ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
& ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΕΤΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ SCHEDULING ΑΛΓΟΡΙΘΜΩΝ ΣΤΟΝ NS-3

ΣΥΓΓΡΑΦΕΑΣ

Χρήστος Νιανιάκας

Α.Μ.: 3017

ΥΠΕΥΘΥΝΟΣ

Χρήστος Μπούρας, PhD

Καθηγητής τμήματος

ΕΠΙΒΛΕΠΟΝΤΕΣ

Κωνσταντίνος Στάμος, PhD

Ιωάννης Ζαούδης, PhD

ΠΑΤΡΑ, ΙΑΝΟΥΑΡΙΟΣ 2013

Αντί προλόγου

Η ολοκλήρωση της παρούσας διπλωματικής εργασίας,
σηματοδοτεί και την ολοκλήρωση των προπτυχιακών σπουδών μου,
στο τμήμα των Μηχανικών Η/Υ του πανεπιστημίου Πατρών.

Μου δίνεται έτσι η δυνατότητα να αντιληφθώ συνολικά
την εμπειρία τους, και η ευκαιρία να κάνω τον απολογισμό τους.

Θέλω να ευχαριστήσω εγκάρδια,
όλους όσους συνέβαλαν στη διαδρομή αυτή.

Οφείλω όμως ειδικά να αναφερθώ:

Στα μέλη της υπεύθυνης ομάδας, της παρούσας διπλωματικής εργασίας.
Με την εμπιστοσύνη της ανάθεσης, μου έδωσαν τη δυνατότητα
να ασχοληθώ με το ενδιαφέρον θέμα που πραγματεύεται.
Ενώ με την πολύτιμη βοήθεια και καθοδήγηση τους,
συνέβαλαν ουσιαστικά, στην επιτυχή ολοκλήρωση της.

Στο τμήμα των Μηχανικών Η/Υ και τους ανθρώπους του.
Θεωρώ τιμή μου, που έχω υπάρξει μέλος του.

Στους φίλους και την οικογένεια μου.

Κλείνοντας,
θα ήθελα να αφιερώσω την προσπάθεια αυτή,
στους γονείς μου, που χάρη σε αυτούς,
έχουν νόημα όλα τα παραπάνω.

Χρήστος

Per aspera ad astra …

Περίληψη

Η εξάπλωση του διαδικτύου, είχε σαν αποτέλεσμα την εξυπηρέτηση εφαρμογών από IP δίκτυα, που δεν είχαν προβλεφθεί κατά τη διάρκεια της δημιουργίας των IP πρωτοκόλλων. Σε ένα IP δίκτυο, η κύρια υπηρεσία που παρέχεται είναι η best effort. Λόγω των περιορισμένων πόρων των δικτύων, σε περίπτωση αυξημένου φόρτου, ενδέχεται να εμφανιστεί το ανεπιθύμητο φαινόμενο της συμφόρησης, κάτι που μπορεί να προκαλέσει αυξημένες καθυστερήσεις στην παράδοση των πακέτων, ακόμη και την απώλεια τους. Εφαρμογές που επιθυμούν κάποια ποιότητα υπηρεσίας, ενδέχεται να αντιμετωπίσουν πρόβλημα στην απόδοση τους.

Για την αντιμετώπιση τέτοιου είδους φαινομένων, προτάθηκαν διάφοροι τρόποι διαχείρισης των περιορισμένων πόρων του δικτύου. Ένας από αυτούς, είναι και η αρχιτεκτονική DiffServ. Η αρχιτεκτονική, ορίζει μια σειρά μηχανισμών που επιδρούν στις ροές των πακέτων. Μέρος των μηχανισμών αυτών, είναι και οι αλγόριθμοι χρονοδρομολόγησης που διαχειρίζονται τη σειρά εξόδου των πακέτων από τις ουρές αναμονής.

Η υλοποίηση των δικτύων, προϋποθέτει την προσεκτική σχεδίαση και μελέτη τους. Ο ns-3, είναι ένας σύγχρονος εξομοιωτής δικτύων, που μπορεί να χρησιμοποιηθεί στη μελέτη και σχεδίαση IP δικτύων. Λόγω όμως της πρόσφατης δημιουργίας του, δεν παρέχει ακόμη υποστήριξη για την αρχιτεκτονική DiffServ.

Στα πλαίσια της διπλωματικής αυτής, μελετήθηκαν οι δυνατότητες που παρέχει ο ns-3 στα ασύρματα δίκτυα και δημιουργήθηκε ένα καινούργιο module για τον ns-3, στο οποίο υλοποιήθηκαν μηχανισμοί της αρχιτεκτονικής. Στη συνέχεια, δημιουργήθηκε μια τοπολογία δικτύου, που μέσα από μια σειρά προσομοιώσεων, αναλύθηκε η συμπεριφορά του δικτύου σε περιπτώσεις συμφόρησης, καθώς η επίδραση που είχαν σε αυτήν οι μηχανισμοί DiffServ του καινούργιου module.

Abstract

The Internet expansion resulted in serving applications through IP networks, that had not been considered during the design of the IP protocols. In IP networks, the main service provided is best effort. Due to the limited network resources, during increased loads, it is possibly for undesirable congestion to emerge, which can lead to increased packet delays and even in packet losses. Applications that require a quality of service (QoS) to function properly, may encounter performance issues during congestion.

In order to confront the above issues, there have been proposed various mechanisms for managing the limited network resources. The DiffServ architecture is one of them. The architecture, defines a set of mechanisms that can be used from network routers and react on passing packet flows. Part of these mechanisms, are scheduling algorithms for managing multiple queue's.

The implementation of networks, requires careful design and testing. The ns-3 simulator, is a state-of-art network simulator, that can be used in simulating IP networks. Due to its relative early existence, it lacks support for the DiffServ architecture.

As part of this thesis, there where studied the capabilities of ns-3 in wireless networks. Also, a new module was created for ns-3, which included mechanism defined in DiffServ architecture. Then, a network topology was created and its performance was evaluated, in cases of network congestion and when the implemented DiffServ mechanisms were applied.

Περιεχόμενα

Αντί προλόγου i

Περίληψη i

Abstract ii

Περιεχόμενα iii

Κατάλογος σχημάτων vii

Κατάλογος πινάκων viii

1 ΕΙΣΑΓΩΓΗ 1

1.1 Το διαδίκτυο . . . 1

1.2 Ποιότητα υπηρεσίας . . . 2

1.3 Μοντέλα αναφοράς . . . 3

1.4 Εξομοιωτής δικτύων ns-3 . . . 4

2 ΠΟΙΟΤΗΤΑ ΥΠΗΡΕΣΙΑΣ 5

2.1 Εισαγωγή . . . 5

2.2 Μετρικές Ποιότητας . . . 6

2.3 Τύποι Ποιότητας Υπηρεσίας . . . 7

2.3.1 Integrated services (IntServ) . . . 7

2.3.2 Differentiated Services (DiffServ) . . . 8

3 DIFFERENTIATED SERVICES 9

3.1 Εισαγωγή . . . 9

3.2 Service Level Agreement . . . 11

3.3 Per Hop Behaviors . . . 12

3.4 Μηχανισμοί DiffServ . . . 13

3.5 Μηχανισμός ταξινόμησης . . . 14

3.6 Αστυνόμευση της κίνησης . . . 15

3.7 Αλγόριθμοι Token bucket και Leaky bucket . . . 15

3.8 Active Queue Management . . . 16
3.8.1 Explicit Congestion Notification (ECN) . . . 17
3.8.2 Random early detection (RED) . . . 18
3.8.3 Weighted random early detection (WRED) . . . 19
3.9 Χρονοδρομολόγηση . . . 19
3.9.1 First In First Out (FIFO) . . . 19
3.9.2 Priority queuing (PQ) . . . 20
3.9.3 Fair queuing (FQ) . . . 20
3.9.4 Weighted fair queuing (WFQ) . . . 20
3.9.5 Round Robin (RR) . . . 21
3.9.6 Weighted round robin (WRR) . . . 21
3.9.7 Deficit (weighted) round robin (DWRR) . . . 21
3.9.8 Modified deficit round robin (MDRR) . . . 22

4 ΕΞΟΜΟΙΩΤΗΣ ΔΙΚΤΥΩΝ ns-3 27
4.1 Εισαγωγή . . . 27
4.2 Η δομή του ns-3 . . . 28
4.3 Βασικές δομές του ns-3 . . . 29
4.4 ns-3 modules . . . 31

5 ns-3 ΚΑΙ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ 33
5.1 Εισαγωγή . . . 33
5.2 ns-3 modules . . . 34
5.2.1 antenna module . . . 34
5.2.2 aodv module . . . 34
5.2.3 buildings module . . . 35
5.2.4 dsvd module . . . 37
5.2.5 dsr module . . . 38
5.2.6 energy module . . . 39
5.2.7 lte module . . . 40
5.2.8 mesh module . . . 41
5.2.9 mobility module . . . 41
5.2.10 olsr module . . . 42
5.2.11 propagation module . . . 42
5.2.12 spectrum module . . . 43
5.2.13 uan module . . . 44
5.2.14 wifi module . . . 46
5.2.15 wimax module . . . 48
5.3 Συμπεράσματα . . . 49

6 ΥΛΟΠΟΙΗΣΗ ΤΟΥ diffserv module **51**
6.1 Εισαγωγή . . . 51
6.2 Δομή του module . . . 52
6.3 diffserv model . . . 52
6.3.1 Κλάση: DiffServQueue . . . 53
6.3.2 Κλάση: DiffServ . . . 55
6.3.3 Κλάση: Policy . . . 56
6.3.4 Κλάση: Mdrr . . . 60
6.4 diffserv helper . . . 62
6.5 diffserv utils . . . 62

7 ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ diffserv module **63**
7.1 Εισαγωγή . . . 63
7.2 Παραγωγή κίνησης . . . 64
7.2.1 VoIP model . . . 64
7.2.2 Video streaming model . . . 65
7.3 Τοπολογία πειραμάτων . . . 65
7.4 Διεξαγωγή πειραμάτων . . . 68
7.4.1 Πείραμα 1 . . . 69
7.4.2 Πείραμα 2 . . . 70
7.4.3 Πείραμα 3 . . . 71
7.4.4 Πείραμα 4 . . . 72
7.4.5 Πείραμα 5 . . . 73
7.4.6 Πείραμα 6 . . . 74
7.5 Συμπεράσματα . . . 75

8 ΕΠΙΛΟΓΟΣ **77**

Α΄ Μετρήσεις Πειραμάτων **79**

Βιβλιογραφία **87**

Η σελίδα έχει μείνει εσκεμμένα κενή

Κατάλογος σχημάτων

1.1 Απεικόνιση μέρους του Internet από το project Opte [28]. 2

3.1 Αναπαράσταση ενός DS domain. 10
3.2 Διάγραμμα ροής DiffServ μηχανισμών. 13
3.3 Πεδίο TOS της IPv4 επικεφαλίδας. 14
3.4 figure 14
3.5 Πεδίο DSCP των IPv4 και IPv6 επικεφαλίδων. 14

4.1 Η δομή του ns-3. 28

6.1 Διάγραμμα ροής μηχανισμών της μεθόδου: DoEnqueue. 54

7.1 Αναπαράσταση της τοπολογίας των πειραμάτων από τον NetAnim. 66

Κατάλογος πινάκων

1.1 Επίπεδα μοντέλου αναφοράς OSI. 3
1.2 Επίπεδα πρωτοκόλλων του Internet (TCP/IP). 4

4.1 ns-3 modules 31

Περιεχόμενα **61**

7.1 Παράμετροι OnOff VoIP μοντέλου. 65
7.2 Παράμετροι τοπολογίας δοκιμών. 66
7.3 Βελτίωση του jitter με χρήση του MDRR STRICT. 72
7.4 Βελτίωση της απόδοσης των εφαρμογών με χρήση του MDRR ALTERNATE. 74

A'.1 Test1, Flow:1. 80
A'.2 Test1, Flow:2. 80
A'.3 Test1, Flow:3. 80
A'.4 Test1, Flow:4. 80
A'.5 Test1, Flow:5. 80
A'.6 Test1, Flow:6. 80
A'.7 Test2, Flow:1. 81
A'.8 Test2, Flow:2. 81
A'.9 Test2, Flow:3. 81
A'.10 Test2, Flow:4. 81
A'.11 Test2, Flow:5. 81
A'.12 Test2, Flow:6. 81
A'.13 Test2, Flow:7. 82
A'.14 Test3, Flow:1. 82
A'.15 Test3, Flow:2. 82
A'.16 Test3, Flow:3. 82
A'.17 Test3, Flow:4. 82
A'.18 Test3, Flow:5. 82
A'.19 Test3, Flow:6. 83

Α΄.20 Test3, Flow:7. 83
Α΄.21 Test4, Flow:1. 83
Α΄.22 Test4, Flow:2. 83
Α΄.23 Test4, Flow:3. 83
Α΄.24 Test4, Flow:4. 83
Α΄.25 Test4, Flow:5. 84
Α΄.26 Test4, Flow:6. 84
Α΄.27 Test4, Flow:7. 84
Α΄.28 Test5, Flow:1. 84
Α΄.29 Test5, Flow:2. 84
Α΄.30 Test5, Flow:3. 84
Α΄.31 Test5, Flow:4. 85
Α΄.32 Test5, Flow:5. 85
Α΄.33 Test5, Flow:6. 85
Α΄.34 Test5, Flow:7. 85
Α΄.35 Test6, Flow:1. 85
Α΄.36 Test6, Flow:2. 85
Α΄.37 Test6, Flow:3. 86
Α΄.38 Test6, Flow:4. 86
Α΄.39 Test6, Flow:5. 86
Α΄.40 Test6, Flow:6. 86
Α΄.41 Test6, Flow:7. 86

Η σελίδα έχει μείνει εσκεμμένα κενή

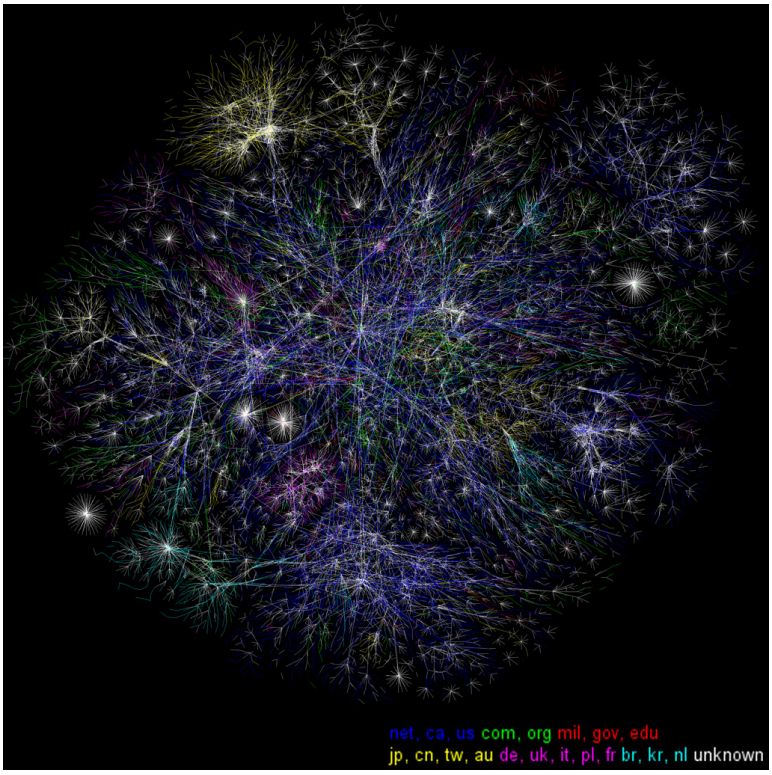
Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

1.1 Το διαδίκτυο

Διανύοντας ήδη τη δεύτερη δεκαετία του 21ου αιώνα, ζούμε σε μια εποχή όπου η ψηφιακή τεχνολογία έχει γίνει κομμάτι της καθημερινότητάς μας. Η συνεχιζόμενη εξέλιξη και ωρίμανση της επιστήμης των υπολογιστών, έδωσε σταδιακά τα μέσα για τη μετάβαση στην ψηφιακή εποχή. Κεντρικό ρόλο στην ολοκλήρωση της μετάβασης αυτής, έπαιξε η δυνατότητα ανταλλαγής ψηφιακών δεδομένων μέσω δικτύων και τηλεπικοινωνιακών μέσων. Η εξάπλωση του διαδικτύου (Internet), έκανε δυνατή τη διασύνδεση πολλών και ετερογενών δικτύων σε παγκόσμιο επίπεδο (Σχήμα: 1.1).

Η διασύνδεση των δικτύων μέσω του διαδικτύου, έγινε εφικτή με τη χρήση ενός κοινού συνόλου πρωτοκόλλων επικοινωνίας, που ονομάζεται Internet Protocol Suite, κοινώς γνωστού και ως TCP/IP, λόγω της σπουδαιότητας των δύο αυτών πρωτοκόλλων του διαδικτύου. Τα πρωτόκολλα του διαδικτύου, σχεδιάστηκαν με την υπόθεση ότι η υποδομή του δικτύου είναι εγγενώς μη αξιόπιστη και δυναμική με την έννοια της διαθεσιμότητας των κόμβων και των συνδέσμων που το απαρτίζουν. Έτσι δεν υπάρχει κάποιος κεντρικός συντονισμός ή έλεγχος της απόδοσης του δικτύου. Επίσης, για τη διατήρηση χαμηλής πολυπλοκότητας στο δίκτυο, το μεγαλύτερο βάρος διεξαγωγής της επικοινωνίας έχει μεταφερθεί στα άκρα της επικοινωνίας, κάτι που ονομάζεται και *αρχή από άκρο-σε-άκρο* (end-to-end principle). Αυτό έχει σαν συνέπεια, η κύρια λειτουργία των δρομολογητών που βρίσκονται στο μονοπάτι μετάδοσης, να είναι η προώθηση κάθε πακέτου στον αμέσως επόμενο γνωστό προορισμό (hop), μέχρι αυτό να καταλήξει στον τελικό προορισμό του.



Σχήμα 1.1: Απεικόνιση μέρους του Internet από το project Opte [28].

1.2 Ποιότητα υπηρεσίας

Λόγω των σχεδιαστικών αρχών των πρωτοκόλλων του διαδικτύου, η κύρια υπηρεσία που παρέχεται σε ένα IP δίκτυο, είναι η *παράδοση καλύτερης προσπάθειας* (best effort delivery). Υπηρεσία που δεν δίνει εγγυήσεις για την παράδοση των δεδομένων ή την παροχή κάποιας προτεραιότητας στην εξυπηρέτηση μιας εφαρμογής, ακόμα και όταν αυτή είναι αναγκαία για την καλή λειτουργία της. Έτσι η υπηρεσία που λαμβάνει τελικά ο χρήστης, εξαρτάται άμεσα από τον τρέχοντα φόρτο του δικτύου που τον εξυπηρετεί.

Σύγχρονες εφαρμογές του διαδικτύου, όπως οι ροές πολυμέσων (streaming media), η IP τηλεφωνία (VoIP), η τηλεδιάσκεψη (Videoconferencing), καθώς επίσης και κρίσιμες εφαρμογές τηλεϊατρικής και ελέγχου συστημάτων από απόσταση, απαιτούν την παροχή κάποιας ποιότητας υπηρεσίας (*Quality of Service - QoS*), για να εξασφαλίσουν την καλή απόδοση τους. Αν και είναι ακόμα υπό μελέτη η παροχή ποιότητας υπηρεσίας σε μεγάλα και ποικιλόμορφα δίκτυα (όπως το διαδίκτυο), έχουν προταθεί δύο βασικές αρχιτεκτονικές για το σκοπό αυτό. Η πρώτη ονομάζεται Integrated Services (IntServ), η οποία χρησιμοποιεί ένα πρωτόκολλο κατακράτησης πόρων: Resource Reservation Protocol (RSVP), για την εξυπηρέτηση κάθε ροής δεδομένων του δικτύου. Η δεύτερη ονομάζεται Differentiated Services (DiffServ), η οποία κατηγοριοποιεί τις ροές δεδομένων σε κάποιες καθορισμένες κλάσεις εξυπηρέτησης. Έπειτα γίνεται διαφοροποιημένη διαχείριση των κλάσεων, με στόχο την παροχή διαφορετικής ποιότητας υπηρεσίας σε κάθε κλάση.

Η αρχιτεκτονική DiffServ, βασίζεται σε μια σειρά μηχανισμών που εφαρμόζονται από τους δρομολογητές του δικτύου, στις διερχόμενες ροές δεδομένων. Οι μηχανισμοί αναλαμβάνουν την κατηγοριοποίηση των διερχόμενων πακέτων κάθε ροής στις προκαθορισμένες κλάσεις εξυπηρέτησης, καθώς και την εξυπηρέτησή των κλάσεων με διαφοροποιημένο τρόπο. Κάθε κλάση, οδηγείται (συνήθως) και σε χωριστή ουρά αναμονής (queue), για την προσωρινή αποθήκευση των διερχόμενων πακέτων μέχρι να γίνει εφικτή η μετάδοσή τους από το δίκτυο. Τμήμα της διαχείρισης των ουρών αναμονής, είναι οι αλγόριθμοι χρονοδρομολόγησης, οι οποίοι αναλαμβάνουν τη διαχείριση της σειράς εξόδου των πακέτων που βρίσκονται σε αυτές. Ο αλγόριθμος Modified Deficit Round Robin (MDRR), είναι παράδειγμα αλγόριθμου χρονοδρομολόγησης που θα μας απασχολήσει στα πλαίσια της διπλωματικής αυτής.

1.3 Μοντέλα αναφοράς

Μιλώντας για δίκτυα υπολογιστών, είναι σημαντικό να αναφερθεί ο τρόπος οργάνωσης τους. Τα συστήματα επικοινωνίας, όπως τα δίκτυα υπολογιστών, είναι ένα σύνολο υλικού και λογισμικού που λειτουργούν συντονισμένα για την παροχή της επικοινωνίας. Στη σχεδίαση τους έχει επικρατήσει η λογική των επιπέδων δομημένα σε στοίβα. Κάθε επίπεδο έχει μια καθορισμένη λειτουργία, ένα σύνολο πρωτοκόλλων για να την διεκπεραιώσει, εξυπηρετεί το επάνω επίπεδο του, και εξυπηρετείται από το κάτω. Έτσι δημιουργείται μια αλυσίδα που ενώνει τα επίπεδα μεταξύ τους και φέρουν εις πέρας την επικοινωνία.

Ο οργανισμός ISO (International Organization for Standardization) δημιούργησε για το σκοπό αυτό το μοντέλο αναφοράς OSI (Open Systems Interconnection), που αποτελείται από επτά επίπεδα (Πίνακας: 1.1). Σε κάθε επίπεδο μεταδίδονται τα δεδομένα του πρωτοκόλλου, που ονομάζονται Protocol Data Units (PDU), προς το κάτω επίπεδο, όπου όταν παραλαμβάνονται ονομάζονται Service Data Units (SDU). Μετατρέπονται σε PDU του επιπέδου και συνεχίζουν την κάθοδό τους. Η μέθοδος μετατροπής των δεδομένων κατά την κάθοδό τους στα επίπεδα, είναι αυτή της ενθυλάκωσης (encapsulation), όπου προστίθενται στα δεδομένα επικεφαλίδες (headers) ή ουρές (footers), ή και τα δύο, και αφαιρούνται κατά την άνοδο των δεδομένων στα επίπεδα του παραλήπτη.

7	Εφαρμογών (Application)
6	Παρουσίασης (Presentation)
5	Συνόδου (Session)
4	Μεταφοράς (Transport)
3	Δικτύου (Network)
2	Ζεύξης δεδομένων (Data link)
1	Φυσικό (Physical)

Πίνακας 1.1: Επίπεδα μοντέλου αναφοράς OSI.

Αντίστοιχα ο οργανισμός IETF (Internet Engineering Task Force), που είναι υπεύθυνος για τη δημιουργία των πρωτοκόλλων του Internet, είχε προηγηθεί, με τη δημιουργία αντίστοιχης δομής στα πρωτόκολλα του Internet και της επικοινωνίας μεταξύ τους με τη μέθοδο της ενθυλάκωσης. Κατά την ερμηνεία του Andrew S. Tanenbaum [1], τα πρωτόκολλα του Internet αποτελούν πέντε επίπεδα (Πίνακας: 1.2).

5	Εφαρμογών (Application)
4	Μεταφοράς (Transport)
3	Διαδικτύου (Internet)
2	Ζεύξης δεδομένων (Data link)
1	Φυσικό (Physical)

Πίνακας 1.2: Επίπεδα πρωτοκόλλων του Internet (TCP/IP).

Σημαντική διαφορά μεταξύ των δύο προσεγγίσεων, είναι ότι ο διαχωρισμός των πρωτοκόλλων του διαδικτύου σε επίπεδα, δεν ήταν σχεδιαστική αρχή, με την αυστηρή έννοια του OSI μοντέλου. Έτσι η άμεση ή αυστηρή αντιστοίχηση μεταξύ των δύο, να θεωρείται λάθος. Οι μηχανισμοί παροχής ποιότητας υπηρεσίας που έχουν αναφερθεί, υλοποιούνται κυρίως στο επίπεδο διαδικτύου των πρωτοκόλλων του Internet, κάνοντας χρήση μέρους της επικεφαλίδας του IP πρωτοκόλλου (Internet Protocol).

1.4 Εξομοιωτής δικτύων ns-3

Ένα από τα πεδία εφαρμογής των υπολογιστών, είναι η προσομοίωση συστημάτων, όπως η προσομοίωση δικτύων υπολογιστών. Παράδειγμα σύγχρονου εξομοιωτή είναι ο εξομοιωτής δικτύων ns-3. Ως ένα σχετικά πρόσφατο project ανοιχτού και ελεύθερου κώδικα (ξεκίνησε το 2006), βρίσκεται σε έντονη ανάπτυξη από μια μεγάλη και ενθουσιώδη κοινότητα εθελοντών. Δημιουργήθηκε με πρωτεύοντα στόχο, τη χρήση του από την ακαδημαϊκή και εκπαιδευτική κοινότητα.

Ο εξομοιωτής είναι πλέον δομημένος σε μορφή λειτουργικών μονάδων (modules), κάτι που διευκολύνει την ανάπτυξη, τη συντήρηση και τη χρήση του. Στα πλαίσια της διπλωματικής αυτής, έγινε μελέτη των δυνατοτήτων που προσφέρει ο εξομοιωτής για τα ασύρματα δίκτυα. Επίσης έγινε επέκταση των λειτουργιών του εξομοιωτή με την ανάπτυξη ενός καινούργιου module. Σε αυτό υλοποιήθηκαν μηχανισμοί της αρχιτεκτονικής DiffServ συμπεριλαμβανομένου και του αλγόριθμου χρονοδρομολόγησης MDRR.

Κεφάλαιο 2

ΠΟΙΟΤΗΤΑ ΥΠΗΡΕΣΙΑΣ

2.1 Εισαγωγή

Σε ένα δίκτυο υπολογιστών ή γενικά σε ένα δίκτυο μεταγωγής πακέτων, με τον όρο ποιότητα υπηρεσίας (Quality of Service - QoS), αναφερόμαστε στους μηχανισμούς διαχείρισης των πόρων του δικτύου, που αποσκοπούν στο να παρέχουμε διαφοροποιημένη υπηρεσία στις ροές δεδομένων των εφαρμογών και κατά συνέπεια στους χρήστες που τις χρησιμοποιούν. Αναφερόμαστε επίσης στην παροχή εγγυήσεων για την απόδοση μιας ροής δεδομένων. Οι εγγυήσεις απόδοσης είναι ιδιαίτερα χρήσιμες, σε περιπτώσεις όπου η χωρητικότητα του δικτύου είναι περιορισμένη και εξυπηρετούνται εφαρμογές που έχουν συγκεκριμένες απαιτήσεις στην εξυπηρέτηση τους. Για παράδειγμα, υπάρχουν εφαρμογές που απαιτούν σταθερό ρυθμό μετάδοσης (bit rate) και είναι ευαίσθητες στην καθυστέρηση (delay), όπως οι εφαρμογές ροής πολυμέσων (streaming multimedia).

Σε ένα δίκτυο βασισμένο στο πρωτόκολλο IP (Internet Protocol) και κατά συνέπεια στο διαδίκτυο, η κύρια υπηρεσία που παρέχεται είναι η μετάδοση αυτοδύναμων πακέτων με τη μέθοδο της καλύτερης προσπάθειας. Αυτό έχει σαν συνέπεια όλα τα πακέτα να αντιμετωπίζονται όμοια και χωρίς κάποιες εγγυήσεις. Το δίκτυο προσπαθεί να εξυπηρετήσει όση περισσότερη κίνηση μπορεί κάθε στιγμή.

Ένα ανεπιθύμητο φαινόμενο που μπορεί να παρουσιαστεί σε ένα δίκτυο, είναι το φαινόμενο της συμφόρησης. Αυτό εμφανίζεται όταν υπάρχει περισσότερη εισερχόμενη κίνηση σε ένα δρομολογητή του δικτύου, από αυτήν που μπορεί να εξυπηρετήσει. Τα πακέτα διασχίζοντας το δρομολογητή, αποθηκεύονται προσωρινά σε ουρές εισόδου περιμένοντας την επεξεργασία τους και ουρές εξόδου περιμένοντας την μετάδοση τους. Σε περιπτώσεις συμφόρησης εμφανίζεται, αυξανόμενη καθυστέρηση των διερχόμενων πακέτων, όσο αυξάνει η πληρότητα των ουρών, και πιθανή απόρριψη πακέτων όταν αυτές γεμίσουν, λόγω της πεπερασμένης χωρητικότητας τους. Το φαινόμενο της συμφόρησης είναι ιδιαίτερα ανεπιθύμητο σε εφαρμογές που απαιτούν κάποια συγκεκριμένη ποιότητα υπηρεσίας από το δίκτυο για να λειτουργήσουν απροβλημάτιστα. Το δίκτυο κατά συνέπεια χρειάζεται κάποιο μηχανισμό διαχείρισης των πόρων του, για να μπορέσει να εξυπηρετήσει και τέτοιου είδους εφαρμογές.

2.2 Μετρικές Ποιότητας

Παρακάτω αναφέρονται οι ποιο κοινές μετρικές δικτύων που επηρεάζουν την απόδοση των εφαρμογών που εξυπηρετούν:

- **Χωρητικότητα (bandwidth)**: Είναι ο ρυθμός μετάδοσης των δεδομένων μετρημένος σε bits per second (bps). Η χωρητικότητα χαρακτηρίζεται από τις παρακάτω ποσότητες:
  - *Μέγιστο μέγεθος καταιγισμού (maximum throughput).*
  - *Μέγιστη χωρητικότητα (peak bandwidth).*
  - *Ελάχιστη εγγυημένη χωρητικότητα (guaranteed bandwidth).*
  - *Μέση χωρητικότητα (mean bandwidth).*

  Το διαθέσιμο bandwidth είναι αποτέλεσμα της χωρητικότητας του καναλιού μετάδοσης, της τεχνολογίας μετάδοσης που χρησιμοποιείται, καθώς επίσης και του ενδεχόμενου καταμερισμού που μπορεί να πραγματοποιείται (στο συνολικό διαθέσιμο bandwidth), λόγω εξυπηρέτησης πολλαπλών χρηστών.

- **Καθυστέρηση (delay)**: Είναι ο χρόνος που χρειάζεται ένα πακέτο για να μεταδοθεί, από την έναρξη της μετάδοσης μέχρι να παραληφθεί και το τελευταίο bit του πακέτου. Η καθυστέρηση είναι άθροισμα των παρακάτω επιμέρους χρόνων:
  - ***Χρόνος μετάδοσης***: Ο χρόνος που απαιτείται για την τοποθέτηση κάθε bit ενός πακέτου στο κανάλι μετάδοσης. Είναι ο λόγος του μεγέθους του μεταδιδόμενου πακέτου σε bit προς το διαθέσιμο bandwidth.
  - ***Χρόνος διάδοσης***: Ο χρόνος που χρειάζεται το πρώτο bit ενός πακέτου να φτάσει στον προορισμό του. Είναι ο λόγος της απόστασης του καναλιού μετάδοσης προς την ταχύτητα διάδοσης μέσα στο κανάλι.
  - ***Χρόνος επεξεργασίας***: Ο χρόνος που δαπανάται σε ένα δρομολογητή του δικτύου κατά την επεξεργασία του πακέτου, όπως για ανίχνευση και διόρθωση λαθών, εξέταση των επικεφαλίδων του πακέτου για τη δρομολόγησή του στον επόμενο προορισμό, λειτουργίες κρυπτογράφησης κ.ο.κ. Είναι ανάλογη του πλήθους και της υπολογιστικής πολυπλοκότητας των εργασιών που εκτελούνται, και αντιστρόφως ανάλογη της διαθέσιμης υπολογιστικής ισχύος του δρομολογητή.
  - ***Χρόνος αναμονής***: Ο συνολικός χρόνος αναμονής ενός πακέτου στις ουρές (εισόδου και εξόδου) ενός δρομολογητή, που δαπανάται μέχρι να επεξεργαστεί ή να αποσταλεί. Είναι ανάλογος της πληρότητας των ουρών κατά την άφιξη του πακέτου στο δρομολογητή.

- **Διακύμανση καθυστέρησης πακέτων (Packet Delay Variation - PVD) ή jitter.** Είναι η διαφορά της καθυστέρησης μεταξύ δύο διαδοχικών πακέτων. Εφαρμογές πραγματικού χρόνου χρειάζονται να διατηρείται το jitter σε χαμηλά επίπεδα για να έχουν καλή απόδοση.

- **Απώλεια πακέτων (packet loss).** Είναι το ποσοστό των πακέτων που μεταδόθηκαν, αλλά δεν παραδόθηκαν ποτέ στον προορισμό τους, ή παραδόθηκαν με λάθη που αδυνατεί να διορθώσει ο κώδικας ανίχνευσης και διόρθωσης λαθών του δικτύου. Η απώλεια πακέτων μπορεί να ευθύνεται σε αποτυχία του δικτύου (κόμβων, συνδέσμων), σε συμφόρηση, καθώς και στην παρουσία υψηλού επιπέδου θορύβου στο κανάλι κατά τη μετάδοση.

