ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

"Σχεδιασμός, Ανάπτυξη και Υλοποίηση Συστήματος Διαχείρισης Δικτυακών Πόρων"

ΑΝΤΩΝΑΣ-ΜΠΕΛΕΓΡΙΝΗΣ ΑΝΤΩΝΙΟΣ
ΑΜ: 2018

Επιβλέπων καθηγητής: **Ευάγγελος Πάλλης**

Ημερομηνία: **11/12/2013**

Σε όλους όσους

στάθηκαν δίπλα μου,

με ιδιαίτερη αγάπη

Abstract

The continuous evolution of IP based networks and their wide spread and use over the last decade has led to the development of new systems and the upgrade of old, in order to meet the ever changing need for management. The current network infrastructure rapidly came to the point in which the demand exceeded the supply and so, a serious downgrading of connections' quality as well as an improper resource distribution has been observed.

This Thesis focuses on the development of a system that is capable of providing Quality of Service (QoS) as well as the deployment of mechanisms to efficiently manage network resources, distribute Quality of Service and enable cooperation and interoperability between different autonomous network systems in an automated way. For this purpose, it proposes the implementation of a Network Resource Management system that can a) dimension the network, b) implement the dimensioning model and c) accept SLA / QoS requests, utilizing not only current architectures (DiffServ, MPLS) but future ones as well (Content Aware Networks).

Σύνοψη

Η συνεχής εξέλιξη των δικτύων που βασίζονται στο IP πρωτόκολλο και η ευρύτατη διάδοση και χρήση τους τα τελευταία χρόνια καθοδηγεί την ανάγκη για την ανάπτυξη νέων συστημάτων και την αναβάθμιση των υπαρχόντων, προκειμένου να καλυφθούν οι συνεχώς μεταβαλλόμενες ανάγκες για τη διαχείριση τους. Οι τρέχουσες υποδομές των δικτύων έφτασαν γρήγορα στο σημείο στο οποίο η ζήτηση ξεπέρασε την προσφορά και έτσι παρατηρήθηκε όχι μόνο μια σοβαρή υποβάθμιση της ποιότητας των συνδέσεων αλλά και η κακή κατανομή των πόρων.

Κεντρικό αντικείμενο αυτής της πτυχιακής εργασίας αποτελεί η ανάπτυξη ενός συστήματος για παροχή Ποιότητας Υπηρεσίας (Quality of Service – QoS) καθώς και η ανάπτυξη μηχανισμών για την αποδοτική διαχείριση των πόρων, τον καταμερισμό της ποιότητας υπηρεσίας και τη δυνατότητα συνεργασίας και διαλειτουργικότητας μεταξύ διαφορετικών αυτόνομων δικτυακών τμημάτων με αυτοματοποιημένο τρόπο. Για το σκοπό αυτό προτείνεται η υλοποίηση ενός συστήματος διαχείρισης δικτυακών πόρων, το οποίο υποστηρίζει τις λειτουργίες α) της μοντελοποίησης δικτύου, β) εφαρμογής του μοντέλου διαστασιοποίησης στην τελική κατάσταση γ) της αποδοχής SLA/QoS αιτημάτων, αξιοποιώντας όχι μόνο υπάρχουσες αρχιτεκτονικές (DiffServ, MPLS) αλλά και μελλοντικές (Content Aware Networks – CAN).

Ευχαριστίες

Καταρχήν θέλω να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Δρ. Ευάγγελο Πάλλη για την σημαντική καθοδήγηση και πολύτιμη βοήθεια που μου προσέφερε για την επιτυχή ολοκλήρωση της πτυχιακής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τους υποψήφιους διδάκτορες κ. Πέτρο Αναπλιώτη, κ. Ευάγγελο Μαρκάκη και κ. Ανάργυρο Σιδέρη όπως και τα υπόλοιπα μέλη του εργαστηρίου "ΠΑΣΙΦΑΗ" για την συνεχή τους υποστήριξη και άψογη συνεργασία.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου για την ανεκτίμητη συμπαράσταση και κατανόηση που έδειξε όλα τα χρόνια των σπουδών μου.

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

Λεξιλόγιο

AF Assured Forwarding

ATM Asynchronous Transfer Mode

AS Autonomous System

BA Behavior Aggregate

BE Best Effort

CAN Content Aware Network

CIR Committed Information Rate

CoS Class of Service

DiffServ / DS Differentiated Services

DSCP DiffServ Code Point

DNS Domain Name Service

EF Expedited Forwarding

FEC Forwarding Equivalence Class

FI Future Internet

FTP File Transfer Protocol

HTTP Hyper Text Transfer Protocol

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IntServ / IS Integrated Services

IP Internet Protocol

LER Label Edge Router

LSP Label-Switched Path

LSR Label Switch Router

MANE Media-Aware Network Element

MPLS Multi-Protocol Label Switching

NRM Network Resource Management

OA Ordered Aggregate

PHB Per-Hop Behavior

PIR Peak Information Rare

QoE Quality of Experience

QoS Quality of Service

RFC Request for Comments

RSVP Resource Reservation Protocol

SLA Service Level Agreement

SLS Service Level Specification

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

SP Service Provider

SQL Structed Query Language

TCP Transmission Control Protocol

TE Traffic Engineering

ToS Type of Service

UDP User Datagram Protocol

VM Virtual Machine

XML Extended Markup Language

1 Εισαγωγή

1.1 Μελλοντικό διαδίκτυο

Κατά τις τελευταίες δεκαετίες, το διαδίκτυο έχει εξελιχθεί και ο αριθμός των υπολογιστών που συνδέονται σε αυτό έχει σημειώσει ραγδαία αύξηση. Παρόλα αυτά, η αρχιτεκτονική του βασίζεται ακόμα στις αυθεντικές αρχές σχεδίασης ενός ακαδημαϊκού δικτύου, το οποίο σχεδιάστηκε για να συνδέει μόνο μερικούς υπολογιστές μεταξύ τους. Από τότε, το διαδίκτυο έχει αλλάξει δραματικά τόσο σε μέγεθος όσο και στο τρόπο που χρησιμοποιείται. Πέρα από την ακαδημαϊκή του χρήση, το διαδίκτυο χρησιμοποιείται σήμερα ως μία επιχειρηματική πλατφόρμα και έχει γίνει κεντρικό μέρος της κοινωνικής ζωής. Νέες τεχνολογίες, όπως η διανομή περιεχομένου (Content Distribution), η ψηφιακή διαδραστική τηλεόραση (Digital Interactive Television), η κατά-παραγγελία εικόνας και ήχου (Video / Audio on Demand), η ζωντανή μετάδοση (Live Streaming) και η κοινωνική δικτύωση (Social Networking), πλούσιες σε περιεχόμενο δημιουργημένο από χρήστες, με υψηλές απαιτήσεις παροχής ποιότητας της υπηρεσίας / εμπειρίας (Quality of Service / Experience), αυξάνουν αυστηρά την ανάγκη για πόρους στα δίκτυα και επομένως δημιουργούν νέες προκλήσεις στη λειτουργία, τη διαμόρφωση και τη διαχείριση τους.

Προς αυτή την κατεύθυνση, οι σημερινές ερευνητικές και τεχνολογικές δράσεις εστιάζονται (μεταξύ των άλλων) στην ανάπτυξη αρχιτεκτονικών και στην υλοποίηση τεχνικών με στόχο την αναγνώριση του περιεχομένου από το στρώμα δικτύου (Content Awareness) και την προσαρμογή των διαθέσιμων δικτυακών πόρων ανάλογα με τις ανάγκες της υπηρεσίας. Σε λογικό επίπεδο, τα δίκτυα αυτά επικάθονται πάνω από το TCP / IP στρώμα των παραδοσιακών / συμβατικών δικτύων, ενώ σε ιδεατό επίπεδο μπορούν να ενοποιήσουν δίκτυα διαφορετικών τεχνολογιών, αποκρύπτοντας τεχνολογικές και διαχειριστικές ετερογένειες από τους τελικούς χρήστες (παροχείς και καταναλωτές υπηρεσιών), προσφέροντας έναν ενιαίο και διαφανή τρόπο για την ανάπτυξη και την κατανάλωση υπηρεσιών. Για την επίτευξη αυτού του σκοπού είναι απαραίτητη α) η αναβάθμιση των συμβατικών δρομολογητών ώστε να έχουν επίγνωση του περιεχομένου (CAN routers) και β) η υλοποίηση των απαραίτητων διαχειριστικών οντοτήτων (Network Managers), οι οποίες θα αναλάβουν το συντονισμό και την παραμετροποίηση των αναβαθμισμένων δρομολογητών καθώς και την διάδραση με τους διαχειριστές των συμβατικών δικτύων και τους παροχείς υπηρεσιών.

1.2 Στόχος Πτυχιακής Εργασίας

Στα πλαίσια αυτά, η παρούσα πτυχιακή εργασία επικεντρώνεται στο σχεδιασμό, την ανάπτυξη και την υλοποίηση των δομικών στοιχείων αλλά και των διεπαφών ενός συστήματος διαχείρισης δικτυακών πόρων σε ένα αυτόνομο σύστημα (Intra-domain Network Resource Manager / Intra-NRM), το οποίο θα δύναται να παραμετροποιήσει το δίκτυο με τέτοιο τρόπο ώστε να παρέχει κλιμακωτά επίπεδα παροχής ποιότητας υπηρεσίας. Το σύστημα αυτό εκμεταλλευόμενο τις υπάρχουσες τεχνολογίες παροχής QoS (DiffServ, MPLS) σε συνδυασμό με την πληροφορία από τα ανώτερα στρώματα

(CAN layer / SLAs) θα παρέχει τη δυνατότητα προσαρμογής του δικτύου στις απαιτήσεις των υπηρεσιών.

1.3 Διάρθρωση Πτυχιακής Εργασίας

Ακολουθώντας την εισαγωγή, στο κεφάλαιο 2 παρουσιάζονται όλες οι απαραίτητες θεωρητικές πληροφορίες που απαιτούνται ώστε ο αναγνώστης να εξοικειωθεί και να κατανοήσει τη διαχείριση δικτύων. Στο κεφάλαιο 3 παρουσιάζονται οι βασικότερες τεχνολογίες παροχής ποιότητας υπηρεσίας. Στο κεφάλαιο 4 παρουσιάζετε η γενική αρχιτεκτονική των συστημάτων διαχείρισης δικτύων καθώς και το σύστημα διαχείρισης δικτυακών πόρων που αναπτύχτηκε στα πλαίσια αυτής της εργασίας. Γίνετε, επίσης, μία αναφορά στα εργαλεία που χρησιμοποιήθηκαν. Στο κεφάλαιο 5 γίνεται η αξιολόγηση του προτεινόμενου συστήματος και η παρουσίαση των αποτελεσμάτων που προκύπτουν από τις πειραματικές μετρήσεις που πραγματοποιήθηκαν. Τέλος, στο κεφάλαιο 6 διατυπώνονται τα συμπεράσματα της εργασίας αυτής και παρατίθενται κάποιες προτάσεις για μελλοντική έρευνα.

2 Διαχείριση Δικτύων - Τρέχουσες Προσεγγίσεις

2.1 Διαχείριση Δικτύων

Η διαχείριση του δικτύου είναι το σύνολο των ενεργειών που εξασφαλίζει ότι όλοι οι πόροι του δικτύου θα χρησιμοποιηθούν όσο το δυνατόν καλύτερα. Αποτελείται από τη διατήρηση όλων των πόρων του δικτύου σε καλή κατάσταση και τη διάθεσή τους στους χρήστες, σύμφωνα με τις προσδοκίες των χρηστών. Τα πρώτα δίκτυα υπολογιστών ήταν απλά και αποτελούνταν κυρίως από έναν ή περισσότερους κεντρικούς υπολογιστές που συνδεόταν με μερικά περιφερειακά. Δεδομένου ότι οι πόροι των δικτύων ήταν αρκετά περιορισμένοι, η διαχείριση αυτών των δικτύων ήταν σχετικά απλή και ο υπάρχουσες τεχνικές ήταν επαρκείς.

Σε αντιδιαστολή, τα δίκτυα σήμερα είναι πολύ πιο πολύπλοκα και, επιπλέον, η πολυπλοκότητα τους αυξάνεται διαρκώς. Η πολυπλοκότητα ενός δικτύου προκύπτει κυρίως από δύο πτυχές: τον μεγάλο αριθμό των συσκευών και την ετερογένεια των συσκευών στο δίκτυο. Τυπικά δίκτυα μπορεί να περιλαμβάνουν εκατοντάδες ή χιλιάδες συσκευές. Είναι αρκετά κοινό για επιχειρήσεις και οργανισμούς να διαθέτουν δίκτυα με έναν υπολογιστή ανά άτομο. Εκτός αυτού, τα σημερινά δίκτυα αποτελούνται συνήθως από μια ποικιλία συσκευών όπως υπολογιστές, σταθμούς εργασίας, εξυπηρετητές αρχείων, εκτυπωτές, δρομολογητές, μεταγωγείς, τείχη προστασίας, κ.λπ. Κάθε συσκευή στο δίκτυο μπορεί να έχει κατασκευαστεί από διαφορετικούς προμηθευτές και ως εκ τούτου, ακόμη και τα χαρακτηριστικά των παρόμοιων συσκευών μπορεί να είναι πολύ ανόμοια.

Ένα πρόσθετο πρόβλημα είναι ότι τα δίκτυα είναι δυναμικά. Τυπικά, τα δίκτυα αλλάζουν συνεχώς. Προστίθενται συχνά περισσότερες συσκευές ή παλαιότερες συσκευές αντικαθίστανται με νεότερες. Καθώς αυξάνεται ο αριθμός των συσκευών σε ένα τοπικό δίκτυο, αυτό πρέπει συχνά να χωριστεί σε τμήματα τα οποία συνδέονται με δρομολογητές. Περαιτέρω, οι προμηθευτές εφεύρουν τακτικά νέες συσκευές που προσφέρουν βελτιωμένες δυνατότητες. Αυτές οι συσκευές συχνά ενσωματώνονται στα δίκτυα, αντικαθιστώντας παλαιότερες συσκευές.

Η έλλειψη αξιοπιστίας των στοιχείων ενός δικτύου είναι ένα άλλο κίνητρο για την καλή διαχείριση του. Τα μεγάλα δίκτυα κατασκευάζονται, τυπικά, χρησιμοποιώντας συνδέσεις επικοινωνίας που διασχίζουν μεγάλες αποστάσεις. Αυτές οι συνδέσεις μπορεί να αποτύχουν επειδή υπόκεινται σε εχθρικούς περιβαλλοντικούς παράγοντες όπως οι καιρικές συνθήκες, η κοπή των καλωδίων ή κάποια δυσλειτουργία. Επιπλέον, τα μηχανήματα μπορεί να καταρρεύσουν λόγω σφαλμάτων στο λειτουργικό τους σύστημα, οι εκτυπωτές ενδέχεται να μπλοκάρουν ή να ξεμείνουν από χαρτί / τόνερ, οι δίσκοι μπορεί να καταστραφούν προκαλώντας τους διακομιστές αρχείων να μην είναι διαθέσιμοι ή τα συστατικά των μηχανημάτων μπορεί γενικά να αποτύχουν.

Τέλος, με την εμφάνιση και την ευρεία, πλέον, χρήση των πολυμεσικών εφαρμογών, οι οποίες είναι συνήθως λιγότερο ανθεκτικές στις διακυμάνσεις της καθυστέρησης, προκαλούνται δύο προβλήματα. Πρώτον, οι παραδοσιακές υλοποιήσεις των εφαρμογών πραγματικού χρόνου (real time) δεν αποδίδουν επαρκώς όταν "τρέχουν" πάνω από το σημερινό διαδίκτυο, επειδή οι μεταβολές στην καθυστέρηση είναι πολύ ακραίες και πάρα πολλά πακέτα απορρίπτονται. Δεύτερον, αυτές οι εφαρμογές συνήθως

δεν υποχωρούν υπό την παρουσία συμφόρησης. Όταν, δηλαδή, μάχονται για το εύρος ζώνης με παραδοσιακές εφαρμογές δεδομένων, οι εφαρμογές δεδομένων καταλήγουν να λαμβάνουν πολύ μικρό εύρος ζώνης. Έτσι, όχι μόνο δεν αποδίδουν πάντα με ικανοποιητικό τρόπο, αλλά συχνά παρεμβαίνουν με τις εφαρμογές δεδομένων.

2.2 Τεχνολογία Μηχανίκευσης Κίνησης (Traffic Engineering)

Συμφόρηση στο δίκτυο μπορεί να προκληθεί από την έλλειψη πόρων του δικτύου ή την άνιση κατανομή της κίνησης. Στη πρώτη περίπτωση, όλοι οι δρομολογητές και οι συνδέσεις είναι υπερφορτωμένες και η μόνη λύση είναι η παροχή περισσότερων πόρων αναβαθμίζοντας την υποδομή. Στη δεύτερη περίπτωση, ορισμένα τμήματα του δικτύου είναι υπερφορτωμένα ενώ άλλα τμήματα είναι ελαφρώς φορτωμένα. Η άνιση κατανομή της κίνησης μπορεί να προκληθεί από τα τρέχοντα πρωτόκολλα δυναμικής δρομολόγησης, επειδή πάντα επιλέγουν τις πιο κοντές διαδρομές για να διαβιβάσουν τα πακέτα. Ως αποτέλεσμα, οι δρομολογητές και οι συνδέσεις κατά μήκος της συντομότερης διαδρομής μεταξύ δύο κόμβων μπορεί να κορεστούν, ενώ οι δρομολογητές και οι συνδέσεις κατά μήκος μίας μεγαλύτερης διαδρομής να είναι σε αδράνεια. Για απλά δίκτυα, μπορεί να είναι δυνατό για τους διαχειριστές δικτύου να ρυθμίσουν με μη αυτόματο τρόπο το φόρτο των συνδέσεων, έτσι ώστε κίνηση να είναι ομοιόμορφα κατανεμημένη. Για σύνθετα δίκτυα αντίθετα, όπως αυτά των πάροχων υπηρεσιών, αυτό είναι σχεδόν αδύνατο.

Η μηχανίκευση κίνησης ορίζεται ως η πτυχή της μηχανίκευσης δικτύων που ασχολείται με το θέμα της αξιολόγησης και της βελτιστοποίησης των επιδόσεων των δικτύων. Η ΤΕ περιλαμβάνει την εφαρμογή των τεχνολογιών και των επιστημονικών αρχών για τη μέτρηση, τον χαρακτηρισμό, τη μοντελοποίηση και τον έλεγχο της κίνησης στα δίκτυα. Βασικοί στόχοι της ΤΕ αποτελούν η ενίσχυση της απόδοσης ενός δικτύου, τόσο σε επίπεδο κίνησης όσο και πόρων. Αυτό επιτυγχάνεται με την αντιμετώπιση των απαιτήσεων της κίνησης σε αποδόσεις, χρησιμοποιώντας παράλληλα τους πόρους του δικτύου οικονομικά και αξιόπιστα. Στα μετρικά απόδοσης της κίνησης περιλαμβάνονται η καθυστέρηση, η διακύμανση της καθυστέρησης, η απώλεια πακέτων, το εύρος ζώνης, κ.λπ.

Ένα πρακτικό πλεονέκτημα της συστηματικής εφαρμογής των αρχών της τεχνολογίας μηχανίκευσης κίνησης στα δίκτυα είναι ότι βοηθά στον εντοπισμό και τη οριοθέτηση στόχων και προτεραιοτήτων, όσον αφορά την βελτιστοποίηση της ποιότητας των υπηρεσιών που παρέχονται στους τελικούς χρήστες. Η εφαρμογή των αρχών της τεχνολογίας μηχανίκευσης κίνησης βοηθά επίσης στην μέτρηση και την ανάλυση της επίτευξης αυτών των στόχων. Αυτοί οι στόχοι μπορεί να επιτευχθούν μέσω της διαμόρφωσης της χωρητικότητας και τη διαχείριση της κίνησης. Η διαμόρφωση της χωρητικότητας περιλαμβάνει τη σχεδίαση της χωρητικότητας, τον έλεγχο της δρομολόγησης καθώς και τη διαχείριση των πόρων. Στους πόρους του δικτύου με ιδιαίτερο ενδιαφέρον περιλαμβάνονται το εύρος ζώνης, ο χώρος προσωρινής αποθήκευσης (buffer) και οι υπολογιστικοί πόροι. Ομοίως, η διαχείριση της κίνησης περιλαμβάνει λειτουργίες ελέγχου κίνησης των κόμβων, όπως η διαμόρφωση της κίνησης, η διαχείριση της ουράς, ο προγραμματισμός και άλλες λειτουργίες που

ρυθμίζουν τη ροή της κίνησης μέσω του δικτύου ή που διαχωρίζουν τη πρόσβαση σε πόρους δικτύου μεταξύ των διάφορων πακέτων ή ροών.

2.2.1 Μετρικά Απόδοσης Δικτύου

Η IETF[1] έχει αναπτύξει και τυποποιήσει ένα πλήθος μετρικών για την μέτρηση της ποιότητας, της απόδοσης και της αξιοπιστίας ενός δικτύου:

- **Availability**
  Ως διαθεσιμότητα ορίζεται το ποσοστό του χρόνου ανά έτος που ένα σύστημα ή μία υπηρεσία είναι διαθέσιμη στους χρήστες (π.χ., 98%) [RFC 2678]. Η διαθεσιμότητα μπορεί να μειωθεί όταν υπάρχει συμφόρηση δεδομένων ή όταν συγκεκριμένες συσκευές σε ένα δίκτυο χρησιμοποιούν περισσότερους πόρους από ότι ήταν αναμενόμενο.
- **Throughput**
  Είναι ο ρυθμός αποστολής, μετάδοσης ή λήψης των δεδομένων. Μπορεί να χρησιμοποιηθεί και για την εκτίμηση του διαθέσιμου εύρους ζώνης ενός καναλιού ή της τρέχουσας χρήσης του [RFC 3148]. Αρκετές πολυμεσικές εφαρμογές απαιτούν υψηλή και σταθερή διεκπεραιωτικότητα για τη σωστή λειτουργία τους. Πολύ χαμηλή διεκπεραιωτικότητα ή μεγάλες διακυμάνσεις της στη διάρκεια του χρόνου, υποδεικνύουν προβλήματα στη μεταφορά των δεδομένων. Μετριέται σε μονάδα πληροφορίας ανά χρονική στιγμή (π.χ., Bytes / sec).
- **One-Way Delay**
  Ο χρόνος που θα κάνει ένα πακέτο να φτάσει από τον αποστολέα στον παραλήπτη ονομάζετε μονόδρομη καθυστέρηση [RFC 2679]. Προκαλείτε κυρίως από τις ουρές και την επεξεργασία πακέτων στους δρομολογητές, όπως και από την διάδοση. Μεγάλη καθυστέρηση οδηγεί συνήθως σε μικρή ταχύτητα μετάδοσης δεδομένων. Για την μέτρηση, τα ρολόγια του αποστολέα και του παραλήπτη πρέπει να είναι συγχρονισμένα.
- **Jitter**
  Δείχνει την διακύμανση της καθυστέρησης που μεσολαβεί μεταξύ της αποστολής και της λήψης δύο επιλεγμένων πακέτων [RFC 3393]. Είναι γνωστό και ως Packet Delay Variation. Διακύμανση της καθυστέρησης εμφανίζετε συνήθως όταν τα πακέτα ακολουθούν διαφορετικές διαδρομές στο δίκτυο. Χρησιμεύει κυρίως σε εφαρμογές πραγματικού χρόνου οι οποίες χρειάζονται μικρές τιμές διακύμανσης της καθυστέρησης για να λειτουργήσουν σωστά.
- **Packet Loss**
  Αριθμός των πακέτων που χάθηκαν κατά την διάρκεια μίας σύνδεσης [RFC 2680]. Εμφανίζεται συνήθως ως Packet Loss Ratio. Πακέτα μπορεί να χαθούν όταν ο buffer ενός δρομολογητή γεμίσει ή όταν υπάρχουν λάθη σε ένα πακέτο. Πολλές απώλειες πακέτων μπορούν να επηρεάσουν την ταχύτητα λήψης των δεδομένων όπως και την ποιότητα της υπηρεσίας.

