ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Υλοποίηση του μηχανισμού leaky bucket για τον προσομοιωτή NS-3»

ΜΠΑΛΤΖΗΣ ΠΕΤΡΟΣ
Α.Μ. 3701

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ

Χρήστος Μπούρας, Καθηγητής

ΕΠΙΒΛΕΠΟΝΤΕΣ

Κωνσταντίνος Στάμος

Πρόλογος

Τελειώνοντας την παρούσα διπλωματική εργασία θα ήθελα να απευθύνω ευχαριστίες στα άτομα που με βοήθησαν, χωρίς την βοήθεια των οποίων πιθανόν να μην ήταν δυνατή η εκπόνησή της.

Κατ' αρχάς, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κ. Χρήστο Μπούρα (Καθηγητής τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών) για την επίβλεψη της εργασίας, τις πολύτιμες συμβουλές του και την συμπαράστασή του ώστε να ολοκληρωθεί η παρούσα εργασία αλλά και τη δυνατότητα που μου παρείχε ν' ασχοληθώ με ένα τόσο ενδιαφέρον θέμα.

Ακόμα θα ήθελα να απευθύνω ιδιαίτερες ευχαριστίες στους Δρ. Κώστα Στάμο και Γιάννη Ζαούδη όχι μόνο για την βοήθειά τους, η οποία υπήρξε καταλυτική στην εκπόνηση αυτής της διπλωματικής εργασίας αλλά και για την συμπαράστασή τους καθ' όλη τη διάρκεια της συνεργασίας μου με την Ερευνητική Μονάδα 6 του Ερευνητικού Ακαδημαϊκού Ινστιτούτου Τεχνολογίας Υπολογιστών.

Κλείνοντας, θα ήθελα να ευχαριστήσω την οικογένεια μου που χωρίς την ψυχική και υλική βοήθειά τους και την αμέριστη συμπαράσταση τους όποτε τη χρειαζόμουν, δεν θα ήταν δυνατό να ολοκληρώσω τις προπτυχιακές μου σπουδές.

Πάτρα, Φεβρουάριος 2012
Πέτρος Χ. Μπαλτζής

Στην οικογένειά μου

Περιεχόμενα

Λίστα Εικόνων

ΕΙΣΑΓΩΓΗ

Η αξία της προσομοίωσης στον τομέα της δικτύωσης έχει από καιρό αναγνωριστεί. Με τη χρήση της επιτρέπεται ο ταχύτερος και ευκολότερος πειραματισμός καθώς και η επικύρωση των μοντέλων, αρχιτεκτονικών και μηχανισμών. Ένας μακρύς κατάλογος πρωτοκόλλων δικτύωσης έχει μελετηθεί με τη χρήση προσομοίωσης. Έπειτα τα πρωτόκολλα δοκιμάζονται σε πραγματικές συνθήκες και περιβάλλοντα παραγωγής. Χρησιμοποιώντας προσομοίωση ένας ερευνητής μπορεί να επικυρώσει την εργασία του και να την εξειδικεύσει σε τέτοια κλίμακα που δεν είναι ούτε εύκολο αλλά ούτε και φθηνό να επιτευχθεί. Επιπλέον, μπορεί να ξεπεράσει πρακτικές δυσχέρειες που με άλλο τρόπο θα μπορούσαν να αποδειχθούν ανυπέρβλητες, σε μια πρώτη φάση για την ανάπτυξης της ιδέας.

Η πραγματοποίηση αξιολόγησης και δοκιμών σε πραγματικά δίκτυα θα απαιτούσαν μεγάλες επενδύσεις στην ανάπτυξη λογισμικού και πολύ συχνά ανάπτυξη υλικού. Επιπλέον, θα απαιτούνταν ένα μεγάλο τμήμα εξοπλισμού για τη διεξαγωγή πειραμάτων σε μεγάλη κλίμακα, έχοντας πολλαπλές πηγές και προορισμούς διαφόρων τύπων. Επίσης, ιδιαίτερα χρήσιμα και πολύτιμα είναι: η αξιολόγηση μικρής κλίμακας σε ένα εργαστήριο, τα σενάρια ευρείας περιοχής, καθώς και οι συνηθισμένες προσομοιώσεις, ωστόσο το καθένα από αυτά αντιμετωπίζει δυσκολίες όταν δοκιμάζεται χρησιμοποιώντας πραγματικά δίκτυα. Συγκεκριμένα, είναι δύσκολο να αναπαραχθεί το πραγματικό μίγμα ροών και τοπολογιών, μπορεί να απαιτηθεί σημαντική προσπάθεια, πολλά έξοδα και η επανάληψη των πειραμάτων σε ελεγχόμενες συνθήκες μπορεί να είναι δύσκολη. Ως εκ τούτου, η πραγματική δοκιμή του δικτύου είναι πάρα πολύ δαπανηρή για τις περισσότερες ερευνητικές μονάδες, τουλάχιστον στα πρώτα στάδια της ανάπτυξης. Δοκιμές σε ένα περιβάλλον προσομοίωσης είναι η καταλληλότερη και η πιο ανέξοδη λύση για την αξιολόγηση των μηχανισμών δικτύωσης και των πρωτοκόλλων.

Ίσως το πιο ευρέως χρησιμοποιούμενο περιβάλλον προσομοίωσης σε έρευνα είναι ο προσομοιωτής ns-2 1.18.3, ο οποίος παρέχει έναν ανθεκτικό πυρήνα προσομοίωσης με ένα πολύ μεγάλο αριθμό είτε ενσωματωμένων είτε πρόσθετων επεκτάσεων από τους ερευνητές. Η αξία του ns-2 ως περιβάλλον προσομοίωσης έχει αναγνωριστεί από καιρό στον τομέα της έρευνας των δικτύων. Ωστόσο, ο ns-2 έχει και τις ελλείψεις του 1.18.3 λόγω της μακράς ιστορίας του και των αρχικών παραδοχών της αρχιτεκτονικής του, οι οποίες και οδήγησαν στην πρωτοβουλία για την ανάπτυξη μιας τελικής αρχιτεκτονικής για την αντικατάστασή του, που ονομάζεται ns-31.18.3. Αν και ο ns-3 είναι βελτιωμένος σε πολλές πτυχές έναντι του ns-2, είναι ουσιαστικά διαφορετικά και ασύμβατα περιβάλλοντα, πράγμα που σημαίνει ότι τα περισσότερα add-on που

είναι διαθέσιμα για τον ns-2 δεν είναι διαθέσιμα για τον ns-3. Ο ns-3 είναι ένας νέος προσομοιωτής ο οποίος γράφτηκε από το μηδέν και δεν είναι μια εξέλιξη του ns-2 1.18.3. Είναι μια σύγχρονη προσέγγιση ενός προσομοιωτή που στοχεύει να επιτύχει πολύ καλύτερες επιδόσεις από τον ns-2.

Μία από τις αρχιτεκτονικές που υποστηρίζεται στον ns-2 είναι το DiffServ. Το Differentiated Services (DiffServ) framework είναι μια από τις βασικές αρχιτεκτονικές που έχουν προταθεί για QoS (ποιότητα στην υπηρεσία) στο Διαδίκτυο και υποστηρίζεται ευρέως από πωλητές δικτυακού εξοπλισμού. Εισάγει τις διαφορετικές κατηγορίες υπηρεσιών και την εφαρμογή μηχανισμών για τον διαφορετικό χειρισμό των πακέτων που ανήκουν σε κάθε μια από αυτές τις κατηγορίες υπηρεσιών. Σύμφωνα με το μοντέλο συγχώνευσης ροών, τα πακέτα δικαιούται μια ορισμένη υπηρεσία ή ανήκουν σε μια συγκεκριμένη συγχώνευση και έτσι σημειώνονται με μια διακριτή τιμή τη λεγόμενη Differentiated Services Code Point (DSCP), μια μοναδική κεφαλίδα πακέτου, στην οποία βασίζονται οι δρομολογητές για την παροχή διαφοροποιημένων υπηρεσιών. Η αρχιτεκτονική DiffServ αποτελείται από μηχανισμούς ταξινόμησης, αστυνόμευσης και μορφοποίησης της κυκλοφορίας.

Η μορφοποίηση της κίνησης είναι ο έλεγχος της κίνησης των πακέτων σε ένα δίκτυο υπολογιστών προκειμένου να βελτιστοποιηθεί και να δοθεί εγγύηση καλής εκτέλεσης, να βελτιωθεί η λανθάνουσα κατάσταση και να αυξηθεί το χρησιμοποιήσιμο εύρος ζώνης για ορισμένα είδη πακέτων καθυστερώντας άλλα είδη πακέτων, ανάλογα με ορισμένα κριτήρια. Εάν μια σύνδεση γίνεται κορεσμένη σε κάποιο σημείο όπου υπάρχει ένα σημαντικό επίπεδο συμφόρησης (είτε στο upstream είτε στο downstream), ο χρόνος απόκρισης μπορεί να αυξηθεί σημαντικά, έτσι ένας αλγόριθμος μορφοποίησης είναι απαραίτητος. Ένας σημαντικός αλγόριθμος για τη μορφοποίηση, ο οποίος έλειπε από τον ns-3, είναι ο Leaky Bucket, ο οποίος είναι το επίκεντρο των εργασιών μας σε αυτήν τη διπλωματική εργασία. Σε αυτήν ο ns-3 χρησιμοποιήθηκε ως βασικό εργαλείο για την υλοποίηση και την προσομοίωση του μηχανισμού Leaky Bucket με τη χρήση του οποίου προχωρήσαμε στην διαμόρφωση των εισερχόμενων ροών πάνω από ενσύρματα και ασύρματα δίκτυα.

ΔΗΜΟΣΙΕΎΣΕΙΣ ΣΕ ΣΥΝΕΔΡΙΑ

Από την εργασία που έγινε στα πλαίσια της διπλωματικής αυτής προέκυψε η παρακάτω δημοσίευση η οποία και παρουσιάστηκε σε διεθνές συνέδριο με κριτές:

P. Baltzis, C. Bouras, K. Stamos, G. Zaoudis, “Implementation of leaky bucket module for simulations in NS - 3” Workshop on ICT - Contemporary Communication and Information Technology, Split - Dubrovnik, 15 - 17 September 2011.

ΠΟΙΟΤΗΤΑ ΥΠΗΡΕΣΙΑΣ
(QoS – Quality Of Service)

1.1. Εισαγωγή

Σε ένα πραγματικό IP δίκτυο, η βασική υπηρεσία που προσφέρεται είναι η υπηρεσία Best Effort (καλύτερης προσπάθειας). Σύμφωνα με αυτή κάθε πακέτο που φτάνει σε ένα δρομολογητή δέχεται την ακόλουθη επεξεργασία:

- Αρχικά γίνεται έλεγχος για το που θα σταλεί το πακέτο που μόλις έφτασε.
- Στη συνέχεια το πακέτο στέλνεται στη γραμμή εξόδου για το επόμενο hop. Εάν δεν είναι δυνατό το πακέτο να σταλεί άμεσα αυτό αποθηκεύεται προσωρινά σε μια ουρά εξόδου.
- Εάν η ουρά αυτή είναι γεμάτη το πακέτο απορρίπτεται. Σε περίπτωση που όταν φτάσει το πακέτο η ουρά περιέχει ήδη άλλα πακέτα τότε το πακέτο αυτό δέχεται επιπλέον καθυστέρηση σύμφωνα με το χρόνο που απαιτείται ώστε τα παλιότερα πακέτα να φύγουν από την ουρά.

Ουσιαστικά στην best effort υπηρεσία όλα τα πακέτα αντιμετωπίζονται όμοια και δεν υπάρχουν εγγυήσεις, διαφοροποιήσεις ή προσπάθεια επιβολής δικαιοσύνης. Εντούτοις το δίκτυο προσπαθεί να προωθήσει όσο περισσότερη κίνηση μπορεί με «λογική» ποιότητα.

Στο δίκτυο πολλές φορές παρουσιάζεται το φαινόμενο της συμφόρησης, που ουσιαστικά εμφανίζεται όταν ένας δρομολογητής αποθηκεύει πακέτα σε μια ουρά εξόδου, γεγονός που συμβαίνει όταν λαμβάνει περισσότερα πακέτα από αυτά που μπορεί να μεταδώσει. Στη διάρκεια της περιόδου συμφόρησης είναι λογικό τα πακέτα να δέχονται μεγαλύτερη καθυστέρηση ενώ όταν η ουρά εξόδου γεμίσει, τότε αυτά απορρίπτονται.

