένων. Για το λόγο αυτό, θεωρείται χρήσιμο να κρατούνται από το σύστημα για κάποιο χρονικό διάστημα κάποιες πληροφορίες για τα τούνελ που είχαν εγκατασταθεί στο σύστημα. Με αυτό τον τρόπο καθίσταται δυνατή η εύρεση της διευθύνσεως του κακόβουλου χρηστή, είτε αυτός πρόκειται για κάποιον IPv4, IPv6 ή 6to4 εξωτερικό χρηστή είτε κάποιον 6to4 από το ίδιο το τοπικό 6to4 site. Το τμήμα αυτό αποτελείται από το αντικείμενο *sixtofourTunStatsLogRotation*, που καθορίζει το χρόνο για τον οποίο διατηρούνται οι έγγραφες και τον πίνακα *sixtofourTunStatsTable*, ο οποίος σε κάθε έγγραφη του κρατά ένα tunnel που είχε εγκατασταθεί από ένα 6to4 interface του συστήματος, για επικοινωνία κάποιου τοπικού 6to4 host με κάποιο απομακρυσμένο 6to4 ή host που ανήκει στο native IPv6 site.

5.2.1 sixtofourTunStatsLogRotation

Το αντικείμενο αυτό καθορίζει για πόσο χρόνο διατηρούνται τα διάφορα στατιστικά στοιχεία πριν μετατραπούν σε log file και φυλαχτούν σε κάποιο αποθηκευτικό χώρο του συστήματος.
Είναι τύπου Integer32 και η χρονική διάρκεια καθορίζεται σε λεπτά.

5.2.2 sixtofourTunStatsTable

Ο πίνακας με τα στατιστικά στοιχεία για το κάθε τούνελ που εγκαταστάθηκε στο σύστημα. Κάθε εγγραφή του πίνακα αντιστοιχεί σε ένα τούνελ που έχει εγκατασταθεί

μεταξύ του τοπικού 6to4 router και του 6to4 ή relay router προορισμού και η διάκριση των έγγραφων γίνεται μέσω τριών δεικτών. Ο πρώτος, *sixtofourTunStatsRmtAddr* είναι η διεύθυνση του απομακρυσμένου χρηστή με τον οποίο έχει επικοινωνήσει ο χρηστής που ανήκει στο τοπικό 6to4 site. Ο δεύτερος, *sixtofourTunStatsIfIndex*, είναι η τιμή του *sixtofourIfIndex* του 6to4 interface που χρησιμοποιήθηκε για το συγκεκριμένο τούνελ που καταγράφηκε. Και τέλος είναι ο δείκτης *sixtofourTunStatsIndex*, ο οποίος καθορίζει μια μοναδική τιμή για το κάθε ζευγάρι των πιο πάνω δεικτών.
Ο πίνακας αυτός διατηρείται για το χρονικό διάστημα που καθορίζει το αντικείμενο *sixtofourTunStatsLogRotation*, και ακολούθως τα στοιχεία φυλάσσονται σε log file στο σύστημα.

5.2.2.1 sixtofourTunStatsRmtAddr

Η διεύθυνση του απομακρυσμένου κόμβου με τον οποίο έγινε η επικοινωνία κατά την οποία εγκαταστάθηκε το συγκεκριμένο τούνελ. Ο χρηστής αυτός μπορεί να ανήκει είτε σε κάποιο άλλο 6to4 site είτε σε native IPv6 domain. Στην τελευταία περίπτωση, η διεύθυνση του relay router που χρησιμοποιείται κρατείται στο αντικείμενο *sixtofourTunStatsRelayAddr*. Για την πρώτη περίπτωση δεν χρειάζεται να κρατείται η IPv4 διεύθυνση του 6to4 router αφού μπορεί να εξαχθεί από την 6to4 διεύθυνση προορισμού.
Το αντικείμενο αυτό χρησιμοποιείται, σε συνδυασμό με τα δυο επόμενα αντικείμενα που ακολουθούν, σαν δείκτης για τον καθορισμό του συγκεκριμένου τούνελ που εγκαταστάθηκε με IPv6 διεύθυνση προορισμού των πακέτων την τιμή του *sixtofourTunStatsRmtAddr*, από το 6to4 interface με *sixtofourIfIndex* την τιμή του *sixtofourTunStatsIfIndex* και με αριθμό έγγραφης μοναδικό για τον συνδυασμό των δυο πιο πάνω την τιμή του *sixtofourTunStatsIndex*.
Το αντικείμενο αυτό είναι τύπου InetAddressIPv6.

5.2.2.2 sixtofourTunStatsIfIndex

Η τιμή του *sixtofourIfIndex* για το 6to4 interface που χρησιμοποιήθηκε για το συγκεκριμένο tunnel. Το αντικείμενο αυτό σε συνδυασμό με το προηγούμενο και το επόμενο χρησιμεύουν ως δείκτες για τον καθορισμό του συγκεκριμένου tunnel που χρησιμοποιήθηκε.
Πρόκειται για αντικείμενο τύπου Integer32.

5.2.2.3 sixtofourTunStatsIndex

Ο τρίτος δείκτης για τον καθορισμό του συγκεκριμένου tunnel. Το αντικείμενο αυτό παρέχει μια μοναδική τιμή στο ζεύγος τιμών των αντικειμένων *sixtofourTunStatsRmtAddr* και *sixtofourTunStatsIfIndex* έτσι ώστε να είναι δυνατή η διάκριση των tunnels που έχουν εγκατασταθεί στο σύστημα. Ο καθορισμός της τιμής στο συγκεκριμένο αντικείμενο επαφίεται στον διαχειριστή, με τη μόνη απαίτηση ο συνδυασμός των τριών να είναι μοναδικός.
Και το αντικείμενο αυτό είναι τύπου Integer32.

5.2.2.4 sixtofourTunStatsSrcAddr

Η διεύθυνση του κόμβου μέσα από το τοπικό 6to4 site ο οποίος αντάλλαξε πακέτα με τον απομακρυσμένο κόμβο, η διεύθυνση του οποίου δίνεται από το πρώτο αντικείμενο, μέσα από το συγκεκριμένο tunnel. Η διεύθυνση χρειάζεται για τον εντοπισμό κακόβουλων χρηστών που προέρχονται μέσα από το τοπικό 6to4 site, (κάτι που δεν χρειάζεται στην περίπτωση που ο 6to4 router εκτελεί το ingress filtering και επιτρέπει μόνο σε έγκυρα πακέτα να εξέλθουν το site) αλλά κυρίως για σύγκριση και αντιστοίχηση των συγκεκριμένων στατιστικών στοιχείων με τα αντίστοιχα που έχουν καταγράφει για κάποια επίθεση.
Το αντικείμενο αυτό είναι τύπου InetAddressIPv6.

5.2.2.5 sixtofourTunStatsStrtTime

Η χρονική στιγμή κατά την οποία εγκαταστάθηκε το tunnel. Η καταγραφή του χρόνου γίνεται βάση του *sysUpTime* αντικειμένου (MIB II) την συγκεκριμένη χρονική στιγμή. Η διατήρηση του χρόνου τόσο εγκατάστασης του τούνελ όσο και περάτωσης (επόμενο αντικείμενο) είναι σημαντική για τον προσδιορισμό της κάθε επίθεσης.
Ο τύπος του αντικειμένου είναι TimeTicks, όπως ορίζονται στο SNMPv2-SMI

5.2.2.6 sixtofourTunStatsEndTime

Η χρονική στιγμή περάτωσης του τούνελ. Και σε αυτή την περίπτωση η καταγραφή του χρόνου γίνεται βάση του *sysUpTime* και το αντικείμενο είναι και αυτό τύπου TimeTicks.

5.2.2.5 sixtofourTunStatsRelayAddr

Η IPv4 διεύθυνση του relay router που χρησιμοποιήθηκε στην περίπτωση που η επικοινωνία ήταν με χρηστή του native IPv6 δικτύου. Η χρησιμότητα του αντικειμένου αυτού είναι πολύ μεγάλη, τόσο για την αναγνώριση κακόβουλων relay ή relay που δεν εκτελούν τους ελέγχους ασφαλείας, έτσι ώστε να απαγορευτεί η πρόσβαση σε αυτούς μέσω των ACLs, όσο και για την καταγραφή των κακόβουλων IPv4 χρηστών που χρησιμοποιούν τον 6to4 μηχανισμό για να αποκρύψουν τα ίχνη τους.
Στην περίπτωση που η επικοινωνία γίνεται κατευθείαν μεταξύ του τοπικού 6to4 router και του 6to4 router του site προορισμού, επιστρέφεται η τιμή 0.0.0.0.
Το αντικείμενο αυτό είναι τύπου IpAddress.

5.3 *SIXTOFOUR-MIB*

```
SIXTOFOUR-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Integer32, IpAddress,
    TimeTicks                FROM SNMPv2-SMI
    InetAddressIPv6          FROM INET-ADDRESS-MIB
    DisplayString,
    RowStatus                FROM SNMPv2-TC;


sixtofourMIB MODULE-IDENTITY
    LAST-UPDATED "200504180000z" -- April 18,2005
    ORGANIZATION "NETMODE-NTUA"
    CONTACT-INFO
          "postal:   Demetris Philippides
                     Athanasios Douitsis
                     Georgios Koutepas
                     NETMODE lab
                     Zografou Campus,
                     Athens,Greece
           email:    demfilip@netmode.ntua.gr
                     aduitsis@netmode.ntua.gr
                     gkoutep@netmode.ntua.gr "


    DESCRIPTION
          "The MIB module for management of 6to4 tunnel
           mechanisms"

  REVISION     "200504180000z"

   DESCRIPTION
      "draft version"


    ::= { netmode  5 }


iso ::= { 1 }

org ::= { iso  3 }

dod ::= { org  6 }

internet ::= { dod  1 }

private ::= { internet  4 }

enterprises ::= { private  1 }

ntua ::= { enterprises  969 }

netmode ::= { ntua  1 }

sixtofour ::= { netmode  5 }

sixtofourMIBObjects OBJECT IDENTIFIER ::= { sixtofourMIB 1 }
```

```
sixtofour     OBJECT IDENTIFIER ::= { sixtofourMIBObjects 1 }

-- the 6to4 MIB-Group
--
-- a collection of objects providing information about
-- the 6to4 tunnel mechanism



sixtofourTunnelMode OBJECT-TYPE
      SYNTAX     INTEGER   {
                           sit(1),   --Simple Internet Transition
                           noipv6-ipv4tunnel(2)
                             }
      MAX-ACCESS read-only
      STATUS     current
      DESCRIPTION
              "Checks whether or not the tunnel device,
              used by the system for the encapsulation
              of the ipv6 packets under ipv4 header with
              he value 41 in the protocol field for the
              transmission over the ipv4 infrastructure
              internet), exists. A value of sit(1)
              indicates that the tunnel device has been
              installed on the system and the 6to4
              interface can be configured and operate
              properly. Value of noipv6-ipv4tunnel(2)
              indicates that no tunnel device has been
              installed on the system, probably the
              system does not support IPv6, and in this
              case the configuration of a 6to4 pseudo-
              interface. is not possible. "
     ::= { sixtofour 1 }


sixtofourIfNumber  OBJECT-TYPE
     SYNTAX  Integer32
     MAX-ACCESS  read-only
     STATUS  current
     DESCRIPTION
              "The number of 6to4 interfaces (regardless
              of their current state) present on this
              system."
    ::= { sixtofour 2 }


sixtofourIfTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF SixtofourIfEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The (conceptual) table containing
            information on 6to4 interfaces. It can also
            be used for row creation or deletion. The
            number of entries is given by the value of
            sixtofourIfNumber."
    ::= { sixtofour 3 }


sixtofourIfEntry OBJECT-TYPE
    SYNTAX     SixtofourIfEntry
```

```
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry (conceptual row) containing the
            information on a particular 6to4 interface.
            Entries in the sixtofourTable are created
            and deleted using the sixtofourIfRowStatus
            object.
            All writable objects can change values while
            sixtofourIfRowStatus is 'active'."
    INDEX      { sixtofourIfIndex }
    ::= { sixtofourIfTable 1 }

SixtofourIfEntry ::= SEQUENCE {
    sixtofourIfIndex                    Integer32,
    sixtofourIfStatus                   INTEGER,
    sixtofourIfMIBIndex                 Integer32,
    sixtofourIfLocalIpv4Address         IpAddress,
    sixtofourIfLocal6to4Address         InetAddressIPv6,
    sixtofourIfLocal6to4AddrPfxLength   InetAddressIPv6,
    sixtofourIfHopLimit                 Integer32,
    sixtofourIfMTU                      Integer32,
    sixtofourIfSecurity                 INTEGER,
    sixtofourIfRowStatus                RowStatus

    }


sixtofourIfIndex OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
             "A unique non-zero value identifying
              the particular 6to4 pseudo-interface."
     ::= { sixtofourIfEntry 1 }



sixtofourIfStatus OBJECT-TYPE
      SYNTAX  INTEGER {
                    up(1),       -- ready to pass packets
                    down(2)
                      }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
               " The state of the 6to4 interface.  When
               a Managed system initializes,  all 6to4
               interfaces start with sixtofourIfStatus
               in the no6to4interface(3) state, which
               indicates that no 6to4 interface presents
               on the system.
               As a result of either explicit management
               action or per configuration information
               retained by the managed system,
               sixtofourIfStatus is then changed to
               the up(1) state or down(2) state."
        ::= { sixtofourIfEntry 2 }


sixtofourIfMIBIndex OBJECT-TYPE
```

```
     SYNTAX     Integer32
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
             " The index value which uniquely identifies
             the local interface of the 6to4 mechanism.
             The interface identified by a particular
             value of this index is the same interface
             as identified by the same value of the
             ifIndex present in the ifMIB.
             The agent returns a value of zero if there
             is not a sixtofour interface  "
      ::= { sixtofourIfEntry 3 }


sixtofourIfLocalIpv4Address OBJECT-TYPE
    SYNTAX     IpAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The unique ipv4 address assigned to the
            6to4 pseudo-interface. It is the address of
            the tunnel endpoint (entry node) used for
            communication over the ipv4 infrastructure
            (internet)and it is used for the
            configuration of the 6to4 address."
     ::= { sixtofourIfEntry 4}


sixtofourIfLocal6to4Address OBJECT-TYPE
     SYNTAX     InetAddressIPv6
     MAX-ACCESS read-create
     STATUS     current
     DESCRIPTION
             "The 6to4 address of the specific
             interface. It is created by the 6to4 prefix
             2002::/16 and the 32 bits of the ipv4
             address. If the system does not support
             automatic configuration it can be
             configured manual by the administrator.
             It is the prefix advertised by the 6to4
             site. The length of the prefix is given by
             the sixtofourIfLocal6to4AddrPfxLength
             object.  "
     ::= { sixtofourIfEntry 5 }


sixtofourIfLocal6to4AddrPfxLength  OBJECT-TYPE
     SYNTAX      Integer32
     MAX-ACCESS  read-create
     STATUS      current
     DESCRIPTION
             "The length of the prefix (in bits)
             associated with the 6to4 address of this
             entry."
    ::= { sixtofourIfEntry 6 }


sixtofourIfHopLimit OBJECT-TYPE
        SYNTAX     Integer32 --(0 | 1..255)
        MAX-ACCESS read-create
        STATUS     current
```

```
        DESCRIPTION
             "The IPv4 TTL to use in the outer IP
              header.  A value of 0 indicates that the
              value is copied from the payloads header."
     ::= { sixtofourIfEntry 7 }


sixtofourIfMTU OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-write
        STATUS     current
      DESCRIPTION
              "The MTU to use for the IPv4 transmission
              of the encapsulated packets through the
              tunnel. A value of 0 indicates that the
              system uses IPv4 Path MTU Discovery
              Protocol. If not, the DF bit is not set."
     ::= { sixtofourIfEntry 8 }


sixtofourIfIpSec OBJECT-TYPE
    SYNTAX     INTEGER {
                   yes(1),  -- IPsec security
                no(2)    -- no ipsec
                }
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
            " The value yes(1) indicates that ipv4 ipSec
            is used between the tunnel endpoints for
            authentication or encryption or both,so that
            ipv6 packets could not be modified.
            The value no(2) indicates that the ipSec is
            not supported"
     ::= { sixtofourIfEntry 9 }


sixtofourIfStatus OBJECT-TYPE
    SYNTAX     RowStatus
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
            "The status of this row, by which new
            entries may be created, or old entries
            deleted from this table.
            The creation of a new row is only possible
            if a unique global ipv4 address is assigned
            to the object sixtofourIfLocalIpv4Address.
            Creation of a new table requires that the
            object sixtofourTunnelMode have been
            assigned to the value sit(1).
            In order for a row to be set to`CreateAndGo'
            all objects expect sixtofourIfMTU,
            sixtofourIfHopLimit and sixtofourIpSec need
            to be set .Creating a row in this table will
            cause an interface index to be assigned by
            the agent in an implementation-dependent
            manner, and corresponding rows will be
            instantiated in the ifTable. The status of
            this row will become active as soon as the
            agent assigns the interface index,
```

```
            regardless of whether the interface is
            operationally up.
            Deleting a row in this table will likewise
            delete the corresponding row in the ifTable.
            The entry may not be deleted or deactivated
            by setting its value to 'destroy' or
            'notInService' if there is any associated
            entry in sixtofourRelayTable."
    ::= { sixtofourIfEntry 10 }



sixtofourRelayTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF SixtofourRelayEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The table containing information on the
            Relay Routers each 6to4 pseudo-interface is
            configured to use when sending packets to
            native IPv6 domains."

    ::= { sixtofour 4 }

sixtofourRelayEntry OBJECT-TYPE
    SYNTAX     SixtofourRelayEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry containing the information for the
            relay routers of a particular 6to4
            interface.
            Entries in the sixtofourRelayTable are
            created and deleted using the
            sixtofourRelayRowStatus object. All writable
            objects can change values while
            sixtofourRelayRowStatus is 'active'."
    INDEX      {  sixtofourRelayIndex }
    ::= { sixtofourRelayTable 1 }

SixtofourRelayEntry ::= SEQUENCE {
    sixtofourRelayIndex                 Integer32,
    sixtofourRelayIfIndex               Integer32,
    sixtofourRelayDestAddr              InetAddressIPv6,
    sixtofourRelayDestAddrPrefix        InetAddressIPv6,
    sixtofourRelayAddress               InetAddressIPv6,
    sixtofourRelayAnycastStatus         INTEGER,
    sixtofourRelayRouteMetric           Integer32,
    sixtofourRelayRowStatus             RowStatus

    }

sixtofourRelayIndex OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "A unique non-zero value identifying the
            particular relay router for the specific
            6to4 interface."
    ::= { sixtofourRelayEntry 1 }
```

```
sixtofourRelayDestAddr OBJECT-TYPE
      SYNTAX     InetAddressIPv6
      MAX-ACCESS read-create
      STATUS     current
      DESCRIPTION
             "The address of the native ipv6 network
             to which the packets will be send
             through the specific relay router. The
             prefix of the network is given by the
             sixtofourRelayDestPrefix"
    ::= { sixtofourRelayEntry 2 }


sixtofourRelayDestAddrPrefix OBJECT-TYPE
      SYNTAX     InetAddressIPv6
      MAX-ACCESS read-create
      STATUS     current
      DESCRIPTION
             "The subnet mask (prefix) associated with
             the IPv6 address sixtofourRelayDestAddr of
             this entry.The value of the prefix is an
             IPv6 address with all the network bits set
             to 1 and all the hosts bits set to 0."
      ::= { sixtofourRelayEntry 3 }



sixtofourRelayAddress OBJECT-TYPE
      SYNTAX     InetAddressIPv6
      MAX-ACCESS read-create
      STATUS     current
      DESCRIPTION
             "The address of the relay router which is
             used by the specific interface to send
             packets to a native ipv6 domain. "
    ::= { sixtofourRelayEntry 4 }

sixtofourRelayAnycastStatus OBJECT-TYPE
      SYNTAX     INTEGER {
                   anycastAddr(1),     -- use the anycast
                   configuredAddr(2)  -- not use the anycast
                          }
      MAX-ACCESS read-only
      STATUS     current
      DESCRIPTION
             "The value anycastAddr(1) indicates that
             this entry uses as relay route the anycast
             address 2002:c058:6301:: for the nearest
             relay router.The configuredAddr(2) means
             that this entry does not use the anycast
             address but a route for a specific relay
             router, configured by the administrator."
    ::= { sixtofourRelayEntry 5 }



sixtofourRelayRouteMetric  OBJECT-TYPE
      SYNTAX     Integer32
      MAX-ACCESS read-create
      STATUS     current
```

```
      DESCRIPTION
              "The primary routing metric for this route.
              The semantics of this metric are determined
              by the routing-protocol specified in the
              route's ipRouteProto value.  If this metric
              is not used,its value should be set to -1."
        ::= { sixtofourRelayEntry 6 }


sixtofourRelayRowStatus OBJECT-TYPE
     SYNTAX     RowStatus
     MAX-ACCESS read-create
     STATUS     current
     DESCRIPTION
              "The status column used for creating,
              modifying, and deleting instances of the
              columnar objects in the Relay Routers'
              table.
              In order a new entry (row) to be created
              for the sixtofourRelayTable, the object
              sixtofourRelayIfIndex has to be assigned
              with a valid value of the index
              corresponding the 6to4 interface associated
              with the specific route. The
              sixtofourRelayRowStatus can be set to
              'active' if all objects except
              sixtofourRelayRouteMetric have already been
              assigned."
    ::= { sixtofourRelayEntry 7 }




-- Security, Sanity Checks and Statistic Information For The 6to4
-- Mechanism


sixtofourSecurity  OBJECT IDENTIFIER ::= { sixtofour 5 }
sixtofourACL       OBJECT IDENTIFIER ::= { sixtofourSecurity 1 }
sixtofourTunStats  OBJECT IDENTIFIER ::= { sixtofourSecurity 2 }
sixtofourACLInV6   OBJECT IDENTIFIER ::= { sixtofourACL 1 }
sixtofourACLInV4   OBJECT IDENTIFIER ::= { sixtofourACL 2 }
sixtofourACLOutV6  OBJECT IDENTIFIER ::= { sixtofourACL 3 }
sixtofourACLOutV4  OBJECT IDENTIFIER ::= { sixtofourACL 4 }


-- The Access Control List for the case of IPv6 packets
from the     -- local 6to4 site arriving(direction in)
at the 6to4 router in order -- to be forwarded to the
6to4 pseudo-interface where the             --
encapsulation will take place.

sixtofourACLInV6Action OBJECT-TYPE
     SYNTAX     INTEGER {
        deny(1),        -- not allow the packets match with the
                        -- entries of the table and allow all the
                        -- others

        allow(2)        -- allow only those which match with the
                        -- entries of the table and not allow all the
                        -- others
```

```
MAX-ACCESS read-write
     STATUS     current
     DESCRIPTION
            "The general action to be used for this
            table. A value of 1(deny) indicates that the
            ACL filters out allow the packets that match
            with the entries of the table and allows all
            the others. A value of 2(allow) indicates
            that the ACL allows the packets that match
            with the entries of the table and filters out
            all the others"
     ::= { sixtofourACLInV6 1 }


sixtofourACLInV6Table OBJECT-TYPE
     SYNTAX     SEQUENCE OF SixtofourACLInV6Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
          "The Access Control List's Table for the case of
          IPv6 packets from the local 6to4 site arriving
          the 6to4 router, where they will be forwarded to
          the 6to4 interface for encapsulation.
          The following checks are necessary for the secure
          operation of the 6to4 mechanism. If any of the
          following addresses are not filter out by the
          tunnel device of the system, they should be added
          in the list, so that packets with the addresses
          given below will be rejected.
          IPv6 Sanity Checks (destination address):
           0::/16 (compatible, mapped addresses, loopback,
                   unspecified, ...)
           fe80::/10 (link-local)
           fec0::/10 (site-local)
           ff00::/8 (any multicast)

          V4ADDR of the source 6to4(2002::V4ADDR/48)
          adrress different from IPv4 address of the
          external interface of the 6to4 site.”
     ::= { sixtofourACLInV6 2}


 sixtofourACLInV6Entry OBJECT-TYPE
     SYNTAX     SixtofourACLInV6Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "An entry (conceptual row) containing the
            information on a particular rule of this 6to4
            interface"
     INDEX   { sixtofourACLInV6IfIndex,
               sixtofourACLInV6Index }
     ::= { sixtofourACLInV6Table 1 }

 SixtofourACLInV6Entry ::= SEQUENCE {
         sixtofourACLInV6IfIndex               Integer32,
         sixtofourACLInV6Index                 Integer32,
         sixtofourACLInV6Status                INTEGER,
         sixtofourACLInV6SrcAddr               InetAddressIPv6,
         sixtofourACLInV6SrcAddrPfxLgth        Integer32,
         sixtofourACLInV6DstAddr               InetAddressIPv6,
```

```
        sixtofourACLInV6DstAddrPfxLgth        Integer32,
        sixtofourACLInV6Protocol              INTEGER,
        sixtofourACLInV6RowStatus             RowStatus
            }

 sixtofourACLInV6IfIndex     OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "The index value which uniquely identifies the
            6to4 interface to which this entry of the ACL
            is applied. The interface identified by a
            particular value of this index is the same
            interface as identified by the same value of
            sxtofourIfIndex."
     ::= { sixtofourACLInV6Entry 1 }


 sixtofourACLInV6Index OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
             "The value which uniquely identifies the rule
             of the ACL among the rules of the same 6to4
             interface. The way this value is chosen is
             implementation specific but the value of
             sixtofourACLInV6IfIndex/
             sixtofourACLInV6Index pair must be unique and
             remain constant for the life of the
             interface."
     ::= { sixtofourACLInV6Entry 2 }


sixtofourACLInV6Status OBJECT-TYPE
     SYNTAX     INTEGER {
        on(1),        -- apply the specific rule

        off(2)        -- do not apply the rule
     }
     MAX-ACCESS read-create
     STATUS     current
     DESCRIPTION
            "The status of this rule. A value of 1
            indicates that the rule of this entry is
            applied to every packet. The value of 2
            indicates that this rule is not to be used.
            If the status of the rule is not assigned the
            system gives the value 2."
     ::= { sixtofourACLInV6Entry 3 }


 sixtofourACLInV6SrcAddr  OBJECT-TYPE
     SYNTAX     InetAddressIPv6
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
            "The address to be compared with the IPv6
            source address of the incoming packet, after
            the mask sixtofourACLInV6DstAddrPfxLgth has
```

```
            been applied to the source address."
        ::= { sixtofourACLInV6Entry 4 }


    sixtofourACLInV6SrcAddrPfxLgth   OBJECT-TYPE
        SYNTAX     Integer32 (0..128)
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
               "The length of the prefix (mask) to be applied
               at the IPv6 source address of the incoming
               packets in order to match the
               sixtofourACLInV6SrcAddr."
        ::= { sixtofourACLInV6Entry 5 }


    sixtofourACLInV6DstAddr    OBJECT-TYPE
        SYNTAX     InetAddressIPv6
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The address to be compared with the IPv6
                destination address of the incoming packet,
                after the sixtofourACLInV6DstAddrPfxLgth has
                been applied to the destination address."
        ::= { sixtofourACLInV6Entry 6 }


    sixtofourACLInV6DstAddrPfxLgth     OBJECT-TYPE
        SYNTAX     Integer32 (0..128)
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
               "The length of the prefix (mask) to be applied
               at the IPv6 destination address of the
               incoming packets in order to match the
               sixtofourACLInV6DstAddr."
        ::= { sixtofourACLInV6Entry 7 }


    sixtofourACLInV6Protocol OBJECT-TYPE
        SYNTAX     INTEGER {
           tcp(1),
           udp(2),
           icmpv6(3),
           other(4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
               "The type of the upper layer protocol assigned
               in the IPv6 header "
        ::= { sixtofourACLInV6Entry 8 }


    sixtofourACLInV6RowStatus          OBJECT-TYPE
        SYNTAX      RowStatus
        MAX-ACCESS read-create
        STATUS     current
        DESCRIPTION
               "The status of this row, by which new entries
```

```
            may be created, or old entries deleted from
            this table. The agent need not support setting
            this object to createAndWait or notInService
            since the only writable object of this table
            can automatically take a value in case of not
            assigned one by the administrator. Writable
            objects in rows of corresponding tables such
            as the sixtofourIfTable may be modified while
            this row is active. The status of this row is
            set to active only if a valid
            sixtofourACLInV6IfIndex, responding to the
            6to4 interface, and the sixtofourACLInV6Action
            are assigned."
     ::= { sixtofourACLInV6Entry 9 }



-- The Access Control List for the case of IPv4 packets from the
-- tunnel arriving(direction in) the 6to4 pseudo-interface where the
-- decapsulation will take place.

sixtofourACLInV4Action OBJECT-TYPE
     SYNTAX     INTEGER {
        deny(1),        -- not allow the packets match with the
                        -- entries of the table and allow all the
                        -- others

        allow(2)        -- allow only those which match with the
                        -- entries of the table and not allow all the
                        -- others
     }
     MAX-ACCESS read-write
     STATUS     current
     DESCRIPTION
            "The general action to be used for this
            table. A value of 1(deny) indicates that the
            ACL filters out allow the packets that match
            with the entries of the table and allows all
            the others. A value of 2(allow) indicates
            that the ACL allow the packets that match
            with the entries of the table and filters out
            all the others"
     ::= { sixtofourACLInV4 1 }


sixtofourACLInV4Table OBJECT-TYPE
     SYNTAX     SEQUENCE OF SixtofourACLInV4Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
           " The Access Control List's Table for the case
          of the IPv4 packets(where under the IPv4 header
          IPv6 packets are encapsulated) from the tunnel
          arriving the 6to4 interface, where they will be
          decapsulated.
          The following checks are necessary for the secure
          operation of the 6to4 mechanism. If any of the
          following addresses are not filter out by the
          tunnel device of the system, they should be added
          in the list, so that packets with the addresses
```

```
        given below will be rejected.
        IPv4 Sanity Checks (source and destination
        address):
         0.0.0.0/8 (not assigned yet)
         10.0.0.0/8 (private)
         127.0.0.0/8 (loopback)
         172.16.0.0/12 (private)
         192.168.0.0/16 (private)
         169.254.0.0/16 (IANA Assigned
                         DHCP link-local
         224.0.0.0/4 (multicast)
         240.0.0.0/4 (reserved and broadcast).”
    ::= { sixtofourACLInV4 2}


sixtofourACLInV4Entry OBJECT-TYPE
    SYNTAX     SixtofourACLInV4Entry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "An entry (conceptual row) containing the
           information on a particular rule of this 6to4
           interface"
    INDEX   { sixtofourACLInV4IfIndex,
              sixtofourACLInV4Index }
    ::= { sixtofourACLInV4Table 1 }

SixtofourACLInV4Entry ::= SEQUENCE {
        sixtofourACLInV4IfIndex               Integer32,
        sixtofourACLInV4Index                 Integer32,
        sixtofourACLInV4Status                INTEGER,
        sixtofourACLInV4SrcAddr               InetAddressIPv6,
        sixtofourACLInV4SrcAddrPfxLgth        Integer32,
        sixtofourACLInV4DstAddr               InetAddressIPv6,
        sixtofourACLInV4DstAddrPfxLgth        Integer32,
        sixtofourACLInV4RowStatus             RowStatus
            }

sixtofourACLInV4IfIndex     OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The index value which uniquely identifies the
           6to4 interface to which this entry of the ACL
           is applied. The interface identified by a
           particular value of this index is the same
           interface as identified by the same value of
           sxtofourIfIndex."
    ::= { sixtofourACLInV4Entry 1 }


sixtofourACLInV4Index OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The value which uniquely identifies the rule
            of the ACL among the rules of the same 6to4
            interface. The way this value is chosen is
            implementation specific but it must be unique
```

```
            for
            sixtofourACLInV4IfIndex/sixtofourACLInV4Index
            pair and remain constant for the life of the
            interface."
    ::= { sixtofourACLInV4Entry 2 }


sixtofourACLInV4Status OBJECT-TYPE
    SYNTAX     INTEGER {
       on(1),        -- apply the specific control

       off(2)        -- do not apply the control
    }
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
           "The status of this rule. A value of 1
           indicates that the rule of this entry is
           applied to every packet. The value of 2
           indicates that this rule is not applied at the
           packets.
           If the status of the rule is not assigned the
           system gives the value 2."
    ::= { sixtofourACLInV4Entry 3 }


 sixtofourACLInV4SrcAddr  OBJECT-TYPE
    SYNTAX     IpAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The address to be compared with the IPv4
           source address of the incoming packet, after
           the mask sixtofourACLInV4DstAddrPfxLgth has
           been applied to the source address."
    ::= { sixtofourACLInV4Entry 4 }


  sixtofourACLInV4SrcAddrPfxLgth   OBJECT-TYPE
    SYNTAX     Integer32 (0..32)
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The length of the prefix to be applied at the
           IPv4 source address of the incoming packets in
           order to match the sixtofourACLInV4SrcAddr."
    ::= { sixtofourACLInV4Entry 5 }


 sixtofourACLInV4DstAddr    OBJECT-TYPE
    SYNTAX     IpAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The address to be compared with the IPv4
            destination address of the incoming packets,
            after the sixtofourACLInV4DstAddrPfxLgth has
            been applied to the destination address."
    ::= { sixtofourACLInV4Entry 6 }
```

```
 sixtofourACLInV4DstAddrPfxLgth     OBJECT-TYPE
     SYNTAX     Integer32 (0..32)
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
            "The length of the prefix to be applied at the
            IPv4 destination address of the incoming
            packets in order to match the
            sixtofourACLInV4DstAddr."
     ::= { sixtofourACLInV4Entry 7 }


  sixtofourACLInV4RowStatus          OBJECT-TYPE
     SYNTAX      RowStatus
     MAX-ACCESS read-create
     STATUS     current
     DESCRIPTION
            "The status of this row, by which new entries
            may be created, or old entries deleted from
            this table. The agent need not support setting
            this object to createAndWait or notInService
            since the only writable object of this table
            can automatically take a value in case of not
            assigned one by the administrator. Writable
            objects in rows of corresponding tables such
            as the sixtofourIfTable may be modified while
            this row is active. The status of this row is
            set to active only if a valid
            sixtofourACLInV6IfIndex, responding to the
            6to4 interface, and the sixtofourACLInV6Action
            are assigned."

     ::= { sixtofourACLInV4Entry 8 }



-- The Access Control List for the case of the IPv6 packets having
-- decapsulated and leaving the 6to4 pseudo-interface with
-- destination the local 6to4 site.

sixtofourACLOutV6Action OBJECT-TYPE
     SYNTAX     INTEGER {
        deny(1),        -- not allow the packets match with the
                        -- entries of the table and allow all the
                        -- others

        allow(2)        -- allow only those which match with the
                        -- entries of the table and not allow all the
                        -- others
     }
     MAX-ACCESS read-write
     STATUS     current
     DESCRIPTION
            "The general action to be used for this table.
            A value of 1(deny) indicates that the ACL
            filters out allow the packets that match with
            the entries of the table and allows all the
            others. A value of 2(allow) indicates that the
            ACL allow the packets that match with the
            entries of the table and filters out all the
            others"
```

```
    ::= { sixtofourACLOutV6 1 }


sixtofourACLOutV6Table OBJECT-TYPE
     SYNTAX     SEQUENCE OF sixtofourACLOutV6Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
          " The Access Control List's Table for the case
          of the IPv6 packets having decapsulated and
          leaving the 6to4 pseudo-interface with
          destination the 6to4 site.
          The following checks are necessary for the secure
          operation of the 6to4 mechanism. If any of the
          following addresses are not filter out by the
          tunnel device of the system, they should be added
          in the list, so that packets with the addresses
          given below will be rejected.
          IPv6 Sanity Checks (source address)
          V4ADDR of the IPv6 destination address of the
          packet (2002:V4ADDR::/48) different from the
          site's IPv4 address.
          If the IPv6 source address is 2002::/16 and
          V4ADDR of the source packet is different from the
          IPv4 address of the source.”
     ::= { sixtofourACLOutV6 2}


 sixtofourACLOutV6Entry OBJECT-TYPE
     SYNTAX     SixtofourACLOutV6Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "An entry (conceptual row) containing the
            information on a particular rule of this 6to4
            interface"
     INDEX   { sixtofourACLOutV6IfIndex,
               sixtofourACLOutV6Index }
     ::= { sixtofourACLOutV6Table 1 }


 SixtofourACLOutV6Entry ::= SEQUENCE {
         sixtofourACLOutV6IfIndex               Integer32,
         sixtofourACLOutV6Index                 Integer32,
         sixtofourACLOutV6Status                INTEGER,
         sixtofourACLOutV6SrcAddr               InetAddressIPv6,
         sixtofourACLOutV6SrcAddrPfxLgth        Integer32,
         sixtofourACLOutV6DstAddr               InetAddressIPv6,
         sixtofourACLOutV6DstAddrPfxLgth        Integer32,
         sixtofourACLOutV6Protocol              INTEGER,
         sixtofourACLOutV6RowStatus             RowStatus
             }

 sixtofourACLOutV6IfIndex     OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "The index value which uniquely identifies the
            6to4 interface to which this entry of the ACL
            is applied. The interface identified by a
```

```
            particular value of this index is the same
            interface as identified by the same value of
            sxtofourIfIndex."
        ::= { sixtofourACLOutV6Entry 1 }


 sixtofourACLOutV6Index OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The value which uniquely identifies the rule
                of the ACL among the rules of the same 6to4
                interface. The way this value is chosen is
                implementation specific but it must be unique
                for sixtofourACLOutV6IfIndex/
                sixtofourACLOutV6Index pair and remain
                constant for the life of the interface."
        ::= { sixtofourACLOutV6Entry 2 }


sixtofourACLOutV6Status OBJECT-TYPE
        SYNTAX     INTEGER {
           on(1),         -- apply the specific rule

           off(2)         -- do not apply the rule
        }
        MAX-ACCESS read-create
        STATUS     current
        DESCRIPTION
               "The status of this rule. A value of 1
               indicates that the rule of this entry is
               applied to every packet. The value of 2
               indicates that this rule is not applied at the
               packets.
               If the status of the rule is not assigned the
               system gives the value 2."
        ::= { sixtofourACLOutV6Entry 3 }



 sixtofourACLOutV6SrcAddr  OBJECT-TYPE
        SYNTAX     InetAddressIPv6
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
               "The address to be compared with the IPv6
               source address of the outgoing packet, after
               the sixtofourACLOutV6DstAddrPfxLgth has been
               applied to the source address."
        ::= { sixtofourACLOutV6Entry 4 }


  sixtofourACLOutV6SrcAddrPfxLgth   OBJECT-TYPE
        SYNTAX     Integer32 (0..128)
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
               "The length of the prefix to be applied at the
               source address of the outgoing packets in
               order to match the sixtofourACLOutV6SrcAddr."
```

```
    ::= { sixtofourACLOutV6Entry 5 }


sixtofourACLOutV6DstAddr    OBJECT-TYPE
    SYNTAX     InetAddressIPv6
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The address to be compared with the IPv6
            destination address of the outgoing packet,
            after the sixtofourACLOutV6DstAddrPfxLgth has
            been applied to the destination address."
    ::= { sixtofourACLOutV6Entry 6 }


sixtofourACLOutV6DstAddrPfxLgth     OBJECT-TYPE
    SYNTAX     Integer32 (0..128)
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The length of the prefix to be applied at the
           destination address of the outgoing packets in
           order to match the sixtofourACLOutV6DstAddr"
    ::= { sixtofourACLOutV6Entry 7 }


 sixtofourACLOutV6Protocol OBJECT-TYPE
    SYNTAX     INTEGER {
       tcp(1),
       udp(2),
       icmpv6(3),
       other(4)
    }
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The type of the upper layer protocol assigned
           in the IPv6 header "
    ::= { sixtofourACLOutV6Entry 8 }


 sixtofourACLOutV6RowStatus          OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
           "The status of this row, by which new entries
           may be created, or old entries deleted from
           this table. The agent need not support setting
           this object to createAndWait or notInService
           since the only writable object of this table
           can automatically take a value in case of not
           assigned one by the administrator. Writable
           objects in rows of corresponding tables such
           as the sixtofourIfTable may be modified while
           this row is active. The status of this row is
           set to active only if a valid
           sixtofourACLInV6IfIndex, responding to the
           6to4 interface, and the sixtofourACLInV6Action
           are assigned."
```

```
     ::= { sixtofourACLOutV6Entry 9 }



-- The Access Control List for the case of the IPv4 packets, where
-- under the IPv4 header are the encapsulated IPv6 packets, leaving
-- the 6to4 pseudo-interface through the tunnel.

sixtofourACLOutV4Action OBJECT-TYPE
     SYNTAX     INTEGER {
        deny(1),        -- not allow the packets match with the
                        -- entries of the table and allow all the
                        -- others

        allow(2)        -- allow only those which match with the
                        -- entries of the table and not allow all the
                        -- others
     }
     MAX-ACCESS read-write
     STATUS     current
     DESCRIPTION
            "The general action to be used for this table.
            A value of 1(deny) indicates that the ACL
            filters out allow the packets that match with
            the entries of the table and allows all the
            others. A value of 2(allow) indicates that the
            ACL allow the packets that match with the
            entries of the table and filters out all the
            others"
     ::= { sixtofourACLOutV4 1 }



sixtofourACLOutV4Table OBJECT-TYPE
     SYNTAX     SEQUENCE OF sixtofourACLOutV4Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
           " The Access Control List's Table for the case
          of the IPv4 packets(where under the IPv4 header
          are the encapsulated IPv6 packets) leaving the
          6to4 pseudo-interface through the tunnel.
          The following checks are necessary for the secure
          operation of the 6to4 mechanism. If any of the
          following addresses are not filter out by the
          tunnel device of the system, they should be added
          in the list, so that packets with the addresses
          given below will be rejected.
          IPv4 Sanity Checks (source and destination
          address):
           0.0.0.0/8 (not assigned yet)
           10.0.0.0/8 (private)
           127.0.0.0/8 (loopback)
           172.16.0.0/12 (private)
           192.168.0.0/16 (private)
           169.254.0.0/16 (IANA Assigned
                           DHCP link-local
           224.0.0.0/4 (multicast)
           240.0.0.0/4 (reserved and broadcast).”
     ::= { sixtofourACLOutV4 2}
```

```
 sixtofourACLOutV4Entry OBJECT-TYPE
     SYNTAX     SixtofourACLOutV4Entry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "An entry (conceptual row) containing the
            information on a particular rule of this 6to4
            interface"
     INDEX   { sixtofourACLOutV4IfIndex,
               sixtofourACLOutV4Index }
     ::= { sixtofourACLOutV4Table 1 }

 SixtofourACLOutV4Entry ::= SEQUENCE {
         sixtofourACLOutV4IfIndex               Integer32,
         sixtofourACLOutV4Index                 Integer32,
         sixtofourACLOutV4Status                INTEGER,
         sixtofourACLOutV4SrcAddr               InetAddressIPv6,
         sixtofourACLOutV4SrcAddrPfxLgth        Integer32,
         sixtofourACLOutV4DstAddr               InetAddressIPv6,
         sixtofourACLOutV4DstAddrPfxLgth        Integer32,
         sixtofourACLOutV4RowStatus             RowStatus
             }

 sixtofourACLOutV4IfIndex     OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "The index value which uniquely identifies the
            6to4 interface to which this entry of the ACL
            is applied. The interface identified by a
            particular value of this index is the same
            interface as identified by the same value of
            sxtofourIfIndex."
     ::= { sixtofourACLOutV4Entry 1 }


 sixtofourACLOutV4Index OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
             "The value which uniquely identifies the rule
             of the ACL among the rules of the same 6to4
             interface. The way this value is chosen is
             implementation specific but it must be unique
             for
             sixtofourACLOutV4IfIndex/sixtofourACLOutV4Ind
             ex pair and remain constant for the life of
             the interface."
     ::= { sixtofourACLOutV4Entry 2 }


sixtofourACLOutV4Status OBJECT-TYPE
     SYNTAX     INTEGER {
        on(1),        -- apply the specific rule

        off(2)        -- do not apply the rule
     }
```

```
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
           "The status of this rule. A value of 1
           indicates that the rule of this entry is
           applied to every packet. The value of 2
           indicates that this rule is not applied at the
           packets.
           If the status of the rule is not assigned the
           system gives the value 2."
    ::= { sixtofourACLOutV4Entry 3 }


sixtofourACLOutV4SrcAddr  OBJECT-TYPE
    SYNTAX     IpAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The address to be compared with the IPv4
           source address of the external IPv4 header of
           the outgoing packet, after the
           sixtofourACLOutV4DstAddrPfxLgth has been
           applied to the source address."
    ::= { sixtofourACLOutV4Entry 4 }


 sixtofourACLOutV4SrcAddrPfxLgth   OBJECT-TYPE
    SYNTAX     Integer32 (0..32)
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The length of the prefix to be applied at the
           source address of the external IPv4 header of
           the outgoing packets in order to match the
           sixtofourACLOutV4SrcAddr."
    ::= { sixtofourACLOutV4Entry 5 }


sixtofourACLOutV4DstAddr    OBJECT-TYPE
    SYNTAX     IpAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The address to be compared with the
            destination address of the external IPv4
            header of the outgoing packet, after the
            sixtofourACLOutV4DstAddrPfxLgth has been
            applied."
    ::= { sixtofourACLOutV4Entry 6 }


sixtofourACLOutV4DstAddrPfxLgth   OBJECT-TYPE
    SYNTAX     Integer32 (0..32)
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The length of the prefix to be applied at the
           destination address of the external IPv4
           header of the outgoing packets in order to
           match the sixtofourACLOutV4DstAddr"
```

```
     ::= { sixtofourACLOutV4Entry 7 }


  sixtofourACLOutV4RowStatus   OBJECT-TYPE
     SYNTAX      RowStatus
     MAX-ACCESS read-create
     STATUS     current
     DESCRIPTION
            "The status of this row, by which new entries
            may be created, or old entries deleted from
            this table. The agent need not support setting
            this object to createAndWait or notInService
            since the only writable object of this table
            can automatically take a value in case of not
            assigned one by the administrator. Writable
            objects in rows of corresponding tables such
            as the sixtofourIfTable may be modified while
            this row is active. The status of this row is
            set to active only if a valid
            sixtofourACLInV6IfIndex, responding to the
            6to4 interface, and the sixtofourACLInV6Action
            are assigned."

     ::= { sixtofourACLOutV4Entry 8 }



-- The Table with the statistics for every 6to4 tunnel established to
-- the system. This table will help for the determination of
-- malicious actions which are either against the 6to4 mechanism or
-- they are making use of the mechanism for other attacks, from users
-- inside and outside the 6to4 site.


sixtofourTunStatsLogRotation   OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The time (in minutes) the statistic information
            about the tunnel will be kept by the system,
            before rotating the data to a new file"
     ::= { sixtofourTunStats 1}



sixtofourTunStatsTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF SixtofourTunStatsEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The (conceptual) table containing the statistic
            information on every 6to4 tunnel established by
            the system ."
    ::= { sixtofourTunStats 2 }


sixtofourTunStatsEntry OBJECT-TYPE
    SYNTAX     SixtofourTunStatsEntry
```

```
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry (conceptual row) containing the
            information about an established tunnel for the
            inderface identified by the value of
            sixtofourTunStatsIfIndex  with destination the
            address identified by the
            sixtofourTunStatsRmtAddr."

    INDEX      {  sixtofourTunStatsRmtAddr,
                  sixtofourTunStatsIfIndex,
                  sixtofourTunStatsIndex     }
    ::= { sixtofourTunStatsTable 1 }


SixtofourTunStatsEntry ::= SEQUENCE {
    sixtofourTunStatsRmtAddr        InetAddressIPv6,
    sixtofourTunStatsIfIndex        Integer32,
    sixtofourTunStatsIndex          Integer32,
    sixtofourTunStatsSrcAddr        InetAddressIPv6,
    sixtofourTunStatsStrtTime       TimeTicks,
    sixtofourTunStatsEndTime        TimeTicks,
    sixtofourTunStatsRelayAddr      IpAddress
        }


sixtofourTunStatsRmtAddr OBJECT-TYPE
     SYNTAX     InetAddressIPv6
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
            "The destination IPv6 address of the packet. The
            destination might be either a 6to4 or a 'native'
            IPv6 address."
     ::= { sixtofourTunStatsEntry 1 }



sixtofourTunStatsIfIndex   OBJECT-TYPE
      SYNTAX  Integer32
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
               " The index value which uniquely identifies
               the 6to4 interface used for this entry's
               tunnel. The interface identified by a
               particular value of this index is the same
               interface as identified by the same value of
               sxtofourIfIndex."
        ::= { sixtofourTunStatsEntry 2 }


sixtofourTunStatsIndex    OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION
             "The value which uniquely identifies this tunnel
             among the other established from the same 6to4
             pseudo-interface for the same destination
```

```
              address.
              The way this value is chosen is implementation
              specific but it must be unique for the
              sixtofourTunStatsRmtAddr/
              sixtofourTunStatsIfIndex pair."
       ::= { sixtofourTunStatsEntry 3 }


sixtofourTunStatsSrcAddr   OBJECT-TYPE
    SYNTAX     InetAddressIPv6
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The IPv6 address of the local 6to4 host who
            exchanged traffic, with the node with address the
            sixtofourTunStatsRmtAddr, through this tunnel"
     ::= { sixtofourTunStatsEntry 4}


sixtofourTunStatsStrtTime   OBJECT-TYPE
     SYNTAX     TimeTicks
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
            " The value of sysUpTime when the the 6to4 tunnel
            established.  "
     ::= { sixtofourTunStatsEntry 5 }


sixtofourTunStatsEndTime   OBJECT-TYPE
     SYNTAX      TimeTicks
     MAX-ACCESS  read-only
     STATUS      current
     DESCRIPTION
            " The value of sysUpTime when the the 6to4 tunnel
            ended."
    ::= { sixtofourTunStatsEntry 6 }


sixtofourTunStatsRelayAddr   OBJECT-TYPE
        SYNTAX     IpAddress
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
            "The IPv4 address of the relay used in the case
            of the communication of a local 6to4 user with a
            'native' IPv6 user. If the communication took
            place among 6to4 users the value of this object
            is set to 0.0.0.0. This is also useful for the
            trace of a malicious IPv4 node that pretends to
            be the Relay"
     ::= { sixtofourTunStatsEntry 7 }



END
```