[1] http://www.ietf.org/about/

2.3 Ποιότητα Υπηρεσίας

Υπάρχουν εφαρμογές, όπως και πελάτες, που απαιτούν ισχυρότερες εγγυήσεις απόδοσης από το δίκτυο από ότι "το καλύτερο που μπορεί να γίνει υπό αυτές τις συνθήκες". Οι εφαρμογές πολυμέσων ειδικότερα, χρειάζονται συχνά ένα ελάχιστο εύρος ζώνης και μία μέγιστη καθυστέρηση ώστε να λειτουργήσουν σωστά. Η κίνηση στα σημερινά δίκτυα λαμβάνει μεταχείριση βέλτιστης προσπάθειας (Best Effort) του δικτύου, χωρίς εγγυήσεις για την αξιοπιστία, την καθυστέρηση, τη διακύμανση καθυστέρησης ή άλλα χαρακτηριστικά απόδοσης. Με την υπηρεσία παράδοσης BE, ωστόσο, μία μόνο εφαρμογή με υψηλές απαιτήσεις εύρους ζώνης μπορεί να οδηγήσει σε κακή ή μη αποδεκτή απόδοση για όλες τις εφαρμογές. Η ποιότητα της υπηρεσίας είναι η δυνατότητα για την παροχή διαφορετικών προτεραιοτήτων σε διαφορετικές εφαρμογές, χρήστες ή ροές δεδομένων ή την εγγύηση ενός ορισμένου επιπέδου απόδοσης σε μια ροή δεδομένων.

Μια εύκολη λύση για τη παροχή υψηλής ποιότητας της υπηρεσίας είναι να δημιουργηθεί ένα δίκτυο με αρκετή χωρητικότητα ώστε να υποστηρίξει οποιαδήποτε κίνηση θα ριχτεί σε αυτό. Αυτή η λύση ονομάζετε υπερπαροχή (overprovisioning). Το δίκτυο που προκύπτει θα είναι σε θέση να μεταφέρει την κίνηση των εφαρμογών χωρίς σημαντικές απώλειες και, υποθέτοντας ένα αξιοπρεπές σύστημα δρομολόγησης, θα μπορεί να παραδώσει τα πακέτα με χαμηλή καθυστέρηση. Έως κάποιο βαθμό, το τηλεφωνικό σύστημα χρησιμοποιεί αυτή τη λύση. Υπάρχει, απλά, τόσο μεγάλη διαθέσιμη χωρητικότητα που η ζήτηση μπορεί σχεδόν πάντα να τηρηθεί.

Το πρόβλημα με αυτή τη λύση είναι ότι είναι ακριβή. Πρόκειται ουσιαστικά για την επίλυση ενός προβλήματος, ρίχνοντάς του χρήματα. Μηχανισμοί ποιότητας υπηρεσίας επιτρέπουν σε ένα δίκτυο με λιγότερη χωρητικότητα να πληρεί τις απαιτήσεις των εφαρμογών εξίσου καλά, με χαμηλότερο κόστος. Επιπλέον, η υπερπαροχή βασίζεται στην αναμενόμενη κίνηση. Τίποτα όμως δεν μπορεί να εγγυηθεί ότι η κίνηση δεν θα αλλάξει σε τεράστιο βαθμό. Με τους μηχανισμούς παροχής ποιότητας της υπηρεσίας, το δίκτυο μπορεί να τιμήσει τις αποδόσεις που εγγυάται, ακόμη και όταν η κίνηση κάνει αιχμές, με το κόστος της μη αποδοχής κάποιων αιτημάτων. Για τη διασφάλιση της ποιότητας των παρεχόμενων υπηρεσιών ο χρήστης κάνει μία συμφωνία με το δίκτυο η οποία ονομάζετε συμφωνία παροχής υπηρεσιών (Service Level Agreement).

2.3.1 Service Level Agreement

Ένα SLA είναι ένας επίσημος ορισμός της σχέσης που υπάρχει μεταξύ ενός πάροχου υπηρεσιών και των πελατών του. Ένα SLA μπορεί να οριστεί και να χρησιμοποιηθεί στα πλαίσια της κάθε βιομηχανίας και χρησιμοποιείται για να προσδιορίσει αυτό που ο πελάτης θα μπορούσε να αναμένει από τον πάροχο, τις υποχρεώσεις του πελάτη καθώς και του πάροχου, τις επιδόσεις, τη διαθεσιμότητα και την ασφάλεια της υπηρεσίας, καθώς και τις διαδικασίες που πρέπει να ακολουθούνται για την εξασφάλιση της συμμόρφωσης με το SLA. Τα SLAs χρησιμοποιούνται συχνά όταν κάποιες εταιρείες αναθέτουν καθήκοντα που θεωρούνται εκτός του πεδίου των βασικών αρμοδιοτήτων τους σε τρίτους. Η λειτουργία και η συντήρηση των δικτύων υπολογιστών

ανατίθεται από πολλές εταιρείες σε πάροχους δικτύου, καθιστώντας την υποστήριξη των SLAs ένα σημαντικό θέμα στα πλαίσια των δικτύων υπολογιστών.

Ένα SLA θα περιλαμβάνει συνήθως πληροφορίες όπως η περιγραφή της φύσης των υπηρεσιών που πρέπει να παρέχονται, το αναμενόμενο επίπεδο απόδοσης της υπηρεσίας, ειδικά η αξιοπιστία και η ανταπόκριση, η διαδικασία για την αναφορά προβλημάτων με την υπηρεσία, το χρονικό πλαίσιο για τον εντοπισμό και την επίλυση των προβλημάτων, οι συνέπειες για τον πάροχο υπηρεσιών όταν δεν εκπληρώσει τις υποχρεώσεις του και ρήτρες διαφυγής και περιορισμούς. Δεν είναι αναγκαίο όλα τα συστατικά ενός SLA να είναι παρών σε όλες τις συμβάσεις.

SLS Element / Clause	***Description and Attributes***
SLS Identification	A unique identification key (set by service provider).
Network Connectivity Services Req.	
Topology & scope: (pipe, hose, funnel, multicast tree, etc.) and scope (ingress, egress points)	Identifies: the edge points of the topological region over which the QoS applies (IP addresses or layer 2 identifiers); the topology (pipe, hose, etc.). Attributes: Ingress-Egress points (e.g., set of IP addresses of the edge routers); Type of topology.
Connectivity class (guarantees): quantitative / qualitative: delay, jitter, loss, availability	Describes the performance guarantees a provider agrees to offer to the packets entitled to this SLS inside the connectivity class. Attributes: delay, loss, jitter, values.
Bandwidth (capacity)	Depending on the topology the capacity is specified abstracted based on the notion of traffic trunks. This is done in the most general case as a traffic matrix. Attributes: Value of capacity assured between edge points.
Traffic Processing Req.	
Access and transfer rules: Ingress flow Id, Egress flow Id, Ingress point, Egress point	Describe the flows to which the committed treatment is to be done. Attributes: DiffServ Code Points (DSCPs), source, destination, application / content.
Access and transfer rules: QoS Guarantees: Class, (dropping, re-marking, shaping)	The former defined class in the Connectivity Service parameters should be specified.
Access and transfer rules: QoS guarantees: Conformance algorithm	Describes the criteria to decide in-profile or out-profile upon input traffic. Only in-profile traffic can get QoS guarantees specified by Connectivity Guarantee clause. Traffic Control (TC) information is needed to configure the traffic conditioners at the edge router.
Access and transfer rules: QoS Guarantees: Excess traffic treatment	Describes how the excess traffic will be treated: dropping, re-marking, shaping, adapting.
Routing and Forwarding rules	Describes possible constraints on the way to compute the paths and constraints on forwarding.
Security requirements	Describes details on how to apply security services to the flows. Attributes: Levels of security required and associated parameters.
Adaptation requirements	Describe the condition (thresholds, etc.) under which the adaptation is allowed.

Reliability	Describes the allowed figure of non-availability of the service. Attribute: Mean Down-time per year (MDT) etc.
Services Assessment Req.	
Monitoring methodology	Describes a selection of procedures for monitoring tasks.
Monitoring tasks	Describes how the monitoring actions to supervise SLS fulfilment. Specific monitoring tasks: time windows, sampling rules, active / passive procedures, and entities involved.
Notification and Reports	Describes the details of reports and notification: time, level of information aggregation, etc. There will be regular reports or event-triggered notification.
Network Allowed actions	
Availability and service schedule	Describes possible time intervals allowed for service invocation. Attribute: Timetable for delivery planning.
Invocation methods	Optional attribute used if invocation of the services is a separate phase w.r.t. the subscription. It describes the conditions of invocation.
Modification permission; Connectivity services; Traffic Processing; Service Assessment	Describes modification permissions and conditions for the three categories. Attribute: ranges of modifications allowed for the three categories.

Πίνακας 1 – Στοιχεία ενός τυπικού SLA

3 Τεχνολογίες Παροχής Ποιότητας Υπηρεσίας

3.1 Integrated Services

Η αρχιτεκτονική των ενοποιημένων υπηρεσιών (Integrated Services) ορίζει μια σειρά από επεκτάσεις για το παραδοσιακό μοντέλο BE του Διαδικτύου με στόχο να επιτρέψει τη παροχή QoS στις υπηρεσίες από άκρη έως άκρη. Ένα από τα βασικά στοιχεία της αρχιτεκτονικής είναι ένα σύνολο από ορισμούς υπηρεσιών. Το τρέχον σύνολο των υπηρεσιών αποτελείται από ελεγχόμενο φόρτο και εγγυημένες υπηρεσίες. Η αρχιτεκτονική υποθέτει ότι χρησιμοποιείται ένας μηχανισμός για τη μεταφορά πληροφοριών σε δρομολογητές, έτσι ώστε να μπορούν να παρέχουν τις ζητούμενες υπηρεσίες στις ροές που τις χρειάζονται. Ενώ το πρωτόκολλο δέσμευσης πόρων (Resource Reservation Protocol) είναι το πιο γνωστό παράδειγμα ενός τέτοιου μηχανισμού, η IntServ αρχιτεκτονική έχει σχεδιαστεί για να μπορεί να φιλοξενήσει και άλλους μηχανισμούς.

3.1.1 Resource Reservation Protocol

Το RSVP χρησιμοποιείται από ένα τερματικό (host) για να ζητήσει ειδική ποιότητα υπηρεσιών από το δίκτυο για συγκεκριμένες ροές δεδομένων των εφαρμογών. Το RSVP χρησιμοποιείται, επίσης, από τους δρομολογητές για να παραδώσει τις αιτήσεις QoS σε όλους τους κόμβους κατά μήκος της διαδρομής των ροών και να καθιερώσει και να διατηρήσει την κατάσταση ώστε να παρέχει την αιτούμενη υπηρεσία. Οι RSVP αιτήσεις θα εξασφαλίσουν γενικά την διατήρηση των πόρων σε κάθε κόμβο κατά μήκος της διαδρομής των δεδομένων και ζητούν πόρους σε μία μόνο κατεύθυνση. Ως εκ τούτου, το RSVP αντιμετωπίζει έναν αποστολέα ξεχωριστά από ένα δέκτη, αν και η ίδια εφαρμογή μπορεί να λειτουργήσει τόσο ως αποστολέας όσο και ως δέκτης ταυτόχρονα. Λειτουργεί πάνω από το στρώμα δικτύου, καταλαμβάνοντας τη θέση του πρωτοκόλλου μεταφοράς στη στοίβα πρωτοκόλλων. Ωστόσο, δεν μεταφέρει δεδομένα εφαρμογών, αλλά μοιάζει με τα πρωτόκολλα ελέγχου, όπως το ICMP ή κάποια πρωτόκολλα δρομολόγησης. Δεν είναι όμως ούτε ένα πρωτόκολλο δρομολόγησης. Έχει σχεδιαστεί για να λειτουργεί με τα τωρινά και τα μελλοντικά πρωτόκολλα δρομολόγησης και συμβουλεύεται την τοπική βάση δρομολόγησης για να αποκτήσει τις διαδρομές. Τα πρωτόκολλα δρομολόγησης καθορίζουν που θα προωθηθούν τα πακέτα. Το RSVP ενδιαφέρεται μόνο για το QoS των πακέτων.

Η λειτουργία του δρομολογητή που δημιουργεί τα διαφορετικά επίπεδα QoS λέγεται έλεγχος τις κυκλοφορίας (traffic control). Αυτή η λειτουργία υλοποιείται από τρείς μηχανισμούς. Τον χρονοπρογραμματιστή πακέτων (scheduler), τον ταξινομητή (classifier) και τον έλεγχο αποδοχής.

3.1.1.1 Packet Scheduler

Ο χρονοπρογραμματιστής πακέτων διαχειρίζεται τη προώθηση των διαφορετικών ροών των πακέτων χρησιμοποιώντας ένα σύνολο από ουρές και ίσως και άλλους μηχανισμούς όπως χρονόμετρα. Πρέπει να υλοποιηθεί στο σημείο όπου τα πακέτα εισέρχονται στην ουρά. Η βασική λειτουργία του είναι να αναδιατάσσει την ουρά εξόδου. Η απλούστερη προσέγγιση είναι ένα σύστημα προτεραιότητας, κατά το οποίο τα πακέτα μπαίνουν σε σειρά προτεραιότητας και τα πακέτα με την μεγαλύτερη προτεραιότητα φεύγουν πρώτα. Αυτή η προσέγγιση δίνει απόλυτη προτεραιότητα σε κάποια πακέτα έναντι άλλων, που σημαίνει ότι αν υπάρχουν αρκετά πακέτα υψηλής προτεραιότητας, τα πακέτα χαμηλής προτεραιότητας ενδέχεται να μην καταφέρουν να φύγουν ποτέ. Μια εναλλακτική προσέγγιση είναι να δίνεται σε όλα τα πακέτα ένα μέρος του εύρους ζώνης, ανάλογα με την προτεραιότητα τους.

3.1.1.2 Packet Classifier

Για τον σκοπό του ελέγχου της κυκλοφορίας, κάθε εισερχόμενο πακέτο πρέπει να αντιστοιχίζονται σε κάποια κατηγορία. Όλα τα πακέτα στην ίδια κατηγορία λαμβάνουν την ίδια μεταχείριση από το χρονοπρογραμματιστή. Αυτή η χαρτογράφηση εκτελείται από τον ταξινομητή. Η επιλογή μιας κατηγορίας μπορεί να βασίζεται στο περιεχόμενο της υπάρχουσας κεφαλίδας των πακέτων ή / και ένα πρόσθετο αριθμό ταξινόμησης που έχει προστεθεί σε κάθε πακέτο. Μια κατηγορία μπορεί να αντιστοιχεί σε πολλές ροές, όπως όλες οι ροές βίντεο ή όλες οι ροές που αναλογούν σε ένα συγκεκριμένο δίκτυο. Από την άλλη πλευρά, μία ροή μπορεί να αντιστοιχεί σε μόνο μία κατηγορία. Μια κατηγορία είναι μια αφηρημένη έννοια που μπορεί να είναι τοπική σε ένα συγκεκριμένο δρομολογητή. Το ίδιο πακέτο, δηλαδή, μπορεί να ταξινομηθεί διαφορετικά από διαφορετικούς δρομολογητές κατά μήκος της διαδρομής. Για παράδειγμα, οι δρομολογητές κορμού μπορεί να επιλέξουν να χαρτογραφήσουν πολλές ροές σε λίγες συγκεντρωτικές κατηγορίες, ενώ οι δρομολογητές πιο κοντά στην περιφέρεια, όπου υπάρχει πολύ λιγότερη συμφόρηση, μπορεί να χρησιμοποιήσουν μια ξεχωριστή κατηγορία για κάθε ροή.

3.1.1.3 Admission Control

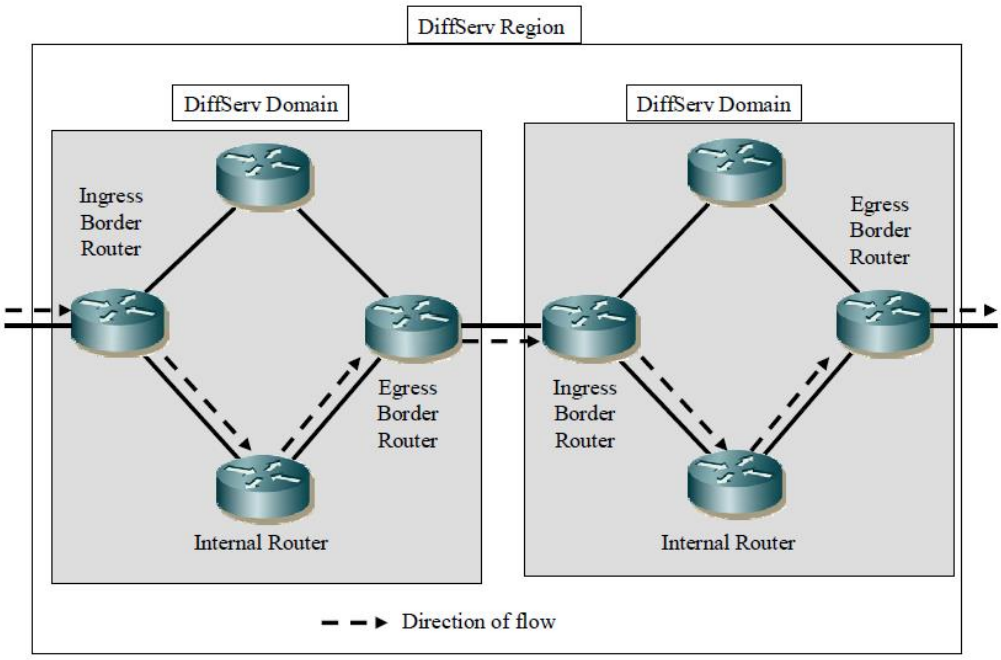
Ο έλεγχος αποδοχής υλοποιεί έναν αλγόριθμο που χρησιμοποιούν οι δρομολογητές ή τα τερματικά για να καθορίσουν αν μπορούν να χορηγηθούν οι αιτούμενες υπηρεσίες σε μια νέα ροή, χωρίς να επηρεάζονται οι προηγούμενες εγγυήσεις. Ο έλεγχος αποδοχής συντελείται σε κάθε κόμβο για να πάρει μια τοπική απόφαση αποδοχής / απόρριψης, τη στιγμή που ένας κόμβος ζητήσει μία υπηρεσία κατά μήκος ενός μονοπατιού του δικτύου. Εκτός από την εξασφάλιση των εγγυήσεων, θα πρέπει να ασχολείται με την επιβολή διαχειριστικών πολιτικών στις δεσμεύσεις πόρων. Μερικές πολιτικές ίσως απαιτούν έλεγχο ταυτότητας εκείνων που ζητούν δεσμεύσεις.

3.2 Differentiated Services

Η αρχιτεκτονική των διαφοροποιημένων υπηρεσιών (Differentiated Services) βασίζεται σε ένα απλό μοντέλο όπου η κίνηση που εισέρχεται σε ένα δίκτυο ταξινομείται και ενδεχομένως κατατάσσεται στα όρια του δικτύου και ανατίθεται σε διαφορετικά σύνολα συμπεριφορών (Behavior Aggregates). Κάθε BA προσδιορίζεται από ένα μοναδικό κωδικοσημείο (DSCP). Εντός του πυρήνα του δικτύου, τα πακέτα προωθούνται σύμφωνα με την ανά-κόμβο συμπεριφορά (Per-Hop Behavior) που σχετίζεται με το DSCP.

3.2.1 Περιοχή, Τομείς και Κόμβοι

Μια DS περιοχή (region) είναι ένα σύνολο από ένα ή περισσότερους συνεχόμενους DS τομείς. Οι περιοχές είναι σε θέση να υποστηρίζουν διαφοροποιημένες υπηρεσίες κατά μήκος των διαδρομών που συνδέουν τους τομείς εντός της περιοχή. Οι τομείς σε μια περιοχή μπορεί να υποστηρίζουν εσωτερικά διαφορετικές ομάδες PHB και διαφορετικές αντιστοιχίσεις DSCP σε PHB. Ωστόσο, για να επιτραπούν υπηρεσίες που εκτείνονται σε όλους τους τομείς, οι τομείς πρέπει να καθιερώσουν ένα SLA το οποίο ορίζει, είτε ρητά είτε εμμέσως, μία συμφωνία διαχείρισης κίνησης (Traffic Conditioning Agreement) η οποία καθορίζει τον τρόπο με τον οποίο η κίνηση διαχειρίζεται στα άκρα των δύο τομέων. Είναι πιθανό πολλοί τομείς μέσα σε μια περιοχή να υιοθετήσουν μία κοινή πολιτική παροχής υπηρεσίας και να υποστηρίξουν ένα κοινό σύνολο ομάδων PHB, εξαλείφοντας έτσι την ανάγκη για διαχείριση της κυκλοφορίας μεταξύ των διαφορετικών τομέων.



Εικόνα 1 - DS περιοχή με δύο τομείς

Ένας DS τομέας (domain) είναι μια συνεχόμενη σειρά DS κόμβων (nodes) που λειτουργούν με κοινή πολιτική παροχής υπηρεσίας και κοινό σύνολο ομάδων PHB,

εφαρμοσμένων σε κάθε κόμβο. Ένας τομέας έχει ένα καλά καθορισμένο όριο που αποτελείται από συνοριακούς (border) κόμβους που μπορούν να συνδέουν διαφορετικούς τομείς μεταξύ τους και διαχειρίζονται την εισερχόμενη κυκλοφορία για να εξασφαλιστεί ότι τα πακέτα που διέρχονται από τον τομέα είναι κατάλληλα σηματοδοτημένα ώστε να επιλεγεί μία PHB από μία από τις ομάδες που υποστηρίζονται. Κόμβοι εντός του τομέα (internal) συνδέονται μόνο με άλλους συνοριακούς ή εντός του τομέα κόμβους. Επιλέγουν τη συμπεριφορά προώθησης για τα πακέτα με βάση το DSCP τους και χαρτογραφούν την τιμή σε μία από τις υποστηριζόμενες PHBs, χρησιμοποιώντας είτε τη συνιστώμενη χαρτογράφηση ή μία τοπικά προσαρμοσμένη.

3.2.2 Κατηγορίες Υπηρεσιών

Μια κατηγορία υπηρεσιών αντιπροσωπεύει ένα σύνολο κίνησης που απαιτεί ειδικά χαρακτηριστικά (π.χ., καθυστέρηση, απώλειες, κ.λπ.) από το δίκτυο. Εννοιολογικά, μια κατηγορία υπηρεσιών αναφέρετε σε εφαρμογές με παρόμοια χαρακτηριστικά και απαιτήσεις επιδόσεων, όπως μία κατηγορία υψηλού εύρους ζώνης για εφαρμογές όπως η μεταφορά αρχείων και το ηλεκτρονικό ταχυδρομείο, ή μία κατηγορία χαμηλής καθυστέρησης για κίνηση πραγματικού χρόνου όπως το βίντεο και οι υπηρεσίες τηλεφωνίας. Ενώ οι διαφοροποιημένες υπηρεσίες είναι μια γενική αρχιτεκτονική που μπορεί να χρησιμοποιηθεί για να εφαρμόσει μια ποικιλία από υπηρεσίες, έχουν οριστεί τρεις θεμελιώδεις κατηγορίες οι οποίες προορίζονται για γενική χρήση.