Ωστόσο υπάρχουν εφαρμογές που απαιτούν ορισμένες εγγυήσεις (κυρίως σε καθυστέρηση και απόρριψη πακέτων) όπως οι εφαρμογές voice over IP και Videoconference. Αυτές προκείμενου να πετύχουν τις εγγυήσεις ποιότητας που εξασφαλίζουν τη σωστή λειτουργία τους πρέπει να διασχίζουν στο δίκτυο άδειες ή σχεδόν άδειες ουρές, γεγονός που για να συμβεί πρέπει να υπάρξουν μηχανισμοί που θα τις διασφαλίσουν.

Ένας τρόπος προκειμένου να υπάρξει παροχή εγγυήσεων σε κάποια κίνηση είναι η διαχείριση ορισμένων πακέτων διαφορετικά έναντι των υπολοίπων. Στο σημείο αυτό ουσιαστικά εισέρχεται η έννοια της ποιότητας υπηρεσίας (Quality of Service). Ένας ορισμός της είναι: η ικανότητα ενός στοιχείου του δικτύου να παρέχει ένα επίπεδο διαβεβαίωσης (εγγύησης) σε ένα υποσύνολο κίνησης ότι οι απαιτήσεις υπηρεσίας της μπορεί να επιτευχθούν με συγκεκριμένη (πολύ μεγάλη) πιθανότητα.

Ουσιαστικά οι μηχανισμοί του Quality of service δεν παρέχουν μεγαλύτερη χωρητικότητα στο δίκτυο ή κάτι παρόμοιο, αλλά απλώς κάνουν καλύτερη διαχείριση

του δικτύου ώστε να χρησιμοποιείται πιο αποδοτικά και σύμφωνα με τις απαιτήσεις των εφαρμογών

1.2. QoS Μετρικές

Οι μετρικές που ενδιαφέρουν τις εφαρμογές που ζητούν ποιότητα υπηρεσίας στην εξυπηρέτηση είναι οι ακόλουθες:

- **Χωρητικότητα (Bandwidth):** Χωρητικότητα είναι το μέτρο της μετάδοσης των δεδομένων, που εκφράζεται συνήθως σε Kbits per second (Kbps) ή megabits per second (Mbps). Δείχνει πόσα δεδομένα μπορούν να μεταδοθούν σε ένα δίκτυο. Αύξηση του εύρους ζώνης σημαίνει αύξηση και του αριθμού μεταφοράς των δεδομένων. Το Bandwidth αποτελείται από 4 ποσότητες οι οποίες είναι :
    1. Το μέγιστο μέγεθος καταιγισμού.
    2. Η μέγιστη χωρητικότητα (peak bandwidth).
    3. Η ελάχιστη εγγυημένη χωρητικότητα.
    4. Η μέση χωρητικότητα.

  και η τιμή του εξαρτάται από τη φυσική διαμόρφωση του μονοπατιού κίνησης όπου κινούνται τα δεδομένα και από το πόσα ακόμα μονοπάτια μοιράζονται το ίδιο link του φυσικού μέσου με το μονοπάτι αυτό.
- **Καθυστέρηση (delay):** Καθυστέρηση είναι το χρονικό διάστημα που πρέπει να περάσει από τη στιγμή που ένας κόμβος στέλνει ένα μήνυμα σε ένα άλλο κόμβο μέχρι τη στιγμή που ο άλλος κόμβος το λαμβάνει. Αποτελείται από τη καθυστέρηση μετάδοσης, την καθυστέρηση σε ένα μονοπάτι μεταφοράς (ή αλλιώς καθυστέρηση διάδοσης) και τη καθυστέρηση σε μια συσκευή εντός ενός μονοπατιού μεταφοράς, για παράδειγμα σε ένα δρομολογητή. Latency ενός δρομολογητή λέμε το χρόνο που μεσολαβεί από τη στιγμή που λαμβάνει το πακέτο ο δρομολογητής μέχρι τη στιγμή που το αναμεταδίδει. Όσο μεγαλύτερη είναι η καθυστέρηση, τόσο μεγαλύτερες είναι οι απαιτήσεις για την αποδοτική λειτουργία του πρωτοκόλλου μεταφοράς. Δηλαδή για το TCP πρωτόκολλο, υψηλά επίπεδα καθυστέρησης σημαίνουν τη μεταφορά μεγαλύτερων ποσοτήτων δεδομένων στο δίκτυο.
- **Στιγμιαία διακύμανση καθυστέρησης πακέτων (Instantaneous Packet Delay Variation - IPDV) ή jitter:** Ουσιαστικά το jitter αναφέρεται σε ζεύγη πακέτων και είναι η διαφορά μεταξύ της καθυστέρησης του πρώτου πακέτου

από το δεύτερο. Υπάρχουν πολλές περιπτώσεις εφαρμογών video ή φωνής σε ένα δίκτυο, όπου τα πακέτα δε φτάνουν στον προορισμό τους με συνεχόμενη σειρά ή σε μία χρονική βάση. Έτσι πολλές από αυτές τις εφαρμογές απαιτούν να έχουν ένα άνω όριο για το jitter προκειμένου η απόδοσή τους να είναι καλή.

- **Απώλεια πακέτων (One-Way Packet Loss – OWPL):** Μια παράμετρος που ενδιαφέρει πολλές εφαρμογές είναι η απώλεια πακέτων που μεταδόθηκαν από την πηγή και δεν λήφθηκαν από τον παραλήπτη ή παραλήφθηκαν με λάθη. Η απώλεια πακέτων προκαλείται είτε από απώλεια κάποιου link, είτε εξαιτίας προβλημάτων στη ρύθμιση των συσκευών του δικτύου είτε τέλος από συμφόρηση στο δίκτυο. Γενικά η επίδραση της απώλειας πακέτων στις εφαρμογές μπορεί να είναι καταστροφική υποβαθμίζοντας την απόδοσή τους. Επίσης σε πολλές εφαρμογές ενδεχόμενη αποστολή ξανά ενός χαμένου πακέτου δεν έχει καμιά απολύτως σημασία και αντιθέτως δυσχεραίνει την λειτουργία της εφαρμογής παρά την βοηθά. Ένα παράδειγμα τέτοιας εφαρμογής είναι η τηλεδιάσκεψη.

1.3. Τύποι QoS

Στην πράξη υπάρχουν 2 τύποι παροχής ποιότητας υπηρεσίας και απευθύνονται κατά βάση σε διαφορετικές εφαρμογές.

- Στην πρώτη περίπτωση παρουσιάζεται η κράτηση πόρων (resource reservation = integrated service), όπου οι πόροι του δικτύου διατίθενται με βάση τις ανάγκες των εφαρμογών. Πιο συγκεκριμένα για κάθε πελάτη που επιθυμεί κάποια ποιότητα υπηρεσίας γίνεται στο δίκτυο κράτηση πόρων ώστε να εξυπηρετούνται οι εφαρμογές του.
- Η δεύτερη περίπτωση είναι η διαδικασία παροχής προτεραιότητας σε ορισμένα πακέτα (differentiated service). Η κίνηση του δικτύου διαχωρίζεται και οι πόροι διανέμονται δίκαια με βάση τα κριτήρια αστυνόμευσης και διαχείρισης του bandwidth. Προκείμενου να επιτευχθεί ποιότητα στην υπηρεσία, οι διαχωρισμοί (classifications) που έχουν μεγαλύτερες απαιτήσεις απολαμβάνουν προνομιακή μεταχείριση από το δίκτυο.

Οι παραπάνω τύποι ποιότητας υπηρεσίας μπορούν να εφαρμοστούν είτε σε μεμονωμένες ανεξάρτητες ροές, είτε σε συνενώσεις ροών(aggregates).

Για την κάλυψη των αναγκών για όλους τους τύπους ποιότητας υπηρεσίας υπάρχουν τα ακόλουθα πρωτόκολλα και αλγόριθμοι που χρησιμοποιούνται:

- Reservation Protocol (RSVP), επίσης γνωστό και ως integrated services. Αυτό είναι ένα πρωτόκολλο που χρησιμοποιείται κυρίως σε ροές και σπανιότερα σε συνενώσεις ροών και ο ρόλος του είναι να κάνει κράτηση πόρων
- Differentiated Services. Αποτελεί μια αρχιτεκτονική (ένα πλαίσιο) που παρέχει ένα απλό τρόπο να κατηγοριοποιεί και να θέτει σε προτεραιότητα ροές αλλά και κυρίως συνενώσεις ροών
- Multi Protocol Label switching (MPLS). Λειτουργεί όμοια με τη diffserv αρχιτεκτονική αφού μαρκάρει τα πακέτα στα σημεία εισόδου στο δίκτυο και τα απομαρκάρει στα σημεία εξόδου. Όμως, στόχος του μαρκαρίσματος δεν είναι να παρέχει προτεραιότητα στους δρομολογητές όπως συμβαίνει στη diffserv αλλά να καθορίσει το hop στον επόμενο δρομολογητή.

1.3.1. Integrated Service (IntServ)

Στο σημείο αυτό περιγράφεται επιγραμματικά ο τρόπος λειτουργίας του πρωτοκόλλου RSVP. Σύμφωνα με αυτό πρέπει κατά μήκος όλης της διαδρομής που ακολουθούν τα πακέτα να γίνουν κρατήσεις πόρων σύμφωνα με τις ανάγκες της εφαρμογής. Η διαδικασία κράτησης πόρων είναι ακολουθιακή και ο πρώτος δρομολογητής στέλνει κατάλληλο μήνυμα στον επόμενο όπου ζητά κράτηση πόρων. Η διαδικασία αυτή εξελίσσεται μέχρι να φτάσει στον παραλήπτη, ο οποίος τότε στέλνει στην αντίθετη διαδρομή επιβεβαιώσεις κράτησης. Ουσιαστικά το RSVP πρωτόκολλο εφαρμόζεται στις Integrated υπηρεσίες, οι οποίες αποτελούνται από 2 διαφορετικούς τύπους:

- Guaranteed. Ουσιαστικά αυτή η υπηρεσία είναι η πλησιέστερη δυνατή στα αφιερωμένα ιδεατά κυκλώματα (dedicated virtual circuits) 1.18.3.Η υπηρεσία αυτή παρέχει σε κάθε εφαρμογή συγκεκριμένες εγγυήσεις που αποτελούνται από καθορισμένα όρια στην από άκρο σε άκρο μέγιστη καθυστέρηση της ροής ενώ ταυτόχρονα διασφαλίζει και εγγυάται συγκεκριμένη χωρητικότητα. Με άλλα λόγια εγγυάται ότι τα όλα πακέτα θα φτάσουν στον προορισμό σε συγκεκριμένο χρονικό διάστημα, με την μόνη προϋπόθεση ότι αυτά δεν θα ξεπεράσουν το προφίλ που είχαν δηλώσει όταν ζήτησαν την υπηρεσία αυτή.

- Controlled Load. Αυτή είναι ισοδύναμη με την υπηρεσία καλύτερης προσπάθειας σε συνθήκες έλλειψης φόρτου. Στην πραγματικότητα είναι καλύτερη από την υπηρεσία καλύτερης προσπάθειας αλλά σε καμία περίπτωση δεν παρέχει τις απόλυτες εγγυήσεις που παρέχει η Guaranteed υπηρεσία. Η υπηρεσία αυτή σχεδιάστηκε προκειμένου να εξυπηρετήσει εφαρμογές πραγματικού χρόνου, οι οποίες με την βοήθεια buffers μπορούν να αντιμετωπίσουν το πρόβλημα του jitter. Επίσης απευθύνεται και σε «ελαστικές» εφαρμογές όπως e-mail και ftp που δεν έχουν ιδιαίτερες απαιτήσεις εκτός από την μέση καθυστέρηση. Ουσιαστικά στόχος της υπηρεσίας είναι εφαρμογές όπου τους αρκούν εγγυήσεις για μέση καθυστέρηση και δεν απαιτούν εγγυήσεις για συγκεκριμένο jitter. Η υπηρεσία αυτή μπορεί να χαρακτηριστεί ότι προσεγγίζει την υπηρεσία καλύτερης προσπάθειας σε συνθήκες ελάχιστου φόρτου, ακόμη και στην περίπτωση που το δίκτυο λειτουργεί σε κατάσταση συμφόρησης.