6. ΥΛΟΠΟΙΗΣΗ ΤΗΣ 6TO4 MIB

Η υλοποίηση της 6to4 MIB έγινε σε λειτουργικό σύστημα Suse Linux, έκδοση 9.2, με πυρήνα 2.6.8. Για την υλοποίηση του agent χρησιμοποιήθηκε η τελευταία έκδοση του NET-SNMP 5.2.1 [*www.net-snmp.org*] και ο κώδικας του προγράμματος του agent γράφτηκε σε γλώσσα προγραμματισμού C . Στο παράρτημα Α παρατίθεται γενικό εγχειρίδιο λειτουργίας της τελευταίας έκδοσης του NET-SNMP, ιδιαίτερα για τα χαρακτηριστικά της που χρησιμοποιήθηκαν για την συγκεκριμένη υλοποίηση της MIB.

Αν και για την υλοποίηση, όπως αναφέρθηκε χρησιμοποιήθηκε η τελευταία έκδοση του συγκεκριμένου λειτουργικού, εντούτοις, όπως αποδείχθηκε, η υποστήριξη που παρέχουν γενικά τα LINUX distributions για το IPv6 είναι σε αρχικό-ερευνητικό στάδιο, με πολλές ελλείψεις και λίγο documentation τόσο για το IPv6 όσο, και περισσότερο, για τον 6to4 μηχανισμό. Αυτό κατέστησε εξαιρετικά δύσκολη και χρονοβόρα την αναζήτηση των διαφόρων πληροφοριών που έπρεπε να αντλεί ο agent, για να επιστρέφεται αυτή η πληροφορία, μέσω του SNMP agent. Στις περισσότερες περιπτώσεις η άντληση των πληροφοριών που χρειάζονται για το κάθε αντικείμενο έγινε μέσα από το /proc filesystem του LINUX, και μόνο σαν τελευταία επιλογή χρησιμοποιήθηκαν κάποια sockets, για τα οποία το documentation είναι ανύπαρκτο.

Επίσης, λόγω της έλλειψης documentation για τα εγγράψιμα (settable) αντικείμενα στην τελευταία έκδοση του NET-SNMP αλλά και της περιπλοκότητας υλοποίησης τους, τα αντικείμενα της MIB υλοποιήθηκαν μόνο για ανάγνωση (read-only), χωρίς δυνατότητα να τεθεί η τιμή του αντικειμένου μέσω SNMP. Το configuration των 6to4 interfaces χρειάζεται να γίνεται manual από τον διαχειριστή.

Εκτός όμως από την τελευταία του αδυναμία όσον αφορά τα settable αντικείμενα, κατά τα αλλά η τελευταία έκδοση του NET-SNMP προσφέρει αρκετές βελτιώσεις σε σχέση με παλαιότερες εκδόσεις και εργαλεία που κάνουν το έργο του προγραμματιστή απλούστερο. Συγκεκριμένα, με προσθήκη κάποιων configuration αρχείων που καλούνται από το εργαλείο mib2c σαν παράμετροι, ο προγραμματιστής έχει την δυνατότητα να παράγει τον κύριο σκελετό του προγράμματος για τα διάφορα mib-modules που αργότερα θα ενσωματωθούν στον agent, μόνο από τον ορισμό της MIB. Ειδικά για την sixtofourMIB χρησιμοποιήθηκαν οι πιο κάτω εντολές:

- Περίπτωση scalar αντικειμένου που δεν ανήκει σε κάποιο πίνακα:

  mib2c –c mib2c.scalar.conf <ονομα αντικειμενου>

- Περίπτωση πίνακα:

  mib2c –c mib2c.mfd.conf <ονομα πινακα>

Με αυτό τον τρόπο απαλλάσσεται πλέον ο προγραμματιστής από την ανάγκη για βαθύτερη γνώση του SNMP πρωτοκόλλου για δημιουργία τέτοιων εφαρμογών SNMP. Ιδιαίτερα στην δεύτερη περίπτωση όπου η διαδικασία είναι πιο περίπλοκη λόγω του πίνακα με πολλαπλά αντικείμενα και των δεικτών, το *mib2c.mdf.conf* αρχείο, το οποίο εντάχθηκε στο NET-SNMP από την έκδοση 5 και μετά, απλοποίησε

σε μεγάλο βαθμό την διαδικασία της υλοποίησης πινάκων, κάτι που ήταν ιδιαίτερα πολύπλοκο σε προηγούμενες εκδόσεις. Συγκεκριμένα, όταν κληθεί σαν παράμετρος του mib2c το συγκεκριμένο configuration αρχείο παράγει σε γλώσσα C τον σκελετό του κώδικα που θα χρησιμοποιηθεί για την υλοποίηση του πίνακα, με τις κατάλληλες δομές δεδομένων και τις συναρτήσεις που θα καλέσει ο agent. Έτσι, ουσιαστικά η εργασία του προγραμματιστή περιορίζεται στα:

- Πως και από που θα αντλήσει τα δεδομένα για τα αντικείμενα του πίνακα
- Στη διαχείριση των δεδομένων αυτών:
    - Να τα φέρει δηλαδή στην μορφή που καταλαβαίνει ο agent
    - Να τα αντιστοιχίσει τους με τις ASN τιμές που θα επιστρέφουν (περιπτώσεις αντικειμένων τύπου INTEGER και αντικείμενα DisplayString με μεταβλητό μέγεθος)
- Στην διαχείριση των δεικτών (ανάθεση δηλαδή δείκτη/ων σε κάθε γραμμή του πίνακα, προσθαφαίρεση δεικτών κ.ο.κ).

Για την συγκεκριμένη υλοποίηση της MIB, και στους δυο πίνακες που υλοποιήθηκαν (*sixtofourIfTable* και *sixtofourRelayTable*) χρησιμοποιήθηκαν για την φύλαξη και διαχείριση των δεικτών συνδεδεμένες λίστες, έτσι ώστε να γίνεται εύκολα τόσο προσθήκη νέων 6to4 interfaces και routes προς relay routers αντίστοιχα, καθώς και διαγραφή τους.
Επιπλέον, σημαντική βελτίωση που έχει γίνει στη νέα έκδοση του NET-SNMP είναι ότι πλέον δεν χρειάζεται ξεχωριστή υλοποίηση για το κάθε είδος SNMP αίτησης. Δηλαδή, στην συγκεκριμένη περίπτωση δεν χρειάζεται να υλοποιηθούν ξεχωριστά η *snmpget* και η *snmpgetnext* περίπτωση, αφού πλέον ο προγραμματιστής πρέπει να καθορίσει μόνο την εύρεση και επεξεργασία των δεδομένων, τα οποία τοποθετεί στον container από όπου πλέον ο agent ανάλογα με την κάθε αίτηση παίρνει τα κατάλληλα δεδομένα που χρειάζεται.

Όσον αφορά την εύρεση και άντληση των δεδομένων, οι τιμές για τα περισσότερα αντικείμενα αντλούνται μέσα από το /proc - filesystem του Linux, με χρήση ενός file descriptor . Μόνο σε δυο περιπτώσεις όπου χρειάστηκε η IPv4 διεύθυνση κάποιου interface χρησιμοποιήθηκαν sockets.

Ο κύριος όγκος των πληροφοριών που επιστρέφουν τα αντικείμενα της MIB πάρθηκαν από τα αρχεία του ***/proc/net/***, όπου μέσα φυλάσσονται όλες οι πληροφορίες που αφορούν στη δικτύωση (networking) του μηχανήματος, και τα οποία είναι read-only αρχεία. Συγκεκριμένα χρησιμοποιήθηκαν από το */proc/net/* τα αρχεία */proc/net/if_inet6* , */proc/net/ipv6_route* , */proc/net/dev* καθώς και το sub-directory */proc/net/dev_snmp6/*. Επιπλέον, εκτός από το */proc/net/* χρειάστηκε και η άντληση κάποιων στοιχείων από το ***/proc/sys/net/ipv6/***. Γενικά στο /proc/sys/ βρίσκονται οι διάφορες παράμετροι που μπορούν να μεταβληθούν, και τα αρχεία αυτά είναι read-write. Στην συνεχεία περιγράφονται τα δυο πιο σημαντικά αρχεία για την υλοποίηση της MIB, το καθένα από τα οποία χρησιμοποιήθηκε για την άντληση των δεδομένων από τους δυο πίνακες της MIB αντίστοιχα, και ακολούθως αναφέρεται συνοπτικά η διαδικασία υλοποίησης των διαφόρων αντικειμένων.

6.1 Εγγραφές του /proc - filesystem

6.1.1 Αρχείο /proc/net/ if_inet6

Στο συγκεκριμένο αρχείο δίνονται όλα τα interfaces (physical και virtual) που έχουν εγκατασταθεί και ρυθμιστεί στο σύστημα. Έτσι, για την περίπτωση της υλοποίησης της 6to4 MIB μπορούμε να δούμε αν υπάρχουν σηκωμένα 6to4 interfaces στο σύστημα (διευθύνσεις με πρόθεμα 2002::) , ποσά υπάρχουν και ποιες οι 6to4 διευθύνσεις τους.
Όλες οι διευθύνσεις στις οποίες έγιναν configure τα interfaces δίνονται με ειδικό format, χωρίς τον διαχωρισμό **:** ανά δυο byte (4 δεκαεξαδικά ψηφία) όπως είναι το κανονικό format της IPv6 διεύθυνσης. Στο παράδειγμα πιο κάτω φαίνεται μόνο το 6to4 interface.

```
# cat /proc/net/if_inet6
20029366e42200000000000000000001 05 30 00 80 tun6to4
+--------------------------------+ ++ ++ ++ ++ +-----+
|                                  |  |  |  |  |
1                                  2  3  4  5  6
```

όπου:

1. Η IPv6 διεύθυνση σε 32 δεκαεξαδικούς χαρακτήρες, χωρίς την άνω και κάτω τελεία (:) για διαχωρισμό.
2. Netlink device number (interface index) σε δεκαεξαδική μορφή. (ο αύξων αριθμός της κάθε έγγραφη όπως φαίνεται μετά από την κλήση της *ip addr*)
3. Το μέγεθος(μήκος σε bytes) του προθέματος σε δεκαεξαδική μορφή.(site prefix)
4. Scope value (για συμβατότητα με παλαιότερες εκδόσεις πυρήνα)
5. Interface flags
6. Το όνομα του interface.