2.3 Τύποι Ποιότητας Υπηρεσίας

Στα σύγχρονα IP δίκτυα μεταγωγής πακέτων, έχουν προταθεί δύο βασικές αρχιτεκτονικές παροχής ποιότητας υπηρεσίας. Όπως έχει ήδη αναφερθεί, είναι η Integrated services (IntServ) και η Differentiated services (DiffServ). Η IntServ αρχιτεκτονική είναι προγενέστερη και βασίζεται στην κατακράτηση πόρων σε κάθε ροή που εξυπηρετεί (χαρακτηρίζεται και ως σύστημα: fine-grained QoS). Από την άλλη, η DiffServ βασίζεται στη κατηγοριοποίηση των ροών σε κλάσεις, και διαχείριση των κλάσεων αυτών (έτσι χαρακτηρίζεται και ως σύστημα: coarse-grained QoS). Παρακάτω περιγράφονται ποιο αναλυτικά οι δύο αρχιτεκτονικές.

2.3.1 Integrated services (IntServ)

Βασίζεται στο πρωτόκολλο RSVP (Resource Reservation Protocol), που αναλαμβάνει το συντονισμό της κατακράτησης των πόρων του δικτύου σε κάθε ροή. Σαν ροή, ορίζεται μια ανεξάρτητη μη διευθυνόμενη ροή δεδομένων μεταξύ δύο εφαρμογών. Κάθε ροή χαρακτηρίζεται από την παρακάτω πεντάδα:

- *IP διεύθυνση αποστολέα*
- *IP διεύθυνση παραλήπτη*
- *Αριθμός πρωτοκόλλου μεταφοράς*
- *Αριθμός port αποστολέα*
- *Αριθμός port παραλήπτη*

Υπάρχουν δύο τύποι υπηρεσιών που μπορεί να παρέχει το πρωτόκολλο RSVP. Η πρώτη είναι μια αυστηρή και εγγυημένη υπηρεσία (guaranteed) που παρέχει αυστηρές εγγυήσεις στη μέγιστη καθυστέρηση από άκρο σε άκρο, καθώς επίσης και εγγυημένη χωρητικότητα, υπό την προϋπόθεση ότι το προφίλ της κίνησης συμμορφώνεται με αυτό της κράτησης που έχει γίνει.
Η δεύτερη είναι μια υπηρεσία ελεγχόμενου φόρτου (controlled load), που παρέχει μια υπηρεσία καλύτερη από την best effort με χαμηλή καθυστέρηση, και προϋποθέτει χαμηλό ή μέτριο φόρτο στο δίκτυο.

Πλεονέκτημα της αρχιτεκτονικής είναι η εγγυημένη παροχή ποιότητας υπηρεσίας σε κάθε ροή που εξυπηρετείται.

Μειονέκτημα της αρχιτεκτονικής είναι η δυσκολία επεκτασιμότητας (scalability). Η υλοποίηση της αρχιτεκτονικής προϋποθέτει ότι κάθε συσκευή που συμμετέχει στην μετάδοση θα πρέπει να υποστηρίζει το πρωτόκολλο RSVP, για να έχει τη δυνατότητα δέσμευσης των πόρων στο δίκτυο. Επίσης λόγω του ότι οι κρατήσεις που έχουν γίνει στο δίκτυο πρέπει να ανανεώνονται περιοδικά, προστίθεται φόρτος στο δίκτυο από την λειτουργία του πρωτοκόλλου. Η αποτυχία ανανέωσης μιας κράτησης, λόγω πιθανής απώλειας πακέτων που χρησιμοποιούνται για τη λειτουργία, μπορεί να προκαλέσει τη λήξη μιας κράτησης μετά από κάποιο χρονικό διάστημα (timeout). Τέλος, κάθε δρομολογητής θα πρέπει να διατηρεί όλες τις ενεργές κρατήσεις που διέρχονται από το μονοπάτι που ανήκει. Αυτό μπορεί να προκαλέσει σημαντικά προβλήματα στη διαχείριση μεγάλου όγκου κρατήσεων. Τα παραπάνω μειονεκτήματα έχουν αντιμετωπιστεί με βελτιώσεις και προσθήκες στο RSVP πρωτόκολλο, που βοηθούν στην ευκολότερη επεκτασιμότητα του.

2.3.2 Differentiated Services (DiffServ)

Τα προβλήματα που παρουσιάζονται στην παροχή ποιότητας υπηρεσίας σε κάθε ροή χωριστά, οδήγησαν στις προδιαγραφές της DiffServ αρχιτεκτονικής. Η ομαδοποίηση των ροών σε κλάσεις εξυπηρέτησης και η παροχή ποιότητας υπηρεσίας σε αυτές, είναι ένας τρόπος να μειωθεί δραστικά ο φόρτος που προέκυπτε με τις χωριστές κρατήσεις για κάθε ροή. Επίσης έπρεπε να αντιμετωπιστεί η πολυπλοκότητα υλοποίησης του πρωτοκόλλου από το δίκτυο, όπως και ο παραγόμενος φόρτος προς το δίκτυο κατά τη λειτουργία του (overhead).

Η αρχιτεκτονική DiffServ βασίζεται στη χρήση του πεδίου TOS της IP επικεφαλίδας (που μετονομάστηκε σε πεδίο DSCP). Σε κάθε πακέτο, το πεδίο μαρκάρεται με κωδικό που αντιστοιχεί σε κάποια κλάση εξυπηρέτησης. Έπειτα προωθείται με την Per Hop Behavior (PHB) της κλάση που ανήκει. Ο όρος PHB αναφέρεται στο είδος της πολιτικής που εφαρμόζεται και την προτεραιότητα που λαμβάνουν τα πακέτα μιας κλάσης κατά την προώθηση τους από το δίκτυο. Η κλάση εξυπηρέτησης που χρησιμοποιείται για τα πακέτα μιας ροής, είναι αποτέλεσμα συμφωνίας μεταξύ της εφαρμογής που την παράγει και του δικτύου. Οι απαιτήσεις της εφαρμογής, οι δυνατότητες του δικτύου τη δεδομένη στιγμή καθώς και η συμμόρφωση της εφαρμογής στη συμφωνία, επηρεάζουν την ποιότητα υπηρεσίας που τελικά παρέχεται.

Πλεονέκτημα της αρχιτεκτονικής είναι η ευκολότερη επεκτασιμότητα, η μειωμένη πολυπλοκότητα στην υλοποίηση και ο μειωμένος φόρτος (overhead) που προστίθεται στο δίκτυο κατά τη λειτουργία της, σε σχέση με την IntServ αρχιτεκτονική.

Μειονέκτημα της αρχιτεκτονικής είναι η δυσκολία πρόβλεψης της συμπεριφοράς του δικτύου από άκρη σε άκρη. Η αρχιτεκτονική ορίζει τις κλάσεις εξυπηρέτησης και την ποιότητα υπηρεσίας που παρέχει κάθε μία, αλλά δεν επιβάλει τον τρόπο υλοποίησης τους. Το δίκτυο μπορεί να αποτελείται από τμήματα που χρησιμοποιούν διαφορετική πολιτική στην προώθηση του ίδιου πακέτου. Έτσι μπορεί η ποιότητα υπηρεσίας που τελικά θα λάβει το πακέτο να μην είναι η αναμενόμενη.

Κεφάλαιο 3

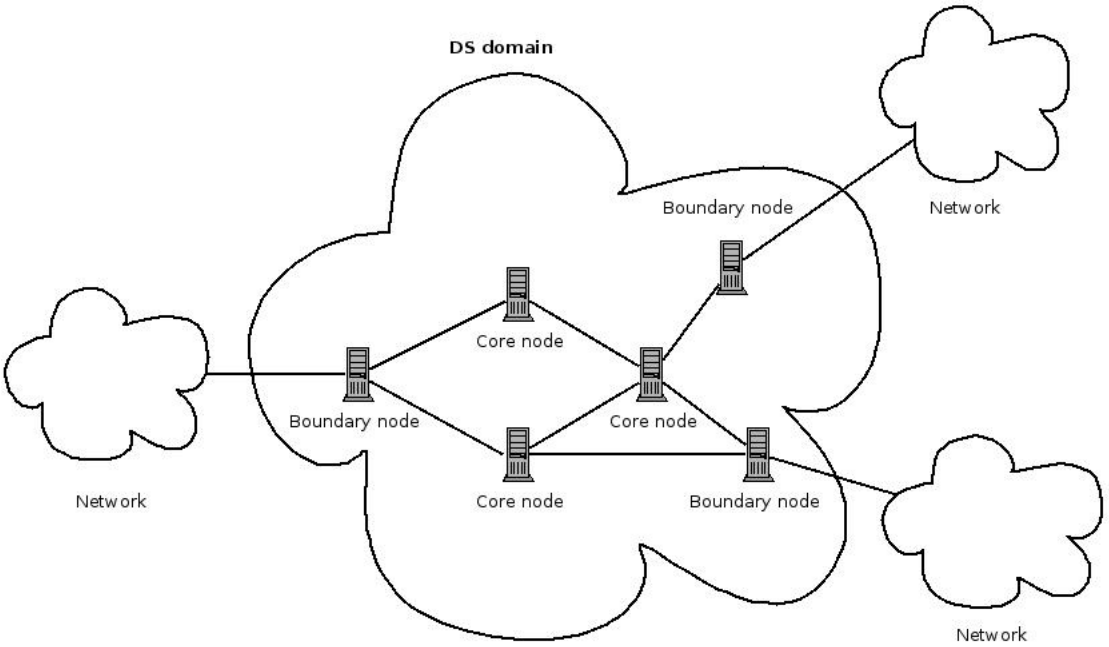
DIFFERENTIATED SERVICES

3.1 Εισαγωγή

Όπως ήδη έχει αναφερθεί, η DiffServ αρχιτεκτονική επιτυγχάνει την παροχή ποιότητας υπηρεσίας, με την ταξινόμηση της κίνησης σε κλάσεις εξυπηρέτησης και τη διαφοροποιημένη διαχείριση αυτών. Η αρχιτεκτονική ορίζει μια σειρά μηχανισμών που εφαρμόζονται πάνω στις διερχόμενες ροές που εξυπηρετεί το δίκτυο. Ένα δίκτυο που υποστηρίζει την αρχιτεκτονική, ονομάζεται DS περιοχή (region). Η περιοχή αυτή μπορεί να αποτελείται από έναν ή περισσότερους DS domain (Σχήμα: 3.1). Ως DS domain ορίζεται ένα δίκτυο (ή μέρος ενός δικτύου) που λειτουργεί κάτω από την ίδια διαχείριση εφαρμόζοντας κοινή πολιτική για κάθε κλάση εξυπηρέτησης που υποστηρίζει. Ο DS domain μπορεί να αποτελείται από έναν ή και περισσότερους κόμβους και έχει καλώς ορισμένα όρια. Οι κόμβοι στα όρια εισόδου του domain (ingress nodes) και στα όρια εξόδου (egress nodes) ονομάζονται και συνοριακοί κόμβοι (boundary nodes) και διαχωρίζονται από τους υπόλοιπους κόμβους στο εσωτερικό (core) του domain. Ο λόγος του διαχωρισμού αυτού έχει να κάνει με τη διαφορετική λειτουργία που επιτελούν οι δύο ομάδες, μέσω των μηχανισμών που εκτελούν.

Οι μηχανισμοί που εκτελούνται στα όρια ενός DS domain, εκτελούν την ταξινόμηση και τον ελέγχο της κίνησης. Η ταξινόμηση αντιστοιχίζει τα εισερχόμενα πακέτα σε μία από τις προκαθορισμένες κλάσεις εξυπηρέτησης του domain. Ο έλεγχος της κίνησης περιλαμβάνει, τη μέτρηση του προφίλ της κίνησης, το μαρκάρισμα της αν βρίσκεται εντός ή εκτός του προφίλ που έχει συμφωνηθεί, και τη διαμόρφωση ή απόρριψη της κίνησης, ώστε να διατηρείται μια επιθυμητή ταχύτητα στη ροή και να αντιμετωπίζονται περιπτώσεις συμφόρησης. Στο εσωτερικό του domain, κάθε κόμβος αναλαμβάνει την προώθηση της κίνησης, με την εφαρμογή της Per Hop Behavior (PHB) πολιτικής που αντιστοιχεί σε κάθε κλάση εξυπηρέτησης.

Στο κεφάλαιο που ακολουθεί γίνεται αναλυτική περιγραφή όλων των στοιχείων που συνθέτουν την αρχιτεκτονική DiffServ.



Σχήμα 3.1: Αναπαράσταση ενός DS domain.

3.2 Service Level Agreement

Η εφαρμογή που επιθυμεί κάποια ποιότητα υπηρεσίας, συνάπτει με το δίκτυο μια συμφωνία που ονομάζεται Service Level Agreement (SLA) και καθορίζει τις δεσμεύσεις του δικτύου προς την εφαρμογή και τις υποχρεώσεις της εφαρμογής προς το δίκτυο. Η εφαρμογή κοινοποιεί στο δίκτυο τις απαιτήσεις που έχει η κίνηση που παράγει. Έπειτα το δίκτυο, ανάλογα με τις δυνατότητες του τη δεδομένη στιγμή, προτείνει στην εφαρμογή την παροχή κάποιων εγγυήσεων. Μπορεί λόγω φόρτου στο δίκτυο ή πολύ υψηλών απαιτήσεων της εφαρμογής, οι εγγυήσεις να είναι υποδεέστερες των απαιτήσεων της εφαρμογής και να πρέπει να γίνει κάποιος συμβιβασμός. Αφού τελικά επιτευχθεί συμφωνία, γίνεται σύναψη της SLA που δεσμεύει και τις δύο πλευρές.

Η ποιότητα της υπηρεσίας που παρέχει το δίκτυο, μπορεί να μετρηθεί από μετρικές που παρουσιάστηκαν ήδη στο προηγούμενο κεφάλαιο και είναι οι ακόλουθες:

- *Χωρητικότητα (bandwidth)*
- *Καθυστέρηση (delay)*
- *Διακύμανση καθυστέρησης πακέτων (jitter)*
- *Απώλεια πακέτων (packet loss)*

Κάθε εφαρμογή προϋποθέτει κάποια όρια από το δίκτυο στις παραπάνω μετρικές για να λειτουργήσει απροβλημάτιστα. Έτσι αποτελούν μέρος της συμφωνίας με το δίκτυο.

Με τη σύναψη της SLA, πρέπει να πληρείτε επίσης μια σειρά προϋποθέσεων από το δίκτυο που συνοψίζονται παρακάτω:

- Σταθερή λειτουργία των δύο τελευταίων επιπέδων του OSI στο δίκτυο (φυσικό και ζεύξης δεδομένων).
- Bit Error Rate $\leq 10^{-12}$
- Αξιοπιστία του εξοπλισμού.
- Επιλογή του MTU (Maximum Transmission Unit) δηλαδή του μέγιστου μεγέθους πακέτου της ροής, για την αποφυγή κατακερματισμού των πακέτων.
- Λειτουργία του δικτύου με over provision χαρακτήρα για την αποφυγή συμφόρησης, που σημαίνει διατήρηση χαμηλότερου ρυθμού στην τροφοδότηση πακέτων στο δίκτυο από τον ρυθμό εξυπηρέτησης τους.

Η εκπλήρωση των παραπάνω προϋποθέσεων δίνει τη δυνατότητα στο δίκτυο να εφαρμόσει τους μηχανισμούς που διαθέτει και να προωθήσει την διερχόμενη κίνηση βάση της SLA που έχει συμφωνηθεί.

3.3 Per Hop Behaviors

Με τον όρο Per Hop Behavior (PHB), αναφερόμαστε στην πολιτική που εφαρμόζεται και στην προτεραιότητα που δίνεται στα πακέτα, κατά την προώθηση τους μέσα στο δίκτυο. Περιλαμβάνει την πολιτική χρονοδρομολόγησης, αστυνόμευσης και διαμόρφωσης της κίνησης καθώς και την πολιτική που εφαρμόζεται στις ουρές αναμονής των πακέτων. Κάθε PHB αποσκοπεί στην παροχή ποιότητας υπηρεσίας με συγκεκριμένα χαρακτηριστικά. Έτσι η υποστήριξη μιας PHB από το δίκτυο, απαιτεί την εφαρμογή κατάλληλης πολιτικής που έχει σαν αποτέλεσμα την επιθυμητή ποιότητα υπηρεσίας. Όπως έχει ήδη αναφερθεί, στα πακέτα γίνεται χρήση του πεδίου DSCP της IP επικεφαλίδας, για την κωδικοποίηση της κλάσης που αυτό ανήκει. Κάθε PHB που έχει προταθεί, συνοδεύεται και από τον αντίστοιχο κωδικό στο DSCP πεδίο. Οι PHB που έχουν προταθεί μέχρι σήμερα είναι οι εξής:

- ***Default PHB***: Είναι best-effort υπηρεσία. Με αυτήν εξυπηρετούνται οι ροές που δεν επιθυμούν κάποια ποιότητα υπηρεσίας, καθώς και οι ροές που δεν αντιστοιχούν σε καμία άλλη διαθέσιμη κλάση εξυπηρέτησης (το πεδίο DSCP των πακέτων τους δεν αντιστοιχίζεται σε κάποια από αυτές). Η προτεινόμενη DSCP τιμή είναι: 000000.

- ***Class-Selector PHBs***: Χρησιμοποιείται για τη διατήρηση συμβατότητας με το πεδίο precedence, όπως αυτό οριζόταν στο πεδίο TOS της επικεφαλίδας. Εδώ αντιστοιχίζονται οι τιμές της μορφής: xxx000, όπου οι συνδυασμοί των τριών πρώτων ψηφίων λαμβάνουν αντίστοιχη εξυπηρέτηση σε ένα DiffServ κόμβο, με αυτή που θα λάμβαναν και σε ένα κόμβο που βασίζεται στην IP-precedence ταξινόμηση και προώθηση της κίνησης. Με αυτό τον τρόπο διασφαλίζεται η συνύπαρξη των δύο συστημάτων.

- ***Expedited Forwarding PHB***: Είναι αντίστοιχη της Guaranteed υπηρεσίας που παρέχεται από την αρχιτεκτονική IntServ. Είναι premium υπηρεσία που παρέχει χαμηλή καθυστέρηση, χαμηλό jitter, μικρή απώλεια πακέτων και εγγυημένη χωρητικότητα. Εφαρμογές ευαίσθητες σε αυτές τις παραμέτρους (όπως έχουν αναφερθεί στην ενότητα: 1.2) εξυπηρετούνται στην κλάση αυτή. Σε περιπτώσεις συμφόρησης, προτείνεται η χρήση της από τις ποιο κρίσιμες εφαρμογές μόνο, καθώς είναι αδύνατη η παροχή premium υπηρεσίας σε μεγάλο όγκο της διερχόμενης κίνησης. Η προτεινόμενη DSCP τιμή είναι: 101110.

- ***Assured Forwarding PHB***: Είναι αντίστοιχη της Controlled load υπηρεσίας που παρέχεται από την αρχιτεκτονική IntServ. Ορίζονται τέσσερις AF κλάσεις, όπου σε κάθε μία υπάρχουν τρία επίπεδα απόρριψης της κίνησης. Έτσι δημιουργούνται συνολικά δώδεκα επίπεδα εξυπηρέτησης. Σε κάθε κλάση διατίθεται μέρος του διαθέσιμου bandwidth και του χώρου προσωρινής αποθήκευσης (buffer space). Τα επίπεδα απόρριψης κάθε κλάσης μπορούν να χρησιμοποιηθούν σε ροές που βγαίνουν εκτός του προφίλ της κίνησης που έχει συμφωνηθεί. Οι τιμές DSCP είναι της μορφής: xyzab0, όπου με xyz: 001/010/011/100 κωδικοποιούνται οι τέσσερις κλάσεις και με ab: 01/10/11 τα τρία επίπεδα απόρριψης σε κάθε κλάση.

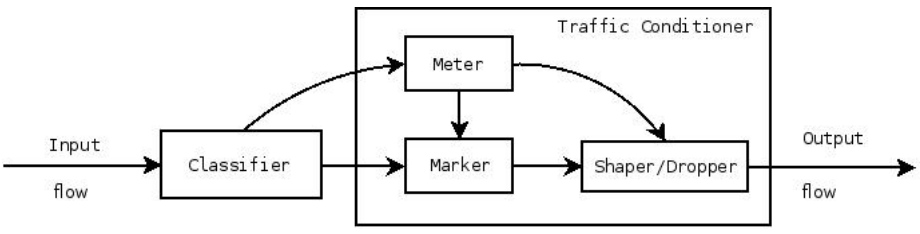
3.4 Μηχανισμοί DiffServ

Κάθε πακέτο που εισέρχεται σε ένα DS domain επεξεργάζεται από μια σειρά μηχανισμών που αναφέρονται συνοπτικά παρακάτω (Σχήμα: 3.2):

- **Μηχανισμός ταξινόμησης (Classifier)**
  Αντλεί πληροφορίες από τις επικεφαλίδες των πακέτων για την αναγνώριση και την ταξινόμηση στην PHB που τους αντιστοιχεί. Εμφανίζεται σε δύο εκδοχές: (I) *MF (Multi-Field)* που εφαρμόζεται στα όρια ενός DS domain και αναγνωρίζει τη ροή που ανήκει κάθε πακέτο, διαβάζοντας πολλαπλά πεδία των επικεφαλίδων ενός πακέτου. (II) *BA (Behavior-Aggregate)* που εφαρμόζεται στο εσωτερικό ενός DS domain και αναγνωρίζει τα πακέτα βάση του πεδίου DSCP (Differentiated Services Code Point) της IP επικεφαλίδας τους.

- **Μηχανισμός ελέγχου της κίνησης (Traffic conditioner)**
  Αποτελεί μια ομάδα μηχανισμών που αναφέρονται παρακάτω:

  - **Μηχανισμός μέτρησης (Meter)**
    Αναλαμβάνει τη μέτρηση του προφίλ της κίνησης και το επίπεδο συμμόρφωσής της, βάσει ενός επιθυμητού προφίλ. Στην απλούστερη περίπτωση τα πακέτα χαρακτηρίζονται ως εντός ή εκτός προφίλ. Τα αποτελέσματα της μέτρησης κοινοποιούνται στους μηχανισμούς που ακολουθούν για να οδηγήσουν τη συμπεριφορά τους.
  - **Μηχανισμός μαρκαρίσματος (Marker)**
    Μαρκάρει το πεδίο DSCP της IP επικεφαλίδας του πακέτου, με τον κωδικό της PHB που προέκυψε κατά την ταξινόμηση ή τη μέτρηση του προφίλ της κίνησης του πακέτου.
  - **Μηχανισμός μορφοποίησης/απόρριψης (Shaper/Dropper)**
    Ο μηχανισμός μορφοποίησης διασφαλίζει σταθερό ρυθμό στη ροή των πακέτων και εξομαλύνει ενδεχόμενες εκρήξεις (bursts) σε αυτή. Σε περίπτωση συμφόρησης ή για την αντιμετώπιση πακέτων εκτός προφίλ, μπορεί να χρησιμοποιηθεί ο μηχανισμός απόρριψης.



Σχήμα 3.2: Διάγραμμα ροής DiffServ μηχανισμών.

Ακολουθεί μια αναλυτική περιγραφή του τρόπου λειτουργίας των παραπάνω μηχανισμών καθώς και αλγόριθμων που χρησιμοποιούνται για την υλοποίηση τους.

3.5 Μηχανισμός ταξινόμησης

Η ταξινόμηση της κίνησης είναι ο πρώτος μηχανισμός που εφαρμόζεται κατά την άφιξη των πακέτων σε κάποιο κόμβο ενός DS domain. Ο μηχανισμός εφαρμόζεται σε κάθε εισερχόμενο πακέτο, έτσι η ταχύτητα εκτέλεσης του μηχανισμού συμβάλει στο συνολικό χρόνο επεξεργασίας του πακέτου. Ο χρόνος επεξεργασίας θα πρέπει να είναι μικρότερος του ρυθμού άφιξης των πακέτων, για να αποτρέπεται η δημιουργία συνωστισμού στην ουρά εισόδου του κόμβου.

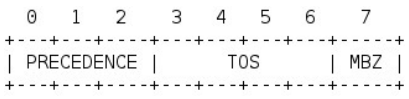
Στα όρια εισόδου ενός DS domain εφαρμόζεται η **Multi-Field (MF)** ταξινόμηση, όπου σε κάθε πακέτο εξετάζονται οι επικεφαλίδες του, για τα παρακάτω πεδία:

- *IP διεύθυνση αποστολέα*.
- *IP διεύθυνση παραλήπτη*.
- *Αριθμός πρωτοκόλλου μεταφοράς*.
- *Αριθμός port αποστολέα*.
- *Αριθμός port παραλήπτη*.
- *Κωδικός πεδίου DSCP*.

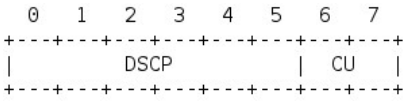
Οι παραπάνω πληροφορίες δίνουν τη δυνατότητα αναγνώρισης της ροής που ανήκει κάθε πακέτο και της ταξινόμησης του στην PHB που αντιστοιχεί.

Στην περίπτωση της **Behavior-Aggregate (BA)** ταξινόμησης, το μόνο πεδίο που ελέγχεται είναι ο κωδικός *DSCP* της επικεφαλίδας. Βάση του κωδικού αυτού γίνεται η ταξινόμηση του πακέτου στην PHB που αντιστοιχεί.

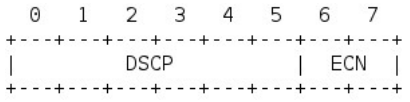
Το πεδίο *DSCP* υπάρχει και στις δύο εκδόσεις του IP πρωτοκόλλου (IPv4 και IPv6). Εμφανίστηκε στην IPv4 επικεφαλίδα, αντικαθιστώντας το πεδίο *Type of Service (TOS)* (Σχήμα: 3.3). Αρχικά, στα 8 bit του πεδίου *TOS*, δημιουργήθηκε το πεδίο DSCP μήκους 6 bit [7], αφήνοντας τα τελευταία 2 bit αχρησιμοποίητα (Σχήμα: 3.4). Αργότερα, ορίστηκε ο μηχανισμός *ECN* [15] που έκανε χρήση και των δύο αυτών bit (Σχήμα: 3.5). Στην IPv6 επικεφαλίδα, τα δύο πεδία αναφέρονται συνολικά ως: *Traffic Class*.



Σχήμα 3.3: Πεδίο TOS της IPv4 επικεφαλίδας.



Σχήμα 3.4: figure
Πεδίο DSCP της IPv4 επικεφαλίδας (χωρίς ECN).



Σχήμα 3.5: Πεδίο DSCP των IPv4 και IPv6 επικεφαλίδων.

3.6 Αστυνόμευση της κίνησης

Με τον όρο αυτό περιγράφουμε τις διαδικασίες διαχείρισης της κίνησης ανάλογα με τα αποτελέσματα που προκύπτουν από τη μέτρηση του προφίλ της. Οι διαδικασίες περιγράφονται στην SLA που έχει συμφωνηθεί. Η αστυνόμευση της κίνησης εφαρμόζεται συνήθως στα όρια ενός DS domain. Μπορεί να περιλαμβάνει το μαρκάρισμα των εκτός προφίλ πακέτων, σε κλάση εξυπηρέτησης χαμηλότερης προτεραιότητας, ή την απόρριψη τους. Η αστυνόμευση που εφαρμόζεται σε κάθε κλάση εξυπηρέτησης μπορεί επίσης να μεταβάλλεται κατά τη διάρκεια της ημέρας, του φόρτου στο δίκτυο ή και του είδους της κίνησης που εξυπηρετείται.

3.7 Αλγόριθμοι Token bucket και Leaky bucket

Είναι αλγόριθμοι που ελέγχουν τη ροή των πακέτων, για τη συμμόρφωση τους σε καθορισμένα όρια χωρητικότητας (bandwidth) και εκρηκτικότητας (burstiness). Η εκρηκτικότητα είναι το μέτρο της ανομοιομορφίας ή της διακύμανσης που παρουσιάζει μια ροή.

Ο αλγόριθμος **Token bucket** βασίζεται σε ένα χώρο αποθήκευσης μέγιστης χωρητικότητας **b** (κάδος) που δέχεται πακέτα με σταθερό ρυθμό **r**. Για κάθε πακέτο που ελέγχεται, υπολογίζεται το μήκος του πακέτου σε bytes και εάν υπάρχει διαθέσιμος χώρος (tokens) στον κάδο. Εάν υπάρχει, το πακέτο θεωρείται εντός προφίλ. Έτσι αφαιρείται το μήκος του πακέτου από τη διαθέσιμη χωρητικότητα του κάδου και προωθείται με την πολιτική που του αντιστοιχεί. Αλλιώς το πακέτο θεωρείται εκτός προφίλ. Η πολιτική που εφαρμόζεται στα πακέτα εκτός προφίλ εξαρτάται από την SLA που έχει συμφωνηθεί (μπορεί να απορριφθεί ή να αυξηθεί η πιθανότητα απόρριψης του σε περίπτωση μελλοντικής συμφόρησης). Ο κάδος αναγνωρίζει ως εντός προφίλ, ροές που έχουν μέσο ρυθμό μετάδοσης r και παρουσιάζουν μέγιστες εκρήξεις ίσες με το μέγεθος b του κάδου.

Ο αλγόριθμος **Leaky bucket** ακολουθεί παρόμοια λογική. Η λειτουργία του είναι ανάλογη ενός τρύπιου κουβά που γεμίζει με νερό. Στον κουβά εισέρχεται νερό με αυθαίρετο ρυθμό και εξέρχεται με σταθερό ρυθμό από την τρύπα που έχει. Λόγω της πεπερασμένης του χωρητικότητας, στην περίπτωση που γεμίσει, το νερό που περισσεύει θα ξεχειλίζει από τον κουβά (θα απορρίπτεται). Έτσι για την υλοποίηση του, μπορεί να χρησιμοποιηθεί μια ουρά (queue) σταθερής χωρητικότητας. Τα εισερχόμενα πακέτα αποθηκεύονται προσωρινά σε αυτήν όσο ο ρυθμός άφιξης είναι μεγαλύτερος του ρυθμού εξυπηρέτησης και η ουρά γεμίζει. Όταν γεμίσει, τα πακέτα που ακολουθούν απορρίπτονται μέχρι να βρεθεί χώρος ξανά. Ο αλγόριθμος επιτυγχάνει έτσι ένα σταθερό ρυθμό εξόδου της κίνησης, εξαλείφοντας τυχόν εκρήξεις σε αυτήν.

Οι αλγόριθμοι μπορούν να χρησιμοποιηθούν στη μορφοποίηση ή την αστυνόμευση της κίνησης. Είναι δυνατή η διαδοχική εφαρμογή των παραπάνω αλγόριθμων σε επίπεδα, δηλαδή με χρήση διαφορετικών τιμών στις παραμέτρους τους. Έτσι προκύπτουν διαφορετικές κλάσεις εξυπηρέτησης σε κάθε επίπεδο με δυνατότητα χωριστής διαχείρισης τους.

3.8 Active Queue Management

Ο όρος Active Queue Management (AQM) αναφέρεται στις τεχνικές διαχείρισης των ουρών αναμονής ενός δρομολογητή, για την αποφυγή της συμφόρησης σε αυτές. Περιλαμβάνει την προληπτική απόρριψη πακέτων ή και τη χρήση του μηχανισμού ECN (Explicit Congestion Notification) για την ειδοποίηση του αποστολέα. Η διατήρηση πολλαπλών ουρών αναμονής σε ένα δρομολογητή αποσκοπεί στην εξυπηρέτηση κάθε κλάσης στη δική της ουρά. Έτσι η ποιότητα υπηρεσίας που προσφέρει κάθε κλάση, ανάγεται στην κατάλληλη διαχείριση της ουρά αναμονής που την εξυπηρετεί.

Οι ουρές αναμονής που χρησιμοποιεί ένας δρομολογητής είναι συνήθως τύπου drop-tail. Δηλαδή δέχονται πακέτα όσο έχουν διαθέσιμο χώρο, διαφορετικά τα απορρίπτουν. Η ενεργή διαχείριση αποσκοπεί στην καλή τους απόδοση, αποφεύγοντας τη συμφόρηση τους. Κάτι τέτοιο επιτυγχάνεται συνήθως διατηρώντας ένα ή περισσότερα επίπεδα απόρριψης, που αντιστοιχεί διαφορετική πιθανότητα απόρριψης στο κάθε ένα. Έτσι ο μηχανισμός απορρίπτει (ή μαρκάρει μέσω του ECN) πακέτα ακόμα και όταν η ουρά δεν έχει γεμίσει.

Οι ουρές τύπου drop-tail, παρουσιάζουν προβλήματα στη διαχείριση εκρήξεων από ροές και ευνοούν την εμφάνιση του φαινομένου που ονομάζεται TCP καθολικός συντονισμός (TCP global synchronization). Το φαινόμενο αυτό οφείλεται στον αλγόριθμο διαχείρισης της απόρριψης πακέτων από το πρωτόκολλο TCP που ονομάζεται slow-start. Στην περίπτωση απόρριψης πακέτων το πρωτόκολλο θεωρεί ότι έχει συμβεί συμφόρηση κάπου στο δίκτυο. Έτσι μειώνει το ρυθμό αποστολής του για κάποιο χρονικό διάστημα και σταδιακά τον επαναφέρει προσπαθώντας να εντοπίσει εάν η συμφόρηση έχει παρέλθει. Μια συγχρονισμένη απόρριψη πακέτων από διαφορετικές TCP ροές, θα προκαλέσει τη συγχρονισμένη εφαρμογή του αλγορίθμου από αυτές, κάτι που θα οδηγήσει στην επανάληψη της συμφόρησης μετά από κάποιο χρονικό διάστημα και την περιοδική επανάληψη του φαινομένου.

Οι μηχανισμοί AQM αντιμετωπίζουν τις παραπάνω αδυναμίες παρέχοντας επίσης καλή απόδοση στις ουρές αναμονής με τη διατήρηση λίγων πακέτων σε αυτές, συμβάλλοντας στην παροχή χαμηλής καθυστέρησης και jitter από το δίκτυο. Ο αντίλογος στη χρήση AQM μηχανισμών που απορρίπτουν απευθείας πακέτα (χωρίς να κάνουν χρήση του μηχανισμού ECN) είναι ότι έτσι δεν γίνεται η καλύτερη δυνατή διαχείριση των πόρων του δικτύου, καθώς μπορεί να απορρίπτονται πακέτα που πληρούν τις προδιαγραφές, ενώ υπάρχει ακόμα διαθέσιμος χώρος για την εξυπηρέτησή τους.