1.3.2. Differentiated Service (DiffServ)

Η αρχιτεκτονική DiffServ σε αντίθεση με την IntServ που κάνει κράτηση πόρων, δεν εφαρμόζει καμιά τέτοια διαδικασία απλώς αναγνωρίζει κάποιες ροές πακέτων και τις διαχειρίζεται προνομιακά έναντι των υπολοίπων. Ο τρόπος που υλοποιείται, δηλαδή οι μηχανισμοί DiffServ, παρουσιάζονται αναλυτικά στο επόμενο κεφάλαιο. Στο σημείο αυτό θα περιγραφεί λίγο ο τρόπος με τον οποίο ο διαχειριστής του δικτύου και οι χρήστες που επιζητούν παροχή ποιότητας υπηρεσιών για τις εφαρμογές συμφωνούν στα χαρακτηριστικά της. Καταρχήν το κοινά αποδεκτό σύνολο των παραμέτρων που προσδιορίζουν την ποιότητα υπηρεσίας που παρέχεται από το δίκτυο είναι το ακόλουθο όπως παρουσιάστηκε και παραπάνω σε αυτό το κεφάλαιο:

- Καθυστέρηση
- IP Packet Delay Variation (IPDV) ή Jitter
- Χωρητικότητα
- Απώλεια πακέτων

Όταν ένας πελάτης επιθυμεί μια συγκεκριμένη ποιότητα υπηρεσίας τότε γνωστοποιεί στο δίκτυο για όλες τις παραπάνω ποσότητες τις τιμές που η εφαρμογή (ή οι εφαρμογές) του χρειάζεται ώστε η απόδοσή της να είναι καλή. Στη συνέχεια το

δίκτυο ελέγχει αν μπορεί να προσφέρει τις εγγυήσεις που ζητά ο πελάτης και είτε τις αποδέχεται, είτε προτείνει αυτό τις εγγυήσεις που μπορεί να παρέχει σύμφωνα με τις δυνατότητές του. Στη συνέχεια και εφόσον συμφωνήσουν στις εγγυήσεις που θα παρέχονται και στα χαρακτηριστικά της κίνησης που θα εισάγει ο πελάτης στο δίκτυο τότε υπογράφεται μια συμφωνία όπου προσδιορίζει αναλυτικά όλα τα παραπάνω. Η συμφωνία αυτή καλείται SLA (Service Level Agreement) και είναι δεσμευτική και για τις 2 πλευρές.

Στη συνέχεια προκειμένου να οριστεί μια υπηρεσία παροχής ποιότητας υπάρχει μια σειρά προϋποθέσεων που πρέπει να εκπληρούνται 1.18.3:

- Σταθερή λειτουργία του φυσικού επιπέδου και του επιπέδου σύνδεσης δεδομένων
- Bit error rate <=10^(-12)
- Αξιοπιστία του εξοπλισμού
- Επιλογή του MTU (μέγιστη μονάδα μετάδοσης) για την αποφυγή κατακερματισμού των πακέτων. Η ποσότητα MTU είναι το μέγιστο μέγεθος πακέτου της ροής.
- Over provisioning χαρακτήρας του δικτύου. Η προϋπόθεση αυτή δηλώνει ουσιαστικά πως θα πρέπει ο ρυθμός άφιξης πακέτων στο δίκτυο να είναι μικρότερος από το ρυθμό εξυπηρέτησής τους.

Αφού αυτές πλέον θεωρείται ότι εκπληρώνονται ορίζονται διάφορες υπηρεσίες με συνδυασμούς μηχανισμών και τεχνικών αστυνόμευσης, δρομολόγησης, διαχείρισης ουρών κλπ. Γενικά έχουν προταθεί 2 είδη DiffServ υπηρεσιών –(per hop behaviors) που περιγράφονται παρακάτω. Με τον όρο per hop behavior καλείται η «συμπεριφορά προώθησης» (forwarding behaviour) που εφαρμόζεται στα πακέτα σε κάθε κόμβο του DiffServ domain.

- Expedited Forwarding (EF) 1.18.3. Σε αυτή την κατηγορία υπηρεσιών στόχο αποτελεί η ελαχιστοποίηση της καθυστέρησης και του jitter ενώ παράλληλα στοχεύει ώστε να παρέχει ποιότητα υπηρεσίας στον υψηλότερο βαθμό. Τα πακέτα που υπερβαίνουν το προφίλ της κίνησης που έχει συμφωνηθεί ότι θα εισάγει ο χρήστης (στο SLA που υπογράφηκε) απορρίπτονται. Γενικά οι υπηρεσίες αυτές της κατηγορίες εξομοιώνουν τη λειτουργία μιας εικονικής μισθωμένης γραμμής.

- Assured Forwarding (AF) 1.18.3. Η κατηγορία αυτή διαθέτει το πολύ 4 κλάσεις εξυπηρέτησης και το πολύ 3 επίπεδα απόρριψης για κάθε κλάση. Η AF κίνηση

που υπερβαίνει τα χαρακτηριστικά διανέμεται με όχι τόσο μεγάλη πιθανότητα όσο η εντός προφίλ κίνηση, γεγονός που σημαίνει ότι μπορεί να υποβιβάζεται αλλά δεν σημαίνει απαραίτητα ότι απορρίπτεται.

Το DiffServ υποθέτει ότι υπάρχει μια συμφωνία παροχής υπηρεσίας(SLA), μεταξύ δικτύων που μοιράζονται ένα σύνορο. Στο επόμενο κεφάλαιο παρουσιάζονται αναλυτικά οι μηχανισμοί του DiffServ και στο τέλος του κεφαλαίου αναλύονται διεξοδικότερα οι DiffServ υπηρεσίες. Μέσα σε αυτές είναι και ο αλγόριθμος του leaky bucket που είναι άλλωστε αυτό που μελετήθηκε αλλά και υλοποιήθηκε για τον ns-3 στα πλαίσια αυτής της διπλωματικής.

ΜΗΧΑΝΙΣΜΟΙ DiffServ

1.4. Εισαγωγή

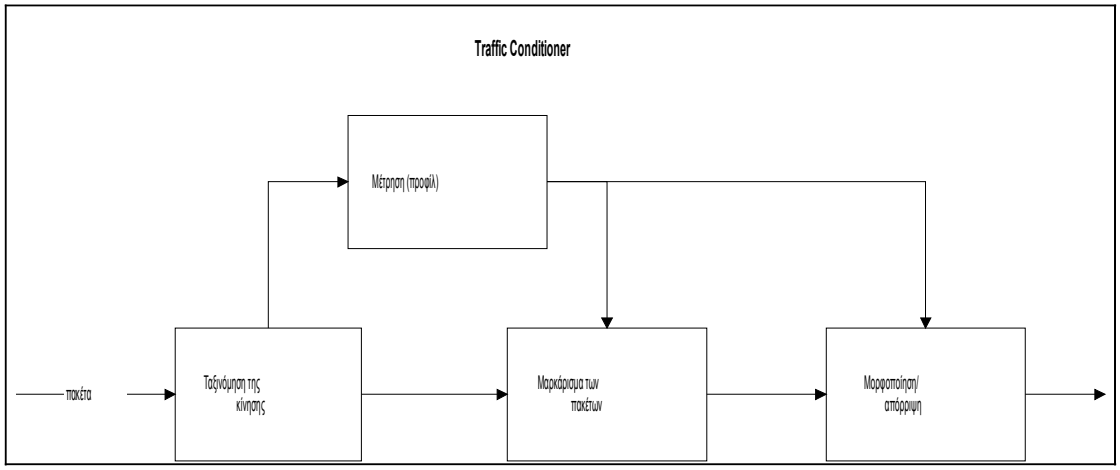
Όπως αναφέραμε στο προηγούμενο κεφάλαιο προκειμένου να υλοποιηθεί μια υπηρεσία παροχής ποιότητας υπηρεσίας και ειδικότερα ποιότητα υπηρεσίας με τη μέθοδο DiffServ, απαιτείται να λειτουργήσουν στο δίκτυο μια σειρά από μηχανισμούς. Αυτοί ενεργούν πάνω στις ροές και αναλυτικά είναι οι ακόλουθοι.

- Ταξινόμηση των πακέτων (packet classification). Ο μηχανισμός αυτός ταξινομεί τα πακέτα που φτάνουν σε ένα κόμβο σε ροές ή συνενώσεις ροών ώστε στη συνέχεια αυτά να εξυπηρετηθούν κατάλληλα.
- Μαρκάρισμα (marking) των πακέτων. Με το μηχανισμό αυτό τα πακέτα μαρκάρονται ανάλογα με την κλάση στην οποία ανήκουν (προέκυψε από τον προηγούμενο μηχανισμό) είτε με βάση άλλα κριτήρια όπως τα χαρακτηριστικά της κίνησης που παρουσιάζουν κλπ.
- Μέτρηση (metering) της κίνησης. Στην προκειμένη περίπτωση ο μηχανισμός αυτός ελέγχει το προφίλ της κίνησης που δέχεται και το συγκρίνει με το προσυμφωνηθέν προφίλ κίνησης όπως προκύπτει από το SLA που έχει υπογράψει με τον διαχειριστή του δικτύου. Στη συνέχεια ο μηχανισμός αυτός διαχωρίζει τα πακέτα σε έναν αριθμό κατηγοριών (ανάλογα αν βρίσκονται στα νόμιμα πλαίσια ή όχι). Ο αριθμός των κατηγοριών αυτών εξαρτάται από τη συμφωνία που έχει γίνει με το δίκτυο όπου επίσης καθορίζεται η μεταχείριση που θα έχουν τα πακέτα όλων των κατηγοριών.
- Μηχανισμός μορφοποίησης (shaping) της κίνησης όπου τροποποιούνται τα χαρακτηριστικά της κίνησης που έλαβε ο κόμβος. Επίσης αντί του μηχανισμού αυτού μπορεί να υπάρχει μηχανισμός απόρριψης (dropping) των πακέτων.

Γενικά η σειρά με την οποία συνήθως αυτοί χρησιμοποιούνται είναι και η σειρά με την οποία παρουσιάστηκαν. Πρέπει στο σημείο αυτό να αναφέρουμε ότι είναι επίσης δυνατό οι μηχανισμοί μαρκαρίσματος και μέτρησης του προφίλ της κίνησης να εμφανίζονται αντίστροφα, δηλαδή πρώτα μέτρηση του προφίλ της κίνησης και ύστερα με βάση αυτό το κριτήριο μαρκάρισμα των πακέτων. Επίσης μετά από τη διαδικασία μέτρησης του προφίλ, σε ορισμένες «κατηγορίες» πακέτων (και κυρίως στα νόμιμα πακέτα) συνήθως δεν εφαρμόζεται κανένας περαιτέρω μηχανισμός και εισάγονται έτσι στο δίκτυο. Οι μηχανισμοί αυτοί και η σειρά με την οποία συνήθως εφαρμόζονται παρουσιάζεται σχηματικά στην Εικόνα 1.

Στο σημείο αυτό είναι αναγκαίο να τονιστεί πως όλοι οι παραπάνω μηχανισμοί και λειτουργικότητες εφαρμόζονται στους συνοριακούς κόμβους (edge routers) σε

ένα DiffServ enabled domain. Αντίθετα στους ενδιάμεσους κόμβους (core routers) η DiffServ αρχιτεκτονική προσδιορίζει πως οι παραπάνω μηχανισμοί δεν έχουν καμία εφαρμογή.



Εικόνα 1 Οι βασικοί μηχανισμοί data path και η σειρά με την οποία εκτελούνται

1.5. Ταξινόμηση της κίνησης

Η ταξινόμηση της κίνησης είναι το πρώτο σημείο στο οποίο βασίζεται η παροχή ποιότητας υπηρεσίας στην εξυπηρέτηση των πακέτων και για το λόγο αυτό αποτελεί ιδιαίτερα σημαντικό παράγοντα. Η ταξινόμηση των πακέτων προκειμένου να εξυπηρετηθούν κατάλληλα σε ένα δίκτυο που υποστηρίζει QoS γίνεται είτε σε επίπεδο ροών, είτε σε επίπεδο συνενώσεων ροών (aggregates). Η διαδικασία αυτή γίνεται κυρίως με τον έλεγχο της επικεφαλίδας κάθε πακέτου και την άντληση από εκεί κάποιας πληροφορίας με βάση την οποία γίνεται η ταξινόμηση. Γενικά ο μηχανισμός αυτός απαιτείται να είναι πολύ γρήγορος , ακολουθώντας το ρυθμό άφιξης των πακέτων, και ιδιαίτερα ακριβής.
Θεωρητικά οι ροές χαρακτηρίζονται από μια πεντάδα που αποτελείται από:

- Την IP διεύθυνση του αποστολέα
- Τον αριθμό port του αποστολέα
- Την IP διεύθυνση του παραλήπτη
- Τον αριθμό port του παραλήπτη
- Το πρωτόκολλο που χρησιμοποιείται.