6.1.2 Αρχείο /proc/net/ ipv6_route

Στο αρχείο αυτό υπάρχουν τα IPv6 δρομολόγια (routes) που αντιστοιχούν στο κάθε interface. Και σε αυτό το αρχείο, οι IPv6 διευθύνσεις δίνονται με ειδικό format, χωρίς τον χαρακτήρα : για διαχωρισμό. Στο πιο κάτω παράδειγμα φαίνεται μια έγγραφη για το 6to4 interface.

```
# cat /proc/net/ipv6_route

3ffe0000000000000000000000000000 10 00000000000000000000000000000000 00
+--------------------------------+ ++ +--------------------------------+ ++
|                                  |  |                                  |
1                                  2  3                                  4

00000000000000000000000c25f6c5b 00000001 00000001 00000001 00000003 tun6to4
+--------------------------------+ +------+ +------+ +------+ +------+ +-----+
|                                  |        |        |        |        |
5                                  6        7        8        9        10
```

όπου:

1. Το IPv6 δίκτυο προορισμού σε 32 δεκαεξαδικούς χαρακτήρες, χωρίς τον διαχωρισμό **:** ανά 2 byte.
2. Το μήκος του προθέματος του προορισμού σε δεκαεξαδικούς αριθμούς.
3. Η IPv6 διεύθυνση προελεύσεως του δικτύου από όπου προέρχονται τα πακέτα που θα σταλούν στον προορισμό. Δίνεται σε 32 δεκαεξαδικούς χαρακτήρες, χωρίς τον διαχωρισμό **:** ανά 2 byte.
4. Το μήκος του προθέματος του δικτύου προελεύσεως.
5. Ο άμεσος επόμενος κόμβος στον οποίο θα δρομολογηθούν τα πακέτα (next hop), σε δεκαεξαδική μορφή χωρίς τους χαρακτήρες διαχωρισμού **:** . (Σε περίπτωση που ο κόμβος προορισμού δεν είναι εξωτερικός, το πεδίο αυτό έχει την τιμή 00000000000000000000000000000000).
6. Το metric του συγκεκριμένου route σε δεκαεξαδική μορφή για την περίπτωση πολλαπλών routes με το ίδιο πρόθεμα.
7. Μετρητής της χρήσης του συγκεκριμένου route.
8. Flags
9. Το όνομα του interface (physical ή logical) που θα προωθήσει τα πακέτα στον επόμενο router ή κατευθείαν στον προορισμό τους.

6.2 Υλοποίηση των αντικειμένων της MIB

Για τα δυο πρώτα αντικείμενα, τα οποία δεν ανήκουν σε κάποιο πίνακα (scalar αντικείμενα) έγινε η χρήση του config αρχείου mib2c.scalar.conf που περάστηκε σαν παράμετρος –c του mib2c. Και στις δυο περιπτώσεις δημιουργούνται δυο αρχεία, το *όνομα_αντικειμένου**.c*** και το *όνομα_αντικειμένου**.h***. Οι όποιες αλλαγές και προσθήκες γίνονται στο *.c* αρχείο.

Στο πρώτο αντικείμενο, sixtofourTunnelMode, δεν χρειάζεται η ανάγνωση από κάποιο αρχείο και απλώς με χρήση της εντολής *ip tunnel show mode sit* το σύστημα

μας επιστρέφει αν υπάρχει εγκαταστημένη η IPv6-IPv4 tunnel device (sit στα LINUX), αναγκαία προϋπόθεση για την δημιουργία και λειτουργία του μηχανισμού. Στην περίπτωση που υπάρχει η sit device επιστρέφουμε την τιμή 1, και το πρόγραμμα από μόνο του κάνει την αντιστοίχηση με την τιμή sit(1) που επιστρέφει σε περίπτωση ανάλογης SNMP αίτησης. Τα αντίστοιχα συμβαίνουν και στην περίπτωση που δεν υπάρχει εγκατεστημένη η sit device στο σύστημα, οπόταν και το πρόγραμμα μας επιστρέφει την τιμή 2.

Στο δεύτερο αντικείμενο, τον αριθμό των 6to4 interfaces που υπάρχουν στο σύστημα, τον παίρνουμε από το αρχείο */proc/net/if_inet6*. Συγκεκριμένα, αφού ανοίξουμε το αρχείο, ελέγχουμε γραμμή-γραμμή τις εγγραφές, και μετρούμε τα interfaces με διευθύνσεις που περιέχουν το 6to4 πρόθεμα 2002::/16.

Στην συνέχεια, μετά τα scalar αντικείμενα, υλοποιούνται οι δυο πίνακες της MIB. Σε αυτή την περίπτωση γίνεται χρήση του config αρχείου *mib2c.mfd.conf* σαν παράμετρος –c κατά την κλήση της *mib2c* εντολής, και ο κώδικας που παράγεται έχει καθορισμένες τις κυριότερες δομές που χρειάζονται καθώς και τις συναρτήσεις που πρέπει να υλοποιηθούν και τις οποίες καλεί ο agent. Μεγάλο πλεονέκτημα επίσης που προσφέρει το *mfd* config αρχείο, είναι ότι ο κώδικας για τα διάφορα σταδία της εκτέλεσης μιας αίτησης, παράγεται σε ξεχωριστά, μικρά τμήματα κώδικα, καθιστώντας έτσι το όλο πρόγραμμα πιο λειτουργικό και εύχρηστο για τον προγραμματιστή. (Περισσότερα για το mib2c στο Παραρτημα Α).

Αρχικά υλοποιείται ο πρώτος πίνακας *sixtofourIfTable* ο οποίος δίνει τις διάφορες πληροφορίες για τα 6to4 interfaces που έχουν γίνει configure στο σύστημα. Το πρόγραμμα ανοίγει αρχικά το */proc/net/if_inet6* αρχείο, για να διαπιστώσει αν υπάρχουν στο σύστημα κάποια 6to4 interfaces. Αν όχι, δεν γίνεται καμία περαιτέρω ενέργεια και δεν επιστρέφεται καμία τιμή. Στην περίπτωση που υπάρχουν κάποια 6to4 interfaces, το πρόγραμμα παίρνει μια-μια τις εγγραφές του *if_inet6* αρχείου που αναφέρονται σε 6to4 interfaces και αντιστοιχεί την κάθε μια με ένα μοναδικό δείκτη, τον *sixtofourIfIndex*. Για την διαχείριση των δεικτών χρησιμοποιείται η δομή του SNMP, *user context*, που είναι ουσιαστικά μια συνδεδεμένη λίστα, έτσι ώστε να γίνεται εύκολα η προσθαφαίρεση εγγραφών από τον πίνακα και η ανακατανομή των τιμών των δεικτών. Ακολούθως, υπολογίζονται ένα-ένα τα αντικείμενα του πίνακα που αντιστοιχούν στον κάθε δείκτη-έγγραφη, και μέσω της δομής *sixtofourIfTable_rowreq_ctx* γίνεται το πέρασμα τους στις συναρτήσεις get, οι οποίες θα επιστρέψουν τις τιμές που ζητήθηκαν από την εκάστοτε SNMP αίτηση. Επίσης, δηλώνοντας ένα εξωτερικό δυσδιάστατο πίνακα **sixtofourinterfaceindex[]*, όπου η κάθε γραμμή του είναι το όνομα ενός από τα υπάρχοντα στο σύστημα 6to4 interfaces, και ο αριθμός της γραμμής, ο δείκτης προς το συγκεκριμένο interface, περνάμε στον δεύτερο πίνακα (*sixtofourRelayTable*) τις τιμές έτσι ώστε να γίνει η αντιστοίχηση του κάθε route (δρομολογίου) προς τον relay router με interface του πρώτου πίνακα στο οποίο έχει γίνει configure. Πιο κάτω περιγράφεται ο τρόπος άντλησης των πληροφοριών που το κάθε αντικείμενο πρέπει να επιστρέψει στον agent.

Μετά τον *sixtofourIfIndex*, υπολογίζεται το *sixtofourIfStatus* για το κάθε interface μέσα από το ***if_inet6*** αρχείο, αφού μόνο τα 6to4 interfaces που είναι σηκωμένα έχουν έγγραφη στο πεδίο της IPv6 (6to4) διευθύνσεως (στα υπόλοιπα που είναι ‘κάτω’ υπάρχει μόνο η IPv4 διεύθυνση). Στην περίπτωση που το interface είναι σηκωμένο, δίνεται στην ορισμένη για το αντικείμενο integer μεταβλητή η τιμή 1 και ακολούθως

καλείται η συνάρτηση που την αντιστοιχεί (mapping function) με την τιμή up(1), για να μεταβιβαστεί τέλος στην αντίστοιχη για το αντικείμενο συνάρτηση get για να δημοσιευθούν τα δεδομένα.

Επόμενο αντικείμενο είναι το *sixtofourIfMIBIndex*, που επιστρέφει την τιμή του δείκτη *ifIndex* από το ifTable(MIB II) που αντιστοιχεί στο συγκεκριμένο interface. Οι πληροφορίες για αυτό το αντικείμενο βρίσκονται στο directory ***/proc/net/dev_snmp6/*** στο αρχείο με το όνομα του κάθε interface. Πιο κάτω φαίνεται ένα μέρος του αρχείου, όπου για την περίπτωση της υλοποίησης της συγκεκριμένης MIB χρειάζεται μόνο η πρώτη γραμμή, στην οποία δίνεται η τιμή του ifIndex για το αντίστοιχο 6to4 interface μέσα στο ifTable, όπου βρίσκονται όλα τα interfaces του συστήματος.

```
# cat /proc/net/dev_snmp6/tun6to4

ifIndex                                 6
Icmp6InMsgs                             53
Icmp6InErrors                           0
Icmp6InDestUnreachs                     10
Icmp6InPktTooBigs                       0
Icmp6InTimeExcds                        29
Icmp6InParmProblems                     0
Icmp6InEchos                            0
Icmp6InEchoReplies                      14
```

Ακολούθως, βρίσκεται η IPv4 διεύθυνση στην οποία έγινε configure το interface. Και σε αυτήν την περίπτωση, τα δεδομένα προέρχονται από το if_inet6 αρχείο. Συγκεκριμένα, από την IPv6 (6to4) διεύθυνση του interface, αφαιρούνται τα πρώτα 2 bytes με το 2002:: πρόθεμα και από τα υπόλοιπα 4 bytes (8 δεκαεξαδικά ψηφία), με χρήση της συνάρτησης *hextoint()* η οποία μετατρέπει το δεκαεξαδικό αριθμό σε δεκαδικό, παίρνουμε την IPv4 διεύθυνση, στην κανονική δεκαδική της μορφή. Επειδή όμως οι IPv4 διευθύνσεις, ορίζονται στην MIB σαν ASN τύπου *IpAddress,* πράγμα που απαιτεί να δίνεται η διεύθυνση σε network byte order μορφή και να επιστρέφει ο agent την κανονική δεκαδική, γίνεται χρήση της συνάρτησης ***in_addr_t inet_addr(const char *cp)***. Η *inet_addr()* μετατρέπει την IPv4 διεύθυνση από την κανονική της μορφή (με δεκαδικούς αριθμούς και τέλειες) σε network byte order, για να επιστραφεί στον agent.
Εναλλακτικά, για τον υπολογισμό της IPv4 διευθύνσεως του 6to4 interface θα μπορούσε να χρησιμοποιηθεί η συνάρτηση
static int get_addr(int sock, char * ifname, struct sockaddr * ifaddr,char *ipv4)
η οποία ορίζεται στο *functions.h* και χρησιμοποιείται σε επόμενο αντικείμενο. Η συνάρτηση αυτή, παίρνοντας τα δεδομένα από κάποιο socket με κλήση *ioctl(),* επιστρέφει την IPv4 διεύθυνση του interface που του ορίζουμε.

Μετά την IPv4, υπολογίζεται η 6to4 διεύθυνση του interface (αντικείμενο *sixtofourIfLocal6to4Address*), που δίνεται σε δεκαεξαδική μορφή χωρίς τα διαχωριστικά **:** στο αρχειο ***if_inet6***. Το συγκεκριμένο αντικείμενο στην MIB ορίζεται σαν *Ipv6InetAddress*, που και σε αυτή την περίπτωση τα δεδομένα πρέπει να δοθούν στον agent σε network byte order, 16 bytes αυτή την φορά. Για αυτό στα αντικείμενα που επιστρέφουν IPv6 διεύθυνση χρησιμοποιείται η ***int inet_pton(int af, const char *src, void *dst)*** η οποία μετατρέπει το char string src, που περιέχει την IPv6 διεύθυνση στην κανονική της μορφή (με τους δεκαεξαδικούς αριθμούς και τις άνω-και-κάτω τελείες) σε network byte order της αντίστοιχης οικογένειας διευθύνσεων (AF_INET6 στην περίπτωση μας όπου πρόκειται για IPv6 διευθύνσεις) και την τοποθετεί στην δομή dest. Συγκεκριμένα, η διεύθυνση μετατρέπεται σε δομή τύπου *struct in6_addr* και ακολούθως αντιγράφεται στην dest, η οποία πρέπει να έχει οριστεί σε μέγεθος *sizeof(struct in6_addr)*.

Η τιμή για το επόμενο αντικείμενο, *sixtofourIfLocal6to4AddrPfxLength*, δίνεται στο 3^0 πεδίο της κάθε γραμμής, η οποία αντιστοιχεί σε ένα interface, στο αρχείο ***if_inet6***. Επειδή και πάλι δίνεται σε δεκαεξαδική μορφή, γίνεται και εδώ χρήση της συνάρτησης *hextoint()*, έτσι ώστε να πάρουμε το δεκαδικό μήκος του προθέματος, το οποίο θα πάρει και θα εμφανίσει ο agent.

Τα δυο επόμενα αντικείμενα αφορούν παραμέτρους δρομολόγησης των IPv4 πακέτων όπου περιέχουν τα ενθυλακωμένα IPv6 πακέτα.. Το πρώτο αντικείμενο, το *sixtofourIfHopLimit*, δίνεται μέσα από την εντολή του συστήματος ***ip tunnel show***, όπου επιστρέφονται όλα τα tunnel interfaces που χρησιμοποιεί το σύστημα με τις παραμέτρους του καθενός.
Συγκεκριμένα, επιστρέφει για παράδειγμα για ένα 6to4 interface (π.χ tun6to4)

```
# ip tunnel show

tun6to4: ipv6/ip  remote any  local 147.102.228.34  dev eth0  ttl 66
```

από όπου παίρνουμε την τιμή του ttl για να την επιστρέψουμε στον agent.

Το επόμενο αντικείμενο είναι το *sixtofourIfMTU*, η τιμή του οποίου βρίσκεται στο αρχείο ***/proc/sys/net/ipv6/conf/<ονομα interface>*** . Στο αρχείο αυτό υπάρχει μόνο η τιμή του MTU στην δεκαδική της μορφή, η οποία αντιγράφεται σε μια integer μεταβλητή και μέσω της συνάρτησης get μεταβιβάζεται στον agent.

Τελευταίο αντικείμενο του πίνακα των interfaces είναι το *sixtofourIfIpSec*, που φανερώνει κατά πόσο χρησιμοποιείται το IPv4 IPsec πρωτόκολλο ασφάλειας κατά την μετάδοση των IPv4 πακέτων, μέσα στα οποία βρίσκονται ενθυλακωμένα τα IPv6 πακέτα. Γενικά, όταν στο σύστημα έχει εγκατασταθεί το IPSec πρωτόκολλο, τα πιο κάτω read-only αρχεία δημιουργούνται στο directory ***/proc/net/*** :
/proc/net/ipsec_eroute
/proc/net/ipsec_klipsdebug
/proc/net/ipsec_spi
/proc/net/ipsec_spigrp
/proc/net/ipsec_spinew

/proc/net/ipsec_tncfg
/proc/net/ipsec_version

Για την υλοποίηση της 6to4 MIB, χρησιμοποιείται μόνο το ***ipsec_tncfg*** αρχείο, στο οποίο φαίνονται ποια ‘εικονικά΄(virtual) IPSec interfaces βρίσκονται πάνω από ‘φυσικά΄ interfaces (π.χ Ethernet), μέσω των οποίων τα πακέτα θα προωθηθούν, αφού γίνουν οι κατάλληλες ενέργειες από το IPSec.
Πιο κάτω φαίνεται η μορφή ενός ipsec_tncfg αρχείου :

```
# cat /proc/net/ipsec_tncfg

ipsec0 -> eth0 mtu=16260 -> 1500
ipsec1 -> NULL mtu=0 -> 0
ipsec2 -> NULL mtu=0 -> 0
ipsec3 -> NULL mtu=0 -> 0
```

όπου :

το πρώτο πεδίο είναι το όνομα του virtual interface, το δεύτερο είναι το όνομα του physical interface που βρίσκεται κάτω από το ‘εικονικό’ ipsec στην περίπτωση που υπάρχει ή NULL αν δεν υπάρχει. Μετά, δίνεται το MTU του ipsec virtual interface και τέλος το MTU του φυσικού interface.
Έτσι, αν θέλουμε να ελέγξουμε την εφαρμογή IPsec στο 6to4 interface, πρώτα ελέγχουμε για την ύπαρξη των πιο πάνω αρχείων, και συγκεκριμένα ελέγχεται μόνο η ύπαρξη του ***ipsec_tncfg***. Αν το αρχείο δεν υπάρχει επιστρέφεται η τιμή 2, όπου θα κληθεί ακολούθως η ανάλογη συνάρτησης αντιστοίχησης, και τελικά η get επιστρέφει την τιμή no(2), υποδηλώνοντας ότι δεν υπάρχει εγκατεστημένο το IPsec στο σύστημα. Στην περίπτωση που υπάρχει, ανοίγεται το αρχείο, και μια-μια γραμμή ελέγχονται τα ipsec virtual interfaces, αν τα φυσικά interfaces στα οποία είναι συνδεδεμένα, είναι τα ίδια μέσω των οποίων μεταδίδονται τα 6to4 πακέτα. Αν γίνει η ταύτιση, επιστρέφεται η τιμή 1, που υποδηλώνει την χρήση IPsec πρωτοκόλλου ασφαλείας από το συγκεκριμένο 6to4 interface.

Ακολουθεί ο υπολογισμός των στοιχείων του δεύτερου πίνακα *sixtofourRelayTable*. Αρχικά γίνεται ο έλεγχος αν υπάρχουν στο σύστημα 6to4 interfaces μέσα από το */proc/net/if_inet6* αρχείο, και αν όχι δεν γίνεται καμία περαιτέρω ενέργεια και δεν επιστρέφεται στον agent καμία τιμή. Αν διαπιστωθεί η ύπαρξη 6to4, ανοίγεται το αρχείο ***/proc/net/ipv6_route*** όπου μέσα βρίσκονται όλα τα IPv6 routes του συστήματος (στην ουσία πρόκειται για τον πίνακα δρομολόγησης του μηχανήματος). Στο σημείο αυτό πρέπει να γίνουν οι κατάλληλοι ελέγχοι έτσι ώστε να ληφθούν μόνο τα routes που έχουν γίνει configure για το 6to4 interface και επιπλέον να είναι δρομολόγια προς native IPv6 προορισμούς, διαμέσου κάποιου Relay Router. Επιπλέον, πρέπει να ελέγχονται και να απορρίπτονται διευθύνσεις με τα πιο κάτω προθέματα:

- *2002::/16* 6to4 κόμβος.

- *fe80::/16* link-local διεύθυνση
- *fec0::/16* site-local διεύθυνση
- *ff00::/16* any multicast
- *::/16* loopback διευθύνσεις, compatible κ.λ.π

Αφού έχουν παρθεί τα διάφορα routes προς τους Relay Routers, γίνεται η αντιστοίχηση τους με τον δείκτη του πίνακα, τον *sixtofourRelayIndex*, και ακολούθως μέσω της εξωτερικής μεταβλητής ***sixtofourinterfaceindex*** γίνεται η αντιστοιχήσει του route με το 6to4 interface στο οποίο έχει ανατεθεί, και συγκεκριμένα δίνεται η τιμή του αντίστοιχου *sixtofourIfIndex* στον δείκτη *sixtofourRelayIfIndex* για να επιστραφεί από τον agent.

Ακολούθως, υπολογίζονται τα υπόλοιπα αντικείμενα του πίνακα ξεκινώντας από το *sixtofourRelayDestAddr*, που είναι η IPv6 διεύθυνση του δικτύου προορισμού. Η τιμή αυτή βρίσκεται σε δεκαεξαδική μορφή στο πρώτο πεδίο κάθε γραμμής του αρχείου ***ipv6_route***, σε μορφή string, χωρίς το διαχωρισμό **:** . Αφού πάρουμε την τιμή την μετατρέπουμε στην κανονική μορφή με την βοήθεια των *sscanf()* και *fprintf()*, έτσι ώστε να την περάσουμε στην συνάρτηση ***inet_pton(),*** αφού είναι ορισμένη σαν InetAddressIPv6 και έτσι στον agent πρέπει να δοθεί η τιμή της σε network byte order.

Αμέσως μετά υπολογίζεται το πρόθεμα του δικτύου προορισμού, η μάσκα δηλαδή η οποία θα εφαρμοστεί στην πιο πάνω διεύθυνση για να προκύψει το δίκτυο προορισμού, το οποίο γράφεται στο πεδίο destination της εξωτερικής IPv4 επικεφαλίδας, και το οποίο είναι ο τελικός προορισμός των πακέτων. Η τιμή του μήκους του προθέματος δίνεται στο δεύτερο string κάθε γραμμής του αρχείου ipv6_route, επίσης σε δεκαεξαδική μορφή. Χρησιμοποιώντας την *hextoint()* συνάρτηση μετατρέπεται σε δεκαδική μορφή, και ακολούθως ορίζουμε το πρόθεμα σε μορφή IPv6 διευθύνσεως, με τους πρώτους ίσους με τον αριθμό του προθέματος χαρακτήρες (που αντιστοιχεί ο καθένας σε ένα bit) να παίρνουν την τιμή 1, και τα υπόλοιπα 0. Και πάλι χρησιμοποιείται η ***inet_pton()*** συνάρτηση έτσι ώστε να μετατραπεί η διεύθυνση σε network byte order και επιστρέφεται η τιμή στον agent.

Το επόμενο αντικείμενο είναι η διεύθυνση του Relay Router στον οποίο θα σταλούν τα πακέτα, για να γίνει η απενθυλάκωση τους και κατόπιν να αποσταλούν στους native IPv6 προορισμούς τους. Η διεύθυνση αυτή βρίσκεται στο 5^0 πεδίο της κάθε εγγραφής του αρχείου ***ipv6_route***, όπου αναγράφεται η next hop διεύθυνση του κάθε route. Ακολουθείται και σε αυτή την περίπτωση η διαδικασία που ακολουθήθηκε στις δυο πιο πάνω περιπτώσεις για την επιστροφή της τιμής στον agent.

Το αν η πιο πάνω διεύθυνση του Relay Router είναι η anycast διεύθυνση προς τον πλησιέστερο relay router καθορίζεται από το επόμενο αντικείμενο, όπου γίνεται η σύγκριση της πιο πάνω διεύθυνσης με την anycast **2002:c058:6301::**, και σε περίπτωση ταύτισης, επιστρέφεται η τιμή 1 που αντιστοιχίζεται με το anycastAddress(1) που θα επιστρέψει τελικά ο agent. Αν δεν υπάρξει ταύτιση, επιστρέφεται η τιμή 2.

Τέλος, υπολογίζεται το metric που αντιστοιχεί στο κάθε route, πάλι από το αρχείο ***ipv6_route***, όπου δίνεται στο 6^0 πεδίο της κάθε έγγραφης σε δεκαεξαδική μορφή. Για τον υπολογισμό του χρησιμοποιήθηκε μεταβλητή τύπου long, επειδή το

συγκεκριμένο αντικείμενο μπορεί να πάρει τιμή μέχρι τον δεκαεξαδικό αριθμό ffffffff. Αφού μετατραπεί με την βοήθεια της συνάρτησης *hextoint()* σε δεκαδικό, επιστρέφεται κανονικά στον agent.

Γενικά, όταν γίνεται κάποια SNMP αίτηση ακολουθείται η ίδια διαδικασία. Αν η αίτηση ζητά πληροφορίες για κάποιο από τα scalar αντικείμεναα της MIB, ο agent καλεί τη συνάρτηση του mib module που ζητείται στην συγκεκριμένη αίτηση και αντλεί τα δεδομένα όπως έχει περιγράφει πιο πάνω. Στην περίπτωση που γίνεται αίτηση για κάποιο/ κάποια αντικείμενα που ανήκουν σε κάποιον από τους πίνακες, φορτώνεται η συνάρτηση cache_load, και αφού ανοιχτεί το κυρίως αρχείο του κάθε πίνακα (το */proc/net/if_inet6* στην περίπτωση του sixtofourIfTable ή το */proc/net/ipv6_route* στην περίπτωση του sixtofourRelayTable) , κάθε έγκυρη εγγραφή του οποίου αντιστοιχείται σε ένα δείκτη του πίνακα, αντλούνται τα δεδομένα, όπως έχει περιγράφει πιο πάνω, για την κάθε έγγραφη (την καθε γραμμή δηλαδή του πίνακα) και τους γίνεται η ανάθεση του αντιστοίχου δείκτη. Συμπληρώνεται η δομή *row-context* για την κάθε γραμμή, και ακολούθως, αφού προσπελαστούν όλες οι έγκυρες γραμμές του αρχείου και συμπληρωθούν όλοι οι row contexts, όλα τα δεδομένα τοποθετούνται στο container. Από το σημείο αυτό και μετά τελειώνουν οι διάφορες ενέργειες που έχει καθορίσει ο προγραμματιστής, αφού ο agent πλέον βρίσκει μέσα στο container το αντικείμενο που έχει ζητηθεί και επιστρέφει τις κατάλληλες τιμές. Ο container διατηρεί ένα είδος cache με τις τιμές των αντικειμένων για κάποιο χρονικό διάστημα (τίθεται από τον διαχειριστή μέσω της σταθεράς <*ΟΝΟΜΑ ΠΙΝΑΚΑ*>*_CACHE_TIMEOUT*), και απαντά σε όσες αιτήσεις γίνουν σε αυτό το διάστημα με τις τιμές των αντικειμένων που διατηρούνται στη cache. Όταν παρέλθει αυτός ο χρόνος, επαναλαμβάνεται η πιο πάνω διαδικασία στην περίπτωση νέας αίτησης.

Ο πλήρης κώδικας του προγράμματος δίνεται στο παράρτημα Γ, και η διαδικασία υλοποίησης μιας MIB με NET-SNMP, η οποία αποτελείται από scalar αντικείμενα και πίνακες, δίνεται στο παράρτημα Α.

6.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ AGENT

Ακολούθως δίνονται αποτελέσματα εκτέλεσης εντολών *snmpwalk* στον agent που υλοποιήθηκε για την SIXTOFOUR-MIB.