Υπάρχει πληθώρα μηχανισμών και αλγόριθμων στη βιβλιογραφία που εφαρμόζονται σαν τεχνικές AQM και η έρευνα προς αυτή την κατεύθυνση συνεχίζεται. Παρακάτω παρουσιάζονται κάποια χαρακτηριστικά παραδείγματα.

3.8.1 Explicit Congestion Notification (ECN)

Ο μηχανισμός αναλαμβάνει την ειδοποίηση του αποστολέα για πιθανή συμφόρηση του δικτύου που τον εξυπηρετεί. Ο αποστολέας έτσι μπορεί να αντιληφθεί την επερχόμενη συμφόρηση πριν συμβεί κάποια απώλεια πακέτων και να αντιδράσει, μειώνοντας το ρυθμό αποστολής του, μέχρις ότου αυτή παρέλθει. Όπως αναφέρθηκε και προηγουμένως, η λειτουργία του βασίζεται στη χρήση των δύο bit του πεδίου ECN, που ορίστηκε στην IP επικεφαλίδα. Σε αυτά κωδικοποιούνται τέσσερις καταστάσεις που είναι οι εξής:

- 00: *Non-ECT*
- 01: *ECT(0)*
- 10: *ECT(1)*
- 11: *CE*

Η κατάσταση **Non-ECT** κωδικοποιεί τη μη υποστήριξη του μηχανισμού στα δύο άκρα επικοινωνίας για να μην γίνει χρήση του. Οι καταστάσεις **ECT{0,1}** κωδικοποιούν την υποστήριξη του μηχανισμού από τα δύο άκρα της επικοινωνίας. Τέλος η κατάσταση **CE** κωδικοποιεί την ύπαρξη συμφόρησης.

Σε περίπτωση χρήσης του ECN, αν σε μια ουρά υπό τον έλεγχο AQM υπάρχει ενδεχόμενο συμφόρησης, ο AQM της ουράς μπορεί να ειδοποιήσει τον παραλήπτη για τη συμφόρηση αυτή, μαρκάροντας στα διερχόμενα πακέτα το πεδίο ECN ως CE. Με τη σειρά του ο παραλήπτης μπορεί να ειδοποιήσει (echo) για την επικείμενη συμφόρηση τον αποστολέα και αυτός τελικά να μειώσει το ρυθμό μετάδοσης, αποτρέποντας έτσι τη συμφόρηση χωρίς να απορριφθούν πακέτα. Η ειδοποίηση για συμφόρηση γίνεται στο επίπεδο μεταφοράς του παραλήπτη με τη χρήση κάποιου πρωτοκόλλου μεταφοράς του διαδικτύου, όπως το TCP. Στην περίπτωση χρήσης του UDP πρωτοκόλλου θεωρητικά είναι δυνατή η ειδοποίηση για συμφόρηση, αλλά μόνο στο επίπεδο της εφαρμογής, καθώς το πρωτόκολλο δεν υποστηρίζει την αποστολή πακέτων ελέγχου από τον παραλήπτη προς τον αποστολέα (όπως το πρωτόκολλο TCP). Πρακτικά όμως αυτό ακόμη δεν γίνεται, γιατί δεν δίνεται πρόσβαση στα ECN bit από τις υπάρχουσες βιβλιοθήκες.

3.8.2 Random early detection (RED)

Ο μηχανισμός προλαμβάνει τη συμφόρηση μιας ουράς, παρακολουθώντας το μέσο όρο των πακέτων που βρίσκονται στην αναμονή. Σε περίπτωση ενδεχόμενης συμφόρησης, απορρίπτει πακέτα ή ειδοποιεί μέσω του μηχανισμού ECN γι αυτήν. Ο μηχανισμός βασίζεται στη χρήση πιθανοτήτων για το ενδεχόμενο απόρριψης. Όσο η πληρότητα της ουράς παραμένει σε χαμηλά επίπεδα τα εισερχόμενα πακέτα γίνονται δεκτά. Με την αύξηση των πακέτων στην αναμονή, αυξάνει και η πιθανότητα απόρριψης τους μέχρι τη μέγιστη δυνατή τιμή 1. Ο μηχανισμός αποδέχεται εκρήξεις στην κίνηση, όσο αυτές δεν χρησιμοποιούν μεγάλο μέρος της διαθέσιμης χωρητικότητας και αποτρέπει την εμφάνιση του φαινομένου του καθολικού συντονισμού που αναφέρθηκε προηγουμένως.

Η λειτουργία του μηχανισμού βασίζεται στην παρακολούθηση της μέσης πληρότητας της ουράς και την πιθανότητα απόρριψης των εισερχόμενων πακέτων. Για το σκοπό αυτό διατηρεί τρεις μεταβλητές:

- *min_threshold*
- *max_threshold*
- *max_possibility*

Η μέση πληρότητα που παρουσιάζει η ουρά, συγκρίνεται με τα όρια που ορίζουν οι παραπάνω παράμετροι και προκύπτει η πιθανότητα απόρριψης των πακέτων. Ο αλγόριθμος είναι ο εξής:

- Όσο η μέση πληρότητα της ουράς βρίσκεται κάτω από το min_threshold, τα πακέτα γίνονται δεκτά.
- Όσο η μέση πληρότητα της ουράς βρίσκεται μεταξύ min_threshold και max_threshold, η πιθανότητα απόρριψης αυξάνει γραμμικά στο διάστημα $[0, max\_possibility]$.
- Όσο η μέση πληρότητα της ουράς ξεπερνάει το max-threshold, κάθε πακέτο απορρίπτεται.

Αν και ο μηχανισμός μπορεί να προλαμβάνει τη συμφόρηση, η καλή απόδοσή του εξασφαλίζεται με προσεκτική ρύθμιση των παραμέτρων του. Επίσης δεν προβλέπει διαφοροποιημένη διαχείριση σε κλάσεις εξυπηρέτησης, όπως ορίζονται στην αρχιτεκτονική DiffServ. Για την αντιμετώπιση των παραπάνω αδυναμιών έχουν προταθεί αρκετές παραλλαγές του μηχανισμού. Ακολουθεί μία από αυτές.

3.8.3 Weighted random early detection (WRED)

Ο μηχανισμός αποτελεί παραλλαγή του μηχανισμού RED και υποστηρίζει πολλαπλά επίπεδα απόρριψης, δίνοντας τη δυνατότητα διαχείρισης πολλαπλών κλάσεων εξυπηρέτησης με διαφοροποιημένο τρόπο. Για τη λειτουργία του, χρησιμοποιεί σε κάθε κλάση διαφορετικά όρια (min_threshold, max_threshold, max_possibility), κάτι που του επιτρέπει να ελέγχει την απόρριψη των πακέτων ανάλογα με τα όρια της κλάσης που ανήκουν. Τα όρια μπορούν να ρυθμιστούν συναρτήσει της προτεραιότητας που λαμβάνει κάθε κλάση εξυπηρέτησης, δηλαδή να είναι είναι ποιο αυστηρά όσο η προτεραιότητα της μικραίνει. Ο αλγόριθμος διαχείρισης κάθε κλάσης που προκύπτει ακολουθεί τη λογική του RED. Με τη χρήση του WRED, η κίνηση υψηλής προτεραιότητας έχει χαμηλότερη πιθανότητα απόρριψης σε σχέση με την υπόλοιπη, έτσι έχει τη δυνατότητα μέσα από μία μοναδική ουρά εξυπηρέτησης, να παρέχει την επιθυμητή ποιότητα υπηρεσίας που ορίζει η κλάση.

3.9 Χρονοδρομολόγηση

Η διατήρηση πολλαπλών ουρών αναμονής από ένα δρομολογητή που υποστηρίζει την αρχιτεκτονική DiffServ, δημιουργεί την ανάγκη διαχείρισης της σειράς εξόδου των πακέτων από αυτές. Καθώς η μετάδοση των πακέτων γίνεται σειριακά μέσα από ένα κανάλι επικοινωνίας, στην περίπτωση ύπαρξης πολλαπλών ουρών αναμονής, πρέπει να εφαρμόζεται ένας αλγόριθμος. Αυτός αποφασίζει για τη σειρά εξυπηρέτησης των διαθέσιμων ουρών και το χρόνο που διατίθεται (ή την ποσότητα δεδομένων που μεταδίδονται) για την εξυπηρέτηση κάθε ουράς. Ο τρόπος διαχείρισης αυτών των παραμέτρων από τον αλγόριθμο χρονοδρομολόγησης, συμβάλει τελικά στην ποιότητα υπηρεσίας που παρέχεται από το δίκτυο σε κάθε κλάση εξυπηρέτησης. Παρακάτω, παρουσιάζονται μερικοί από τους αλγόριθμους χρονοδρομολόγησης που έχουν προταθεί στη βιβλιογραφία.

3.9.1 First In First Out (FIFO)

Είναι ένας διαδεδομένος αλγόριθμος χρονοδρομολόγησης με πολύ απλή υλοποίηση. Ο αλγόριθμος εξυπηρετεί τα δεδομένα βάση της σειράς άφιξης τους. Οι ουρές (queues) κάνουν χρήση του αλγόριθμου, καθώς τα δεδομένα που τις διασχίζουν εισέρχονται και εξέρχονται με τον τρόπο αυτό. Ο χρόνος εξυπηρέτησης αυξάνει γραμμικά με το πλήθος των στοιχείων που περιμένουν στην ουρά, έτσι δεν είναι αποδοτικός σε περιπτώσεις μεγάλης συμφόρησης. Επίσης αδυνατεί να διαχειριστεί πολλαπλές ουρές ή να αλλάξει την προτεραιότητα των δεδομένων που διαχειρίζεται, έτσι δεν μπορεί να χρησιμοποιηθεί αποκλειστικά. Λόγω πάντως της χρήσης του από τις ουρές, χρησιμοποιείται κατά κόρον στα δίκτυα και συνήθως σε συνδυασμό με άλλους αλγόριθμους.

3.9.2 Priority queuing (PQ)

Η λειτουργία του αλγόριθμου βασίζεται στην προτεραιότητα που έχει κάθε στοιχείο που διαχειρίζεται. Η λειτουργία του είναι αντίστοιχη της ουράς, με τη διαφορά της ύπαρξης των προτεραιοτήτων στα εισερχόμενα στοιχεία και τη δυνατότητα εξυπηρέτησης τους, βάση αυτών. Προηγείται πάντα το στοιχείο με τη μεγαλύτερη προτεραιότητα, ενώ στην περίπτωση ίδιας προτεραιότητας, προηγείται το στοιχείο που εισήλθε πρώτο στην ουρά (δηλαδή χρησιμοποιείται ο αλγόριθμος FIFO). Στους δρομολογητές μπορεί να εφαρμοστεί με την αντιστοίχηση προτεραιοτήτων στις ουρές που διαχειρίζεται. Κατά τη λειτουργία του, ο αλγόριθμος ελέγχει με τη σειρά όλες τις ουρές που διαχειρίζεται (ξεκινώντας από την ουρά υψηλότερης προτεραιότητας) και μεταδίδει το πρώτο πακέτο που είναι διαθέσιμο. Έπειτα ξαναρχίζει από την αρχή τον έλεγχο. Η ουρά υψηλής προτεραιότητας λαμβάνει προνομιακή μεταχείριση και έτσι επιτυγχάνεται χαμηλή καθυστέρηση στην κίνηση της, κάτι όμως που μπορεί να οδηγήσει και σε λιμοκτονία (starvation) τις υπόλοιπες, εάν η κίνηση στην ουρά υψηλής προτεραιότητας είναι μεγάλη.

3.9.3 Fair queuing (FQ)

Είναι αλγόριθμος που στοχεύει στη δίκαιη (fair) εξυπηρέτηση των στοιχείων που διαχειρίζεται. Στην περίπτωση των δικτύων μεταγωγής πακέτων, επιτρέπει σε ένα δρομολογητή με πολλαπλές ουρές αναμονής πακέτων, να μοιράζει δίκαια τη διάθεση του καναλιού μετάδοσης. Η λειτουργία του βασίζεται στην παροχή ενός μέσου ρυθμού μετάδοσης σε κάθε ουρά που διαχειρίζεται και υπολογίζεται από το λόγο: $\frac{R}{N}$ (όπου R: ο ρυθμός μετάδοσης του καναλιού και N: ο αριθμός των ουρών στην αναμονή). Με τον τρόπο αυτό κάθε ουρά μεταδίδει (κατά μέσο όρο) την ίδια ποσότητα δεδομένων αποφεύγοντας το φαινόμενο της λιμοκτονίας. Από την άλλη, η ομοιόμορφη διαχείριση αποτρέπει την παροχή κάποιας προτεραιότητας όταν είναι επιθυμητή. Ο αλγόριθμος είναι ειδική περίπτωση του αλγόριθμου Weighted fair queuing (WFQ), ο οποίος αποδίδει κάποιο βάρος σε κάθε ουρά. Έτσι στην περίπτωση του FQ αποδίδεται το ίδιο βάρος σε κάθε ουρά.

3.9.4 Weighted fair queuing (WFQ)

Όπως αναφέρθηκε προηγουμένως, ο αλγόριθμος αποδίδει κάποιο βάρος σε κάθε ουρά που διαχειρίζεται. Έστω N ο αριθμός των ουρών που έχουν πακέτα στην αναμονή, R ο ρυθμός μετάδοσης του καναλιού που τα εξυπηρετεί και σε κάθε ουρά αντιστοιχεί ένα βάρος w_i με $i \in [1, N]$. Ο μέσος ρυθμός μετάδοσης κάθε ουράς υπολογίζεται από το λόγο: $\frac{R*w_i}{(w_1+w_2+...+w_N)}$. Η χρήση του αλγόριθμου (WFQ) μαζί με την εφαρμογή του αλγόριθμου Leaky bucket στην εισερχόμενη κίνηση, εγγυάται την ύπαρξη ενός άνω ορίου στη μέγιστη συνολική καθυστέρηση, κάτι που χρησιμεύει στην παροχή ποιότητας υπηρεσίας. Τα βάρη μπορεί να ρυθμίζονται και δυναμικά.

Μπορούμε για παράδειγμα να αναθέτουμε το βάρος: $w_i = \frac{1}{c_i}$, όπου c_i το κόστος για τη μετάδοση ενός bit της ροής $i \in [1, N]$. Το αποτέλεσμα της τεχνικής αυτής ονομάζεται **proportional fairness** και βρίσκει εφαρμογή στα ασύρματα δίκτυα, καθώς μπορεί να μεγιστοποιήσει τη συνολική διαμεταγωγή τους, παρέχοντας ταυτόχρονα και μια εγγυημένη ελάχιστη υπηρεσία.

3.9.5 Round Robin (RR)

Ο αλγόριθμος είναι απλός στην υλοποίηση του. Βασίζεται στην ισοδύναμη κυκλική εξυπηρέτηση των πόρων που διαχειρίζεται, αποφεύγοντας το φαινόμενο της λιμοκτονίας. Χρησιμοποιεί μια παράμετρο που καθορίζει το διάστημα που αφιερώνει ο αλγόριθμος στην εξυπηρέτηση κάθε πόρου. Βρίσκει εφαρμογή σε πολλές περιπτώσεις χρονοδρομολόγησης, μαζί και στα δίκτυα μεταγωγής πακέτων. Όταν διαχειρίζεται πολλαπλές ουρές σε ένα δρομολογητή, καθώς δεν προσφέρει κάποια προτεραιότητα στη διαχείριση τους, μπορεί να παρέχει best-effort υπηρεσία. Έτσι στην περίπτωση της DiffServ αρχιτεκτονικής μπορεί να χρησιμοποιηθεί κάποια από τις παραλλαγές του αλγόριθμου, που περιγράφονται παρακάτω.

3.9.6 Weighted round robin (WRR)

Στην παραλλαγή αυτή του round robin, ο αριθμός **n** των πακέτων που εξυπηρετείται σε κάθε ουρά, υπολογίζεται από τον τύπο: $n = normalized(\frac{Weight}{MeanPacketSize})$. Για να λειτουργήσει ο αλγόριθμος, χρειάζεται να γνωρίζει το μέσο μέγεθος των πακέτων που εξυπηρετεί, κάτι που δεν μπορεί να υπολογιστεί πάντα με ακρίβεια (όπως στην περίπτωση που η ουρά εξυπηρετεί μεταβλητού μήκους πακέτα). Αυτό έχει σαν αποτέλεσμα τη μειωμένη απόδοση του αλγόριθμου. Για την αντιμετώπιση του προβλήματος αυτού, έχει προταθεί η παραλλαγή που ακολουθεί.

3.9.7 Deficit (weighted) round robin (DWRR)

Η λειτουργία του αλγόριθμου βασίζεται στη διατήρηση ενός μετρητή σε κάθε ουρά που εξυπηρετείται, που ονομάζεται deficit counter. Κάθε ουρά που διαθέτει πακέτα για μετάδοση, εξυπηρετείται μόνο όσο ο deficit counter που διαθέτει είναι θετικός. Η τιμή του μετρητή εκφράζεται σε bytes. Αυξάνεται κατά μία ποσότητα που ονομάζεται quantum και μειώνεται από το μέγεθος σε byte κάθε πακέτου που εξυπηρετεί. Η τιμή του quantum εκφράζει το μέσο όρο των bytes που εξυπηρετεί κάθε ουρά σε ένα πλήρη κύκλο του αλγόριθμου. Καθώς ο deficit counter παρακολουθεί τον αριθμό των bytes που μεταδίδει κάθε ουρά, συμβάλει στη διατήρηση ενός μέσου όρου στον αριθμό bytes που εξυπηρετεί κάθε ουρά, όπως ορίζεται από την τιμή quantum. Ο αλγόριθμος δεν χρησιμοποιεί το μέσο μέγεθος των πακέτων που εξυπηρετεί για τη λειτουργία του. Αποφεύγοντας έτσι πιθανά προβλήματα απόδοσης, όπως στην περίπτωση του WRR.

Ο αλγόριθμος λειτουργεί ως εξής:

- Ο deficit counter κάθε ουράς, αρχικά είναι 0.
- Οι ουρές εξετάζονται κυκλικά. Όταν βρεθεί ουρά που έχει πακέτα στην αναμονή, προστίθεται στον deficit counter η τιμή quantum.
- Ελέγχεται το μέγεθος του πακέτου που βρίσκεται στην κορυφή:
  - Αν είναι μικρότερο του deficit counter, το πακέτο εξυπηρετείται και αφαιρείται το μέγεθος του από τον deficit counter.
  - Αλλιώς, η ουρά χάνει τη σειρά της και προστίθεται στον deficit counter η τιμή quantum.
- Όταν ο deficit counter μιας ουράς γίνει μηδέν ή αρνητικός, προστίθεται σε αυτόν η τιμή quantum.

Ακολουθεί μια εξελιγμένη παραλλαγή του αλγόριθμου, που προσφέρει πολλές επιλογές παραμετροποίησης της συμπεριφοράς του.

3.9.8 Modified deficit round robin (MDRR)

Ο αλγόριθμος round robin καθώς και οι παραλλαγές που παρουσιάστηκαν προηγουμένως, στοχεύουν στην δίκαιη μεταχείριση των ουρών αναμονής που διαχειρίζονται. Στα πλαίσια της αρχιτεκτονικής DiffServ, οι κλάσεις εξυπηρέτησης που ορίζονται, έχουν στόχο την παροχή διαφορετικής ποιότητας υπηρεσίας η κάθε μία. Η εξυπηρέτηση κάθε κλάσης σε διαφορετική ουρά αναμονής, επιφέρει την ανάγκη χρήσης ενός αλγόριθμου που μπορεί να διαχειριστεί με διαφορετική προτεραιότητα τις ουρές. Ο αλγόριθμος MDRR είναι παραλλαγή του deficit round robin, που προσφέρει διαφορετική προτεραιότητα σε κάθε ουρά που διαχειρίζεται. Υποστηρίζει επίσης δύο διαφορετικούς τρόπους λειτουργίας, που αλλάζουν τη συμπεριφορά του στη σειρά διαχείρισης των ουρών. Παρακάτω παρουσιάζεται αναλυτικά ο τρόπος λειτουργίας του αλγόριθμου, καθώς και ένα παράδειγμα χρήσης του για να γίνει ευκολότερα κατανοητός ο τρόπος λειτουργίας του.

Όπως και στην περίπτωση του deficit round robin, ο αλγόριθμος διατηρεί σε κάθε ουρά που διαχειρίζεται δύο μεταβλητές, την τιμή quantum και τον deficit counter. Ο ρόλος των μεταβλητών παραμένει ο ίδιος. Η τιμή quantum εκφράζει το μέσο αριθμό των bytes που μεταδίδει κάθε ουρά σε ένα πλήρη κύκλο. Ο deficit counter μετρά τον αριθμό των bytes που μετέδωσε κάθε ουρά. Επιπλέον μπορεί να οριστεί σε κάθε ουρά μια τιμή βάρους που επηρεάζει τον υπολογισμό της τιμή quantum.

Η εξυπηρέτηση κάθε ουράς ακολουθεί τη λογική του deficit round robin. Ο deficit counter κάθε ουράς είναι αρχικά 0. Όταν έρθει η σειρά της ουράς να μεταδώσει, αυξάνεται ο deficit counter με την τιμή quantum που της αντιστοιχεί. Η ουρά μεταδίδει όσο ο deficit counter παραμένει θετικός, μειώνοντας τον κάθε φορά με το μέγεθος του πακέτου (σε bytes) που μετέδωσε. Όταν ο deficit counter μηδενιστεί ή γίνει αρνητικός, προστίθεται σε αυτόν η τιμή quantum και αρχίζει η εξυπηρέτηση της επόμενης ουράς που έχει σειρά. Τέλος, αν η ουρά θέλει να στείλει πακέτο μεγαλύτερο του deficit counter, χάνει τη σειρά της και προσθέτει σε αυτόν την τιμή quantum που της αντιστοιχεί.

Ο καθιερωμένος τρόπος υπολογισμού (αν και μπορεί να διαφέρει) βασίζεται στον τύπο: $D = MTU + (weight - 1) * 512$, όπου:

- *D*: Η τιμή quantum που αντιστοιχεί στην ουρά.
- *MTU*: Η μέγιστη μονάδα μετάδοσης του δικτύου (Maximum Transmission Unit). Ουσιαστικά είναι το μέγιστο μέγεθος πακέτου που υποστηρίζει το δίκτυο.
- *weight*: Το βάρος της ουράς.

Όπως φαίνεται από τον τύπο υπολογισμού, για βάρος ίσο με 1, η τιμή quantum ισούται με την τιμή του MTU. Σε κάθε ακέραια αύξηση του βάρους, η τιμή quantum αυξάνεται κατά 512 bytes. Κατά τον υπολογισμό του quantum το αποτέλεσμα στρογγυλοποιείται στο ακέραιο μέρος με αποκοπή, καθώς εκφράζει αριθμό bytes. Γενικά προτείνεται η ρύθμιση της ουράς χαμηλότερης προτεραιότητας με βάρος 1 και τη σταδιακή του αύξηση στις υπόλοιπες ουρές. Προτείνεται επίσης η χρήση χαμηλών τιμών στα βάρη συνολικά, για τη διατήρηση σε χαμηλά επίπεδα της καθυστέρησης και του jitter συνολικά. Το jitter επηρεάζεται όσο αυξάνει η τιμή του βάρους. Έτσι στην περίπτωση ουρών ίδιας προτεραιότητας (δηλαδή ανάθεσης ίδιας τιμής στο βάρος), προτείνεται η χρήση των χαμηλότερων δυνατών τιμών.

Όπως αναφέρθηκε, ο αλγόριθμος υποστηρίζει δύο τρόπους λειτουργίας στη σειρά εξυπηρέτησης των ουρών, που είναι οι: **Alternate** και **Strict**. Η διαφορά τους είναι στον τρόπο που εξυπηρετούν την ουρά υψηλής προτεραιότητας σε σχέση με τις υπόλοιπες. Λαμβάνοντας υπόψιν ότι αναφερόμαστε στην εξυπηρέτηση μιας ουράς μόνο όταν αυτή έχει δεδομένα σε αναμονή, λειτουργούν ως εξής:

- Σε **Alternate** mode: Η ουρά υψηλής προτεραιότητας εξυπηρετείται εκ περιτροπής με τις υπόλοιπες. Αν θεωρήσουμε ότι Q_P είναι η ουρά υψηλής προτεραιότητας και Q_1, Q_2 δύο ακόμη ουρές. Ο αλγόριθμος θα εξυπηρετήσει την Q_P πρώτα, έπειτα την Q_1, μετά την Q_P ξανά, έπειτα την Q_2, κ.ο.κ. Εφαρμόζουμε δηλαδή τον αλγόριθμο round robin σε δύο επίπεδα, μεταξύ της Q_P και της ουράς που έχει σειρά από τον round robin μεταξύ των Q_1 και Q_2.
- Σε **Strict** mode: Η ουρά υψηλής προτεραιότητας εξυπηρετείται όσο περιέχει δεδομένα προς μετάδοση. Αλλιώς εξυπηρετούνται οι υπόλοιπες με τον αλγόριθμο round robin.

Συγκρίνοντας τις δύο προσεγγίσεις, με τη χρήση του Alternate mode γίνεται ποιο δύσκολη η πρόβλεψη της καθυστέρησης και του jitter. Οι τιμές τους γενικά αυξάνονται στην ουρά υψηλής προτεραιότητας, σε σχέση με τη χρήση του Strict mode. Αυτό οφείλεται στην εκ περιτροπής εξυπηρέτηση της σε Alternate mode, σε σχέση με την αποκλειστική εξυπηρέτηση της σε Strict. Όταν εξυπηρετείται κάποια από τις υπόλοιπες ουρές, η ουρά υψηλής προτεραιότητας θα πρέπει να περιμένει τη σειρά της για να μπορέσει να μεταδώσει. Από την άλλη, η χρήση του Alternate mode δίνει στην ουρά υψηλής προτεραιότητας περισσότερες ευκαιρίες να μεταδώσει σε κάθε κύκλο του αλγόριθμου. Αυτό έχει σαν αποτέλεσμα να χρησιμοποιεί περισσότερη από τη διαθέσιμη χωρητικότητα του δικτύου σε σχέση με τις άλλες ουρές, που είναι ανάλογη του αριθμού των ουρών που εξυπηρετεί ο αλγόριθμος. Τέλος με τη χρήση του Alternate mode αποτρέπεται το φαινόμενο της λιμοκτονίας που μπορεί να εμφανιστεί με τη χρήση του Strict (σε περίπτωση μεγάλης κίνησης στην ουρά υψηλής προτεραιότητας).

Ακολουθεί ένα παράδειγμα των βημάτων που ακολουθεί ο MDRR σε Alternate mode:
Έστω ότι ένας δρομολογητής χρησιμοποιεί τις ουρές: Q_P, Q_1, Q_2, για την εξυπηρέτηση της διερχόμενης κίνησης ενός δικτύου με MTU = 1500 bytes. Τα βάρη που έχουν ανατεθεί σε κάθε ουρά είναι:

- $Q_P : W_{Q_P} = 3$
- $Q_1 : W_{Q_1} = 2$
- $Q_2 : W_{Q_2} = 1$

Το quantum που αντιστοιχεί σε κάθε ουρά, κάνοντας χρήση του καθιερομένου τύπου, είναι:

- $Q_P : D_{Q_P} = 1500 + (W_{PQ} - 1) * 512 = 2524$ bytes
- $Q_1 : D_{Q_1} = 1500 + (W_{PQ} - 1) * 512 = 2012$ bytes
- $Q_2 : D_{Q_2} = 1500 + (W_{PQ} - 1) * 512 = 1500$ bytes

Σε κάθε ουρά βρίσκονται σε αναμονή πακέτα διαφορετικού μεγέθους. Κάνοντας χρήση του συμβολισμού x/y, για την περιγραφή x πακέτων μεγέθους y και θεωρώντας την κορυφή της ουράς αριστερά, οι ουρές παρουσιάζουν αρχικά την εξής δομή:

- $Q_P : [6/300, 2/1500, 4/300]$
- $Q_1 : [2/1500, 4/300, 1/1500]$
- $Q_2 : [5/300, 1/1500, 5/300]$

Ο deficit counter κάθε ουράς, έχει την τιμή 0 αρχικά: $DC_{Q_P} = DC_{Q_1} = DC_{Q_2} = 0$.

Εφαρμόζοντας τον αλγόριθμο MDRR σε Alternate mode για την εξυπηρέτηση των ουρών, προκύπτουν τα παρακάτω βήματα κατά την εκτέλεση του:

1. Εξυπηρετείται η ουρά Q_P.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_P} += D_{Q_P} => DC_{Q_P} = 0 + 2524 = 2524.$
   - Εξυπηρετούνται με τη σειρά τα πρώτα 6 πακέτα μεγέθους 300 bytes.
     Ο deficit counter γίνεται: $DC_{Q_P} = 2524 - (6 * 300) = 724.$
   - Το επόμενο πακέτο, είναι μεγαλύτερο του deficit counter.
     Η ουρά έχει πάρει την μορφή: [2/1500, 4/300].

2. Εξυπηρετείται η ουρά Q_1.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_1} += D_{Q_1} => DC_{Q_1} = 0 + 2012 = 2012.$
   - Εξυπηρετείται το πρώτο πακέτο μεγέθους 1500 bytes.
     Ο deficit counter γίνεται: $DC_{Q_1} = 2012 - (1 * 1500) = 512.$
   - Το επόμενο πακέτο, είναι μεγαλύτερο του deficit counter.
     Η ουρά έχει πάρει την μορφή: [1/1500, 4/300, 1/1500]

3. Εξυπηρετείται η ουρά Q_P.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_P} += D_{Q_P} => DC_{Q_P} = 724 + 2524 = 3248.$
   - Εξυπηρετούνται τα 2 πακέτα μεγέθους 1500 bytes.
     Ο deficit counter γίνεται: $DC_{Q_P} = 3248 - (2 * 1500) = 248.$
   - Το επόμενο πακέτο, είναι μεγαλύτερο του deficit counter.
     Η ουρά έχει πάρει την μορφή: [4/300]

4. Εξυπηρετείται η ουρά Q_2.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_2} += D_{Q_2} => DC_{Q_2} = 0 + 1500 = 1500.$
   - Εξυπηρετούνται με τη σειρά τα πρώτα 5 πακέτα μεγέθους 300 bytes.
     Ο deficit counter γίνεται: $DC_{Q_2} = 1500 - (5 * 300) = 0.$
     Η ουρά έχει πάρει την μορφή: [1/1500, 5/300]

5. Εξυπηρετείται η ουρά Q_P.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_P} += D_{Q_P} => DC_{Q_P} = 248 + 2524 = 2772.$
   - Εξυπηρετούνται τα εναπομείναντα 4 πακέτα μεγέθους 300 bytes.
   - Η ουρά έχει μείνει κενή. Ο deficit counter της ουράς γίνεται 0.

6. Εξυπηρετείται η ουρά Q_1.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_1} += D_{Q_1} => DC_{Q_1} = 512 + 2012 = 2524.$
   - Εξυπηρετείται το πακέτο μεγέθους 1500 bytes.
     Ο deficit counter γίνεται: $DC_{Q_1} = 2524 - (1 * 1500) = 1024.$
   - Εξυπηρετούνται επίσης τα 3 πρώτα πακέτα μεγέθους 300 bytes.
     Ο deficit counter γίνεται: $DC_{Q_1} = 1024 - (3 * 300) = 124.$
   - Το επόμενο πακέτο, είναι μεγαλύτερο του deficit counter.
     Η ουρά έχει πάρει την μορφή: [1/300, 1/1500]

7. Εξυπηρετείται η ουρά Q_2.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_2} += D_{Q_2} => DC_{Q_2} = 0 + 1500 = 1500.$
   - Εξυπηρετείται το πακέτο μεγέθους 1500 bytes.
     Ο deficit counter γίνεται: $DC_{Q_2} = 1500 - (1 * 1500) = 0.$
     Η ουρά έχει πάρει την μορφή: [5/300]

8. Εξυπηρετείται η ουρά Q_1.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_1} += D_{Q_1} => DC_{Q_1} = 124 + 2012 = 2136.$
   - Εξυπηρετείται το πακέτο μεγέθους 300 bytes.
     Ο deficit counter γίνεται: $DC_{Q_1} = 2136 - (1 * 300) = 1836.$
   - Εξυπηρετείται επίσης το πακέτο μεγέθους 1500 bytes.
     Η ουρά έχει μείνει κενή. Ο deficit counter της ουράς γίνεται 0.

9. Εξυπηρετείται η ουρά Q_2.

   - Προστίθεται το quantum στον deficit counter:
     $DC_{Q_2} += D_{Q_2} => DC_{Q_2} = 0 + 1500 = 1500.$
   - Εξυπηρετούνται με τη σειρά τα υπόλοιπα 5 πακέτα μεγέθους 300 bytes.
   - Η ουρά έχει μείνει κενή. Ο deficit counter της ουράς γίνεται 0.

Όλες οι ουρές έχουν αδειάσει.

Κεφάλαιο 4

ΕΞΟΜΟΙΩΤΗΣ ΔΙΚΤΥΩΝ ns-3

4.1 Εισαγωγή

Η παρούσα διπλωματική εργασία βασίστηκε στον εξομοιωτή δικτύων ns-3, που είναι το τρίτο μέλος της οικογένειας των εξομοιωτών ns (network simulator). Όπως και οι προκάτοχοί του, ανήκει στην κατηγορία των εξομοιωτών δικτύων διακριτών γεγονότων (discrete event network simulator). Οι εξομοιωτές της κατηγορίας αυτής, μοντελοποιούν τη λειτουργία ενός συστήματος, ως μια ακολουθία διακριτών γεγονότων στο χρόνο, με κάθε γεγονός να συμβάλει στην διαμόρφωση της κατάστασης του συστήματος.