Κάνοντας τελικά ταξινόμηση ανά ροή με βάση αυτή την πεντάδα είναι μια διαδικασία αρκετά δύσκολη παρότι όλα αυτά τα πεδία υπάρχουν στην IP επικεφαλίδα. (στο σημείο αυτό να σημειώσουμε ότι όλοι αυτοί οι μηχανισμοί

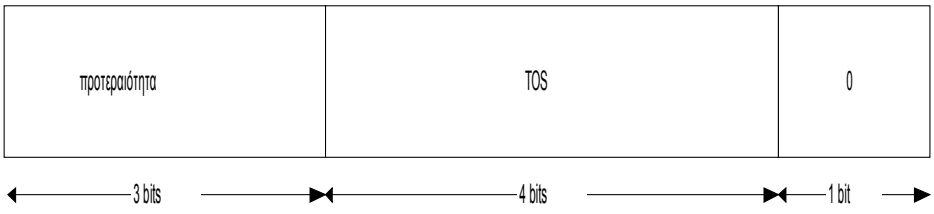
λειτουργούν στο επίπεδο δικτύου του OSI μοντέλου). Η δυσκολία έγκειται στο γεγονός ότι απαιτείται η διαδικασία της ταξινόμησης να γίνεται άμεσα και συνεπώς ο

έλεγχος τόσων πεδίων απαιτεί μεγάλη επεξεργαστική ισχύ. Η μέθοδος αυτή της ταξινόμησης εφαρμόζεται μόνο όταν θέλουμε απαραιτήτως να κάνουμε ταξινόμηση με βάση ξεχωριστές ροές όπως θέλουμε να κάνουμε πολλές φορές στην DiffServ αρχιτεκτονική στα σημεία εισόδου της κίνησης σε DiffServ enabled domains. Τέλος αυτή η μέθοδος ονομάζεται Multifield classification.

Αντίθετα στην περίπτωση όπου επιθυμούμε να κάνουμε ταξινόμηση σε συνενώσεις ροών τότε αρκεί να χρησιμοποιήσουμε ένα συνδυασμό των παραπάνω πεδίων της πεντάδας που χαρακτηρίζει μια ροή, ή ακόμη και ένα μόνο πεδίο. Η περίπτωση αυτή είναι πιο εύκολη να γίνει και μπορεί τελικά να πραγματοποιείται ταχύτατα σε σύγκριση με τον έλεγχο όλης της πεντάδας.

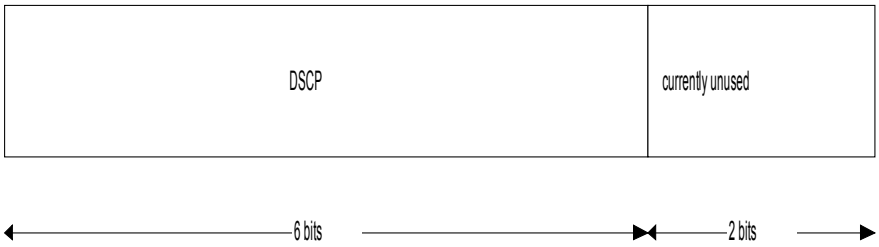
Στην πραγματικότητα ισχύει πως η ταξινόμηση των πακέτων επιθυμούμε να γίνει σε έναν περιορισμένο αριθμό κατηγοριών (κλάσεων) και συνεπώς αρκεί να χρησιμοποιήσουμε ένα σταθερό πεδίο στην επικεφαλίδα των πακέτων. Η μέθοδος αυτή είναι σαφώς απλούστερη και πιο αποδοτική και στην περίπτωση της DiffServ αρχιτεκτονικής ονομάζεται behaviour aggregate classification, και πρέπει να παρατηρήσουμε ότι η ταξινόμηση που επιτυγχάνει είναι σε επίπεδο aggregates.

Η behaviour aggregate ταξινόμηση πραγματοποιείται χρησιμοποιώντας μια οκτάδα από bits που υπάρχει την επικεφαλίδα των IPv4 πακέτων και η οποία ονομάζεται ToS octet. Σε αυτή τα τρία πρώτα bits δηλώνουν την προτεραιότητα κάθε πακέτου και συνεπώς υπάρχουν 8 διαφορετικές κλάσεις προτεραιότητας. Τα επόμενα 4 bits χαρακτηρίζουν το είδος της υπηρεσίας που επιθυμεί η εφαρμογή, δηλαδή ελαχιστοποίηση της καθυστέρησης, η ελαχιστοποίηση της απώλειας πακέτων κλπ.



Εικόνα 2 Το TOS octet της IPv4 επικεφαλίδας

Ακόμη, πρόσφατα καθορίστηκε στο ToS octet τα 6 πιο σημαντικά bits να αναπαριστούν το DiffServ Code Point το οποίο ουσιαστικά δημιουργεί 64 δυνατούς συνδυασμούς για τη διαχείριση ουρών και χρονοδρομολόγησης των IP πακέτων. Τέλος να αναφέρουμε πως αντίστοιχο πεδίο έχει οριστεί και για το πρωτόκολλο IPv6.



Εικόνα 3 Η IPv4 επικεφαλίδα σύμφωνα με την DiffServ αρχιτεκτονική

1.6. Μηχανισμοί μαρκαρίσματος, μέτρησης της κίνησης, μορφοποίησης και απόρριψης πακέτων

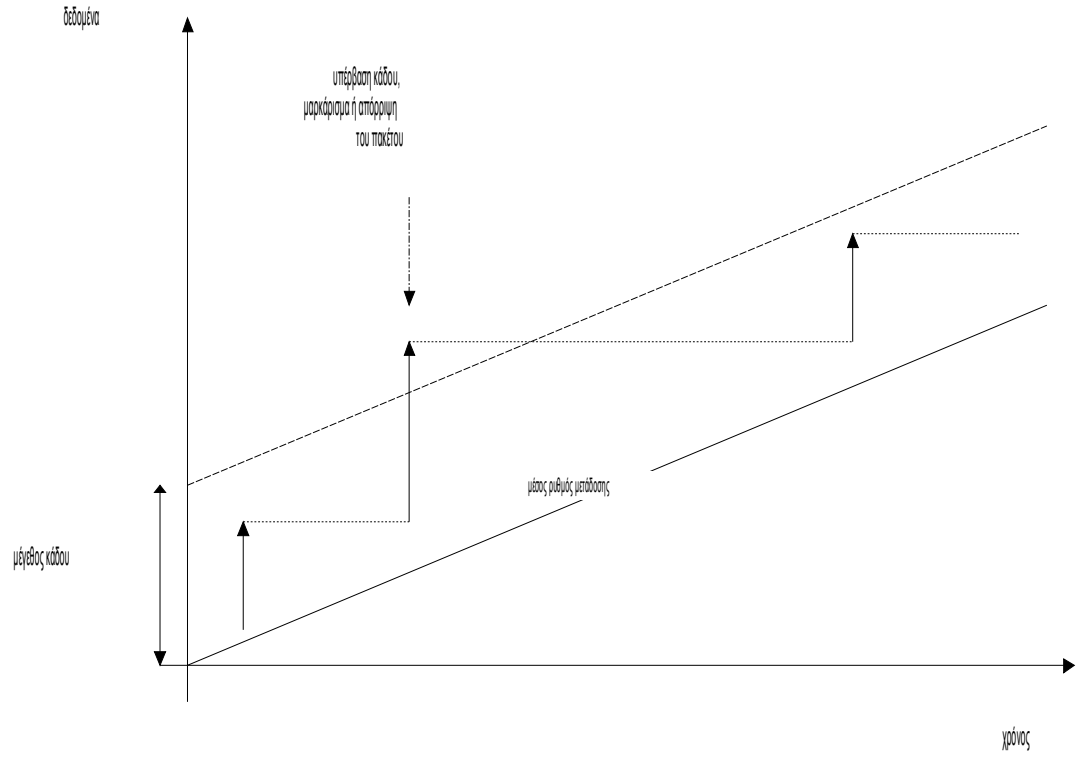
Γενικά οι μηχανισμοί μαρκαρίσματος των πακέτων, μέτρησης του προφίλ της κίνησης και μορφοποίησης ή απόρριψης της κίνησης ονομάζονται όλοι μαζί μηχανισμοί ελέγχου της κίνησης (traffic conditioning). Συνήθως οι μηχανισμοί αυτοί εφαρμόζονται στον αποστολέα, στα σημεία εισόδου της κίνησης σε κάποιο domain. Εν τούτοις, έχει αναφερθεί 1.18.3πως μπορεί ο μηχανισμός μέτρησης να βρίσκεται στον παραλήπτη με ορισμένες βέβαια προϋποθέσεις. Για να είναι αυτό εφικτό απαιτείται από το δίκτυο να υποστηρίζει σε όλους τους δρομολογητές του την λειτουργικότητα του ECN (Explicit Congestion Notification) που είναι μια λειτουργία ελέγχου συμφόρησης. Στην πράξη το ECN είναι ένα bit στην επικεφαλίδα των πακέτων που τίθεται στην τιμή 1 όταν ανιχνεύσει στο δίκτυο συμφόρηση. Με τον τρόπο αυτό ενημερώνονται οι υπόλοιποι κόμβοι από τους οποίους περνά το συγκεκριμένο πακέτο πως σε κάποιο σημείο στο δίκτυο παρατηρήθηκε συμφόρηση. Αναλυτικότερα η λειτουργικότητα του ECN περιγράφεται σε επόμενη ενότητα.

Οι μηχανισμοί ελέγχου ης κίνησης που παρουσιάζουμε στις επόμενες παραγράφους της ενότητας υποθέτουν πως το μαρκάρισμα και η μέτρηση των πακέτων γίνεται στα σημεία εισόδου στο δίκτυο.

1.6.1. Αλγόριθμοι Token Bucket και Leaky Bucket

Ένας απλός μηχανισμός για τον έλεγχο της κίνησης, που διαχωρίζει τα πακέτα σε δύο κατηγορίες, σε αυτά που είναι εντός προφίλ και αντίστροφα σε όσα είναι εκτός προφίλ είναι η εφαρμογή κάποιου από τους αλγορίθμους token ή leaky bucket 1.18.3. Η λειτουργία τους βασίζεται στην ίδια λογική αλλά επιτυγχάνουν διαφορετικά αποτελέσματα όπως θα παρουσιαστούν αμέσως.

Ο αλγόριθμος token bucket καθορίζει δύο μεταβλητές, το μέσο ρυθμό αποστολής πακέτων r και το μέγιστο μέγεθος του κάδου b. Σε αυτόν παράγονται tokens με ρυθμό ίσο με το μέσο ρυθμό που καθορίστηκε, και αν αυτά δεν χρησιμοποιούνται συσσωρεύονται στο κάδο μέχρι το πολύ b. Όταν φτάσει ένα πακέτο, αν υπάρχει ελεύθερο token, τότε θεωρείται ότι το token ανατίθεται στο πακέτο αυτό και το πακέτο χαρακτηρίζεται σαν εντός προφίλ. Αντίθετα αν φτάσει ένα πακέτο και δεν υπάρχει ελεύθερο token, τότε το πακέτο μαρκάρεται ως εκτός προφίλ έτσι ώστε αργότερα να δεχτεί ανάλογη μεταχείριση, όπως εξυπηρέτηση με ελάχιστη ποιότητα ή ακόμη και απόρριψη ανάλογα με το SLA που έχει υπογραφεί. Συμπερασματικά λοιπόν ο αλγόριθμος token bucket καθορίζει το μέσο ρυθμό μετάδοσης και επομένως επιτρέπει διακυμάνσεις του στιγμιαίου ρυθμού. Επίσης ο ρόλος του κάδου, που έχει μέγιστο μέγεθος b, είναι ιδιαίτερα σημαντικός αφού επιτρέπει να μαρκάρονται σαν κίνηση εντός προφίλ, εκρήξεις που δεν ξεπερνούν όμως την τιμή b.