3.2.2.1 Προώθηση Βέλτιστης Προσπάθειας

Η BE προώθηση συνήθως ρυθμίζετε με κάποια εγγύηση εύρους ζώνης. Τα πακέτα υπό διέλευση μπορεί να χαθούν, να αλλαχθεί η σειρά τους, να διπλοτυπηθούν ή να καθυστερήσουν τυχαία. Σε γενικές γραμμές, τα δίκτυα έχουν κατασκευαστεί ώστε να περιορίζουν αυτή τη συμπεριφορά, αλλά η αλλαγή στο φόρτο τους μπορεί να ωθήσει οποιοδήποτε δίκτυο σε μια τέτοια κατάσταση. Η κίνηση των εφαρμογών στο διαδίκτυο που χρησιμοποιούν την BE προώθηση αναμένεται να είναι ελαστική, δηλαδή ότι ο αποστολέας της κίνησης θα είναι σε θέση να ρυθμίσει το ρυθμό μετάδοσης της σε απάντηση στις αλλαγές σε διαθέσιμο εύρος ζώνης, απώλειες ή καθυστέρηση. Για τη BE προώθηση παρέχεται μια μόνο τιμή DSCP για να προσδιοριστεί η κίνηση και μια ουρά (queue) για να αποθηκευτεί. Απαιτείται η ενεργός διαχείριση της ουράς για την προστασία του δικτύου και τον περιορισμό των καθυστερήσεων.

3.2.2.2 Εξασφαλισμένη Προώθηση

Η εξασφαλισμένη προώθηση (Assured Forwarding) προορίζεται για δίκτυα που προσφέρουν SLAs με μέτριο ρυθμό μετάδοσης (π.χ., Frame Relay, ATM). Είναι, ουσιαστικά, μια βελτιωμένη BE. Η κίνηση αναμένεται και πάλι να είναι ελαστική. Ο δέκτης θα ανιχνεύει τις απώλειες ή την μεταβολή της καθυστέρησης στο δίκτυο και θα παρέχει ανατροφοδότηση (feedback), έτσι ώστε ο αποστολέας να ρυθμίσει την ταχύτητα μετάδοσης για να προσεγγίσει τη διαθέσιμη χωρητικότητα. Για την AF προώθηση

παρέχονται πολλαπλές DSCP τιμές (12 σε σύνολο, 4 κλάσεις με 3 προτεραιότητες απόρριψης η καθεμία) για τον προσδιορισμό της κίνησης, μια κοινή ουρά για να αποθηκεύετε το σύνολο και απαιτείται η ενεργός διαχείριση της ουράς για την προστασία του δικτύου και τον περιορισμό των καθυστερήσεων. Η κίνηση μετριέται καθώς εισέρχεται στο δίκτυο, και σηματοδοτείται διαφορετικά ανάλογα με το ρυθμό άφιξης του συνόλου. Η παραδοχή είναι ότι είναι φυσιολογικό για χρήστες κατά καιρούς να χρησιμοποιούν μεγαλύτερη χωρητικότητα (capacity) από ότι ορίζει η σύμβασή τους, ίσως μέχρι κάποιο όριο. Ωστόσο, εάν χρειαστεί να απορριφθεί κίνηση για να διαχειριστεί η ουρά, αυτή η περίσσεια κίνηση θα απορριφθεί πρώτη.

	AF1	**AF2**	**AF3**	**AF4**
Low Drop Pr.	001010	010010	011010	100010
Medium Drop Pr.	001100	010100	011100	100100
High Drop Pr.	001110	010110	011110	100110

Πίνακας 2 – Τυπικές DSCP τιμές για την AF

3.2.2.3 Επισπευσμένη Προώθηση

Η πρόθεση της επισπευσμένης προώθησης (Expedited Forwarding) είναι να παρέχει ένα δομικό στοιχείο για υπηρεσίες χαμηλής απώλειας, χαμηλής καθυστέρησης καθώς και χαμηλής διακύμανσης της καθυστέρησης. Μπορεί να χρησιμοποιηθεί για την κατασκευή μιας ενισχυμένης BE, όπου η κίνηση παραμένει ευάλωτη σε απώλειες που οφείλονται σε σφάλματα γραμμής και αναδιάταξη πακέτων κατά τη διάρκεια αλλαγών στη δρομολόγηση. Ωστόσο, χρησιμοποιώντας τεχνικές των ουρών, η πιθανότητα της καθυστέρησης ή η διακύμανση στη καθυστέρηση ελαχιστοποιείται. Για το λόγο αυτό, η EF γενικά χρησιμοποιείται για να μεταφέρει ανελαστικές εφαρμογές πραγματικού χρόνου, όπως η φωνή ή το βίντεο, οι οποίες στέλνουν πακέτα με το ρυθμό που παράγονται από τον κωδικοποιητή, ανεξάρτητα από τη διαθέσιμη χωρητικότητα. Αυτές οι εφαρμογές έχουν τη δυνατότητα να υπερχειλίσουν ή να διακόψουν τη λειτουργία ενός δικτύου αν δεν ελεγχθούν. Για την προστασία του δικτύου, θα πρέπει να ελέγχετε η κίνηση σε διάφορα σημεία για να διασφαλιστεί ότι δεν έχει υπερβεί η χωρητικότητα των ουρών. Τότε η κίνηση θα πρέπει να εισέρχεται σε μία ουρά μικρής καθυστέρησης για να εξασφαλιστεί ότι η διακύμανση στην καθυστέρηση δεν θα δημιουργήσει προβλήματα, ώστε να καλυφθούν οι ανάγκες της εφαρμογής .

3.3 Multi-Protocol Label Switching (MPLS)

Όταν ένα πακέτο από ένα ασυνδεσιστρεφές (connectionless) πρωτόκολλο του στρώματος δικτύου ταξιδεύει από ένα δρομολογητή στον επόμενο, κάθε δρομολογητής παίρνει μια ανεξάρτητη απόφαση προώθησης για το συγκεκριμένο πακέτο. Δηλαδή, κάθε δρομολογητής αναλύει την επικεφαλίδα του πακέτου και κάθε δρομολογητής τρέχει έναν αλγόριθμο δρομολόγησης του στρώματος δικτύου. Κάθε δρομολογητής επιλέγει ανεξάρτητα έναν επόμενο κόμβο για το πακέτο, με βάση την ανάλυση της επικεφαλίδας του πακέτου και τα αποτελέσματα της λειτουργίας του αλγόριθμου δρομολόγησης.

Οι επικεφαλίδες των πακέτων περιέχουν πολύ περισσότερες πληροφορίες από ό, τι απαιτείται απλά για να επιλεχθεί ο επόμενος κόμβος. Η επιλογή του επόμενου κόμβου μπορεί να θεωρηθεί ως η σύνθεση δύο λειτουργιών. Η πρώτη λειτουργία χωρίζει ολόκληρο το σύνολο των πιθανών πακέτων σε ένα σύνολο κλάσεων ισοδύναμης προώθησης (Forwarding Equivalence Classes) και η δεύτερη χαρτογραφεί κάθε FEC σε έναν επόμενο κόμβο. Όσον αφορά την απόφαση προώθησης, διαφορετικά πακέτα τα οποία αντιστοιχίζονται στην ίδια FEC είναι δυσδιάκριτα. Όλα τα πακέτα που ανήκουν σε μια συγκεκριμένη FEC και που ταξιδεύουν από ένα συγκεκριμένο κόμβο θα ακολουθήσουν τον ίδιο δρόμο.

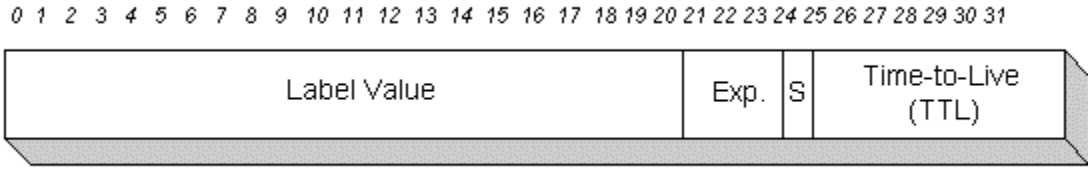
Στο MPLS, η αντιστοίχιση ενός συγκεκριμένου πακέτου σε μια συγκεκριμένη FEC γίνεται μόνο μία φορά, καθώς το πακέτο εισέρχεται στο δίκτυο. Η FEC στην οποία έχει αντιστοιχηθεί το πακέτο, κωδικοποιείται ως μια σύντομη τιμή σταθερού μήκους γνωστή ως ετικέτα (label). Όταν ένα πακέτο προωθείται στον επόμενο κόμβο, η ετικέτα στέλνεται μαζί με αυτό. Στους επόμενους κόμβους, δεν αναλύεται περαιτέρω η επικεφαλίδα του στρώματος δικτύου του πακέτου. Αντίθετα, η ετικέτα χρησιμοποιείται ως δείκτης σε ένα πίνακα, που προσδιορίζει τον επόμενο κόμβο και μια νέα ετικέτα. Η παλιά ετικέτα αντικαθίσταται με τη νέα ετικέτα, και το πακέτο προωθείται στον επόμενο κόμβο. Αυτό έχει έναν αριθμό πλεονεκτημάτων έναντι της συμβατικής προώθησης του στρώματος δικτύου:

- Η προώθηση μπορεί να γίνει από μηχανήματα (π.χ., Switches) τα οποία είναι ικανά να αναζητήσουν και να αντικαταστήσουν την ετικέτα, αλλά είτε δεν είναι σε θέση να αναλύσουν καθόλου τις επικεφαλίδες του στρώματος δικτύου ή δεν είναι ικανά να το κάνουν με επαρκή ταχύτητα.
- Δεδομένου ότι ένα πακέτο έχει αντιστοιχηθεί σε μια FEC όταν εισέρχεται στο δίκτυο, ο δρομολογητής εισόδου μπορεί να χρησιμοποιήσει κάθε πληροφορία που έχει σχετικά με το πακέτο, ακόμα και αν οι πληροφορίες αυτές δεν προέρχονται από την επικεφαλίδα του στρώματος δικτύου. Για παράδειγμα, πακέτα που φθάνουν σε διαφορετικές θύρες μπορούν να ανατεθούν σε διαφορετικές FECs. Στη συμβατική προώθηση, από την άλλη πλευρά, μπορούν να εξεταστούν μόνο οι πληροφορίες που ταξιδεύουν με το πακέτο στην επικεφαλίδα του.
- Ένα πακέτο που εισέρχεται στο δίκτυο από έναν συγκεκριμένο δρομολογητή μπορεί να καταταχθεί σε διαφορετική FEC από ότι αν έμπαινε από κάποιον άλλο και ως αποτέλεσμα, αποφάσεις προώθησης που εξαρτώνται από το δρομολογητή εισόδου μπορούν να παρθούν / αλλάξουν εύκολα. Αυτό δεν μπορεί να γίνει στη συμβατική προώθηση, δεδομένου ότι η ταυτότητα του δρομολογητή εισόδου ενός πακέτου δεν ταξιδεύει με το πακέτο.
- Η διαδικασίες που καθορίζουν τον τρόπο με τον οποίο ένα πακέτο αντιστοιχείται σε μια FEC μπορούν να γίνουν όλο και πιο περίπλοκες, χωρίς αντίκτυπο σε όλους τους δρομολογητές που απλώς προωθούν επισημασμένα πακέτα.
- Μερικές φορές είναι επιθυμητό ένα πακέτο να ακολουθήσει μία συγκεκριμένη διαδρομή η οποία ρητά επιλέγεται κατά ή πριν από τη στιγμή την οποία το πακέτο εισέρχεται στο δίκτυο, αντί να επιλεγεί από τον κανονικό, δυναμικό αλγόριθμο δρομολόγησης, όπως το πακέτο ταξιδεύει μέσω του δικτύου. Στη

συμβατική προώθηση, αυτό απαιτεί το πακέτο να φέρει μαζί του μια κωδικοποίηση της διαδρομής του.

Μερικοί δρομολογητές αναλύουν την επικεφαλίδα του στρώματος δικτύου ενός πακέτου όχι μόνο για να επιλέξουν τον επόμενο κόμβο, αλλά και για να καθορίσουν την προτεραιότητα (precedence) ή την κατηγορία υπηρεσίας (class of service) του πακέτου. Μπορούν στη συνέχεια να εφαρμόσουν διαφορετικά όρια απόρριψης (discard) ή χρονοπρογραμματισμό (schedule) σε διαφορετικά πακέτα.

3.3.1 Επικεφαλίδα

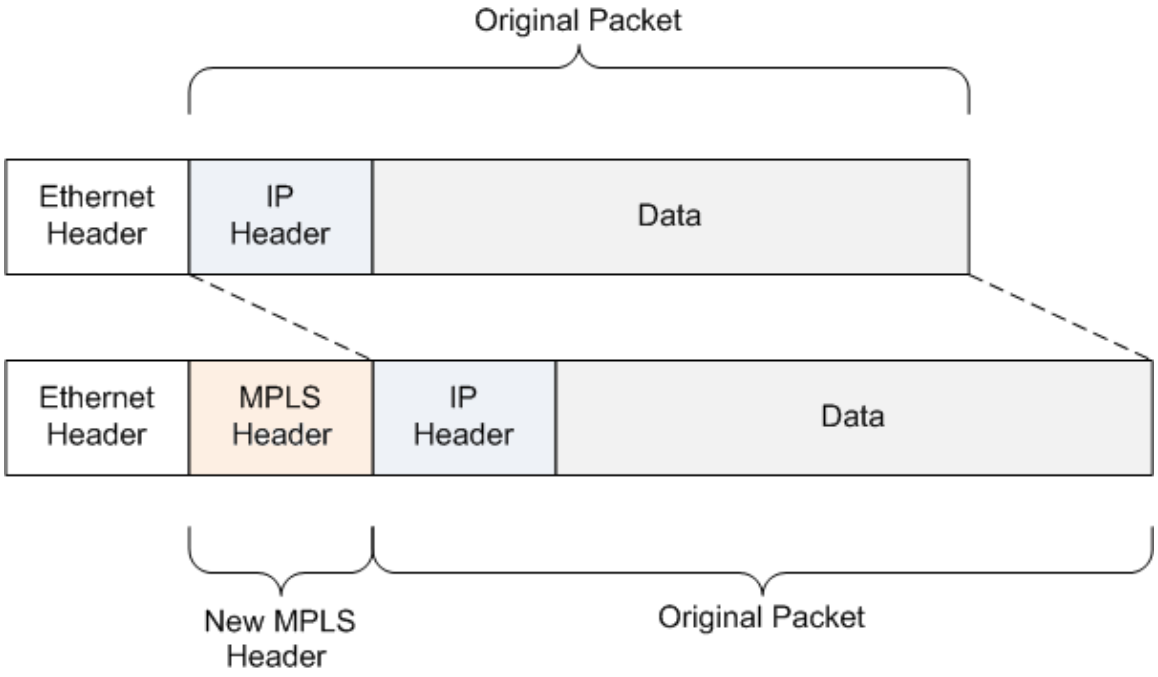


Εικόνα 2 - Δομή MPLS επικεφαλίδας

Η MPLS επικεφαλίδα αποτελείται από 4 πεδία και έχει μήκος 32 bit / 4 byte (Εικόνα 2). Εμφανίζεται μετά τις επικεφαλίδες του στρώματος ζεύξης δεδομένων, αλλά πριν από κάθε επικεφαλίδα του επιπέδου δικτύου (Εικόνα 3). Σε κάποιες περιπτώσεις είναι χρήσιμο να υπάρχει ένα πιο γενικό μοντέλο, κατά το οποίο ένα πακέτο φέρει μια σειρά από ετικέτες, οργανωμένες ως μία στοίβα με προτεραιότητα τελευταία-μέσα, πρώτη-έξω (Last-In, First-Out) και ονομάζετε στοίβα ετικετών. Η επικεφαλίδα του στρώματος δικτύου ακολουθεί αμέσως μετά την ετικέτα, η οποία έχει το πεδίο S ορισμένο. Τα πεδία της MPLS επικεφαλίδας είναι τα εξής:

- **Bottom of Stack (S)**
  Αυτό το πεδίο παίρνει την τιμή 1 για την τελευταία ετικέτα της στοίβας και 0 για όλες τις υπόλοιπες.
- **Time to Live (TTL)**
  Το πεδίο αυτό οριοθετεί το χρόνο ζωής του πακέτου. Έχει μήκος 8 bit και χρησιμεύει στο να απορρίπτονται πακέτα που για διάφορους λόγους περιφέρονται άσκοπα στο δίκτυο. Ουσιαστικά είναι ένας μετρητής. Όταν ένα πακέτο φτάσει σε έναν δρομολογητή, ο δρομολογητής μειώνει το πεδίο TTL κατά 1. Όταν μηδενιστεί, το πακέτο απορρίπτεται.
- **Experimental Use (EXP)**
  Αυτό το πεδίο των 3 bit αρχικά είχε ως στόχο να μεταφέρει πληροφορίες για την κατηγορία της υπηρεσίας. Η ακριβής χρήση του δεν συμφωνήθηκε και ορίστηκε για πειραματική χρήση.
- **Label Value**
  Αυτή το 20-bit πεδίο μεταφέρει την πραγματική τιμή της ετικέτας. Όταν ένα πακέτο παραλαμβάνεται, αναζητάτε η τιμή της ετικέτας στην κορυφή της στοίβας. Ως αποτέλεσμα της επιτυχούς αναζήτησης μπορεί να βρεθεί ο επόμενος κόμβος στον οποίο το πακέτο πρέπει να προωθηθεί όπως και η

πράξη που πρόκειται να πραγματοποιηθεί στην ετικέτα πριν τη προώθηση. Αυτή η λειτουργία μπορεί να περιλαμβάνει την αντικατάσταση της ετικέτας που βρίσκεται στη κορυφή της στοίβας με μια άλλη, ή την αφαίρεση (pop) μίας καταχώρησης από την στοίβα ετικετών, ή την αντικατάσταση της ετικέτας και στη συνέχεια την ώθηση (push) μίας ή περισσότερων καταχωρήσεων στη στοίβα ετικετών. Επιπρόσθετα, μπορεί να βρεθεί η ενθυλάκωση του στρώματος ζεύξη δεδομένων και ενδεχομένως άλλες πληροφορίες που απαιτούνται, προκειμένου να διαβιβάσει σωστά το πακέτο.



Εικόνα 3 - Εισαγωγή MPLS επικεφαλίδας στο πακέτο

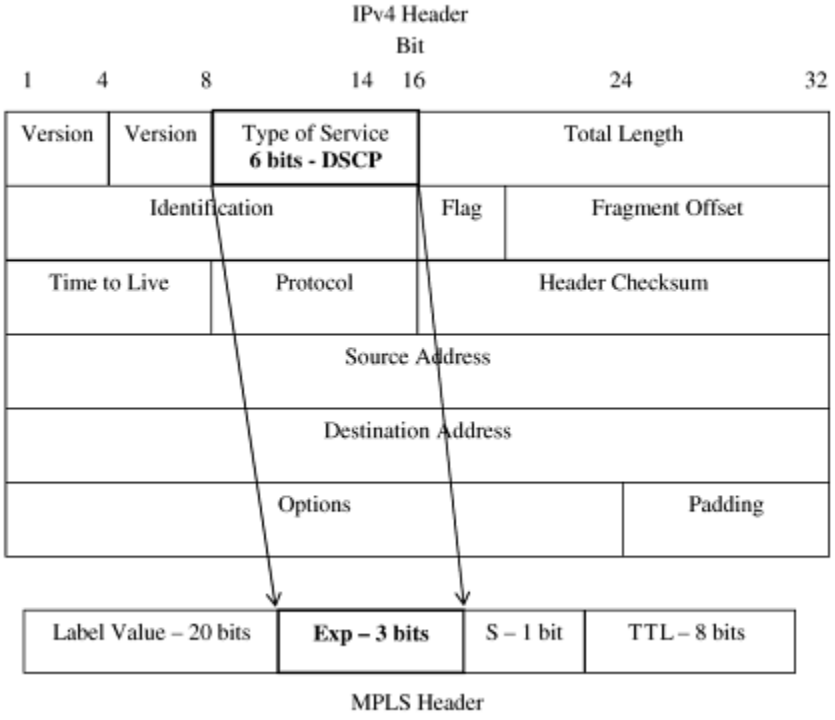
3.4 MPLS over DiffServ

Σε ένα MPLS δίκτυο, όταν μία ροή δεδομένων ακολουθεί μία κοινή διαδρομή μπορεί να εγκατασταθεί ένα μονοπάτι (Label Switched Path) χρησιμοποιώντας πρωτόκολλα σηματοδότησης. Ο δρομολογητής εισόδου (Label Edge Router) προσθέτει μία επικεφαλίδα σε κάθε πακέτο και το προωθεί στον επόμενο κόμβο. Στον κάθε επόμενο δρομολογητή κατά μήκος του LSP, η ετικέτα χρησιμοποιείται για να προωθηθεί το πακέτο στον επόμενο κόμβο.

Σε ένα DiffServ τομέα, όλα τα πακέτα που διασχίζουν μια διαδρομή και απαιτούν την ίδια μεταχείριση συνιστούν μία BA. Στον κόμβο εισόδου του DiffServ τομέα, τα πακέτα ταξινομούνται και μαρκάρονται με μία DSCP τιμή, η οποία αντιστοιχεί στη BA. Σε κάθε κόμβο διέλευσης, το DSCP χρησιμοποιείται για την επιλογή της PHB, που καθορίζει τον χρονοπρογραμματισμό και σε ορισμένες περιπτώσεις, την πιθανότητα απόρριψης για κάθε πακέτο.

Για την υποστήριξη του MPLS πάνω από το DiffServ, το DSCP πεδίο της επικεφαλίδας του στρώματος δικτύου αντιστοιχίζεται με το EXP πεδίο της MPLS επικεφαλίδας ή ενσωματώνεται μέσα στην επικεφαλίδα του στρώματος ζεύξης δεδομένων. Αυτό είναι απαραίτητο επειδή οι MPLS δρομολογητές δεν εξετάζουν την επικεφαλίδα του στρώματος δικτύου κατά τη διάρκεια της προώθησης. Έχουν οριστεί δύο τύποι LSP.