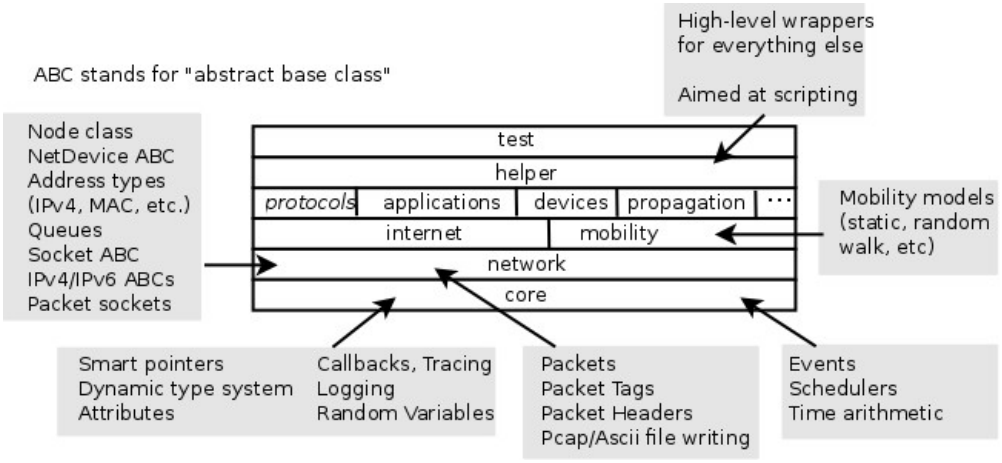
Στόχος του ns-3 είναι να παρέχει ένα ανοιχτό περιβάλλον προσομοίωσης που θα προτιμάται από την ερευνητική κοινότητα, παρέχοντας υποστήριξη για τις σύγχρονες ανάγκες της εξομοίωσης δικτύων και ελευθερία στην ανάπτυξη και επέκταση του εξομοιωτή. Ο εξομοιωτής διανέμεται κάτω από την άδεια ελεύθερου λογισμικού GNU GPLv2, και αυτή τη στιγμή βρίσκεται σε ενεργή ανάπτυξη από μια μεγάλη και ενθουσιώδη κοινότητα, που συμβάλει στην επέκταση των δυνατοτήτων του εξομοιωτή και την επαλήθευση του κώδικα υλοποίησης του.

Σε σχέση με τον προκάτοχό του (ns-2), είναι ένα καινούργιο project, χωρίς να διατηρεί κάποια συμβατότητα σε επίπεδο πηγαίου κώδικα, κάτι όμως που δεν αποκλείει την μεταφορά (porting) των μοντέλων που έχουν δημιουργηθεί για τον ns-2. Μέχρι στιγμής, ο ns-3 δεν έχει φτάσει την πληρότητα που προσφέρει ο ns-2, στα υποστηριζόμενα μοντέλα. Λόγω όμως της κατανεμημένης ανάπτυξης που πραγματοποιείται, οι ελλείψεις μειώνονται σημαντικά με την πάροδο του χρόνου. Ο ns-3 πάντως προσφέρει καινούργιες δυνατότητες όπως για παράδειγμα, σωστή υποστήριξη πολλαπλών διεπαφών (interfaces) σε ένα κόμβο, χρήση IP διευθυνσιοδότησης και εκτεταμένη συμβατότητα με τα πρωτόκολλα του Internet, υποστήριξη λεπτομερέστερων μοντέλων για ασύρματα δίκτυα (802.11), κ.α.

Ακολουθεί μια περιγραφή των βασικών στοιχείων που απαρτίζουν τον εξομοιωτή, των λειτουργιών που επιτελούν, καθώς και του τρόπου οργάνωσης τους.

4.2 Η δομή του ns-3

Ο πυρήνας του ns-3, καθώς και τα μοντέλα που υποστηρίζει είναι υλοποιημένα σε C++. Η δομή οργάνωσης τους αποτελεί μια στοίβα, όπου η λειτουργία κάθε επιπέδου βασίζεται στα προηγούμενα (Σχήμα: 4.1). Οι τοπολογίες που γράφουν οι χρήστες του εξομοιωτή (topology scripts), υλοποιούνται επίσης σε C++. Υπάρχει επιπλέον υποστήριξη της Python για τη συγγραφή τους, καθώς το μεγαλύτερο μέρος του API (Application programming interface) που προσφέρει ο εξομοιωτής, είναι διαθέσιμο και σε αυτήν.



Σχήμα 4.1: Η δομή του ns-3.

Τη στιγμή αυτή, ο εξομοιωτής βρίσκεται στην έκδοση: **ns-3.16**, ακολουθώντας συνήθως ένα χρονοδιάγραμμα τριών/τεσσάρων μηνών μεταξύ των εκδόσεων. Οι εκδόσεις είναι προσανατολισμένες στην τήρηση του χρονοδιαγράμματος και όχι στην υλοποίηση συγκεκριμένων χαρακτηριστικών σε κάθε έκδοση. Στην ανάπτυξη του ns-3 χρησιμοποιείται το mercurial, ως σύστημα διαχείρισης του πηγαίου κώδικα (revision control). Παράλληλα με τις εκδόσεις, διατηρείται ένα development repository (ns-3-dev), στο οποίο γίνεται η κύρια ανάπτυξη, και προκύπτει κάθε επόμενη έκδοση του ns-3, καθώς επίσης και άλλα repositories που χρησιμοποιούνται κατά κύριο λόγο στην ανάπτυξη νέων χαρακτηριστικών. Αξίζει επίσης να ειπωθεί ότι στην έκδοση ns-3.11, έγινε αλλαγή του τρόπου μετάφρασης του εξομοιωτή και αντίστοιχη οργάνωση του πηγαίου κώδικα, από μονολιθική βιβλιοθήκη σε modular.

Η διανομή του κώδικα κάθε έκδοσης, γίνεται με τα καθιερωμένα tarballs ή με τη χρήση του mercurial, όπου μπορεί να γίνει clone οποιουδήποτε από τα διαθέσιμα repositories. Ο πηγαίος κώδικας των modules που αποτελούν τον εξομοιωτή, βρίσκεται μέσα στο φάκελο *src/* του ns-3. Δηλαδή στην περίπτωση της τελευταίας έκδοσης, βρίσκεται στη διαδρομή: *ns-allinone-3.16/ns-3.16/src* . Ο φάκελος *scratch/* του ns-3, χρησιμοποιείται για την αποθήκευση των τοπολογιών που δημιουργεί ο χρήστης, αν και η χρήση του δεν είναι δεσμευτική.

Η διανομή *ns-allinone-3.16/*, πέρα από τον ns-3, περιλαμβάνει επίσης τα παρακάτω εργαλεία:

- *NetAnim (netanim-3.103/)*
  Ένα παραθυρικό περιβάλλον φτιαγμένο σε Qt, που χρησιμοποιεί xml trace αρχεία [1], για την οπτική αναπαράσταση των τοπολογιών, των πακέτων που μεταδίδουν καθώς και στατιστικών γι αυτά. Η δημιουργία του xml αρχείου καταγραφής, γίνεται μέσα στον πηγαίο κώδικα κάθε τοπολογίας (με τη βοήθεια του netanim module που υπάρχει στον ns-3).

- *Network Simulation Cradle (nsc-0.5.3/)*
  Μια βιβλιοθήκη που επιτρέπει τη χρήση πραγματικών TCP/IP stacks μέσα στο περιβάλλον του ns-3. Υποστήριξη του nsc είχε και ο ns-2.

- *PyBindGen (pybindgen-0.15.0.809/)*
  Μια Python βιβλιοθήκη που παράγει Python bindings για κώδικα γραμμένο σε C και C++. Χρησιμοποιείται στην αυτόματη παραγωγή Python bindings για το API του ns-3.

Είναι επίσης διαθέσιμο το repository: **ns-3-allinone**, το οποίο παρέχει Python scripts, που βοηθούν στο clone οποιουδήποτε διαθέσιμου repository **(download.py)**, καθώς και στη απλούστευση της μετάφρασης του εξομοιωτή **(build.py)**, διευκολύνοντας τους καινούργιους χρήστες. Το build.py είναι διαθέσιμο και στις εκδόσεις του ns-3.

Η μετάφραση του ns-3, όπως και η εκτέλεση των τοπολογιών που δημιουργεί ο χρήστης, γίνονται μέσω του συστήματος waf, που είναι υλοποιημένο σε Python. Μέσω παραμέτρων που υποστηρίζει το waf, ο χρήστης μπορεί εύκολα, να ρυθμίσει το είδος της μετάφρασης (debug, release, optimized) που επιθυμεί, ποια από τα διαθέσιμα modules θέλει να συμπεριλάβει στο build, καθώς επίσης και να ελέγχει την εκτέλεση των τοπολογιών που δημιουργεί.

4.3 Βασικές δομές του ns-3

Η προσομοίωση ενός δικτύου υπολογιστών, περιλαμβάνει τον καθορισμό της τοπολογίας του. Μαθηματικά, η τοπολογία ενός δικτύου μπορεί να περιγραφεί μέσω της θεωρίας των *γράφων*. Ο *γράφος* αποτελεί ένα διατεταγμένο σύνολο: $G = (V, E)$, όπου: *V*, το σύνολο των κόμβων και: *E*, το σύνολο των ακμών που συνδέουν τους κόμβους. Ένας κόμβος, μπορεί να αναπαραστήσει έναν υπολογιστή που είναι συνδεδεμένος στο δίκτυο, ενώ μια ακμή, τη σύνδεση που υπάρχει μεταξύ δύο κόμβων, μέσω ενός καναλιού επικοινωνίας. Ο σκοπός των δικτύων υπολογιστών, είναι η ανταλλαγή δεδομένων μεταξύ των κόμβων. Έτσι για την προσομοίωση τους, πρέπει να υπάρχει τρόπος αναπαράστασης της κίνησης που δημιουργείται στο δίκτυο. Ακολουθούν, βασικές δομές που περιλαμβάνονται στον ns-3 και του δίνουν τη δυνατότητα να περιγράψει τις εργασίες που αναφέρθηκαν.

[1] Υπάρχει επίσης το **PyViz** για την οπτική αναπαράσταση των εξομοιώσεων, που υλοποιείται στο **visualizer module** του ns-3. Σε αντίθεση με το NetAnim, λειτουργεί σε πραγματικό χρόνο χωρίς να χρησιμοποιεί trace files.

Node

Στον ns-3, ο κόμβος (*Node*) έχει την έννοια μιας υπολογιστικής συσκευής που είναι συνδεδεμένη σε ένα δίκτυο. Στον κόμβο, αντιστοιχίζονται οι υπόλοιπες δομές που περιγράφουν τη σύνδεση του στο δίκτυο, καθώς και τη λειτουργία του.

Channel

Αποτελεί το κανάλι που συνδέει δύο κόμβους μεταξύ τους, δηλαδή του φυσικού μέσου μετάδοσης και της τεχνολογίας που χρησιμοποιείται για τη μετάδοση. Έτσι είναι διαθέσιμοι διαφορετικοί τύποι καναλιών, όπως: *point-to-point, csma, wifi, wimax*, κ.α.

Net Device

Αποτελεί τη συσκευή που συνδέει κάθε κόμβο με το κανάλι. Συμπεριλαμβάνει, τη συσκευή που συνδέει στο δίκτυο έναν υπολογιστή (*Network Interface Card - NIC*), καθώς και τους απαραίτητους drivers της συσκευής που επιτρέπουν στο λειτουργικό σύστημα του υπολογιστή να χρησιμοποιήσει τη συσκευή. Έτσι για κάθε τύπο καναλιού, υπάρχει το αντίστοιχο Net Device: *PointToPointNetDevice, CsmaNetDevice, WifiNetDevice, WimaxNetDevice* κ.α. Κάθε κόμβος χρησιμοποιεί ένα Net Device για κάθε κανάλι που είναι συνδεδεμένος. Καθώς είναι δυνατή η εγκατάσταση πολλαπλών Net Devices σε ένα κόμβο, είναι δυνατή η σύνδεση ενός κόμβου σε πολλαπλά κανάλια.

Application

Αποτελεί την εφαρμογή που δημιουργεί την κίνηση στο δίκτυο, αποστέλλοντας ή λαμβάνοντας δεδομένα. Οι εφαρμογές προσαρτώνται στους κόμβους του δικτύου, ενώ υπάρχει η δυνατότητα χρήσης πολλαπλών εφαρμογών στον ίδιο κόμβο.

Topology Helpers

Η σύνθεση μιας τοπολογίας μεταξύ άλλων περιλαμβάνει, έναν αριθμό κόμβων που συνδέονται μεταξύ τους μέσω καναλιών, κάνοντας χρήση των Net Devices. Την εγκατάσταση πρωτοκόλλων επικοινωνίας και την ανάθεση διευθύνσεων. Τη δρομολόγηση, ίσως, των διερχόμενων πακέτων μεταξύ πολλαπλών Net Devices ενός κόμβου, καθώς και την εγκατάσταση εφαρμογών στους κόμβους, για τη δημιουργία της κίνησης. Λόγω του ότι οι παραπάνω λειτουργίες χρησιμοποιούνται κατά κόρον, έχουν δημιουργηθεί Topology Helpers, που διευκολύνουν και επιταχύνουν συνηθισμένες εργασίες και τη διασύνδεση μεγάλων τοπολογιών με πολλούς κόμβους.

4.4 ns-3 modules

Η τρέχουσα έκδοση του ns-3, αποτελείται από **40 modules**. Τα ονόματα τους φαίνονται στον πίνακα που ακολουθεί:

antenna	energy	propagation
aodv	flow-monitor	spectrum
applications	internet	stats
bridge	lte	tap-bridge
brite	mesh	test
buildings	mobility	tools
click	mpi	topology-read
config-store	netanim	uan
core	network	virtual-net-device
csma	nix-vector-routing	visualizer
csma-layout	olsr	wifi
dsdv	openflow	wimax
dsr	point-to-point	
emu	point-to-point-layout	

Πίνακας 4.1: ns-3 modules

Το **core** module, αποτελεί τον πυρήνα του εξομοιωτή και παρέχει βασικές λειτουργίες που επιτελούνται σε έναν εξομοιωτή διακριτών γεγονότων, όπως: παραγωγή ψευδοτυχαίων αριθμών, ορισμός και διαχείριση γεγονότων και παρακολούθηση του χρόνου της εξομοίωσης. Παρέχει επίσης: ένα object model με δυνατότητες αυτόματης διαχείρισης μνήμης και object aggregation, attribute και callback systems, καθώς και συστήματα logging και tracing. Ο ns-3 έχει τη δυνατότητα να λειτουργεί και σε πραγματικό χρόνο, κάτι που του δίνει τη δυνατότητα να χρησιμοποιεί πραγματικά network stacks (όπως το nsc που αναφέρθηκε παραπάνω).

Το **network** module, περιέχει ένα packet framework, που μπορεί να χρησιμοποιηθεί για τη δημιουργία διαφορετικού τύπου πακέτων. Επίσης περιέχει βασικές κλάσεις για τη δημιουργία κόμβων (Nodes) και Net Devices, διαφορετικού τύπου. Παρέχει επίσης δύο socket APIs, το ns-3 socket API, και βάσει αυτού, ένα "POSIX-like" socket api. Τέλος ορίζει μια βασική κλάση για τη δημιουργία ουρών αναμονής των πακέτων (queues), υλοποιώντας επίσης δύο διαδεδομένους τύπους: Drop Tail (FIFO) και Random Early Detection (RED).

Τα δύο παραπάνω modules, σχεδιάστηκαν ώστε να αποτελούν ένα γενικό πυρήνα εξομοίωσης για ποικιλία δικτύων, πέρα από τα IP δίκτυα στα οποία επικεντρώνεται ο ns-3.

Το module **point-to-point** χρησιμοποιείται για τη δημιουργία *point-to-point* καναλιών και Net Devices, ενώ το module **point-to-point-layout** χρησιμοποιείται για διευκόλυνση της δημιουργίας των *point-to-point* τοπολογιών: *dumbbell, grid* και *star*. Το module **csma** χρησιμοποιείται για τη δημιουργία *csma* καναλιών και Net Devices, ενώ το module **csma-layout** για τη δημιουργία της *csma* τοπολογίας αστέρα. Το module **bridge** χρησιμοποιείται για τη "γεφύρωση" πολλαπλών *csma* δικτύων (όπως ορίζεται στο πρωτόκολλο: *IEEE 802.1D*). Το module **internet** υλοποιεί πρωτόκολλα του Internet, παρέχοντας πρωτόκολλα όπως: *IPv4, ARP, UDP, TCP, IPv6, Neighbor Discovery* κ.α., ενώ το module **applications** χρησιμοποιείται για τη δημιουργία εφαρμογών.

Τα modules: **flow-monitor**, **stats**, **tools** και **visualizer**, βοηθούν στην ανάλυση των προσομοιώσεων. Το module **emu** επιτρέπει την αλληλεπίδραση του ns-3 με πραγματικά δίκτυα, ενώ τα modules: **tap-bridge** και **virtual-net-device**, επιτρέπουν την αλληλεπίδραση πραγματικών εφαρμογών και πρωτοκόλλων με το περιβάλλον του ns-3.

Επίσης, υπάρχουν modules για την υποστήριξη εξωτερικών βιβλιοθηκών και εργαλείων όπως: το **click** για το *Click Modular Router* [30], το **mpi** για την εκτέλεση κατανεμημένων προσομοιώσεων μέσω του πρωτοκόλλου *MPI (Message Passing Interface)*, το **openflow** για τη συνεργασία με το πρωτόκολλο *OpenFlow*, το **netanim** για τη συνεργασία με το περιβάλλον NetAnin (που αναφέρθηκε προηγουμένως), το **brite** για τη συνεργασία με το περιβάλλον *BRITE* που δημιουργεί ρεαλιστικές διαδικτυακές τοπολογίες, καθώς και το **topology-read** για τη δημιουργία πολύπλοκων τοπολογιών με τη βοήθεια εξωτερικών εργαλείων. Για τη δρομολόγηση μεγάλων τοπολογιών, υπάρχει το module **nix-vector-routing**, που υποστηρίζει *IPv4 point-to-point* και *csma* δίκτυα.

Τέλος, υπάρχουν αρκετά modules που βρίσκουν εφαρμογή στην προσομοίωση ασύρματων δικτύων. Καθώς ένας από τους στόχους της διπλωματικής, είναι η ανάλυση των δυνατοτήτων που προσφέρει ο ns-3 στα ασύρματα δίκτυα, γίνεται αναλυτική περιγραφή τους στο κεφάλαιο που ακολουθεί.

Κεφάλαιο 5

ns-3 ΚΑΙ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ

5.1 Εισαγωγή

Η προσομοίωση, μπορεί να συμβάλει καθοριστικά στη μελέτη ενός δικτύου. Μέσω αυτής, μπορούν να δοκιμαστούν οι παράμετροι που το συνθέτουν (σε ποικιλία σεναρίων) και να βελτιστοποιηθούν. Η διαδικασία αυτή μπορεί να συμβάλει στη συνολική μείωση του κόστους δημιουργίας και εγκατάστασης ενός δικτύου, καθώς οι δοκιμές σε πραγματική κλίμακα μπορούν να διεξαχθούν πολύ αργότερα και με την ελάχιστη δυνατή συχνότητα. Η πρακτική χρησιμότητα των εξομοιωτών, έγκειται στην επακριβή αναπαράσταση των φαινομένων που προσομοιώνουν (μέσω των μοντέλων που χρησιμοποιούν), καθώς και στις καλές επιδόσεις. Οι παράμετροι αυτοί όμως, δεν μπορούν να ικανοποιηθούν ταυτόχρονα. Έτσι, η ισορροπία που προσφέρει ένας εξομοιωτής σε αυτές, καθορίζει τελικά και τη χρησιμότητά του.

Τη στιγμή αυτή, ο ns-3 περιέχει **15 modules**, που βρίσκουν εφαρμογή σε ασύρματα δίκτυα. Σε αυτά, υλοποιούνται μοντέλα: διαγραμμάτων ακτινοβολίας κεραιών, απώλειας και καθυστέρησης διάδοσης της ακτινοβολίας σε διάφορα περιβάλλοντα (όπως παρουσία κτηρίων), του φάσματος της ηλεκτρομαγνητικής ακτινοβολίας, πηγών ενέργειας και συσκευών κατανάλωσης, κίνησης (για την προσομοίωση της κίνησης φορητών συσκευών), υποβρύχιας ακουστικής επικοινωνίας (καθώς και υποβρύχιων σκαφών που τη χρησιμοποιούν). Υλοποιούνται επίσης μοντέλα που αφορούν τις τεχνολογίες: *Wifi*, *LTE* και *WiMAX*, καθώς και πρωτόκολλα που βρίσκουν εφαρμογή σε ασύρματα *ad-hoc* δίκτυα.

Στο κεφάλαιο αυτό, θα γίνει περιγραφή των δυνατοτήτων που παρέχει κάθε module, συνοψίζοντας στο τέλος τις δυνατότητες που δίνουν συνολικά στον εξομοιωτή.

5.2 ns-3 modules

5.2.1 antenna module

Το module χρησιμοποιείται για την υλοποίηση μοντέλων διαγραμμάτων ακτινοβολίας (radiation patterns) κεραιών. Το διάγραμμα ακτινοβολίας μιας κεραίας, είναι το γράφημα της απολαβής (gain) που παρουσιάζει η κεραία συναρτήσει της κατεύθυνσης. Η απολαβή είναι μέτρο απόδοσης της κεραίας προς μία κατεύθυνση, εκφράζεται σε dB και προκύπτει από το λόγο της εκπεμπόμενης (ή λαμβανόμενης) ισχύος προς μία κατεύθυνση, σε σχέση με μια ιδανική ισοτροπική κεραία, που οδηγείται από το ίδιο σήμα εισόδου/εξόδου.
Μέχρι στιγμής έχουν υλοποιηθεί τα παρακάτω μοντέλα:

- **Isotropic Antenna Model**.
  Είναι ένα θεωρητικό ισοτροπικό μοντέλο, όπου η κεραία παρουσιάζει ομοιόμορφη απολαβή $0dB$ προς κάθε κατεύθυνση.

- **Cosine Antenna Model**.
  Χρησιμοποιείται στον υπολογισμό ρεαλιστικών διαγραμμάτων ακτινοβολίας, κάνοντας χρήση ενός εκθετικού συνημιτονικού κανόνα για τον καθορισμό του σχήματος τους.

- **Parabolic Antenna Model**.
  Χρησιμοποιείται συνήθως στη μοντελοποίηση των κεραιών που συνθέτουν τις κυψέλες των κυψελωτών δικτύων.

5.2.2 aodv module

Στο module υλοποιείται το πρωτόκολλο δρομολόγησης: *AODV (Ad Hoc On-Demand Distance Vector)* [19], που σχεδιάστηκε με σκοπό να χρησιμοποιείται από κινητούς κόμβους ασύρματων ad-hoc δικτύων. Τα ασύρματα ad-hoc δίκτυα, αποτελούνται από ασύρματες συσκευές (κινητές ή ακίνητες) που έχουν ισότιμο ρόλο και τη δυνατότητα να συνδέονται ελεύθερα μεταξύ τους (όταν βρίσκονται εντός εμβέλειας). Η επικοινωνία διεξάγεται δυναμικά μεταξύ των κόμβων, χωρίς να βασίζεται σε κάποια προ-υπάρχουσα υποδομή. Δημιουργώντας έτσι ένα αποκεντρωμένο δίκτυο, όπου κάθε κόμβος συμμετέχει στη δρομολόγηση της κίνησης, ανάλογα με τις ανάγκες επικοινωνίας και των διαθέσιμων συνδέσμων μεταξύ των κόμβων (όπου στην περίπτωση ύπαρξης κινητών κόμβων, μπορεί να υπάρχει συχνή αλλαγή τους).

Το πρωτόκολλο *AODV*, δημιουργεί μονοπάτια επικοινωνίας μεταξύ των κόμβων μόνο όταν ζητηθούν (reactive protocol). Όταν κάποιος κόμβος θέλει να μεταδώσει, κάνει broadcast μια αίτηση σύνδεσης (Route Request - RREQ) με τον κόμβο που επιθυμεί. Η αίτηση προωθείται από τους κόμβους του δικτύου, ενώ κάθε κόμβος καταγράφει τον προηγούμενο κόμβο που την έστειλε. Η διαδικασία αυτή, δημιουργεί μια σειρά προσωρινών μονοπατιών προς τον αρχικό κόμβο.

Μόλις η αίτηση φτάσει στον κόμβο προορισμού της, χρησιμοποιεί το μονοπάτι που έχει δημιουργηθεί και ειδοποιεί τον αρχικό κόμβο (Route Reply - RREP). Αυτός με τη σειρά του αρχίζει να επικοινωνεί, ενώ στην περίπτωση εύρεσης πολλαπλών μονοπατιών, χρησιμοποιεί το μονοπάτι με τα λιγότερα hops.

Σε περίπτωση απώλειας της σύνδεσης, ειδοποιείται ο αποστολέας (Route Error - RERR) και χρησιμοποιεί την ίδια διαδικασία για τη δημιουργία μιας καινούργιας σύνδεσης. Οι πίνακες δρομολόγησης που διατηρούνται στους κόμβους συντηρούνται συστηματικά, απομακρύνοντας τις ανενεργές καταχωρίσεις που δημιουργούνται. Για την παρακολούθηση των αιτήσεων που δημιουργούνται, γίνεται χρήση ενός μοναδικού αριθμού σε κάθε αίτηση (RREQ ID), που δίνει τη δυνατότητα στους κόμβους να αναγνωρίζουν κάθε αίτηση, ώστε να την προωθούν από μία φορά.

Η υλοποίηση του πρωτοκόλλου συνδυάζει την aodv υλοποίηση του ns-2, καθώς και την υλοποίηση AODV-UU [29]. Λειτουργεί κάνοντας χρήση του πρωτοκόλλου μεταφοράς UDP και του IPv4 πρωτοκόλλου του διαδικτύου.

5.2.3 buildings module

Το module χρησιμοποιείται για την υλοποίηση μοντέλων επίδρασης των κτηρίων σε ασύρματες επικοινωνίες. Το module, παρέχει συνοπτικά τις εξής δυνατότητες:

- Δημιουργία κτηρίων μέσα στο περιβάλλον της προσομοίωσης.
- Προσθήκη κόμβων εντός και εκτός των κτηρίων.
- Υποστήριξη μοντέλων απώλειας διάδοσης.

Ποιο αναλυτικά:

Για τη δημιουργία κτηρίων, παρέχονται οι τύποι κτηρίων: κατοικία, γραφείο, εμπορικό, με υλικά κατασκευής των εξωτερικών τοίχων: ξύλο, τσιμέντο (με παράθυρα ή χωρίς), πέτρινα μπλοκ. Υπάρχει δυνατότητα ρύθμισης του αριθμού των ορόφων, και του αριθμού των δωματίων (στις δύο διαστάσεις του επιπέδου). Κάθε κτήριο αποτελεί ένα ορθογώνιο παραλληλεπίπεδο, όπου κάθε όροφος χωρίζεται σε δωμάτια ίδιου μεγέθους. Όλοι οι όροφοι είναι πανομοιότυποι.

Η προσθήκη κόμβων στο περιβάλλον προσομοίωσης, δίνει τη δυνατότητα τοποθέτησης των κόμβων, εκτός των κτηρίων ή εντός κάποιου δωματίου από τα διαθέσιμα κτήρια.

Τα μοντέλα απώλειας διάδοσης που χρησιμοποιούνται, γενικά λαμβάνουν υπόψιν τη θέση των κόμβων που επικοινωνούν (σε σχέση με τα κτήρια), την απόσταση τους, τον τύπο των κτηρίων και το υλικό κατασκευής των τοίχων (όπως αναφέρθηκαν προηγουμένως), για να υπολογίσουν την απώλεια διάδοσης. Ακολουθεί μια περιγραφή των μοντέλων που έχουν υλοποιηθεί μέχρι στιγμής, αναφέροντας παράλληλα το πεδίο εφαρμογής τους:

- **ITU P.1238**: Είναι μοντέλο απώλειας διάδοσης, στο εσωτερικό ενός κτηρίου. Ο υπολογισμός του, λαμβάνει υπόψιν τον τύπο του κτηρίου (κατοικία, γραφείο, εμπορικό), τη συχνότητα μετάδοσης, τον αριθμό των ορόφων που απέχουν οι κόμβοι (≥ 1), καθώς και την απόσταση των κόμβων μεταξύ τους σε μέτρα (> 1).

- **BuildingsPropagationLossModel**: Είναι κλάση που υλοποιεί μοντέλα απώλειας διάδοσης, για φαινόμενα που δημιουργούνται κατά τη διάδοση ηλεκτρομαγνητικής ακτινοβολίας, παρουσία κτηρίων:

  - **External Wall Loss (EWL)**: Υλοποιεί ένα μοντέλο απώλειας διάδοσης, κατά τη διείσδυση της ακτινοβολίας σε εξωτερικό τοίχο, συναρτήσει του υλικού κατασκευής του. Βρίσκει εφαρμογή στην επικοινωνία μεταξύ κόμβων εντός-εκτός κτηρίων.
  - **Internal Walls Loss (IWL)**: Υλοποιεί ένα μοντέλο απώλειας διάδοσης, κατά τη διείσδυση της ακτινοβολίας στους εσωτερικούς τοίχους ενός κτηρίου. Χρησιμοποιεί τη θεώρηση ότι υπάρχει σταθερή απώλεια από κάθε τοίχο, και προσεγγίζει τον αριθμό των τοίχων που παρεμβάλλονται μεταξύ δύο κόμβων, χρησιμοποιώντας την απόσταση manhattan μεταξύ των δωματίων που παρεμβάλλονται.
  - **Height Gain Model (HG)**: Μοντελοποιεί την αύξηση της απολαβής λόγω αύξησης του ύψους που βρίσκεται ο κόμβος (όροφος κτηρίου). Βρίσκει εφαρμογή στις εντός-εκτός κτηρίων επικοινωνίες, όπου οι κόμβοι έχουν υψομετρική διαφορά.
  - **Shadowing Model**: Μοντελοποιεί το ομώνυμο φαινόμενο. Το φαινόμενο παρουσιάζεται λόγω ύπαρξης εμποδίων στο μονοπάτι διάδοσης. Η υλοποίηση περιλαμβάνει τρεις περιπτώσεις για τη θέση των κόμβων σε σχέση με τα κτήρια: εντός, εκτός, εντός-εκτός.

  Η κλάση περιλαμβάνει παραμέτρους, που καθορίζουν τη λειτουργία των παραπάνω μοντέλων. Μεταξύ άλλων είναι, η συχνότητα εκπομπής, το ύψος του κτηρίου, το περιβάλλον (αστικό, ήμι-αστικό, ανοιχτή περιοχή) και το μέγεθος της πόλης (μικρή, μεσαία, μεγάλη).

- Υλοποιούνται επίσης μοντέλα, που συνδυάζουν διάφορα μοντέλα απώλειας διάδοσης (όπως τα παραπάνω), αυτοματοποιώντας τη διαδικασία υπολογισμού της απώλειας διάδοσης.

  – **HybridBuildingsPropagationLossModel**
  Συνδυάζει γνωστά μοντέλα απώλειας διάδοσης, λαμβάνοντας υπόψιν τη θέση των κόμβων. Τα μοντέλα που περιλαμβάνει είναι:

    * **OkumuraHataPropagationLossModel (OH)**.
    * **ItuR1411LosPropagationLossModel** και **ItuR1411NlosOverRooftopPropagationLossModel (I1411)**.
    * **ItuR1238PropagationLossModel (I1238)**.
    * Τα μοντέλα: **EWL**, **HG**, **IWL**, που περιγράφηκαν παραπάνω.

  Με τη βοήθεια ενός αλγόριθμου, που λαμβάνει υπόψιν τη θέση των κόμβων (εντός-εκτός κτηρίων), την απόσταση μεταξύ τους, καθώς και το ύψος που βρίσκονται, χρησιμοποιεί κάποιο (ή συνδυασμούς) από τα παραπάνω μοντέλα για να υπολογίσει την απώλεια διάδοσης. Ο συνδυασμός διαφορετικών μοντέλων που πραγματοποιείται, μπορεί να δημιουργήσει ασυνέχειες στον υπολογισμό, που μπορούν όμως να αντιμετωπιστούν με προσεκτική ρύθμιση των διαθέσιμων παραμέτρων.

  – **OhBuildingsPropagationLossModel**
  Κάνει χρήση του μοντέλου: *Okumura Hata (OH)* και των μοντέλων: *EWL*, *HG*, *IWL*. Δημιουργήθηκε με σκοπό την αποφυγή των ασυνεχειών του προηγούμενου μοντέλου, έχοντας όμως λιγότερη συνολική ακρίβεια. Χρησιμοποιείται και εδώ αλγόριθμος που αποφασίζει, ανάλογα με τη θέση των κόμβων, ποιο (ή ποια) από τα διαθέσιμα μοντέλα θα χρησιμοποιηθούν.

5.2.4 dsvd module

Στο module υλοποιείται το πρωτόκολλο δρομολόγησης: *Destination-Sequenced Distance Vector (DSDV)*, που βρίσκει εφαρμογή σε κινητά ad-hoc ασύρματα δίκτυα (Mobile Ad-hoc Networks - MANETs), που είναι ένας τύπος ασύρματων ad-hoc δικτύων. Στα δίκτυα τύπου MANET, οι κόμβοι του είναι κινητές συσκευές ελεύθερες να κινηθούν προς κάθε κατεύθυνση, κάτι που μπορεί να προκαλέσει συχνή αλλαγή των συνδέσμων μεταξύ τους.

Το πρωτόκολλο *DSVD*, χρησιμοποιεί πίνακες δρομολόγησης στους κόμβους, που ενημερώνονται με την κατάσταση των συνδέσμων μεταξύ τους (pro-active protocol). Ο πίνακας δρομολόγησης ενός κόμβου, περιέχει πληροφορίες για κάθε κόμβο που είναι προσβάσιμος (άμεσα ή έμμεσα) από αυτόν.

Μια καταχώρηση στον πίνακα δρομολόγησης, περιλαμβάνει τη διεύθυνση του κόμβου προορισμού, το επόμενο hop προς αυτόν, τον αριθμό των hops μέχρι τον προορισμό, έναν αύξοντα αριθμό που παράγεται από τον κόμβο προορισμού, καθώς και το χρόνο της τελευταίας ενημέρωσης για την κατάσταση του κόμβου προορισμού.