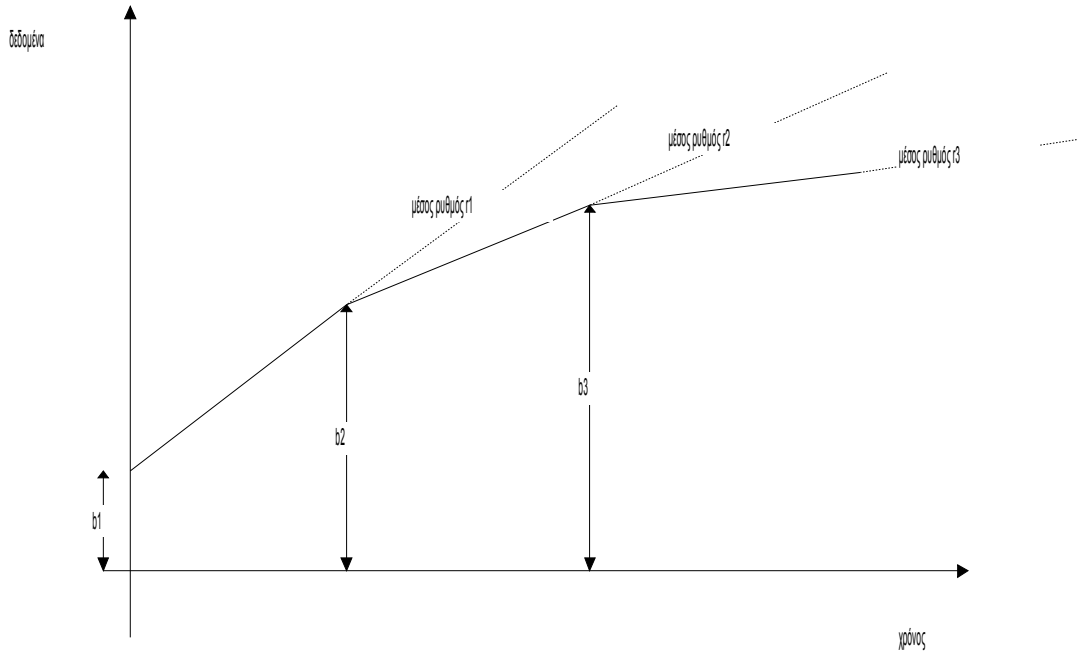


Εικόνα 4 Η λειτουργία του μηχανισμού token bucket

Ο μηχανισμός του Leaky Bucket είναι αυτός ο οποίος μας απασχόλησε στα πλαίσια αυτής της διπλωματικής εργασίας. Έτσι αναλύεται λεπτομερώς σε επόμενα

κεφάλαια και συγκεκριμένα στο 1.11.4. Αρχικά αναλύεται η δομή του ns-3 στον οποίο και προστέθηκε. Εφόσον γίνει κατανοητή η θεωρία του συγκεκριμένου αλγορίθμου παρουσιάζεται ο τρόπος με τον οποίο προστέθηκε στον προσομοιωτή καθώς και διάφορα πειράματα που επιβεβαίωναν την ορθή του λειτουργία.

Τέλος θα πρέπει να αναφέρουμε πως αν επιθυμούμε τον έλεγχο της κίνησης σε περισσότερα από 2 επίπεδα τότε μπορεί να χρησιμοποιηθεί κάποιος από τους παραπάνω αλγορίθμους διαδοχικά. Έτσι τα πακέτα κατηγοριοποιούνται σε περισσότερα επίπεδα και μπορούν στη συνέχεια τα πακέτα κάθε επιπέδου να μαρκάρονται και να εξυπηρετούνται ξεχωριστά. Η Εικόνα 5 περιγράφει ένα τέτοιο παράδειγμα.



Εικόνα 5 Ένας μηχανισμός κατηγοριοποίησης της κίνησης σε 3 επίπεδα

1.6.2. Αστυνόμευση της κίνησης

Η λειτουργία αυτή πραγματοποιείται επίσης στα σημεία εισόδου της κίνησης σε ένα DiffServ domain. Η αστυνόμευση έχει την έννοια του ελέγχου της κίνησης με βάση ένα συγκεκριμένο προφίλ που έχει συμφωνηθεί και στη συνέχεια τη λήψη συγκεκριμένων αποφάσεων για τον χειρισμό της κίνησης που ξεφεύγει από το συμφωνηθέν προφίλ. Οι αποφάσεις αυτές μπορεί να είναι είτε μαρκάρισμα των

πακέτων σε μικρότερη κλάση εξυπηρέτησης, είτε να εξυπηρετηθούν χωρίς εγγυημένη ποιότητα ή τέλος στη χειρότερη περίπτωση να απορριφθούν. Στο σημείο αυτό πρέπει να σημειωθεί πως οι αποφάσεις αυτές επίσης έχουν συμφωνηθεί εκ των προτέρων μεταξύ του πελάτη και του διαχειριστή του δικτύου (SLA). Τα κριτήρια αστυνόμευσης που χρησιμοποιούνται μπορεί να είναι με βάση τη χρονική στιγμή στη διάρκεια της μέρας, βάση της πηγής και του προορισμού ή γενικότερα με βάση κάθε δεδομένο της κίνησης.

Παράλληλα η λειτουργία της μορφοποίησης της κίνησης που περιγράφηκε προηγουμένως επιτυγχάνει να διαμορφώνει την κίνηση εξαλείφοντας εκρήξεις και επίσης μπορεί να έχει προβλεφθεί τα πακέτα που κανονικά απορρίπτονται (πακέτα εκτός προφίλ) να αποθηκεύονται προσωρινά και να διοχετεύονται αργότερα στο δίκτυο και αφού έχει εξομαλυνθεί η εκρηκτικότητα της μετάδοσής τους. Επομένως είναι δυνατό οι μηχανισμοί αστυνόμευσης και μορφοποίησης της κίνησης να χρησιμοποιηθούν συνδυασμένα ώστε ένα μέρος των πακέτων που θεωρούνται εκτός προφίλ από τον μηχανισμό αστυνόμευσης να μορφοποιείται και να μεταδίδεται. Γενικά το πεδίο αυτό είναι ανοικτό και μπορούν να παρουσιαστούν διάφοροι μηχανισμοί που συνδυάζουν μηχανισμούς αστυνόμευσης και μορφοποίησης.

1.7. Διαχείριση ουρών (Queue management)

Το θέμα της διαχείρισης των ουρών αποτελεί ένα σημαντικό και κρίσιμο ζήτημα για το διαχειριστή του δικτύου προκειμένου να είναι σε θέση να προσφέρει ποιότητα υπηρεσίας στις διάφορες ροές όπως έχει συμφωνήσει. Επίσης η διαχείριση των ουρών είναι μια βασική προϋπόθεση για τη λειτουργία του μηχανισμού της χρονοδρομολόγησης που θα περιγράψουμε στην επόμενη ενότητα. Προκειμένου το δίκτυο να ικανοποιήσει όλες τις εγγυήσεις παροχής ποιότητας υπηρεσίας πρέπει να χειρίζεται τα πακέτα κάθε κλάσης ποιότητας σε ξεχωριστή ουρά ώστε να μπορεί να εφαρμόζει τον κατάλληλο μηχανισμό χρονοδρομολόγησης. Σε αντίθετη περίπτωση δεν είναι δυνατό ο μηχανισμός χρονοδρομολόγησης να διαχωρίσει τις διαφορετικές κλάσεις ποιότητας και να προσφέρει επομένως τις κατάλληλες εγγυήσεις στις αντίστοιχες ροές. Πιο αναλυτικά, αναφέροντας ένα παράδειγμα, αν δεν γίνει διαχωρισμός των κλάσεων ποιότητας σε διαφορετικές ουρές, θα συσσωρεύονται στην ίδια ουρά ροές με διαφορετικές απαιτήσεις με αποτέλεσμα είτε πακέτα να απορρίπτονται (αν γεμίσει η ουρά) είτε να παρουσιάζεται μεγάλη καθυστέρηση. Συνέπεια όλων αυτών είναι το δίκτυο να μην μπορεί να παρέχει τις καλύτερες

εγγυήσεις και αντίστοιχα η απόδοση που επιτυγχάνουν οι εφαρμογές των πελατών να υποβαθμίζεται σημαντικά.

Οι κατεξοχήν λειτουργίες του διαχειριστή των ουρών παρουσιάζονται αμέσως παρακάτω και επιγραμματικά συνοψίζονται στην σωστή λειτουργία των ουρών και στη χρήση μηχανισμών για τον έλεγχο τους.

- Είσοδος ενός πακέτου στη σωστή ουρά με βάση τη κατηγοριοποίηση του πακέτου από τον αντίστοιχο μηχανισμό.
- Απόρριψη ενός πακέτου στην περίπτωση που η ουρά που πρέπει να εισαχθεί είναι γεμάτη.
- Απομάκρυνση ενός πακέτου από την κορυφή της ουράς όταν το ζητήσει ο χρονοδρομολογητής προκειμένου να μεταδοθεί στον επόμενο κόμβο.
- Έλεγχος της κατάστασης της ουράς, δηλαδή της μέσης πληρότητάς της και ανάληψη πρωτοβουλιών ανάλογα με αυτή την τιμή, με στόχο τη διατήρηση της μέσης πληρότητας σε χαμηλά επίπεδα. Οι πρωτοβουλίες που μπορεί να αναλάβει είναι οι ακόλουθες:
    - Αφαίρεση ενός πακέτου από την ουρά και απόρριψή του στην περίπτωση που η ουρά έχει αρχίσει να γεμίζει.
    - Μαρκάρισμα ενός πακέτου όταν η ουρά παρουσιάζει μεγάλη πληρότητα (ECN).

Γενικά λοιπόν παρατηρείται ότι εκτός από τις κλασικές λειτουργίες υποδοχής και αποχώρησης ενός πακέτου, ο διαχειριστής μιας ουράς ενδιαφέρεται και για την αποδοτική λειτουργίας της που εξασφαλίζεται κυρίως μέσα από τη διατήρηση σε χαμηλά επίπεδα της μέσης πληρότητάς της. Αυτό δικαιολογείται από το γεγονός ότι διατηρώντας χαμηλά τη μέση πληρότητα τότε οι ουρές μπορούν να απορροφούν εύκολα εκρήξεις της κίνησης. Αντίθετα, αν η μέση πληρότητα ήταν υψηλή τότε πλήθος πακέτων κατά τη διάρκεια εκρήξεων θα απορρίπτονταν. Επίσης η μικρή πληρότητα μιας ουράς συνεπάγεται πως η μέση καθυστέρηση εξυπηρέτησης θα παραμένει χαμηλή, γεγονός που είναι ιδιαίτερα επιθυμητό.

Το θέμα της διαχείρισης των ουρών γίνεται ακόμα επιτακτικότερο και πιο κρίσιμο ειδικά σε καταστάσεις συμφόρησης του δικτύου όπου πρέπει πλέον οι ουρές να αντιδράσουν σωστά και άμεσα. Το κυριότερο πρόβλημα είναι πως να προσδιοριστούν συγκεκριμένες στρατηγικές αποφάσεις για την ανάληψη δράσεων. Μια από αυτές τις αποφάσεις είναι πότε αποφασίζεται να απορρίπτονται πακέτα, δηλαδή αν απορρίπτονται πακέτα μόλις φτάσουν στην ουρά ή επιτρέπεται να απορρίπτονται πακέτα που βρίσκονται μέσα στην ουρά προκειμένου να εξυπηρετηθούν άλλα

μεγαλύτερης προτεραιότητας. Επίσης κρίσιμη απόφαση είναι με βάση ποια κριτήρια και πληροφορίες απορρίπτονται τα πακέτα, αφού μπορεί να κρατούνται γενικές πληροφορίες για όλη την κίνηση ή αντίθετα για κάθε είδος κίνησης ξεχωριστά.

Συμπερασματικά, οι παραπάνω στρατηγικές αποφάσεις για τη λειτουργία της διαχείρισης ουρών επηρεάζουν άμεσα την απόδοση των ίδιων των ουρών και κατ' επέκταση όλου του δικτύου. Γενικό στόχο αποτελεί η δίκαια διαχείριση των ουρών για όλες τις κλάσεις ποιότητας χωρίς σε καμία περίπτωση να παραβούμε τις συμφωνίες που έχουν υπογραφεί με τους πελάτες του δικτύου.

Επιστρέφοντας στο θέμα της συμφόρησης πρέπει να τονιστεί πως η ύπαρξη συμφόρησης στο δίκτυο συνεπάγεται και αύξηση του μέσου μεγέθους των ουρών. Για την αποφυγή της συμφόρησης υπάρχουν συγκεκριμένοι μηχανισμοί από το πρωτόκολλο TCP στο επίπεδο μεταφοράς σύμφωνα με το OSI μοντέλο. Παράλληλα με αυτούς, οι διαχειριστές έχουν αναπτύξει και δικούς τους μηχανισμούς που θα παρουσιαστούν παρακάτω. Γενικά οι μηχανισμοί αυτοί προσπαθούν να βελτιώσουν το πρόβλημα της συμφόρησης που παρατηρείται στο δίκτυο και οφείλεται σε αποστολή πακέτων με ρυθμό υψηλότερο από αυτό που μπορεί να αντιμετωπίσει το δίκτυο και δεν οφείλεται σε μικροεκρήξεις παροδικού χαρακτήρα. Οι μηχανισμοί που μπορεί να χρησιμοποιήσει ο διαχειριστής του δικτύου είναι:

- Απόρριψη των πακέτων. Ο μηχανισμός αυτός έχει διπλό αποτέλεσμα καθώς αφενός μειώνει άμεσα το φόρτο του δικτύου και αφετέρου ενημερώνει άμεσα το πρωτόκολλο TCP για συμφόρηση. Αυτό επιτυγχάνεται αφού το TCP θεωρεί ότι κάθε απώλεια πακέτου οφείλεται σε συμφόρηση και στη συνέχεια ενεργοποιεί αυτόματα το μηχανισμό του για την αποφυγή συμφόρησης.
- Μαρκάρισμα των πακέτων. Η δεύτερη αυτή μέθοδος είναι λιγότερη καταστροφική από την πρώτη αφού δεν απορρίπτει πακέτα αλλά και λιγότερο άμεση αφού το δίκτυο δεν «αποφορτίζεται» άμεσα.