3.4.1 EXP-inferred LSP



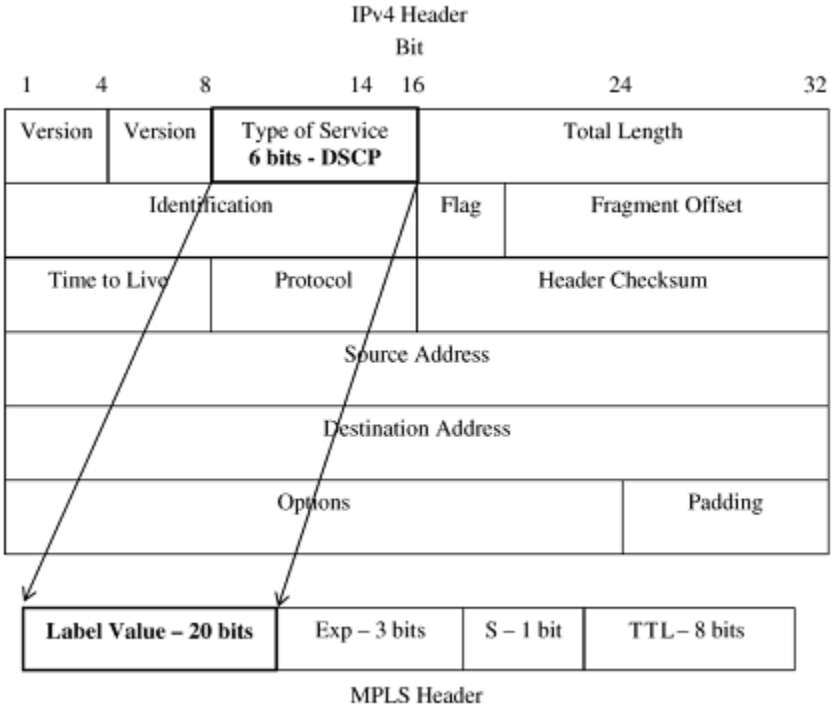
Εικόνα 4 - EXP-inferred αντιστοίχιση

Ένα LSP μπορεί να χρησιμοποιηθεί για να υποστηρίξει ένα ή περισσότερα σύνολα από BAs που μοιράζονται περιορισμούς διάταξης (Ordered Aggregate). Αυτά τα LSPs μπορούν να υποστηρίξουν μέχρι και οκτώ BAs μίας δεδομένης FEC. Το πεδίο EXP της MPLS επικεφαλίδας χρησιμοποιείται από τους δρομολογητές για τον προσδιορισμό της PHB που πρέπει να εφαρμοστεί στα πακέτα. Η αντιστοίχιση του πεδίου EXP στην PHB για ένα δεδομένο LSP, είτε σηματοδοτείται ρητά με την εισαγωγή της ετικέτας ή βασίζετε σε μία προκαθορισμένη αντιστοίχιση.

3.4.2 Label-inferred LSP

Ένα ξεχωριστό LSP μπορεί να καθοριστεί για ένα μόνο ζεύγος OA, FEC. Οι PHBs σηματοδοτούνται ρητά κατά το χρόνο της εισαγωγής της ετικέτας, έτσι ώστε οι εσωτερικοί δρομολογητές (Label Switched Router) να μπορούν να συμπεράνουν

αποκλειστικά από την τιμή της ετικέτας την PHB που πρέπει να εφαρμοστεί σε ένα πακέτο. Η προτεραιότητα απόρριψης που εφαρμόζεται από τον LSR στο πακέτο, μεταφέρεται στο εσωτερικό της MPLS επικεφαλίδας χρησιμοποιώντας το πεδίο EXP. Όταν η επικεφαλίδα δεν χρησιμοποιείται (π.χ., MPLS Over ATM), η προτεραιότητα απόρριψης μεταφέρεται στο εσωτερικό της επικεφαλίδας του στρώματος ζεύξης δεδομένων χρησιμοποιώντας συγκεκριμένα πεδία προτεραιότητας απόρριψης (π.χ., ATM CLP).



Εικόνα 5 - Label-inferred αντιστοίχιση

3.5 Δίκτυα Επίγνωσης Περιεχομένου

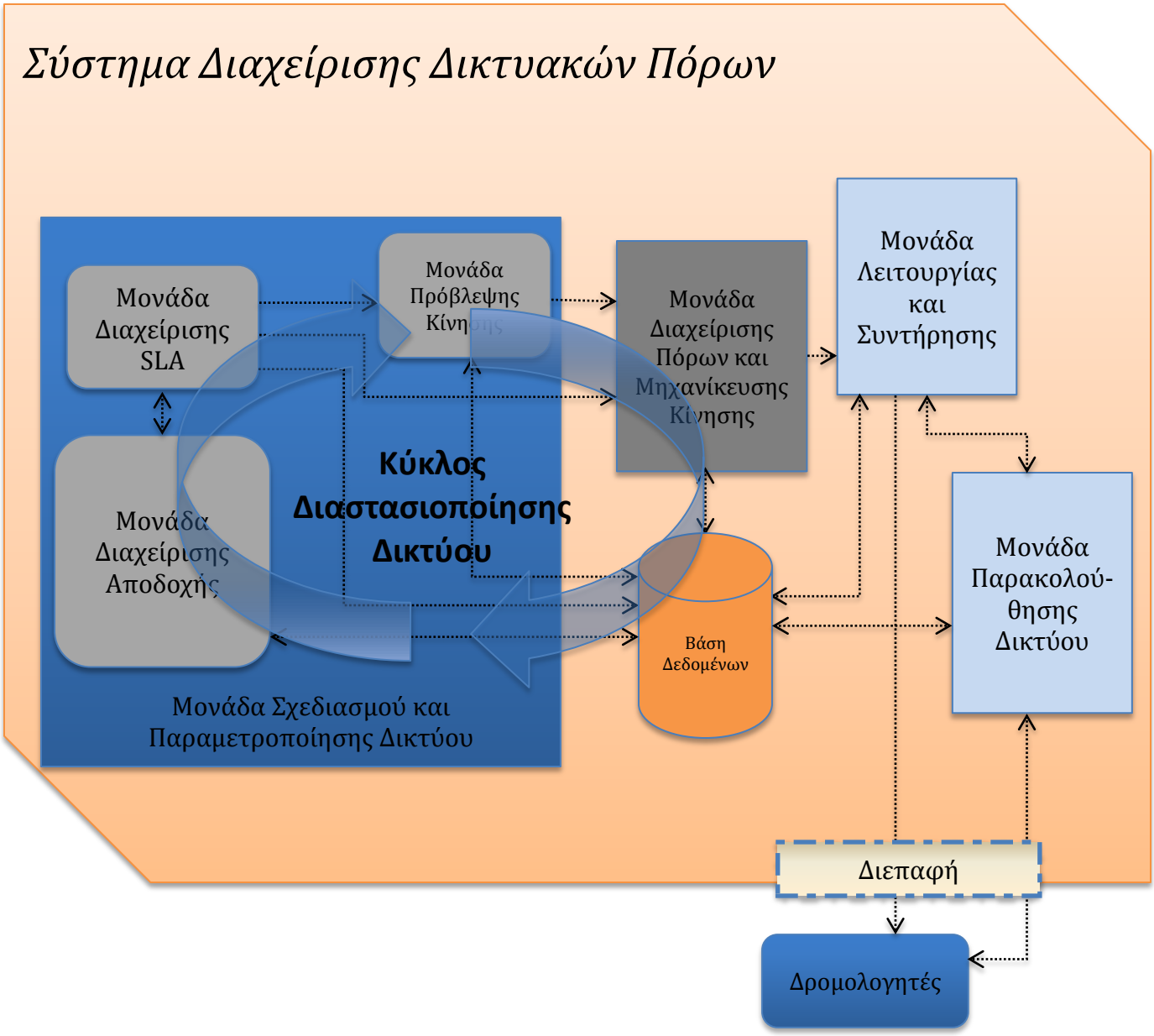
Τα δίκτυα επίγνωσης περιεχομένου (CAN) αποτελούν μια νέα, ολοκληρωμένη αρχιτεκτονική, που έχει ως στόχο την μέγιστη εκμετάλλευση των διαθέσιμων δικτυακών πόρων στο επίπεδο της μεταφοράς δεδομένων. Ο όρος επίγνωση περιεχομένου προτείνει μια νέα λειτουργία κατά την οποία οι δρομολογητές θα είναι ικανοί να επιθεωρούν τα πακέτα των ροών που εισέρχονται σε αυτά και να αναγνωρίζουν το είδος της εφαρμογής που μεταφέρουν. Έπειτα, με βάση αυτή την πληροφορία και αξιολογώντας τις δικτυακές απαιτήσεις των εφαρμογών αυτών (π.χ., εύρος ζώνης, καθυστέρηση, κ.λπ.), οι διαχειριστές δικτύου θα δύνανται να δημιουργούν εικονικά δίκτυα που αρχικά θα διασφαλίζουν κλιμακωτά επίπεδα παροχής ποιότητας υπηρεσίας και εν συνεχεία θα δρομολογούν την κίνηση σε αυτά, ανάλογα με τις απαιτήσεις σε δικτυακούς πόρους. Μερικές από τις κύριες λειτουργίες των CANs παραθέτονται παρακάτω:

- Αναγνώριση του τύπου του περιεχομένου των ροών στο επίπεδο δικτύου από τους δρομολογητές. Αυτή η λειτουργία επιτελείται είτε αυτόματα (εξετάζοντας όλα τα πακέτα που διέρχονται από αυτούς) είτε χρησιμοποιώντας συγκεκριμένη σηματοδοσία από τα υψηλότερα στρώματα (για μερική επιθεώρηση πακέτων). Για το σκοπό αυτό χρησιμοποιούνται γνωστές μέθοδοι, στηριζόμενες είτε σε ευριστικές τεχνικές (heuristics) είτε σε τεχνικές μηχανικής μάθησης (machine learning).
- Κατηγοριοποίηση και διαφοροποίηση υπηρεσιών με βάση το περιεχόμενο. Η λειτουργία αυτή επιτυγχάνεται χρησιμοποιώντας γνωστές δικτυακές τεχνολογίες παροχής ποιότητας υπηρεσίας (IntServ, DiffServ, MPLS, κ.λπ.). Ως αποτέλεσμα, παρέχεται η δυνατότητα δημιουργίας ευέλικτων συμβολαίων παροχής υπηρεσιών (SLAs).
- Επεξεργασία κίνησης με βάση το περιεχόμενο. Η λειτουργία αυτή περιλαμβάνει διαδικασίες όπως η ταξινόμηση (classification) σε κατηγορίες, το μαρκάρισμα (marking) για κατανομή σε κλάσεις, η μορφοποίηση (shaping) και η απόρριψη (dropping) της κίνησης .
- Δρομολόγηση και προώθηση με βάση το περιεχόμενο. Σε αυτή τη λειτουργία δημιουργούνται εικονικά από άκρο-έως-άκρο μονοπάτια με την έννοια των παράλληλων δικτύων (parallel internets), με στόχο την κλιμακωτή χρησιμοποίηση / διαχείριση των δικτυακών πόρων ανάλογα με τις απαιτήσεις.

4 Σύστημα Διαχείρισης Δικτυακών Πόρων

4.1 Αρχιτεκτονική

Όλα τα συστήματα διαχείρισης δικτυακών πόρων σχεδιάζονται με σκοπό την παροχή καλύτερων υπηρεσιών στους τελικούς χρήστες ενός δικτύου και τα περισσότερα από αυτά τα συστήματα χρησιμοποιούν την ίδια βασική δομή. Για να επιτραπεί και να γίνει σωστή διαχείριση ενός δικτύου, πρέπει το σύστημα να παρέχει κάποιες βασικές λειτουργίες:



Εικόνα 6 – Αρχιτεκτονική συστήματος διαχείρισης δικτυακών πόρων

- **Σχεδιασμός και παραμετροποίηση**
  Αρχικά, πρέπει να προβλεφθεί η κίνηση του δικτύου. Αυτή η διαδικασία μπορεί να γίνει συνήθως βασιζόμενη στις υπάρχουσες και στις αναμενόμενες προδιαγραφές παροχής υπηρεσίας (Service Level Specification) ή βασιζόμενη σε ιστορικά δεδομένα χρήσης του δικτύου. Είναι μία απαραίτητη διαδικασία για τις λειτουργίες της μηχανίκευσης κίνησης αφού εκτιμά τους διαθέσιμους πόρους. Έπειτα πρέπει να διαπραγματευτούν τα συμβόλαια μεταξύ των χρηστών και των πάροχων υπηρεσιών. Σκοπός αυτής της διαπραγμάτευσης είναι: α) η θέσπιση παραμέτρων QoS ανάμεσα στους χρήστες και τους πάροχους και β) η μέγιστη εκμετάλλευση των διαθέσιμων πόρων εξασφαλίζοντας μόνο την πραγματική ζήτηση ανά πάσα στιγμή. Η διαπραγμάτευση γίνεται βάση των διαθέσιμων πόρων μέσω ενός πρωτοκόλλου διαπραγμάτευσης.
- **Διαχείριση πόρων και μηχανίκευση κίνησης**
  Σκοπός αυτής της λειτουργίας είναι η ρύθμιση του δικτύου με τέτοιο τρόπο ώστε να πληρούνται οι απαιτήσεις των συμφωνημένων SLSs. Ο διαμοιρασμός της κίνησης στο δίκτυο γίνεται βάση των διαθέσιμων πόρων και των QoS περιορισμών, όσο το δυνατόν πιο αποτελεσματικά. Για να επιτευχθεί αυτό, χρησιμοποιούνται συνήθως πρωτόκολλα όπως το RSVP για τη δέσμευση των πόρων του δικτύου, το MPLS για τη σηματοδότηση και τη δρομολόγηση της κίνησης και το DiffServ για την αστυνόμευση και την ταξινόμηση της κίνησης σε κατηγορίες. Τέλος, η κίνηση μπορεί να προσαρμοστεί (π.χ., χαμηλότερη ανάλυση βίντεο) ανάλογα με τις ανάγκες.
- **Παρακολούθηση**
  Η παρακολούθηση τις κίνησης του δικτύου παρέχει πληροφορίες για τις επιδόσεις του δικτύου. Αυτό μπορεί να περιλαμβάνει το διαθέσιμο εύρος ζώνης μίας ζεύξης, τα χαμένα πακέτα, την καθυστέρηση των πακέτων κ.λπ. Οι πληροφορίες που συγκεντρώνονται είναι χρήσιμες για τον καθορισμό του κατά πόσο οι στόχοι απόδοσης του δικτύου ικανοποιούνται. Το πιο γνωστό πρωτόκολλο παρακολούθησης δικτύων είναι το SNMP, το οποίο στέλνει ερωτήματα στους δρομολογητές από ένα κεντρικό σύστημα διαχείρισης για να λάβει τις πληροφορίες.
- **Λειτουργία και συντήρηση**
  Αυτή η λειτουργία διασφαλίζει τη καλή λειτουργία του δικτύου και των συστημάτων που το απαρτίζουν. Επιτρέπει τον εντοπισμό, την καταγραφή, τη διάγνωση και ίσως τη διόρθωση προβλημάτων. Βλάβες στο δίκτυο μπορεί να προκαλέσουν απαράδεκτη υποβάθμιση του δικτύου και είναι σημαντικό να μπορούν να απομονωθούν.
- **Βάση δεδομένων**
  Στη βάση δεδομένων αποθηκεύονται πληροφορίες σχετικά με το δίκτυο, όπως οι IP των κόμβων, η τοπολογία του δικτύου, το διαθέσιμο εύρος ζώνης των ζεύξεων, κ.λπ. Όλες οι μονάδες ενός συστήματος διαχείρισης δικτυακών πόρων επικοινωνούν άμεσα με τη βάση δεδομένων, είτε για να αποθηκεύσουν, είτε για να πάρουν πληροφορίες.

4.2 Σχεδίαση Συστήματος

Στα πλαίσια αυτής της πτυχιακής εργασίας αναπτύχθηκε ένα σύστημα διαχείρισης δικτυακών πόρων (ΣΔΔΠ) που ασχολείται κυρίως με τη μηχανίκευση της κίνησης. Το σύστημα μπορεί να χωριστεί σε 3 βασικές μονάδες και οι λειτουργίες της καθεμίας παρουσιάζονται παρακάτω.

4.2.1 Μονάδα Μηχανίκευσης Κίνησης

Είναι η κεντρική μονάδα του συστήματος διαχείρισης δικτυακών πόρων και είναι υπεύθυνη για τη διαχείριση του υποκείμενου δικτύου. Παρέχει τέσσερις βασικές λειτουργίες διαχείρισης:

- **Διαχείριση MPLS**
  Με αυτή τη λειτουργία ο διαχειριστής του δικτύου μπορεί να συνδεθεί στους δρομολογητές και να προσθέσει / αφαιρέσει συγκεκριμένα LSPs ή να ρυθμίσει τις παραμέτρους τους για να φιλοξενήσει την αναμενόμενη κίνηση. Μπορεί, επίσης, να δει πια LSP είναι εγκατεστημένα σε κάθε δρομολογητή.
- **Διαχείριση DiffServ**
  Παρέχει τρόπους διαχείρισης των υποστηριζόμενων κατηγοριών υπηρεσιών. Επιτρέπει στο διαχειριστή να συνδεθεί σε όλους τους δρομολογητές του υποκείμενου δικτύου και να θέσει όλες τις υποστηριζόμενες κατηγορίες υπηρεσιών με προκαθορισμένες παραμέτρους ή να αφαιρέσει όλες τις κατηγορίες υπηρεσιών.
- **Υποστήριξη MPLS over DiffServ**
  Επιτρέπει την υποστήριξη του MPLS πάνω από το DiffServ αντιστοιχίζοντας, ανά LSP, το DSCP πεδίο της επικεφαλίδας του στρώματος δικτύου με το EXP πεδίο της MPLS επικεφαλίδας (Πίνακας 3).
- **Διαχείριση SLA**
  Ο κύριος ρόλος αυτής της λειτουργίας είναι να λαμβάνει αιτήματα διαχείρισης των SLAs από τον πάροχο υπηρεσιών (Service Provider). Ένα τέτοιο αίτημα περιέχει κανόνες αστυνόμευσης (policing) της κίνησης. Αφού λάβει ένα αίτημα, η μονάδα συνδέετε στο δρομολογητή εισόδου (ingress router), σηματοδοτεί την κίνηση και επιβάλλει τους κανόνες.

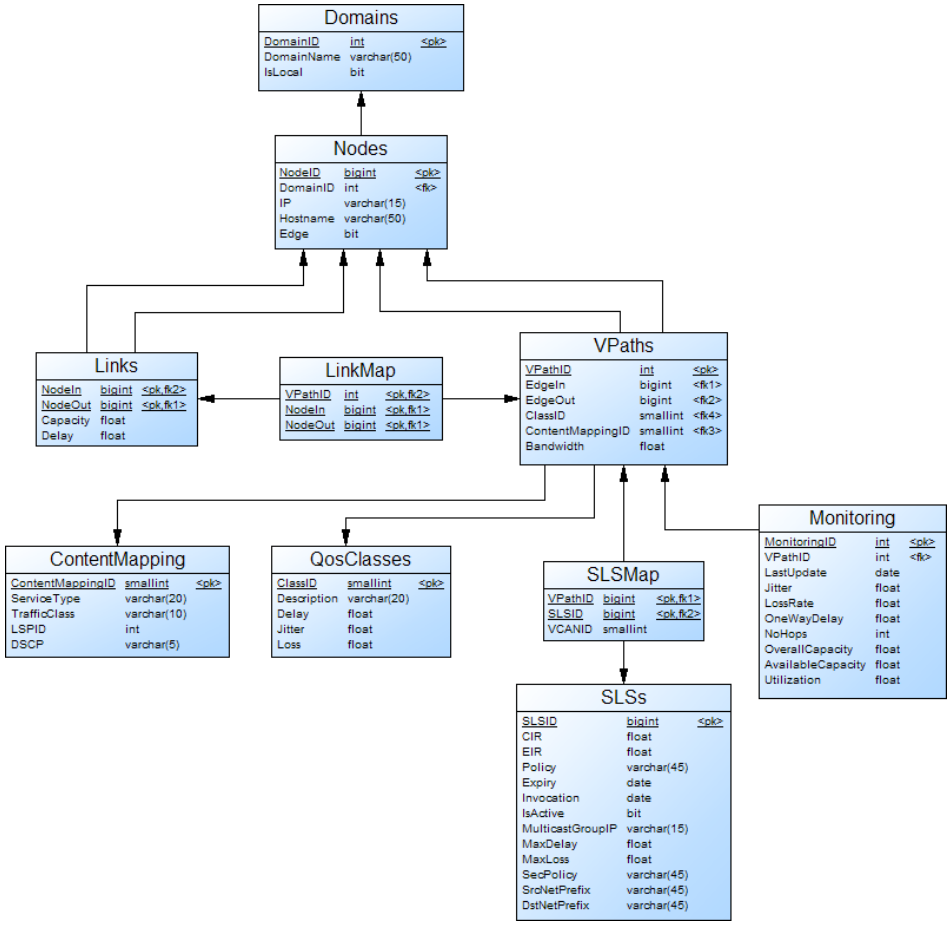
Class	DSCP	EXP	ToS
BE	0x00	0	0x00
EF	0x2e	1	0xb8
AF11	0x0a	2	0x28
AF12	0x0c	3	0x30
AF21	0x12	4	0x48
AF22	0x14	5	0x50

Πίνακας 3 – Αντιστοιχίσεις

Τέλος, παρέχετε η δυνατότητα αναίρεσης όλων των αλλαγών και επαναφοράς του δικτύου στην αρχική του κατάσταση.

4.2.2 Μονάδα Διαχείρισης Βάσης Δεδομένων

Η συγκεκριμένη μονάδα είναι υπεύθυνη για την διαχείριση των ζωτικών πληροφοριών που σχετίζονται με τους πόρους του δικτύου. Μπορεί να εξάγει ή να εισάγει πληροφορίες στη βάση δεδομένων, ανάλογα με τη λειτουργία της μονάδας διαχείρισης πόρων. Περιέχει πληροφορίες όπως τα διαθέσιμα LSPs, την τοπολογία του δικτύου, την χωρητικότητα των ζεύξεων και τις IP των κόμβων (Εικόνα 7).



Εικόνα 7 – Διάγραμμα πινάκων της βάσης δεδομένων

4.2.3 Μονάδα Διασύνδεσης

Αυτή η μονάδα είναι στην ουσία μία μονάδα-δαίμονας (daemon), η οποία ανοίγει ένα TCP socket και αναμένει για εισερχόμενες συνδέσεις. “Τρέχει” στο παρασκήνιο (background) όλων των δρομολογητών του δικτύου πυρήνα (core network) και επιτρέπει στη μονάδα διαχείρισης πόρων να συνδεθεί στους δρομολογητές.

4.3 Εργαλεία Υλοποίησης

Παρακάτω παρουσιάζονται τα βασικά εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του ΣΔΔΠ.

4.3.1 Γλώσσα Προγραμματισμού

Για την υλοποίηση του ΣΔΔΠ χρησιμοποιήθηκε η Python, μια υψηλού επιπέδου γλώσσα προγραμματισμού ηλεκτρονικών υπολογιστών με χαρακτηριστικά που την καθιστούν κατάλληλη για γενική χρήση. Η γλώσσα είναι διαδραστική και αντικειμενοστραφής και σχεδιάστηκε για να είναι σαφής και κατανοητή. Πρώτα κυκλοφόρησε το 1991 και δημιουργήθηκε για να χρησιμοποιεί ελάχιστες εντολές. Το λογισμικό ανοικτού κώδικα είναι χτισμένο γύρω από μια σειρά αγγλικών λέξεων και δεν απαιτεί τη χρήση σημείων στίξης. Για παράδειγμα, όταν άλλες γλώσσες χρησιμοποιούν παρενθέσεις, η Python χρησιμοποιεί εσοχές. Απαιτεί, επίσης, λιγότερο συγκεκριμένη σύνταξη.

4.3.2 Βάση Δεδομένων

Για τη δημιουργία της βάσης δεδομένων χρησιμοποιήθηκε η MySQL. Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Μια σχεσιακή βάση δεδομένων αποθηκεύει δεδομένα σε ξεχωριστούς πίνακες, αντί να τα τοποθετεί όλα σε μία μεγάλη αποθήκη. Οι δομές της βάσης οργανώνονται σε φυσικά αρχεία βελτιστοποιημένα για ταχύτητα. Χρησιμοποιεί την SQL (Structed Query Language), την τυπική γλώσσα ερωτημάτων για βάσεις δεδομένων. Τα κυριότερα πλεονεκτήματά της είναι η απόδοση, το ελάχιστο έως μηδενικό κόστος, η ευκολία στη χρήση της και η μεταφερσιμότητα.