- Πριν γίνει configure κανένα 6to4 interface στο σύστημα:

```
# snmpwalk -c public localhost sixtofourMIB

SIXTOFOUR-MIB::sixtofourTunnelMode.0 = INTEGER: sit(1)
SIXTOFOUR-MIB::sixtofourIfNumber.0 = INTEGER: 0
```

Σχήμα 28: Αποτελέσματα *snmpwalk* (1/4)

- Γίνεται configure ένα 6to4 interface στην διεύθυνση *147.102.228.34* και καθορίζονται δυο routes προς δυο relay routers

```
# snmpwalk -c public localhost sixtofourMIB

SIXTOFOUR-MIB::sixtofourTunnelMode.0 = INTEGER: sit(1)
SIXTOFOUR-MIB::sixtofourIfNumber.0 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourIfStatus.1 = INTEGER: up(1)
SIXTOFOUR-MIB::sixtofourIfMIBIndex.1 = INTEGER: 5
SIXTOFOUR-MIB::sixtofourIfLocalIpv4Address.1 = IpAddress:
147.102.228.34
SIXTOFOUR-MIB::sixtofourIfLocal6to4Address.1 = STRING:
2002:9366:e422:0:0:0:0:1
SIXTOFOUR-MIB::sixtofourIfLocal6to4AddrPfxLength.1 = INTEGER: 48
SIXTOFOUR-MIB::sixtofourIfHopLimit.1 = INTEGER: 66
SIXTOFOUR-MIB::sixtofourIfMTU.1 = INTEGER: 1480
SIXTOFOUR-MIB::sixtofourIfIpSec.1 = INTEGER: no(2)
SIXTOFOUR-MIB::sixtofourRelayIfIndex.1 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayIfIndex.2 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayDestAddr.1 = STRING: 3ffe:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddr.2 = STRING: 0:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.1 = STRING:
1111:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.2 = STRING:
0:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.1 = STRING:
2002:c25f:6c5b:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.2 = STRING:
2002:c058:6301:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.1 = INTEGER:
configuredAddr(2)
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.2 = INTEGER:
anycastAddr(1)
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.1 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.2 = INTEGER: 1
```

Σχήμα 29: Αποτελέσματα *snmpwalk* (2/4)

- Προστίθεται άλλο ένα route, έτσι ώστε πακέτα προς το grnet (2001:648:2ffc::/48) να προωθούνται μέσω του relay router με διεύθυνση 128.107.240.254 (2002:806b:f0fe::)

```
# snmpwalk -c public localhost sixtofourMIB

SIXTOFOUR-MIB::sixtofourTunnelMode.0 = INTEGER: sit(1)
SIXTOFOUR-MIB::sixtofourIfNumber.0 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourIfStatus.1 = INTEGER: up(1)
SIXTOFOUR-MIB::sixtofourIfMIBIndex.1 = INTEGER: 5
```

```
SIXTOFOUR-MIB::sixtofourIfLocalIpv4Address.1 = IpAddress:
147.102.228.34
SIXTOFOUR-MIB::sixtofourIfLocal6to4Address.1 = STRING:
2002:9366:e422:0:0:0:0:1
SIXTOFOUR-MIB::sixtofourIfLocal6to4AddrPfxLength.1 = INTEGER: 48
SIXTOFOUR-MIB::sixtofourIfHopLimit.1 = INTEGER: 66
SIXTOFOUR-MIB::sixtofourIfMTU.1 = INTEGER: 1480
SIXTOFOUR-MIB::sixtofourIfIpSec.1 = INTEGER: no(2)
SIXTOFOUR-MIB::sixtofourRelayIfIndex.1 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayIfIndex.2 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayIfIndex.3 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayDestAddr.1 = STRING:
2001:648:2ffc:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddr.2 = STRING: 3ffe:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddr.3 = STRING: 0:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.1 = STRING:
1111:1111:1111:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.2 = STRING:
1111:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.3 = STRING:
0:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.1 = STRING:
2002:806b:f0fe:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.2 = STRING:
2002:c25f:6c5b:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.3 = STRING:
2002:c058:6301:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.1 = INTEGER:
configuredAddr(2)
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.2 = INTEGER:
configuredAddr(2)
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.3 = INTEGER:
anycastAddr(1)
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.1 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.2 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.3 = INTEGER: 1
```

Σχήμα 30: Αποτελέσματα *snmpwalk* (3/4)

- Αφαίρεση του τελευταίου route προς τον relay router που είχε προστεθεί και δημιουργία ακόμη ενός 6to4 interface στην διεύθυνση *147.102.228.7:*

```
# snmpwalk -c public localhost sixtofourMIB

SIXTOFOUR-MIB::sixtofourTunnelMode.0 = INTEGER: sit(1)
SIXTOFOUR-MIB::sixtofourIfNumber.0 = INTEGER: 2
SIXTOFOUR-MIB::sixtofourIfStatus.1 = INTEGER: up(1)
SIXTOFOUR-MIB::sixtofourIfStatus.2 = INTEGER: up(1)
SIXTOFOUR-MIB::sixtofourIfMIBIndex.1 = INTEGER: 7
SIXTOFOUR-MIB::sixtofourIfMIBIndex.2 = INTEGER: 6
SIXTOFOUR-MIB::sixtofourIfLocalIpv4Address.1 = IpAddress:
147.102.228.7
SIXTOFOUR-MIB::sixtofourIfLocalIpv4Address.2 = IpAddress:
```

```
147.102.228.34
SIXTOFOUR-MIB::sixtofourIfLocal6to4Address.1 = STRING:
2002:9366:e407:0:0:0:0:1
SIXTOFOUR-MIB::sixtofourIfLocal6to4Address.2 = STRING:
2002:9366:e422:0:0:0:0:1
SIXTOFOUR-MIB::sixtofourIfLocal6to4AddrPfxLength.1 = INTEGER: 48
SIXTOFOUR-MIB::sixtofourIfLocal6to4AddrPfxLength.2 = INTEGER: 48
SIXTOFOUR-MIB::sixtofourIfHopLimit.1 = INTEGER: 72
SIXTOFOUR-MIB::sixtofourIfHopLimit.2 = INTEGER: 66
SIXTOFOUR-MIB::sixtofourIfMTU.1 = INTEGER: 1480
SIXTOFOUR-MIB::sixtofourIfMTU.2 = INTEGER: 1480
SIXTOFOUR-MIB::sixtofourIfIpSec.1 = INTEGER: no(2)
SIXTOFOUR-MIB::sixtofourIfIpSec.2 = INTEGER: no(2)
SIXTOFOUR-MIB::sixtofourRelayIfIndex.1 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayIfIndex.2 = INTEGER: 2
SIXTOFOUR-MIB::sixtofourRelayIfIndex.3 = INTEGER: 2
SIXTOFOUR-MIB::sixtofourRelayDestAddr.1 = STRING: 2001:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddr.2 = STRING: 3ffe:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddr.3 = STRING: 0:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.1 = STRING:
1111:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.2 = STRING:
1111:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayDestAddrPrefix.3 = STRING:
0:0:0:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.1 = STRING:
2002:8c6d:106:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.2 = STRING:
2002:c25f:6c5b:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAddress.3 = STRING:
2002:c058:6301:0:0:0:0:0
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.1 = INTEGER:
configuredAddr(2)
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.2 = INTEGER:
configuredAddr(2)
SIXTOFOUR-MIB::sixtofourRelayAnycastStatus.3 = INTEGER:
anycastAddr(1)
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.1 = INTEGER: 100
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.2 = INTEGER: 1
SIXTOFOUR-MIB::sixtofourRelayRouteMetric.3 = INTEGER: 1
```

Σχήμα 31: Αποτελέσματα *snmpwalk* (4/4)

ΠΑΡΑΡΤΗΜΑ Α

NET-SNMP 5 TUTORIAL

Εγκατάσταση:

Χρησιμοποιώντας την τελευταία έκδοση του Net-SNMP 5.2.1 (net-snmp-5.2.1.tar.gz) εκτελούμε τα πιο κάτω βήματα για την εγκατάσταση του προγράμματος σε περιβάλλον Linux:

- *./configure --with-perl-modules*

Έτσι ώστε να γίνει και εγκατάσταση των Perl modules δίνοντας μας τη δυνατότητα να κάνουμε χρήση του εργαλείου ***mib2c*** για παραγωγή κώδικα και του ***snmpconf*** για δημιουργία configuration file για τον snmpd δαίμονα**.**

- *make*

- Εκτελούμε την επόμενη εντολή σαν root

  make install

Αν με το ./configure δεν είχαμε δηλώσει `--with-prefix=$PATH` όπου PATH το directory που επιθυμούμε να γίνει η εγκατάσταση, τότε τα διάφορα αρχεία του Net-SNMP θα έχουν εγκατασταθεί στο /usr/local/.

- Κάνουμε configure τον agent. Συγκεκριμένα με χρήση του ***snmpconf*** δημιουργούμε το configuration file *snmpd.conf* για τον snmpd δαίμονα του SNMP.

Η διαδικασία έχει ως εξής:

- Βρίσκουμε που πρέπει να γίνει η εγκατάσταση του configuration file χρησιμοποιώντας το debug output του agent:

  $ snmpd -Dread_config -H 2>&1 | grep "config path" | sort -u

  Και στην συνεχεία ελέγχουμε αν υπάρχουν ήδη κάποια configuration files:

  $ snmpd -Dread_config -H 2>&1 | grep "Reading" | sort -u

- Χρησιμοποιώντας το perl script *snmpconf* και δίνοντας

  snmpconf -r none -g basic_setup

  θέτουμε τις βασικές επιλογές που παρέχει το snmpd.

Τελικά , αφού έχουμε δώσει τα στοιχεία που είχαν ζητηθεί δημιουργείται το πιο κάτω configuration αρχείο όπου το εγκαθιστούμε στη θέση που είχαμε βρει πιο πάνω, και συγκεκριμένα για την περίπτωση μας στο `/usr/local/share/`.

```
##############################################################################
######
#
# snmpd.conf
#
#   - created by the snmpconf configuration program
#
##############################################################################
######
# SECTION: Access Control Setup
#
#   This section defines who is allowed to talk to your running
#   snmp agent.

# rwuser: a SNMPv3 read-write user
#   arguments:  user [noauth|auth|priv] [restriction_oid]

rwuser  demetris auth

# rouser: a SNMPv3 read-only user
#   arguments:  user [noauth|auth|priv] [restriction_oid]

rouser  stelios auth .1.3.6.1.4.1.8072

# rocommunity: a SNMPv1/SNMPv2c read-only access community name
#   arguments:  community [default|hostname|network/bits] [oid]

rocommunity  public

# rwcommunity: a SNMPv1/SNMPv2c read-write access community name
#   arguments:  community [default|hostname|network/bits] [oid]

rwcommunity  private 127.0.0.1




##############################################################################
######
# SECTION: System Information Setup
#
#   This section defines some of the information reported in
#   the "system" mib group in the mibII tree.

# syslocation: The [typically physical] location of the system.
#   Note that setting this value here means that when trying to
#   perform an snmp SET operation to the sysLocation.0 variable will
make
#   the agent return the "notWritable" error code.  IE, including
#   this token in the snmpd.conf file will disable write access to
#   the variable.
#   arguments:  location_string

syslocation  laptop-anywhere
```

```
# syscontact: The contact information for the administrator
#   Note that setting this value here means that when trying to
#   perform an snmp SET operation to the sysContact.0 variable will
make
#   the agent return the "notWritable" error code.  IE, including
#   this token in the snmpd.conf file will disable write access to
#   the variable.
#   arguments:  contact_string

syscontact  "Demetris Philipides"
```

και τέλος θέτουμε τον agent σε λειτουργία καλώντας *snmpd*.

- **Παραγωγή του κυρίως σκελετού του Κώδικα- Χρήση mib2c**

Στην περίπτωση που θέλουμε να επεκτείνουμε τον ήδη υπάρχον agent ώστε να διαχειρίζεται και τα αντικείμενα κάποιας MIB την οποία δεν υλοποιεί πρέπει αρχικα να εγκαταστήσουμε την MIB μας στο directory /usr/local/share/mibs/ και να θέσουμε την MIBS environment variable έτσι ώστε να περιλαμβάνει το MIB file που θα χρησιμοποιήσουμε εκτελώντας:
MIBS=+MYMIB-MIB
ή
MIBS=ALL
και τέλος
export MIBS

Έχοντας γνωστοποιήσει την mib, μπορούμε κάνοντας χρήση του εργαλείου mib2c:

```
mib2c [-h] -c CONFIGFILE [-I PATH] [-f OUTNAME] [-
i][q][-S VAR=VAL] MIBNODE
```

να παράξουμε κώδικα για την υπάρχουσα mib μας, παρακάμπτοντας έτσι μπόλικο προγραμματιστικό φόρτο που έχει να κάνει καθαρά με το SNMP πρωτόκολλο.
Ανάλογα με τα αντικείμενα της mib μας, αν είναι μοναδικά scalar αντικείμενα ή πίνακες, και αναλόγως των τιμών που επιστρέφουν χρησιμοποιείται το κατάλληλο configfile που περνιέται στην παράμετρο –c του mib2c.
Η συγκεκριμένη αναφορά αναφέρεται τόσο σε μοναδικά scalar αντικείμενα, όσο και σε υλοποίηση πίνακα αντικειμένων, και σαν MIB αναφοράς θα χρησιμοποιείται η SIXTOFOUR-MIB (όπου θα χρησιμεύσει για γενίκευση κώδικα με το mib2c), με κάποια όμως υποθετικά αντικείμενα για χάρη γενικότητας.
Σαν πρώτο στάδιο για την συγκεκριμένη υλοποίηση έγινε χρήση των πιο κάτω εντολών:

1.mib2c -c mib2c.scalar.conf MIBNODE

όπου παράγεται ο σκελετός του κώδικα σε C για το αντικείμενο της mib MIBNODE με χρήση του config file **`mib2c.scalar.conf`** και δημιουργούνται τα αρχεία MIBNODE.c και MIBNODE.h που η μορφή τους φαίνεται πιο κάτω. Είναι καλό, αφού φορτώσουμε την MIB, και πριν καλέσουμε το mib2c να ελέγξουμε την MIB για τυχόν συντακτικά λάθη δίνοντας :

snmptranslate -IR MIBNODE

όπου αν η MIB δεν περιέχει συντακτικά λάθη εμφανίζεται στην οθόνη:

MYMIB :: MIBNODE

Πρέπει να σημειωθεί ότι ο agent υλοποιήθηκε μόνο για read-only αντικείμενα, έτσι ώστε να υλοποιεί μόνο τις εντολές snmpget και snmpgetnext. Στον κώδικα που ακολουθεί δεν γίνεται λόγος για αντικείμενα που μπορούν να τεθούν (settable – εντολή snmpset) μεσώ SNMP.

MIBNODE.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        : mib2c.scalar.conf,v 1.8 2004/10/14 12:57:34 dts12 Exp $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "MIBNODE.h"


/** Αρχικοποιεί το MIBNODE module */
void
init_MIBNODE(void)
{
    /* ορίζεται ο OBJECT IDENTIFIER ανάλογα με κάτω από ποιο κλάδο
     * του δέντρου των MIB έχει οριστεί η δική μας MIB και το oid που
     * εμείς ορίσαμε στο αντικείμενο στην MIB  */
    static oid MIBNODE_oid[] = { 1,3,6,1,χ,χ,χ,χ };

    netsnmp_register_scalar(
        netsnmp_create_handler_registration("MIBNODE",
handle_MIBNODE,
                                MIBNODE_oid, OID_LENGTH(MIBNODE_oid),
                                HANDLER_CAN_RONLY
        ));
}

int
handle_MIBNODE(netsnmp_mib_handler *handler,
                           netsnmp_handler_registration *reginfo,
                           netsnmp_agent_request_info   *reqinfo,
                           netsnmp_request_info         *requests)
```

```
{

    /* Μια καινοτομία του version 5 και έπειτα είναι ότι κάνοντας το
     * αντικείμενο register σαν "instance" δεν χρειάζεται να
     * υλοποιεί ειδική περίπτωση για κλήση της  GETNEXT αφού το
     * πρόγραμμα την υλοποιεί από μόνο του.   */

    /* Επιπλέον ο instance handler 'κρατά' μόνο μια αίτηση κάθε φορά
       έτσι ώστε να μην χρειάζεται να προσπελάσει μια λίστα από
       αιτήσεις   */


    switch(reqinfo->mode) {

        case MODE_GET:

   /* Δίνουμε τον κώδικα που θέλουμε να εκτελεστεί όταν ο agent πάρει
      snmpget και snmpgetnext εντολή. Συνήθως ανοίγει κάποιο αρχείο ή
      κάποιο socket για να πάρουμε τα δεδομένα που θα επιστρέψει ο
      agent    */

   /* Η snmp_set_var_typed_value είναι η συνάρτηση που επιστρέφει τα
      δεδομένα σε μια αίτηση. Όπου XXX ένας δείκτης στα δεδομένα που
      θέλουμε να επιστρέψει ο agent,στον οποίο έχουμε αναθέσει τιμή
      πιο πάνω. Όπου YYY το μέγεθος των δεδομένων που θα επιστρέψει ο
      agent.
      Το πεδίο type που στο συγκεκριμένο παράδειγμα έχει γενικευθεί
      σαν ASN_OCTET_STR, θα μπορούσε να ήταν οποιοσδήποτε άλλος ASN
      τύπος(ASN_INTEGER,ASN_BOOLEAN κ.ο.κ) ανάλογα με το πως έχει
      δηλωθεί το αντικείμενο στην   εκάστοτε MIB, από την οποία θα
      'παραχθεί' ο κώδικας για τα modules του agent.  */

            snmp_set_var_typed_value(requests->requestvb,
ASN_OCTET_STR,
                                     (u_char *) /* XXX: a pointer to
the scalar's data */,
                                     /* YYY: the length of the data
in bytes */);
            break;


        default:
            /* we should never get here, so this is a really bad
error */
            snmp_log(LOG_ERR, "unknown mode (%d) in
handle_MIBNODE\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
```

Στο πιο πάνω κομμάτι κώδικα χρησιμοποιούνται οι πιο κάτω συναρτήσεις όπου ορίζονται στα αρχεία:

- Στο **agent_handler.h**

***netsnmp_handler_registration** * **netsnmp_create_handler_registration**(const char *name,*

```
  Netsnmp_Node_Handler* handler_access_method,
  oid *reg_oid,
 size_t reg_oid_len,
  int modes);
```

οπου

name: το όνομα του handler και ανατίθεται στο netsnmp_mib_handler -> handler_name

handler_access_method: ένας δείκτης συνάρτησης που χρησιμοποιείται σαν η μέθοδος πρόσβασης για το συγκεκριμένο handler registration.

oid: ο object identifier για το συγκεκριμένο αντικείμενο όπως έχει δηλωθεί στην αρχή του κώδικα. Είναι αυτός που χρησιμοποιείται από τον agent για να εντοπίζει κάθε φορά το αντικείμενο στο οποίο αναφέρεται η αίτηση.

reg_oid_len: το μέγεθος του oid.

modes : Στην δική μας περίπτωση χρησιμοποιούμε το HANDLER_CAN_RONLY, δηλαδή ο handler απλώς διαβάζει δεδομένα. Το HANDLER_CAN_RONLY ισούται με το HANDLER_CAN_GET_AND_GETNEXT. Η μεταβλητή modes μπορεί επιπλέον να οριστεί σαν HANDLER_CAN_SET,HANDLER_CAN_GETBULK, HANDLER_CAN_NOT_CREATE και HANDLER_CAN_BABY_STEP.

Η πάνω συνάρτηση δημιουργεί αρχικά ένα instance_handler και ακολούθως τον κάνει register με τον συγκεκριμένο object identifier, ώστε να μπορεί να καλείται από τον agent όταν υπάρχει αντίστοιχη αίτηση.

- Στο **snmp_client.c**.

int snmp_set_var_typed_value (netsnmp_variable_list * *newvar*,
u_char *type*,
const u_char * *val_str*,
size_t *val_len*)

όπου:

newvar : η δομή στην οποία θα ανατεθούν οι τιμές των μεταβλητών type, val_str και val_len και η οποία γνωστοποιείται στον agent.

type : Ο ASN τύπος που θα επιστρέψει ο agent. Πρέπει να είναι ένας από τους πρωταρχικούς ASN τύπους (INTEGER, OCTET STRING, BOOLEAN κ.ο.κ).

val_str: Ένας buffer που περιέχει τα δεδομένα που θα αντιγράφουν στην newvar.

val_len : Το μέγεθος των δεδομένων που θα αντιγράφουν στην newvar.

Η συνάρτηση αυτή χρησιμεύει για να επιστρέφει τα δεδομένα σε μια αίτηση SNMP μεσώ του δείκτη requestvb της δομής net SNMP_request_info. Επιπλέον, η ίδια συνάρτηση θα χρησιμοποιόταν και στην περίπτωση που υλοποιούσαμε αντικείμενα που μπορούν να γίνουν set μεσώ SNMP. Συγκεκριμένα, καλώντας τη συνάρτηση, θέτει τα δεδομένα στην δομή net_snmp_variable_list.

MIBNODE.h

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        : mib2c.scalar.conf,v 1.8 2004/10/14 12:57:34 dts12 Exp $
 */
#ifndef MIBNODE_H
#define MIBNODE_H

/* function declarations */
void init_MIBNODE(void);
Netsnmp_Node_Handler handle_MIBNODE;

#endif /* MIBNODE_H */
```

Αφού γράφουν τα διάφορα modules για τα αντικείμενα της MIB, πρέπει να μεταφερθούν στο directory /directory εγκατάστασης/net-snmp 5.2.1/agent/mibgroup/ .
Ακολούθως, ξανακάνουμε configure συμπεριλαμβάνοντας και τα modules που έχουμε δημιουργήσει.
> ./configure --with-mib-modules="MIBNODE1 MIBNODE2 κ.τ.λ" --with-perl-modules

Εναλλακτικά, αν τα mib-modules είναι πολλά τα τοποθετούμε σε ένα sub-directory του /mibgroup/ για παράδειγμα /mibgroup/mymib/και γράφουμε στο /mibgroup/ ένα header file mymib.h όπου ορίζουμε μέσα ποια mib-modules θα γίνουν configure:

config_require(mymib/MIBNODE1)
config_require(mymib/MIBNODE2)
config_require(mymib/MIBNODE3)

> ./configure --with-mib-modules="mymib" --with-perl-modules

> make
> make install

Και τέλος ξεκινούμε τον snmpd δαίμονα δίνοντας :
> snmpd *[OPTIONS] [LISTENING ADDRESSES]*

Μόλις ο agent ξεκινήσει, καλείται από το κάθε mib-module η συνάρτηση ***init_MIBNODE()*** όπου δημιουργούνται και γίνονται register οι διάφοροι handlers. Όταν στην συνεχεία υπάρχει snmprequest τότε καλείται η ***handle_MIBNODE()*** αναλόγως του MIBNODE το οποίο δίνεται στην εκάστοτε εντολή. Για την περίπτωση μας, σε εντολές snmpget και snmpgetnext μπαίνει στην case MODE_GET και παίρνει τα δεδομένα που του έχουμε ορίσει.

2. mib2c –c mib2c.mfd.conf MIBTABLE

όπου MIBTABLE ένας πίνακας της MIB που έχουμε γράψει και γνωστοποιήσει όπως πιο πάνω. Το συγκεκριμένο config file όπως και το προηγούμενο για τα μοναδικά

αντικείμενα απαλλάσσει τον προγραμματιστή από πολλές γραμμές κώδικα που έχουν να κάνουν καθαρά με το SNMP πρωτόκολλο, αφήνοντας του ουσιαστικά μόνη επιβάρυνση το φόρτωμα και την διαχείριση των δεδομένων και των δεικτών της MIB. Το *mib2c.mfd.conf* (όπου mfd -> "Mib For Dummies") παράγει κώδικα σε C και δημιουργεί όλες τις απαραίτητες δομές και συναρτήσεις για την διαχείριση των τιμών των στοιχείων της MIB, χρησιμοποιώντας απλά από τον ορισμό του πίνακα στην MIB. Για αυτό και σε αυτή την περίπτωση χρειάζεται πριν καλέσουμε την mib2c ένας έλεγχος για συντακτικά λάθη. Επίσης, άλλο σημαντικό πλεονέκτημα που προσφέρει είναι ότι ξεχωρίζει τις μεθόδους που χρησιμοποιούνται για εντοπισμό των δεδομένων μιας εισερχόμενης αίτησης από τις μεθόδους διαχείρισης και τις μεθόδους επιστροφής των δεδομένων.

Συγκεκριμένα, ο κώδικας που παράγεται από το MFD configuration file χωρίζεται στις πιο κάτω τρεις κατηγόριες:

- **Data structures** : περιέχουν τα δεδομένα που χρειάζονται για να απαντήσει ο agent σε μια αίτηση. Υπάρχουν τέσσερις κυρίως δομές δεδομένων που χρησιμοποιούνται από το MFD module:
  i. ***user context :*** Είναι ένας δείκτης που παρέχεται στον χρηστή κατά την αρχικοποίηση του module και μπορεί να χρησιμοποιεί είτε για την πρόσβαση σε εξωτερικά δεδομένα (αντί χρήσης external variable) είτε σαν δείκτης σε συνδεδεμένη λίστα που θα φυλάει δυναμικά τις τιμές των δεικτών όπως στην συγκεκριμένη υλοποίηση του πίνακα TABLE μιας υποτιθέμενης MIB, όπως αυτό παράγεται με χρήση του *mib2c* με *–c* παράμετρο το *mib2c.mfd.conf*. Συγκεκριμένα θα είναι δείκτης στην *netsnmp_data_list* που ορίζεται κατά την αρχικοποίηση του module στο TABLE.c

```
        /*
         * a netsnmp_data_list is a simple way to store void
         * pointers. A simple string token is used to add,
         * find or remove pointers.
         */
    user_context = netsnmp_create_data_list("TABLE_NAME",
NULL, NULL);
```

  ii. ***mib context:*** Δομή που φυλάει τους δείκτες για μια στήλη του πίνακα.
  iii. ***data context :*** Περιέχει όλα τα δεδομένα που χρειάζονται για να πάρουμε ή να θέσουμε μια τιμή. Το MFD configuration file παράγει μια data context structure με επαρκή χώρο για όλα τα αντικείμενα που ορίζονται στην MIB που θα υλοποιήσουμε.
  iv. ***row request context :*** είναι αυτό που συσχετίζει μεταξύ τους όλα τα αλλά contexts. Ο container του πίνακα θα έχει ένα row request context για κάθε στήλη του πίνακα.

  Οι πιο πάνω δομές δεδομένων ορίζονται στο αρχείο TABLE.h. Ενδεικτικά φαίνονται μερικές από τις δομές που παράγονται για τα δυο υποθετικά αντικείμενα του πίνακα TABLE :

/***** Data context ******/**

```
typedef struct TABLE_data_s {
```

```
        /*
         *
TABLE_NODE1(2)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
         */
   char   TABLE_NODE1[16];
size_t      TABLE_NODE1_len; /* αριθμος των στοιχειων τυπου
                                    char (οχι bytes)*/

        /*
         *
TABLE_NODE2(4)/INTEGER/ASN_INTEGER/long(u_long)//l/A/w/E/r/d/h
         */
   u_long   TABLE_NODE2;

.....

} TABLE_data;



/*******   Mib Context    *********/
typedef struct TABLE_mib_index_s {

        /*
         *
sixtofourRelayIndex(1)/INTEGER32/ASN_INTEGER/long(long)//l/a/w/e/r/d/
h
         */
   long   sixtofourRelayIndex;


} TABLE_mib_index;



/******** Row request (rowreq) context   *********/
typedef struct TABLE_rowreq_ctx_s {

    /** this must be first for container compare to work */
    netsnmp_index        oid_idx;
    oid                  oid_tmp[MAX_TABLE_IDX_LEN];

    TABLE_mib_index        tbl_idx;

    TABLE_data              data;

    /*
     ********** user context pointer (provided during registration)
     */
    TABLE_registration_ptr TABLE_reg;

    netsnmp_data_list             *TABLE_data_list;

} TABLE_rowreq_ctx;
```

- **Data lookup:** όπου είναι η ανεύρεση των σωστών τιμών για μια αίτηση. Τις τιμές αυτές τις παίρνουμε είτε από κάποια συνδεδεμένη λίστα, από κάποιο

αρχείο κειμένου, μια βάση δεδομένων ή από το API κάποιας συσκευής. Όλες αυτές οι περιπτώσεις χειρίζονται από το *netsnmp_container* interface. Η default μέθοδος, που μπορεί να χειριστεί οποιαδήποτε κατάσταση είναι η μέθοδος *container-cached*, η οποία χρησιμοποιήθηκε και για την υλοποίηση του TABLE της sixtofour-MIB. Η μέθοδος αυτή συνδυάζει δυο χαρακτηριστικά του net-snmp: τον *cache helper* και τον *netsnmp_container*. Συγκεκριμένα, την πρώτη φορά που λαμβάνεται μια αίτηση για τον πίνακα καλείτε η a *cache_load* routine με ένα δείκτη σε ένα *netsnmp_container*. Αυτή η συνάρτηση έχει πρόσβαση στα δεδομένα, και αφού τα ανακτήσει προσθέτει όλες τις στήλες στον container. Από το σημείο αυτό και μετά, μέχρι να λήξει ο χρόνος προθεσμίας της cache όπου θα γίνει ανανέωση της cache, ο container είναι υπεύθυνος να βρίσκει τις κατάλληλες στήλες για τις αιτήσεις που έρχονται.
Δηλαδή, στην συγκεκριμένη περίπτωση, ο ρόλος του helper είναι απλός: ελέγχει αν τα δεδομένα που έχουν φορτωθεί είναι 'εμπρόθεσμα' ανάλογα με το χρόνο που εμείς έχουμε ορίσει σαν cache->timeout, και αν όχι καλεί την "load_cache" routine να φορτώσει ξανά τα νέα δεδομένα. Επιπλέον, έχουμε τη δυνατότητα να ορίσουμε για timeout τιμή το -1, υποχρεώνοντας την cache να ξαναφορτώνεται μετά από κάθε αίτηση. Σε περίπτωση που ο agent ανιχνεύσει μια 'ληγμένη' cache, καλεί την συνάρτηση *free_cache* για να καθαρίσει την cache και ακολούθως ξανακαλεί την *load_cache* για να ξαναφορτώσει τα νέα δεδομένα στον container. Επιπλέον, είναι διαθέσιμος ένας αριθμός flags χρησιμοποιηθούν που καθορίζουν την συμπεριφορά της cache, οι οποίες αναφέρονται και εξηγούνται αναλυτικά στην ιστοσελίδα για τον cache_handler helper.

Η διαδικασία της ανεύρεσης των σωστών τιμών και η εισαγωγή του στον container, υλοποιείται κυρίως στα TABLE_data_access αρχεία, όπου και αυτά είναι το κυριότερο μέρος του κώδικα που πρέπει να εισαγάγει ο χρηστής, καθορίζοντας κυρίως πως θα χειρίζονται και πως θα φορτώνονται οι δείκτες και τα δεδομένα στη cache καθώς και από που θα ανακτώνται τα απαραίτητα δεδομένα.
Πιο κάτω δίνεται και επεξηγείται ο κώδικας που παράγεται καλώντας το mib2c με παράμετρο –c το mib2.mfd.conf:

TABLE_data_access.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *       version : 1.12 $ of : mfd-data-access.m2c,v $
 *
 * $Id:$
 */
/* standard Net-SNMP includes */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>

/* include our parent header */
#include "TABLE.h"
```

```
#include "TABLE_data_access.h"

/*
 * TABLE is subid 8 of <the upper node of TABLE on the mib-tree>.
 * Its status is Current.
 * OID: .1.3.6.1.3.x.x.x.x.x, length: y
*/

/**
 * ΑΡΧΙΚΟΠΟΙΣΗ ΤΗΣ ΜΕΘΟΔΟΥ ΠΡΟΣΒΑΣΗΣ ΣΤΑ ΔΕΔΟΜΕΝΑ ΤΟΥ TABLE
 *
 * Η μέθοδος αυτή καλείται κατά το ξεκίνημα για να επιστρέψει
 * την δέσμευση των κατάλληλων δομών δεδομένων που θα χρησιμοποιηθούν
 * σε αυτή την φάση της ανεύρεσης των δεδομένων.
 * Η συνάρτηση  TABLE_init_data ορίζεται στο TABLE.c
 *
 * @param TABLE_reg
 *        Δείκτης σε TABLE_registration
 *
 * @retval MFD_SUCCESS : επιτυχία.
 * @retval MFD_ERROR   : ανεπανόρθωτο σφάλμα.
 */
int
TABLE_init_data(TABLE_registration_ptr TABLE_reg)
{
    DEBUGMSGTL(("verbose:TABLE:TABLE_init_data","called\n"));


    return MFD_SUCCESS;
} /* TABLE_init_data */

/**
 * container-cached overview
 *
 */

/**********************************************************************
***
 *
 * cache
 *

**********************************************************************
**/
/**
 * ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΟΥ CONTAINER
 *
  * @param container_ptr_ptr :  δείκτης σε δείκτη container. Αν
  * δημιουργήσουμε ένα custom container, θα χρησιμοποιήσουμε αυτή την
  * παράμετρο για να τον επιστρέψουμε στον MFD helper. Αν τεθεί σε
  * NULL, ο MFD helper θα δεσμεύσει για μας ένα container.
  * @param  cache : Ένας δείκτης σε μια δομή cache(τον helper μας
  * στην συγκεκριμένη περίπτωση). Χρησιμοποιώντας αυτόν τον δείκτη
  * θέτουμε τον χρόνο κατά τον οποίον ο container παραμένει έγκυρος
  * και τις υπόλοιπες flags.
  *
  * Και αυτή η συνάρτηση καλείται κατά το ξεκίνημα του agent,
  * επιτρέποντας στον χρηστή να καθορίσει τα διάφορα ζητήματα που
  * αφορούν στην πρόσβαση στα δεδομένα. Η default συμπεριφορά είναι
  * να χρησιμοποιείται ο cache helper που απλώς ελέγχει το κατά πόσο
```

```
  * τα δεδομένα του container είναι ενημερωμένα βάση του timeout που
  * ορίζει ο χρηστής, και αν ο χρόνος έχει λήξει καλούνται οι
  * cache_free και η cache_load συναρτήσεις.
  *
  * @remark : Επίσης, σε αυτό το σημείο πρέπει να γίνουν οι οποίες
  * αρχικοποιήσεις χρειάζονται για την πρόσβαση στα δεδομένα, όπως να
  * ανοικτή η σύνδεση με άλλη διεργασία που θα παρέχει τα δεδομένα,
  * να ανοικτή μια βάση δεδομένων κ.λ.π
  */
void
TABLE_container_init(netsnmp_container **container_ptr_ptr,
                     netsnmp_cache *cache)
{
    DEBUGMSGTL(("verbose:TABLE:TABLE_container_init","called\n"));

    if((NULL == cache) || (NULL == container_ptr_ptr)) {
        snmp_log(LOG_ERR,"bad params to TABLE_container_init\n");
        return;
    }

    /*
     * Σε αυτό το σημείο μπορεί να γίνει χρήση κάποιου custom
     * container. Αν δεν δημιουργηθεί κάποιος και θέσουμε τον
     * container_ptr_ptr ίσο με NULL, ο MFD helper θα δεσμεύσει ένα
     * για μας.
     */

    *container_ptr_ptr = NULL;

    /*
      *     TODO:345:A: Στο σημείο αυτό ο χρηστή πρέπει να θέσει τις
      * ιδιότητες τις cache τις παραμέτρους που επιθυμεί. Αν
      * επιθυμείται αποδέσμευση της cache, πρέπει να τεθεί το
      * cache->enabled ίσο με μηδέν. Αν επιθυμούμε να χρησιμοποιούμε
      * τον default container, αρκεί θέσουμε τον χρόνο ανανέωσης της
      * cache TABLE_CACHE_TIMEOUT ίσο με τον χρόνο που επιθυμούμε τα
      * δεδομένα να ανανεώνονται στον container.