Οι κόμβοι ενημερώνουν τους πίνακες δρομολόγησης που διαθέτουν, ανταλλάσσοντας μεταξύ τους μηνύματα που περιλαμβάνουν τα πεδία: διεύθυνση του κόμβου προορισμού, αύξοντα αριθμό του κόμβου προορισμού και αριθμό των hops μέχρι τον κόμβο προορισμού. Η ενημέρωση γίνεται με δύο τρόπους, με πλήρεις περιοδικές (periodic) ενημερώσεις (στις οποίες κάθε κόμβος κάνει broadcast όλο τον πίνακα δρομολόγησης) και μικρές trigger ενημερώσεις (μεταξύ των περιοδικών), όπου γίνεται broadcast των καταχωρήσεων του πίνακα δρομολόγησης που υφίστανται αλλαγή, όταν αυτές συμβαίνουν. Στην επεξεργασία των ενημερώσεων από τους κόμβους, ελέγχεται κάθε καταχώρηση αν διαθέτει μεγαλύτερο αύξοντα αριθμό σε σχέση με τον πίνακα δρομολόγησης (για να γίνει δεκτή), ενώ στην περίπτωση όπου ο αριθμός δεν έχει αλλάξει, έχει προτεραιότητα η καταχώρηση με τα λιγότερα hop.

5.2.5 dsr module

Στο module υλοποιείται το πρωτόκολλο δρομολόγησης: *Dynamic Source Routing (DSR)* [22], που έχει σχεδιαστεί ειδικά για κινητά multi-hop ασύρματα ad-hoc δίκτυα, τα οποία ανήκουν στην κατηγορία των ασύρματων ad-hoc δικτύων. Με τον όρο multi-hop, περιγράφεται η επικοινωνία μεταξύ κόμβων που δεν είναι γειτονικοί, παρεμβάλλονται δηλαδή και άλλοι κόμβοι στα ενδιάμεσα. Έτσι στην περίπτωση των ad-hoc δικτύων, θα πρέπει οι ενδιάμεσοι κόμβοι να δρομολογήσουν την κίνηση αυτή. Όπως και στο πρωτόκολλο *AODV*, τα μονοπάτια επικοινωνίας δημιουργούνται δυναμικά όταν οι κόμβοι τα χρειάζονται. Το πρωτόκολλο αποτελείται από δύο βασικούς μηχανισμούς:

- **Route Discovery**.
  Χρησιμοποιείται για την ανακάλυψη μονοπατιών μεταξύ των κόμβων που θέλουν να επικοινωνήσουν.

- **Route Maintenance**.
  Χρησιμοποιείται κατά τη διάρκεια της επικοινωνίας μεταξύ των κόμβων, ανιχνεύοντας την κατάσταση του μονοπατιού που χρησιμοποιείται.

Κάθε κόμβος πρέπει να διατηρεί μία Route Cache, στην οποία διατηρεί πληροφορίες δρομολόγησης (για την ύπαρξη συνδέσμων μεταξύ άλλων κόμβων). Επίσης, το πρωτόκολλο κάνει χρήση της τεχνική δρομολόγησης: *explicit source routing*, στην οποία τα πακέτα μεταφέρουν στην επικεφαλίδα τους τη σειρά των κόμβων που πρέπει να διασχίσει το πακέτο, απαλλάσσοντας τους κόμβους από την ανάγκη διατήρησης πινάκων δρομολόγησης για την προώθηση τους.

Όταν ένας κόμβος θέλει να μεταδώσει σε κάποιον άλλο, ελέγχει την Route Cache που διατηρεί, για πληροφορίες δρομολόγησης προς αυτόν. Αν δεν υπάρχουν, ενεργοποιεί το μηχανισμό *Route Discovery*. Ο κόμβος κάνει broadcast μια αίτηση *RREQ* που παραλαμβάνεται από τους γειτονικούς του. Κάθε κόμβος που τη λαμβάνει ελέγχει αν είναι ο παραλήπτης και την αναμεταδίδει στους γειτονικούς του κόμβους. Η αίτηση περιλαμβάνει ένα μοναδικό αριθμό (Identification) για να αναγνωρίζει κάθε κόμβος ποιες αιτήσεις έχει επεξεργαστεί. Στην αίτηση διατηρείται επίσης μία λίστα όλων των κόμβων που έχει διασχίσει. Όταν φτάσει η αίτηση στον παραλήπτη, ειδοποιεί τον αποστολέα (*RREP*) με τη λίστα των κόμβων που βρήκε στην αίτηση. Ο αποστολέας λαμβάνοντας την απάντηση, αποθηκεύει στην *Route Cache* που διατηρεί, τη διαδρομή που περιέχει η απάντηση, ώστε να μπορεί να τη χρησιμοποιήσει για μετάδοση.

Κατά τη μετάδοση, χρησιμοποιείται ο μηχανισμός *Route Maintenance*, για τον έλεγχο του ενεργού μονοπατιού. Κάθε κόμβος είναι υπεύθυνος για το σύνδεσμο με τον επόμενο κόμβο που προωθεί την επικοινωνία. Με τη χρήση τεχνικών επαλήθευσης, γίνεται δυνατός ο έλεγχος της κατάστασης των συνδέσμων. Σε περίπτωση που ένας σύνδεσμος πάψει να υπάρχει, γίνεται προσπάθεια χρήσης άλλης διαδρομής (εφόσον είναι διαθέσιμη στη Route Cache), ή επανάληψη του μηχανισμού route discovery για την εύρεση καινούργιας διαδρομής.

5.2.6 energy module

Το module χρησιμοποιείται για την υλοποίηση μοντέλων πηγών ενέργειας και συσκευών που την καταναλώνουν. Βρίσκει εφαρμογή σε φορητές συσκευές, που κάνουν χρήση αυτόνομων πηγών ενέργειας και χρειάζεται να μελετηθεί η κατανάλωση που παρουσιάζουν σε διάφορες εφαρμογές. Σε ένα κόμβο μπορεί να υπάρχει μία (ή περισσότερες) πηγές ενέργειας, όπου κάθε πηγή χρησιμοποιείται από ένα (ή περισσότερα) μοντέλα συσκευών που καταναλώνουν ενέργεια. Ο κόμβος μπορεί να ενημερώνεται για την κατάσταση της πηγής όπως και για την περίπτωση εξάντλησης της. Ο υπολογισμός της κατανάλωσης, γίνεται με περιοδικό έλεγχο των συνδεδεμένων συσκευών και υπολογισμό της συνολικής κατανάλωσης, καθώς και μετά από ενημέρωση της πηγής από συσκευή για αλλαγή της κατάστασης της. Η συσκευή, βρίσκεται σε κάποια κατάσταση που καθορίζει την κατανάλωση της. Οι πιθανές καταστάσεις της συσκευής μπορεί να είναι πεπερασμένες ή και όχι.
Τη στιγμή αυτή, τα διαθέσιμα μοντέλα κατανάλωσης συσκευών είναι:

- **SimpleDeviceEnergyModel**.
  Είναι απλό μοντέλο που παρέχει τη δυνατότητα ρύθμισης της κατανάλωσης από το χρήστη. Προορίζεται κυρίων για τη δοκιμή των μοντέλων πηγών ενέργειας.

- **WifiRadioEnergyModelPhyListener**.
  Είναι γενικό μοντέλο κατανάλωσης συσκευών *Wi-Fi*. Υποστηρίζει ρύθμιση των παραμέτρων κατανάλωσης της συσκευής, καθώς και τέσσερις καταστάσεις κατά τη λειτουργία της *(Tx, Rx, Idle, Sleep)*, με διαφορετική κατανάλωση σε κάθε μία.

Ενώ υπάρχουν διαθέσιμα, τα παρακάτω μοντέλα πηγών ενέργειας:

- **BasicEnergySource**.
  Είναι μοντέλο πηγής ενέργειας, με γραμμικό τρόπο φόρτισης και αποφόρτισης.

- **LiIonEnergySource**.
  Είναι γενικό μοντέλο μπαταριών ιόντων λιθίου. Οι παράμετροι του μοντέλου μπορούν να ρυθμιστούν για την προσομοίωση οποιουδήποτε τύπου.

- **RvBatteryModel**.
  Είναι αναλυτικό μη-γραμμικό μοντέλο μπαταριών, που μπορεί να προσομοιώσει με επιτυχία τα φαινόμενα *Rate Capacity*[1] και *Recovery*[2] που εμφανίζουν οι μπαταρίες.

5.2.7 lte module

Στο module υλοποιείται το μοντέλο προσομοίωσης *LENA*, που χρησιμοποιείται στην προσομοίωση ασύρματων δικτύων βασισμένα στο πρότυπο: *LTE (Long-Term Evolution)*. Το πρότυπο *LTE*, βελτιώνει την απόδοση των υπαρχόντων δικτύων κινητής τηλεφωνίας, κάνοντας χρήση καινούργιων τεχνικών ψηφιακή επεξεργασίας σημάτων (Digital Signal Proccessing - DSP), απλουστεύοντας παράλληλα την αρχιτεκτονική του δικτύου, μετατρέποντάς το σε δίκτυο μεταγωγής πακέτων μέσω του πρωτοκόλλου IP. Η αρχιτεκτονική του *LENA*, αποτελείται από δύο κύρια μέρη: το μοντέλο *LTE* και το μοντέλο *EPC (Evolved Packet Core)*. Το δίκτυο της αρχιτεκτονικής, αποτελείται από ασύρματες συσκευές: *User Equipment (UE)*, που συνδέονται σε *EnodeB (eNB)* κόμβους. Αυτοί με τη σειρά τους μπορούν και συνδέονται μέσω κόμβων: *SGW/PGW (Serving Gateway / Packet Data Network Gateway)*, σε άλλα δίκτυα.

Το μοντέλο *LTE*, υποστηρίζει την αξιολόγηση των παρακάτω στοιχείων ενός *LTE* συστήματος: Radio Resource Management (RRM), χρονοδρομολόγηση πακέτων με υποστήριξη QoS, συντονισμό των παρεμβολών μεταξύ κυψελών και Dynamic Spectrum Access (DSA).

Ο όρος RRM, αναφέρεται σε τεχνικές που εφαρμόζονται σε ασύρματα δίκτυα, για την αποδοτική αξιοποίηση των περιορισμένων πόρων του δικτύου (του συχνοτικού φάσματος). Ο όρος DSA, αναφέρεται σε τεχνικές που εφαρμόζονται σε ασύρματα δίκτυα, για την αύξηση της συνολικής απόδοσης του δικτύου. Είναι δυνατή η χρήση του antenna module για τη σχεδίαση των κεραιών των UE και eNB. Επίσης είναι δυνατή η χρήση του buildings module και των μοντέλων διάδοσης που υποστηρίζει (που σχεδιάστηκαν αρχικά για το *LTE*).

Το μοντέλο *EPC*, υποστηρίζει την προσομοίωση της IP συνδεσιμότητας μέσω του μοντέλου *LTE*. Μπορεί να προσομοιώσει πολλαπλά *UEs*, συνδεδεμένα σε πολλαπλά *eNBs* που είναι συνδεδεμένα σε ένα μοναδικό *SGW/PGW* κόμβο. Μέχρι στιγμής, υπάρχει υποστήριξη για το πρωτόκολλο διαδικτύου IPv4, και τα πρωτόκολλα μεταφοράς TCP και UDP.

[1] Είναι φαινόμενο που μειώνει τη διάρκεια ζωής μιας μπαταρίας, όταν γίνεται κατανάλωση μεγαλύτερη των προδιαγραφών της.

[2] Περιγράφει την αύξηση στη ζωή μιας μπαταρίας, όταν δεν γίνεται συνεχής κατανάλωση από την συσκευή.

5.2.8 mesh module

Στο module υλοποιείται ένα Net Device βασισμένο στο πρότυπο: *IEEE 802.11s*. Το πρότυπο επεκτείνει τις δυνατότητες του υπό-επιπέδου *MAC (Medium Access Control)*, όπως αυτό ορίζεται στο σύνολο πρωτοτύπων: *IEEE 802.11*, δίνοντας τη δυνατότητα δημιουργίας *WLAN (Wireless Local Area Network) mesh* δικτύων. Ένα ασύρματο δίκτυο τύπου mesh *(Wireless Mesh Network - WMN)*, είναι ένας ειδικός τύπος ασύρματων ad-hoc δικτύων. Απαρτίζεται από σταθμούς πελάτες (όπως οι κόμβοι των ασύρματων ad-hoc δικτύων), ενώ παράλληλα μπορεί να περιλαμβάνει δρομολογητές και πύλες δικτύων (gateways). Έτσι έχει μια ποιο οργανωμένη δομή από τα υπόλοιπα ασύρματα ad-hoc δίκτυα. Το πρότυπο ορίζει το πρωτόκολλο δρομολόγησης: *Hybrid Wireless Mesh Protocol (HWMP)*, που βασίζεται στο πρωτόκολλο *AODV*. Η επικοινωνία γίνεται κάνοντας χρήση κάποιου από τα υπάρχοντα πρωτόκολλα ασύρματης επικοινωνίας: *IEEE 802.11{a,b,g,n}*. Οι σταθμοί ονομάζονται: Mesh STAs και συνδέονται μεταξύ τους, έχοντας παράλληλα τη δυνατότητα να συνδεθούν και σε ασύρματα σημεία πρόσβασης (Access Points - APs) ή πύλες, αποκτώντας πρόσβαση σε άλλα δίκτυα (μαζί και το δίκτυο mesh που ανήκουν).

5.2.9 mobility module

Το module χρησιμοποιείται για την μοντελοποίηση της κινητικότητας των κόμβων. Προς το παρόν, υποστηρίζει το καρτεσιανό σύστημα συντεταγμένων, αν και δεν αποκλείεται η υλοποίηση και χρήση άλλων συστημάτων συντεταγμένων. Η αρχική τοποθέτηση των κόμβων, γίνεται με τη βοήθεια *position allocators*. Η μετέπειτα κίνηση τους εξαρτάται από το μοντέλο κίνησης που έχει διαλέξει ο χρήστης. Τα διαθέσιμα μοντέλα κίνησης είναι:

- **ConstantPosition**: Διατηρεί σταθερή θέση, μέχρι να γίνει ρητή αλλαγή της.
- **ConstantVelocity**: Διατηρεί σταθερή ταχύτητα, μέχρι να γίνει ρητή αλλαγή της.
- **ConstantAcceleration**: Διατηρεί σταθερή επιτάχυνση, μέχρι να γίνει ρητή αλλαγή της.
- **GaussMarkov**: Υλοποιεί μια τρισδιάστατη εκδοχή του μοντέλου κινητικότητας Gauss-Markov. Το μοντέλο χρησιμοποιεί μνήμη για να αποθηκεύει το ιστορικό της κίνησης του αντικειμένου, και το χρησιμοποιεί για να υπολογίσει την αλλαγή των παραμέτρων της. Το μοντέλο είναι κατάλληλο για μοντελοποίηση της κίνησης αεροπλάνων. Η κίνηση οριοθετείται από ένα τρισδιάστατο "κουτί".
- **Hierarchical**: Δίνει τη δυνατότητα αλλαγής της θέσης ενός αντικειμένου (child), σε σχέση με την αλλαγή της θέσης ενός αντικειμένου αναφοράς (parent).
- **RandomDirection2D**: Αλλάζει τυχαία την κατεύθυνση και την ταχύτητα των αντικειμένων, περιμένοντας ένα ορισμένο χρονικό διάστημα μεταξύ των αλλαγών.
- **RandomWalk2D**: Αλλάζει τυχαία την κατεύθυνση και την ταχύτητα των αντικειμένων, μέχρι να διανύσουν κάποια καθορισμένη απόσταση ή να παρέλθει κάποιος καθορισμένος χρόνος. Χαρακτηρίζεται και ως μοντέλο Brownian κίνησης.

- **RandomWaypointMobility**: Κάθε αντικείμενο κάνει παύση αρχικά, μέσω μιας τυχαίας μεταβλητής. Έπειτα διαλέγει ένα προορισμό (μέσω κάποιου position allocator) και κινείται προς αυτόν διαλέγοντας τυχαία κάποια σταθερή ταχύτητα. Μόλις φτάσει εκεί, επαναλαμβάνει την ίδια διαδικασία. Βρίσκει εφαρμογή σε δισδιάστατες αλλά και τρισδιάστατες εφαρμογές (αν και δεν υπάρχει ακόμη τρισδιάστατος position allocator).
- **SteadyStateRandomWaypointMobility**: Βασίζεται στο προηγούμενο μοντέλο, με τη διαφορά ότι οι μεταβλητές (παύσης, ταχύτητας, θέσης) ακολουθούν ομοιόμορφη κατανομή. Βρίσκει εφαρμογή σε δισδιάστατες εφαρμογές.
- **WaypointMobility**: Η κίνηση βασίζεται στην επίσκεψη του αντικειμένου σε μια σειρά σημείων (waypoints). Κάθε αντικείμενο ξεκινά με μηδενική ταχύτητα από την αρχική του θέση, κινείται με σταθερή ταχύτητα μεταξύ των σημείων και σταματάει την κίνηση του όταν φτάσει στο τελικό σημείο.

5.2.10 olsr module

Στο module υλοποιείται το πρωτόκολλο δρομολόγησης: *Optimized Link State Routing Protocol (OLSR)* [20] για δίκτυα MANET και γενικά για ασύρματα ad-hoc δίκτυα. Στο πρωτόκολλο, κάθε κόμβος διαλέγει μερικούς από τους γειτονικούς του ως: Multi-point Relays (MPRs). Οι MPRs είναι υπεύθυνοι για την προώθηση κίνησης που ελέγχει την κατάσταση του δικτύου (proactive protocol), κάτι που ελαχιστοποιεί το overhead της λειτουργίας του πρωτοκόλλου. Οι MPRs χρησιμοποιούνται για τον υπολογισμό των ποιο σύντομων διαδρομών για τη διεξαγωγή της επικοινωνίας μεταξύ των κόμβων. Το μοντέλο λειτουργεί με το πρωτόκολλο διαδικτύου: IPv4 και βασίζεται στην υλοποίηση του ns-2 (αν και παρουσιάζουν διαφορές).

5.2.11 propagation module

Το module χρησιμοποιείται για τη δημιουργία μοντέλων απώλειας και καθυστέρησης διάδοσης. Μεταξύ άλλων μοντέλων απώλειας διάδοσης που υποστηρίζονται, είναι και τα μοντέλα: **OkumuraHataPropagationLossModel**, **ItuR1411LosPropagationLossModel** και **ItuR1411NlosOverRooftopPropagationLossModel**, που βρίσκουν εφαρμογή και στο **buildings** module, καθώς και το: **Kun2600MhzPropagationLossModel**, που είναι εμπειρικό μοντέλο και βρίσκει εφαρμογή στην απώλεια διάδοσης κυμάτων (συχνότητας: $2600MHz$) σε αστικές περιοχές.

Μέχρι στιγμής, τα μοντέλα καθυστέρησης διάδοσης που υποστηρίζονται είναι:

- **RandomPropagationDelayModel**, για τυχαία καθυστέρηση.
- **ConstantSpeedPropagationDelayModel**, για σταθερή καθυστέρηση.

5.2.12 spectrum module

Το module χρησιμοποιείται για τη μοντελοποίηση του φάσματος των ασύρματων καναλιών επικοινωνίας. Το module παρέχει ένα framework που επιτρέπει τη δημιουργία φασματικών μοντέλων για ασύρματα κανάλια επικοινωνίας, καθώς και ένα interface για την επικοινωνία του ασύρματου καναλιού και του επιπέδου *PHY*. Αυτό δίνει τη δυνατότητα δημιουργίας Net Devices που υποστηρίζουν το φασματικό μοντέλο του καναλιού.

Με τη βοήθεια των παραπάνω, έχουν υλοποιηθεί τα μοντέλα καναλιών:

- **SimpleSpectrumChannel**
  Σε αυτό όλες οι συσκευές του καναλιού χρησιμοποιούν το ίδιο συχνοτικό μοντέλο.
- **MultiModelSpectrumChannel**
  Σε αυτό υπάρχει η δυνατότητα χρήσης διαφορετικών συχνοτικών μοντέλων από τις συσκευές.

Έχει υλοποιηθεί επίσης το μοντέλο απώλειας διάδοσης του *Friis*.
Ενώ στο επίπεδο *PHY*, έχουν υλοποιηθεί τα παρακάτω μοντέλα:

- **WaveformGenerator**: Είναι συσκευή που εκπέμπει ένα προκαθορισμένο σήμα, σε προκαθορισμένες χρονικές στιγμές. Δεν χρησιμοποιείται για τη μετάδοση δεδομένων, μπορεί όμως να χρησιμοποιηθεί για τη δημιουργία παρεμβολών σε άλλες ασύρματες συσκευές.
- **MicrowaveOven**: Αντίστοιχα, είναι μοντέλο για τη δημιουργία παρεμβολών από φούρνο μικροκυμάτων.
- **SpectrumAnalyzer**: Είναι συσκευή για τη μέτρηση της του φάσματος της ακτινοβολίας, σε κάποιο καθορισμένο σημείο στο χώρο.
- **OfdmSpectrumWifiPhy**: Είναι μοντέλο υπολογισμού σφαλμάτων μετάδοσης, που βασίζεται στον υπολογισμό του *ESNR (Estimated Signal to Noise Ration)*. Βρίσκει εφαρμογή σε κανάλια που χρησιμοποιούν τη διαμόρφωση *OFDM (Orthogonal frequency-division multiplexing)* του πρωτοκόλλου *IEEE 802.11g*.
- **ShannonSpectrumWifiPhy**: Είναι μοντέλο υπολογισμού σφαλμάτων μετάδοσης σε κανάλια που χρησιμοποιείται σχεδόν όλη η χωρητικότητά τους (πλησιάζουν το όριο του *Shannon*). Υπολογίζει μια σταθερή τιμή για το *SINR (Signal to Interference plus Noise Ratio)*, βάσει της χωρητικότητας του καναλιού. Η προσέγγιση αυτή, παρέχει ικανοποιητική ακρίβεια με χαμηλή υπολογιστική πολυπλοκότητα. Βρίσκει εφαρμογή σε κανάλια που χρησιμοποιούν τη διαμόρφωση *OFDM* του πρωτοκόλλου *IEEE 802.11g*.

5.2.13 uan module

Το module χρησιμοποιείται για την μοντελοποίηση υποβρύχιων ακουστικών δικτύων *(Underwater Acoustic Network - UAN)*. Ο σκοπός του είναι να παρέχει ένα αξιόπιστο και ρεαλιστικό εργαλείο στην προσομοίωση υποβρύχιων ασύρματων δικτύων, σε πραγματικές συνθήκες. Για το σκοπό αυτό, παρέχει ακριβή μοντελοποίηση του υποβρύχιου ακουστικού καναλιού, το μοντέλο ενός διαδεδομένου ακουστικού modem (*WHOI acoustic modem*), μερικά πρωτόκολλα του υπό-επιπέδου *MAC*, καθώς και δύο μοντέλα αυτόνομων υποβρύχιων οχημάτων (*Autonomous Underwater Vehicle - AUV*). Έτσι το module είναι χωρισμένο σε τέσσερα κύρια μέρη: *channel, PHY, MAC* και *AUV*.

Η προσομοίωση του καναλιού παρέχει μερικά μοντέλα διάδοσης που χρησιμοποιούνται συχνά στην πράξη. Κάθε μοντέλο διάδοσης βασίζεται σε ένα προφίλ καθυστέρησης ισχύος (*Power Delay Profile - PDP*) και σε πληροφορίες απώλειας διάδοσης (*pathloss information*) για να υπολογίσει την ισχύ του λαμβανόμενου σήματος. Μέχρι στιγμής, υποστηρίζονται τρία μοντέλα διάδοσης:

- **Ideal channel model**.
  Υποθέτει μηδενική απώλεια διάδοσης μέσα σε μία κυλινδρική περιοχή, ορισμένη από το χρήστη.

- **Thorp propagation**.
  Υπολογίζει την απώλεια διάδοσης, κάνοντας χρήση της προσέγγισης του Thorp. Βασίζεται σε αντίστοιχη υλοποίηση του ns-2.

- **Bellhop propagation model**.
  Βασίζεται στη χρήση δεδομένων διάδοσης, αποθηκευμένων σε βάσεις δεδομένων. Κάνει χρήση ενός αρχείου ρυθμίσεων, που ορίζει τη γεωγραφική θέση και τη λεπτομέρεια των δεδομένων που είναι επιθυμητά. Τα δεδομένα περιλαμβάνουν στοιχεία για το κανάλι όπως: το προφίλ της ταχύτητας του ήχου, η στάθμη της επιφάνειας της θάλασσας, το ανάγλυφο του πυθμένα της, καθώς και το βάθος του νερού. Κάνει χρήση ενός *Gaussian ray tracing* αλγόριθμου, που με τη χρήση πολλαπλών επαναλήψεων, υπολογίζει την απώλεια διάδοσης.

Το επίπεδο *PHY*, βασίζεται στη χρήση μοντέλων για τον υπολογισμό του λόγου του σήματος προς το θόρυβο *(SINR)* και του ρυθμού απώλειας πακέτων *(PER)*, που καθορίζουν και την επιτυχή λήψη των πακέτων. Για τα μοντέλα αυτά, έχουν δημιουργηθεί διαφορετικές υλοποιήσεις.

Όσον αφορά τα μοντέλα **PER**, οι διαθέσιμες υλοποιήσεις είναι:

- **Default (simple)**: Το οποίο ελέγχει το εισερχόμενο πακέτο ως προς ένα κατώφλι:
  - Αν το *SINR* είναι κάτω από το κατώφλι, υπάρχει λάθος.
  - Αλλιώς, δεν υπάρχει.
- **Micromodem FH-FSK**: Υπολογίζει την πιθανότητα σφάλματος, υποθέτοντας συνελικτικό κώδικα: ρυθμού $1/2$ και μήκος περιορισμού 9, μαζί με CRC (Cyclic Redundancy Check) έλεγχο με δυνατότητα διόρθωσης μέχρι 1 bit.

Ενώ όσον αφορά τα **SINR** μοντέλα:

- **Default**: Στο μοντέλο αυτό, χρησιμοποιείται η υπόθεση ότι όλη η εκπεμπόμενη ενέργεια λαμβάνεται από τον δέκτη, χωρίς την ύπαρξη διασυμβολικής παρεμβολής (Inter Symbol Interference - ISI). Η ισχύς κάθε λαμβανόμενου σήματος που παρεμβάλλεται, προστίθεται στο συνολικό θόρυβο του περιβάλλοντος.
- **WHOI Micromodem FH-FSK**: Το modem WHOI Micromodem λειτουργεί με τη μέθοδο: *FH-FSK (Frequency Hopping - Frequency Shift Keying)*, χρησιμοποιώντας ένα κοινό μοτίβο *"αναπήδησης" (hopping)* μεταξύ όλων των κόμβων. Στο μοντέλο, προσομοιώνεται η λειτουργία αυτή, υπολογίζοντας την λαμβανόμενη ισχύ μόνο κατά τη διάρκεια ενός συμβόλου. Η ενέργεια κάθε παρακείμενου σήματος που λαμβάνεται, θεωρείται δια συμβολική παρεμβολή και υπολογίζεται σαν πρόσθετος θόρυβος από το περιβάλλον. Σήματα που παρεμβάλλονται μέσα στο χρόνο ενός συμβόλου, υπολογίζονται επίσης σαν πρόσθετος θόρυβος.
- **Frequency filtered SINR**: Λειτουργεί όπως το *Default*, με τη διαφορά ότι λαμβάνει υπόψιν σαν παρεμβολή, μόνο όταν υπάρξει επικάλυψη στη συχνότητα των λαμβανόμενων πακέτων.

Συμπεριλαμβάνονται τρία μοντέλα υποβρύχιου **MAC**:

- **CW-MAC**: Χρησιμοποιεί ένα παράθυρο σύνδεσης με υποδοχές (όπως το: IEEE 802.11 DCF). Οι κόμβοι έχουν ένα σταθερό παράθυρο σύνδεσης, που μετριέται σε αριθμό υποδοχών. Στην περίπτωση που αντιληφθούν ότι το κανάλι χρησιμοποιείται, διαλέγουν τυχαία μια άλλη υποδοχή για να μεταδώσουν.
- **RC-MAC**: Είναι πρωτόκολλο που διαχωρίζει δυναμικά τη διαθέσιμη χωρητικότητα σε, κανάλι δεδομένων και κανάλι ελέγχου. Το πρωτόκολλο υποθέτει ότι όλη η κίνηση εξυπηρετείται από μία πύλη (gateway). Η παρούσα υλοποίηση, υποθέτει μία μοναδική πύλη και single-hop δίκτυο (όλοι οι κόμβοι είναι γειτονικοί).
- **Simple ALOHA**: Είναι το απλούστερο πρωτόκολλο. Οι κόμβοι μεταδίδουν αμέσως τα πακέτα που έχουν. Σε περίπτωση σύγκρουσης περιμένουν για ένα τυχαίο διάστημα και ξαναπροσπαθούν.

Το μοντέλο **AUV**, περιλαμβάνει μοντέλα που περιγράφουν την κίνηση και την κατανάλωση, αυτόνομων υποβρύχιων οχημάτων.

Όσον αφορά την κίνηση, δίνονται στο χρήστη οι δυνατότητες:

- Να προγραμματίσει το μονοπάτι της κίνησης του AUV.
- Να ελέγξει την ταχύτητα, το βάθος, την κατεύθυνση και την κλίση του AUV.
- Να δώσει εντολή στο AUV να αναδυθεί, ή να καταδυθεί σε καθορισμένο βάθος.

Μέχρι στιγμής έχουν υλοποιηθεί δύο μοντέλα AUV: το *Remous* και το *Seaglider*, που χρησιμοποιούν σταθερή ταχύτητα και τα δύο και διαθέτουν διαφορετικές επιχειρησιακές παραμέτρους, όπως: κατανάλωση, μέγιστο επιχειρησιακό βάθος, κ.α.

Όσον αφορά την κατανάλωση ενέργειας, δίνονται στο χρήση οι δυνατότητες:

- Να χρησιμοποιήσει ένα συγκεκριμένο προφίλ για το ακουστικό modem.
- Να χρησιμοποιήσει ένα μοντέλο κατανάλωσης ενέργειας.
- Να ανιχνεύσει (trace) την κατανάλωση των συστημάτων πλοήγησης, καθώς και του υποβρύχιου ακουστικού εξοπλισμού, μέσω του προφίλ του ακουστικού modem.

Για κάθε τύπο *AUV (Remous, Seaglider)*, έχει δημιουργηθεί και το αντίστοιχο μοντέλο κατανάλωσης. Το μοντέλο κατανάλωσης του ακουστικού modem, βασίζεται στο *WHOI modem*, που λειτουργεί σε τέσσερα επίπεδα κατανάλωσης, ανάλογα με τη κατάσταση που βρίσκεται (Tx, Rx, Idle, Sleep). Τέλος έχει υλοποιηθεί ένα γενικό μοντέλο *Li-Ion* μπαταριών, για χρήση σαν πηγή ενέργειας. Στο *Seaglider*, το μοντέλο κατανάλωσης είναι ακριβές (λόγω της διαθεσιμότητας σχετικών πληροφοριών), ενώ στο *Remous* όχι τόσο (λόγω έλλειψης σχετικών πληροφοριών).

5.2.14 wifi module

Στο module υλοποιείται ένα Net Device βασισμένο στο πρότυπο *IEEE 802.11*. Το σύνολο προτύπων: *IEEE 802.11* περιγράφει την υλοποίηση ασύρματων τοπικών δικτύων (*WLAN*). Το module παρέχει:

- Βασική υποστήριξη: **802.11 DCF (Distributed Coordination Function)**, με λειτουργίες: *infrastructure* και *ad-hoc*.
- Τα πρότυπα: **802.11{a,b,g}**, του φυσικού επιπέδου.
- Το πρότυπο: **802.11e**, με την επέκταση QoS που παρέχει *(Enhanced Distributed Channel Access - EDCA)*.
- Ποικιλία μοντέλων απώλειας διάδοσης όπως τα: **Nakagami, Rayleigh, Friis, LogDistance, FixedRss** και **Random**.

- Δύο μοντέλα καθυστέρησης διάδοσης. Ένα βασισμένο στην απόσταση και ένα τυχαίο (random).
- Ποικιλία αλγόριθμων ελέγχου του ρυθμού μετάδοσης όπως τα: **Aarf, Arf, Cara, Onoe, Rraa, ConstantRate** και **Minstrel**.
- Το πρότυπο: **802.11s** που υλοποιείται στο **mesh** module.

Η υλοποίηση χωρίζεται στα επίπεδα: **PHY, MAC low, MAC high** και **Rate control algorithms** (που χρησιμοποιούνται από τα μοντέλα του MAC low).

- Το επίπεδο **PHY** βασίζεται στο αντίστοιχο μοντέλο που είχε υλοποιηθεί για τον εξομοιωτή: *Yet Another Network Simulator (YANS)*. Ενώ είναι δυνατή η χρήση πολλαπλών Net Devices σε ένα κόμβο, δεν υπάρχει (προς το παρόν) υποστήριξη για διακαναλική παρεμβολή ή σύζευξη.
- Το επίπεδο **MAC low**, είναι οργανωμένο σε τρία μέρη:
  - *MacLow*, που διεκπεραιώνει τις συναλλαγές: *RTS/CTS/DATA/ACK*.
  - *DcfManager* και *EdcaTxopN*, που υλοποιούν τις λειτουργίες: *DCF* και *EDCAF*.
  - *DcaTxop*, *EdcaTxop*, που χειρίζονται τις ουρές αναμονής των πακέτων, τον κατακερματισμό τους και τις αναμεταδόσεις των πακέτων (όταν είναι απαραίτητες).
    * Το *DcaTxop*, χρησιμοποιείται από *MAC high* που δεν υποστηρίζουν QoS, ενώ για τη μετάδοση των πλαισίων κάνει χρήση του *DCF*.
    * Το *EdcaTxopN*, χρησιμοποιείται από *MAC high* που υποστηρίζουν QoS, ενώ εκτελεί και QoS λειτουργίες (όπως η *MSDU* του *802.11n*).
- Στο επίπεδο **MAC high**, υπάρχουν τρία μοντέλα:
  - *Access Point (AP)*.
  - *non-AP Station (STA)*
  - *STA* σε IBSS (Independent Basic Service), κοινώς γνωστού και ως ad-hoc δίκτυο.
- Οι **Rate control algorithms** (*αλγόριθμοι ελέγχου του ρυθμού μετάδοσης*) που έχουν υλοποιηθεί για το επίπεδο *MAC low*, είναι:
  OnoeWifiManager, IdealWifiManager, AarfcdWifiManager, AarfWifiManager, ArfWifiManager, AmrrWifiManager, ConstantRateWifiManager, MinstrelWifiManager, CaraWifiManager και *RraaWifiManager*

Τα μοντέλα μπορούν να λειτουργήσουν υποστηρίζοντας QoS (τύπου *802.11e* WMN), με τη ρύθμιση μιας παραμέτρου. Στην περίπτωση αυτή, η κίνηση χωρίζεται σε τέσσερις κατηγορίες (Access Categories - ACs):

- *AC_VO*, για κίνηση με ήχο.
- *AC_VI*, για κίνηση με video.
- *AC_BE*, για κίνηση best effort.
- *AC_BK*, για κίνηση παρασκηνίου.