4.3.3 Λογισμικό

4.3.3.1 TC

Το TC (Traffic Control) χρησιμοποιήθηκε για τη ταξινόμηση της κίνησης. Επιτρέπει τη δημιουργία κατηγοριών υπηρεσιών με διαφορετικά χαρακτηριστικά (π.χ., μέτριο εύρος ζώνης, χαμηλή καθυστέρηση, μεγάλη προτεραιότητα απόρριψης, κ.λπ.) για τη διαμόρφωση και τη κατηγοριοποίηση της κίνησης.

4.3.3.2 MPLS

Το MPLS χρησιμοποιήθηκε για την εγκατάσταση εικονικών μονοπατιών πάνω στο πραγματικό δίκτυο. Τα μονοπάτια επιτρέπουν την ομαλή δρομολόγηση της κίνησης από κόμβο σε κόμβο βάση μίας ετικέτας, χωρίς την περίπλοκη αναζήτηση IP για κάθε πακέτο. Χρησιμοποιήθηκε, ακόμα, για την υποστήριξη του MPLS over DiffServ.

4.3.3.3 Iptables

Το Iptables είναι ένα εργαλείο φιλτραρίσματος πακέτων. Το φιλτράρισμα γίνεται βάση των πληροφοριών που κουβαλάνε μαζί τους τα πακέτα στην επικεφαλίδα. Χρησιμοποιήθηκε για το μαρκάρισμα της κίνησης.

4.3.3.4 Tcpdump

Το tcpdump είναι ένας ανιχνευτής / αναλυτής πακέτων. Παρέχει τη δυνατότητα εμφάνισης των πακέτων στην οθόνη κάθε χρονική στιγμή ή την καταγραφή τους στο σκληρό δίσκο για περαιτέρω ανάλυση.

4.3.3.5 Iperf

Το iperf είναι μια γεννήτρια κίνησης. Μπορεί να δημιουργήσει TCP και UDP ροές δεδομένων και να μετρήσει την απόδοση ενός δικτύου που τις μεταφέρει.

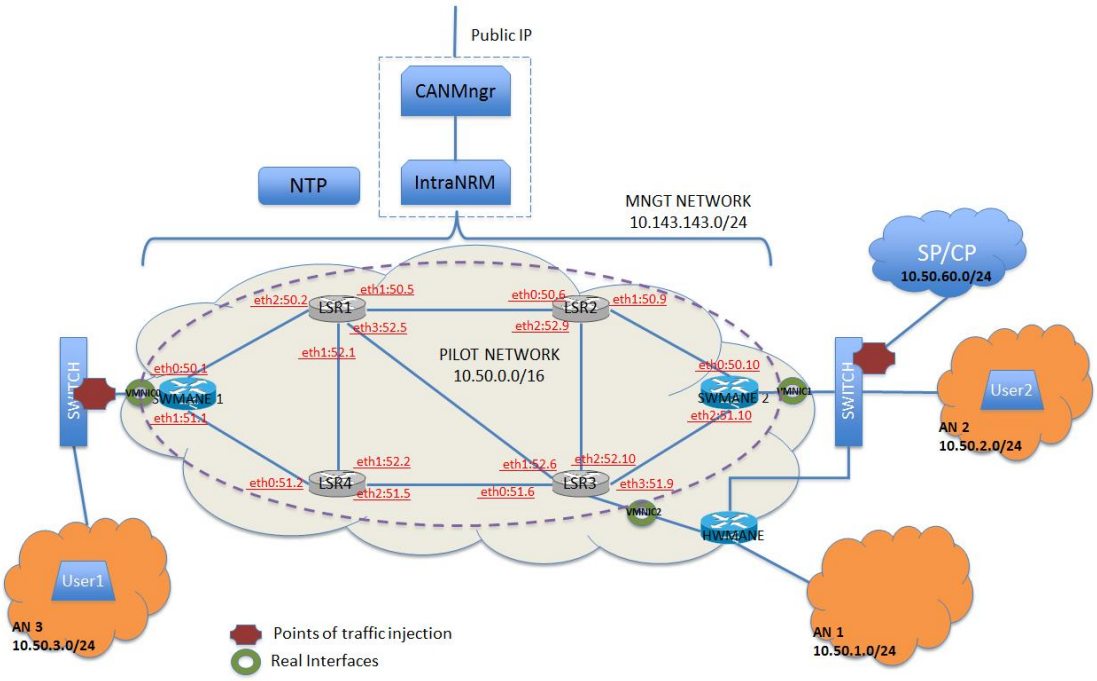
4.3.4 Διεπαφές

Για την επικοινωνία του ΣΔΔΠ με τους δρομολογητές του δικτύου χρησιμοποιήθηκαν stream sockets. Το socket είναι μια αφαιρετική έννοια, μέσω της οποίας μια εφαρμογή μπορεί να στείλει και να λάβει δεδομένα, με τον ίδιο τρόπο που ένα ανοιχτό αρχείο επιτρέπει σε μια εφαρμογή να διαβάζει και να γράφει δεδομένα σε ένα σταθερό μέσο αποθήκευσης. Το socket επιτρέπει σε μια εφαρμογή να συνδεθεί σε ένα δίκτυο και να επικοινωνήσει με άλλες εφαρμογές οι οποίες επίσης είναι συνδεδεμένες στο ίδιο δίκτυο. Τα stream sockets χρησιμοποιούν το TCP ως πρωτόκολλο μεταφοράς και με αυτό τον τρόπο παρέχουν αξιόπιστες υπηρεσίες μεταφοράς δεδομένων.

5 Πειραματικό Μέρος

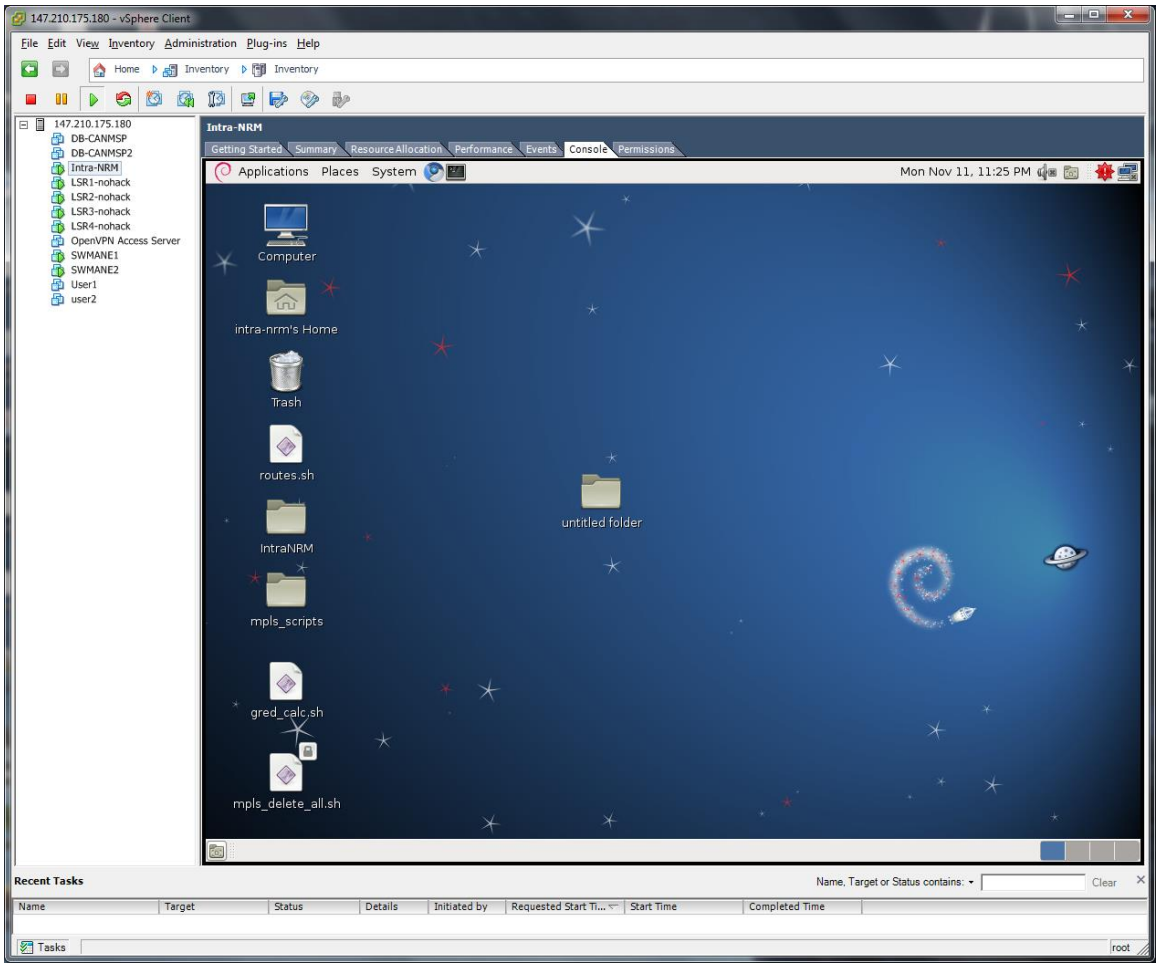
5.1 Δημιουργία Εικονικού Δικτύου

Για τις δοκιμές του συστήματος διαχείρισης δικτυακών πόρων δημιουργήθηκε στο εργαστήριο "ΠΑΣΙΦΑΗ" του Τ.Ε.Ι Κρήτης ένα εικονικό δίκτυο που προσομοιώνει ένα αυτόνομο σύστημα (Autonomous System) και περιλαμβάνει 4 δρομολογητές κορμού (LSR1, LSR2, LSR3 και LSR4) και 3 δρομολογητές άκρων (SWMANE1, SWMANE2 και HWMANE). Κάθε δρομολογητής άκρων συνδέετε με ένα δίκτυο πρόσβασης (AN1, AN2 και AN3) τα οποία περιέχουν έναν αριθμό χρηστών. Το ΣΔΔΠ (IntraNRM) συνδέεται απευθείας σε όλους τους δρομολογητές μέσω μιας διεπαφής. Η εν λόγω πλατφόρμα έχει υλοποιηθεί στα πλαίσια του ερευνητικού έργου ALICANTE [1].



Εικόνα 8 – Η πλατφόρμα δοκιμών

Όλα τα προαναφερθέντα μηχανήματα, "στήθηκαν" στην πλατφόρμα διαχείρισης εικονικών μηχανών ESXi (Εικόνα 9). Μια εικονική μηχανή (Virtual Machine) είναι ένας εικονικός υπολογιστής, δηλαδή μία υλοποίηση ενός πραγματικού υπολογιστή χρησιμοποιώντας κατάλληλο λογισμικό που εκτελεί προγράμματα όπως ένα φυσικό μηχάνημα. Για τα VMs χρησιμοποιήθηκε το λειτουργικό σύστημα Linux. Συγκεκριμένα, για όλους τους δρομολογητές χρησιμοποιήθηκε η διανομή Debian, χωρίς γραφικό περιβάλλον, με μία ενημερωμένη έκδοση του πυρήνα 2.6.35 για την υποστήριξη του MPLS. Για τους χρήστες και το ΣΔΔΠ χρησιμοποιήθηκε η διανομή Debian, χωρίς και με γραφικό περιβάλλον αντίστοιχα, με τον πυρήνα 3.2.52.



Εικόνα 9 – Το περιβάλλον εργασίας του ESXi

5.2 Υποστηριζόμενες Λειτουργίες του ΣΔΔΠ

5.2.1 Αρχικές Ρυθμίσεις

Για την αποτελεσματική χρήση του ΣΔΔΠ είναι απαραίτητο να γίνουν κάποιες αρχικές ρυθμίσεις που αφορούν την κατάσταση (σχεδίαση / παραμετροποίηση) του δικτύου:

- **Αρχικοποίηση Βάσης Δεδομένων**
  Πληροφορίες όπως οι IP των διεπαφών των δρομολογητών, τα ονόματά τους και το αν βρίσκονται στο κέντρο ή στα άκρα του δικτύου είναι απαραίτητες για την επικοινωνία. Πρέπει, μεταξύ άλλων, να συμπληρωθούν πληροφορίες σχετικά με τα διαθέσιμα LSP και τις υποστηριζόμενες DiffServ κλάσεις.
- **Αρχικοποίηση Επικοινωνίας με τους δρομολογητές**
  Όπως προαναφέρθηκε, για να διαχειριστεί το δίκτυο ο ΣΔΔΠ θα πρέπει να επικοινωνήσει με τους δρομολογητές. Έτσι, η μονάδα διασύνδεσης σε κάθε

δρομολογητή θα πρέπει να μπει σε κατάσταση "ακοής". Αυτό γίνεται ξεκινώντας ένα δαίμονα σε κάθε δρομολογητή, εκτελώντας την εντολή ***>python ManeModule.py start***. Υποστηρίζεται επίσης η διακοπή της λειτουργίας ή η επανεκκίνηση του δαίμονα με τις εντολές ***>python ManeModule.py stop*** και ***>python ManeModule.py restart*** αντίστοιχα.

5.2.2 Λειτουργία Διαχείρισης MPLS

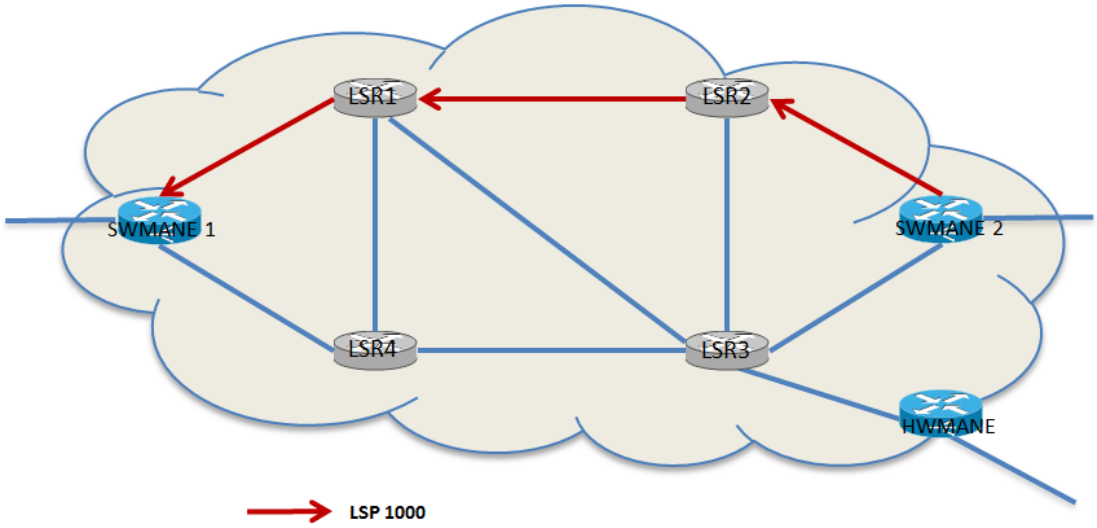
Αφού έχει συμπληρωθεί η βάση δεδομένων και οι δαίμονες σε όλους τους δρομολογητές είναι σε κατάσταση "ακοής", μπορεί να ξεκινήσει η διαχείριση του δικτύου. Το πρώτο βήμα είναι η εγκατάσταση των LSPs. Εκτελώντας την εντολή ***>python intraNRM.py setPath label***, όπου ***label*** η τιμή της ετικέτας (π.χ., 1000), το ΣΔΔΠ θα ελέγξει αν η βάση δεδομένων περιέχει πληροφορίες που αφορούν το LSP με ετικέτα τον αριθμό 1000. Στη περίπτωση που το LSP δεν υποστηρίζεται, επιστρέφει στην οθόνη ένα μήνυμα λάθους. Αντίθετα, στη περίπτωση που το LSP υποστηρίζεται, εξάγει τις πληροφορίες που χρειάζεται από τη βάση δεδομένων, συνδέεται στους κατάλληλους δρομολογητές και εκτελεί τις εντολές εγκατάστασης του LSP (Εικόνα 10).

```
root@intra-nrm:/home/intra-nrm/Desktop/MyIntraNRM# python intraNRM.py setPath 1000
10.143.143.3
LABELSPACE entry dev eth0 labelspace 0
NHLFE entry key 0x000003e8 mtu 1492 propagate_ttl proto static
        push gen 1000 set eth0 10.50.50.9

10.143.143.12
LABELSPACE entry dev eth1 labelspace 0
LABELSPACE entry dev eth0 labelspace 0
NHLFE entry key 0x000003e8 mtu 1496 propagate_ttl proto static
        set eth0 10.50.50.5
ILM entry label gen 1000 labelspace 0 proto static
        forward key 0x000003e8

10.143.143.11
LABELSPACE entry dev eth1 labelspace 0
LABELSPACE entry dev eth2 labelspace 0
NHLFE entry key 0x000003e8 mtu 1496 propagate_ttl proto static
        set eth2 10.50.50.1
ILM entry label gen 1000 labelspace 0 proto static
        forward key 0x000003e8

10.143.143.2
LABELSPACE entry dev eth0 labelspace 0
NHLFE entry key 0x000003e8 mtu 65535 propagate_ttl proto static
        pop peek
ILM entry label gen 1000 labelspace 0 proto static
        forward key 0x000003e8
```

Εικόνα 10 - Εγκατάσταση του LSP με label 1000

Η κατάργηση ενός LSP γίνεται με την εκτέλεση της εντολής ***>python intraNRM.py clearPath label***, όπου ***label*** η τιμή της ετικέτας. Η διαδικασία που

ακολουθείται είναι αντίστοιχη αυτής της εγκατάστασης. Τέλος, μπορεί να γίνει έλεγχος των υποστηριζόμενων LSP εκτελώντας την εντολή ***>python intraNRM.py getInfo***.



Εικόνα 11 - Η διαδρομή του LSP με label 1000

5.2.3 Λειτουργία Διαχείρισης DiffServ

Στη συνέχεια πρέπει να εγκατασταθούν οι DiffServ κλάσεις. Οι κλάσεις που υποστηρίζονται είναι οι BE, EF, AF1 και AF2, με τις AF να έχουν από δύο προτεραιότητες απόρριψης έκαστη (AF11, AF12, AF21, AF22). Εκτελώντας την εντολή ***>python intraNRM.py setClass default***, εγκαθίστανται όλες οι υποστηριζόμενες κλάσεις (Εικόνα 12) σε όλες τις διεπαφές όλων των δρομολογητών του δικτύου. Το συνολικό εύρος ζώνης έχει οριστεί στα 100Mbit και κάθε κλάση λαμβάνει από 20Mbit, εκτός της EF που λαμβάνει 40Mbit. Εκτελώντας την εντολή ***>python intraNRM.py setClass delete***, διαγράφονται όλες οι κλάσεις.

```
root@swmane2:~# tc -s qdisc show dev eth0
qdisc dsmark 1: root refcnt 3 indices 0x0040 default_index 0x0161
 Sent 11247 bytes 71 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc htb 2: parent 1: r2q 10 default 0 direct_packets_stat 0
 Sent 11247 bytes 71 pkt (dropped 0, overlimits 4 requeues 0)
 backlog 0b 0p requeues 0
qdisc gred 5: parent 2:10
 DP:1 (prio 1) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)  ewma 7 Plog 26 Scell_log 16
 DP:2 (prio 2) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)  ewma 7 Plog 25 Scell_log 16
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc gred 6: parent 2:20
 DP:1 (prio 1) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)  ewma 7 Plog 25 Scell_log 17
 DP:2 (prio 2) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)  ewma 6 Plog 23 Scell_log 16
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 3: parent 2:50 limit 10p
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc red 4: parent 2:60 limit 60Kb min 15Kb max 45Kb
 Sent 11247 bytes 71 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
  marked 0 early 0 pdrop 0 other 0
```

Εικόνα 12 - Οι εγκατεστημένες DiffServ κλάσεις[2] στον SWMANE2

Κατόπιν, υποστηρίζεται η αντιστοίχιση του DSCP πεδίου της επικεφαλίδας του στρώματος δικτύου με το EXP πεδίο της MPLS επικεφαλίδας (MPLS over DiffServ). Οι αντιστοιχίσεις είναι προκαθορισμένες (Πίνακας 3) και γίνονται ανά LSP, με την εκτέλεση της εντολής >***python intraNRM.py mpls2ds label***, όπου ***label*** η τιμή της ετικέτας (Εικόνα 13).

[2] 5: AF1, 6: AF2, 3: EF, 4: BE

```
root@intra-nrm:/home/intra-nrm/Desktop/MyIntraNRM# python intraNRM.py mpls2ds 1000
10.143.143.3
NHLFE entry key 0x000003e8 mtu 1492 propagate_ttl proto static
        ds2exp mask->0x0f | 0..1->00000000 | | 2->00000004 | | 3->00000000 | | 4->00000005 | | 5..9->00000000 | | 10->0000000
2 | | 11->00000000 | | 12->00000003 | | 13->00000000 | | 14->00000001 | | 15->00000021 | | 16->00000000 | | 17->00000002 | |
18->00000004 | | 19->00000000 | | 20->00000005 | | 21..25->00000000 | | 26->00000002 | | 27->00000000 | | 28->00000003 | | 29
->00000000 | | 30->00000001 | | 31..33->00000000 | | 34->00000004 | | 35->00000000 | | 36->00000005 | | 37..41->00000000 | |
42->00000002 | | 43->00000000 | | 44->00000003 | | 45->00000000 | | 46->00000001 | | 47..49->00000000 | | 50->00000004 | | 51
->00000000 | | 52->00000005 | | 53..57->00000000 | | 58->00000002 | | 59->00000000 | | 60->00000003 | | 61->00000000 | | 62->
00000001 | | 63->00000000 |
        push gen 1000 exp2tc | 63..0->00000000 | | 1->000000b8 | | 2->00000028 | | 3->00000030 | | 4->00000048 | | 5->0000005
0 | | 6..7->00000000 |
        set eth0 10.50.50.9

10.143.143.12
NHLFE entry key 0x000003e8 mtu 1496 propagate_ttl proto static
        exp2tc | 0->00000000 | | 1->000000b8 | | 2->00000028 | | 3->00000030 | | 4->00000048 | | 5->00000050 | | 6..7->000000
00 |
        set eth0 10.50.50.5

10.143.143.11
NHLFE entry key 0x000003e8 mtu 1496 propagate_ttl proto static
        exp2tc | 0->00000000 | | 1->000000b8 | | 2->00000028 | | 3->00000030 | | 4->00000048 | | 5->00000050 | | 6..7->000000
00 |
        set eth2 10.50.50.1

10.143.143.2
NHLFE entry key 0x000003e8 mtu 65535 propagate_ttl proto static
        exp2tc | 0->00000000 | | 1->000000b8 | | 2->00000028 | | 3->00000030 | | 4->00000048 | | 5->00000050 | | 6..7->000000
00 |
        pop peek
```

Εικόνα 13 - Αντιστοίχιση DSCP με EXP για το LSP με label 1000

5.2.4 Λειτουργία Διαχείρισης SLA

Τέλος, η κίνηση του δικτύου πρέπει να μαρκαριστεί και να αστυνομευθεί. Για αυτό το σκοπό, το ΣΔΔΠ πρέπει να μπει σε κατάσταση "ανάλυσης", εκτελώντας την εντολή >***python intraNRM.py parse***. Τα SLAs ορίζονται από τον SP, ο οποίος στέλνει στο ΣΔΔΠ αιτήματα για τη διαχείριση τους.