     */
    cache->timeout = TABLE_CACHE_TIMEOUT; /* seconds */
} /* TABLE_container_init */



/**
 * ΦΟΡΤΩΜΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΗΣ CACHE
 *
 * TODO:350:M: Πρέπει να υλοποιηθεί το φόρτωμα των δεδομένων στην
 * cache
 *
 * @param container δείκτης σε container που περιέχει τα αντικείμενα
 * που πρέπει να φορτωθούν
  *
 * @retval MFD_SUCCESS              : επιτυχία.
 * @retval MFD_RESOURCE_UNAVAILABLE : Δεν έχει πρόσβαση στην πηγή
 *                                    δεδομένων
 * @retval MFD_ERROR                : άλλο σφάλμα.
 *
 * Η συνάρτηση αυτή καλείται όταν υπάρχει κάποια αίτηση για να
 * φορτώσει τους δείκτες και προαιρετικά τα δεδομένα για κάθε στήλη
 * δεδομένων.
```

```
 * Το πρωτεύον ζήτημα σε αυτή την συνάρτηση είναι διαχείριση των
 * δεικτών της MIB, αλλά αν η πρόσβαση στα δεδομένα μας είναι εύκολη
 * (π.χ από ένα αρχείο κειμένου ή ένα socket, και φυλάγονται σε μια
 * δομή στην μνήμη, είναι καλό η ανανέωση τους να γίνεται εδώ. Αν
 * όμως η διαδικασία εμπλέκει περισσότερη δουλεία όπως αλληλεπίδραση
 * με αλλά υπάρχοντα δεδομένα και υπολογισμούς για την εύρεση τιμών,
 * τότε συνίσταται η συγκεκριμένη συνάρτηση να χρησιμοποιείται μόνο
 * για να θέτει τους δείκτες και να σώζει απλώς πληροφορίες που θα
 * είναι χρήσιμες στη συνεχεία. Σε αυτή την περίπτωση, οι
 * πληροφορίες αυτές θα χρησιμοποιηθούν από την συνάρτηση
 * TABLE_row_prep() για να δημοσιεύσει τα δεδομένα.

 * @note
 * Αν απαιτείται συνέπεια μεταξύ των στηλών του πίνακα(όπως για
 * παράδειγμα στην περίπτωση που θέλουμε τα στατιστικά για κάθε στήλη
 * να είναι από το ίδιο time frame), τότε η πρόσβαση στα δεδομένα
 * πρέπει να γίνεται μεσώ αυτής της συνάρτησης.
 */
int
TABLE_cache_load(netsnmp_container *container)
{
    TABLE_rowreq_ctx *rowreq_ctx;
    size_t                count = 0;

    /*Ο κώδικας που παράγεται με το mib2c βασίζεται σε πηγή
     * δεδομένων η οποία είναι κάποιο αρχείο κειμένου που μπορεί να
     * διαβαστεί και να 'παρσαριστεί' στις επιμέρους μεταβλητές.
     */
    FILE *filep;
    char line[MAX_LINE_SIZE];

    /*
     * προσωρινή φύλαξη των τιμών των δεικτών
     */
        /*
         *
sixtofourRelayIndex(1)/INTEGER32/ASN_INTEGER/long(long)//l/a/w/e/r/d/
h
         */
   long   sixtofourRelayIndex;

    DEBUGMSGTL(("verbose:TABLE:TABLE_cache_load","called\n"));

    /*
    ************************************************
    ***           START EXAMPLE CODE                 ***
    ***---------------------------------------------***/
    /*
     * ανοιγει το αρχειο δεδομενων.
     */
    filep = fopen("/etc/dummy.conf", "r");
    if(NULL ==  filep) {
        return MFD_RESOURCE_UNAVAILABLE;
    }

    /*
    ***---------------------------------------------***
    ***              END  EXAMPLE CODE              ***
    ************************************************/
    /*
     * TODO:351:M: |-> Σε αυτό το σημείο πρέπει να
```

```
 * φορτωθούν/ανανεωθούν τα δεδομένα στον container του πίνακα.
 * Πρέπει να προσπελαστούν τα δεδομένα, να δεσμευθεί ο rowreq
 * context, να υπολογιστούν και να τεθούν οι δείκτες [και
 * προαιρετικά, για τις περιπτώσεις που αναφέρονται πιο πάνω τα
 * δεδομένα, και να εισαχθούν στον container.
 */
while( 1 ) {
/*
**************************************************
***             START EXAMPLE CODE             ***
***---------------------------------------------***/
/*
 * παίρνει τα περιεχόμενα μιας γραμμής αγνοώντας τις κενές
 * γραμμές
 */
do {
    if (!fgets(line, sizeof(line), filep)) {
        /* we're done */
        fclose(filep);
        filep = NULL;
    }
} while (filep && (line[0] == '\n'));

/*
 * ελεγχος αν εχει φτασει στο τελος των δεδομενων
 */
if(NULL == filep)
    break;

/*
 * Σε αυτό το σημείο πρέπει να 'παρσάρει' τα δεδομένα της γραμμής
 *στις επιμέρους μεταβλητές που θα εισαγάγει στον container
 */
/*
***---------------------------------------------***
***               END  EXAMPLE CODE             ***
**************************************************/

    /*
     * TODO:352:M: |   |-> Στο σημείο αυτό πρέπει να υπολογίσει
     * τους δείκτες και να τους θέσει στον νέο rowreq context του
     * πίνακα. Μια καλή λύση για την διαχείριση δεικτών που
     * αυξάνονται και μειώνονται κατά την διάρκεια της
     * λειτουργίας του agent, είναι η χρήση του δείκτη user
     * context (TABLE_data_list), ο οποίος έχει αρχικοποιηθεί να
     * δείχνει στην συνδεδεμένη λίστα "TABLE" στο αρχείο TABLE.c.
     * Έτσι με χρήση συνδεδεμένης λίστας γίνεται ευκολότερη η
     * δυναμική προσθαφαίρεση δεικτών. Για την διαχείριση της
     * συνδεδεμένης λίστας μπορεί να γίνει χρήση των έτοιμων
     * snmp-εντολών του αρχείου TABLE_enums.h.
     */
    rowreq_ctx = TABLE_allocate_rowreq_ctx();
    if (NULL == rowreq_ctx) {
        snmp_log(LOG_ERR, "memory allocation failed\n");
        return MFD_RESOURCE_UNAVAILABLE;
    }
    if(MFD_SUCCESS != TABLE_indexes_set(rowreq_ctx
                          , sixtofourRelayIndex
           )) {
        snmp_log(LOG_ERR,"error setting index while loading "
                 "TABLE cache.\n");
```

```
            TABLE_release_rowreq_ctx(rowreq_ctx);
            continue;
        }

        /*
         * TODO:352:r: |   |-> Σε αυτό το σημείο πρέπει να
         * δημοσιοποιεί τα δεδομένα του data context του πίνακα.
         * (ειδάλλως θα περιμένει μέχρι την συνάρτηση row_prep).Η
         * ανάθεση γίνεται στο κατάλληλο κάθε φορά πεδίο της δομής
         * rowreq_ctx->data.<αντικείμενο του πίνακα> .(η δομή αυτή
         * ορίζεται στο TABLE.h)
         */
    /*
     * TRANSIENT or semi-TRANSIENT data:
     * αρχικά θα αντιγράψει ή θα σώσει τις πληροφορίες που του
     * χρειάζονται για την row_prep.
     */

    /* Ενδεικτικά, φαίνεται ο κώδικας που παράγεται για ένα
     * αντικείμενο τύπου Display String και ένα τύπου INTEGER.

    /*
     * υπολογίζει/σώζει τα δεδομένα για το πρώτο αντικείμενο του
     * πίνακα TABLE_NODE1
     * Περίπτωση Display String
     *
TABLE_NODE1(2)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
     */
    /*
     * TODO:246:r: |-> Ορισμένοι ASN τύποι χρειάζονται να γίνει η
     * συσχέτιση τους(mapping) με τις τιμές που θα επιστρέψουν.
     * Χαρακτηριστικότερο παράδειγμα είναι ο τύπος INTEGER όπου μια
     * ακέραια μεταβλητή συμβολίζει μια γραμματοσειρά με ειδικό νόημα
     * για την εκάστοτε MIB. Επίσης, αυτό το συσχετισμό συνίσταται
     * και σε μεταβλητές που επιστρέφουν τύπους Display String, για
     * να γίνεται αν είναι αναγκαία η απαιτούμενη επαναδέσμευση
     * μνήμης και να μην προκύψει τέτοιο πρόβλημα κατά την εκτέλεση
     *του προγράμματος.

     * Γίνεται ο συσχετισμός μεταξύ των 'φυσικών' τιμών των
     * μεταβλητών της στήλης του πίνακα και των αντικειμένων της
     * MIB.(προαιρετικά, ο χρηστής καλείτε να αφαιρέσει τα σχόλια αν
     * επιθυμεί συσχέτιση-mapping. Είναι υποχρεωτική μόνο σε
     * περίπτωση INTEGER)
     *
     * if(MFD_SUCCESS !=
     *    TABLE_NODE1_map(&rowreq_ctx->data.TABLE_NODE1, &rowreq_ctx-
>data.TABLE_NODE1_len,
     *                    <μεταβλητή που κρατά την τιμή του αντικειμένου
     * όπως την πήρε από την πηγή δεδομένων που έχει ανοίξει- έστω
     * TABLE_NODE1>,<μέγεθος της μεταβλητής-έστω TABLE_NODE1_len>,
     * 0)) {
     *    return MFD_ERROR;
     * }
     */
    /*
     * ελέγχουμε ότι υπάρχει διαθέσιμη μνήμη για τα δεδομένα του
     * συγκεκριμένου αντικειμένου
     */
    if ((NULL == rowreq_ctx->data. TABLE_NODE1) || (rowreq_ctx->data.
TABLE_NODE1_len < TABLE_NODE1_len)) {
```

```
        snmp_log(LOG_ERR,"not enough space for value\n");
        return MFD_ERROR;
    }
    rowreq_ctx->data. TABLE_NODE1_len = TABLE_NODE1_len;
    memcpy( rowreq_ctx->data. TABLE_NODE1, TABLE_NODE1,
            rowreq_ctx->data.TABLE_NODE1_len * sizeof(rowreq_ctx-
>data. TABLE_NODE1 [0]) );


     * υπολογίζει/σώζει τα δεδομένα για το πρώτο αντικείμενο του
     * πινακα TABLE_NODE2
     * Περιπτωση INTEGER
TABLE_NODE2(4)/INTEGER/ASN_INTEGER/long(u_long)//l/A/w/E/r/d/h
     */
    /*
     * TODO:246:r: |-> Σε αυτό το σημείο πρέπει να γίνει ο
     * συσχετισμός(mapping) μεταξύ των 'φυσικών τιμών των μεταβλητών,
     * έτσι όπως έχουν υπολογιστεί, με τις τιμές της MIB
     * Στην περίπτωση απαρίθμησης όπως γίνεται με τα αντικείμενα
     * τύπου INTEGER είναι αναγκαίος ο συσχετισμός(mapping)
     * Σημείωση: Οι συναρτήσεις για την συσχέτιση των μεταβλητών
     * ορίζονται στο αρχείο TABLE_data_get.c.
    */
    if(MFD_SUCCESS !=
       TABLE_NODE2_map(&rowreq_ctx->data.TABLE_NODE2, TABLE_NODE2 ))
{
        return MFD_ERROR;
    }



        /*
         * εισάγονται οι τιμές στον container
         */
        CONTAINER_INSERT(container, rowreq_ctx);
        ++count;
    }

    /*
    ************************************************
    ***             START EXAMPLE CODE             ***
    ***-----------------------------------------------***/
    if(NULL != filep)
        fclose(filep);
    /*
    ***-----------------------------------------------***
    ***              END  EXAMPLE CODE             ***
    ************************************************/

    DEBUGMSGT(("verbose:TABLE:TABLE_cache_load",
               "inserted %d records\n", count));

    return MFD_SUCCESS;
} /* TABLE_cache_load */


/**
 * 'ΚΑΘΑΡΙΣΜΟΣ' ΤΗΣ CACHE :
 * όταν ο agent δει ότι χρόνος που τα δεδομένα της cache είναι έγκυρα
 * έχει τελειώσει, καλείται η ρουτίνα για 'ξεφόρτωμα-εκκαθάριση' της
 * cache. Ο κώδικας του MFD interface καλεί την cache_free συνάρτηση
```

```
 * για να επιτρέψει στον handler να κάνει το απαραίτητο καθάρισμα. Ο
 * χρηστής δεν πρέπει να μετακινήσει αντικείμενα από τον container
 * και ούτε να σβήσει κάτι, αφού αυτό θα διακανονιστεί αυτόματα από
 *τον container
 *
 * @param container : ο container με όλα τα τρέχοντα αντικείμενα
 *
 * Η κλήση αυτή γίνεται πριν όλα τα αντικείμενα αφαιρεθούν από τον
 * container. Αν πριν από αυτό χρειάζεται κάποια επιπλέον επεξεργασία
 * πρέπει να γίνει εδώ.
 * @note
 *  Ο MFD helper θα αποδεσμεύσει όλους τους row contexts.
 *
 */
void
TABLE_cache_free(netsnmp_container *container)
{
    DEBUGMSGTL(("verbose:TABLE:TABLE_cache_free","called\n"));

    /*
     * TODO:380:M: Σε αυτό το σημείο πρέπει να γίνει ο
     * 'καθαρισμός΄ της cache.
     * Αν η απόκτηση των δεδομένων γίνεται προσπελαύνοντας απλώς
     * κάποια εξωτερικά αρχεία κειμένου και κάποια sockets, τότε δεν
     * χρειάζεται επιπλέον ενέργεια για αυτό το στάδιο από τον
     * χρηστή, αφού το flushing της cache που ,ούτως ή άλλως γίνεται,
     * ικανοποιεί τις ανάγκες μας.
     */
} /* TABLE_cache_free */


/**
 * ΠΡΟΕΤΟΙΜΑΣΙΑ ΣΤΗΛΗΣ ΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑ.
 *
 * Όταν ο agent έχει εντοπίσει την προς επεξεργασία στήλη, καλείται
 * αυτή η συνάρτηση για να προετοιμάσει την συγκεκριμένη στήλη για
 * επεξεργασία.
 * ΑΝ τα δεδομένα έχουν διαβαστεί, υπολογιστεί και ο data context
 * έχει δημοσιοποιηθεί κατά την φάση του υπολογισμού των δεικτών, στη
 * cache_load συνάρτηση, στο στάδιο αυτό δεν θα χρειαστεί να γίνει
 * καμία ενέργεια.
 * ΑΞΙΖΕΙ ΝΑ ΣΗΜΕΙΩΘΕΙ ότι ενώ η cache_load καλείται μόνο αφού λήξει
 * ο χρόνος που έχει οριστεί από τον χρηστή όπου δεδομένα παύουν να
 * είναι έγκυρα, η row_prep συνάρτηση καλείται κάθε φορά που γίνεται
 * κάποια αίτηση. Για αυτό δεδομένα που χρειάζονται συνεχή ανανέωση
 * πρέπει να δημοσιοποιούνται μεσώ αυτής της συνάρτησης και όχι της
 * cache_load.
 *
 * @param rowreq_ctx : δεικτη σε ενα context.
 *
 * @retval MFD_SUCCESS     : επιτυχια.
 * @retval MFD_ERROR       : αλλο σφαλμα.
 */
int
TABLE_row_prep( TABLE_rowreq_ctx *rowreq_ctx)
{
    DEBUGMSGTL(("verbose:TABLE:TABLE_row_prep","called\n"));

    netsnmp_assert(NULL != rowreq_ctx);

    /*
```

```
     * TODO:390:o: Σε αυτό το σημείο πρέπει να γίνει η επεξεργασία
     * της στήλης για την οποία έγινε η αίτηση.
     * Αν δεν έχει γίνει στην cache_load η δημοσίευση του data
     * context με τα απαραίτητα προς επιστροφή δεδομένα που ζητούνται
     * από την αίτηση, σε αυτό το σημείο πρέπει να συμπληρωθεί η
     * στήλη.
     */

    return MFD_SUCCESS;
} /* TABLE_row_prep */
```

- **Data manipulation :** όπου είναι η διαδικασία είτε της επιστροφής των ήδη υπαρχόντων από τα προηγούμενα σταδία τιμών, είτε το θέσιμο των νέων τιμών σε περίπτωση settable αντικειμένων. Το δεύτερο ενδεχόμενο δεν θα μας απασχολήσει σε αυτή την φάση.
  Αφού έχει ολοκληρωθεί η ανεύρεση του data context (δηλαδή οι κατάλληλες δομές δεδομένων) του πίνακα στο προηγούμενο 'data lookup' στάδιο του προγράμματος , θα κληθούν οι ρουτίνες get του κάθε κόμβου. Αυτές υλοποιούνται στο αρχείο TABLE_data_get.c που παράγεται από το mib2c –c mib2c.mfd.conf, όπου παράγεται μια get συνάρτηση για κάθε αντικείμενο που ορίζεται στην MIB μας.
  Αν κατά την υλοποίηση της mib μας χρησιμοποιήσουμε τον default data context που παράγεται, και η MIB μας χρησιμοποιεί απλούς τύπους δεδομένων, ο κώδικας που παράγεται για τις get συναρτήσεις χρειάζεται λίγη έως καθόλου αλλαγή. Επίσης, κατά την κλήση της TABLE_data_get.c καλούνται και οι διάφορες ρουτίνες συσχετισμού (mapping routines) που αναφέρθηκαν προηγουμένως. Οι εκάστοτε συσχετισμοί παράγονται βάση της MIB που έχουμε γράψει και δημοσιοποιήσει και τοποθετούνται στο *TABLE_enums.h.*
  Η μόνη περίπτωση που θα χρειαστεί να μεταβάλουμε τον κώδικα που παράγεται είναι στην περίπτωση που ο τύπος των δεδομένων που ορίζονται από την MIB δεν ταιριάζει με τα δεδομένα που επιστρέφει το εκάστοτε λειτουργικό σύστημα. Για παράδειγμα, χρησιμοποιώντας το λειτουργικό σύστημα LINUX, και προσπελαύνοντας κάποια δεδομένα μεσώ της ioctl(), οι τιμές που επιστρέφονται δεν ταιριάζουν συνήθως με το format των δεδομένων της MIB. Σε αυτή την περίπτωση, ο χρηστής πρέπει μέσα στην TABLE_data_get.c να ορίσει την απαραίτητη συσχέτιση μεταξύ των δεδομένων.
  Πιο κάτω φαίνεται ο κώδικας που παράγεται με χρήση του mib2c –c mib2c.mfd.conf για ένα μέρος του πίνακα sixtofourRelay που ορίζεται ολόκληρος στην SIXTOFOUR-MIB, και ο οποίο αποτελείται από τον δείκτη τύπου integer, ένα υποθετικό αντικείμενο TABLE_NODE1 ASN τύπου Display String και ένα αντικείμενο TABLE_NODE2 ASN τύπου INTEGER που παίρνει δυο τιμές, 1 και 2.

TABLE_data_get.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *       version : 1.18.2.1 $ of : mfd-data-get.m2c,v $
```

```
 *
 * $Id:$
 */
/* standard Net-SNMP includes */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>

/* include our parent header */
#include "TABLE.h"


/** @defgroup data_get data_get: Ρουτίνες για την ανάκτηση των
 * δεδομένων
 *
 * TODO:230:M: Πρέπει να υλοποιηθούν οι get routines του πίνακα TABLE
 * TODO:240:M: Πρέπει να υλοποιηθούν οι ρουτίνες συσχέτισης του
 *πίνακα TABLE(mapping routines) όπου καθίσταται αναγκαίο.
 *
 * Οι ρουτίνες αυτές είναι για να παίρνουν τα δεδομένα για τα
μοναδικά αντικείμενα του πίνακα. Σε αυτές θα περαστεί ο row context
όπως βρέθηκε στη προηγούμενη φάση(data lookup) μαζί με ένα δείκτη στη
μνήμη στο σημείο όπου θα αντιγράφει η τιμή του.
 * @{
 */
/* ---------------------------------------------------------------------
 * TODO:200:r: Πρέπει να γίνει η υλοποίηση των συναρτήσεων data
context του πίνακα .
 */
/* Ο συγκεκριμένος κώδικας παράγεται με χρήση του mib2c -c
* mib2c.mfd.conf για την SIXTOFOUR-MIB, για τον πίνακα
* sixtofourRelayTable, και για χάρη γενικότητας θεωρούμε δυο μόνο
* υποθετικά αντικείμενα TABLE_NODE1 και TABLE_NODE2, ένα τύπου
* Display String και ένα τύπου INTEGER που χρησιμοποιεί δυο
* απαριθμήσεις για επιστροφή των τιμών case1(1) και case2(2), καθώς
* και τον δείκτη sixtofourRelayIndex του πίνακα που είναι τύπου
* integer, και απλώς αναθέτει ένα μοναδικό αριθμό σε κάθε έγγραφη του
* πίνακα.

/*---------------------------------------------------------------------
 * SIXTOFOUR-MIB::sixtofourRelayEntry.sixtofourRelayIndex
 * sixtofourRelayIndex is subid 1 of sixtofourRelayEntry.
 * Its status is Current, and its access level is NoAccess.
 * OID: .1.3.6.1.3.19.1.1.8.1.1
 * Description:
         XXXXXXXXXXXXXXXXXXXX
 *
 * Attributes:
 *   accessible 0     isscalar 0     enums  0      hasdefval 0
 *   readable   0     iscolumn 1     ranges 0      hashint   0
 *   settable   0
 *
 * Πρόκειται για αντικείμενο τύπου INTEGER32
 * Ο net-snmp είναι ASN_INTEGER. Ο ορισμός του σε C είναι long.
 *
 *
 * NOTE: Ο κόμβος sixtofourRelayIndex είναι NOT ACCESSIBLE
 *
 */
/**
 * Συσχέτισε μια τιμή όπως έχει παρθεί από το φυσικό της format στο
```

```
 * ζητούμενο MIB format
 *
 * @retval MFD_SUCCESS         : επιτυχία
 * @retval MFD_ERROR           : άλλο σφάλμα
 *
 * @note parameters follow the memset convention (dest, src).
 *
 * @note : η παραγωγή και η χρήση αυτής της συνάρτησης μπορεί να
 * αποφευχθεί ξανατρέχοντας mib2c αφότου προσθέσουμε την επόμενη
 * γραμμή στο αρχείο default-node-sixtofourRelayIndex.m2d :
 *   @eval $m2c_node_skip_mapping = 1@
 *
 * @remark
 * Αν οι τιμές των διάφορων τύπων δεδομένων δεν ταιριάζουν ακριβώς με
 * τις τιμές που ορίζονται στην MIB, σε αυτό το σημείο πρέπει να
 * γίνει ο συσχετισμός(mapping). Στην αντίθετη περίπτωση, να γίνει
 * απλώς η αντιγραφή τους.
*/
int
sixtofourRelayIndex_map(long *mib_sixtofourRelayIndex_val_ptr, long
raw_sixtofourRelayIndex_val)
{
    netsnmp_assert(NULL != mib_sixtofourRelayIndex_val_ptr);

    DEBUGMSGTL(("verbose:TABLE:sixtofourRelayIndex_map","called\n"));

    /*
     * TODO:241:o: |-> Σε αυτο το σημειο πρεπει να γινει ο
     * συσχετισμός του sixtofourRelayIndex, αν οι τιμές του δεν
     * ταιριάζουν με αυτές που ορίζονται στην MIB.
     */
    (*mib_sixtofourRelayIndex_val_ptr) = raw_sixtofourRelayIndex_val;

    return MFD_SUCCESS;
} /* sixtofourRelayIndex_map */


/**
 * set mib index(es)
 *
 * @param tbl_idx : δεικτης σε mib δομη.
 *
 * @retval MFD_SUCCESS     : επιτυχια.
 * @retval MFD_ERROR       : αλλο σφαλμα.
 *
 * @remark
 * Μια χρήσιμη συνάρτηση όπου θέτει όλα τα επιμέρους στοιχεία του
 * πεδίου INDEX της MIB ( αν είναι πολλαπλά) με μια και μόνο κλήση.
 * Προϋποθέτει φυσικά ότι οι τιμές σε C που έχουν ανακτηθεί, έχουν
 * ήδη συσχετιστεί με το format που ορίζεται στην MIB.
 * Στην συγκεκριμένη περίπτωση πρέπει να τεθεί μόνο η τιμή του
 * δείκτη.
 */
int
TABLE_indexes_set_tbl_idx(TABLE_mib_index *tbl_idx, long
sixtofourRelayIndex_val)
{

DEBUGMSGTL(("verbose:TABLE:TABLE_indexes_set_tbl_idx","called\n"));
```

```
    /*
sixtofourRelayIndex(1)/INTEGER32/ASN_INTEGER/long(long)//l/a/w/e/r/d/
h */
    tbl_idx->sixtofourRelayIndex = sixtofourRelayIndex_val;


    return MFD_SUCCESS;
} /* TABLE_indexes_set_tbl_idx */

/**
 * @internal
 * πρέπει να τεθεί ο row context των δεικτών
 *
 * @param rowreq_ctx : ο row context ο οποίος χρειάζεται
 * 'ενημερωμένους' δείκτες
 *
 * @retval MFD_SUCCESS     : επιτυχία.
 * @retval MFD_ERROR       : άλλο σφάλμα.
 *
 * @remark
 * Η συνάρτηση αυτή θέτει τους mib indexes και ακολούθως ανανεώνει
 * τους oid indexes από τους mib indexes.
 */
int
TABLE_indexes_set(TABLE_rowreq_ctx *rowreq_ctx, long
sixtofourRelayIndex_val)
{
    DEBUGMSGTL(("verbose:TABLE:TABLE_indexes_set","called\n"));

    if(MFD_SUCCESS != TABLE_indexes_set_tbl_idx(&rowreq_ctx->tbl_idx
                                   , sixtofourRelayIndex_val
           ))
        return MFD_ERROR;

    /*
     * μετατρεπει τον mib index σε oid index
     */
    rowreq_ctx->oid_idx.len = sizeof(rowreq_ctx->oid_tmp) /
sizeof(oid);
    if(0 != TABLE_index_to_oid(&rowreq_ctx->oid_idx,
                                    &rowreq_ctx->tbl_idx)) {
        return MFD_ERROR;
    }

    return MFD_SUCCESS;
} /* TABLE_indexes_set */


/*----------------------------------------------------------------------
--
 * SIXTOFOUR-MIB::sixtofourRelayEntry.TABLE_NODE1
 * Το TABLE_NODE1 είναι το δεύτερο παρακλάδι του sixtofourRelayEntry.
 * Το status του είναι Current, και το επίπεδο access του ReadOnly
 * OID: .1.3.6.1.3.19.χ.χ.χ.χ.2
 * Description:
       XXXXXXXXXXXXXXXXXXXXX
 *
 * Attributes:
 *   accessible 1     isscalar 0     enums  0      hasdefval 0
 *   readable   1     iscolumn 1     ranges 1      hashint   1
 *   settable   0
```

```
 *   hint: 255a
 *
 * Ranges:  40;
 *
 * Πρόκειται για αντικείμενο DisplayString (βασισμένο στον perltype
 * OCTETSTR)
 *Its syntax is DisplayString (based on perltype OCTETSTR)
 * Ο net-snmp τύπος είναι ASN_OCTET_STR. Ο ορισμός του σε C είναι
 * τύπου char (char)
 * Τα δεδομένα αυτά απαιτούν να ορίζεται το μέγεθος τους   (Max 40)
 */
/**
 * γίνεται η συσχέτιση μιας τιμής από το αρχικό της format στο format
 * που ορίζεται στην mib.
 map a value from its original native format to the MIB format.
 *
 * @retval MFD_SUCCESS         : επιτυχία
 * @retval MFD_ERROR           : Άλλο σφάλμα
 *
 * @note : η παραγωγή και η χρήση αυτής της συνάρτησης μπορεί να
 * αποφευχθεί ξανατρέχοντας mib2c αφότου προσθέσουμε την επόμενη
 * γραμμή στο αρχείο default-node-sixtofourRelayIndex.m2d :
 *   @eval $m2c_node_skip_mapping = 1@
 *
 * @remark
 * Αν οι τιμές των διάφορων τύπων δεδομένων δεν ταιριάζουν ακριβώς με
 * τις τιμές που ορίζονται στην MIB, σε αυτό το σημείο πρέπει να
 * γίνει ο συσχετισμός. Στην αντίθετη περίπτωση, να γίνει απλώς η
 *αντιγραφή τους.
 */
int
TABLE_NODE1_map(char **mib_TABLE_NODE1_val_ptr_ptr, size_t
*mib_TABLE_NODE1_val_ptr_len_ptr, char *raw_TABLE_NODE1_val_ptr,
size_t raw_TABLE_NODE1_val_ptr_len, int allow_realloc)
{
    int converted_len;

    netsnmp_assert(NULL != raw_TABLE_NODE1_val_ptr);
    netsnmp_assert((NULL != mib_TABLE_NODE1_val_ptr_ptr) && (NULL !=
mib_TABLE_NODE1_val_ptr_len_ptr));

    DEBUGMSGTL(("verbose:TABLE:TABLE_NODE1_map","called\n"));

    /*
     * TODO:241:r: |-> Πρέπει να υλοποιηθεί ο συσχετισμός του
     * TABLE_NODE1, που ουσιαστικά στην περίπτωση Display String έχει
     * να κάνει με το δέσμευση μνήμης για τα δεδομένα τα στήλης αν το
     * μέγεθος τις είναι διαφορετικό από αυτό που ορίζει η
     * συγκεκριμένη για τον πίνακα μας mib.
     * Σε περίπτωση που το μέγεθος των δεδομένων που εισέρχονται στον
     * data context είναι καθορισμένο και σταθερό, δεν χρειάζεται να
     * γίνει αυτή η συσχέτιση. Αν όμως, το μέγεθος των δεδομένων της
     * μνήμης δεν ανταποκρίνεται στο μέγεθος των δεδομένων της mib,
     * τίθεται το converted_len ίσο με το μέγεθος που απαιτείται.
     */
    converted_len = raw_TABLE_NODE1_val_ptr_len; /* assume equal */
    if((NULL == *mib_TABLE_NODE1_val_ptr_ptr) ||
(*mib_TABLE_NODE1_val_ptr_len_ptr < converted_len)) {
        if(! allow_realloc) {
            snmp_log(LOG_ERR,"not enough space for value mapping\n");
            return SNMP_ERR_GENERR;
```

```
        }
        *mib_TABLE_NODE1_val_ptr_ptr = realloc(
*mib_TABLE_NODE1_val_ptr_ptr, converted_len *
sizeof(**mib_TABLE_NODE1_val_ptr_ptr));
        if(NULL == *mib_TABLE_NODE1_val_ptr_ptr) {
            snmp_log(LOG_ERR,"could not allocate memory\n");
            return SNMP_ERR_GENERR;
        }
    }
    *mib_TABLE_NODE1_val_ptr_len_ptr = converted_len;
    memcpy( *mib_TABLE_NODE1_val_ptr_ptr, raw_TABLE_NODE1_val_ptr,
converted_len );

    return MFD_SUCCESS;
} /* TABLE_NODE1_map */

/**
 * Ανάκτηση της τρέχουσας τιμής των δεδομένων του αντικειμένου
 * TABLE_NODE1
 *
 * Η τιμή τίθεται βάση του data context της στήλης.
 *
 * @param rowreq_ctx
 *        Δεικτης στον row request context.
 * @param TABLE_NODE1_val_ptr_ptr
 *        Δείκτης προς χώρο μνήμης δεσμευμένο για μια μεταβλητή τύπου
 *         char
 * @param TABLE_NODE1_val_ptr_len_ptr
 *         Δείκτης σε size_t που δείχνει κατά την είσοδο το μέγεθος
 *         μνήμης σε bytes που δείχνει ο TABLE_NODE1 και κατά την
 *         έξοδο περιέχει το μέγεθος των δεδομένων.
 *
 *
 * @retval MFD_SUCCESS         : επιτυχία
 * @retval MFD_SKIP            : προσπέρασε αυτό τον κόμβο(μην
 *                               επιστέψεις τιμή)
 * @retval MFD_ERROR           : άλλο σφάλμα
 *
 * @note : Αν χρειάζονται περισσότερα από
 * (*TABLE_NODE1_val_ptr_len_ptr) bytes μνήμης, να δεσμευτούν
 * κάνοντας χρήση της malloc() και να ανανεωθεί η τιμή του
 * TABLE_NODE1_val_ptr_ptr.
 * Προσοχή: Δεν χρειάζεται αποδέσμευση μνήμης για τον προηγούμενο
 * δείκτη. Ο MFD helper θα απελευθερώσει την μνήμη που δεσμεύτηκε.
 * @remark : Αν όμως κληθεί η συνάρτηση από τον χρηστή πρέπει ο ίδιος
 * να αποδεσμεύσει την μνήμη που θα δεσμεύσει.
 */
int
TABLE_NODE1_get( TABLE_rowreq_ctx *rowreq_ctx, char
**TABLE_NODE1_val_ptr_ptr, size_t *TABLE_NODE1_val_ptr_len_ptr )
{
   /** πρέπει μέχρι εδώ να έχουμε ένα non-NULL pointer και επαρκή
    * χώρο μνήμης  */
   netsnmp_assert( (NULL != TABLE_NODE1_val_ptr_ptr) && (NULL !=
*TABLE_NODE1_val_ptr_ptr));
   netsnmp_assert( NULL != TABLE_NODE1_val_ptr_len_ptr );


    DEBUGMSGTL(("verbose:TABLE:TABLE_NODE1_get","called\n"));

    netsnmp_assert(NULL != rowreq_ctx);
```

```
/*
 * TODO:231:o: |-> Στο σημείο αυτό ο agent πρέπει να πάρει την
 * τρέχουσα τιμή  των δεδομένων του TABLE_NODE1.
 * πρέπει να τεθεί ο (* TABLE_NODE1_val_ptr_ptr ) και (*
 * TABLE_NODE1_val_ptr_len_ptr ) from rowreq_ctx->data
 */
    /*
     * επιβεβαίωσε ότι υπάρχει επαρκής χώρος μνήμης για τα δεδομένα
     *της έγγραφης TABLE_NODE1
     */
    if ((NULL == (* TABLE_NODE1_val_ptr_ptr )) || ((*
TABLE_NODE1_val_ptr_len_ptr ) < rowreq_ctx->data.TABLE_NODE1_len)) {
        /*
         * δέσμευσε χώρο για τα δεδομένα του TABLE_NODE1
         */
        (* TABLE_NODE1_val_ptr_ptr ) = malloc(rowreq_ctx-
>data.TABLE_NODE1_len * sizeof((* TABLE_NODE1_val_ptr_ptr )[0]));
        if(NULL == (* TABLE_NODE1_val_ptr_ptr )) {
            snmp_log(LOG_ERR,"could not allocate memory\n");
            return MFD_ERROR;
        }
    }
    (* TABLE_NODE1_val_ptr_len_ptr ) = rowreq_ctx-
>data.TABLE_NODE1_len;
    memcpy( (* TABLE_NODE1_val_ptr_ptr ), rowreq_ctx-
>data.TABLE_NODE1,
            (* TABLE_NODE1_val_ptr_len_ptr ) * sizeof((*
TABLE_NODE1_val_ptr_ptr )[0]) );

    return MFD_SUCCESS;
} /* TABLE_NODE1_get */

/*---------------------------------------------------------------------
 * Η διαδικασία είναι ίδια όπως για την προηγούμενη έγγραφη, με την
 * μόνη διάφορα ότι εδώ επειδή πρόκειται για αντικείμενο τύπου
 * integer, χρειάζεται υποχρεωτικά η συσχέτιση και επιπλέον δεν
 * χρειάζεται όλη η προηγούμενη διαδικασία για την δέσμευση μνήμης.
 * SIXTOFOUR-MIB::sixtofourRelayEntry. TABLE_NODE2
 * TABLE_NODE2 is subid 3 of sixtofourRelayEntry.
 * Its status is Current, and its access level is ReadOnly.
 * OID: .1.3.6.1.3.χ.χ.χ.χ.1.3
 * Description:
TABLE_NODE1:
       XXXXXXXXXXXXXXXXXXXXXXXXXXX
 *
 * Attributes:
 *   accessible 1     isscalar 0     enums  1      hasdefval 0
 *   readable   1     iscolumn 1     ranges 0      hashint   0
 *   settable   0
 *
 * Enum range: 2/8. Values:  case1(1), case2(2)
 *
 * Το αντικείμενο είναι τύπου INTEGER.
 * Ο net-snmp τύπος είναι ASN_INTEGER. Ο ορισμός του σε C είναι long
 * (u_long)
 */
/*
/**
 * συσχετίζει μια τιμή από το αρχικό της format με το format που
 * ορίζεται στην mib
```

```
 *
 * Τα υπόλοιπα ίδια όπως προηγουμένως
 */
int
TABLE_NODE2_map(u_long *mib_ TABLE_NODE2_val_ptr, u_long
raw_TABLE_NODE2_val)
{
    netsnmp_assert(NULL != mib_TABLE_NODE2_val_ptr);

    DEBUGMSGTL(("verbose:TABLE:TABLE_NODE2_map","called\n"));

    /*
     * TODO:241:o: |-> Να υλοποιηθεί ο συσχετισμός των απαριθμήσεων
     *του TABLE_NODE2.
     * χρησιμοποιεί INTERNAL_* macros που ορίζονται στα αρχεία
     * επικεφαλίδας. Συγκεκριμένα τα  TABLE_NODE2_CASE1 και
     * TABLE_NODE2_CASE2 ορίζονται στο αρχείο TABLE_enums.h
     */
    switch(raw_TABLE_NODE2_val) {
        case INTERNAL_TABLE_NODE2_CASE1:
             *mib_TABLE_NODE2_val_ptr = TABLE_NODE2_CASE1;
             break;

        case INTERNAL_TABLE_NODE2_CONFIGUREDADDR:
             *mib_TABLE_NODE2_val_ptr = TABLE_NODE2_CASE2;
             break;

             default:
                 snmp_log(LOG_ERR, "couldn't map value %d for
TABLE_NODE2\n", raw_TABLE_NODE2_val );
                 return MFD_ERROR;
    }

    return MFD_SUCCESS;
} /* TABLE_NODE2_map */

/**
 * ανάκτηση των δεδομένων του κόμβου TABLE_NODE2.
 *
 * Τίθεται η τιμή χρησιμοποιώντας τον row context της στήλης.
 *
 * @param rowreq_ctx
 *        Δείκτης στον row request context.
 * @param TABLE_NODE2_val_ptr
 *        Δείκτης σε χώρο μνήμης για long μεταβλητή.
 *
 * @retval MFD_SUCCESS         : επιτυχία
 * @retval MFD_SKIP            : προσπέρασε αυτό τον κόμβο(μην
 *                               επιστέψεις τιμή)
 * @retval MFD_ERROR           : Άλλο σφάλμα
 */
int
TABLE_NODE2_get( TABLE_rowreq_ctx *rowreq_ctx, u_long *
TABLE_NODE2_val_ptr )
{
   /** we should have a non-NULL pointer */
   netsnmp_assert( NULL != TABLE_NODE2_val_ptr );


    DEBUGMSGTL(("verbose:TABLE:TABLE_NODE2_get","called\n"));
```

```
    netsnmp_assert(NULL != rowreq_ctx);

/*
 * TODO:231:o: |-> Να ανακτηθεί η τρέχουσα τιμή των δεδομένων του
 * κόμβου TABLE_NODE2. Να τεθεί το (* TABLE_NODE2_val_ptr ) από το
 * rowreq_ctx->data
*/
    (* TABLE_NODE2_val_ptr ) = rowreq_ctx->data.TABLE_NODE2;

    return MFD_SUCCESS;
} /* TABLE_NODE2_get */


/** @} */
```