Στη συνέχεια περιγράφονται τεχνικές και μηχανισμοί που ανήκουν στη δεύτερη κατηγορία.

1.7.1. Explicit Congestion Notification (ECN)

Η μέθοδος αυτή 1.18.3 στηρίζεται στα δύο αχρησιμοποίητα bits του πεδίου DSCP (DiffServ Code Point), τα οποία πλέον ονομάζονται ECN Capable Transport (ECT) και Congestion Experienced (CE) αντίστοιχα. Αυτός ο μηχανισμός ελέγχεται από τα πρωτόκολλα του επιπέδου μεταφοράς και η λειτουργία του είναι απλή. Τα δύο αυτά bits επιτελούν συγκεκριμένες λειτουργίες:

- Το bit ECT τίθεται στην τιμή 1 αν τα άκρα μιας ροής που μεταδίδεται κατανοούν την λειτουργία του bit CE και κατ' επέκταση του όλου αυτού μηχανισμού.
- Το bit CE τίθεται στην τιμή 1 όταν κάποιος δρομολογητής επιθυμεί να ειδοποιήσει για συμφόρηση και το bit ECT είναι ενεργοποιημένο.

Συνεπώς σε κάθε πακέτο το ECT είναι 1 όταν και οι δύο άκρες της ροής κατανοούν τη λειτουργία του μηχανισμού. Κάθε δρομολογητής αν θέλει να ειδοποιήσει για συμφόρηση θέτει το CE στην τιμή 1 αν το ECT είναι ενεργοποιημένο αλλιώς απορρίπτει το πακέτο. Ουσιαστικά με τη μέθοδο αυτή ειδοποιούνται τα πρωτόκολλα με μη καταστροφικό τρόπο αν κατανοούν τη μέθοδο αυτή και σε αντίθετη περίπτωση (δεν καταλαβαίνουν τη μέθοδο) κατανοούν τη συμφόρηση από την απόρριψη του πακέτου. Ένα κρίσιμο σημείο στη μέθοδο αυτή είναι πότε ο δρομολογητής αποφασίζει να ειδοποιήσει για συμφόρηση, αφού σε περιπτώσεις παροδικής συμφόρησης λόγω μικροεκρήξεων της κίνησης, δεν είναι αποδοτικό να ειδοποιείται το πρωτόκολλο καθώς τότε θα υποβαθμιστεί η απόδοση του δικτύου χωρίς λόγο.

1.7.2. Μηχανισμός RED (Random Early Detection)

Ένας δεύτερος μηχανισμός που ειδοποιεί τα πρωτόκολλα για ενδεχόμενη συμφόρηση είναι ο RED. Στους μηχανισμούς αποφυγής συμφόρησης σημαντικό πρόβλημα αποτελεί ο καθορισμός πότε θα αποστέλλεται ειδοποίηση για συμφόρηση και πόσο έντονη αυτή θα είναι. Παράλληλα το θέμα αυτό σχετίζεται και με την διαμόρφωση των ουρών, πως έχουν δηλαδή οριστεί ώστε να είναι ξεχωριστές για κάθε ροή ή εάν επιτρέπεται aggregates να περνούν από την ίδια ουρά. Με το θέμα αυτό ασχολήθηκε για αρκετά χρόνια η IRTF (Internet Research Task Force) που κατέληξε να προτείνει τον μηχανισμό RED ο οποίος στέλνει ειδοποιήσεις για συμφόρηση τυχαία και η συχνότητα με την οποία στέλνονται αυτές εξαρτάται από τη μέση πληρότητα της ουράς 1.18.3.

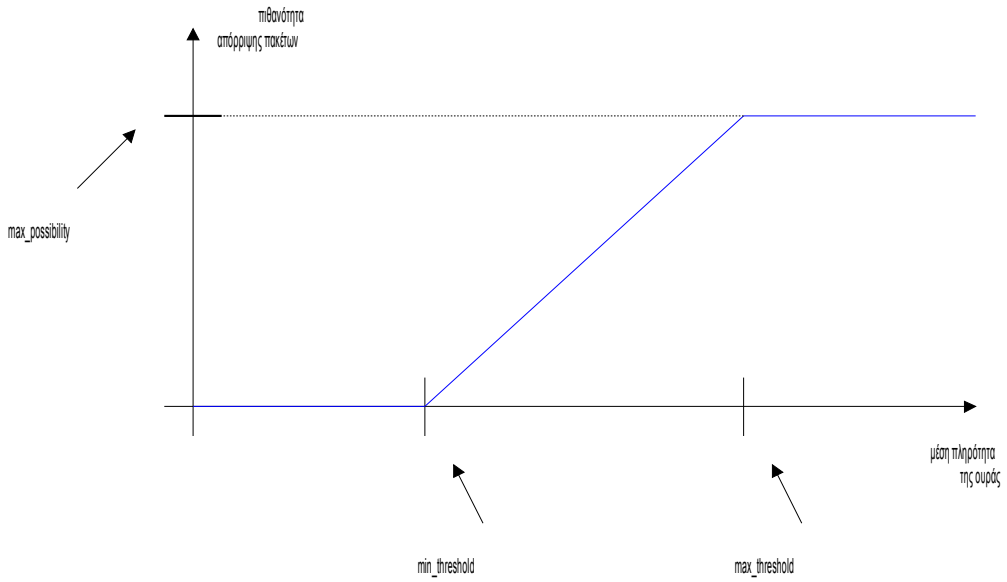
Βασική παράμετρος με βάση την οποία αποφασίζει ο μηχανισμός αυτός είναι η μέση πληρότητα της ουράς. Ο τρόπος με τον οποίο ειδοποιεί τα πρωτόκολλα για συμφόρηση είναι έμμεσος καθώς αυτό γίνεται με απόρριψη πακέτων. Σε κάθε ουρά που εφαρμόζεται ο RED ορίζονται τρία μεγέθη:

- To min threshold
- To max threshold
- To max possibility

Έτσι ο μηχανισμός λειτουργεί ως εξής:

- Εάν η μέση πληρότητα είναι μικρότερη από την τιμή min_threshold τότε όλα τα πακέτα διέρχονται κανονικά και δεν έχουμε καμία απόρριψη.
- Εάν η μέση πληρότητα είναι μεγαλύτερη από την τιμή min_threshold και μικρότερη από την τιμή max_threshold τότε η πιθανότητα απόρριψης αυξάνει γραμμικά από 0 έως την τιμή max_possibility.
- Τέλος αν η μέση πληρότητα ξεπερνά την τιμή του max_threshold, τότε όλα τα πακέτα απορρίπτονται.

Οι τρεις αυτές καταστάσεις στις οποίες μπορεί να βρίσκεται μια ουρά ονομάζονται αντίστοιχα κανονική, αποφυγής συμφόρησης και ελέγχου συμφόρησης. Κρίσιμο παράγοντα για τη λειτουργία του αλγορίθμου αποτελεί η σωστή εκτίμηση της μέσης πληρότητας της ουράς. Αυτή υπολογίζεται κάθε φορά που ένα πακέτο εισέρχεται στην ουρά, και ο υπολογισμός της γίνεται με τη χρήση ενός κατωπερατού φίλτρου. Επίσης ο RED έχει προβλέψει και την περίπτωση όπου μεσολαβεί μεγάλο χρονικό διάστημα μεταξύ δύο απορρίψεων πακέτων και εν τω μεταξύ να έχει παρουσιαστεί συμφόρηση. Αυτό μπορεί να παρατηρηθεί εξαιτίας του γεγονότος ότι η απόρριψη πακέτων γίνεται πιθανοτικά. Λύση σε αυτό το θέμα δίνει ένας μετρητής που χρησιμοποιείται και μετρά τον αριθμό των πακέτων που πέρασαν από την ουρά χωρίς απόρριψη. Έτσι η πιθανότητα απόρριψης πολλαπλασιάζεται τώρα και με την ποσότητα 1/1-c όπου c ο μετρητής αυτός.



Εικόνα 6 Η λειτουργία του μηχανισμού RED

Γενικά ο μηχανισμός αυτός είναι ιδιαίτερα αποδοτικός, αλλά παρουσιάζει σημαντική δυσκολία στη ρύθμιση των παραμέτρων του. Το σημείο που πρέπει να τονιστεί είναι ότι απαιτείται να επιτρέπει να περνούν μικροεκρήξεις χωρίς απόρριψη πακέτων, ενώ αντίθετα θα πρέπει να αντιδρά άμεσα σε περιπτώσεις παρατεταμένης αύξησης της μέσης πληρότητας της ουράς. Τέλος προκειμένου να είναι πραγματικά αποδοτικός ο αλγόριθμος πρέπει πραγματικά να απορρίπτει πακέτα (άρα και να ειδοποιεί για συμφόρηση) από τις ροές που δημιουργούν το πρόβλημα.

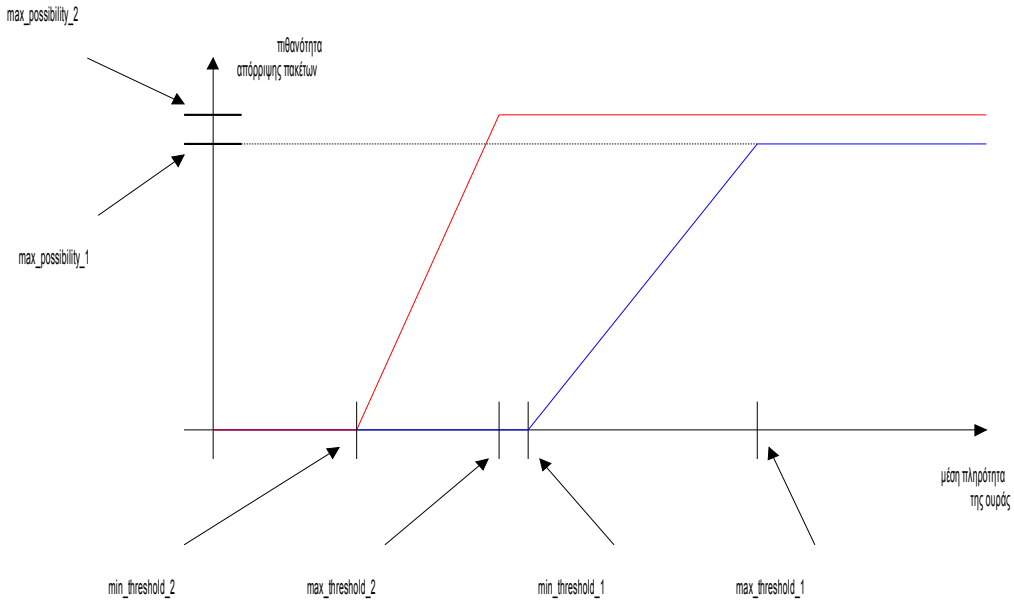
Στη συνέχεια παρουσιάστηκαν διάφορες παραλλαγές του RED που προσπαθούσαν να εξαλείψουν ορισμένα μειονεκτήματα του. Τέτοιες ήταν ο Adaptive και ο Flow RED. Πάντως κυριότερη παραλλαγή του αποτελεί ο Weighted RED που περιγράφεται στην επόμενη ενότητα.

1.7.3. Μηχανισμός Weighted RED

Ο Weighted RED αποτελεί μια σημαντική παραλλαγή του κλασσικού RED μηχανισμού αφού επιτρέπει να συμπεριφέρεται η ουρά με διαφορετικό τρόπο (εφαρμόζοντας διαφορετικά κριτήρια) στα πακέτα μιας ροής. Στην πραγματικότητα δίνει τη δυνατότητα να μεταχειρίζεται τα πακέτα της ίδιας ροής με διαφορετικό τρόπο μεταξύ τους ανάλογα βέβαια με κάποιο κριτήριο.

Ο μηχανισμός αυτός λειτουργεί όπως ο απλός RED με τη διαφορά ότι σε αυτόν ορίζονται περισσότερες τριάδες μεταβλητών (Min_threshold, max_threshold, max_possibility), ίσες σε αριθμό με τον αριθμό των διαφορετικών τρόπων χειρισμού

πακέτων που ζητείται. Ένα παράδειγμα αποτελεί να οριστούν δύο επίπεδα τιμών, με το δεύτερο αυστηρότερο από το πρώτο και να διαχειρίζονται με βάση το πρώτο τα πακέτα που καταφθάνουν κανονικά στην ουρά, ενώ αντίθετα με το δεύτερο επίπεδο τιμών τα πακέτα που είχαν μαρκαριστεί εκτός προφίλ σε προηγούμενο δρομολογητή.