VCAN_ID	CMD	SRC_NET	DST_NET	LSP_ID	CIR	PIR	CoS
32	C	10.50.2.0/24	10.50.3.0/24	1000	4	6	EF
33	C	10.50.3.0/24	10.50.2.0/24	1001	4	6	EF
34	D	10.50.3.0/24	10.50.1.0/24	2000	2	4	BE

Πίνακας 4 – Παράδειγμα αιτήματος από τον SP

Αφού επεξεργαστεί το αίτημα το ΣΔΔΠ, ενημερώνει τη βάση δεδομένων και συνδέετε στο κατάλληλο δρομολογητή για την επιβολή των προδιαγραφών. Αν, για παράδειγμα, φτάσει στο ΣΔΔΠ ένα αίτημα όπως αυτό του Πίνακα 4, για τη πρώτη καταχώρηση θα συνδεθεί στον SWMANE2, θα κατατάξει όλη την κίνηση που προέρχεται από το δίκτυο 10.50.2.0/24 με προορισμό το δίκτυο 10.50.3.0/24 ως EF με δεσμευμένο εύρος ζώνης 4Mbit, μέγιστο εύρος ζώνης 6Mbit και θα την προωθήσει μέσω του LSP με label 1000.

5.2.5 Δρομολόγηση Κίνησης

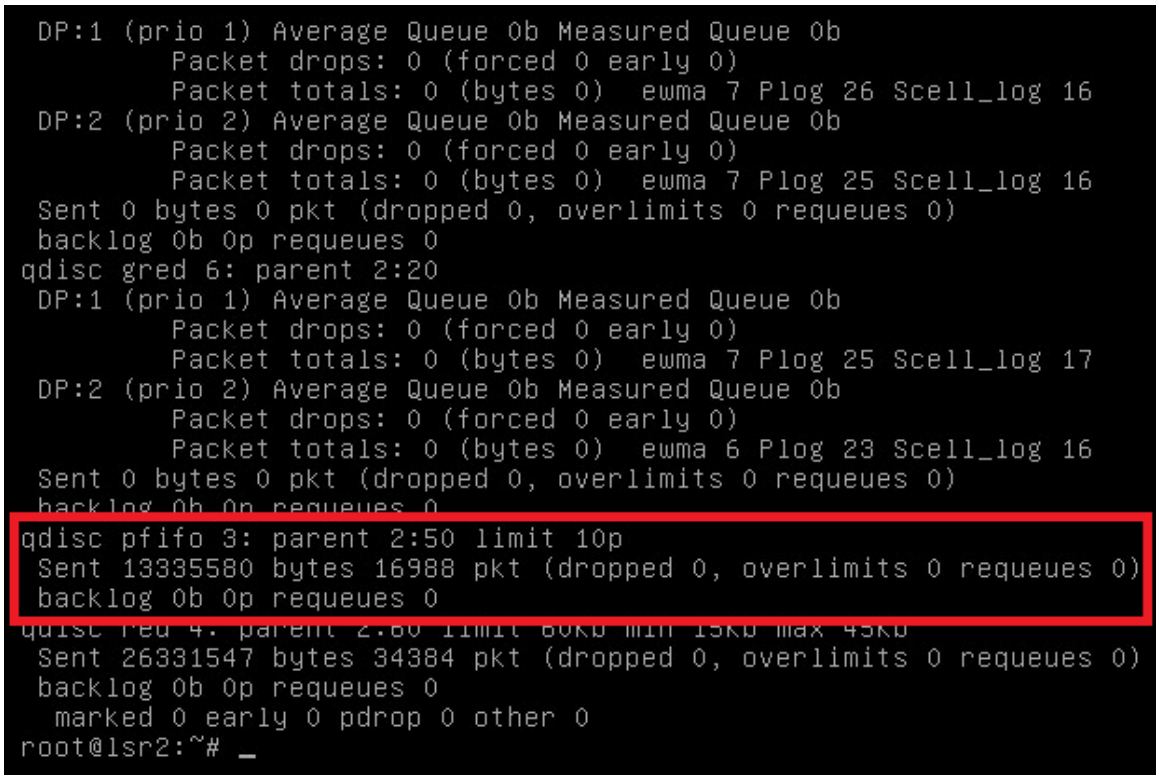
Για την επαλήθευση των παραπάνω ρυθμίσεων στείλαμε UDP κίνηση από το AN3 (User2 – 10.50.2.3) στο AN2 (User1 – 10.50.3.2). Η κίνηση καταγράφηκε σε όλη τη διαδρομή του δικτύου. Αρχικά είδαμε ότι η κίνηση έφτασε στη διεπαφή εισόδου του SWMANE2 χωρίς σήμανση (Εικόνα 14). Έπειτα, ελέγξαμε στους LSR1 και LSR2 ότι η κίνηση μαρκαρίστηκε ως EF, δρομολογήθηκε μέσω του LSP με label 1000 και πέρασε σωστά από την EF κλάση (Εικόνες 15, 16). Τέλος, σιγουρέψαμε ότι η MPLS ετικέτα αφαιρέθηκε σωστά στη διεπαφή εξόδου του SWMANE1 (Εικόνα 17).

```
    10.50.2.3 > 10.50.3.2: udp
19:07:50.412000 IP (tos 0x0, ttl 64, id 4606, offset 0, flags [+], proto UDP (17
), length 1492)
    10.50.2.3.39132 > 10.50.3.2.5001: UDP, length 1470
19:07:50.412021 IP (tos 0x0, ttl 64, id 4606, offset 1472, flags [none], proto U
DP (17), length 26)
    10.50.2.3 > 10.50.3.2: udp
19:07:50.413166 IP (tos 0x0, ttl 64, id 4607, offset 0, flags [+], proto UDP (17
), length 1492)
    10.50.2.3.39132 > 10.50.3.2.5001: UDP, length 1470
19:07:50.413187 IP (tos 0x0, ttl 64, id 4607, offset 1472, flags [none], proto U
DP (17), length 26)
    10.50.2.3 > 10.50.3.2: udp
19:07:50.414334 IP (tos 0x0, ttl 64, id 4608, offset 0, flags [+], proto UDP (17
), length 1492)
    10.50.2.3.39132 > 10.50.3.2.5001: UDP, length 1470
^C
2629 packets captured
2636 packets received by filter
0 packets dropped by kernel
root@swmane2:~# _
```

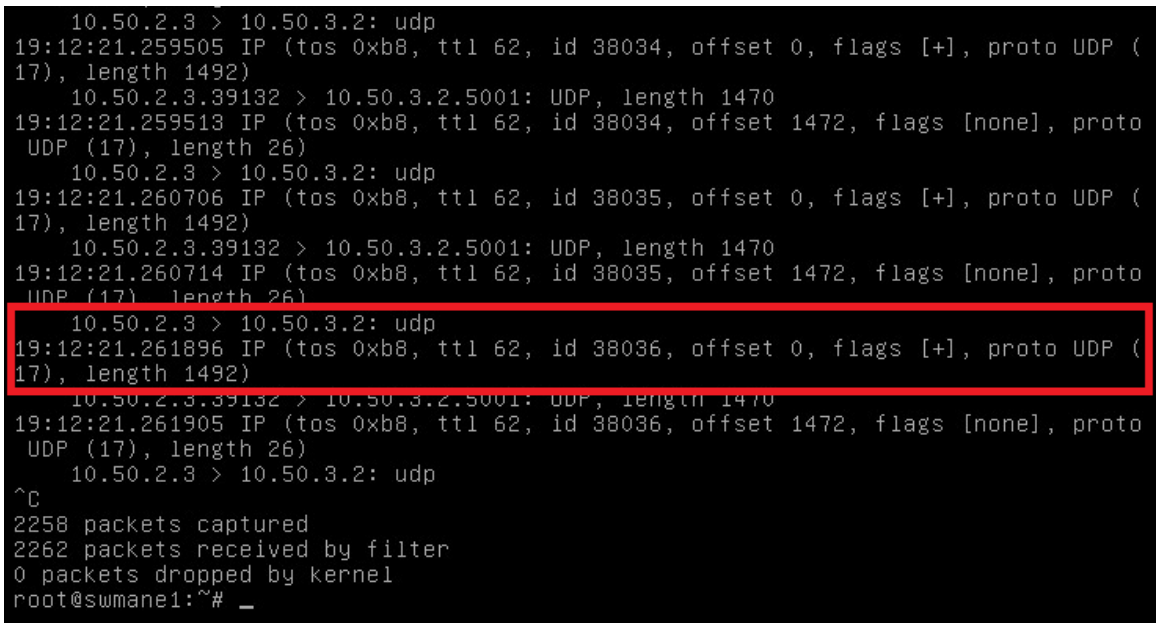
Εικόνα 14 - Μη μαρκαρισμένη κίνηση στην είσοδο του δικτύου

```
    10.50.2.3 > 10.50.3.2: udp
19:14:12.529977 MPLS (label 1000, exp 1, [S], ttl 63)
        IP (tos 0xb8, ttl 63, id 12968, offset 0, flags [+], proto UDP (17), len
gth 1492)
    10.50.2.3.39132 > 10.50.3.2.5001: UDP, length 1470
19:14:12.530118 MPLS (label 1000, exp 1, [S], ttl 63)
        IP (tos 0xb8, ttl 63, id 12968, offset 1472, flags [none], proto UDP (17
), length 26)
    10.50.2.3 > 10.50.3.2: udp
19:14:12.531142 MPLS (label 1000, exp 1, [S], ttl 63)
        IP (tos 0xb8, ttl 63, id 12969, offset 0, flags [+], proto UDP (17), len
gth 1492)
    10.50.2.3.39132 > 10.50.3.2.5001: UDP, length 1470
19:14:12.531276 MPLS (label 1000, exp 1, [S], ttl 63)
        IP (tos 0xb8, ttl 63, id 12969, offset 1472, flags [none], proto UDP (17
), length 26)
    10.50.2.3 > 10.50.3.2: udp
^C
2165 packets captured
2168 packets received by filter
0 packets dropped by kernel
root@lsr1:~# _
```

Εικόνα 15 - Η κίνηση μετά τη σηματοδότηση



Εικόνα 16 – Η κίνηση περνώντας από την EF κλάση



Εικόνα 17 - Η κίνηση κατά την έξοδο από το δίκτυο χωρίς ετικέτα

5.2.6 Πρόσθετα

Το ΣΔΔΠ υποστηρίζει επιπλέον την αναίρεση όλων των αλλαγών και την επαναφορά του δικτύου στην αρχική του κατάσταση εκτελώντας την εντολή ***>python intraNRM.py reset***. Τέλος, η εντολή ***>python intraNRM.py help*** εμφανίζει στην οθόνη ένα σύντομο μήνυμα βοήθειας.

6 Συμπεράσματα

Σκοπός αυτής της εργασίας ήταν ο σχεδιασμός, η ανάπτυξη και η υλοποίηση ενός συστήματος διαχείρισης δικτυακών πόρων για τη παροχή ποιότητας της υπηρεσίας σε ένα δίκτυο. Παρουσιάστηκαν έννοιες όπως η διαχείριση δικτύων και η μηχανίκευση κίνησης και αναλύθηκαν οι βασικότερες τεχνολογίες για την παροχή ποιότητας της υπηρεσίας.

Η διαχείριση δικτύων περιλαμβάνει την ανάπτυξη, την ενσωμάτωση και το συντονισμό του υλικού, του λογισμικού και των ανθρώπινων στοιχείων για την παρακολούθηση, τη δοκιμή, την καταγραφή, τη διαμόρφωση, την ανάλυση, την αξιολόγηση και τον έλεγχο των πόρων και των στοιχείων του δικτύου και για την κάλυψη των απαιτήσεων των χρηστών και των υπηρεσιών με λογικό κόστος. Χρησιμοποιώντας την τεχνολογία του MPLS σε συνδυασμό με το DiffServ καταφέραμε να κατηγοριοποιήσουμε την κίνηση ανάλογα με τις ανάγκες τις σε αποδόσεις (π.χ., χαμηλή καθυστέρηση) και να τη δρομολογήσουμε σωστά για την αποφυγή συμφόρησης.

7 Βιβλιογραφία

[1] FP7-funded ICT project: Media Ecosystem Deployment through Ubiquitous Content-Aware Network Environments, ALICANTE, No248652, 2011, http://www.ict-alicante.eu/

[2] P. Anapliotis, E. Pallis, D. Negru, V. Zacharopoulos. "Enhancing legacy infrastructures with content-aware enablers towards a networked-media platform." In Multimedia and Expo (ICME), 2011 IEEE International Conference on, pp. 1-5. IEEE, 2011.

[3] Harilaos Koumaras, et al. "Media ecosystems: a novel approach for content-awareness in future networks." In The future internet, pp. 369-380. Springer Berlin Heidelberg, 2011.

[4] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, Walter Weiss. "An architecture for differentiated services." RFC 2475, December 1998.

[5] S. Andreozzi, A. Ciuffoletti, A. Ghiselli, D. Antoniades, E. Markatos, M. Polychronakis, P. Trimintzios. "On the integration of passive and active network monitoring in grid systems." In Integrated Research in GRID Computing, pp. 147-161. Springer US, 2007.

[6] Dragos S. Niculescu, Mihai Stanciu, Marius Vochin, Eugen Borcoci, Nikolaos Zotos. "Implementation of a Media Aware Network Element for Content Aware Networks." In CTRQ 2011, The Fourth International Conference on Communication Theory, Reliability and Quality of Service, pp. 78-83, 2011.

[7] Eric Rosen, Arun Viswanathan, Ross Callon. "RFC 3031: Multiprotocol label switching architecture." IETF, January 2001.

[8] Scott Shenker. "Fundamental design issues for the future Internet." Selected areas in Communications, IEEE Journal on 13, no. 7 (1995): 1176-1188.

[9] Tselentis Georgios, ed. "Towards the Future Internet: A European Research Perspective." IOS press, 2009.

[10] Dinesh C. Verma. "Service level agreements on IP networks." Proceedings of the IEEE 92, no. 9 (2004): 1382-1388.

[11] Daniel Awduche, Angela Chiu, Anwar Elwalid, Indra Widjaja, XiPeng Xiao. "Overview and principles of Internet traffic engineering." RFC 3272, May, 2002.

[12] Andrew S. Tanenbaum, David J. Wetherall. "Computer networks." 5th Edition, Englewood Cliffs (NY): Prentice-Hall, 2011.

[13] Evangelos Pallis, et al. "D6.3.1: Network Layer Management and Control - I." Deliverable, Information and Communication Technologies, 2012.

[14] Carsten Maartmann-Moe. "The Need for QoS in the Internet." (2007).

[15] E. Stephan. "IP Performance Metrics (IPPM) Metrics Registry." RFC 4148, August 2005.

[16] Sundeep B. Singh, Girish P. Saraph. "DiffServ over MPLS: Tuning QOS parameters for Converged Traffic using Linux Traffic Control."

[17] Mario Marchese. "QoS over heterogeneous networks." Wiley, 2007.

[18] James F. Kurose, Keith W. Ross. "Computer networking." Pearson Education, 2012.

[19] R. Braden, D. Clark, S. Shenker. "RFC 1633: Integrated services in the Internet architecture: an overview." June 1994.

8 Παράρτημα

8.1 IntraNRM.py

```
__author__ = 'Antonas A.'

import sys, dsManager, lspManager, policer, assorted

if len(sys.argv) >= 2:
    if 'setPath' == sys.argv[1]:
        lspManager.setPath(sys.argv[2])
    elif 'clearPath' == sys.argv[1]:
        lspManager.clearPath(sys.argv[2])
    elif 'getInfo' == sys.argv[1]:
        lspManager.getVPathInfoFromDB("all")
    elif 'setClass' == sys.argv[1]:
        dsManager.setClass(sys.argv[2])
    elif 'mpls2ds' == sys.argv[1]:
        dsManager.mpls2ds(sys.argv[2])
    elif "reset" == sys.argv[1]:
        assorted.reset()
    elif "help" == sys.argv[1]:
        assorted.showHelp()
    elif "parse" == sys.argv[1]:
        policer.parser()
    elif "showVCANs" == sys.argv[1]:
        policer.showVCANs()
    else:
        print "Unknown command"
        assorted.showHelp()
        sys.exit(2)
    sys.exit(0)
else:
    assorted.showHelp()
    sys.exit(2)
```

8.2 lspManager.py

```
__author__ = 'Antonas A.'

import socket, sys, DBHandler, MySQLdb

def getVPathInfoFromDB(label):
    """
```

```
    Fetch various information from the database
    """
    interfaces = []
    nodeIDs = []
    IPs = []
    eths = []
    managementIP = []
    edge = []
    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    if label == "all":
        try:
            Cursor = DBConnection.cursor()
            Cursor.execute("select LSPID from LSPs")
            Row = Cursor.fetchone()
            while Row is not None:
                labels = Row[0]
                print labels
                Row = Cursor.fetchone()
            Cursor.close()

            DBHandler.closeDBConnection(DBConnection)
            return None

        except MySQLdb.Error, e:
            print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
            DBHandler.closeDBConnection(DBConnection)
            sys.exit(1)

    elif label == "mark":
        vcan = []
        srcIP = []
        dstIP = []
        label = []
        rate = []
        trafficClass = []
        iface = []
        node = []
        managementIP = []
        try:
            Cursor = DBConnection.cursor()
            Cursor.execute("select VCANID, SrcNetPrefix, DstNetPrefix, LSPID, CIR,
TrafficClass from SLSs")
            Row = Cursor.fetchone()
```

```
            while Row is not None:
                vcan.append(Row[0])
                srcIP.append(Row[1])
                dstIP.append(Row[2])
                label.append(Row[3])
                rate.append(Row[4])
                trafficClass.append(Row[5])
                Row = Cursor.fetchone()
            Cursor.close()

        except MySQLdb.Error, e:
            print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
            DBHandler.closeDBConnection(DBConnection)
            sys.exit(1)

        for i in range(0, len(label)):
            try:
                Cursor = DBConnection.cursor()
                Cursor.execute("select InterfaceIn from LSPs where LSPID = %s", label[i])
                Row = Cursor.fetchone()
                while Row is not None:
                    iface.append(Row[0])
                    Row = Cursor.fetchone()
                Cursor.close()

            except MySQLdb.Error, e:
                print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
                DBHandler.closeDBConnection(DBConnection)
                sys.exit(1)

        for i in range(0, len(iface)):
            try:
                Cursor = DBConnection.cursor()
                Cursor.execute("select NodeID from Interfaces where InterfaceID = %s",
iface[i])
                Row = Cursor.fetchone()
                while Row is not None:
                    node.append(Row[0])
                    Row = Cursor.fetchone()
                Cursor.close()

            except MySQLdb.Error, e:
                print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
                DBHandler.closeDBConnection(DBConnection)
                sys.exit(1)
```

```
        for i in range(0, len(node)):
            try:
                Cursor = DBConnection.cursor()
                Cursor.execute("select ManagementIP from Nodes where NodeID = %s",
node[i])
                Row = Cursor.fetchone()
                while Row is not None:
                    managementIP.append(Row[0])
                    Row = Cursor.fetchone()
                Cursor.close()

            except MySQLdb.Error, e:
                print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
                DBHandler.closeDBConnection(DBConnection)
                sys.exit(1)

        DBHandler.closeDBConnection(DBConnection)
        return vcan, srcIP, dstIP, label, rate, trafficClass, iface, managementIP

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select InterfaceOut, InterfaceIn from Routes where LSPID = %s
order by HopCounter", label)
        Row = Cursor.fetchone()
        while Row is not None:
            interfaces.append(Row[0])
            interfaces.append(Row[1])
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    for i in range(0, len(interfaces)):
        try:
            Cursor = DBConnection.cursor()
            Cursor.execute("select NodeID, IP, Name from Interfaces where InterfaceID =
%s", interfaces[i])
            Row = Cursor.fetchone()
            while Row is not None:
                nodeIDs.append(Row[0])
                IPs.append(Row[1])
                eths.append(Row[2])
                Row = Cursor.fetchone()
```

```
            Cursor.close()

        except MySQLdb.Error, e:
            print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
            DBHandler.closeDBConnection(DBConnection)
            sys.exit(1)

    for i in range(0, len(nodeIDs)):
        try:
            Cursor = DBConnection.cursor()
            Cursor.execute("select ManagementIP, Edge from Nodes where NodeID = %s",
nodeIDs[i])
            Row = Cursor.fetchone()
            while Row is not None:
                managementIP.append(Row[0])
                edge.append(Row[1])
                Row = Cursor.fetchone()
            Cursor.close()

        except MySQLdb.Error, e:
            print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
            DBHandler.closeDBConnection(DBConnection)
            sys.exit(1)

    DBHandler.closeDBConnection(DBConnection)

    return managementIP, IPs, eths, edge

def setPath(label):
    """
    Set up the LSP
    """
    managementIP, interfaces, eths, edge = getVPathInfoFromDB(label)
    i = 0

    while i < len(managementIP):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        try:
            s.connect((managementIP[i], 50050))

        except socket.error:
            print socket.error, managementIP[i]
            s.close()
            sys.exit(2)
```

```
        print managementIP[i]

        if edge[i] == 1:
            if i == 0:
                s.sendall("mpls labelspace set "+eths[i]+" 0\n"
                        "mpls nhlfe add key "+label+" instructions push gen "+label+" nexthop
"+eths[i]+" "+interfaces[i+1])
                data = s.recv(65535)
                print data
                i += 1

            else:
                s.sendall("mpls labelspace set "+eths[len(eths)-1]+" 0\n"
                        "mpls nhlfe add key "+label+" instructions pop peek\n"
                        "mpls ilm add label gen "+label+" forward "+label)
                data = s.recv(65535)
                print data
                i += 1

        elif edge[i] == 0:
            s.sendall("mpls labelspace set "+eths[i]+" 0\n"
                    "mpls labelspace set "+eths[i+1]+" 0\n"
                    "mpls nhlfe add key "+label+" instructions nexthop "+eths[i+1]+"
"+interfaces[i+2]+"\n"
                    "mpls ilm add label gen "+label+" forward "+label)
            data = s.recv(65535)
            print data
            i += 2

        else:
            print "Not a valid LSP"
            s.close()
            sys.exit(2)

        s.close()

    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    Query = "update LSPs set Applied = %d where LSPID = %d" % (1, int(label))

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute(Query)
        DBConnection.commit()
```

```
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    DBHandler.closeDBConnection(DBConnection)

def clearPath(label):
    """
    Delete a LSP
    """
    managementIP, interfaces, eths, edge = getVPathInfoFromDB(label)
    i = 0

    while i < len(managementIP):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            s.connect((managementIP[i], 50050))

        except socket.error:
            print socket.error, managementIP[i]
            s.close()
            sys.exit(2)

        print managementIP[i]

        if edge[i] == 1:
            if i == 0:
                s.sendall("mpls nhlfe del key "+label+" instructions push gen "+label+"
nexthop "+eths[i]+" "+interfaces[i+1])
                data = s.recv(65535)
                print data
                i += 1

            else:
                s.sendall("mpls nhlfe del key "+label+" instructions pop peek\n"
                        "mpls ilm del label gen "+label+" forward "+label)
                data = s.recv(65535)
                print data
                i += 1

        elif edge[i] == 0:
            s.sendall("mpls nhlfe del key "+label+" instructions nexthop "+eths[i+1]+"
"+interfaces[i+2]+"\n"
```

```
                "mpls ilm del label gen "+label+" forward "+label)
            data = s.recv(65535)
            print data
            i += 2

        else:
            print "Not a valid LSP"
            s.close()
            sys.exit(2)

        s.close()

    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    Query = "update LSPs set Applied = %d where LSPID = %d" % (0, int(label))

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute(Query)
        DBConnection.commit()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    DBHandler.closeDBConnection(DBConnection)
```