MFD_SKIP

Σε αυτό το σημείο αξίζει να γίνει μια αναφορά στις στήλες που για κάποιο λόγο ίσως χρειαστεί να μην υλοποιήσουμε (να μην επιστρέφουν δηλαδή κάποια τιμή) είτε γιατί τα δεδομένα δεν είναι διαθέσιμα, είτε γιατί το εκάστοτε λειτουργικό δεν επιστρέφει τις συγκεκριμένες παραμέτρους είτε γιατί αυτός που θα υλοποιήσει τον agent θεωρεί ασήμαντη την υλοποίηση κάποιων αντικειμένων. Για αυτήν την περίπτωση, η συνάρτηση get του συγκεκριμένου αντικειμένου πρέπει απλώς να επιστρέψει *MFD_SKIP*. Σαν παράδειγμα δίνεται η συνάρτηση get του αντικειμένου ifMtu από την ifMIB, που για κάποιο λόγο θέλουμε να μην υλοποιηθεί:

```
 int
ifMtu_get(ifTable_rowreq_ctx * rowreq_ctx, long *ifMtu_ptr)
{
    netsnmp_assert(NULL != ifMtu_ptr);

    netsnmp_assert(NULL != rowreq_ctx);

    if (rowreq_ctx->data.ifMtu == 0)
       return MFD_SKIP;

    (*ifMtu_ptr) = rowreq_ctx->data.ifMtu;

    return MFD_SUCCESS;
}
```

Στην περίπτωση αυτή, αν κατά την ‘data lookup’ φάση, μεταβιβαστεί στον row context μια στήλης η τιμή 0 για το αντικείμενο ifMtu, τότε σε αίτηση που ζητά επιστροφή της τιμής του αντικειμένου, το πρόγραμμα δεν θα επιστρέψει τίποτα στον χρήση.

ΣΤΑΔΙΑ ΔΙΑΔΙΚΑΣΙΑΣ ΥΛΟΠΟΙΗΣΗΣ ΤΟΥ AGENT

Συνοψίζοντας την διαδικασία παραγωγής κώδικα για πίνακα, παρατίθενται συνοπτικά τα σταδία που πρέπει να ακολουθήσει ο χρηστής για να υλοποιήσει μια MIB και να την προσαρτήσει στον ήδη υπάρχον agent.

1. Θέτει την MIB σαν μεταβλητή περιβάλλοντος και την δημοσιοποιεί.

> *export MIBS=+MYMIB-MIB*

2. Παράγει κώδικα C καλώντας:
   > *mib2c –mib2c.mfd.conf TABLE*
   Όπου TABLE το όνομα του πίνακα όπως ορίζεται στην MIB.
   Σε αυτο το σταδιο πρεπει να εχουν παραχθει τα εξης αρχεια:

```
default-table-TABLE.m2d
TABLE-README-FIRST.txt
TABLE-README-TABLE.txt
TABLE.c
TABLE.h
TABLE_data_access.c
TABLE_data_access.h
TABLE_data_get.c
TABLE_data_get.h
TABLE_data_set.c
TABLE_data_set.h
TABLE_enums.h
TABLE_interface.c
TABLE_interface.h
TABLE_oids.h
TABLE_Makefile
TABLE_subagent.c
```

3. Υλοποιείται ο *netsnmp_container* για πρόσβαση στα δεδομένα.
4. Υλοποιούνται οι συναρτήσεις get για επιστροφή των δεδομένων.
5. Γίνεται αντιγραφή των αρχείων στο /.../agent/mibgroup/ ή σε κάποιο sub-directory όπως περιγράφεται πιο πάνω και εκτελούνται οι εντολές *configure*,*make* και *make install*.
6. Ξεκινούμε το agent δίνοντας :

> ***snmpd*** *[OPTIONS] [LISTENING ADDRESSES]*

7. Γίνονται οι snmpget,snmpgetnext και snmpwalk αιτήσεις στον agent.

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΣΥΝΑΡΤΗΣΕΩΝ

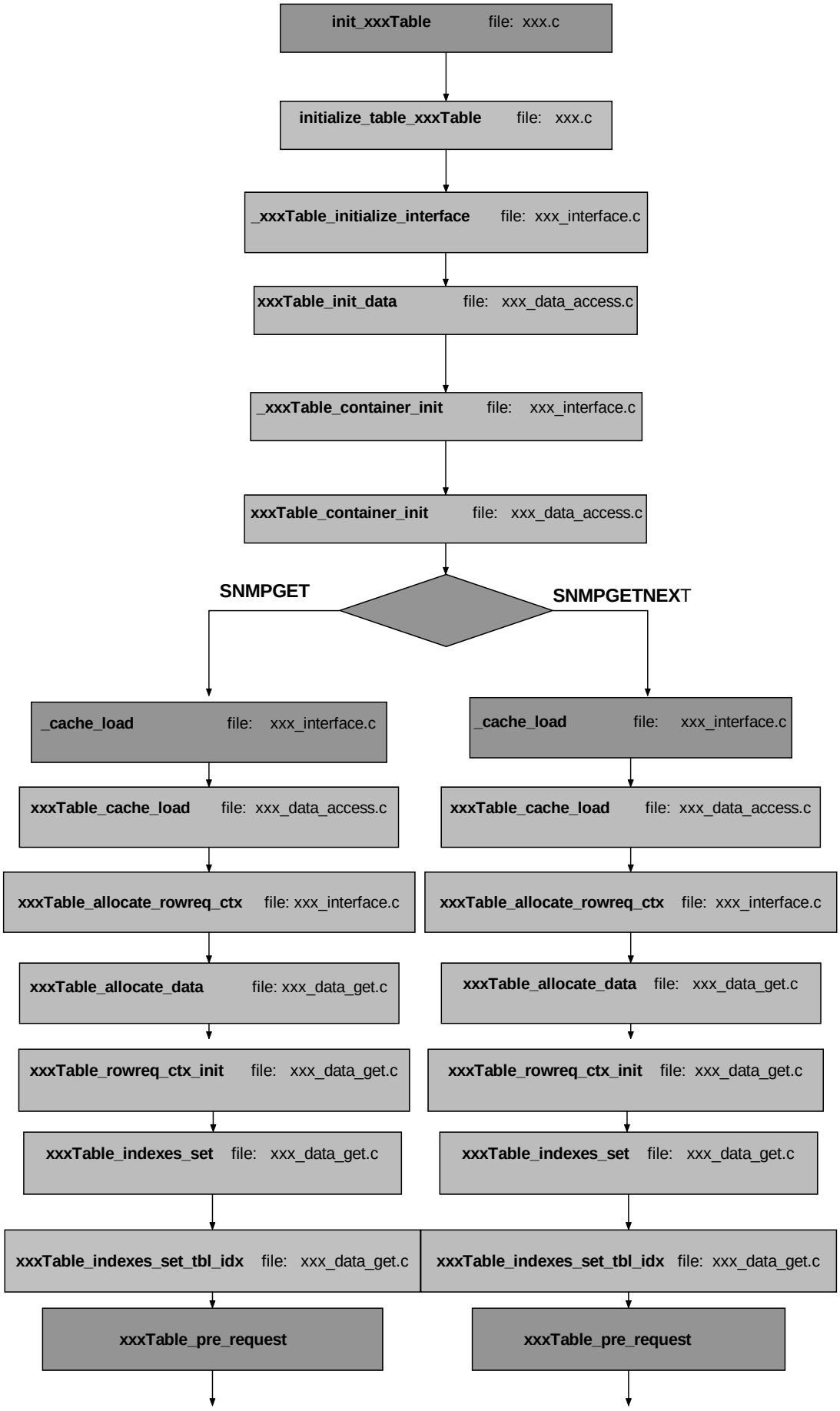
Στη συνεχεία παρουσιάζεται το διάγραμμα ροής των συναρτήσεων όπως αυτές καλούνται από τον agent τόσο κατά την εκτέλεση μιας snmpget όσο και μιας snmpgetnext-snmpwalk αίτησης. Η ροή αυτή των συναρτήσεων μπορεί να φανεί για τον εκάστοτε agent χρησιμοποιώντας τα debug tokens. Για παράδειγμα για τον πίνακα sixtofourRelayTable χρησιμοποιήθηκαν τα πιο κάτω debug tokens:

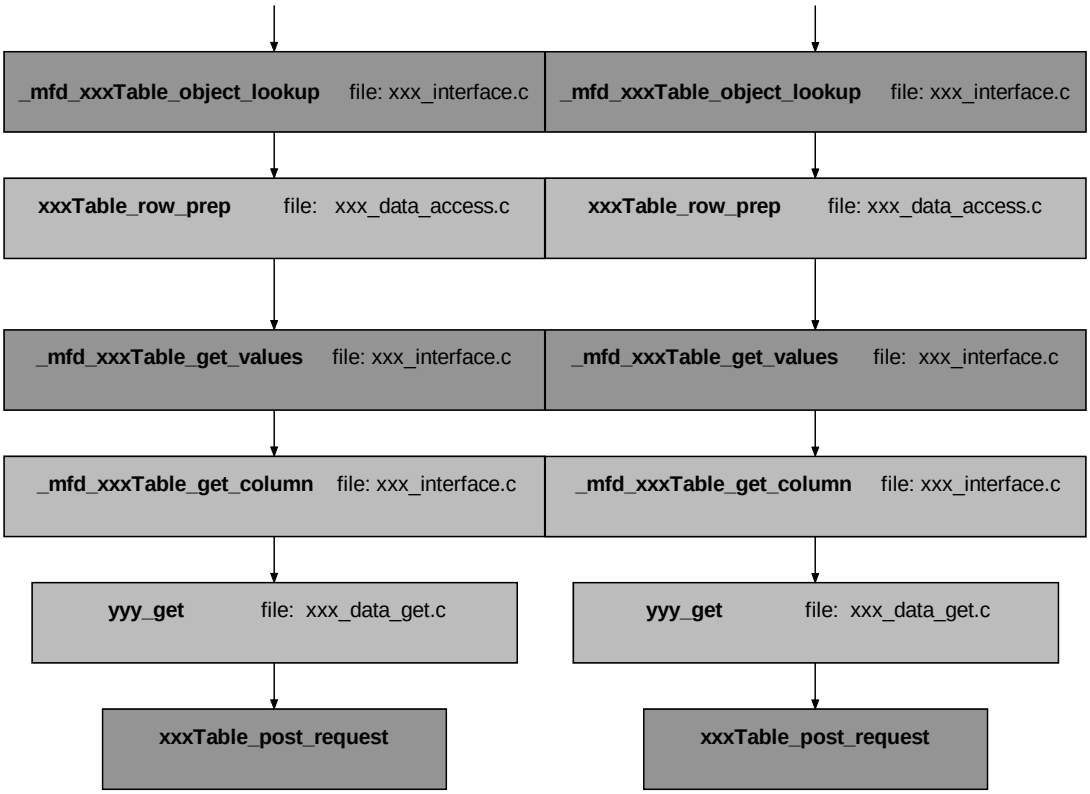
```
snmp_agent
helper:table:req
sixtofourRelayTable
verbose:sixtofourRelayTable
internal:sixtofourRelayTable
```

και δοθηκε για εναρξη του agent η πιο κατω εντολη:

```
> snmpd -f -Le -DsixtofourRelayTable,
verbose:sixtofourRelayTable,internal:sixtofourRelayTable
```

Πιο κάτω φαίνεται το διάγραμμα ροής των συναρτήσεων όπως καλούνται από τον agent. Όπου xxxTable το όνομα του εκάστοτε πίνακα.


init_xxxTable file: xxx.c
initialize_table_xxxTable file: xxx.c
_xxxTable_initialize_interface file: xxx_interface.c
xxxTable_init_data file: xxx_data_access.c
_xxxTable_container_init file: xxx_interface.c
xxxTable_container_init file: xxx_data_access.c
SNMPGET
SNMPGETNEXT
_cache_load file: xxx_interface.c
_cache_load file: xxx_interface.c
xxxTable_cache_load file: xxx_data_access.c
xxxTable_cache_load file: xxx_data_access.c
xxxTable_allocate_rowreq_ctx file: xxx_interface.c
xxxTable_allocate_rowreq_ctx file: xxx_interface.c
xxxTable_allocate_data file: xxx_data_get.c
xxxTable_allocate_data file: xxx_data_get.c
xxxTable_rowreq_ctx_init file: xxx_data_get.c
xxxTable_rowreq_ctx_init file: xxx_data_get.c
xxxTable_indexes_set file: xxx_data_get.c
xxxTable_indexes_set file: xxx_data_get.c
xxxTable_indexes_set_tbl_idx file: xxx_data_get.c
xxxTable_indexes_set_tbl_idx file: xxx_data_get.c
xxxTable_pre_request
xxxTable_pre_request



Σχήμα 32: Διάγραμμα ροής των συναρτήσεων που καλει ο agent

ΠΑΡΑΡΤΗΜΑ Β

ΥΛΟΠΟΙΗΣΗ ΤΟΥ 6TO4 ΜΗΧΑΝΙΣΜΟΥ ΣΕ LINUX

Ο 6to4 μηχανισμός είναι ένας μηχανισμός αυτόματου τούνελ που επιτρέπει σε απομακρυσμένα IPv6 site, στα οποία δεν προσφέρεται από τους παρόχους ISP's IPv6 σύνδεση, να επικοινωνούν μεταξύ τους καθώς και με το native IPv6 δίκτυο. Ο μηχανισμός αυτός, υλοποιείται κυρίως στους συνοριακούς δρομολογητές (6to4 routers), αλλά μπορεί να υλοποιηθεί και σε κάθε host που ανήκει σε ένα IPv4 δίκτυο και διαθέτει μια τουλάχιστον μοναδική παγκόσμια διεύθυνση.

Τα πιο κάτω είναι απαραίτητα για την υλοποίηση του μηχανισμού :

1. Το σύστημα μας να υποστηρίζει IPv6 και να υπάρχει εγκατεστημένη κάποια IPv6-IPv4 (sit) tunnel device, μεσώ της οποίας θα γίνεται η ενθυλάκωση των πακέτων, μέσα την IPv4 επικεφαλίδα.

2. Μια παγκόσμια μοναδική IPv4 διεύθυνση, μέσω της οποίας τα IPv6 πακέτα θα μεταδίδονται πάνω από το ήδη υπάρχον IPv4 δίκτυο(ίντερνετ) και από την οποία θα προκύψει η 6to4 διεύθυνση του site. Συγκεκριμένα, η 6to4 διεύθυνση σχηματίζεται από το πρόθεμα 2002::/16 και τα 4 bytes της IPv4 διεύθυνσης, παίρνοντας τελικά την μορφή 2002:V4ADDR::/48. Στην περίπτωση ενός IPv4 χρηστή που υλοποιεί 6to4, συνήθως η διεύθυνση που χρησιμοποιεί είναι η 2002:V4ADDR::1 (αυτήν την υλοποίηση χρησιμοποιούν και τα LINUX).

3. Η διεύθυνση κάποιου/ων relay routers, μεσώ των οποίων γίνεται η επικοινωνία με το native IPv6 domain. Εναλλακτικά, μπορεί να γίνει χρήση της anycast διεύθυνσης *2002:c058:6301::* (ή *192.88.99.1* η IPv4 αντίστοιχη της, ανάλογα με τις απαιτήσεις του κάθε λειτουργικού συστήματος) όπως καθορίζεται στο RFC 3068, όπου κάθε φορά που υπάρχει μετάδοση προς native IPv6 προορισμό, χρησιμοποιείται ο πλησιέστερος, προς το 6to4 site, διαθέσιμος relay router. Πιο κάτω δίνονται οι IPv4 διευθύνσεις μερικών γνωστών Relay Routers, που μπορεί να χρησιμοποιηθούν αντί της anycast διευθύνσεως:

Όνομα	IPv4 διεύθυνση	Τοποθεσία
gate.ipv6.uni-leipzig.de	139.18.25.33	Leipzig, Germany
6to4.ipv6.fh-regensburg.de	194.95.108.191	Regensburg, Germany
6to4.ipv6.microsoft.com	131.107.33.60	Redmond, North America
ipv6-lab-gw.cisco.com	128.107.240.254	San Jose, North America
kddilab.6to4.jp	202.255.45.5	Tokyo, Japan
6to4.ipv6.ascc.net	140.109.1.6	Taipei, Taiwan
6to4.ipv6.bt.com	193.113.58.75	Adastral Park, UK

Σχήμα 33: Πίνακας γνωστών Relay Routers

Τα σταδία της υλοποίησης του 6to4 μηχανισμού για λειτουργικό σύστημα LINUX δίνονται πιο κάτω:

1. Εγκατάσταση στο σύστημα του IPv6 πρωτοκόλλου, υλοποίηση δηλαδή dual stack, ούτως ώστε το σύστημα να αναγνωρίζει και να προωθεί τα 6to4 πακέτα και να παρέχει τον μηχανισμό για την ενθυλάκωση των πακέτων.

   Αν ο kernel υποστηρίζει IPv6 το πιο κάτω αρχείο πρέπει να υπάρχει στο */proc*-file-system (στα Suse Linux πρέπει να υπάρχει στο */proc/net/*)

```
/proc/net/if_inet6
```

   Ένας έλεγχος του κατά πόσο ο kernel υποστηρίζει IPv6, αν υπάρχει δηλαδή το πιο πάνω αρχείο, δίνεται πιο κάτω:

```
# test -f /proc/net/if_inet6 && echo "Running kernel is IPv6 ready"
```

   Αν δεν επιστραφεί το μήνυμα *"Running kernel is IPv6 ready"*, τότε μάλλον ο kernel δεν υποστηρίζει IPv6.

   Εγκατάσταση-Ενεργοποίηση (loading) του IPv6 module

   Εκτελείται:

```
# modprobe ipv6
```

   Και με την πιο κάτω εντολή ελέγχεται αν έχει ‘φορτωθεί’ το IPv6 module:

```
# lsmod |grep -w 'ipv6' && echo "IPv6 module successfully loaded"
```

Αυτόματη Εγκατάσταση-Ενεργοποίηση (loading) του module

Είναι δυνατό το IPv6 module να φορτώνει αυτόματα, με προσθήκη της πιο κάτω γραμμής στο configuration file του kernel module loader (συνήθως */etc/modules.conf* ή */etc/conf.modules*):

```
alias net-pf-10 ipv6
```

Το αυτόματο φόρτωμα του IPv6 module απενεργοποιείται με:

```
alias net-pf-10 off
```

2. Πρέπει να οριστεί το 6to4 interface και να τεθούν οι διάφορες παράμετροι του. Συγκεκριμένα πρέπει να γίνει configure το 6to4 pseudo-interface στη μοναδική παγκόσμια IPv4 διεύθυνση και να σχηματιστεί βάση της προηγούμενης η 6to4 διεύθυνση, μεσώ της οποίας το site θα είναι γνωστό στο υπόλοιπο internet, μέσα από τις εγγραφές DNS. Και τέλος να καθοριστούν οι παράμετροι που θα χρησιμοποιούν τα ενθυλακωμένα πακέτα κατά την μετάδοση τους όπως το TTL(time to live) και το MTU(maximum transfer unit).