5.2.15 wimax module

Στο module υλοποιείται ένα Net Device βασισμένο στο πρότυπο: *WiMAX*, που ανήκει στην οικογένεια: *IEEE 802.16*. Το module παρέχει:

- Ρεαλιστικό και επεκτάσιμο μοντέλο καναλιού και επίπεδο PHY.
- Ταξινόμηση πακέτων.
- Αποδοτικούς χρονοδρομολογητές ανερχόμενης και κατερχόμενης ζεύξης.
- Υποστήριξη: multicast και broadcast μετάδοσης.
- Δυνατότητα tracing των πακέτων.

Στο υπό-επίπεδο **MAC** παρέχονται δύο βασικοί τρόποι λειτουργίας:

- *Subscriber Station (SS)*
- *Base Station (BS)*

Για τις ζεύξεις έχουν υλοποιηθεί τρεις χρονοδρομολογητές:

- **SIMPLE**, απλός χρονοδρομολογητής FCFS (First Come First Served).
- **RTPS**: χρονοδρομολογητής rtPS.
- **MBQOS**: χρονοδρομολογητής για την ανερχόμενη κίνηση.

Επιπλέον παρέχονται:

- Δυνατότητα χρήσης υπαρχόντων μοντέλων απώλειας και καθυστέρησης διάδοσης, καθώς και των μοντέλων κίνησης (mobility module).
- Υποστήριξη Point-to-Multipoint (PMP) και WirelessMAN-OFDM στο επίπεδο PHY.
- Initial Ranging.
- Service Flow Initialization.
- Management Connection.
- Transport Initialization.
- Υποστήριξη UGS, rtPS, nrtPS, και BE συνδέσεων.

Ενώ προς το παρόν δεν παρέχονται:

- OFDMA PHY layer.
- Link adaptation.
- Mesh topologies.
- ARQ.
- ertPS connection.
- packet header suppression.

5.3 Συμπεράσματα

Όπως έγινε αντιληπτό, ο ns-3 προσπαθεί να καλύψει μια ευρεία κατηγορία ασύρματων επικοινωνιών και δικτύων. Πέρα από τη διάθεση συγκεκριμένων μοντέλων από κάθε module, δίνεται έμφαση (σε ορισμένα) στην παροχή ενός γενικότερου framework, για να είναι ευκολότερη η δημιουργία καινούργιων μοντέλων και αντίστοιχα επέκταση των δυνατοτήτων του εξομοιωτή, ώστε να προσαρμόζεται στις εκάστοτε ερευνητικές ανάγκες. Επίσης υπάρχει η δυνατότητα για το χρήστη, να κάνει συνδυασμένη χρήση των διαθέσιμων modules, κάτι που επιτρέπει πλήρη αξιοποίηση των δυνατοτήτων τους.

Ο ns-3 συνδυάζει ικανοποιητική ακρίβεια, χωρίς να υστερεί σε επιδόσεις. Το πλήθος των μοντέλων που υποστηρίζει για ποικιλία σεναρίων, καθώς και η παραμετροποίηση που προσφέρει στη χρήση του (καθώς και την επέκτασή του), τον καθιστούν ένα χρήσιμο εργαλείο σε πλήθος εφαρμογών, όπως και στην προσομοίωση ασυρμάτων δικτύων. Με τα μοντέλα διαγραμμάτων ακτινοβολίας κεραιών, τα μοντέλα του φάσματος της ακτινοβολίας των ασύρματων καναλιών, καθώς και τα μοντέλα απώλειας και καθυστέρησης διάδοσης της ακτινοβολίας σε διάφορα περιβάλλοντα, γίνεται δυνατή η προσομοίωση και μελέτη της απόδοσης ασύρματων-εναέριων επικοινωνιών. Τα μοντέλα πηγών ενέργειας και συσκευών κατανάλωσης, μαζί με τα μοντέλα περιγραφής της κίνησης, δίνουν τη δυνατότητα προσομοίωσης της κίνησης ασύρματων φορητών συσκευών, που τροφοδοτούνται αυτόνομα. Με τη βοήθεια των μοντέλων δρομολόγησης ασύρματων ad-hoc δικτύων, δίνεται η δυνατότητα δημιουργίας αντίστοιχων δικτύων, στα οποία μπορούν να συνδέονται και φορητές συσκευές που βρίσκονται σε κίνηση. Η παροχή μοντέλων για υποβρύχια ακουστικά δίκτυα, δίνει τη δυνατότητα στον ns-3, να χρησιμοποιηθεί για τη μελέτη συστημάτων σε περιβάλλον που είναι δύσκολη και συγχρόνως δαπανηρή η διεξαγωγή πειραμάτων σε πραγματική κλίμακα. Τέλος, η υποστήριξη διαδεδομένων προτύπων επικοινωνίας, όπως το *Wifi*, το *LTE* και το *WiMAX*, δίνει τη δυνατότητα δοκιμής και αξιολόγηση τους, σε ποικιλία σεναρίων χρήσης.

Η σελίδα έχει μείνει εσκεμμένα κενή

Κεφάλαιο 6

ΥΛΟΠΟΙΗΣΗ ΤΟΥ diffserv module

6.1 Εισαγωγή

Όπως ήδη έχει φανεί, οι δυνατότητες που παρέχονται από τα modules του ns-3 είναι πολλές. Πάντως, δεν υπάρχει ακόμη επίσημη υποστήριξη για την αρχιτεκτονική DiffServ. Αν και υπάρχει διαθέσιμη υλοποίηση του μηχανισμού Leaky Bucket [4], καθώς και μια υλοποίηση της αρχιτεκτονικής DiffServ [6], δεν έχει γίνει ακόμη ενσωμάτωση τους σε κάποια έκδοση του ns-3.

Ένας από τους στόχους της παρούσας διπλωματικής είναι η μελέτη και υλοποίηση *scheduling* αλγορίθμων όπως ο *MDRR*. Όπως έχει αναφερθεί, οι αλγόριθμοι χρονοδρομολόγησης εφαρμόζονται για το συντονισμό της σειράς εξόδου των πακέτων ενός δρομολογητή, που διατηρεί πολλαπλές ουρές αναμονής. Οι πολλαπλές ουρές αναμονής βρίσκουν εφαρμογή στην αρχιτεκτονική *DiffServ*, όπου εξυπηρετείται κάθε κλάση της σε διαφορετική ουρά. Έτσι, στα πλαίσια της διπλωματικής υλοποιήθηκε το **diffserv module**, που περιλαμβάνει μηχανισμούς της αρχιτεκτονικής *DiffServ*. Έχουν υλοποιηθεί μηχανισμοί για την ταξινόμηση των πακέτων (*Multi-Field* και *Behavior Aggregate*), τη δημιουργία SLA για ροές δεδομένων που το επιθυμούν, ποικιλία μηχανισμών μέτρησης της κίνησης, καθώς και μηχανισμού μαρκαρίσματος της κίνησης, ανάλογα με το αποτέλεσμα της μέτρησης. Τέλος, υλοποιήθηκε ο μηχανισμός *MDRR* που χρησιμοποιείται στη χρονοδρομολόγηση των ουρών αναμονής, που ορίζονται από την αρχιτεκτονική που υλοποιήθηκε.

Στο κεφάλαιο αυτό, γίνεται περιγραφή των στοιχείων που συνθέτουν το module που δημιουργήθηκε.

6.2 Δομή του module

Κάθε module του ns-3, βρίσκεται κάτω από το φάκελο *src/*. Για τη δημιουργία του, χρησιμοποιήθηκε ένα python script που παρέχεται με τον εξομοιωτή, για το σκοπό αυτό. Το script ονομάζεται *create-module.py* και βρίσκεται στον *src/* φάκελο. Κάνοντας χρήση templates, δημιουργεί τη δομή των φακέλων, καθώς και κάποια βασικά αρχεία, που χρειάζονται για την περαιτέρω εξέλιξη του module. Κατά την εκτέλεση του καλείται με ένα όρισμα, που είναι το επιθυμητό όνομα του module που θα δημιουργήσει. Ο ns-3 έχει υποστήριξη για την Python 2, έτσι και αυτό το script είναι γραμμένο σε αυτήν. Το λειτουργικό σύστημα που δημιουργήθηκε και δοκιμάστηκε το καινούργιο module είναι το *Arch Linux*, που χρησιμοποιεί το εκτελέσιμο: **python2** για να καλέσει το διερμηνέα της Python 2. Υποθέτοντας ότι βρισκόμαστε ήδη στο φάκελο όπου βρίσκεται η διανομή του ns-3, για τη δημιουργία του diffserv module εκτελέστηκαν οι παρακάτω εντολές:

```
cd ns-allinone-3.16/ns-3.16/src
python2 create-module.py diffserv
```

Έτσι δημιουργήθηκε ένας καινούργιος φάκελος μέσα στο φάκελο: **src/** με όνομα: **diffserv/**. Μέσα σε αυτόν δημιουργήθηκαν επίσης οι φάκελοι: **doc/, examples/, helper/, model/, test/**, καθώς και το αρχείο: **wscript**, που χρησιμοποιείται για τη μετάφραση του module από το waf.

Στο φάκελο **doc/** δημιουργήθηκε ένα template για τη συγγραφή documentation για το καινούργιο module. Στο φάκελο: **examples/** δημιουργήθηκε ένα template για την υλοποίηση παραδειγμάτων που χρησιμοποιούν το καινούργιο module. Στο φάκελο: **helper/** δημιουργήθηκαν templates για την υλοποίηση των helpers του module. Στο φάκελο: **model/** δημιουργήθηκαν templates για την υλοποίηση των μοντέλων του module. Τέλος στο φάκελο: **test/** δημιουργήθηκε ένα template για την υλοποίηση κώδικα δοκιμών για το module.

Οι μηχανισμοί της αρχιτεκτονικής DiffServ, υλοποιήθηκαν στο φάκελο: **model/**, ενώ στο φάκελο: **helper/** υλοποιήθηκαν helpers που διευκολύνουν την εγκατάσταση της αρχιτεκτονικής στις τοπολογίες που ορίζει ο χρήστης. Δημιουργήθηκε επίσης ο φάκελος: **utils/**, για τη φιλοξενία μοντέλων παραγωγής κίνησης, που βρίσκουν εφαρμογή στην δοκιμή και αξιολόγηση του module. Τέλος δημιουργήθηκε και ο φάκελος: **bindings/**, όπου σε αυτόν τοποθετούνται τα python bindings, εφόσον ενεργοποιηθούν κατά τη διάρκεια της μεταγλώττισης του κώδικα.

6.3 diffserv model

Στο φάκελο *models/*, δημιουργήθηκαν τα παρακάτω αρχεία:

- **diffserv-queue.{h,cc}**
- **diffserv.{h,cc}**
- **policy.{h,cc}**
- **mdrr.{h,cc}**

Σε κάθε ζεύγος αρχείων, υλοποιήθηκε και μία κλάση, που συνολικά συνθέτουν την αρχιτεκτονική DiffServ που υλοποιήθηκε. Ακολουθεί αναλυτική περιγραφή τους.

6.3.1 Κλάση: DiffServQueue

Η κλάση υλοποιήθηκε στα αρχεία *diffserv-queue.{h,cc}*. Είναι υπο-κλάση της *ns3::Queue* και είναι η κεντρική κλάση, που συντονίζει όλους τους μηχανισμούς της αρχιτεκτονικής. Όπως έχει αναφερθεί, κεντρικό ρόλο στη συγγραφή τοπολογιών για τον ns-3, παίζουν οι κόμβοι. Σε αυτούς ενσωματώνονται εφαρμογές που δημιουργούν την κίνηση στο δίκτυο, καθώς και Net Devices που συνδέουν τους κόμβους μεταξύ τους μέσω καναλιών, επιτρέποντας τους να επικοινωνούν. Σε κάθε Net Device, υπάρχει τουλάχιστον μία ουρά μετάδοσης (TxQueue). Υπάρχει επίσης η δυνατότητα αλλαγής της προεπιλεγμένης ουράς που χρησιμοποιεί κάποιο Net Device. Έτσι η δημιουργία της αρχιτεκτονικής σαν υπο-κλάση της *ns3::Queue*, δίνει τη δυνατότητα εύκολης ενσωμάτωσης της σε Net Devices κατά βούληση. Κάθε υπο-κλάση της *ns3::Queue*, κληρονομεί τρεις private virtual μεθόδους, που πρέπει να υλοποιήσει. Οι μέθοδοι είναι οι: *DoEnqueue, DoDequeue, DoPeek* και εκτελούνται αντίστοιχα κατά την είσοδο ενός πακέτου στην ουρά, την έξοδο του και όταν ζητείται το στοιχείο στην κορυφή της ουράς. Η ροή της εκτέλεσης των μηχανισμών της αρχιτεκτονικής DiffServ, υλοποιήθηκε στις μεθόδους: *DoEnqueue* και *DoDequeue*.

Στην κλάση ορίστηκε ένας τύπος enum με τις τιμές: *CORE* και *EDGE*. Με τη διατήρηση μιας private μεταβλητής του τύπου αυτού, έγινε δυνατή η λειτουργία της κλάσης με δύο τρόπους, ανάλογα με τη θέση τους στον DiffServ domain που ανήκουν. Κάνοντας χρήση του attribute system που παρέχει ο ns-3, γίνεται δυνατή η αλλαγή της παραμέτρου αυτής από το χρήστη, κατά τη διάρκεια συγγραφής μιας τοπολογίας που κάνει χρήση της αρχιτεκτονικής. Ορίζοντας τη μεταβλητή στην τιμή: *EDGE*, κατά την εκτέλεση της μεθόδου: *DoEnqueue*, η κλάση εκτελεί μηχανισμούς που δρουν στα όρια ενός DiffServ domain. Ενώ αντίστοιχα, με την τιμή: *CORE*, εκτελούνται μηχανισμοί που δρουν στο εσωτερικό του.

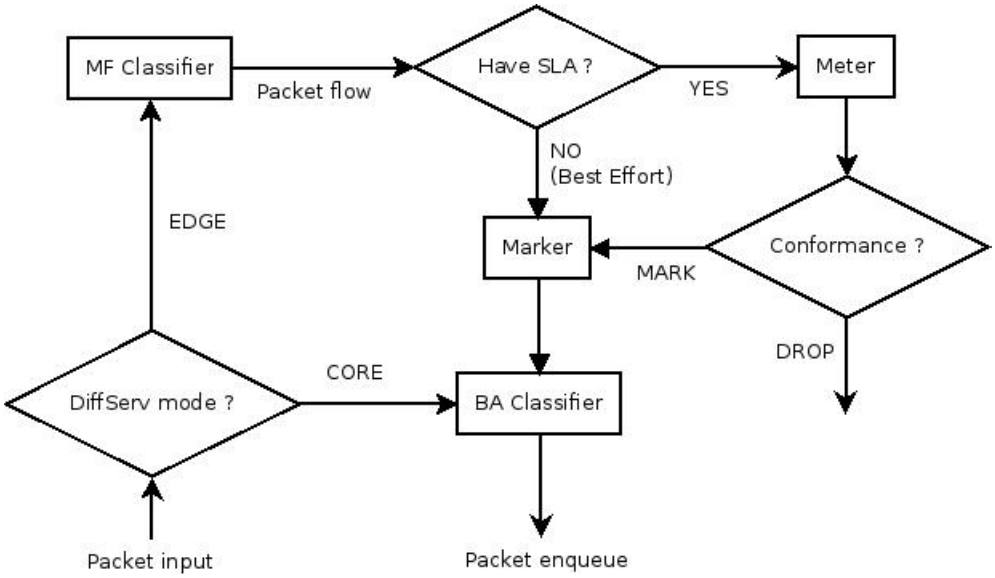
Κατά τη λειτουργία *EDGE*, εκτελείται πρώτα ο μηχανισμός Multi-Field ταξινόμησης, που είναι υλοποιημένος στη μέθοδο: *MfClassifier()*. Η μέθοδος δέχεται ως όρισμα ένα πακέτο (*ns3::Ptr<Packet>*) και επιστρέφει μια δομή (struct) με πληροφορίες που χαρακτηρίζουν τη ροή του πακέτου:

- IP διεύθυνση αποστολέα.
- IP διεύθυνση παραλήπτη.
- Αριθμός πρωτοκόλλου μεταφοράς.
- Αριθμός port αποστολέα.
- Αριθμός port παραλήπτη.

Οι πληροφορίες αυτές, παρέχονται ως όρισμα στη μέθοδο: *GetSla()*, που αναζητά την ύπαρξη κάποιας SLA, για τη ροή του πακέτου.

Κάθε SLA αποτελεί μία δομή (struct), που συμπεριλαμβάνει στοιχεία για τη ροή που αφορά η συγκεκριμένη SLA, καθώς και την πολιτική που πρέπει να ακολουθηθεί για τη ροή αυτή. Η πολιτική, ορίζεται ως δείκτης (τύπου *ns3::Ptr*) προς ένα στιγμιότυπο της κλάσης Policy που είναι υλοποιημένη στα αρχεία: *policy.{h,cc}*. Αναλυτική περιγραφή της κλάσης αυτής θα γίνει στη συνέχεια.

Στην περίπτωση ύπαρξης SLA για τη ροή που ανήκει το πακέτο, στο πακέτο αντιστοιχίζεται το πεδίο DSCP που ορίζεται από αυτήν. Αλλιώς, στο πακέτο αντιστοιχίζεται η επιλογή: *DscpDefault*, για να γίνει best-effort διαχείριση του πακέτου. Στη συνέχεια, εφόσον έχει βρεθεί κάποια SLA για τη ροή του πακέτου, γίνεται μέτρηση του προφίλ της κίνησης, βάση των κανόνων που περιγράφονται στην SLA. Το αποτέλεσμα της μέτρησης, μαζί με τους κανόνες που έχουν συμφωνηθεί για τη ροή, έχουν σαν αποτέλεσμα τη διατήρηση του πεδίου DSCP του πακέτου, εφόσον το πακέτο είναι εντός προφίλ, την αλλαγή του πεδίου ή ακόμη και την απόρριψη του, εφόσον το πακέτο είναι εκτός προφίλ. Στη συνέχεια εκτελείται ο μηχανισμός μαρκαρίσματος που υλοποιείται στη μέθοδο: *Marker()*. Η μέθοδος δέχεται δύο ορίσματα, ένα πακέτο και την τιμή DSCP που πρέπει να μαρκαριστεί το πακέτο. Η τιμή αυτή, προέκυψε από το μηχανισμό μέτρησης ή από το μηχανισμό ταξινόμησης του πακέτου (εάν παραλήφθηκε ο μηχανισμός μέτρησης). Τέλος εκτελείται ο μηχανισμός Behavior Aggregate ταξινόμησης, που είναι υλοποιημένος στη μέθοδο: *BaClassifier()*. Η μέθοδος δέχεται ως όρισμα ένα πακέτο, διαβάζει τη τιμή DSCP της IP επικεφαλίδας του και το προωθεί στην αντίστοιχη ουρά αναμονής που διατηρείται για κάθε κλάση εξυπηρέτησης. Συνολικά διατηρούνται 6 διαφορετικές ουρές: *EF, AF{1-4}, BE*, που αντιστοιχούν στις Per-Hop Behaviors που έχουν ήδη περιγραφεί (Ενότητα: 3.3). Στην περίπτωση της *CORE* λειτουργίας, εκτελείται μόνο ο μηχανισμός της Behaviour Aggregate ταξινόμησης. Έτσι είναι δυνατή η προσομοίωση της συμπεριφοράς των κόμβων ενός DiffServ, όπως αυτή ορίζεται από την αρχιτεκτονική. Η σειρά εκτέλεσης των μηχανισμών, φαίνεται σχηματικά στην εικόνα που ακολουθεί (Σχήμα: 6.1).



Σχήμα 6.1: Διάγραμμα ροής μηχανισμών της μεθόδου: DoEnqueue.

Κατά την εκτέλεση της μεθόδου: *DoDequeue*, χρησιμοποιείται ο αλγόριθμος MDRR που έχει υλοποιηθεί στην κλάση: *Mdrr* και αναλύεται παρακάτω. Οι ουρές αναμονής των κλάσεων εξυπηρέτησης, καθώς και οι μέθοδοι που υλοποιούν τους μηχανισμούς που αναφέρθηκαν, είναι υλοποιημένα στην κλάση: *DiffServ*. Έτσι, κατά την αρχικοποίηση της κλάσης: *DiffServQueue*, στον constructor της δημιουργείται ένα στιγμιότυπο της κλάσης: *DiffServ* και διατηρείται σε μια private μεταβλητή της κλάσης. Η ανάλυση της κλάσης: *DiffServ*, ακολουθεί παρακάτω.

6.3.2 Κλάση: DiffServ

Η κλάση υλοποιήθηκε στα αρχεία: *diffserv.{h,cc}*. Καθώς είναι υπο-κλάση της *ns3::Object*, κληρονομεί τις παρακάτω βασικές ιδιότητες: type και attribute system, object aggregation system και smart-pointer reference counting system. Όπως αναφέρθηκε, η κλάση *DiffServ* διατηρεί τις ουρές αναμονής των κλάσεων εξυπηρέτησης της αρχιτεκτονικής. Οι ουρές, είναι υλοποιημένες με την δομή *std::queue* που παρέχει η βασική βιβλιοθήκη της C++. Διατηρεί επίσης μια ακέραια μεταβλητή για κάθε ουρά, που ορίζει το μέγιστο αριθμό πακέτων που μπορεί να δεχθούν. Οι μεταβλητές αυτές μπορούν να τροποποιηθούν με τη χρήση του attribute system και έχουν αρχική τιμή: 100 για κάθε queue. Σε κάθε ουρά, υλοποιούνται μέθοδοι για την εισαγωγή (Enqueue) και εξαγωγή (Dequeue) πακέτων σε αυτή. Υλοποιείται επίσης και μια τρίτη μέθοδος, που ελέγχει εάν η ουρά είναι άδεια (Empty). Και οι τρεις μέθοδοι που αναφέρθηκαν, βρίσκουν εφαρμογή στη διαχείριση των ουρών έξω από τα όρια της κλάσης, όπως στην κλάση: *Mdrr* όπου υλοποιείται ο αντίστοιχος αλγόριθμος. Τέλος, στην κλάση *DiffServ*, υλοποιούνται οι μηχανισμοί ταξινόμησης, μαρκαρίσματος και μέτρησης της κίνησης, που αναφέρθηκαν προηγουμένως.

Ο μηχανισμός *Multi Field* ταξινόμησης που υλοποιείται στην μέθοδο: *MfClassifier()*, εξάγει τις πληροφορίες που βρίσκονται στις επικεφαλίδες των εισερχομένων πακέτων. Για κάθε εισερχόμενο πακέτο, δημιουργεί ένα αντίγραφό του και από αυτό, εξάγει διαδοχικά τις επικεφαλίδες που περιέχει. Η πρώτη επικεφαλίδα που συναντά είναι του επιπέδου της ζεύξης δεδομένων. Μέχρι στιγμής υπάρχει υποστήριξη του *point-to-point* πρωτοκόλλου *(ppp)*, ενώ στην περίπτωση ύπαρξης άλλης επικεφαλίδας, η συνάρτηση επιστρέφει. Το αποτέλεσμα της πρόωρης επιστροφής, είναι η αντιμετώπιση του πακέτου με πολιτική best-effort. Στην εξαγωγή των επικεφαλίδων ακολουθεί η επικεφαλίδα του IP πρωτοκόλλου, όπου από αυτή λαμβάνονται τα στοιχεία: IP διεύθυνση αποστολέα/παραλήπτη και αριθμός πρωτοκόλλου μεταφοράς που χρησιμοποιείται. Ανάλογα με τον αριθμό αυτό, εξάγεται η αντίστοιχη επικεφαλίδα (TCP ή UDP) και ανακτώνται τελικά: ο αριθμός port αποστολέα/παραλήπτη. Τα στοιχεία αυτά, μαζί με μία boolean μεταβλητή που δηλώνει την κατάσταση επιστροφής της συνάρτησης, οργανώνονται σε μια δομή (struct) και επιστρέφονται. Ο κωδικός ειδοποιεί τους υπόλοιπους μηχανισμούς για το αποτέλεσμα της ταξινόμησης όπως την ενδεχόμενη αποτυχία της, που αναφέρθηκε προηγουμένως.

Ο μηχανισμός Behaviour Aggregate ταξινόμησης, που υλοποιείται στη μέθοδο: *BaClassifier()*, εξάγει αντίστοιχα τις επικεφαλίδες ενός αντιγράφου του εισερχόμενου πακέτου, ελέγχοντας όμως μόνο το πεδίο DSCP της IP επικεφαλίδας και καλώντας την αντίστοιχη μέθοδο Enqueue, με την τιμή DSCP που βρέθηκε στο πακέτο.

Τέλος, ο μηχανισμός μαρκαρίσματος, που υλοποιείται στη μέθοδο: *Marker()*, εξάγει την επικεφαλίδα του επιπέδου ζεύξης δεδομένων (του πρωτοκόλλου ppp προς το παρόν), καθώς και την επικεφαλίδα του IP πρωτοκόλλου, τροποποιεί το πεδίο DSCP αυτής της IP επικεφαλίδας σύμφωνα με την τιμή που παρέχεται από το δεύτερο όρισμα της, επανεισάγει τις επικεφαλίδες αυτές στο πακέτο και το επιστρέφει με τροποποιημένο το πεδίο DSCP.

6.3.3 Κλάση: Policy

Η κλάση υλοποιήθηκε στα αρχεία: *policy.{h,cc}* και είναι υπο-κλάση της *ns3::Object*. Στην κλάση υλοποιούνται διάφοροι μηχανισμοί μέτρησης που μεταφέρθηκαν (port) από τον ns-2. Στην κλάση ορίζονται δύο βασικές δομές (structs): η Actions και η Metrics.

Η Actions, περιέχει μια μεταβλητή ενός enum τύπου που παίρνει τις τιμές: **Null, TSW2CM, TSW3CM, TokenBucket, srTCM, trTCM** και αντιστοιχούν στους μηχανισμούς μέτρησης που έχουν υλοποιηθεί. Η τιμή της μεταβλητής, καθορίζει τον μηχανισμό μέτρησης που χρησιμοποιείται από κάθε στιγμιότυπο της κλάσης. Ορίζονται επίσης τρεις μεταβλητές με ονόματα: Green, Yellow και Red, που αντιστοιχούν σε τρεις διαφορετικές ενέργειες, ανάλογα με το μηχανισμό μέτρησης που χρησιμοποιείται και του αποτελέσματος της μέτρησης. Κάθε μεταβλητή είναι δομή (struct) και περιέχει τις μεταβλητές: Mode και Dscp. Η μεταβλητή Mode, είναι με τη σειρά της τύπος enum με δυο δυνατές τιμές: Drop και Dscp και χρησιμοποιείται για να διαχωρίσει τις δύο δυνατές ενέργειες, απόρριψη του πακέτου ή αντιστοίχιση κάποιας DSCP τιμής σε αυτό. Η μεταβλητή Dscp, αποθηκεύει την τιμή DSPC, όταν η μεταβλητή Mode έχει την αντίστοιχη τιμή.

Η Metrics, περιέχει μεταβλητές που αποθηκεύουν τιμές από τις μετρικές που χρησιμοποιούν οι μηχανισμοί μέτρησης που έχουν υλοποιηθεί. Και για τις δύο δομές, διατηρούνται αντίστοιχες private μεταβλητές στην κλάση, που χρησιμοποιούνται για να ελέγχουν τη συμπεριφορά σε κάθε στιγμιότυπο της.

Η κύρια μέθοδος της κλάσης είναι η: *Meter()* και δέχεται ως όρισμα ένα πακέτο. Ανάλογα με το μηχανισμό μέτρησης που είναι ορισμένος στην αντίστοιχη private μεταβλητή, καλεί και την αντίστοιχη μέθοδο του μηχανισμού. Κάθε μηχανισμός μέτρησης, επιστρέφει μια μεταβλητή του enum τύπου: Conformance, που μπορεί να πάρει τις τιμές: Green, Red, Yellow, και δείχνει το αποτέλεσμα της μέτρησης. Οι μηχανισμοί μέτρησης που υλοποιήθηκαν περιγράφονται παρακάτω:

Null

Είναι μηχανισμός που θεωρεί όλα τα πακέτα ότι είναι εντός προφίλ, συνεπώς σε αυτά αντιστοιχίζεται η τιμή DSCP της ενέργειας: Green, παραμένοντας αμετάβλητη.

TSW2CM

Ο μηχανισμός ονομάζεται: *Time Sliding Window Two Color Marker* και είναι παραλλαγή του μηχανισμού: *TSW3CM* που ακολουθεί. Αποτελείται από δύο επιμέρους μηχανισμούς: τον Rate estimator που μετρά το προφίλ της κίνησης και το Marker που μαρκάρει τα πακέτα, ανάλογα με το αποτέλεσμα της μέτρησης. Στο μηχανισμό *TSW2CM*, τα πακέτα θεωρούνται ως εντός ή εκτός προφίλ. Ο μηχανισμός χρησιμοποιεί από τη δομή Metrics τις τιμές: *avgRate, winLen, CIR* και *arrivalTime*. Με την άφιξη ενός πακέτου, υπολογίζεται ο αριθμός των bytes που αντιστοιχούν στο τρέχον χρονικό παράθυρο: $bytesInTSW = avgRate * winLen$. Σε αυτό αθροίζεται το μέγεθος του πακέτου σε bytes (*PktSize*), χωρίς όμως να υπολογίζεται η επικεφαλίδα του πρωτοκόλλου ζεύξης δεδομένων (η οποία αφαιρείται κατά τον υπολογισμό): $newBytes = bytesInTSW + PktSize$.
Σε μία άλλη μεταβλητή λαμβάνεται το τρέχον δευτερόλεπτο της εξομοίωσης:
now = Simulator::Now().GetSeconds(),
υπολογίζεται εκ' νέου η μετρική:
$avgRate = \frac{newBytes}{(now - arrivalTime) + winLen}$
και ανανεώνεται η μεταβλητή: $arrivalTime = now$
Στη συνέχεια, λαμβάνεται η τιμή μιας τυχαίας μεταβλητής που ακολουθεί την ομοιόμορφη κατανομή (στο διάστημα $[0, 1]$): $x \in uniform(0, 1)$.

- Εάν: $avgRate > CIR$ και $x \leq (1 - (\frac{CIR}{avgRate}))$, το πακέτο θεωρείται εκτός προφίλ και επιστρέφεται η ενέργεια Red.
- Αλλιώς, το πακέτο θεωρείται εντός προφίλ και επιστρέφεται η ενέργεια Green.

TSW3CM

Ο μηχανισμός ονομάζεται: *A Time Sliding Window Three Colour Marker* [13]. Λειτουργεί όπως και ο προηγούμενος μηχανισμός, με τη διαφορά ότι υπολογίζει τρία επίπεδα συμμόρφωσης των πακέτων: Green, Yellow και Red. Οι μεταβλητές που χρησιμοποιούνται είναι οι ίδιες και ανανεώνονται σε κάθε μέτρηση με τον ίδιο τρόπο. Διατηρείται επίσης και η μεταβλητή: *PIR*. Υπολογίζοντας τη μεταβλητή: $rand = avgRate * (1.0 - x)$, όπου πάλι: $x \in uniform(0, 1)$.

- Εάν $rand > PIR$, το πακέτο θεωρείται εκτός προφίλ και επιστρέφεται η ενέργεια Red.
- Εάν: $rand > CIR$, το πακέτο θεωρείται εκτός προφίλ και επιστρέφεται η ενέργεια Yellow.
- Αλλιώς, το πακέτο θεωρείται εντός προφίλ και επιστρέφεται η ενέργεια Green.

TokenBucket

Ο μηχανισμός έχει αναφερθεί και προηγουμένως (Ενότητα: 3.7). Για τη λειτουργία του, χρησιμοποιεί τις μεταβλητές: *CIR, CBS, cBucket* και *arrivalTime*. Υπολογίζει το τρέχον δευτερόλεπτο της εξομοίωσης: $now = Simulator :: Now().GetSeconds()$, καθώς και τη μεταβλητή: $TokenBytes = CIR * (now - arrivalTime)$.

- Εάν $cBucket + TokenBytes \leq CBS$, τότε: $cBucket+ = TokenBytes$.
- Αλλιώς, $cBucket = CBS$.

Ανανεώνεται η μεταβλητή: $arrivalTime = now$ και αποθηκεύεται το μέγεθος του πακέτου σε bytes (χωρίς να λαμβάνεται υπόψιν η επικεφαλίδα του επιπέδου ζεύξης δεδομένων), στη μεταβλητή: *PktSize*.

- Εάν $(cBucket - PktSize) \geq 0$, το πακέτο θεωρείται εντός προφίλ, προστίθεται στον κάδο: $cBucket- = PktSize$ και επιστρέφεται η ενέργεια Green.
- Αλλιώς, το πακέτο θεωρείται εκτός προφίλ και επιστρέφεται η ενέργεια Red.

srTCM

Ο αλγόριθμος ονομάζεται: *Single Rate Three Color Marker* [11]. Για τη λειτουργία του, χρησιμοποιεί τις μεταβλητές: *CIR, CBS, cBucket, EBS, eBucket* και *arrivalTime*. Υπολογίζει το τρέχον δευτερόλεπτο της εξομοίωσης: $now = Simulator :: Now().GetSeconds()$, καθώς και τη μεταβλητή: $TokenBytes = CIR * (now - arrivalTime)$.

- Εάν $cBucket + TokenBytes = CBS$, τότε: cBucket += TokenBytes.
- Αλλιώς, $TokenBytes- = (CBS - cBucket)$ και $cBucket = CBS$.
  - Εάν $eBucket + TokenBytes \leq EBS$, τότε: $eBucket+ = TokenBytes$.
  - Αλλιώς, $eBucket = EBS$.

Ανανεώνεται η μεταβλητή: $arrivalTime = now$ και αποθηκεύεται το μέγεθος του πακέτου (χωρίς να λαμβάνεται υπόψιν η επικεφαλίδα του επιπέδου ζεύξης δεδομένων), στη μεταβλητή: *PktSize*.