8.3 dsManager.py

```
__author__ = 'Antonas A.'

import socket, sys, MySQLdb, DBHandler
from lspManager import getVPathInfoFromDB

def setClass(mode):
    """
    Configure traffic classes
    """
    nodeID = []
    hostname = []
    managementIP = []
```

```
    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select NodeID, Hostname, ManagementIP from Nodes")
        Row = Cursor.fetchone()
        while Row is not None:
            nodeID.append(Row[0])
            hostname.append(Row[1])
            managementIP.append(Row[2])
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    for i in range(0, len(nodeID)):
        if (hostname[i] == "HWMANE") or (hostname[i] == "IntraNRM"):
            continue

        f = open(hostname[i]+".cfg", 'r')
        for line in f:
            x, iface = line.split("\t\t")
            iface = str(iface).replace("\n", "")
            if x == "ingress":
                continue

            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            try:
                s.connect((managementIP[i], 50050))

            except socket.error:
                print socket.error, managementIP[i]
                s.close()
                sys.exit(2)

            if mode == "default":
                s.sendall("tc qdisc add dev "+iface+" root handle 1:0 dsmark indices 64
default_index 0x161\n"
```

```
		"tc filter add dev "+iface+" parent 1:0 protocol all prio 1 tcindex mask
0xfc shift 2 pass_on\n"
		"tc qdisc add dev "+iface+" parent 1:0 handle 2:0 htb\n"
		"tc class add dev "+iface+" parent 2:0 classid 2:1 htb rate 100Mbit ceil
100Mbit\n"
		"tc filter add dev "+iface+" parent 2:0 protocol all prio 1 tcindex mask
0xf0 shift 4 pass_on\n"
		"tc class add dev "+iface+" parent 2:1 classid 2:10 htb rate 20Mbit ceil
60Mbit\n" #AF1
		"tc filter add dev "+iface+" parent 2:0 prio 1 handle 1 tcindex classid
2:10\n"
		"tc qdisc add dev "+iface+" handle 5:0 parent 2:10 gred setup DPs 3
default 2 grio\n"
		"tc filter add dev "+iface+" parent 1:0 prio 1 handle 0x0a tcindex classid
1:111\n" #AF11
		"tc qdisc change dev "+iface+" parent 2:10 gred limit 4000k min 500k
max 1000k burst 520 avpkt 1280 bandwidth 20000000 DP 1 probability 0.01 prio 1\n"
		"tc filter add dev "+iface+" parent 1:0 prio 1 handle 0x0c tcindex classid
1:112\n" #AF12
		"tc qdisc change dev "+iface+" parent 2:10 gred limit 4000k min 500k
max 1000k burst 520 avpkt 1280 bandwidth 20000000 DP 2 probability 0.02 prio 2\n"
		"tc class add dev "+iface+" parent 2:1 classid 2:20 htb rate 20Mbit ceil
60Mbit\n" #AF2
		"tc filter add dev "+iface+" parent 2:0 prio 1 handle 2 tcindex classid
2:20\n"
		"tc qdisc add dev "+iface+" handle 6:0 parent 2:20 gred setup DPs 3
default 2 grio\n"
		"tc filter add dev "+iface+" parent 1:0 prio 1 handle 0x12 tcindex classid
1:121\n" #AF21
		"tc qdisc change dev "+iface+" parent 2:20 gred limit 2800k min 350k
max 700k burst 317 avpkt 1470 bandwidth 20000000 DP 1 probability 0.02 prio 1\n"
		"tc filter add dev "+iface+" parent 1:0 prio 1 handle 0x14 tcindex classid
1:122\n" #AF22
		"tc qdisc change dev "+iface+" parent 2:20 gred limit 2000k min 250k
max 500k burst 226 avpkt 1470 bandwidth 20000000 DP 2 probability 0.04 prio 2\n"
		"tc class add dev "+iface+" parent 2:1 classid 2:50 htb rate 40Mbit ceil
40Mbit\n" #EF
		"tc qdisc add dev "+iface+" handle 3:0 parent 2:50 pfifo limit 50\n"
		"tc filter add dev "+iface+" parent 1:0 prio 1 handle 0x2e tcindex classid
1:151\n"
		"tc filter add dev "+iface+" parent 2:0 prio 1 handle 5 tcindex classid
2:50\n"
		"tc class add dev "+iface+" parent 2:1 classid 2:60 htb rate 20Mbit ceil
60Mbit\n" #BE
		"tc qdisc add dev "+iface+" parent 2:60 handle 4:0 red limit 60k min 15k
max 45k burst 20 avpkt 1000 bandwidth 20000000 probability 0.4\n"
```

```
                "tc filter add dev "+iface+" parent 2:0 prio 1 handle 6 tcindex classid
2:60")
            data = s.recv(65535)
            print data

        elif mode == "delete":
            s.sendall("tc qdisc del dev "+iface+" root")
            data = s.recv(65535)
            print data

        else:
            print "Not a valid argument."

        s.close()

    i += 1
    f.close()

def mpls2ds(label):
    """
    Make the DSCP to EXP conversion
    """
    CLASS  DSCP   EXP  TCINDEX
    AF11   0x0a    1     0xb8
    AF12   0x0c    2     0x28
    AF21   0x12    3     0x30
    AF22   0x14    4     0x48
    EF     0x2e    5     0x50
    BE     0x00    0     0x00
    """
    managementIP, interfaces, eths, edge = getVPathInfoFromDB(label)
    i = 0

    while i < len(managementIP):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        try:
            s.connect((managementIP[i], 50050))

        except socket.error:
            print socket.error, managementIP[i]
            s.close()
            sys.exit(2)

        print managementIP[i]
```

```
        if edge[i] == 1:
            if i == 0:
                s.sendall("mpls nhlfe change key "+label+" instructions push ds2exp 0xf 0x0 -
0x0 0x2e - 0x1 0x0a - 0x2 0x0c - 0x3 0x12 - 0x4 0x14 - 0x5 gen "+label+" exp2tc 0x1 -
0xb8 0x2 - 0x28 0x3 - 0x30 0x4 - 0x48 0x5 - 0x50 0x0 - 0x0 nexthop "+eths[i]+"
"+interfaces[i+1])
                data = s.recv(65535)
                print data
                i += 1

            else:
                s.sendall("mpls nhlfe change key "+label+" instructions exp2tc 0x1 - 0xb8 0x2
- 0x28 0x3 - 0x30 0x4 - 0x48 0x5 - 0x50 0x0 - 0x0 pop peek")
                data = s.recv(65535)
                print data
                i += 1

        elif edge[i] == 0:
            s.sendall("mpls nhlfe change key "+label+" instructions exp2tc 0x1 - 0xb8 0x2 -
0x28 0x3 - 0x30 0x4 - 0x48 0x5 - 0x50 0x0 - 0x0 nexthop "+eths[i+1]+"
"+interfaces[i+2])
            data = s.recv(65535)
            print data
            i += 2

        else:
            print "Not a valid LSP"
            s.close()
            sys.exit(2)

        s.close()
```

8.4 policer.py

```
__author__ = 'Antonas A.'

import socket, sys, MySQLdb, DBHandler, time, os.path
from lspManager import getVPathInfoFromDB

def initialize():

    hostname = ["SWMANE1", "SWMANE2"]
    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None
```

```
    for i in range(0, len(hostname)):
        try:
            Cursor = DBConnection.cursor()
            Cursor.execute("select ManagementIP from Nodes where Hostname = %s",
hostname[i])
            Row = Cursor.fetchone()
            while Row is not None:
                managementIP = Row[0]
                Row = Cursor.fetchone()
            Cursor.close()

        except MySQLdb.Error, e:
            print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
            DBHandler.closeDBConnection(DBConnection)
            sys.exit(1)

        f = open(hostname[i]+".cfg", 'r')
        for line in f:
            x, iface = line.split("\t\t")
            iface = str(iface).replace("\n", "")
            if x == "ingress":
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

                try:
                    s.connect((managementIP, 50050))

                except socket.error:
                    print socket.error, managementIP
                    s.close()
                    sys.exit(2)

                s.sendall("tc qdisc add dev "+iface+" handle ffff: ingress")
                data = s.recv(65535)
                print data
                s.close()
                break
        f.close()

    DBHandler.closeDBConnection(DBConnection)

def clearQDisc(IP, iface):

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
```

```
        s.connect((IP, 50050))

    except socket.error:
        print socket.error, IP
        s.close()
        sys.exit(2)

    s.sendall("tc qdisc del dev "+iface+" handle ffff: ingress")
    data = s.recv(65535)
    print data
    s.close()

def mark(vcan, cmd, srcIP, dstIP, label, rate, peak, trafficClass):

    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select InterfaceOut from LSPs where LSPID = %s", label)
        Row = Cursor.fetchone()
        while Row is not None:
            ifaceID = Row[0]
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select NodeID from Interfaces where InterfaceID = %s", ifaceID)
        Row = Cursor.fetchone()
        while Row is not None:
            nodeID = Row[0]
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)
```

```
    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select ManagementIP, Hostname from Nodes where NodeID =
%s", nodeID)
        Row = Cursor.fetchone()
        while Row is not None:
            managementIP = Row[0]
            hostname = Row[1]
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select SLSID from SLSs where VCANID = %s and SrcNetPrefix =
%s and DstNetPrefix = %s", (vcan, srcIP, dstIP))
        Row = Cursor.fetchone()
        while Row is not None:
            slsid = Row[0]
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    f = open(hostname+".cfg", 'r')
    for line in f:
        x, iface = line.split("\t\t")
        iface = str(iface).replace("\n", "")
        if x == "ingress":
            break
    f.close()

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select InterfaceID from Interfaces where NodeID = %s and Name =
%s", (nodeID, iface))
        Row = Cursor.fetchone()
        while Row is not None:
            interfaceID = Row[0]
```

```
            Row = Cursor.fetchone()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    if trafficClass == "AF11":
        dscp = 0x0a
    elif trafficClass == "AF12":
        dscp = 0x0c
    elif trafficClass == "AF21":
        dscp = 0x12
    elif trafficClass == "AF22":
        dscp = 0x14
    elif trafficClass == "EF":
        dscp = 0x2e
    elif trafficClass == "BE":
        dscp = 0x00
    else:
        print "Traffic class not supported"
        sys.exit(2)

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        s.connect((managementIP, 50050))

    except socket.error:
        print socket.error, managementIP
        s.close()
        sys.exit(2)

    if cmd == "C":
        Query = "insert into Policers (SLSID, InterfaceID, ParentID) values (%d, %d, '%s')"
% (slsid, interfaceID, "ffff:")
        s.sendall(#"ip route add "+dstIP[i]+" via "+eth[i]+" mpls "+str(label[i])+"\n"
                "iptables -t mangle -A FORWARD -p ip -s "+srcIP+" -d "+dstIP+" -j DSCP --
set-dscp "+str(dscp)+"\n"
                "tc filter add dev "+iface+" parent ffff: handle ::"+str(slsid)+" protocol ip prio
1 u32 match ip dst "+dstIP+" match ip src "+srcIP+" police rate "+str(rate)+"Mbit burst
50k drop flowid :"+str(slsid))
        data = s.recv(65535)
        print data
        s.close()
```

```
    elif cmd == "M":
        s.sendall(#"ip route add "+dstIP[i]+" via "+eth[i]+" mpls "+str(label[i])+"\n"
                "iptables -t mangle -R FORWARD -p ip -s "+srcIP+" -d "+dstIP+" -j DSCP --
set-dscp "+str(dscp)+"\n"
                "tc filter change dev "+iface+" parent ffff: handle 800::"+str(slsid)+" protocol
ip prio 1 u32 match ip dst "+dstIP+" match ip src "+srcIP+" police rate "+str(rate)+"Mbit
burst 50k drop flowid :"+str(slsid))
        data = s.recv(65535)
        print data
        s.close()
    elif cmd == "D":
        Query = "delete from Policers where SLSID = %d" % (slsid)
        s.sendall(#"ip route add "+dstIP[i]+" via "+eth[i]+" mpls "+str(label[i])+"\n"
                "iptables -t mangle -D FORWARD -p ip -s "+srcIP+" -d "+dstIP+" -j DSCP --
set-dscp "+str(dscp)+"\n"
                "tc filter del dev "+iface+" parent ffff: handle 800::"+str(slsid)+" protocol ip
prio 1 u32 match ip dst "+dstIP+" match ip src "+srcIP+" police rate "+str(rate)+"Mbit
burst 50k drop flowid :"+str(slsid))
        data = s.recv(65535)
        print data
        s.close()

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute(Query)
        DBConnection.commit()
        Cursor.close()

    except MySQLdb.Error, e:
        print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
        DBHandler.closeDBConnection(DBConnection)
        sys.exit(1)

    DBHandler.closeDBConnection(DBConnection)

def parser():

    original = time.ctime(os.path.getmtime("vcan_params.cfg"))
    modified = time.ctime(os.path.getmtime("vcan_params.cfg"))
    initialize()
    count = 0
    while True:
        if (original == modified) & (count != 0):
            print "File vcan_params.cfg has not been modified."
            time.sleep(60)
            modified = time.ctime(os.path.getmtime("vcan_params.cfg"))
```

```
            continue

        f = open("vcan_params.cfg", 'r')
        f.next()

        for line in f:
            vcan, cmd, cati, srcIP, dstIP, label, rate, peak, trafficClass = line.split("\t\t")
            trafficClass = str(trafficClass).replace("\n", "")
            service = int(cati[3], 16)
            count += 1

            DBConnection = DBHandler.getDBConnection()
            if DBConnection is None:
                return None

            if cmd == "C":
                Query = "insert into SLSs (VCANID, CATI, SrcNetPrefix, DstNetPrefix,
LSPID, CIR, EIR, ServiceTypeID, TrafficClass) values (%d, '%s', '%s', '%s', %d, %f, %f,
%d, '%s')" % (int(vcan), cati, srcIP, dstIP, int(label), float(rate), float(peak), int(service),
trafficClass)

                try:
                    Cursor = DBConnection.cursor()
                    Cursor.execute(Query)
                    DBConnection.commit()
                    Cursor.close()

                except MySQLdb.Error, e:
                    print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
                    DBHandler.closeDBConnection(DBConnection)
                    continue

                mark(vcan, cmd, srcIP, dstIP, label, rate, peak, trafficClass)
                print "VCAN "+vcan+" was successfully created."

            elif cmd == "M":
                Query = "update SLSs set CATI = '%s', LSPID = %d, CIR = %f, EIR = %f,
ServiceTypeID = %d, TrafficClass = '%s' where VCANID = %d and SrcNetPrefix = '%s'
and DstNetPrefix = '%s'" % (cati, int(label), float(rate), float(peak), int(service),
trafficClass, int(vcan), srcIP, dstIP)

                try:
                    Cursor = DBConnection.cursor()
                    Cursor.execute(Query)
                    DBConnection.commit()
                    Cursor.close()
```

```
            except MySQLdb.Error, e:
                print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
                DBHandler.closeDBConnection(DBConnection)
                continue

            mark(vcan, cmd, srcIP, dstIP, label, rate, peak, trafficClass)
            print "VCAN "+vcan+" was successfully modified."

        elif cmd == "D":
            Query = "delete from SLSs where VCANID = %d and SrcNetPrefix = '%s' and
DstNetPrefix = '%s'" % (int(vcan), srcIP, dstIP)
            mark(vcan, cmd, srcIP, dstIP, label, rate, peak, trafficClass)

            try:
                Cursor = DBConnection.cursor()
                Cursor.execute(Query)
                DBConnection.commit()
                Cursor.close()

            except MySQLdb.Error, e:
                print "MySQL Error [%d]: %s" % (e.args[0], e.args[1])
                DBHandler.closeDBConnection(DBConnection)
                continue

            print "VCAN "+vcan+" was successfully removed."

        else:
            print "CMD is wrong. Please check your .cfg file."
            DBHandler.closeDBConnection(DBConnection)
            continue

        DBHandler.closeDBConnection(DBConnection)

    f.close()
    time.sleep(60)
    original = modified
    modified = time.ctime(os.path.getmtime("vcan_params.cfg"))

def deleteVCAN(vcan):

    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    try:
```

```
        Cursor = DBConnection.cursor()
        Query = "delete from SLSs where VCANID = %d" % (int(vcan))
        Cursor.execute(Query)
        DBConnection.commit()
        Cursor.close()
        print "VCAN "+vcan+" was successfully removed."

    except MySQLdb.Error:
        print MySQLdb.Error
        sys.exit(1)

def showVCANs():

    vcan, srcIP, dstIP, label, rate, trafficClass, eth, managementIP =
getVPathInfoFromDB("mark")
    for i in range(0, len(vcan)):
        print vcan[i], srcIP[i], dstIP[i], label[i], rate[i], trafficClass[i]
```

8.5 Daemon.py

```
__author__ = 'Antonas A.'

#!/usr/bin/env python

import sys, os, time, atexit
from signal import SIGTERM

class Daemon:
    """
    A generic daemon class.

    Usage: subclass the Daemon class and override the run() method
    """
    def __init__(self, pidfile, stdin='/dev/null', stdout='/dev/null', stderr='/dev/null'):
        self.stdin = stdin
        self.stdout = stdout
        self.stderr = stderr
        self.pidfile = pidfile

    def daemonize(self):
        """
        do the UNIX double-fork magic, see Stevens' "Advanced
        Programming in the UNIX Environment" for details (ISBN 0201563177)
        http://www.erlenstar.demon.co.uk/unix/faq_2.html#SEC16
        """
```

```
        try:
            pid = os.fork()
            if pid > 0:
                # exit first parent
                sys.exit(0)
        except OSError, e:
            sys.stderr.write("fork #1 failed: %d (%s)\n" % (e.errno, e.strerror))
            sys.exit(1)

        # decouple from parent environment
        os.chdir("/")
        os.setsid()
        os.umask(0)

        # do second fork
        try:
            pid = os.fork()
            if pid > 0:
                # exit from second parent
                sys.exit(0)
        except OSError, e:
            sys.stderr.write("fork #2 failed: %d (%s)\n" % (e.errno, e.strerror))
            sys.exit(1)

        # redirect standard file descriptors
        sys.stdout.flush()
        sys.stderr.flush()
        si = file(self.stdin, 'r')
        so = file(self.stdout, 'a+')
        se = file(self.stderr, 'a+', 0)
        os.dup2(si.fileno(), sys.stdin.fileno())
        os.dup2(so.fileno(), sys.stdout.fileno())
        os.dup2(se.fileno(), sys.stderr.fileno())

        # write pidfile
        atexit.register(self.delpid)
        pid = str(os.getpid())
        file(self.pidfile,'w+').write("%s\n" % pid)

    def delpid(self):
        os.remove(self.pidfile)

    def start(self):
        """
        Start the daemon
        """
```

```
        # Check for a pidfile to see if the daemon already runs
        try:
            pf = file(self.pidfile,'r')
            pid = int(pf.read().strip())
            pf.close()
        except IOError:
            pid = None

        if pid:
            message = "pidfile %s already exist. Daemon already running?\n"
            sys.stderr.write(message % self.pidfile)
            sys.exit(1)

        # Start the daemon
        self.daemonize()
        self.run()

    def stop(self):
        """
        Stop the daemon
        """
        # Get the pid from the pidfile
        try:
            pf = file(self.pidfile,'r')
            pid = int(pf.read().strip())
            pf.close()
        except IOError:
            pid = None

        if not pid:
            message = "pidfile %s does not exist. Daemon not running?\n"
            sys.stderr.write(message % self.pidfile)
            return # not an error in a restart

        # Try killing the daemon process
        try:
            while 1:
                os.kill(pid, SIGTERM)
                time.sleep(0.1)
        except OSError, err:
            err = str(err)
            if err.find("No such process") > 0:
                if os.path.exists(self.pidfile):
                    os.remove(self.pidfile)
            else:
                print str(err)
```

```
            sys.exit(1)

    def restart(self):
        """
        Restart the daemon
        """
        self.stop()
        self.start()

    def run(self):
        """
        You should override this method when you subclass Daemon. It will be called after the process has been
        daemonized by start() or restart().
        """
```

8.6 ManeModule.py

```
__author__ = 'Antonas A.'

import sys, socket, subprocess
from Daemon import Daemon

class ManeModule(Daemon):

    def run(self):
        """
        Open a socket and wait for connections.
        """
        while True:
            host = ''
            port = 50050
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

            try:
                s.bind((host, port))
                s.listen(1)
                conn, addr = s.accept()
                data = conn.recv(65535)

                p = subprocess.Popen(data, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
                for line in p.stdout.readlines():
                    conn.sendall(line)
```

```
            conn.close()

        except socket.error:
            print socket.error
            daemon.restart()

if __name__ == "__main__":
    daemon = ManeModule('/tmp/daemon-example.pid')
    if len(sys.argv) == 2:
        if 'start' == sys.argv[1]:
            daemon.start()
        elif 'stop' == sys.argv[1]:
            daemon.stop()
        elif 'restart' == sys.argv[1]:
            daemon.restart()
        else:
            print "Unknown command"
            sys.exit(2)
        sys.exit(0)
    else:
        print "usage: %s start|stop|restart" % sys.argv[0]
        sys.exit(2)
```

8.7 assorted.py

```
__author__ = 'Antonas A.'

import socket, sys, MySQLdb, DBHandler, time

def reset():
    """
    Reset the entire network
    """
    managementIP = []
    DBConnection = DBHandler.getDBConnection()
    if DBConnection is None:
        return None

    try:
        Cursor = DBConnection.cursor()
        Cursor.execute("select ManagementIP from Nodes")
        Row = Cursor.fetchone()
        while Row is not None:
            managementIP.append(Row[0])
            Row = Cursor.fetchone()
```

```
        Cursor.close()

    except MySQLdb.Error:
        print MySQLdb.Error

    i = 0

    while i < len(managementIP):
        if managementIP[i] == None:
            i += 1
            continue

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            s.connect((managementIP[i], 50050))

        except socket.error:
            print socket.error, managementIP[i]
            s.close()
            sys.exit(2)

        print managementIP[i]

        s.sendall("sh /root/del_mpls.sh")
        data = s.recv(65535)
        print data
        i += 1

        s.close()

def showHelp():
    print "\nUsage: python "+sys.argv[0]+" setPath label\n"\
    "       python "+sys.argv[0]+" clearPath label\n"\
    "       python "+sys.argv[0]+" getInfo\n"\
    "       python "+sys.argv[0]+" mpls2ds label\n"\
    "       python "+sys.argv[0]+" reset\n"\
    "       python "+sys.argv[0]+" serve\n"\
    "       python "+sys.argv[0]+" parse\n"\
    "       python "+sys.argv[0]+" showVCANs\n"\
    "       python "+sys.argv[0]+" setClass mode"

def findLSPs(host):
    """
    Search a host for existing LSPs
    """
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
try:
   s.connect((host, 50050))

except socket.error:
   print socket.error, host
   s.close()
   sys.exit(2)

s.sendall("mpls nhlfe show |grep key |wc -l")
lines = int(s.recv(65535))
s.close()
i = 1
time.sleep(0.1)

while i <= lines:
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   try:
      s.connect((host, 50050))

   except socket.error:
      print socket.error, host
      s.close()
      sys.exit(2)

   a = str(i)
   s.sendall("mpls nhlfe show |grep key |cut -c 17-26 |awk 'NR == "+a+"'")
   data = s.recv(65535)
   print int(data, 16)
   i += 1

   time.sleep(0.1)
   s.close()
```