Δημιουργείται αρχικά η tunnel device:

```
# /sbin/ip tunnel add tun6to4 mode sit ttl <ttldefault> remote any
local <localipv4address>
```

όπου *<ttldefault>* τίθεται το TTL για την εξωτερική IPv4 επικεφαλίδα και *<localipv4address>* η παγκόσμια unicast διεύθυνση που διαθέτουμε, και στην οποία θα γίνει configure το 6to4 interface. Με αυτόν τον τρόπο δημιουργείται το 6to4 interface με όνομα *tun6to4*. Για διεύθυνση του άκρου εξόδου του tunnel δίνεται η τιμή *any*, αφού ο 6to4 είναι μηχανισμός αυτόματου tunneling, και το άκρο του tunnel καθορίζεται δυναμικά.

Ακολούθως, "σηκώνεται" το 6to4 interface:

```
# /sbin/ip link set dev tun6to4 up
```

Και του ανατίθεται η τοπική 6to4 διεύθυνση:

```
# /sbin/ip -6 addr add <local6to4address>/16 dev tun6to4
```

Στην περίπτωση αυτή το πρόθεμα του δικτύου τίθεται σε */16*, έτσι ώστε στον πίνακα δρομολόγησης να οριστεί αυτόματα το route 2002::/16 μέσω του 6to4 interface και να μην γίνεται χρήση κάποιου relay. Εναλλακτικά στην περίπτωση του 6to4 router, το πρόθεμα τίθεται σε */48* και επιπλέον μέσω της εντολής

```
# /sbin/ip -6 route add 2002::/16 dev tun6to4
```

τα πακέτα που προορίζονται προς 6to4 προορισμούς δρομολογούνται μέσω του 6to4 interface κατευθείαν στον 6to4 router προορισμού.

Γενικά, σε συστήματα UNIX, για τον υπολογισμό της 6to4 διευθύνσεως από την IPv4 είναι χρήσιμη η πιο κάτω εντολή (όπου *1.2.3.4* η IPv4 διεύθυνση που διαθέτουμε):

```
ipv4="1.2.3.4"; printf "2002:%02x%02x:%02x%02x::1" `echo $ipv4 | tr
"." " "`
```

3. Πρέπει να οριστούν τα διάφορα routes μεσώ των οποίων θα επικοινωνεί ο 6to4 κόμβος με τα native IPv6 sites (οι διάφοροι δηλαδή relay routers που θα χρησιμοποιεί στην κάθε περίπτωση).

   Στην πιο κάτω παράδειγμα καθορίζεται σαν default route η anycast διεύθυνση για τον πλησιέστερο relay router:

```
# /sbin/ip -6 route add default via ::192.88.99.1 dev tun6to4 metric
1
```

   Σε μερικές εκδόσεις του εργαλείου "ip" δεν υποστηρίζονται οι IPv4-compatible IPv6 διευθύνσεις για gateways, και έτσι σε αυτές τις περιπτώσεις χρησιμοποιείται η IPv6 διεύθυνση:

```
# /sbin/ip -6 route add default via 2002:c058:6301::1 dev tun6to4
metric 1
```

4. (προαιρετικό, μόνο για 6to4 routers) Να καθοριστούν τα router advertisements, να επιτραπεί από το συγκεκριμένο μηχάνημα η προώθηση πακέτων και να εκκινήσει ο router advertisement daemon. Είναι σημαντικό ότι δεν χρειάζεται η εγκατάσταση 6to4 μηχανισμού στον κάθε host, αφού το υπόλοιπο site συμπεριφέρεται σαν native IPv6 και η διαδικασία της ενθυλάκωσης γίνεται μόνο στον 6to4 router σε περίπτωση επικοινωνίας με αλλά sites. Έτσι, οι διάφοροι hosts που βρίσκονται κάτω από το router, σχηματίζουν από μόνοι τους την IPv6 διεύθυνση τους μεσώ της address autoconfiguration που παρέχει το IPv6 πρωτόκολλο, βάση των διαφημίσεων(advertisements) που στέλνει περιοδικά ο 6to4 router, που περιέχουν τα 64 bits 2002::V4ADDR:SubnetID::/64 που περιλαμβάνουν την

IPv4 διεύθυνση και το subnet ID (τον χαρακτηριστικό αριθμό του εκάστοτε υποδικτύου).

Αρχικά, ελέγχεται αν το μηχάνημα προωθεί πακέτα, είναι δηλαδή δρομολογητής:

```
# cat /proc/sys/net/ipv6/conf/forwarding
```

Αν επιστραφεί η τιμή ***1***, τότε το μηχάνημα πρόκειται για δρομολογητή.
Αν όχι, πρέπει μέσω της πιο κάτω εντολής να επιτραπεί στο συγκεκριμένο μηχάνημα να προωθεί πακέτα:

```
# echo 1 > /proc/sys/net/ipv6/conf/forwarding
```

Ακολούθως, καθορίζονται οι πληροφορίες που θα στέλνει μέσω των διαφημίσεων ο router advertisement daemon, που συνήθως είναι:

- Πρόθεμα (prefix)
- Η διάρκεια ζωής του προθέματος.
- Η συχνότητα των διαφημίσεων (προαιρετικό)

Μετά από σωστό configuration, ο δαίμονας στέλνει διαφημίσεις μέσω των interfaces που καθορίστηκαν τις οποίες λαμβάνουν οι clients και καθορίζουν με stateless address autoconfiguration τις διευθύνσεις τους, από το πρόθεμα των διαφημίσεων, και το default route.

Στα Linux χρησιμοποιείται ο radvd δαίμονας, και το config αρχείο του είναι συνήθως το */etc/radvd.conf*. Πιο κάτω δίνεται το *radvd.conf* για την περίπτωση ενός δυναμικού dial-on-demand Linux router, όπου ο radvd στέλνει κάθε φορά το 6to4 πρόθεμα του υποδικτύου (2002:V4ADDR:subnetID::/64) που σχηματίζεται από την δυναμική IPv4 διεύθυνση (V4ADDR).

```
interface eth0 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 0:0:0:f101::/64 {
                AdvOnLink off;
                AdvAutonomous on;
                AdvRouterAddr on;
                Base6to4Interface ppp0;
                AdvPreferredLifetime 20;
                AdvValidLifetime 30;
        };
};
```

Τα αποτελέσματα στην πλευρά του client (υποθέτουμε ότι στο ppp0 έχει ανατεθεί δυναμική IPv4 διεύθυνση *1.2.3.4*)

```
# /sbin/ip -6 addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
qlen 100
   inet6 2002:0102:0304:f101:2e0:12ff:fe34:1234/64 scope global
dynamic
      valid_lft 22sec preferred_lft 12sec
   inet6 fe80::2e0:12ff:fe34:1234/10 scope link
```

Τελος, πρέπει να καθοριστεί στον πίνακα δρομολόγησης του 6to4 router, το route προς το συγκεκριμένο υποδίκτυο (το οποίο πρέπει κάθε φορά να αλλάζει όταν ανατίθεται νέα IPv4 διεύθυνση στο dial-up interface) :

```
# /sbin/ip -6 route add 2002:0102:0304:f101::/64 dev eth0
metric 1
```

ΠΑΡΑΡΤΗΜΑ Γ

ΚΩΔΙΚΑΣ ΤΟΥ AGENT ΓΙΑ ΤΗΝ SIXTOFOUR-MIB

Ακολουθεί ο κώδικας των mib modules, τα οποία ενσωματώθηκαν στον agent, για την υλοποίηση της sixtofour-MIB. Ο κώδικας είναι γραμμένος σε γλώσσα προγραμματισμού C. Για την παραγωγή του αρχικού σκελετού του κώδικα έγινε χρήση του εργαλείου *mib2c* που παρέχει το NET-SNMP και η όλη διαδικασία περιγράφεται στο παράρτημα Α.
Στην συνεχεία παρατίθενται μόνο τα κομμάτια του κώδικα στα οποία έγιναν οι προσθήκες και οι τροποποιήσεις.

Πίνακας *sixtofourIfTable*

sixtofourIfTable_data_access.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *       version : 1.12 $ of : mfd-data-access.m2c
 *
 */
/* standard Net-SNMP includes */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>

/* include our parent header */
#include "sixtofourIfTable.h"
#include "stf_functions.h"   //δίνεται πιο κάτω

#include "sixtofourIfTable_data_access.h"
char sixtofourinterfaceindex[10][30];



/*****************************************************************
 * Συνάρτηση για υπολογισμό της IPv4 διευθύνσεως του 6to4 interface*
 * από την δεκαεξαδική μορφή στην οποία δίνεται στην κανονική της  *
 * δεκαδική                                                        *
 *****************************************************************/

int ipv4_ipv4dot(char ipv4[])
```

```
        {
        static  char *line = NULL;
        static  size_t len = 0;
        static FILE  *filep;
        static char addr[4][3];
        static char address[9], unused[5];
        static int ip4[4],i;
        int  flag=0;

         filep = fopen("/proc/net/if_inet6", "r");
                 if (NULL == filep)
                       return MFD_RESOURCE_UNAVAILABLE;

         while (  getline(&line,&len,filep) !=-1)
                {
                if (strncmp(line,"2002",4)==0) //eleghos an yparxei
6to4 interface
                        {
                        flag=1;
                        sscanf(line,"%4s%8s",unused,address);

sscanf(address,"%2s%2s%2s%2s",addr[0],addr[1],addr[2],addr[3]);
                        for(i=0;i<4;i++)
                        ip4[i]=hextoint(addr[i]);

sprintf(ipv4,"%d.%d.%d.%d\0",ip4[0],ip4[1],ip4[2],ip4[3]);
                        break;
                        }
                 }
        return flag;
}
/*********************************************************************/



/** @defgroup data_access data_access: Routines to access data
 *
 * These routines are used to locate the data used to satisfy
 * requests.
 *
 * @{
 */
/*********************************************************************

******************** Table sixtofourIfTable **********************/

/*
 * sixtofourIfTable is subid 3 of sixtofour.
 * Its status is Current.
 * OID: .1.3.6.1.3.19.1.1.3, length: 9
*/

/**
 * initialization for sixtofourIfTable data access
 *
 * This function is called during startup to allow you to
 * allocate any resources you need for the data table.
 *
 * @param sixtofourIfTable_reg
 *        Pointer to sixtofourIfTable_registration
```

```
 *
 * @retval MFD_SUCCESS : success.
 * @retval MFD_ERROR   : unrecoverable error.
 */
int
sixtofourIfTable_init_data(sixtofourIfTable_registration_ptr
sixtofourIfTable_reg)
{

DEBUGMSGTL(("verbose:sixtofourIfTable:sixtofourIfTable_init_data","ca
lled\n"));

        return MFD_SUCCESS;
} /* sixtofourIfTable_init_data */


/**********************************************************************
 *
 * cache
 *
 **********************************************************************/
/**
 * container initialization
 *
 * @param container_ptr_ptr A pointer to a container pointer. If you
 *        create a custom container, use this parameter to return it
 *        to the MFD helper. If set to NULL, the MFD helper will
 *        allocate a container for you.
 * @param  cache A pointer to a cache structure. You can set the
 *         timeout and other cache flags using this pointer.
 *
 * This function is called at startup to allow you to customize
 * certain
 * aspects of the access method. For the most part, it is for
 * advanced
 * users. The default code should suffice for most cases. If no
 * custom
 *  container is allocated, the MFD code will create one for your.
 *
 *  This is also the place to set up cache behavior. The default, to
 *  simply set the cache timeout, will work well with the default
 *  container. If you are using a custom container, you may want to
 *  look at the cache helper documentation to see if there are any
 *  flags you want to set.
 *
 * @remark
 *  This would also be a good place to do any initialization needed
 *  for you data source. For example, opening a connection to another
 *  process that will supply the data, opening a database, etc.
 */

void
sixtofourIfTable_container_init(netsnmp_container
**container_ptr_ptr,
                        netsnmp_cache *cache)
{

DEBUGMSGTL(("verbose:sixtofourIfTable:sixtofourIfTable_container_init
","called\n"));

    if((NULL == cache) || (NULL == container_ptr_ptr)) {
```

```
        snmp_log(LOG_ERR,"bad params to
sixtofourIfTable_container_init\n");
        return;
    }

    /*
     * For advanced users, you can use a custom container. If you
     * do not create one, one will be created for you.
     */
//δημιουργείται αυτόματα ο container
    *container_ptr_ptr = NULL;


    cache->timeout = SIXTOFOURIFTABLE_CACHE_TIMEOUT;  /* seconds */
} /* sixtofourIfTable_container_init */


/**
 * load cache data
 *
 * Implement sixtofourIfTable cache load
 *
 * @param container container to which items should be inserted
 *
 * @retval MFD_SUCCESS              : success.
 * @retval MFD_RESOURCE_UNAVAILABLE : Can't access data source
 * @retval MFD_ERROR                : other error.
 *
 *  This function is called to cache the index(es)and data
 *  for the every row in the data set.
 *
 * @remark
 *  While loading the cache, the only important thing is the indexes.
 *  If access to your data is cheap/fast it would make sense to
 *  update the data here.
 * @note
 *  If you need consistency between rows (like you want statistics
 *  for each row to be from the same time frame), you should set all
 *  data here.
 *
 */

int
sixtofourIfTable_cache_load(netsnmp_container *container)
{
    sixtofourIfTable_rowreq_ctx *rowreq_ctx;
    size_t                 count = 0;

    /*
     * this code is based on a data source that is a
     * text file to be read and parsed.
     */

    FILE *filep,*filep_check;
    char line[MAX_LINE_SIZE];
    int index_count=0;          //δείκτης για τον επαναπροσδιορισμό
                                  της σωστής κατάταξης κατά την
                                   αφαίρεση στοιχείου από τον
    int rc = MFD_SUCCESS;
    int if_flag=0;              //σημαία που υποδηλώνει την ύπαρξη
                                  του 6to4 interface
```

```
    char *line_check = NULL;
    size_t len = 0;
    char temp[30];
    char devname[30];           //όνομα του 6to4 interface


    /*
     * temporary storage for index values
     */
        /*
         *
sixtofourIfIndex(1)/INTEGER32/ASN_INTEGER/long(long)//l/a/w/e/r/d/h
         */
   long   sixtofourIfIndex;


DEBUGMSGTL(("verbose:sixtofourIfTable:sixtofourIfTable_cache_load","c
alled\n"));

//ΕΛΕΓΧΟΣ ΑΝ ΥΠΑΡΧΕΙ ΣΤΟ ΣΥΣΤΗΜΑ 6TO4 INTERFACE
 filep_check = fopen("/proc/net/if_inet6", "r");
 if (NULL == filep_check)
            return MFD_RESOURCE_UNAVAILABLE;

 while (  getline(&line_check,&len,filep_check) !=-1)
        {
        if (strncmp(line_check,"2002",4)==0)
                {
                if_flag=1;
                sscanf(line_check,"%s %s %s %s %s
%s",devname,devname,devname,devname,devname,devname);
                break;
                }
        }
        fclose(filep_check);


   /*************************************************************
    ***        ΑΝ ΥΠΑΡΧΕΙ ΣΤΟ ΣΥΣΤΗΜΑ 6TO4 INTERFACE           ***
    ***-----------------------------------------------------------***/



 if (if_flag==1)
    {

    /*
     * ανοίγει το κυρίως file.
     */

    index_count=0;
    filep = fopen("/proc/net/if_inet6", "r");
    if (NULL == filep)
            return MFD_RESOURCE_UNAVAILABLE;



    /*
     * Load/update data in the sixtofourIfTable container.
     * loop over your sixtofourIfTable data, allocate a rowreq
     * context, set the index(es) and data and insert
```

```
 * into the container.
 */

while( 1 )   //μέχρι να συμπληρωθούν όλες οι γραμμές του πίνακα
  {
    static int sixtofourIfStatus=1;
    static int sixtofourIfMIBIndex=0;
    static char *line2;
    static size_t len2 = 0;
    static char temp[50]="";
    static char filename3[50]="";
    static char addr4[4][3];
    static char address[9], unused[5];
    static char ipv4[20];
    static int ip4[4],i;
    static in_addr_t sixtofourIfLocalIpv4Address;
    static char stfaddr[33],ip6addr[40];
    static char addr[8][5];
    struct in6_addr sixtofourIfLocal6to4address;
    static size_t sixtofourIfLocal6to4address_len;
    static int sixtofourIfLocal6to4AddrPfxLength,
               sixtofourIfMTU=0;
    static char prefix[3];
    static char *line3 = NULL;
    static size_t len3 = 0;
    FILE  *filep3,*filep2;
    static int check,sixtofourIfHopLimit;
    char temp2[40],temp3[5];
    char command[100];
    static char *line4 = NULL;
    static size_t len4 = 0;
    static char filename[70];
    FILE  *filep4;
    int sixtofourIfIpSec=0;
    char phys_if[30],*line5 = NULL;
    size_t len5 = 0;
    FILE  *filep5;
    unsigned char *u;
    int sockfd, size = 1;
    struct ifreq *ifr;
    struct sockaddr_in sa;
    char ipadr[14],stf_addr[14];



/*
 * παίρνει μια γραμμή (skip blank lines)
 */

   do
            {
            if (!fgets(line, sizeof(line), filep))
                    {
                    /* we're done */
                    fclose(filep);
                    filep = NULL;
                    }

                    if (strncmp(line,"2002",4)!=0)
                          {
                           strcpy(line,"\n");
```

```
                                continue;
                               }


                } while (filep && (line[0] == '\n'));



    /*
     * check for end of data
     */
    if(NULL == filep)
        break;



        /*
         * Αν συνάντησε το ίδιο interface προηγουμένως, να
         * χρησιμοποιήσει τον ίδιο δείκτη. Ειδάλλως, να βρει ένα
         * αχρησιμοποίητο δείκτη και να τον χρησιμοποιήσει
         */

        index_count++;

//η συνδεδεμένη λίστα "sixtofourIfTable" ορίστηκε στο
sixtofourIfTable.h σαν ο user conext μας

        sixtofourIfIndex = se_find_value_in_slist("sixtofourIfTable",
line);

//ΠΕΡΙΠΤΩΣΗ: ΔΕΝ ΥΠΑΡΧΕΙ ΣΤΗ ΛΙΣΤΑ
        if (sixtofourIfIndex == SE_DNE)      //αν ο δείκτης δεν
                                               υπάρχει στην λίστα
                {
                sixtofourIfIndex =
se_find_free_value_in_slist("sixtofourIfTable");   //αν βρει
                                    επιστρέφει το max_value+1

                if (sixtofourIfIndex == SE_DNE)
                           sixtofourIfIndex = 1;  //Νέα λιστα!

                //περίπτωση που έχει πριν αφαιρεθεί αντικείμενο από
                  τη λίστα και τώρα προστίθεται νέο
                if (sixtofourIfIndex != index_count)
                        sixtofourIfIndex=index_count;

                se_add_pair_to_slist("sixtofourIfTable",strdup(line),
sixtofourIfIndex);

                DEBUGMSGTL(("sixtofourIfTable:sixtofourIfIndex", "new
sixtofourIfIndex%d for %s\n",sixtofourIfIndex, line));
                }

//ΠΕΡΙΠΤΩΣΗ: ΥΠΑΡΧΕΙ ΣΤΗΝ ΛΙΣΤΑ ΑΛΛΑ ΕΧΕΙ ΣΒΗΣΤΕΙ ΠΡΟΗΓΟΥΜΕΝΟ
  ΑΝΤΙΚΕΙΜΕΝΟ
        else  if (sixtofourIfIndex != index_count)
                {
                        sixtofourIfIndex=index_count;

se_add_pair_to_slist("sixtofourIfTable",strdup(line),
sixtofourIfIndex);
```

```
                }

         if ((NULL == line) || (0 == sixtofourIfIndex))
                        {
                        rc = MFD_END_OF_DATA;
                        break;
                        }




        /*
         * set indexes in new sixtofourIfTable rowreq context.
         */

        rowreq_ctx = sixtofourIfTable_allocate_rowreq_ctx();
        if (NULL == rowreq_ctx) {
            snmp_log(LOG_ERR, "memory allocation failed\n");
            return MFD_RESOURCE_UNAVAILABLE;
        }
        if(MFD_SUCCESS != sixtofourIfTable_indexes_set(rowreq_ctx
                               , sixtofourIfIndex
               )) {
            snmp_log(LOG_ERR,"error setting index while loading "
                     "sixtofourIfTable cache.\n");
            sixtofourIfTable_release_rowreq_ctx(rowreq_ctx);
            continue;
        }

        /*
         * populate sixtofourIfTable data context.
         */

/*παίρνει τα δεδομένα στη data lookup φάση*/

/******αρχικά κάνει parsing τις εγγραφές της line ***********/

        sscanf(line,"%s %s %s %s %s %s",devname,devname,prefix,
devname,devname,devname);


/**υπολογίζει την τιμή που θα στείλει στο sixtofourRelayIfIndex**/

strcpy(sixtofourinterfaceindex[sixtofourIfIndex],devname);


/************υπολογισμός του sixtofourIfMIBIndex***************/

                strcpy(filename3,"/proc/net/dev_snmp6/");
                strcat(filename3,devname);
                filep2 = fopen(filename3, "r");
                if (NULL == filep2)
                          return MFD_RESOURCE_UNAVAILABLE;
                if( getline(&line2,&len2,filep2) !=-1)
                    {
                    sscanf(line2,"%s  %d",temp,&sixtofourIfMIBIndex);
                    }

                fclose(filep2);
```

```
/*****************************************************************/


/***********υπολογισμός του sixtofourIfLocalIpv4Address************/

                                sscanf(line,"%4s%8s",unused,address);

sscanf(address,"%2s%2s%2s%2s",addr4[0],addr4[1],addr4[2],addr4[3]);
                                for(i=0;i<4;i++)
                                ip4[i]=hextoint(addr4[i]);

sprintf(ipv4,"%d.%d.%d.%d\0",ip4[0],ip4[1],ip4[2],ip4[3]);

sixtofourIfLocalIpv4Address=inet_addr(ipv4);
/*****************************************************************/


/**********υπολογισμός του sixtofourIfLocal6to4Address*************/

                                sscanf(line,"%s",stfaddr);

sscanf(stfaddr,"%4s%4s%4s%4s%4s%4s%4s%4s",addr[0],addr[1],addr[2],add
r[3],addr[4],addr[5],addr[6],addr[7]);

sprintf(ip6addr,"%s:%s:%s:%s:%s:%s:%s:%s",addr[0],addr[1],addr[2
],addr[3],addr[4],addr[5],addr[6],addr[7]);

inet_pton(AF_INET6,ip6addr,&sixtofourIfLocal6to4address);

sixtofourIfLocal6to4address_len=sizeof(sixtofourIfLocal6to4address);

/*****************************************************************/


/********υπολογισμός του sixtofourIfLocal6to4AddrPfxLength*********/

sixtofourIfLocal6to4AddrPfxLength=hextoint(prefix);

/*****************************************************************/


/************υπολογισμός του sixtofourIfHopLimit*****************/

                        strcpy(command,"/sbin/ip tunnel show  >
/usr/local/test ");
                        check=system(command);
                        strcpy(command,"chmod 777 /usr/local/test");
                        check=system(command);



                        filep3 = fopen("/usr/local/test", "r");
                        if (NULL == filep3)
                                return SNMP_ERR_RESOURCEUNAVAILABLE;
                        while (  getline(&line3,&len3,filep3) !=-1)
                                {
                                if (strstr(line3,devname)!=NULL)
                                        {puts(line3);
```

```
                                        sscanf(line3,"%s %s %s %s %s
%s %s %s",temp2,temp2,temp2
,temp2,temp2,temp2,temp3,temp2);
          //περίπτωση που το 6to4 interface είναι πάνω από physical
            interface
                              if (strcmp(temp3,"dev")==0)
                                    sscanf(line3,"%s %s %s %s %s %s
%s %s %s
%s",temp2,temp2,temp2,temp2,temp2,temp2,temp2,temp2,temp2,temp2);


sixtofourIfHopLimit=atoi(temp2);
                                        break;
                                        }
                                  }

                         strcpy(command,"rm /usr/local/test");
                         check=system(command);
                         fclose(filep3);

/*****************************************************************/


/******************υπολογισμός του sixtofourIfMTU*****************/

        strcpy(filename,"/proc/sys/net/ipv6/conf/");

                strcat(filename,devname);
                strcat(filename,"/mtu");
                filep4 = fopen(filename, "r");
                if (NULL == filep4)
                          return SNMP_ERR_RESOURCEUNAVAILABLE;
                if( getline(&line4,&len4,filep4) !=-1)
                        {
                        sscanf(line4,"%d",&sixtofourIfMTU);

                        }
                    fclose(filep4);
/*****************************************************************/


/********υπολογισμός του sixtofourIfSecurity **********************/

        filep5 = fopen("/usr/local/ipsec_tncfg", "r");
        if (NULL == filep5)
                sixtofourIfIpSec=2;

        else
                {

                while (  getline(&line5,&len5,filep5) !=-1)
                        {
                        sscanf(line5,"%s %s
%s",phys_if,phys_if,phys_if);
                        if(strcmp(phys_if,"NULL")==0)
                                {
                                sixtofourIfIpSec=2;
                                break;
                                }

        //βρίσκει την ipv4 διεύθυνση του interface
```

```
                        if (0 > (sockfd = socket(AF_INET, SOCK_DGRAM,
IPPROTO_IP)))
                                return MFD_RESOURCE_UNAVAILABLE;

                        struct sockaddr ifa;
                        get_addr(sockfd, phys_if, &ifa,ipadr);
                        close(sockfd);

                        if (ipv4_ipv4dot(stf_addr)==0) continue;
                        else
                                {
              //την συγκρίνει με την ipv4 address του 6to4 interface
                                if (strcmp(stf_addr,ipadr)==0)
                                        {
                                        sixtofourIfIpSec=1;
                                        break;
                                        }
                                }
                        }
                  fclose(filep5);
                  } //κλείνει το else
```

```
    /*
     * setup/save data for sixtofourIfStatus
     *
sixtofourIfStatus(2)/INTEGER/ASN_INTEGER/long(u_long)//l/A/w/E/r/d/h
     */
    /*
     * Define sixtofourIfStatus mapping.
     * Είναι αντικείμενο τύπου INTEGER και χρειάζεται αντιστοίχηση
     */
    if(MFD_SUCCESS !=
       sixtofourIfStatus_map(&rowreq_ctx->data.sixtofourIfStatus,
sixtofourIfStatus )) {
        return MFD_ERROR;
    }


    /*
     * setup/save data for sixtofourIfMIBIndex
     *
sixtofourIfMIBIndex(3)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/
h
     * Integer based value can usually just do a direct copy.
     */
    rowreq_ctx->data.sixtofourIfMIBIndex = sixtofourIfMIBIndex;



    /*
     * setup/save data for sixtofourIfLocalIpv4Address
     *
sixtofourIfLocalIpv4Address(4)/IPADDR/ASN_IPADDRESS/u_long(u_long)//l
/A/w/e/r/d/h
```

```
     * Integer based value can usually just do a direct copy.
     */
    rowreq_ctx->data.sixtofourIfLocalIpv4Address =
sixtofourIfLocalIpv4Address;


    /*
     * setup/save data for sixtofourIfLocal6to4Address
     *
sixtofourIfLocal6to4Address(5)/InetAddressIPv6/ASN_OCTET_STR/char(cha
r)//L/A/w/e/R/d/H
     */


    /* Το μέγεθος της ipv6 διευθύνσεως είναι σταθερό και έχει δηλωθεί
     * και έτσι δεν χρειάζεται δέσμευση μνήμης
     */

    rowreq_ctx->data.sixtofourIfLocal6to4Address_len =
sixtofourIfLocal6to4address_len;
    memcpy( rowreq_ctx->data.sixtofourIfLocal6to4Address, (u_char
*)&sixtofourIfLocal6to4address,sizeof(sixtofourIfLocal6to4address));



    /*
     * setup/save data for sixtofourIfLocal6to4AddrPfxLength
     *
sixtofourIfLocal6to4AddrPfxLength(6)/INTEGER32/ASN_INTEGER/long(long)
//l/A/w/e/r/d/h
     */
    /*
     * Integer based value can usually just do a direct copy.
     */
    rowreq_ctx->data.sixtofourIfLocal6to4AddrPfxLength =
sixtofourIfLocal6to4AddrPfxLength;


    /*
     * setup/save data for sixtofourIfHopLimit
     *
sixtofourIfHopLimit(7)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/
h
     */
    /*
     * Integer based value can usually just do a direct copy.
     */
    rowreq_ctx->data.sixtofourIfHopLimit = sixtofourIfHopLimit;


    /*
     * setup/save data for sixtofourIfMTU
     *
sixtofourIfMTU(8)/INTEGER32/ASN_INTEGER/long(long)//l/A/W/e/r/d/h
     */
    /*
     * Integer based value can usually just do a direct copy.
     */
    rowreq_ctx->data.sixtofourIfMTU = sixtofourIfMTU;
```

```
    /*
     * setup/save data for sixtofourIfIpSec
     *
sixtofourIfIpSec(9)/INTEGER/ASN_INTEGER/long(u_long)//l/A/w/E/r/d/h
     */
    /*
     * Define sixtofourIfIpSec mapping.
     * Και εδώ χρειάζεται αντιστοίχηση.
    */
    if(MFD_SUCCESS !=
       sixtofourIfIpSec_map(&rowreq_ctx->data.sixtofourIfIpSec,
sixtofourIfIpSec )) {
        return MFD_ERROR;
    }


        /*
         * μπαίνουν όλα τα δεδομένα στον container
         */

        CONTAINER_INSERT(container, rowreq_ctx);

        ++count;
    }

    if(NULL != filep)
        fclose(filep);

DEBUGMSGT(("verbose:sixtofourIfTable:sixtofourIfTable_cache_load",
               "inserted %d records\n", count));


      return rc;

  }  //κλείνει το if (if_flag)