- Εάν $(cBucket - PktSize) \leq 0$, το πακέτο θεωρείται εντός προφίλ: $cBucket- = PktSize$ και επιστρέφεται η ενέργεια: Green.
- Αλλιώς:
  - Εάν $(eBucket - PktSize) \geq 0$, το πακέτο θεωρείται εκτός προφίλ: $eBucket- = PktSize$ και επιστρέφεται η ενέργεια Yellow.
  - Αλλιώς, το πακέτο θεωρείται εκτός προφίλ και επιστρέφεται η ενέργεια Red.

trTCM

Ο αλγόριθμος ονομάζεται: *Two Rate Three Color Marker* [12]. Για τη λειτουργία του, χρησιμοποιεί τις μεταβλητές: *CIR, CBS, cBucket, PIR, PBS, pBucket* και *arrivalTime*. Υπολογίζει το τρέχον δευτερόλεπτο της εξομοίωσης: $now = Simulator :: Now().GetSeconds()$, καθώς και τη μεταβλητή: $TokenBytes = CIR * (now - arrivalTime)$.

- Εάν $(cBucket + TokenBytes) \leq CBS$, τότε: $cBucket+ = TokenBytes$.
- Αλλιώς, $cBucket = CBS$.

Ανανεώνεται η μεταβλητή: $TokenBytes = PIR * (now - arrivalTime)$.

- Εάν $(pBucket + TokenBytes) \leq PBS$, τότε: $pBucket+ = TokenBytes$.
- Αλλιώς, $pBucket = PBS$.

Ανανεώνεται η μεταβλητή: $arrivalTime = now$ και υπολογίζεται το μέγεθος του πακέτου (χωρίς να λαμβάνεται υπόψιν η επικεφαλίδα του επιπέδου ζεύξης δεδομένων), στη μεταβλητή: *PktSize*.

- Εάν $(pBucket - PktSize) < 0$, το πακέτο θεωρείται εκτός προφίλ και επιστρέφεται η ενέργεια Red.
- Αλλιώς:
  - Εάν $(cBucket - PktSize) < 0$, το πακέτο θεωρείται εκτός προφίλ: $pBucket- = PktSize$ και επιστρέφεται η ενέργεια Yellow.
  - Αλλιώς, το πακέτο θεωρείται εντός προφίλ: $cBucket- = PktSize$, $pBucket- = PktSize$ και επιστρέφεται η ενέργεια Green.

6.3.4 Κλάση: Mdrr

Η κλάση υλοποιήθηκε στα αρχεία: *mdrr.{h,cc}* και είναι υπο-κλάση της *ns3::Object*. Περιέχει μία δομή που περιλαμβάνει τις μεταβλητές: *QuantumValue, DeficitCounter* και *Weight* και χρησιμοποιούνται κατά την εξυπηρέτηση των ουρών. Οι τιμές αρχικοποιούνται στον constructor της κλάσης, κάνοντας χρήση του τύπου: $QuantumValue = MTU + ((Weight - 1) * 512)$, όπως έχει αναφερθεί προηγουμένως (Ενότητα: 3.9.8). Περιέχει επίσης έναν enum τύπο που πέρνει τις τιμές: *ALTERNATE* και *STRICT* και αντιστοιχούν στους δύο τρόπους λειτουργίας που ορίζονται για τον αλγόριθμο. Για κάθε τιμή του enum τύπου, υπάρχουν δύο μέθοδοι: *Schedule* και *RoundRobin*. Η *Schedule*, αναλαμβάνει την εξυπηρέτηση της ουράς που έχει σειρά από τον αλγόριθμο, ενώ η *RoundRobin* χρησιμοποιείται για να υπολογιστεί ποια ουρά έχει σειρά να εξυπηρετηθεί στον επόμενο κύκλο του αλγόριθμου.

Για την παρακολούθηση των αλλαγών σε κάθε ουρά, διατηρούνται δύο private μεταβλητές για την *ALTERNATE* λειτουργία και μία για την *STRICT*. Η *ALTERNATE* λειτουργία χρειάζεται δύο μεταβλητές, λόγω της εκ' περιτροπής εξυπηρέτησης της ουράς υψηλής προτεραιότητας και των υπολοίπων. Σε αυτές διατηρούνται: η ουρά που έχει σειρά στον επόμενο γύρο, καθώς και η ουρά που είχε σειρά στον αμέσως προηγούμενο. Οι μεταβλητές είναι τύπου enum, που ορίζεται στο αρχείο: *diffserv.h*, έξω από τα όρια της κλάσης *DiffServ* (εντός όμως του ns3 namespace).

Οι μέθοδοι, κάνουν χρήση των public μεθόδων της κλάσης: *DiffServ*, για να αποκτήσουν πρόσβαση στις ουρές αναμονής που διατηρεί. Το κύριο loop της εξυπηρέτησης κάθε ουράς, παρουσιάζεται παρακάτω (Listing: 6.1). Σε *ALTERNATE* mode, το loop εφαρμόζεται σε κάθε ουρά, ενώ σε *STRICT* mode, ελέγχεται πρώτα η ουρά υψηλής προτεραιότητας (*EF*) και αν είναι άδεια, εφαρμόζεται το παραπάνω loop στις υπόλοιπες.

Οι παραπάνω μέθοδοι, συντονίζονται από τη μέθοδο: *MdrrSchedule*, που ελέγχει μια private μεταβλητή τύπου enum, για το αν βρίσκεται σε *ALTERNATE* ή σε *STRICT* mode. Καθώς χρησιμοποιείται η τιμή του MTU για τον υπολογισμό της τιμής του *Quantum*, διατηρείται μια private μεταβλητή με την τιμή του. Οι παραπάνω μεταβλητές, ελέγχονται μέσω του attribute system που κληρονομεί η κλάση, από την κλάση Object.

Listing 6.1: MDRR dequeue loop.

```
/* Code example of main MDRR dequeue loop. */
// struct xData: holds dequeue data for queue: x
// and initialized: xData.Ready = false; xData.DeficitCounter = 0;
// xData.QuantumValue = MTU + ((xData.Weight - 1) * 512);
Ptr<Packet> p = 0; // Smart pointer for packet to be dequeued
if (!AllQueuesAreEmpty ()) // At least one queue is not empty
  {
    bool done = false;
    while (!done) // Loop through all available queues
      {
        if (!done && QueueInTurn == EF) // Check EF queue
          {
            if (!EmptyEF ())
              {
                if (!EFData.Ready) // Initialize queue for this round
                  {
                    EFData.DeficitCounter += EFData.QuantumValue;
                    EFData.Ready = true;
                  }
                if (EFData.DeficitCounter > 0) // Dequeue allowed
                  {
                    p = EFDequeue ();
                    EFData.DeficitCounter -= p->GetSize ();
                    done = true;
                  }
                else // Dequeue not allowed
                  EFData.Ready = false;
              }
            else // EF queue empty
              {
                m_EFData.DeficitCounter = 0;
                m_EFData.Ready = false;
              }
          }
        if (...) // Check rest of the queues
        RoundRobin (); // Update: QueueInTurn variable
      }
  }
return p; // finally: p points to a packet or 0
```

6.4 diffserv helper

Στο φάκελο *helper/*, δημιουργήθηκαν τα παρακάτω αρχεία:

- **diffserv-helper.{h,cc}**
- **sla-helper.{h,cc}**

Στα αρχεία: *diffserv-helper.{h,cc}*, υλοποιήθηκε η κλάση: *DiffServPointToPointHelper*. Η κλάση είναι μια τροποποιημένη εκδοχή της: *PointToPointHelper*, που βρίσκεται στο φάκελο *helper* του *point-to-point* module. Η κλάση δημιουργήθηκε για τη διευκόλυνση της εγκατάστασης των μοντέλων του diffserv module που δημιουργήθηκαν, σε point-to-point δίκτυα. Σε αυτήν προστέθηκαν οι μέθοδοι: *SetMode(), SetSla(), SetMdrrMode(), SetMtu()* και *SetWeights()*, για τη ρύθμιση των παραμέτρων της αρχιτεκτονικής DiffServ, ενώ τροποποιήθηκε η κύρια μέθοδος: *Install()*, ώστε τα Net Devices που δημιουργεί, να χρησιμοποιούν την κλάση *DiffServQueue* για *TxQueue*.

Στα αρχεία: *sla-helper.{h,cc}*, υλοποιήθηκε η κλάση: *SlaHelper*, που είναι υπο-κλάση της *ns3::Object*. Η κλάση δημιουργήθηκε με σκοπό τη διευκόλυνση της δημιουργίας SLA για τις ροές. Περιλαμβάνει τη μέθοδο: *SetFlow()*, για τη ρύθμιση των παραμέτρων της ροής και τις μεθόδους: *SetType()* και *SetMetrics()* για τη ρύθμιση του αλγόριθμου μέτρησης του προφίλ της κίνησης και των παραμέτρων λειτουργίας του. Περιλαμβάνει επίσης τις μεθόδους: *SetGreen(), SetYellow()* και *SetRed()*, για τη ρύθμιση των ενεργειών που πρέπει να ληφθούν ανάλογα με το αποτέλεσμα της μέτρησης. Τέλος, περιλαμβάνει τη μέθοδο: *Install()*, που εγκαθιστά την SLA που δημιουργήθηκε στον κόμβο που δέχεται ως όρισμα.

6.5 diffserv utils

Στο φάκελο *utils/*, δημιουργήθηκαν τα αρχεία: **traffic-helper.{h,cc}**, που υλοποιήθηκαν σε αυτά οι κλάσεις: *VoIpHelper*, για την παραγωγή *VoIP* κίνησης, καθώς και η κλάση: *VideoTraceHelper* για την παραγωγή *UDP* κίνησης με χρήση video traces. Οι κλάσεις αυτές, είναι υπο-κλάσεις της Object και παρέχουν κοινές μεθόδους:

- **SetDuration**, για τον ορισμό της διάρκειας που θα λειτουργήσει η εφαρμογή.
- **SetDestination**, για τον ορισμό της διεύθυνσης προορισμού που θα εκπέμψει η εφαρμογή.
- **Install**, για την εγκατάσταση της εφαρμογής σε κάποιο κόμβο.

Στο κεφάλαιο που ακολουθεί, γίνεται αναλυτικότερη περιγραφή τους, καθώς και αξιολόγηση της αρχιτεκτονικής DiffServ που υλοποιήθηκε, μέσα από μια σειρά πειραμάτων και μετρήσεων.

Κεφάλαιο 7

ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ diffserv module

7.1 Εισαγωγή

Η αρχιτεκτονική DiffServ, δημιουργήθηκε με σκοπό την καλύτερη διαχείριση των περιορισμένων πόρων των δικτύων, ώστε να μπορούν να εξυπηρετούν αποτελεσματικά και εφαρμογές ευαίσθητες στην καθυστέρηση, το jitter και άλλες μετρικές που επηρεάζουν την απόδοσή τους. Όπως αναφέρθηκε ήδη, η αρχιτεκτονική, ορίζει μια σειρά μηχανισμών που δρουν στις διερχόμενες ροές των πακέτων. Το diffserv module που υλοποιήθηκε και περιγράφηκε στο προηγούμενο κεφάλαιο, περιλαμβάνει τέτοιους μηχανισμούς.

Στο κεφάλαιο αυτό, διεξάγεται μια σειρά πειραμάτων σε μία τοπολογία δικτύου που δημιουργήθηκε με τον ns-3. Στο δίκτυο, δημιουργήθηκαν εφαρμογές VoIP και Video trace, που παράγουν κίνηση υψηλής προτεραιότητας (*foreground*), καθώς και εφαρμογές που προσομοιώνουν την υπόλοιπη κίνηση του Internet (*background*). Αρχικά, χωρίς τη χρήση των μηχανισμών, δημιουργήθηκε κίνηση στο δίκτυο, ικανή να προκαλέσει συμφόρηση και μελετήθηκε η επίδραση της στην απόδοση των εγκατεστημένων εφαρμογών. Έπειτα, χρησιμοποιήθηκαν οι μηχανισμοί του diffserv module και μελετήθηκε η επίδραση τους στην απόδοση των εφαρμογών, κάτω από τις ίδιες συνθήκες φόρτου, συμπεραίνοντας έτσι, την αλλαγή στην απόδοση του δικτύου που προκαλεί η εφαρμογή των μηχανισμών.

7.2 Παραγωγή κίνησης

Τα μοντέλα παραγωγής της κίνησης, βρίσκονται στο φάκελο: **utils/** του diffserv module. Τα μοντέλα παραγωγής της **foreground** κίνησης, βρίσκονται στα αρχεία: *traffic-helper.{h,cc}*. Σε αυτά, υλοποιήθηκαν οι κλάσεις: *VoIpHelper* και *VideoTraceHelper*.

7.2.1 VoIP model

Η κλάση **VoIpHelper**, υλοποιεί ένα μοντέλο VoIP μετάδοσης, μέσω μίας OnOff εφαρμογής. Οι παράμετροι της εφαρμογής, προέρχονται από σχετική δημοσίευση [3]. Όπως αναφέρεται εκεί, χρησιμοποιείται εκθετική κατανομή για τον καθορισμό του χρόνου μετάδοσης. Η εφαρμογή πρέπει να παρουσιάζει μέση διάρκεια μετάδοσης (On) 1.004 sec και μέση διάρκεια παύσης (Off) 1.587 sec. Ο ns-3, παρέχει τυχαίες μεταβλητές που ακολουθούν την εκθετική κατανομή και μπορούν να χρησιμοποιηθούν για τη ρύθμιση των OnOff περιόδων. Για τη χρήση τους πρέπει να υπολογίσουμε τη μέση τιμή (mean) που αντιστοιχεί σε κάθε περίοδο, βάση του μέσου χρόνου μετάδοσης που είναι επιθυμητός (median).
Για την περίοδο **On**:

- Η μέση διάρκειά της πρέπει να είναι: $Mean = 1.004sec$
- καθώς: $Median = \lambda^{-1} * \ln 2$
- συνεπάγεται: $1.004 = \lambda^{-1} * \ln 2 => \lambda = 0.6903856380079136$
- Άρα: $Mean = \lambda^{-1} => Mean = 1.4484658210525192$
- Δηλαδή: $Mean \simeq 1.448$

Για την περίοδο **Off**:

- Η μέση διάρκειά της πρέπει να είναι: $Mean = 1.587sec$
- καθώς: $Median = \lambda^{-1} * \ln 2$
- συνεπάγεται: $1.587 = \lambda^{-1} * \ln 2 => \lambda = 0.43676570923752067$
- Άρα: $Mean = \lambda^{-1} => Mean = 2.289557029890785$
- Δηλαδή: $Mean \simeq 2.29$

Ο ρυθμός μετάδοσης της εφαρμογής καθώς και το πλήθος των δεδομένων που περιέχει κάθε πακέτο που παράγεται (payload), μπορούν να ρυθμιστούν με τη χρήση attributes. Η προεπιλεγμένη τιμή του ρυθμού μετάδοσης είναι: $80Kbps$ και των δεδομένων: $160bytes$. Ο πίνακας που ακολουθεί συνοψίζει τα χαρακτηριστικά τους:

OnOff VoIP μοντέλο				
	Χρόνος On	Χρόνος Off	Ρυθμός μετάδοσης	Φορτίο
Κατανομή	Εκθετική	Εκθετική	80 Kbps	160 bytes
Μέση τιμή	1.448	2.29		
Μέση διάρκεια (sec)	1.004	1.587		

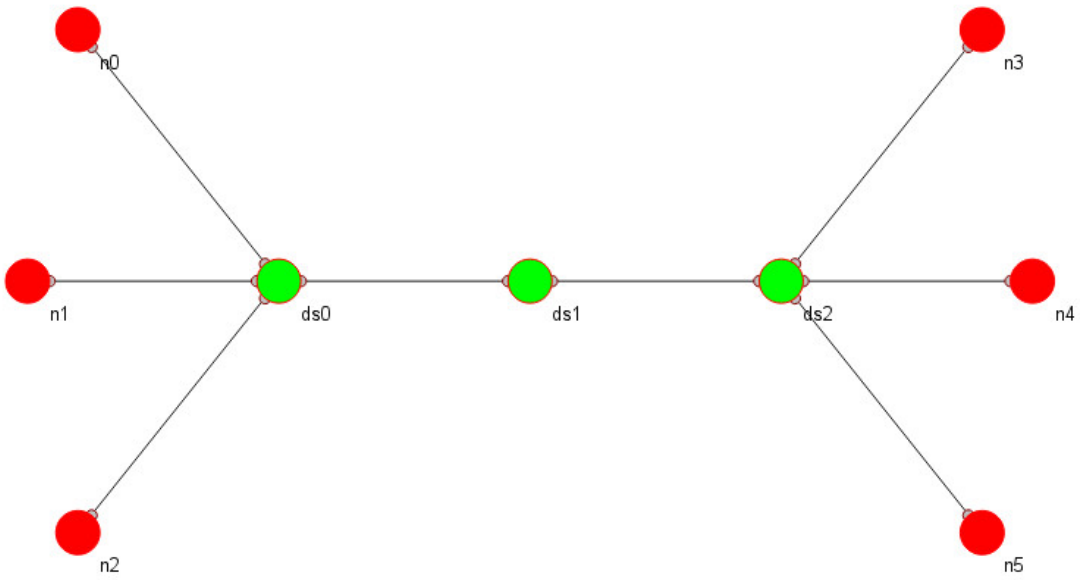
Πίνακας 7.1: Παράμετροι OnOff VoIP μοντέλου.

7.2.2 Video streaming model

Η κλάση **VideoTraceHelper**, χρησιμοποιεί την κλάση: *ns3::UdpTraceClient* που έχει τη δυνατότητα να διαβάζει traces από αρχεία video και αν τα μεταδίδει κάνοντας χρήση του UDP πρωτοκόλλου, προσομοιώνοντας video streaming μεταδόσεις που πραγματοποιούνται στα δίκτυα. Η κλάση μπορεί δεχθεί σαν παράμετρο, το trace αρχείο για να μεταδώσει. Στα πειράματα που ακολουθούν, έγινε χρήση του trace: *video_traces/Terse_Jurassic.dat*, που προέρχεται από μία MPEG 4 έκδοση της ταινίας: *Jurassic Park I*. Σύμφωνα με το αρχείο: *video_traces/Terse_Jurassic.dat*, το trace είναι διάρκειας: $3599880ms$, που αποτελείται από frames με μέγεθος που κυμαίνεται από: 72 bytes μέχρι και 16745 bytes. Με τη βοήθεια του script: *scripts/traces-stats.py*, υπολογίστηκε το μέσο μέγεθος των frames σε: 3831 bytes. Τέλος, ο μέσος ρυθμός μετάδοσης (bit rate) στο αρχείο είναι: $748.35Kbps$ ενώ παρουσιάζει μέγιστη έκρηξη: $3.2Mbps$

7.3 Τοπολογία πειραμάτων

Για την αξιολόγηση του **diffserv module** που υλοποιήθηκε, δημιουργήθηκε μία *dumbbell point-to-point* τοπολογία στο αρχείο: *diffserv-simulator.cc* που βρίσκεται στο φάκελο: *scratch/* του ns-3 (Σχήμα: 7.1). Η τοπολογία περιλαμβάνει, ένα DiffServ domain που δημιουργούν οι κόμβοι: **ds0, ds1, ds2**. Το domain, χρησιμοποιούν οι κόμβοι: **n0, n1, n2** και **n3, n4, n5** για να επικοινωνούν μεταξύ τους.



Σχήμα 7.1: Αναπαράσταση της τοπολογίας των πειραμάτων από τον NetAnim.

Οι παράμετροι των συνδέσμων (*data rate*, *delay*), μπορούν να ρυθμιστούν μέσω attributes. Οι προεπιλεγμένες τιμές τους, φαίνονται στον παρακάτω πίνακα:

Σύνδεσμος		IPv4 base address	Data rate (Mbps)	Delay (ms)
n0	ds0	10.1.1.0	2	10
n1	ds0	10.1.2.0	2	10
n2	ds0	10.1.3.0	2	10
ds0	ds1	10.2.1.0	20	20
ds1	ds2	10.2.2.0	20	20
ds2	n3	10.3.1.0	2	10
ds2	n4	10.3.2.0	2	10
ds2	n5	10.3.3.0	2	10

Πίνακας 7.2: Παράμετροι τοπολογίας δοκιμών.

Η τοπολογία δημιουργήθηκε με σκοπό την εγκατάσταση εφαρμογών (όπως αυτές που περιγράφηκαν στην προηγούμενη ενότητα) στους κόμβους: n{0-2} και την επικοινωνία τους με τους κόμβους n{3-5}, μέσω των κόμβων του domain: ds{0-2}. Η ύπαρξη ενός μοναδικού link στο εσωτερικό του domain, δημιουργεί bottleneck στην προώθηση της κίνησης, κάτι που μπορεί (υπό προϋποθέσεις) να προκαλέσει συμφόρηση στους κόμβους εισόδου του domain.

Σε κάθε κόμβο, ανατέθηκε ένα σύνολο εφαρμογών. Σε όλους τους κόμβους μετάδοσης (n{0-2}), μπορούν να ενεργοποιηθούν VoIP εφαρμογές. Ενώ στον κόμβο: n1, μπορούν να ενεργοποιηθούν video stream εφαρμογές. Το πλήθος των εφαρμογών, μπορεί να ρυθμιστεί δυναμικά μέσω των attributes. Για κάθε τύπο εφαρμογής, υπάρχει επίσης αντίστοιχο flag για τη χρήση του στην τοπολογία κατά την εκτέλεση. Συγκεντρωτικά:

- Ο κόμβος n0, εκπέμπει στον n3, χρησιμοποιώντας:
  - (3) VoIP εφαρμογές, με τις προεπιλεγμένες παραμέτρους (Πίνακας: 7.1).
- Ο κόμβος n1, εκπέμπει στον n4, χρησιμοποιώντας:
  - (2) VoIP εφαρμογές, με τις προεπιλεγμένες παραμέτρους.
  - (1) video streaming εφαρμογή, με το προεπιλεγμένο video trace και μέγιστο μέγεθος πακέτου: $522bytes$.
- Ο κόμβος n2, εκπέμπει στον n5, χρησιμοποιώντας:
  - (1) VoIP εφαρμογή, με τις προεπιλεγμένες παραμέτρους.

Σε κάθε αντιδιαμετρικό κόμβο, δημιουργήθηκαν οι αντίστοιχοι sink servers με τη βοήθεια της κλάσης: *ns3::PacketSinkHelper*. Στις VoIP εφαρμογές, ανατέθηκε το port 1000 και στις video streaming το port 2000.

7.4 Διεξαγωγή πειραμάτων

Για την εκτέλεση της τοπολογίας, πρέπει πρώτα να γίνει μεταγλώττιση του ns-3. Η μεταγλώττιση μπορεί να γίνει εύκολα, με τη βοήθεια του αρχείου *download.py* που βρίσκεται στο φάκελο *ns-allinone-3.16/*, με τις εντολές που φαίνονται παρακάτω:

```
cd ns-allinone-3.16
python2 build.py
```

Η εντολή θα δημιουργήσει την debug έκδοση του εξομοιωτή, περιλαμβάνοντας κάθε στοιχείο του ns-3 που μπορούσε να μεταγλωττιστεί από το σύστημα που την εκτέλεσε. Αν για κάποιο στοιχείο υπάρχει έλλειψη στις εξαρτήσεις του, απενεργοποιείται αυτόματα. Ο ns-3 παρέχει τρία προφίλ μεταγλώττισης: *debug, release* και *optimized*. Η αλλαγή του προφίλ μεταγλώττισης κάνοντας χρήση του αρχείου: *download.py*, μπορεί να γίνει όπως δείχνουν οι επόμενες εντολές:

```
cd ns-allinone-3.16
python2 build.py -- -d optimized
```

Μόλις ο εξομοιωτής είναι έτοιμος για χρήση, η εκτέλεση της τοπολογίας μπορεί να γίνει με τι εντολές:

```
cd ns-allinone-3.16/ns-3.16
python2 waf --run "scratch/diffserv-simulator"
```

με αποτέλεσμα να εκτελεστεί η τοπολογία με τις προεπιλεγμένες ρυθμίσεις της.

Στη τοπολογία που δημιουργήθηκε, έγινε χρήση του συστήματος των attributes που διαθέτει ο ns-3, κάτι που προσφέρει ευελιξία στη ρύθμιση των παραμέτρων της εξομοίωσης. Τα διαθέσιμα attributes, μπορούν να εμφανιστούν με την εντολή:

```
cd ns-allinone-3.16/ns-3.16
python2 waf --run "scratch/diffserv-simulator␣--PrintHelp"
```

Για τη διευκόλυνση της διεξαγωγής των πειραμάτων, δημιουργήθηκε το bash script: *scripts/diffserv-tests.sh*, στο οποίο οργανώθηκαν οι παράμετροι κάθε πειράματος με ξεχωριστό όνομα, κάνοντας δυνατή την επανάληψή του με την εντολή:

```
./scripts/diffserv-tests.sh test1 duration
```

Το οποίο θα εκτελέσει το πείραμα με κωδικό: *test1* για τα δευτερόλεπτα που ορίζει η μεταβλητή: *duration*. Ο κωδικός: *all*, εκτελεί διαδοχικά όλα τα τεστ που δημιουργήθηκαν.

Για την ανάλυσή του, δημιουργήθηκε το script: *flowmonitor.py* σε Python 2. Το script, σαρώνει το XML αρχείο που δημιουργεί το Flow Monitor, και συλλέγει τις μετρήσεις που περιέχει.

Μέσω αυτών, υπολογίζει τις μετρικές:

- $TxDuration = timeLastRxPacket - timeFirstTxPacket$
- $\overline{MeanDelay} = \frac{delaySum}{rxPackets}$
- $\overline{MeanJitter} = \frac{jitterSum}{rxPackets}$
- $\overline{MeanTxPacketSize} = \frac{txBytes}{txPackets}$
- $\overline{MeanRxPacketSize} = \frac{rxBytes}{rxPackets}$
- $\overline{MeanTxBitRate} = \frac{8*txBytes}{(timeLastTxPacket-timeFirstTxPacket)*10^{-9}}$
- $\overline{MeanRxBitRate} = \frac{8*rxBytes}{(timeLastRxPacket-timeFirstRxPacket)*10^{-9}}$
- $\overline{MeanHopCount} = 1 + \frac{timesForwarded}{rxPackets}$
- $throughput = \frac{8*rxBytes}{LastRxPacket-FirstTxPacket}$
- $PacketDeliveryRatio = \frac{rxPackets}{txPackets}$
- $PacketLossRatio = \frac{lostPackets}{txPackets}$

Η εκτέλεση του script με την παράμετρο: *-i* ή *–input* εκτυπώνει στην κονσόλα τα αποτελέσματα κάθε ροής. Ενώ με τη χρήση της παραμέτρου: *-l* ή *–latex* σε ένα φάκελο προορισμού, δημιουργεί το αρχείο *Tests.tex* στον ίδιο φάκελο, σαρώνοντας κάθε XML αρχείο που βρίσκει και μετατρέποντας τα δεδομένα για χρήση με το περιβάλλον του LaTeX. Η λειτουργία υλοποιήθηκε για την ταχύτερη εισαγωγή των αποτελεσμάτων στο παρόν κείμενο.

Οι πίνακες με τις μετρήσεις των πειραμάτων, βρίσκονται στο Παράρτημα A'.

7.4.1 Πείραμα 1

Στο πείραμα αυτό, δοκιμάστηκε η απόδοση των **VoIP** εφαρμογών μέσα στο δίκτυο, χωρίς τη χρήση της αρχιτεκτονικής DiffServ. Οι εφαρμογές ενεργοποιήθηκαν, κάνοντας χρήση της επιλογής: *–VoipFlag=1* του script. Το flag των video streaming εφαρμογών, έμεινε απενεργοποιημένο (από προεπιλογή). Το πλήθος των VoIP εφαρμογών, έμεινε στην προεπιλεγμένη τιμή σε κάθε κόμβο, έτσι δημιουργήθηκαν: 3 στον κόμβο n0, 2 στον κόμβο n1 και 1 στον n2. Η εντολή για την εκτέλεση του πειράματος είναι:

```
cd ns-allinone-3.16/ns-3.16
python2 waf --cwd=../../../traces
--run "scratch/diffserv-simulator␣--VoipFlag=1
--SimName=Test1␣--SimDuration=600"
```

Η εντολή: *–cwd* ανακατευθύνει τα αρχεία που δημιουργεί η εξομοίωση στο φάκελο: *traces/*. Το πείραμα έτρεξε για **600 δευτερόλεπτα** (επιλογή: *–SimDuration*) και αποθήκευσε στο φάκελο *traces/* το XML αρχείο καταγραφής: *Test1.xml*, με στοιχεία που συνέλεξε το *Flow Monitor* (που είναι ενεργοποιημένο στο script από προεπιλογή). Το προφίλ του πειράματος, αποθηκεύτηκε στο bash script: *diffserv-tests.sh* με το όνομα: **test1** και μπορεί να εκτελεστεί με την εντολή: *./diffserv-tests.sh test1 600* (όπου 600, η διάρκεια του πειράματος).

Όλοι οι κόμβοι συνδέονται με PointToPoint Net Devices και διαθέτουν από προεπιλογή, Tx queues μεγέθους 100 bytes. Εκτελώντας το *flowmonitor.py* με όρισμα: *-i Test1.xml*, προβάλλονται στην κονσόλα τα αποτελέσματα του πειράματος.

Τα αποτελέσματα του πειράματος, παρουσιάζονται στους πίνακες: [A'.1 - A'.6]. Οι πίνακες: A'.4, A'.5 και A'.6, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n0. Οι πίνακες: A'.2 και A'.3, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n1. Και ο πίνακας: A'.1, αφορά την κίνηση που δημιουργήθηκε από τον κόμβο: n2.

Όπως φαίνεται από τα δεδομένα των VoIP εφαρμογών:

- Ο κόμβος n0, εμφάνισε:
  - Μέγιστη διαμεταγωγή: 36.706 Kbps.
  - Μέση καθυστέρηση: $\approx 61.6ms$.
  - Μέγιστο jitter: 1.275 μs.
- Ο κόμβος n1, εμφάνισε:
  - Μέγιστη διαμεταγωγή: 34.759 Kbps.
  - Μέση καθυστέρηση: $61.6ms$.
  - Μέγιστο jitter: 0.383 μs.
- Ο κόμβος n2, εμφάνισε:
  - Μέγιστη διαμεταγωγή: 33.504 Kbps.
  - Μέση καθυστέρηση: $61.672ms$.
  - Μέγιστο jitter: 0.018 μs.

Παρατηρούμε ότι η απόδοση των εφαρμογών, κυμάνθηκε στα ίδια επίπεδα, εκτός από το jitter του κόμβου: n0, που εμφανίστηκε αυξημένο, λόγω της ύπαρξη περισσότερων VoIP εφαρμογών σε αυτόν. Από τις μετρικές: PDR και PLR, φαίνεται ότι δεν υπήρξε καμία απώλεια πακέτων.

7.4.2 Πείραμα 2

Στο πείραμα αυτό, έγινε αύξηση της κίνησης στο δίκτυο, ενεργοποιώντας και τη **video streaming** εφαρμογή. Αυτό έγινε προσθέτοντας την επιλογή: *–VtFlag=1* στην εκτέλεση της τοπολογίας. Το προφίλ της δοκιμής, αποθηκεύτηκε με το όνομα **test2**.

Τα αποτελέσματα του πειράματος, φαίνονται στους πίνακες: [A'.7 - A'.13]. Οι πίνακες: A'.11 A'.12 και A'.13, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n0. Οι πίνακες: A'.7, A'.9 και A'.10, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n1. Και ο πίνακας: A'.8, αφορά την κίνηση που δημιουργήθηκε από τον κόμβο: n2.

Όπως φαίνεται από τα δεδομένα για τις VoIP εφαρμογές:

- Ο κόμβος n0, εμφάνισε:
  - Μέγιστη διαμεταγωγή: 36.706 Kbps.
  - Μέση καθυστέρηση: $\approx 61.7ms$.
  - Μέγιστο jitter: 26.671 μs.
- Ο κόμβος n1, εμφάνισε:
  - Μέγιστη διαμεταγωγή: 34.759 Kbps.
  - Μέση καθυστέρηση: $\approx 75.3ms$.
  - Μέγιστο jitter: 11554.798 μs.
- Ο κόμβος n2, εμφάνισε:
  - Μέγιστη διαμεταγωγή: 33.504 Kbps.
  - Μέση καθυστέρηση: $\approx 61.683ms$.
  - Μέγιστο jitter: 21.069 μs

Παρατηρούμε ότι σε όλους τους κόμβους, η μέγιστη διαμεταγωγή και η μέση καθυστέρηση, παρέμεινε στα ίδια επίπεδα. Το jitter από την άλλη, αυξήθηκε σημαντικά στους κόμβους: n0 και n2 (φτάνοντας περίπου στα ίδια επίπεδα), ενώ στον κόμβο: n1 που φιλοξενούσε και την εφαρμογή video stream, το jitter αυξήθηκε υπερβολικά.

Η εφαρμογή video stream του κόμβου n1 εμφάνισε:

- Μέγιστη διαμεταγωγή: 790.267 Kbps.
- Μέση καθυστέρηση: $93.744ms$.
- Μέγιστο jitter: 3961.032 μs.

Ο αρκετά υψηλότερος ρυθμός μετάδοση της video stream εφαρμογής, αυξάνει το φόρτο στο δίκτυο με τα πακέτα που παράγει. Αυτό έχει σαν αποτέλεσμα να μειώνεται συνολικά η απόδοση των VoIP εφαρμογών. Η VoIP εφαρμογή του ίδιου κόμβου (n1), που μοιράζεται το κανάλι επικοινωνίας προς τον κόμβο: ds0, έχει επηρεαστεί ποιο πολύ.

Αρχίζει έτσι να γίνεται αντιληπτή η αρνητική επίδραση, που έχει στις VoIP εφαρμογές, η αύξηση της κίνησης στο δίκτυο.