Εικόνα 7 Η λειτουργία του μηχανισμού Weighted RED

Ολοκληρώνοντας την παρουσίαση του Weighted RED να επισημανθεί ότι είναι ιδιαίτερα χρήσιμος μηχανισμός που έχει ευρεία χρήση σήμερα.

1.8. Χρονοδρομολόγηση

Το επόμενο σημαντικό ζήτημα στη προσπάθεια ενός δικτύου να παρέχει εγγυήσεις ποιότητας είναι το θέμα της χρονοδρομολόγησης. Αναλύοντας την έννοια αυτή, σημαίνει ο τρόπος με τον οποίο χειρίζεται το δίκτυο τις ουρές, δηλαδή ποια ουρά στέλνει δεδομένα και για πόσο χρόνο. Ουσιαστικά ο μηχανισμός αυτός έχει στην διάθεσή του το σύνολο των ουρών που έχει ένας δρομολογητής και αποφασίζει με ποια σειρά θα μεταδώσουν πακέτα και για πόσο διάστημα η καθεμία.

Ο ρόλος του χρονοδρομολογητή είναι ιδιαίτερα κρίσιμος για ένα δίκτυο όταν επιθυμεί να προσφέρει και να παρέχει εγγυήσεις ποιότητας. Αυτό δικαιολογείται από το γεγονός ότι η λειτουργία του δρομολογητή καθορίζει την καθυστέρηση σε κάθε ουρά και τον τρόπο που διαμοιράζεται η γραμμή μετάδοσης μεταξύ των ουρών. Στην

πραγματικότητα ο μηχανισμός του χρονοδρομολογητή είναι αυτός που καθορίζει το είδος της ποιότητας που παρέχει το δίκτυο. Οι παράμετροι αναλυτικά που μπορεί και επηρεάζει είναι:

- Η χωρητικότητα κάθε ροής, αφού μπορεί και ελέγχει κάθε πότε η ροή αυτή θα μεταδίδει.
- Την καθυστέρηση κάθε ροής, αφού ελέγχει όπως αναφέρθηκε παραπάνω το ρυθμό με τον οποίο κάθε ροή μεταδίδει, άρα καθορίζει και το χρονικό διάστημα που τα πακέτα παραμένουν στην ουρά.
- Επίσης ο χρονοδρομολογητής καθορίζει και το jitter, που είναι η διαφορά στην καθυστέρηση μεταξύ 2 διαδοχικών πακέτων.

Γενικά λοιπόν αυτοί είναι οι κύριοι παράμετροι που μπορεί να επηρεάσει ο χρονοδρομολογητής και συνεπώς προδιαγράφει την ποιότητα υπηρεσίας που μπορεί να παρέχει το δίκτυο. Επομένως εξαιτίας της σπουδαιότητας του μηχανισμού αυτού και του γεγονότος ότι οι μηχανισμοί χρονοδρομολόγησης που υπάρχουν είναι αρκετοί, είναι αναγκαίο η επιλογή του καταλληλότερου να γίνεται προσεκτικά και με βάση ορισμένα κριτήρια. Είναι απαραίτητο στην επιλογή του μηχανισμού χρονοδρομολόγησης να ελέγχεται πρώτα το είδος των εγγυήσεων που παρέχει και το βαθμό επιτυχίας του, έτσι ώστε να ταιριάζει απόλυτα στη φύση των εφαρμογών που θα υποστηρίξει.

Στη συνέχεια της ενότητας παρουσιάζονται ορισμένοι τέτοιοι μηχανισμοί, ο τρόπος λειτουργίας τους, το είδος των εγγυήσεων που παρέχουν και τα μειονεκτήματά τους.

1.8.1. FIFO

Ο πρώτος μηχανισμός χρονοδρομολόγησης είναι ο λεγόμενος FIFO και είναι ο παλαιότερος που υπάρχει. Ο μηχανισμός αυτός υποθέτει ότι υπάρχει μόνο μία ουρά και η λογική του είναι ότι εξέρχεται από την ουρά το πρώτο πακέτο που μπήκε, δηλαδή κάθε φορά το παλαιότερο πακέτο μέσα στην ουρά. Ο μηχανισμός αυτός όπως γίνεται σαφές αντιμετωπίζει όλα τα πακέτα όμοια και δεν χρησιμοποιεί καμία έννοια προτεραιότητας.

Ο μηχανισμός αυτός έχει ως πλεονέκτημά του μόνο την απλότητά του και μπορεί να βρει εφαρμογή σε περιπτώσεις γραμμών μετάδοσης πολύ μεγάλης ταχύτητας όπου δεν υπάρχει καθόλου συμφόρηση, και ειδικότερα όταν η χρησιμοποίηση της γραμμής είναι πολύ χαμηλή. Αντιθέτως σε περιπτώσεις συμφόρησης έχει πολύ κακή απόδοση.

Επίσης αντίστοιχα κακή απόδοση παρουσιάζει και στις περιπτώσεις όπου υπάρχουν εφαρμογές με καταιγισμούς που μπορεί να καταλαμβάνουν όλη την ουρά και να απορρίπτονται πακέτα άλλων εφαρμογών. Γενικά ο μηχανισμός αυτός είναι απλός και δεν ενδείκνυται για δρομολογητές που πρέπει να παρέχουν εγγυήσεις ποιότητας.

1.8.2. Priority Queueing (PQ)

Ένας δεύτερος μηχανισμός χρονοδρομολόγησης είναι ο Priority Queueing ο οποίος σε αντίθεση με το μηχανισμό FIFO επιτρέπει διαφορετικές προτεραιότητες και μπορεί να χειριστεί πλήθος ουρών. Η λογική του είναι ότι μια ουρά έχει αυστηρή προτεραιότητα και πάντοτε εξυπηρετείται όταν έχει πακέτα σε βάρος των υπολοίπων ουρών. Ουσιαστικά λοιπόν ο μηχανισμός αυτός συμπεριφέρεται προνομιακά στην ουρά υψηλής προτεραιότητας αφού πάντοτε εκείνη μεταδίδει άμεσα και υποβαθμίζει τις υπόλοιπες.

Στην πράξη, τα πακέτα ανάλογα με την ταξινόμηση που έχουν δεχτεί εισέρχονται στην αντίστοιχη ουρά. Τότε ο PQ ελέγχει πάντοτε τις ουρές με τη σειρά, αρχίζοντας από την ουρά μεγαλύτερης προτεραιότητας και προχωρώντας προς την ουρά μικρότερης προτεραιότητας έως ότου βρει μια ουρά που έχει πακέτο και το μεταδίδει. Στη συνέχεια επαναλαμβάνει ξανά την ίδια διαδικασία.

Ο μηχανισμός αυτός εισάγει στο δίκτυο ένα είδος αδικίας καθώς ανάλογα με το ρυθμό που καταφτάνουν πακέτα στην ουρά υψηλής προτεραιότητας, μπορεί οι άλλες ουρές είτε να εξυπηρετούνται ελάχιστα είτε ακόμη και καθόλου. Το τελευταίο που μπορεί να συμβεί αν ο ρυθμός με τον οποίο εισέρχονται πακέτα στην ουρά υψηλής προτεραιότητας ισούται με το ρυθμό μετάδοσης πάνω στο link. Αυτό μπορεί να διορθωθεί είτε με αστυνόμευση, είτε εφαρμόζοντας μορφοποίηση στην κίνηση υψηλής προτεραιότητας σε κάποιο προηγούμενο σημείο της διαδρομής.

Επικεντρώνοντας τώρα στα θετικά σημεία αυτού του μηχανισμού πρέπει να αναφερθεί ότι μπορεί και προσφέρει πολύ χαμηλή καθυστέρηση στα πακέτα που ανήκουν στην ουρά υψηλής προτεραιότητας. Σε αυτή την περίπτωση αν η ουρά αυτή δεν έχει πακέτα, μόλις θα φτάσει ένα, τότε θα περιμένει μέχρι να μεταδοθεί το πακέτο που μεταδίδεται εκείνη τη στιγμή. Στη περίπτωση που η ουρά έχει πακέτα θα περιμένει μέχρι να σταλούν τα προηγούμενα, κάτι που εξαρτάται από το μέγεθος αυτών και το bandwidth του link.

Γενικά ο μηχανισμός αυτός είναι σημαντικός αφού προσφέρει χαμηλή καθυστέρηση και βρίσκει αρκετές εφαρμογές στην πράξη.

1.8.3. Modified Deficit Round Robin (M-DRR)

Στη συνέχεια ένας άλλος μηχανισμός χρονοδρομολόγησης, με πολλές επιλογές χρήσης και ιδιαίτερα αποδοτικός είναι ο M-DRR. Αυτός στηρίζεται στη λογική του Deficit Round Robin (DRR) 1.18.3 και του Round Robin (RR) 1.18.3, όπου κρίνεται σκόπιμο να περιγραφούν εν συντομία. Ο Round Robin αλγόριθμος χειρίζεται όλες τις ουρές όμοια και τις ελέγχει κυκλικά. Σε όποια βρει πακέτο στην αναμονή το μεταδίδει και συνεχίζει τον κυκλικό έλεγχο. Η μέθοδος συνεπώς αυτή έχει το μειονέκτημα ότι δεν μπορεί να παρέχει εγγυήσεις για την καθυστέρηση. Η μέθοδος DRR αποτελεί μια παραλλαγή της απλής μεθόδου Round Robin, όπου τώρα οι ουρές ελέγχονται ξανά κυκλικά αλλά προσπαθούν να διατηρούν σταθερό το μέσο ρυθμό μετάδοσης. Αυτό επιτυγχάνεται με την ακόλουθη τεχνική, σε κάθε ουρά ορίζονται δύο ποσότητες, το κβάντο Q και το έλλειμμα D. Το κβάντο είναι ο μέγιστος αριθμός bytes που μπορεί να μεταδώσει η ουρά κάθε φορά. Αν μεταδώσει λιγότερα τότε η διαφορά αποθηκεύεται στο έλλειμμα και προστίθεται στην μέγιστη τιμή bytes που θα μεταδώσει την αμέσως επόμενη φορά.

Ο μηχανισμός M-DRR 1.18.3 λειτουργεί ουσιαστικά όπως ο DRR με τη διαφορά ότι εισάγει και μια ουρά προτεραιότητας ώστε να επιτυγχάνει χαμηλή καθυστέρηση. Οι υπόλοιπες ουρές εξυπηρετούνται με τη σειρά σύμφωνα με τον μηχανισμό DRR και η ουρά προτεραιότητας είτε εξυπηρετείται εναλλάξ με τις άλλες είτε κατά απόλυτη προτεραιότητα. Υπάρχουν δύο παραλλαγές του M-DRR όπου το σημείο που διαφέρουν είναι πόσο συχνά εξυπηρετείται η ουρά προτεραιότητας. Αναλυτικότερα αυτές είναι:

- Alternate Priority

  Στη μέθοδο αυτή η ουρά προτεραιότητας εξυπηρετείται εκ περιτροπής με τις υπόλοιπες ουρές, οι οποίες εξυπηρετούνται με τη σειρά. Για παράδειγμα, εξυπηρετείται η ουρά προτεραιότητας, ύστερα η πρώτη από τις άλλες, μετά πάλι η ουρά προτεραιότητας, ύστερα η δεύτερη κοκ.

- Strict Priority

  Αντίθετα στη μέθοδο αυτή η ουρά προτεραιότητας εξυπηρετείται κατά απόλυτη προτεραιότητα για όσο διάστημα έχει πακέτα προς μετάδοση. Όταν δεν έχει, εξυπηρετούνται οι υπόλοιπες σύμφωνα με τον DRR μηχανισμό.

Γενικά λοιπόν η μέθοδος Modified-Deficit Round Robin είναι ευέλικτη και αποδοτική μόνο όμως σε περιπτώσεις όπου δεν υπάρχει μεγάλη συμφόρηση. Γι' αυτό ακριβώς το λόγο προτείνεται να χρησιμοποιείται παράλληλα με ένα μηχανισμό διαχείρισης ουρών που προλαμβάνει τη συμφόρηση, όπως αυτοί που περιγράφηκαν στην προηγούμενη ενότητα.