} /* sixtofourIfTable_cache_load */


/**
 * cache clean up
 *
 * @param container container with all current items

 *
 * @note
 *  The MFD helper will take care of releasing all the row contexts.
 *
 */
void
sixtofourIfTable_cache_free(netsnmp_container *container)
{

DEBUGMSGTL(("verbose:sixtofourIfTable:sixtofourIfTable_cache_free","c
alled\n"));

    /*
     * Free sixtofourIfTable cache.
     */
} /* sixtofourIfTable_cache_free */
```

```
/**
 * prepare row for processing.
 *
 * When the agent has located the row for a request, this function is
 * called to prepare the row for processing. If you fully populated
 * the data context during the index setup phase, you may not need to
 * do anything.
 *
 * @param rowreq_ctx pointer to a context.
 *
 * @retval MFD_SUCCESS     : success.
 * @retval MFD_ERROR       : other error.
 */
int
sixtofourIfTable_row_prep( sixtofourIfTable_rowreq_ctx *rowreq_ctx)
{

DEBUGMSGTL(("verbose:sixtofourIfTable:sixtofourIfTable_row_prep","cal
led\n"));

    netsnmp_assert(NULL != rowreq_ctx);

    return MFD_SUCCESS;
} /* sixtofourIfTable_row_prep */
```

Πίνακας *sixtofourRelayTable*

sixtofourRelayTable_data_access.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *       version : 1.12 $ of : mfd-data-access.m2c,v $
 *
 *
 */
/* standard Net-SNMP includes */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>

/* include our parent header */
#include "sixtofourRelayTable.h"
```

```
#include "stf_functions.h"

#include "sixtofourRelayTable_data_access.h"

extern char sixtofourinterfaceindex[10][30];  //για αντιστοίχηση με
                                              τον πίνακα sixtofourIfTable


/** @defgroup data_access data_access: Routines to access data
 *
 * These routines are used to locate the data used to satisfy
 * requests.
 *
 *
 */
/****************** Table sixtofourRelayTable ***********************

 ****************************************************************/
/*
 * sixtofourRelayTable is subid 4 of sixtofour.
 * Its status is Current.
 * OID: .1.3.6.1.3.19.1.1.4, length: 9
*/

/**
 * initialization for sixtofourRelayTable data access
 *
 * This function is called during startup to allow you to
 * allocate any resources you need for the data table.
 *
 * @param sixtofourRelayTable_reg
 *        Pointer to sixtofourRelayTable_registration
 *
 * @retval MFD_SUCCESS : success.
 * @retval MFD_ERROR   : unrecoverable error.
 */
int
sixtofourRelayTable_init_data(sixtofourRelayTable_registration_ptr
sixtofourRelayTable_reg)
{

DEBUGMSGTL(("verbose:sixtofourRelayTable:sixtofourRelayTable_init_dat
a","called\n"));

    return MFD_SUCCESS;
} /* sixtofourRelayTable_init_data */

/****************************************************************
 *
 * cache
 *
 ****************************************************************/
/**
 * container initialization
 *
 * @param container_ptr_ptr A pointer to a container pointer. If you
 *        create a custom container, use this parameter to return it
 *        to the MFD helper. If set to NULL, the MFD helper will
 *        allocate a container for you.
 * @param  cache A pointer to a cache structure. You can set the
 *         timeout and other cache flags using this pointer.
```

```
 *  This function is called at startup to allow you to customize
 *   certain
 *  aspects of the access method. For the most part, it is for
 *   advanced
 *  users. The default code should suffice for most cases. If no
 *   custom
 *  container is allocated, the MFD code will create one for your.
 *
 *  This is also the place to set up cache behavior. The default, to
 *  simply set the cache timeout, will work well with the default
 *  container. If you are using a custom container, you may want to
 *  look at the cache helper documentation to see if there are any
 *  flags you want to set.
 *
 * @remark
 *  This would also be a good place to do any initialization needed
 *  for you data source. For example, opening a connection to another
 *  process that will supply the data, opening a database, etc.
 */
void
sixtofourRelayTable_container_init(netsnmp_container
**container_ptr_ptr,
                        netsnmp_cache *cache)
{

DEBUGMSGTL(("verbose:sixtofourRelayTable:sixtofourRelayTable_containe
r_init","called\n"));

    if((NULL == cache) || (NULL == container_ptr_ptr)) {
        snmp_log(LOG_ERR,"bad params to
sixtofourRelayTable_container_init\n");
        return;
    }

    /*
     * For advanced users, you can use a custom container. If you
     * do not create one, one will be created for you.
     */
   // Δημιουργείται αυτόματα ο container
    *container_ptr_ptr = NULL;

    /*
     * Set up sixtofourRelayTable cache properties.
     *
     */

    cache->timeout = SIXTOFOURRELAYTABLE_CACHE_TIMEOUT; /* seconds */

} /* sixtofourRelayTable_container_init */

/**
 * load cache data
 *
 * Implement sixtofourRelayTable cache load
 *
 * @param container container to which items should be inserted
 *
 * @retval MFD_SUCCESS              : success.
 * @retval MFD_RESOURCE_UNAVAILABLE : Can't access data source
 * @retval MFD_ERROR                : other error.
 *
```

```
 *  This function is called to cache the index(es)and data
 *  for the every row in the data set.
 *
 * @remark
 * While loading the cache, the only important thing is the indexes.
 * If access to your data is cheap/fast it would make sense to update
 * the data here.
 *
 * @note
 *  If you need consistency between rows (like you want statistics
 *  for each row to be from the same time frame), you should set all
 *  data here.
 *
 */
int
sixtofourRelayTable_cache_load(netsnmp_container *container)
{
    sixtofourRelayTable_rowreq_ctx *rowreq_ctx;
    size_t                 count = 0;

    /*
     * this code is based on a data source that is a
     * text file to be read and parsed.
     */

    FILE *filep,*filep_check;
    char line[MAX_LINE_SIZE];
    int index_count=0;          //δείκτης για τον επαναπροσδιορισμό
                                 της σωστής κατάταξης μετά την
                                 αφαίρεση στοιχείου από τον πίνακα
    int rc = MFD_SUCCESS;
    int if_flag=0;              //σημαία που υποδηλώνει την ύπαρξη
                                  του 6to4 interface
    char *line_check = NULL;
    size_t len = 0;
    char temp[30];
    char devname[10][30];
    static int counter;



    /*
     * temporary storage for index values
     */
        /*
         *
sixtofourRelayIndex(1)/INTEGER32/ASN_INTEGER/long(long)//l/a/w/e/r/d/
h
         */
   long   sixtofourRelayIndex;


DEBUGMSGTL(("verbose:sixtofourRelayTable:sixtofourRelayTable_cache_lo
ad","called\n"));

//ΕΛΕΓΧΟΣ ΑΝ ΥΠΑΡΧΕΙ ΣΤΟ ΣΥΣΤΗΜΑ 6TO4 INTERFACE

 filep_check = fopen("/proc/net/if_inet6", "r");
 if (NULL == filep_check)
             return MFD_RESOURCE_UNAVAILABLE;
```

```
counter=0;
 while (  getline(&line_check,&len,filep_check) !=-1)
        {
        if (strncmp(line_check,"2002",4)==0)
                {
            counter++;
                if_flag=1;
                sscanf(line_check,"%s %s %s %s %s
%s",devname[counter],devname[counter],devname[counter],devname[counte
r],devname[counter],devname[counter]);
puts(devname[counter]);
                }
        }
        fclose(filep_check);


    /*
    ************************************************************
    ***              AN ΥΠΑΡΧΕΙ 6TO4 INTERFACE                ***
    ***----------------------------------------------------------***/



 if (if_flag==1)
    {

    /*
     * ανοίγει το κυρίως αρχείο δεδομένων.
     */

    index_count=0;
    filep = fopen("/proc/net/ipv6_route","r");
    if(NULL ==  filep)
            return MFD_RESOURCE_UNAVAILABLE;



    /*
     * Load/update data in the sixtofourRelayTable container.
     * loop over your sixtofourRelayTable data, allocate a rowreq
     * context,
     * set the index(es) and data and insert into
     * the container.
     */

    while( 1 )
        {
        char reladdrhex[33],maskstr[3],addr[8][5];
        char RelayAddress[40],prefix[33];
        char desthex[33],RelayDest[40],DestAddrPrefix[40];
        char route_metric[9],met[4][3];
        size_t sixtofourRelayAddress_len=0;
        size_t sixtofourRelayDestAddr_len=0;
        size_t sixtofourRelayDestAddrPrefix_len=0;
        char temp[60];
        int sixtofourRelayAnycastStatus=2;
        int mask,maskhex,i;
        long sixtofourRelayRouteMetric;
        struct in6_addr sixtofourRelayDestAddrPrefix,
        sixtofourRelayDestAddr, sixtofourRelayAddress;
        char interface[30];
```

```
        static int sixtofourRelayIfIndex,ff;

    /*
     * παίρνει μια γραμμή παραλείποντας τις κενές και επιπλέον
       απορρίπτει:
     * > fe80::/  - link-local
     * > fe00::/  - site-local
     * > fec0::/
     * > ff00::/  -any multicast
     * > ::/      - loopback,compatible etc
     * > 2002::/  -6to4 address
     * > unreachable
     */
        do
                {
                if (!fgets(line, sizeof(line), filep))
                        {
                        /* we're done */
                        fclose(filep);
                        filep = NULL;
                        }

               ff=0;
              for(i=0;i<=counter;i++)
                    {
                        if(strstr(line,devname[i]) != NULL )
                        {
                        puts("taitia3e interface");
                        ff=1;
                        break;
                        }
                    }
               if(ff==0 ) {strcpy(line,"\n"); continue;} //έλεγχος αν
                                              ανήκει στο 6to4 interface


//αν είναι :: και όχι η default προς relay
              sscanf(line,"%s %s %s %s %s ",temp,temp,temp,temp
,temp);

                 if( (strncmp(line,"0000",4)==0) &&
(strcmp(temp,"00000000000000000000000000000000")==0) )
      {
      strcpy(line,"\n");
      continue;
      }


                 if((strncmp(line,"fe80",4)==0) ||
(strncmp(line,"ff00",4)==0)  || (strncmp(line,"fe00",4)==0) ||
(strncmp(line,"fec0",4)==0) || (strncmp(line,"2002",4)==0) )
strcpy(line,"\n");



                } while (filep && (line[0] == '\n'));

    /*
     * check for end of data
     */
    if(NULL == filep)
```

```
        break;


        /*
         * Αν συνάντησε το ίδιο interface προηγουμένως, να
         * χρησιμοποιήσει τον ίδιο δείκτη. Ειδάλλως, να βρει ένα
         * αχρησιμοποίητο δείκτη και να τον χρησιμοποιήσει
         */

        index_count++;

        //η συνδεδεμένη λίστα "sixtofourRelayTable" ορίστηκε στο
          sixtofourRelayTable.h σαν ο user context μας.
        sixtofourRelayIndex =
se_find_value_in_slist("sixtofourRelayTable", line);

//ΠΕΡΙΠΤΩΣΗ: ΔΕΝ ΥΠΑΡΧΕΙ ΣΤΗΝ ΛΙΣΤΑ
      if (sixtofourRelayIndex == SE_DNE)
                {
                sixtofourRelayIndex =
se_find_free_value_in_slist("sixtofourRelayTable");//αν βρει
                                    επιστρέφει το max_value+1

                if (sixtofourRelayIndex == SE_DNE)
                sixtofourRelayIndex = 1;  /* Νέα λιστα! */

                //Περίπτωση που είχε πριν αφαιρεθεί αντικείμενο
                  μικρότερου δείκτη και τώρα προστίθεται άλλο
                if (sixtofourRelayIndex != index_count)
sixtofourRelayIndex=index_count;


se_add_pair_to_slist("sixtofourRelayTable",strdup(line),
sixtofourRelayIndex);



DEBUGMSGTL(("sixtofourRelayTable:sixtofourRelayIndex", "new
sixtofourRelayIndex%d for %s\n",sixtofourRelayIndex, line));
                }

//ΠΕΡΙΠΤΩΣΗ: ΥΠΑΡΧΕΙ ΣΤΗΝ ΛΙΣΤΑ ΑΛΛΑ ΕΧΕΙ ΣΒΗΣΤΕΙ ΑΝΤΙΚΕΙΜΕΝΟ ΜΕ
  ΜΙΚΡΟΤΕΡΟ ΔΕΙΚΤΗ ΚΑΙ ΠΡΕΠΕΙ ΝΑ ΓΙΝΕΙ ΑΝΑΚΑΤΑΝΟΜΗ ΔΕΙΚΤΩΝ

        else  if (sixtofourRelayIndex != index_count)
                {
                        sixtofourRelayIndex=index_count;

se_add_pair_to_slist("sixtofourRelayTable",strdup(line),
sixtofourRelayIndex);
                }

         if ((NULL == line) || (0 == sixtofourRelayIndex))
                        {
                        rc = MFD_END_OF_DATA;
                        break;
                        }
```

```
        /*
         * set indexes in new sixtofourRelayTable rowreq context.
         */

        rowreq_ctx = sixtofourRelayTable_allocate_rowreq_ctx();
        if (NULL == rowreq_ctx) {
            snmp_log(LOG_ERR, "memory allocation failed\n");
            return MFD_RESOURCE_UNAVAILABLE;
        }
        if(MFD_SUCCESS != sixtofourRelayTable_indexes_set(rowreq_ctx
                              , sixtofourRelayIndex
               )) {
            snmp_log(LOG_ERR,"error setting index while loading "
                     "sixtofourRelayTable cache.\n");
            sixtofourRelayTable_release_rowreq_ctx(rowreq_ctx);
            continue;
        }


        /*
         * populate sixtofourRelayTable data context.
         */

//κάνει parsing τις εγγραφές του κυρίως file
        sscanf(line,"%s %s %s %s %s %s %s %s %s
%s",desthex,maskstr,temp,temp,reladdrhex,route_metric,interface,inter
face,interface,interface);


/*****************παίρνει την τιμή του metric**********************/

        if (strcmp(route_metric,"ffffffff")==0)
sixtofourRelayRouteMetric=-1;
        else
                {

sscanf(route_metric,"%2s%2s%2s%2s",met[0],met[1],met[2],met[3]);

sixtofourRelayRouteMetric=16777216*(hextoint(met[0]))+65536*(hextoint
(met[1]))+256*(hextoint(met[2]))+(hextoint(met[3]));
                }


        i=0;
        while(1)
                {
                i++;
                if(strcmp(sixtofourinterfaceindex[i],interface)==0)
                        {
                        sixtofourRelayIfIndex=i;
                        break;
                        }
                                }



/*****************υπολογισμός του prefix**************************/

        mask=hextoint(maskstr);
        mask=mask/4;
```

```
        for(i=0;i<mask;i++)
                {
                prefix[i]='1';
                }
        for(i=mask;i<32;i++)
                {
                prefix[i]='0';
                }
        prefix[32]='\0';

sscanf(prefix,"%4s%4s%4s%4s%4s%4s%4s%4s",addr[0],addr[1],addr[2],addr
[3],addr[4],addr[5],addr[6],addr[7]);

sprintf(DestAddrPrefix,"%s:%s:%s:%s:%s:%s:%s:%s",addr[0],addr[1],addr
[2],addr[3],addr[4],addr[5],addr[6],addr[7]);
puts(DestAddrPrefix);
      //μετατρέπει την mask σε network byte order για να την περάσει
μετά στον row context

inet_pton(AF_INET6,DestAddrPrefix,&sixtofourRelayDestAddrPrefix);

sixtofourRelayDestAddrPrefix_len=sizeof(sixtofourRelayDestAddrPrefix)
;


/*************υπολογισμός της address του relay router**************/

   //έλεγχος αν η relay address είναι η  anycast address
        if(
(strcmp(reladdrhex,"2002c0586301000000000000000000")==0) ||
(strcmp(reladdrhex,"00000000000000000000000c0586301")==0) )
                {
                sixtofourRelayAnycastStatus=1;

strcpy(RelayAddress,"2002:c058:6301:0000:0000:0000:0000:0000");
                }
        else
                {

sscanf(reladdrhex,"%4s%4s%4s%4s%4s%4s%4s%4s",addr[0],addr[1],addr[2],
addr[3],addr[4],addr[5],addr[6],addr[7]);

      //περίπτωση που εχει δοθει σε ipv6 ipv4-compatible address
            if (strcmp(addr[0],"0000")==0)

      sprintf(RelayAddress,"%s:%s:%s:%s:%s:%s:%s:%s","2002",addr[6],a
ddr[7],addr[0],addr[0],addr[0],addr[0],addr[0]);

            else

sprintf(RelayAddress,"%s:%s:%s:%s:%s:%s:%s:%s",addr[0],addr[1],addr[2
],addr[3],addr[4],addr[5],addr[6],addr[7]);
                }
        inet_pton(AF_INET6,RelayAddress,&sixtofourRelayAddress);
        sixtofourRelayAddress_len=sizeof(sixtofourRelayAddress);



/*************υπολογισμός destination address******************/
```

```
sscanf(desthex,"%4s%4s%4s%4s%4s%4s%4s%4s",addr[0],addr[1],addr[2],add
r[3],addr[4],addr[5],addr[6],addr[7]);


sprintf(RelayDest,"%s:%s:%s:%s:%s:%s:%s:%s",addr[0],addr[1],addr[2],a
ddr[3],addr[4],addr[5],addr[6],addr[7]);

        inet_pton(AF_INET6,RelayDest,&sixtofourRelayDestAddr);
        sixtofourRelayDestAddr_len=sizeof(sixtofourRelayDestAddr);



    /*
     * setup/save data for sixtofourRelayIfIndex
     *
sixtofourRelayIfIndex(2)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/
d/h
     */
    /*
     * Integer based value can usually just do a direct copy.
     */
    rowreq_ctx->data.sixtofourRelayIfIndex = sixtofourRelayIfIndex;


    /*
     * setup/save data for sixtofourRelayDestAddr
     *
sixtofourRelayDestAddr(3)/InetAddressIPv6/ASN_OCTET_STR/char(char)//L
/A/w/e/R/d/H
     */
    /*
     * επειδή το μέγεθος της διευθύνσεως είναι σταθερό και έχει
     * δηλωθεί δεν χρειάζεται να δεσμευτεί μνήμη
     */

       rowreq_ctx->data.sixtofourRelayDestAddr_len =
sixtofourRelayDestAddr_len;
    memcpy( rowreq_ctx->data.sixtofourRelayDestAddr, (u_char
*)&sixtofourRelayDestAddr,sizeof(sixtofourRelayDestAddr) );



    /*
     * setup/save data for sixtofourRelayDestAddrPrefix
     *
sixtofourRelayDestAddrPrefix(4)/InetAddressIPv6/ASN_OCTET_STR/char(ch
ar)//L/A/w/e/R/d/H
     */

        rowreq_ctx->data.sixtofourRelayDestAddrPrefix_len =
sixtofourRelayDestAddrPrefix_len;
    memcpy( rowreq_ctx->data.sixtofourRelayDestAddrPrefix, (u_char
*)&sixtofourRelayDestAddrPrefix,sizeof(sixtofourRelayDestAddrPrefix))
;



    /*
     * setup/save data for sixtofourRelayAddress
```

```
     *
sixtofourRelayAddress(5)/InetAddressIPv6/ASN_OCTET_STR/char(char)//L/
A/w/e/R/d/H
     */

    rowreq_ctx->data.sixtofourRelayAddress_len =
sixtofourRelayAddress_len;
    memcpy( rowreq_ctx->data.sixtofourRelayAddress,(u_char
*)&sixtofourRelayAddress,            sizeof(sixtofourRelayAddress) );


    /*
     * setup/save data for sixtofourRelayAnycastStatus
     *
sixtofourRelayAnycastStatus(6)/INTEGER/ASN_INTEGER/long(u_long)//l/A/
w/E/r/d/h
     */
    /*
     * Define sixtofourRelayAnycastStatus mapping.
     * Επειδή είναι αντικείμενο τύπου INTEGER χρειάζεται αντιστοίχηση
     */

    if(MFD_SUCCESS !=
       sixtofourRelayAnycastStatus_map(&rowreq_ctx->
data.sixtofourRelayAnycastStatus, sixtofourRelayAnycastStatus )) {
        return MFD_ERROR;
    }


    /*
     * setup/save data for sixtofourRelayRouteMetric
     *
sixtofourRelayRouteMetric(7)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/
e/r/d/h
     */

    rowreq_ctx->data.sixtofourRelayRouteMetric =
sixtofourRelayRouteMetric;


        /*
         * Τοποθετούνται τα δεδομένα στον container
         */
        CONTAINER_INSERT(container, rowreq_ctx);
        ++count;
    }


    if(NULL != filep)
        fclose(filep);

DEBUGMSGT(("verbose:sixtofourRelayTable:sixtofourRelayTable_cache_loa
d",
               "inserted %d records\n", count));

    return rc;

  }  //κλείνει το if (if_flag), αν υπήρχε δηλαδή 6to4 interface


} /* sixtofourRelayTable_cache_load */
```

```
/**
 * cache clean up
 *
 * @param container container with all current items
 *
 *  This optional callback is called prior to all
 *  item's being removed from the container. If you
 *  need to do any processing before that, do it here.
 *
 * @note
 *  The MFD helper will take care of releasing all the row contexts.
 *
 */
void
sixtofourRelayTable_cache_free(netsnmp_container *container)
{

DEBUGMSGTL(("verbose:sixtofourRelayTable:sixtofourRelayTable_cache_fr
ee","called\n"));

    /*
     * Free sixtofourRelayTable cache.
     */
} /* sixtofourRelayTable_cache_free */

/**
 * prepare row for processing.
 *
 * When the agent has located the row for a request, this function is
 * called to prepare the row for processing. If you fully populated
 * the data context during the index setup phase, you may not need to
 * do anything.
 *
 * @param rowreq_ctx pointer to a context.
 *
 * @retval MFD_SUCCESS     : success.
 * @retval MFD_ERROR       : other error.
 */
int
sixtofourRelayTable_row_prep( sixtofourRelayTable_rowreq_ctx
*rowreq_ctx)
{

DEBUGMSGTL(("verbose:sixtofourRelayTable:sixtofourRelayTable_row_prep
","called\n"));

    netsnmp_assert(NULL != rowreq_ctx);

    return MFD_SUCCESS;
} /* sixtofourRelayTable_row_prep */
```

stf_functions.c

Χρήσιμες συναρτήσεις που χρησιμοποιούνται από τους δυο πίνακες:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <net/if.h>
#include <net/if_arp.h>
#include <arpa/inet.h>
#include <stf_functions.h>


#define inaddrr(x) (*(struct in_addr *) &ifr->x[sizeof sa.sin_port])



/************************************************************
 ***** Παίρνει με ioctl() την ipv4 address ενός interface*****
 ************************************************************/

static int get_addr(int sock, char * ifname, struct sockaddr *
ifaddr,char *ipv4)
{

struct ifreq *ifr;
struct ifreq ifrr;
struct sockaddr_in sa;

ifr = &ifrr;

ifrr.ifr_addr.sa_family = AF_INET;

strncpy(ifrr.ifr_name, ifname, sizeof(ifrr.ifr_name));

if (ioctl(sock, SIOCGIFADDR, ifr) < 0) {
printf("No %s interface.\n", ifname);
return -1;
}

*ifaddr = ifrr.ifr_addr;
memcpy(ipv4, inet_ntoa(inaddrr(ifr_addr.sa_data)),14);
//printf("Address for %s: %s\n", ifname,
inet_ntoa(inaddrr(ifr_addr.sa_data)));
return 0;
}  //τέλος της get_addr
/************************************************************/


/************************************************************
 *Συνάρτηση για μετατροπή ενός δεκαεξαδικού αριθμού μεγέθους ενός*
 *                        byte σε δεκαδικό                    *
 ************************************************************/

int hextoint(char *hex)
 {
```

```
 static int ret;
 static int a;
 static int b;
 a=(int)hex[0];
 b=(int)hex[1];
 if (a>=(int)'a') a=a-87; else a=a-48;
 if (b>=(int)'a') b=b-87; else b=b-48;
 ret=(16*a+b);
 return ret;
 }

/*********************************************************************/
```

Αντικείμενο *sixtofourTunnelMode*

sixtofourTunnelMode.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        : mib2c.scalar.conf,v 1.8 2004/10/14 12:57:34 dts12
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "sixtofourTunnelMode.h"

/** Initializes the sixtofourTunnelMode module */
void
init_sixtofourTunnelMode(void)
{
    static oid sixtofourTunnelMode_oid[] = { 1,3,6,1,3,19,1,1,1 };

  DEBUGMSGTL(("sixtofourTunnelMode", "Initializing\n"));

    netsnmp_register_scalar(
        netsnmp_create_handler_registration("sixtofourTunnelMode",
handle_sixtofourTunnelMode,
                                sixtofourTunnelMode_oid,
OID_LENGTH(sixtofourTunnelMode_oid),
                                HANDLER_CAN_RONLY
        ));
}

int
handle_sixtofourTunnelMode(netsnmp_mib_handler *handler,
                           netsnmp_handler_registration *reginfo,
                           netsnmp_agent_request_info   *reqinfo,
                           netsnmp_request_info         *requests)
{
static int sixtofourTunnelMode=2;
```

```
static char *line = NULL;
static size_t len = 0;
static char dev[40];
static char command[60];
FILE  *filep;

    switch(reqinfo->mode) {

        case MODE_GET:

      strcpy(command,"ip tunnel show mode sit >
/usr/local/tunnel_mode ");
        if (system(command)==-1)
                        return SNMP_ERR_RESOURCEUNAVAILABLE;

        else
                {
                filep = fopen("/usr/local/tunnel_mode", "r");
                if (NULL == filep)
                         return SNMP_ERR_RESOURCEUNAVAILABLE;

                while (getline(&line,&len,filep) !=-1)
                        {    //έλεγχος αν υπάρχει ipv6/ip tunnel
                        sscanf(line,"%s %s",dev,dev);
                        if (strcmp(dev,"ipv6/ip")==0)
                                {
                                sixtofourTunnelMode=1;
                                break;
                                }
                        }
                        fclose(filep);
                        strcpy(command,"rm /usr/local/tunnel_mode");
                        if (system(command)==-1)
                        return SNMP_ERR_RESOURCEUNAVAILABLE;
                }

            snmp_set_var_typed_value(requests->requestvb,
ASN_INTEGER,(u_char*)&sixtofourTunnelMode,
sizeof(sixtofourTunnelMode));
            break;


        Default:
            /* we should never get here, so this is a really bad
error */
            snmp_log(LOG_ERR, "unknown mode (%d) in
handle_sixtofourTunnelMode\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
```

sixtofourIfNumber.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        : mib2c.scalar.conf,v 1.8 2004/10/14 12:57:34 dts12 Exp $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "sixtofourIfNumber.h"

/** Initializes the sixtofourIfNumber module */
void
init_sixtofourIfNumber(void)
{
    static oid sixtofourIfNumber_oid[] = { 1,3,6,1,3,19,1,1,2 };

  DEBUGMSGTL(("sixtofourIfNumber", "Initializing\n"));

    netsnmp_register_scalar(
        netsnmp_create_handler_registration("sixtofourIfNumber",
handle_sixtofourIfNumber,
                                sixtofourIfNumber_oid,
OID_LENGTH(sixtofourIfNumber_oid),
                                HANDLER_CAN_RONLY
        ));
}

int
handle_sixtofourIfNumber(netsnmp_mib_handler *handler,
                         netsnmp_handler_registration *reginfo,
                         netsnmp_agent_request_info   *reqinfo,
                         netsnmp_request_info         *requests)
{
 static char *line = NULL;
 static size_t len = 0;
 static int sixtofourIfNumber=0;
 static FILE  *filep;

   switch(reqinfo->mode) {

        case MODE_GET:

        sixtofourIfNumber=0;
        filep = fopen("/proc/net/if_inet6", "r");
        if (NULL == filep)
             return SNMP_ERR_RESOURCEUNAVAILABLE;

        while (  getline(&line,&len,filep) !=-1)
                {
                if (strncmp(line,"2002",4)==0)
                        sixtofourIfNumber++;
                }
```

```
        fclose(filep);


            snmp_set_var_typed_value(requests->requestvb,
ASN_INTEGER,
                                     (u_char
*)&sixtofourIfNumber,sizeof(sixtofourIfNumber));
            break;


        default:
            /* we should never get here, so this is a really bad
error */
            snmp_log(LOG_ERR, "unknown mode (%d) in
handle_sixtofourIfNumber\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
```

ΠΑΡΑΠΟΜΠΕΣ - REFERENCES

[1] B. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[2] C. Huitema , "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, June 2001.

[3] P. Savola, and C. Patel , "Security Considerations for 6to4", RFC 3964, December 2004.

[4] R. Gilligan and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, August 2000.

[5] E. Nordmark and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", Work in Progress, March, 2005.

[6] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[7] T. Hain, "6to4-relay discovery and scaling", Work in Progress, July 2000.

[8] R. Hinden, M. O'Dell and S. Deering, "An IPv6 Aggregatable Global Unicast Address Format", RFC 2374, July 1998.

[9] J. Postel, "Internet Protocol", RFC 791, September 1981.

[10] D. Thaler, "IP Tunnel MIB", RFC 2667, August 1999.

[11] B. Carpenter and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529.

[12] F. Templin, T. Gleeson, M. Talwar and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", Work in Progress.

[13] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[14] R. Draves, "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[15] P. Nikander, J. Kempf and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[16] G. Tsirtsis and P. Srisuresh, " Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[17] J. Abley, B. Black and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, August 2003.

[18] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[19] Rose M. Editor, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, Performance Systems International, March 1991.

[20] M. Rose and K. McCloghrie, "Structure and identification of management information for TCP/IP-based internets",RFC 1155.

[21] K. McCloghrie, D. Perkins, and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[22] D. Haskin and S. Onishi, "Management Information Base for IP Version 6: Textual Conventions and General Group", RFC 2465, December 1998.

[23] M. Daniele, B. Haberman, B. Routhier and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

[24] S. Hagen, "O'REILLY: IPv6 Essentials", 1st Edition, July 2002.

[25] J. Davies, "Understanding IPv6", 1st Edition, 2003

[26] D. Mauro and K. Schmidt, "O'REILLY: Essential SNMP", 1st Edition, July 2001

[27] P. Bieringer, "IPv6 & Linux – HowTo", original site of publishing: http://www.bieringer.de/linux/IPv6/

[28] NET-SNMP Web-Site, http://www.net-snmp.org