7.4.3 Πείραμα 3

Στο πείραμα αυτό, ενεργοποιήθηκε η αρχιτεκτονική DiffServ, προσθέτοντας την επιλογή: *–DS=1* και διατηρώντας τις υπόλοιπες ρυθμίσεις. Δεν χρησιμοποιήθηκε η επιλογή: *–Policy*, έτσι δεν δημιουργήθηκαν SLA για τις εφαρμογές. Οι μηχανισμοί ταξινόμησης, προώθησαν κάθε πακέτο με Best Effort πολιτική, εξυπηρετώντας όλη την κίνηση από το αντίστοιχο queue (BE). Το πείραμα έγινε, για να διαπιστωθεί το overhead που προσθέτουν οι μηχανισμοί του module, στην διερχόμενη κίνηση. Το προφίλ του πειράματος, αποθηκεύτηκε με το όνομα: **test3**.

Τα αποτελέσματα του πειράματος, φαίνονται στους πίνακες: [Α'.14 - Α'.20]. Οι πίνακες: Α'.18, Α'.19 και Α'.20, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n0. Οι πίνακες: Α'.14, Α'.16 και Α'.17, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n1. Και ο πίνακας: Α'.15, αφορά την κίνηση που δημιουργήθηκε από τον κόμβο: n2.

Παρατηρώντας τα δεδομένα, βλέπουμε ότι οι μετρικές των εφαρμογών κυμαίνονται στα ίδια επίπεδα με το προηγούμενο πείραμα, κάτι που σημαίνει ότι οι μηχανισμοί ταξινόμησης, μαρκαρίσματος και χρονοδρομολόγησης που υλοποιήθηκαν, δεν προσθέτουν σημαντικό overhead στην προώθηση της κίνησης, ώστε να γίνει αντιληπτό.

7.4.4 Πείραμα 4

Στο πείραμα αυτό, ενεργοποιήθηκε η επιλογή: *–Policy* που τέθηκε στην τιμή 1. Η τιμή της μεταβλητής, ενεργοποιεί και το αντίστοιχο προφίλ αστυνόμευσης. Στο προφίλ 1: ενεργοποιήθηκε η δημιουργία SLA για τις VoIP εφαρμογές. Χρησιμοποιήθηκε το πεδίο *DSCP: EF* και ο μηχανισμός μέτρησης: *Null*, που θεωρεί κάθε πακέτο εντός προφίλ. Έτσι κάθε VoIP πακέτο προωθήθηκε με την αρχική PHB (EF). Αντίστοιχα για την εφαρμογή video streaming, δεν δημιουργήθηκε κάποια SLA και αυτή προωθήθηκε με πολιτική best effort. Τέλος, με χρήση της επιλογής: *–MdrrMode=STRICT*, ο *MDRR* ρυθμίστηκε στην αντίστοιχη λειτουργία.

Τα αποτελέσματα του πειράματος, φαίνονται στους πίνακες: [Α'.21 - Α'.27]. Οι πίνακες: Α'.25, Α'.26 και Α'.27, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n0. Οι πίνακες: Α'.21, Α'.23 και Α'.24, αφορούν την κίνηση που δημιουργήθηκε από τον κόμβο: n1. Και ο πίνακας: Α'.22, αφορά την κίνηση που δημιουργήθηκε από τον κόμβο: n2.

Παρατηρώντας τα δεδομένα, οι εφαρμογές του κόμβου: n1, δεν παρουσίασαν καμία αλλαγή. Στις VoIP εφαρμογές των κόμβων: n{0,2}, υπήρξε μικρή μείωση του jitter σε σχέση με το προηγούμενο πείραμα, όπως φαίνεται και στον πίνακα που ακολουθεί:

	jitter (μsec)		
Flow ID	Πείραμα 3	Πείραμα 4	Διαφορά
2	21.069	21.041	-0.028
5	21.403	21.495	-0.092
6	26.671	26.63	-0.041
7	25.213	25.192	-0.021

Πίνακας 7.3: Βελτίωση του jitter με χρήση του MDRR STRICT.

Παρατηρούμε ότι, παρόλο που οι VoIP εφαρμογές εξυπηρετήθηκαν από την ουρά υψηλής προτεραιότητας και ο MDRR τους έδινε πάντα προτεραιότητα (καθώς λειτουργούσε σε STRICT mode), δεν βελτιώθηκε η απόδοσή τους σημαντικά. Ο φόρτος που παράγει η εφαρμογή video streaming, εξακολουθεί να επιβαρύνει το δίκτυο σημαντικά. Έτσι, συμπεραίνουμε ότι μόνο με την ταξινόμηση της κίνησης και τη χρονοδρομολόγηση, δεν μπορεί να εξασφαλιστεί η απόδοση των VoIP εφαρμογών.

7.4.5 Πείραμα 5

Με τα αποτελέσματα του προηγούμενου πειράματος, στο πείραμα αυτό τέθηκε η επιλογή: *–Policy* στην τιμή 2. Στο προφίλ αυτό, δημιουργήθηκε SLA για την video streaming εφαρμογή, και ενεργοποιήθηκε ο μηχανισμός Token Bucket για τη μέτρηση της κίνησης. Ο αλγόριθμος, ρυθμίστηκε με CIR = 200000 (bits) και CBS = 552 (bytes), που είναι το μέγιστο μέγεθος πακέτου της εφαρμογής, αφιερώθηκε δηλαδή κάδος βάθους 1. Η πολιτική για την εφαρμογή ρυθμίστηκε στην AF11 PHB για τα πακέτα εντός προφίλ και στην απόρριψη των πακέτων εκτός. Ενώ η SLA των VoIP εφαρμογών, παρέμεινε σταθερή (EF PHB, Null). Εξετάζοντας τα δεδομένα του πειράματος στους πίνακες: [A'.28 - A'.34], βλέπουμε ότι:

- Στην εφαρμογή video streaming:
  - Η διαμεταγωγή μειώθηκε στα: $413.873Kbps$.
  - Η μέση καθυστέρηση παρέμεινε στα ίδια επίπεδα.
  - Το jitter αυξήθηκε κατά: 3401.319 μsec.
  - Ενώ εμφανίστηκε απώλεια πακέτων σε ποσοστό: 47.7%.
- Στις VoIP εφαρμογές:
  - Η διαμεταγωγή παρέμεινε στα ίδια επίπεδα.
  - Η μέση καθυστέρηση βελτιώθηκε ελάχιστα σε κάθε εφαρμογή.
  - Το jitter βελτιώθηκε ελάχιστα στις εφαρμογές του κόμβου: n1.
    Ενώ βελτιώθηκε σημαντικά στις εφαρμογές των υπόλοιπων κόμβων, όπου μειώθηκε περίπου στο μισό.

Συμπεραίνουμε έτσι ότι, η απόρριψη των μισών περίπου πακέτων της video stream εφαρμογής από τον κόμβο: ds0, λόγω του Token Bucket που ενεργοποιήθηκε, μείωσε την κίνηση στο υπόλοιπο δίκτυο, επηρεάζοντας θετικά την απόδοση των VoIP εφαρμογών. Οι VoIP εφαρμογές του κόμβου n1, ευνοήθηκαν πολύ λιγότερο, καθώς εξακολουθούν να μοιράζονται το ίδιο κανάλι με τη video stream εφαρμογή, που το απασχολεί με δικά της πακέτα σε πολύ υψηλότερους ρυθμούς.

Φαίνεται έτσι, ότι η εφαρμογή κάποιου μηχανισμού μέτρησης και αστυνόμευσης, στην κίνηση της video streaming εφαρμογής, βελτιώνει την απόδοση των υπολοίπων εφαρμογών. Αυτό συμβαίνει, καθώς μόνο τα μισά από τα πακέτα της εφαρμογής, που παραλήφθηκαν από τον κόμβο: ds0, εξυπηρετήθηκαν από το υπόλοιπο δίκτυο. Η υψηλή απώλεια πακέτων που παρουσιάστηκε με την αστυνόμευση της κίνησης, δείχνει την ανάγκη της προσεκτικότερης ρύθμισης των παραμέτρων του δικτύου. Για το σκοπό αυτό, εκτελέστηκε το πείραμα που ακολουθεί.

7.4.6 Πείραμα 6

Στο τελευταίο αυτό πείραμα, έγινε χρήση του MDRR σε ALTERNATE mode, κάνοντας χρήση της επιλογής: *–MdrrMode=ALTERNATE*. Άλλαξαν επίσης τα βάρη των ουρών: EF (10)και AF1(5), ενώ στις υπόλοιπες έμειναν προεπιλεγμένα (1). Γνωρίζοντας ότι το μέγιστο πακέτο της VoIP εφαρμογής είναι: 190 bytes, ενώ το μέγιστο της video streaming εφαρμογής: 552 bytes, θέσαμε το MTU στο δίκτυο σε: 552 bytes, με την επιλογή: *–MTU=552*. Η επιλογή αυτή, μείωσε την ελάχιστη τιμή του quantum στον MDRR (552 για βάρος 1). Έγινε, με σκοπό να γίνεται ποιο συχνή αλλαγή στην εξυπηρέτηση των ουρών, σε μια προσπάθεια να βελτιωθεί η απόδοση και των δύο τύπων εφαρμογών ταυτόχρονα. Για την αστυνόμευση της video streaming εφαρμογής, χρησιμοποιήθηκε ο μηχανισμός αστυνόμευσης: *TSW3CM*, που ρυθμίστηκε με CIR = 100000 bits και PIR = 150000 bits. Ο μηχανισμός προσφέρει τρία επίπεδα συμμόρφωσης της κίνησης, έτσι οι ενέργειες που τέθηκαν σε αυτά είναι: η AF11 PHB αρχικά (green), η AF21 σαν πρώτο επίπεδο μη συμμόρφωσης (yellow) και απόρριψη σαν δεύτερο (red).

Τα δεδομένα του πειράματος φαίνονται στους πίνακες: [A'.35 - A'.41]. Από αυτά, παρατηρούμε για την εφαρμογή video streaming, αύξηση της διαμεταγωγής κατά: $\approx 75Kbps$, ελάχιστη μείωση της μέσης καθυστέρηση (0.1 ms) και μείωση του jitter κατά: ≈ 1152 μsec.

Η διαφορά της απόδοσης των VoIP εφαρμογών σε σχέση με το προηγούμενο πείραμα, παρουσιάζεται στον παρακάτω πίνακα:

Flow ID	Throughput (Kbps)	$\overline{Delay}$ (ms)	$\overline{jitter}$ (μsec)
2	0	-0.002	-2.247.
3	0	-0.16	-98,8
4	0	-0.178	-11.45
5	0	0	-0,578
6	0	-0.002	-2.442
7	0	-0,002	-3,377

Πίνακας 7.4: Βελτίωση της απόδοσης των εφαρμογών με χρήση του MDRR ALTERNATE.

Παρατηρούμε έτσι μια αύξηση στην απόδοση όλων των εφαρμογών, κάτι που μας δείχνει τη σημασία της προσεκτικής ρύθμισης των παραμέτρων, στους μηχανισμούς του δικτύου.

7.5 Συμπεράσματα

Στα πειράματα που εκτελέστηκαν, έγινε μελέτη της απόδοσης των VoIP εφαρμογών, σε συνθήκες χαμηλού φόρτου. Έπειτα δημιουργήθηκε υψηλός φόρτος στο δίκτυο, από μία video streaming εφαρμογή, με υψηλό ρυθμό μετάδοσης. Στη σειρά, ενεργοποιήθηκαν μηχανισμοί του diffserv module που δημιουργήθηκε και έγινε προσπάθεια ρύθμισης των παραμέτρων του, ώστε να βελτιωθεί η απόδοση των VoIP εφαρμογών κάτω από τις ίδιες συνθήκες φόρτου. Γενικά, οι μηχανισμοί DiffServ μπορούν να βελτιώσουν την απόδοση των εφαρμογών που εξυπηρετούν. Απαιτούν όμως προσεκτική ρύθμιση των παραμέτρων τους για να το επιτύχουν, όπως η ρύθμιση του MTU στο δίκτυο, και η επιλογή του τρόπου λειτουργίας του MDRR. Επιπλέον, η χρήση των μηχανισμών αστυνόμευσης της κίνησης, έδειξε ότι βοηθούν στη βελτίωση της απόδοσης των VoIP εφαρμογών, επηρεάζοντας αρνητικά όμως την απόδοση των εφαρμογών που αστυνομεύουν, εμφανίζοντας υψηλό ποσοστό στην απώλεια πακέτων. Για την αποφυγή της απώλειας πακέτων, μπορεί να γίνει χρήση κάποιου μηχανισμού μορφοποίησης της κίνησης, για τη video stream εφαρμογή, ώστε να μειώνει το ρυθμό αποστολή της στο υπόλοιπο δίκτυο, χωρίς όμως να τα απορρίπτει απευθείας. Η χρήση του μηχανισμού *Leaky Bucket* στην προκειμένη περίπτωση δίνει αυτή τη δυνατότητα. Ενώ έγινε προσπάθεια ενσωμάτωσης του μηχανισμού στο module, δεν στέφθηκε δυστυχώς με επιτυχία και εγκαταλείφθηκε.

Η σελίδα έχει μείνει εσκεμμένα κενή

Κεφάλαιο 8

ΕΠΙΛΟΓΟΣ

Στην παρούσα διπλωματική εργασία, έγινε μια ανασκόπηση του τρόπου λειτουργίας των σύγχρονων IP δικτύων και της ανάγκης μερικών εφαρμογών που εξυπηρετούν, για κάποια ποιότητα υπηρεσίας. Δόθηκε έμφαση στην αρχιτεκτονική DiffServ και παρουσιάστηκε η λειτουργία των μηχανισμών που την απαρτίζουν. Στη συνέχεια, μελετήθηκε ο εξομοιωτής δικτύων: ns-3 και παρουσιάστηκαν οι δυνατότητες που προσφέρει στα ασύρματα δίκτυα. Καθώς ο εξομοιωτής δεν είχε υποστήριξη για την αρχιτεκτονική DiffServ, δημιουργήθηκε ένα καινούργιο module σε αυτόν, στο οποίο υλοποιήθηκαν οι μηχανισμοί ταξινόμησης και μαρκαρίσματος της κίνησης, καθώς και ο μηχανισμός χρονοδρομολόγισης MDRR. Επίσης, μεταφέρθηκαν οι μηχανισμοί μέτρησης και αστυνόμευσης της κίνησης από τον ns-2. Για τη δοκιμή των μηχανισμών του module, δημιουργήθηκε μία τοπολογία, η οποία αξιοποιώντας το attribute system του ns-3, είχε τη δυνατότητα παραμετροποίησης της συμπεριφοράς της κατά την εκτέλεση. Έτσι, δημιουργήθηκαν προφίλ επιλογών που οργανώθηκαν σε ένα bash script. Η τοπολογία, κατέγραφε τη συμπεριφορά των προσομοιώσεων, κάνοντας χρήση του Flow monitor, ενός module που παρέχει ο ns-3 για το σκοπό αυτό. Τέλος, δημιουργήθηκε ένα python script, για την επεξεργασία των αρχείων καταγραφής. Για την εκτέλεση των πειραμάτων, δημιουργήθηκαν μοντέλα VoIP και video streaming εφαρμογών. Μέσα από μια σειρά πειραμάτων που πραγματοποιήθηκαν, φάνηκε η αρνητική επίδραση που μπορεί να έχει στις VoIP εφαρμογές, ο αυξημένος φόρτος στο δίκτυο. Κάνοντας χρήση των μηχανισμών DiffServ που υλοποιήθηκαν, έγινε προσπάθεια ρύθμισης των παραμέτρων της για τη βελτίωση της απόδοσής τους.

Η σελίδα έχει μείνει εσκεμμένα κενή

Παράρτημα Α΄

Μετρήσεις Πειραμάτων

FlowID: 1	
SrcAddress	10.1.3.1
DestAddress	10.3.3.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	596.763 sec
Throughput	33.504 Kbps
$\overline{Txbitrate}$	33.508 Kbps
$\overline{Rxbitrate}$	33.508 Kbps
$\overline{delay}$	61.672 ms
$\overline{jitter}$	0.018 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.1: Test1, Flow:1.

FlowID: 2	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	594.91 sec
Throughput	34.759 Kbps
$\overline{Txbitrate}$	34.763 Kbps
$\overline{Rxbitrate}$	34.763 Kbps
$\overline{delay}$	61.678 ms
$\overline{jitter}$	0.383 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.2: Test1, Flow:2.

FlowID: 3	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	593.844 sec
Throughput	33.412 Kbps
$\overline{Txbitrate}$	33.415 Kbps
$\overline{Rxbitrate}$	33.415 Kbps
$\overline{delay}$	61.681 ms
$\overline{jitter}$	0.341 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.3: Test1, Flow:3.

FlowID: 4	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	591.641 sec
Throughput	36.706 Kbps
$\overline{Txbitrate}$	36.71 Kbps
$\overline{Rxbitrate}$	36.71 Kbps
$\overline{delay}$	61.689 ms
$\overline{jitter}$	0.518 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.4: Test1, Flow:4.

FlowID: 5	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	1000
TxDuration	583.837 sec
Throughput	33.443 Kbps
$\overline{Txbitrate}$	33.447 Kbps
$\overline{Rxbitrate}$	33.447 Kbps
$\overline{delay}$	61.694 ms
$\overline{jitter}$	0.789 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.5: Test1, Flow:5.

FlowID: 6	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	584.596 sec
Throughput	31.986 Kbps
$\overline{Txbitrate}$	31.989 Kbps
$\overline{Rxbitrate}$	31.989 Kbps
$\overline{delay}$	61.691 ms
$\overline{jitter}$	1.275 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.6: Test1, Flow:6.

FlowID: 1	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	2000
TxDuration	596.974 sec
Throughput	790.267 Kbps
$\overline{Txbitrate}$	790.652 Kbps
$\overline{Rxbitrate}$	790.353 Kbps
$\overline{delay}$	93.744 ms
$\overline{jitter}$	3961.032 μsec
$\overline{Txpacketsize}$	516.461 bytes
$\overline{Rxpacketsize}$	516.459 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.7: Test2, Flow:1.

FlowID: 2	
SrcAddress	10.1.3.1
DestAddress	10.3.3.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	596.763 sec
Throughput	33.504 Kbps
$\overline{Txbitrate}$	33.508 Kbps
$\overline{Rxbitrate}$	33.508 Kbps
$\overline{delay}$	61.683 ms
$\overline{jitter}$	21.069 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.8: Test2, Flow:2.

FlowID: 3	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	594.91 sec
Throughput	34.759 Kbps
$\overline{Txbitrate}$	34.763 Kbps
$\overline{Rxbitrate}$	34.763 Kbps
$\overline{delay}$	75.391 ms
$\overline{jitter}$	11424.14 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.9: Test2, Flow:3.

FlowID: 4	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	593.844 sec
Throughput	33.412 Kbps
$\overline{Txbitrate}$	33.415 Kbps
$\overline{Rxbitrate}$	33.415 Kbps
$\overline{delay}$	75.376 ms
$\overline{jitter}$	11554.798 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.10: Test2, Flow:4.

FlowID: 5	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	591.641 sec
Throughput	36.706 Kbps
$\overline{Txbitrate}$	36.71 Kbps
$\overline{Rxbitrate}$	36.71 Kbps
$\overline{delay}$	61.7 ms
$\overline{jitter}$	21.495 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.11: Test2, Flow:5.

FlowID: 6	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	1000
TxDuration	583.837 sec
Throughput	33.443 Kbps
$\overline{Txbitrate}$	33.447 Kbps
$\overline{Rxbitrate}$	33.447 Kbps
$\overline{delay}$	61.708 ms
$\overline{jitter}$	26.671 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.12: Test2, Flow:6.

FlowID: 7	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	584.596 sec
Throughput	31.986 Kbps
$\overline{Txbitrate}$	31.989 Kbps
$\overline{Rxbitrate}$	31.989 Kbps
$\overline{delay}$	61.704 ms
$\overline{jitter}$	25.213 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.13: Test2, Flow:7.

FlowID: 1	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	2000
TxDuration	596.974 sec
Throughput	790.267 Kbps
$\overline{Txbitrate}$	790.652 Kbps
$\overline{Rxbitrate}$	790.353 Kbps
$\overline{delay}$	93.744 ms
$\overline{jitter}$	3961.032 μsec
$\overline{Txpacketsize}$	516.461 bytes
$\overline{Rxpacketsize}$	516.459 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.14: Test3, Flow:1.

FlowID: 2	
SrcAddress	10.1.3.1
DestAddress	10.3.3.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	596.763 sec
Throughput	33.504 Kbps
$\overline{Txbitrate}$	33.508 Kbps
$\overline{Rxbitrate}$	33.508 Kbps
$\overline{delay}$	61.683 ms
$\overline{jitter}$	21.069 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.15: Test3, Flow:2.

FlowID: 3	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	594.91 sec
Throughput	34.759 Kbps
$\overline{Txbitrate}$	34.763 Kbps
$\overline{Rxbitrate}$	34.763 Kbps
$\overline{delay}$	75.391 ms
$\overline{jitter}$	11424.14 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.16: Test3, Flow:3.

FlowID: 4	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	593.844 sec
Throughput	33.412 Kbps
$\overline{Txbitrate}$	33.415 Kbps
$\overline{Rxbitrate}$	33.415 Kbps
$\overline{delay}$	75.376 ms
$\overline{jitter}$	11554.798 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.17: Test3, Flow:4.

FlowID: 5	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	591.641 sec
Throughput	36.706 Kbps
$\overline{Txbitrate}$	36.71 Kbps
$\overline{Rxbitrate}$	36.71 Kbps
$\overline{delay}$	61.7 ms
$\overline{jitter}$	21.495 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.18: Test3, Flow:5.

FlowID: 6	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	1000
TxDuration	583.837 sec
Throughput	33.443 Kbps
$\overline{Txbitrate}$	33.447 Kbps
$\overline{Rxbitrate}$	33.447 Kbps
$\overline{delay}$	61.708 ms
$\overline{jitter}$	26.671 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.19: Test3, Flow:6.

FlowID: 7	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	584.596 sec
Throughput	31.986 Kbps
$\overline{Txbitrate}$	31.989 Kbps
$\overline{Rxbitrate}$	31.989 Kbps
$\overline{delay}$	61.704 ms
$\overline{jitter}$	25.213 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.20: Test3, Flow:7.

FlowID: 1	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	2000
TxDuration	596.974 sec
Throughput	790.267 Kbps
$\overline{Txbitrate}$	790.652 Kbps
$\overline{Rxbitrate}$	790.353 Kbps
$\overline{delay}$	93.744 ms
$\overline{jitter}$	3961.034 μsec
$\overline{Txpacketsize}$	516.461 bytes
$\overline{Rxpacketsize}$	516.459 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.21: Test4, Flow:1.

FlowID: 2	
SrcAddress	10.1.3.1
DestAddress	10.3.3.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	596.763 sec
Throughput	33.504 Kbps
$\overline{Txbitrate}$	33.508 Kbps
$\overline{Rxbitrate}$	33.508 Kbps
$\overline{delay}$	61.683 ms
$\overline{jitter}$	21.041 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.22: Test4, Flow:2.

FlowID: 3	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	594.91 sec
Throughput	34.759 Kbps
$\overline{Txbitrate}$	34.763 Kbps
$\overline{Rxbitrate}$	34.763 Kbps
$\overline{delay}$	75.391 ms
$\overline{jitter}$	11424.14 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.23: Test4, Flow:3.

FlowID: 4	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	593.844 sec
Throughput	33.412 Kbps
$\overline{Txbitrate}$	33.415 Kbps
$\overline{Rxbitrate}$	33.415 Kbps
$\overline{delay}$	75.376 ms
$\overline{jitter}$	11554.798 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.24: Test4, Flow:4.

FlowID: 5	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	591.641 sec
Throughput	36.706 Kbps
$\overline{Txbitrate}$	36.71 Kbps
$\overline{Rxbitrate}$	36.71 Kbps
$\overline{delay}$	61.7 ms
$\overline{jitter}$	21.403 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.25: Test4, Flow:5.

FlowID: 6	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	1000
TxDuration	583.837 sec
Throughput	33.443 Kbps
$\overline{Txbitrate}$	33.447 Kbps
$\overline{Rxbitrate}$	33.447 Kbps
$\overline{delay}$	61.708 ms
$\overline{jitter}$	26.63 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.26: Test4, Flow:6.

FlowID: 7	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	584.596 sec
Throughput	31.986 Kbps
$\overline{Txbitrate}$	31.989 Kbps
$\overline{Rxbitrate}$	31.989 Kbps
$\overline{delay}$	61.704 ms
$\overline{jitter}$	25.192 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.27: Test4, Flow:7.

FlowID: 1	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	2000
TxDuration	596.972 sec
Throughput	413.873 Kbps
$\overline{Txbitrate}$	790.652 Kbps
$\overline{Rxbitrate}$	413.918 Kbps
$\overline{delay}$	92.986 ms
$\overline{jitter}$	7362.353 μsec
$\overline{Txpacketsize}$	516.461 bytes
$\overline{Rxpacketsize}$	517.095 bytes
PDR	0.523
PLR	0.477

Πίνακας Α'.28: Test5, Flow:1.

FlowID: 2	
SrcAddress	10.1.3.1
DestAddress	10.3.3.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	596.763 sec
Throughput	33.504 Kbps
$\overline{Txbitrate}$	33.508 Kbps
$\overline{Rxbitrate}$	33.508 Kbps
$\overline{delay}$	61.677 ms
$\overline{jitter}$	10.347 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.29: Test5, Flow:2.

FlowID: 3	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	594.91 sec
Throughput	34.759 Kbps
$\overline{Txbitrate}$	34.763 Kbps
$\overline{Rxbitrate}$	34.763 Kbps
$\overline{delay}$	74.865 ms
$\overline{jitter}$	11111.927 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.30: Test5, Flow:3.

FlowID: 4	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	593.844 sec
Throughput	33.412 Kbps
$\overline{Txbitrate}$	33.415 Kbps
$\overline{Rxbitrate}$	33.415 Kbps
$\overline{delay}$	74.835 ms
$\overline{jitter}$	11230.687 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.31: Test5, Flow:4.

FlowID: 5	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	591.641 sec
Throughput	36.706 Kbps
$\overline{Txbitrate}$	36.71 Kbps
$\overline{Rxbitrate}$	36.71 Kbps
$\overline{delay}$	61.695 ms
$\overline{jitter}$	12.902 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.32: Test5, Flow:5.

FlowID: 6	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	1000
TxDuration	583.837 sec
Throughput	33.443 Kbps
$\overline{Txbitrate}$	33.447 Kbps
$\overline{Rxbitrate}$	33.447 Kbps
$\overline{delay}$	61.701 ms
$\overline{jitter}$	15.294 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.33: Test5, Flow:6.

FlowID: 7	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	584.596 sec
Throughput	31.986 Kbps
$\overline{Txbitrate}$	31.989 Kbps
$\overline{Rxbitrate}$	31.989 Kbps
$\overline{delay}$	61.697 ms
$\overline{jitter}$	12.921 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.34: Test5, Flow:7.

FlowID: 1	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	2000
TxDuration	596.974 sec
Throughput	489.787 Kbps
$\overline{Txbitrate}$	790.652 Kbps
$\overline{Rxbitrate}$	489.84 Kbps
$\overline{delay}$	92.885 ms
$\overline{jitter}$	6210.442 μsec
$\overline{Txpacketsize}$	516.461 bytes
$\overline{Rxpacketsize}$	516.754 bytes
PDR	0.619
PLR	0.381

Πίνακας Α'.35: Test6, Flow:1.

FlowID: 2	
SrcAddress	10.1.3.1
DestAddress	10.3.3.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	596.763 sec
Throughput	33.504 Kbps
$\overline{Txbitrate}$	33.508 Kbps
$\overline{Rxbitrate}$	33.508 Kbps
$\overline{delay}$	61.679 ms
$\overline{jitter}$	12.594 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.36: Test6, Flow:2.

FlowID: 3	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	594.91 sec
Throughput	34.759 Kbps
$\overline{Txbitrate}$	34.763 Kbps
$\overline{Rxbitrate}$	34.763 Kbps
$\overline{delay}$	75.031 ms
$\overline{jitter}$	11210.755 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.37: Test6, Flow:3.

FlowID: 4	
SrcAddress	10.1.2.1
DestAddress	10.3.2.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	593.844 sec
Throughput	33.412 Kbps
$\overline{Txbitrate}$	33.415 Kbps
$\overline{Rxbitrate}$	33.415 Kbps
$\overline{delay}$	75.013 ms
$\overline{jitter}$	11341.136 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.38: Test6, Flow:4.

FlowID: 5	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49154
DestPort	1000
TxDuration	591.641 sec
Throughput	36.706 Kbps
$\overline{Txbitrate}$	36.71 Kbps
$\overline{Rxbitrate}$	36.71 Kbps
$\overline{delay}$	61.695 ms
$\overline{jitter}$	13.48 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.39: Test6, Flow:5.

FlowID: 6	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49155
DestPort	1000
TxDuration	583.837 sec
Throughput	33.443 Kbps
$\overline{Txbitrate}$	33.447 Kbps
$\overline{Rxbitrate}$	33.447 Kbps
$\overline{delay}$	61.703 ms
$\overline{jitter}$	17.736 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.40: Test6, Flow:6.

FlowID: 7	
SrcAddress	10.1.1.1
DestAddress	10.3.1.2
Protocol	17 (UDP)
SrcPort	49153
DestPort	1000
TxDuration	584.596 sec
Throughput	31.986 Kbps
$\overline{Txbitrate}$	31.989 Kbps
$\overline{Rxbitrate}$	31.989 Kbps
$\overline{delay}$	61.699 ms
$\overline{jitter}$	16.298 μsec
$\overline{Txpacketsize}$	188.0 bytes
$\overline{Rxpacketsize}$	188.0 bytes
PDR	1.0
PLR	0.0

Πίνακας Α'.41: Test6, Flow:7.

Βιβλιογραφία

[1] *"Δίκτυα Υπολογιστών (4η Έκδοση)."*
Andrew S. Tanenbaum, Κλειδάριθμος, 2003.
ISBN: 960-209-689-6

[2] *"Ποιότητα Υπηρεσίας σε IP Δίκτυα."*
Δημήτριος Πρίμπας, Διπλωματική Εργασία, 2002.

[3] *"Enhancing the DiffServ Architecture of α Simulation Environment."*
Χρήστος Μπούρας, Δημήτριος Πρίμπας, Αφροδίτη Σεβαστή, Ανδρέας Βαρνάβας, Δημοσίευση, 2002.
URL: http://ru6.cti.gr/ru6/publications/3543894.pdf

[4] *"Υλοποίηση του Μηχανισμού Leaky Bucket για τον προσομοιωτή ns-3."*
Πέτρος Μπαλζής, Διπλωματική Εργασία, 2011.

[5] *"Implementation of α leaky bucket module for simulations in NS-3."*
Χρήστος Μπούρας, Πέτρος Μπαλζής, Κώστας Στάμος, Ιωάννης Ζαούδης, Δημοσίευση, 2011.
URL: http://ru6.cti.gr/ru6/publications/2070PID2029707.pdf

[6] *"A DiffServ model for the NS-3 simulator."*
Sanjay Ramroop, Διπλωματική Εργασία, 2011.
URL: http://www.eng.uwi.tt/depts/elec/staff/rvadams/sramroop/index.htm

RFCs

[7] RFC 2474: *"Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers"*
URL: http://www.rfc-editor.org/rfc/rfc2474.txt

[8] RFC 2475: *"An Architecture for Differentiated Services"*
URL: http://www.rfc-editor.org/rfc/rfc2475.txt

[9] RFC 2597: *"Assured Forwarding PHB Group"*
URL: http://www.rfc-editor.org/rfc/rfc2597.txt

[10] RFC 2598: *"An Expedited Forwarding PHB"*
URL: https://www.rfc-editor.org/rfc/rfc2598.txt

[11] RFC 2697: *"A Single Rate Three Color Marker"*
URL: https://www.rfc-editor.org/rfc/rfc2697.txt

[12] RFC 2698: *"A Two Rate Three Color Marker"*
URL: https://www.rfc-editor.org/rfc/rfc2698.txt

[13] RFC 2859: *"A Time Sliding Window Three Colour Marker (TSWTCM)"*
URL: https://www.rfc-editor.org/rfc/rfc2859.txt

[14] RFC 3140: *"Per Hop Behavior Identification Codes"*
URL: http://www.rfc-editor.org/rfc/rfc3140.txt

[15] RFC 3168: *"The Addition of Explicit Congestion Notification (ECN) to IP"*
URL: http://www.rfc-editor.org/rfc/rfc3168.txt

[16] RFC 3246: *"An Expedited Forwarding PHB (Per-Hop Behavior)"*
URL: http://www.rfc-editor.org/rfc/rfc3246.txt

[17] RFC 3247: *"Supplemental Information for the New Definition of the EF PHB (Expedited Forwarding Per-Hop Behavior)"*
URL: http://www.rfc-editor.org/rfc/rfc3247.txt

[18] RFC 3260: *"New Terminology and Clarifications for Diffserv"*
URL: http://www.rfc-editor.org/rfc/rfc3260.txt

[19] RFC 3561: *"Ad hoc On-Demand Distance Vector (AODV) Routing"*
URL: https://www.rfc-editor.org/rfc/rfc3561.txt

[20] RFC 3626: *Optimized Link State Routing Protocol (OLSR)*
URL: https://www.rfc-editor.org/rfc/rfc3626.txt

[21] RFC 4594: *"Configuration Guidelines for DiffServ Service Classes"*
URL: https://www.rfc-editor.org/rfc/rfc4594.txt

[22] RFC 4728: *"The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4"*
URL: https://www.rfc-editor.org/rfc/rfc4728.txt

Web URLs

[23] Εξομοιωτής δικτύων ns-3: http://www.nsnam.org/

[24] ns-3.16 Tutorial.
URL: http://www.nsnam.org/docs/release/3.16/tutorial/singlehtml/index.html

[25] ns-3.16 Manual.
URL: http://www.nsnam.org/docs/release/3.16/manual/singlehtml/index.html

[26] ns-3.16 Model Library.
URL: http://www.nsnam.org/docs/release/3.16/models/singlehtml/index.html

[27] ns-3.16 doxygen API.
URL: http://www.nsnam.org/docs/release/3.16/doxygen/index.html

[28] The Opte project: http://www.opte.org/

[29] AODV-UU: http://aodvuu.sourceforge.net/

[30] Click Modular Router: http://www.read.cs.ucla.edu/click/

[31] OpenFlow: http://www.openflow.org/

[32] Jurassic Park I, MPEG 4 (high quality) video trace:
http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html

[33] LaTeX project: http://www.latex-project.org/

[34] Dia: https://live.gnome.org/Dia

Η σελίδα έχει μείνει εσκεμμένα κενή