8.8 DBHandler.py

```
"""
 DBHandler.py

 Author    : Katia Sarsempagieva
 Contact   : katias@iit.demokritos.gr
 Date      : 24 September 2012
 Location  : NCSR Demokritos,
             Institute of Information Technology and Telecommunications,
             Media - Networks Laboratory
```

```
  Description: Python script for all the Database I/O Communications.
"""

import sys
import MySQLdb
import time

import Globals
import General

from array import array



def getDBConnection():
  """ Reads the configuration file passed as a parameter,
      Establishes a MySQL Connection
      Returns the Connection to the Caller Function """
  try:
    Connection = MySQLdb.connect(host = Globals.DBServer, user =
Globals.DBUsername, passwd = Globals.DBPassword, db = "netDB")
    return Connection
  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at getDBConnection()")
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))
    return None


def closeDBConnection(Connection):
  """ Closes the Connection given as an arument """
  try:
    Connection.close()
  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at closeDBConnection()")
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))


def getMonitoringMeasurements(SourceHomebox, DestinationHomebox, ServiceType):
  """ Searches the vcanDB for MonitoringMeasurements which depict
      the Active One-Way Measurements starting from the SourceHomebox
      and finishing in the DestinationHomebox for a specific ServiceType """
  DBConnection = getDBConnection()
```

```
  if DBConnection == None:
    return None

  try:
    Cursor = DBConnection.cursor()
    Query = Globals.MonitoringQuery % (ServiceType, SourceHomebox,
DestinationHomebox)
    Cursor.execute(Query)
    Result = Cursor.fetchone()
    Cursor.close()
    closeDBConnection(DBConnection)

    if Result == None:
      return None

    Measurements = [None, None, None, None, None]
    Measurements[0] = Result[0]
    Measurements[1] = Result[1]
    Measurements[2] = Result[2]
    Measurements[3] = Result[3]
    Measurements[4] = Result[4]

    return Measurements
  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at getMonitoringMeasurements()")
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))
    return None


def getVPathInterface(SourceHomebox, DestinationHomebox, Protocol):
  """ Searches in the vcanDB for the VPathID - First Hop Out Interface IP
      information of a certain VPath starting from the Source Homebox IP,
      ending at the Destination Homebox IP and requesting a specific protocol """
  DBConnection = getDBConnection()
  if DBConnection == None:
    return None

  try:
    Cursor = DBConnection.cursor()
    Query = Globals.VPathInterfaceQuery % (Protocol, SourceHomebox,
DestinationHomebox)
    Cursor.execute(Query)
    Result = Cursor.fetchone()
    Cursor.close()
```

```
    closeDBConnection(DBConnection)

    if Result == None:
      return None

    VPathInterface = [None, None]
    VPathInterface[0] = Result[0]
    VPathInterface[1] = Result[1]

    return VPathInterface
  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at getVPathInterface()")
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))
    return None


def getVPathIDs():
  """ Searches in the vcanDB and retrieves all the VPathIDs. """
  if Globals.DBOK == False:
    return None

  DBConnection = getDBConnection()
  if DBConnection is None:
    return None

  VPathIDs = array('i') #Declare an array of integers
  try:
    Cursor = DBConnection.cursor()
    Cursor.execute(Globals.SelectVPathIDs)
    Row = Cursor.fetchone()

    while Row is not None:
      VPathIDs.append(Row[0])
      Row = Cursor.fetchone()

    Cursor.close()
    closeDBConnection(DBConnection)
    return VPathIDs

  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at getVPathIDs()")
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))
```

```
    return None

def MonitoringExists(VPathID):
  """ Checks the vcanDB.Monitoring table to see if a record with a VPathID
      value already exists."""
  DBConnection = getDBConnection()
  if DBConnection == None:
    return None

  try:
    Cursor = DBConnection.cursor()
    Query = Globals.MonitoringExistsQuery % (VPathID)
    Cursor.execute(Query)
    Result = Cursor.fetchone()
    Cursor.close()
    closeDBConnection(DBConnection)

    if Result == None:
      return False

    return True
  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at MonitoringExists()")
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))
    return True

def setMonitoringMeasurements(ActiveMeasurements, VPathID):
  """ Updates the Monitoring table of the vcanDB, setting the values obtained by the
      ActiveMeasurements table for the specific VPathID table key """
  if ActiveMeasurements == None:
    return

  DBConnection = getDBConnection()
  if DBConnection == None:
    return

  Query = ""
  LastUpdate   = ActiveMeasurements[0]
  Jitter       = float(ActiveMeasurements[1])
  LostPackets  = float(ActiveMeasurements[2])
  SentPackets  = float(ActiveMeasurements[3])
  AverageDelay = float(ActiveMeasurements[4])
  Hops         = int(ActiveMeasurements[5])
```

```
  PythonLastUpdate = time.gmtime();
  try:
    PythonLastUpdate = time.strptime(LastUpdate, "%Y/%m/%d %H:%M:%S")
  except:
    PythonLastUpdate = time.strptime(LastUpdate, "%m %d %H:%M:%S %Y")

  MySQLLastUpdate = time.strftime("%Y-%m-%d %H:%M:%S", PythonLastUpdate)
  LossRate = (100 * LostPackets) / (SentPackets)

  if MonitoringExists(VPathID):
    Query = Globals.MonitoringUpdateQuery % (MySQLLastUpdate, Jitter, LossRate,
AverageDelay, Hops, VPathID)
  else:
    Query = Globals.MonitoringInsertQuery % (VPathID, MySQLLastUpdate, Jitter,
LossRate, AverageDelay, Hops)

  try:
    Cursor = DBConnection.cursor()
    Cursor.execute(Query)
    DBConnection.commit()
    Cursor.close()
    closeDBConnection(DBConnection)
  except MySQLdb.Error, e:
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR,
"Exception at setMonitoringMeasurements(x, %d)" % (VPathID))
    General.ShowMessage(Globals.MOD_DATABASE, Globals.TYPE_ERROR, "Error
%d: %s" % (e.args[0], e.args[1]))
    return
```

8.9 General.py

```
"""
  General.py

  Author     : Katia Sarsempagieva
  Contact    : katias@iit.demokritos.gr
  Date       : 24 September 2012
  Location   : NCSR Demokritos,
               Institute of Information Technology and Telecommunications,
               Media - Networks Laboratory

  Description: General library-like script with useful functions.
"""

import Globals
```

```
import General
import logging

def getServiceConfiguration():
  """ Reads the Configuration File (whose Full Path Name is stored as a global variable in
the Globals.py).
      Retrieves the configuration data, checks its concistency and updates the Global
variables with their
      values. Also, checks whether a Module (ex. Database, SNMP, Web Service) will be
started or not, based
      on the existance of the correct configuration values. """
  # Try to open the configuration file
  try:
    ConfFile = open(Globals.ConfigurationFile);
  except:
    General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_ERROR, "Exception
at getServiceConfiguration()")
    General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_ERROR, "Failed to
open configuration file [%s]" % Globals.ConfigurationFile)
    General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_CRITICAL,
"Exiting.")
    return False


  WSPortStr = ""
  SNMPVersionStr = ""
  SNMPTimeoutStr = ""
  SNMPRetriesStr = ""
  SNMPPollingStr = ""
  SOCKPortStr = ""
  SOCKTimeoutStr = ""

  #If you managed to open it, read it line by line
  if ConfFile.closed:
    General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_ERROR, "Failed to
open configuration file [%s]" % Globals.ConfigurationFile)
    General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_CRITICAL,
"Exiting.")
    return False
  else:
    for ConfLine in ConfFile:
      LoweredLine = ConfLine.strip().lower()
      if not LoweredLine.startswith("#"):   #Avoid all commented out lines, starting with
"#"
        NormalLine = ConfLine.strip()
        Comment = NormalLine.find("#")       #Get the comment position inside the line
```

```
if LoweredLine.startswith("address"):
  if Comment == -1:
    Globals.WSServer = NormalLine[7:]
  else:
    Globals.WSServer = NormalLine[7:Comment]
  Globals.WSServer = Globals.WSServer.strip()

elif LoweredLine.startswith("port"):
  if Comment == -1:
    WSPortStr = NormalLine[4:]
  else:
    WSPortStr = NormalLine[4:Comment]
  WSPortStr = WSPortStr.strip()

elif LoweredLine.startswith("logginglevel"):
  if Comment == -1:
    Globals.WSLoggingLevel = NormalLine[12:]
  else:
    Globals.WSLoggingLevel = NormalLine[12:Comment]
  Globals.WSLoggingLevel = Globals.WSLoggingLevel.strip()

elif LoweredLine.startswith("mysqlserver"):
  if Comment == -1:
    Globals.DBServer = NormalLine[11:]
  else:
    Globals.DBServer = NormalLine[11:Comment]
  Globals.DBServer = Globals.DBServer.strip()

elif LoweredLine.startswith("username"):
  if Comment == -1:
    Globals.DBUsername = NormalLine[8:]
  else:
    Globals.DBUsername = NormalLine[8:Comment]
  Globals.DBUsername = Globals.DBUsername.strip()

elif LoweredLine.startswith("password"):
  if Comment == -1:
    Globals.DBPassword = NormalLine[8:]
  else:
    Globals.DBPassword = NormalLine[8:Comment]
  Globals.DBPassword = Globals.DBPassword.strip()

elif LoweredLine.startswith("snmphost"):
  if Comment == -1:
    Globals.SNMPHost = NormalLine[8:]
```

```
  else:
    Globals.SNMPHost = NormalLine[8:Comment]
  Globals.SNMPHost = Globals.SNMPHost.strip()

elif LoweredLine.startswith("snmpcommunity"):
  if Comment == -1:
    Globals.SNMPCommunity = NormalLine[13:]
  else:
    Globals.SNMPCommunity = NormalLine[13:Comment]
  Globals.SNMPCommunity = Globals.SNMPCommunity.strip()

elif LoweredLine.startswith("snmpversion"):
  if Comment == -1:
    SNMPVersionStr = NormalLine[11:]
  else:
    SNMPVersionStr = NormalLine[11:Comment]
  SNMPVersionStr = SNMPVersionStr.strip()

elif LoweredLine.startswith("snmptimeout"):
  if Comment == -1:
    SNMPTimeoutStr = NormalLine[11:]
  else:
    SNMPTimeoutStr = NormalLine[11:Comment]
  SNMPTimeoutStr = SNMPTimeoutStr.strip()

elif LoweredLine.startswith("snmpretries"):
  if Comment == -1:
    SNMPRetriesStr = NormalLine[11:]
  else:
    SNMPRetriesStr = NormalLine[11:Comment]
  SNMPRetriesStr = SNMPRetriesStr.strip()

elif LoweredLine.startswith("snmppolling"):
  if Comment == -1:
    SNMPPollingStr = NormalLine[11:]
  else:
    SNMPPollingStr = NormalLine[11:Comment]
  SNMPPollingStr = SNMPPollingStr.strip()

if LoweredLine.startswith("socketserver"):
  if Comment == -1:
    Globals.SOCKServer = NormalLine[12:]
  else:
    Globals.SOCKServer = NormalLine[12:Comment]
  Globals.SOCKServer = Globals.SOCKServer.strip()
```

```
        elif LoweredLine.startswith("socketport"):
          if Comment == -1:
            SOCKPortStr = NormalLine[10:]
          else:
            SOCKPortStr = NormalLine[10:Comment]
          SOCKPortStr = SOCKPortStr.strip()

        elif LoweredLine.startswith("sockettimeout"):
          if Comment == -1:
            SOCKTimeoutStr = NormalLine[13:]
          else:
            SOCKTimeoutStr = NormalLine[13:Comment]
          SOCKTimeoutStr = SOCKTimeoutStr.strip()

    ConfFile.close()

    #Now, see that the WS Configuration is fine.
    if len(Globals.WSServer) == 0:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid Web Server's Address")
      Globals.WSOK = False

    try:
      Globals.WSPort = int(WSPortStr)
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid Web Server's Port Number")
      Globals.WSOK = False

    #Check the Database Connection Configuration
    if len(Globals.DBServer) == 0:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid MySQL Server's Address")
      Globals.DBOK = False

    if len(Globals.DBUsername) == 0:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid MySQL Username")
      Globals.DBOK = False

    #Check the SNMP Connection Configuration
    if len(Globals.SNMPHost) == 0:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Host Address")
      Globals.SNMPOK = False
```

```
    if len(Globals.SNMPCommunity) == 0:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Community String")
      Globals.SNMPOK = False

    try:
      Globals.SNMPVersion = int(SNMPVersionStr)
      if ((Globals.SNMPVersion < 1) or (Globals.SNMPVersion > 3)):
        General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Version Number. Setting version to [2]")
        Globals.SNMPVersion = 2
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Version Number. Setting version to [2]")
      Globals.SNMPVersion = 2

    try:
      Globals.SNMPTimeout = int(SNMPTimeoutStr)
      if (Globals.SNMPTimeout < 1):
        General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Timeout Value. Setting to [1] Minute")
        Globals.SNMPTimeout = 60000000 #Microseconds
      else:
        Globals.SNMPTimeout = Globals.SNMPTimeout * 1000 #Convert milliseconds to
microseconds
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Timeout Value. Setting to [1] Minute")
      Globals.SNMPTimeout = 60000000 #Microseconds

    try:
      Globals.SNMPRetries = int(SNMPRetriesStr)
      if (Globals.SNMPRetries < 0):
        General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Retries Value. Setting to [5] Retries")
        Globals.SNMPRetries = 5
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Retries Value. Setting to [5] Retries")
      Globals.SNMPRetries = 5

    try:
      Globals.SNMPPolling = int(SNMPPollingStr)
      if (Globals.SNMPPolling < 0):
        General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Polling Value. Setting to [5] Minutes")
```

```
        Globals.SNMPPolling = 5
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid SNMP Polling Value. Setting to [5] Minutes")
      Globals.SNMPPolling = 5

    #Now, see that the Socket Configuration is fine.
    if len(Globals.SOCKServer) == 0:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid Socket Server's Address")
      Globals.SOCKOK = False

    try:
      Globals.SOCKPort = int(SOCKPortStr)
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid Socket Server's Port Number")
      Globals.SOCKOK = False

    try:
      Globals.SOCKTimeout = int(SOCKTimeoutStr)
    except ValueError:
      General.ShowMessage(Globals.MOD_CONFIG, Globals.TYPE_WARNING,
"Invalid Socket's Timeout Value. Setting to [15] seconds")
      Globals.SOCKTimeout = 15

    return True

def getLoggingLevel():
  """Checks the Global Variable 'WSLoggingLevel' and returns the actual logging level
object
     to be used by the Web Service"""
  # Set the logging level according to the Configuration. If not given, set to "Critical"
  if Globals.WSLoggingLevel.strip().lower().startswith("debug"):
    return logging.DEBUG
  elif Globals.WSLoggingLevel.strip().lower().startswith("info"):
    return logging.INFO
  elif Globals.WSLoggingLevel.strip().lower().startswith("warning"):
    return logging.WARNING
  elif Globals.WSLoggingLevel.strip().lower().startswith("error"):
    return logging.ERROR
  else:
    return logging.CRITICAL

def parseVPIF(VPIF):
```

```
  """Parses the VPIF command sent by a MANE to the Socket Server module. Validates
its format
    and if validation passes, returns an array containing the district values of the string
command"""
  VPIFLength = len(VPIF)
  if VPIFLength < 5:
    return None

  Delimiters = VPIF.count(';')
  if not Delimiters == 3:
    return None

  HBSource = ""
  HBDestination = ""
  Protocol = ""

  #Find the delimiter of the Code Word to pass it
  try:
    Delimiter = VPIF.index(";")
  except:
    return None
  VPIF = VPIF[Delimiter+1 :]

  #Now find the delimiter of the HomeBox Source IP
  try:
    Delimiter = VPIF.index(";")
  except:
    return None
  HBSource = VPIF[: Delimiter ]
  if len(HBSource.strip()) == 0:
    return None

  #Find the delimiter of the HomeBox Destination IP
  VPIF = VPIF[Delimiter+1 :]
  try:
    Delimiter = VPIF.index(";")
  except:
    return None
  HBDestination = VPIF[: Delimiter]
  if len(HBDestination.strip()) == 0:
    return None

  #Finally, the remaining string (with no delimiters) is the Protocol. Get it.
  Protocol = VPIF[Delimiter+1 :]
  if len(Protocol.strip()) == 0:
    return None
```

```
  VPIFTable = [None, None, None]
  VPIFTable[0] = HBSource.strip()
  VPIFTable[1] = HBDestination.strip()
  VPIFTable[2] = Protocol.strip()

  return VPIFTable

def ShowMessage(Module, Type, Message):
  """Formats and Colours a 'print' message, according to the module that want's to print
something
     and the state of the message (ex. Warning, Error etc)"""
  FinalMessage = ""

  ModuleFormat = ""
  TypeFormat = ""

  if Module == Globals.MOD_GENERAL:
    ModuleFormat = "\033[1m[General        ]\033[0m %s "
  elif Module == Globals.MOD_CONFIG:
    ModuleFormat = "\033[1m[Configuration  ]\033[0m %s "
  elif Module == Globals.MOD_SOCK_SERV:
    ModuleFormat = "\033[1m[Socket Server  ]\033[0m %s "
  elif Module == Globals.MOD_WEB_SERV:
    ModuleFormat = "\033[1m[Web Service    ]\033[0m %s "
  elif Module == Globals.MOD_DATABASE:
    ModuleFormat = "\033[1m[Database Server]\033[0m %s "
  elif Module == Globals.MOD_SNMP:
    ModuleFormat = "\033[1m[SNMP Client    ]\033[0m %s "

  if Type == Globals.TYPE_ERROR:
    TypeFormat = "\033[91m%s\033[0m"
  elif Type == Globals.TYPE_WARNING:
    TypeFormat = "\033[93m%s\033[0m"
  elif Type == Globals.TYPE_CRITICAL:
    TypeFormat = "\033[91m\033[1m%s\033[0m"
  elif Type == Globals.TYPE_BEGIN:
    TypeFormat = "\033[92m%s\033[0m"
  elif Type == Globals.TYPE_END:
    TypeFormat ="\033[94m%s\033[0m"
  elif Type == Globals.TYPE_NONE:
    TypeFormat = "%s"

  FinalMessage = TypeFormat % Message
  FinalMessage = ModuleFormat % FinalMessage
  print FinalMessage
```

```
def EmptySet(ATable):
  """Runs through a python array and checks if it's empty or not."""
  Size = len(ATable)
  if Size == 0:
    return True

  for Index in range(0, Size):
    if not ATable[Index] == None:
      return False

  return True
```

8.10 Globals.py

```
"""
  Globals.py

  Author     : Katia Sarsempagieva
  Contact    : katias@iit.demokritos.gr
  Date       : 24 September 2012
  Location   : NCSR Demokritos,
               Institute of Information Technology and Telecommunications,
               Media - Networks Laboratory

  Description: A Global Script containing Application Global Variables.
"""

#################### GLOBAL VARIABLES ##################

#Path of the configuration file
ConfigurationFile = "Configuration"

#Web Server's Configuration
WSServer       = ""
WSPort         = 0
WSLoggingLevel = ""

#Database Connection's Configuration
DBServer   = ""
DBUsername = "root"
DBPassword = "alicante"

#SNMP Connection's Configuration
```

```
SNMPHost      = ""
SNMPCommunity = ""
SNMPVersion   = 0
SNMPTimeout   = 0
SNMPRetries   = 0
SNMPPolling   = 0

#Socket Connection's Configuration
SOCKServer = ""
SOCKPort = 0
SOCKTimeout = 0

#Parameters to know what is working and what's not
SNMPOK = True
DBOK = True
WSOK = True
SOCKOK = True


######################## MESSAGES ######################

ErrorText = "\033[91m%s\033[0m"
CriticalErrorText = "\033[91m\033[1m%s\033[0m"
WarningText = "\033[93m%s\033[0m"


SNMPNOK_MSG = "SNMP Connections will not be started."

DBNOK_MSG   = "Database Communications will not be started."

WSNOK_MSG   = "Web Services will not be started."

SOCKNOK_MSG = "Socket Communication will not be started."

MOD_GENERAL = 0
MOD_CONFIG = 1
MOD_SOCK_SERV = 2
MOD_WEB_SERV = 3
MOD_DATABASE = 4
MOD_SNMP = 5

TYPE_ERROR = 0
TYPE_WARNING = 1
TYPE_CRITICAL = 2
TYPE_NONE = 4
TYPE_BEGIN = 5
```

```
TYPE_END = 6

##################### MYSQL QUERIES ####################

MonitoringQuery = " \
select                                  \n\
  M.Jitter,                              \n\
  M.LossRate,                               \n\
  M.OneWayDelay,                                \n\
  M.NoHops,                                 \n\
  M.AvailableCapacity                           \n\
from VPaths VP                              \n\
join Monitoring M                             \n\
  on VP.VPathID = M.VPathID                         \n\
join Interfaces Iin                           \n\
  on VP.InterfaceIn = Iin.IP                        \n\
join Interfaces Iout                          \n\
  on VP.InterfaceOut = Iout.IP                      \n\
join Nodes Nin                                \n\
  on Nin.NodeID = Iin.NodeID                        \n\
join Nodes Nout                               \n\
  on Nout.NodeID = Iout.NodeID                      \n\
join ServicingSegments HBSource                   \n\
  on HBSource.NodeID = Nin.NodeID                   \n\
join ServicingSegments HBDest                     \n\
  on HBDest.NodeID = Nout.NodeID                    \n\
join ContentMapping CM                            \n\
  on VP.ContentMappingID = CM.ContentMappingID  \n\
where lower(CM.ServiceType) = lower('%s')       \n\
and HBSource.IP = '%s'                            \n\
and HBDest.IP =  '%s'                             \n\
LOCK IN SHARE MODE                                  \n"


VPathInterfaceQuery =  " \
select                                  \n\
  VP.VPathID,                               \n\
  LM.InterfaceOut                               \n\
from VPaths VP                                  \n\
join Interfaces Iin                           \n\
  on VP.InterfaceIn = Iin.IP                        \n\
join Interfaces Iout                          \n\
  on VP.InterfaceOut = Iout.IP                      \n\
join Nodes Nin                                \n\
  on Nin.NodeID = Iin.NodeID                        \n\
join Nodes Nout                               \n\
```

```
  on Nout.NodeID = Iout.NodeID                \n\
join ServicingSegments HBSource               \n\
  on HBSource.NodeID = Nin.NodeID             \n\
join ServicingSegments HBDest                 \n\
  on HBDest.NodeID = Nout.NodeID              \n\
join Protocols Prot                           \n\
  on Prot.ProtocolName = VP.ProtocolName      \n\
join LinkMap LM                               \n\
  on LM.VPathID = VP.VPathID                  \n\
where lower(Prot.ProtocolName) = lower('%s')  \n\
and HBSource.IP = '%s'                        \n\
and HBDest.IP = '%s'                          \n\
and LM.HopCounter = 1                         \n\
LOCK IN SHARE MODE;                           \n"


SelectVPathIDs = "select VPathID from VPaths"

MonitoringExistsQuery = "select 1 from Monitoring where VPathID = %d"

MonitoringInsertQuery = " \
insert into Monitoring                        \n\
(VPathID, LastUpdate, Jitter, LossRate, OneWayDelay, NoHops) \n\
values                                        \n\
(%d, '%s', %f, %f, %f, %d);                   \n"

MonitoringUpdateQuery = " \
update Monitoring set   \n\
  LastUpdate = '%s',    \n\
  Jitter = %f,          \n\
  LossRate = %f,        \n\
  OneWayDelay = %f,     \n\
  NoHops = %d           \n\
where VPathID = %d;     \n"
```