1.9. Υπηρεσίες DiffServ

1.9.1. EF-based υπηρεσίες

Οι υπηρεσίες EF-based έχουν σαν στόχο να προσομοιώσουν τις εικονικές μισθωμένες γραμμές για την εξυπηρέτηση κίνησης που προέρχεται από συνένωση ροών, όπως προκύπτει από την εφαρμογή των μηχανισμών ταξινόμησης και μαρκαρίσματος. Οι υπηρεσίες αυτές απευθύνονται σε εφαρμογές πραγματικού χρόνου ευαίσθητες στο jitter και στην απώλεια πακέτων που απαιτούν εγγυήσεις χωρητικότητας. Ο ορισμός τους βασίζεται στα ακόλουθα σημεία:

- Στον καθορισμό της διεπαφής μεταξύ diffServ domains έτσι ώστε και αυτό να εμφανίζει συμπεριφορά EF-based.
- Στον περιορισμό της χωρητικότητας που αφιερώνεται στην υπηρεσία για να αποφευχθεί αποκλεισμός της best effort κίνησης
- Σε μια αρχική στατική ρύθμιση των δικτυακών συσκευών για την παροχή της υπηρεσίας.
- Στην ελαχιστοποίηση των μηχανισμών που χρησιμοποιούνται ανά κόμβο.

Τα παραπάνω σημεία αποτελούν ένα συνοπτικό ορισμό των EF-based υπηρεσιών. Μια από αυτές είναι η IP Premium όπου ορίζεται αναλυτικά στο 1.18.3. Η υπηρεσία αυτή παρέχεται σε συνενώσεις IP κίνησης. Οι μηχανισμοί που απαιτούνται για την υλοποίηση της είναι αρχικά ένας μηχανισμός αποδοχής κλήσης, όπου συνήθως η αποδοχή κλήσης γίνεται με βάση τα SLAs. Επίσης απαιτείται ένας μηχανισμός ταξινόμησης σε συνενώσεις όπου γίνεται με βάση τις διευθύνσεις πηγής και προορισμού. Σύμφωνα με τον ορισμό της IP Premium υπηρεσίας αν τα πακέτα περιέχουν διευθύνσεις που εξυπηρετούνται από την υπηρεσία αυτή εντάσσονται στο aggregate της κίνησης στο interface εισόδου του router. Στην συνέχεια και αφού απομακρυνόμαστε από την πηγή η ταξινόμηση γίνεται αποκλειστικά με βάση το DSCP στην επικεφαλίδα του πακέτου. Η τιμή για το DSCP που συνιστάται για την υπηρεσία IP Premium είναι η τιμή 101110. Τέλος αν εμφανίζονται κοντά στην πηγή τους πακέτα να περιέχουν το IP premium DSCP αλλά μη συμβατό ζεύγος πηγής προορισμού απορρίπτονται σαν μη νόμιμα.

Η συνένωση IP Premium ροής που μπαίνει σε ένα DiffServ domain αρχικά υπόκειται σε μηχανισμό αστυνόμευσης. Αν η ροή βρίσκεται στο domain του χρήστη, τότε η αστυνόμευση γίνεται με βάση ένα token bucket και τις παραμέτρους του SLA μεταξύ του χρήστη και του δικτύου πρόσβασης. Επίσης σε αυτό το σημείο ο χρήστης μπορεί να μορφοποιήσει και την κίνηση του αφού η υπηρεσία αυτή δεν επιτρέπει

μορφοποίηση σε άλλο σημείο του δικτύου. Η διαδικασία της μορφοποίησης θεωρείται σημαντική καθώς οι buffers στις δικτυακές συσκευές κατά μήκος του μονοπατιού που ακολουθείται είναι περιορισμένοι με αποτέλεσμα αν δεν έχει μορφοποιηθεί επαρκώς η κίνηση, τότε αυτοί να γεμίζουν και τα πακέτα να απορρίπτονται. Μια πρόταση που έχει παρουσιαστεί είναι να γίνεται μορφοποίηση από την ίδια την πηγή.

Επίσης στον ορισμό της IP Premium υπηρεσίας αναφέρεται ότι ο μηχανισμός αστυνόμευσης πρέπει να στηρίζεται σε ένα token bucket με βάθος μερικά πακέτα και χωρητικότητα ελαφρά μεγαλύτερη από αυτή που εγγυάται η υπηρεσία ότι θα εξυπηρετηθεί, προκειμένου να απορροφώνται και κάποιες μικρές αλλοιώσεις στα χαρακτηριστικά της κίνησης. Επειδή αποτελεί στόχο η ελαχιστοποίηση των μηχανισμών που χρησιμοποιούνται ανά κόμβο έχει προταθεί η αστυνόμευση να γίνεται μόνο στους κόμβους εισόδου σε DiffServ domains. Επίσης αυτό μπορεί να αποφευχθεί αν η κίνηση προέρχεται από κάποιο άλλο «έμπιστο» domain.

Για την υλοποίηση της IP Premium υπηρεσίας είναι απαραίτητο να χρησιμοποιείται αυστηρή κατά προτεραιότητα χρονοδρομολόγηση προκειμένου να επιτύχουμε την ελάχιστη δυνατή καθυστέρηση των πακέτων των IP συνενώσεων. Επίσης η υπηρεσία αυτή δεν απαιτεί μηχανισμούς διαχείρισης ουρών αφού προφανώς έχουν πολύ μικρό μήκος. Τέλος απαραίτητοι θεωρούνται οι μηχανισμοί διάδοσης των κανόνων εφαρμογής της υπηρεσίας και οι μηχανισμοί παρακολούθησης και καταγραφής της λειτουργίας της υπηρεσίας.

Μια δεύτερη υπηρεσία EF-based είναι η λεγόμενη QBone Premium η οποία ορίστηκε από τη ομάδα QBone του Internet2. Η υπηρεσία αυτή έχει σαν στόχο την παροχή εγγυήσεων χωρητικότητας χωρίς απώλειες πακέτων, με την ελάχιστη δυνατή καθυστέρηση και jitter σε κίνηση με περιορισμένο μέγιστο ρυθμό που διατρέχει πολλαπλά domains. Σε αυτή την υπηρεσία κρίσιμος παράγοντας αποτελεί επίσης η χρονοδρομολόγηση των πακέτων όπου πρέπει να είναι αυστηρής προτεραιότητας. Η πηγή καθορίζει δύο παραμέτρους, το μέγιστο ρυθμό και το μέγιστο μέγεθος έκρηξης και όταν η κίνηση ξεπερνά αυτό το προφίλ τότε τα πακέτα απορρίπτονται στο σημείο εισόδου που διαπιστώνεται η παράβαση.

Επικεντρώνοντας το ενδιαφέρον στις διαφορές της QBone Premium υπηρεσίας σε σχέση με την IP Premium διαπιστώνεται ότι η κυριότερη διαφορά τους είναι το γεγονός ότι η QBone Premium απαιτεί οι συνοριακοί κόμβοι ενός domain να υποστηρίζουν την μορφοποίηση της συνένωσης ροών που εξέρχονται από αυτούς.

Γενικά όλες οι υπηρεσίες που βασίζονται στο μοντέλο DiffServ και επομένως και η IP Premium παρουσιάζουν μια σημαντική δυσκολία που συνίσταται στη μελέτη και ανάλυση της ποιότητας υπηρεσίας που παρέχουν. Όπως είναι γνωστό αφορούν

συνενώσεις ροών που η μορφή και τα χαρακτηριστικά τους δεν μπορούν να προσεγγιστούν εύκολα, ενώ επίσης δέχονται και αλλοίωση από τα IP δίκτυα. Συνεπώς για να υλοποιηθεί μια τέτοια υπηρεσία πρέπει εκτός από τον καθορισμό των παραμέτρων που αναφέρθηκαν παραπάνω, να ακολουθηθεί και μια πειραματική διαδικασία σε συνθήκες πραγματικής λειτουργίας προκειμένου να ρυθμιστούν όλες οι παράμετροι του δικτύου και να προκύψει το αναμενόμενο αποτέλεσμα.

Παράλληλα με τις δύο EF-based υπηρεσίες που αναφέρθηκαν, στο 1.18.3 περιγράφεται και άλλη μια υπηρεσία όπου εγγυάται χωρητικότητα και καθυστέρηση, απαιτώντας από την πηγή να ορίσει το μέγιστο ρυθμό μετάδοσης και το μέγιστο μέγεθος της έκρηξής της . Ο μηχανισμός χρονοδρομολόγησης που εφαρμόζεται στην υπηρεσία αυτή είναι ο μηχανισμός προτεραιότητας (priority queuing).

Συνολικά, στα 1.18.3 και 1.18.3 παρουσιάζονται έρευνες για την απόδοση των υπηρεσιών αυτών. Τα σημεία τα οποία εξετάζονται είναι η επίδραση του μεγέθους της ουράς στην οποία εισέρχονται τα EF πακέτα, του αλγόριθμου δρομολόγησης και του μεγέθους των πακέτων. Παράλληλα ελέγχεται η επίδραση της ύπαρξης ή όχι best –effort κίνησης στην εγγυημένη, από την EF-based υπηρεσία, ποιότητα. Από τη διερεύνηση προέκυψε ότι όσο αφορά τη χρονοδρομολόγηση, η χρήση ουράς προτεραιότητας δίνει για κάθε μέγεθος πακέτου την ελάχιστη καθυστέρηση. Επίσης οι αλγόριθμοι priority Queueing (PQ) και Weighted Fair Queueing (WFQ) παρουσιάζουν παρόμοιες μέσες τιμές για το jitter ενώ ο PQ ενδεχομένως να ελαχιστοποιεί το jitter σε σχέση με τον WFQ μόνο στην περίπτωση που ο WFQ έχει αριθμό ουρών μεγαλύτερο από δύο. Τα συμπεράσματα αυτά επαληθεύτηκαν και στην περίπτωση όπου υπήρχε και best-effort κίνηση.

Ορισμένες επιπλέον παρατηρήσεις είναι ότι η μέση καθυστέρηση φαίνεται να αυξάνεται γραμμικά με την αύξηση του μεγέθους του πακέτου, ενώ η αύξηση παρουσιάζεται περισσότερο απότομη με την απουσία best – effort κίνησης. Στη συνέχεια επικεντρώνοντας στο jitter παρουσιάζεται κατά την απουσία best effort κίνησης να αυξάνεται γραμμικά ενώ με την παρουσία της παρουσιάζει μεταβολές χωρίς να ακολουθεί συγκεκριμένη μορφή.

Τέλος στο 1.18.3 έχει πραγματοποιηθεί μια σειρά πειραμάτων για την απόδοση που αντιλαμβάνεται η EF κίνηση κατά μήκος ενός DiffServ domain όπου υπάρχουν πολλά σημεία συγκέντρωσης ροών και επομένως πιθανά σημεία συμφόρησης. Τα βασικά συμπεράσματα είναι ότι η καθυστέρηση από άκρο σε άκρο είναι ανάλογη του αριθμού των σημείων που παρουσιάζουν συμφόρηση στη διαδρομή των πακέτων από την πηγή στον προορισμό. Παράλληλα οι καθυστερήσεις των EF πακέτων οφείλονται κυρίως στην καθυστέρηση στις ουρές. Επίσης η συνένωση πολλών ροών οδηγεί στην αύξηση της καθυστέρησης και του jitter και ιδίως στη δεύτερη περίπτωση η επίδραση

του βαθμού συνένωσης είναι αντιστρόφως ανάλογη του μεγέθους των EF πακέτων. Παράλληλα jitter εμφανίζεται ακόμη και στις περιπτώσεις όπου δεν έχουμε best effort κίνηση αλλά υπάρχουν συνενώσεις EF κίνησης. Το γεγονός αυτό μάλλον οφείλεται στη συγκέντρωση πακέτων στις ουρές της EF κίνησης καθώς απομακρυνόμαστε από ην είσοδο του DiffServ domain.

Ανακεφαλαιώνοντας λοιπόν, στην υλοποίηση EF-based υπηρεσιών πρέπει να δοθεί ιδιαίτερη προσοχή για την εξάλειψη των παρενεργειών στην απόδοση εξαιτίας της δημιουργίας συνενώσεων καθώς η EF κίνηση διατρέχει ένα DiffServ enabled domain. Για να επιτευχθεί αυτό πρέπει να διατηρείται το φορτίο σε χαμηλά επίπεδα προκειμένου να μπορεί το δίκτυο να απορροφά τις εκρήξεις χωρίς επιβάρυνση της απόδοσης. Συνεπώς η σωστή υλοποίηση μιας EF-based υπηρεσίας πρέπει να επιτυγχάνει δύο πράγματα:

- Να έχουν ρυθμιστεί οι buffers να έχουν τόσες θέσεις ώστε να απορροφούν τις εκρήξεις και συνάμα
- Το μέγεθος των buffers να μην προκαλεί αύξηση της καθυστέρησης από άκρο σε άκρο λόγω παρατεταμένης παραμονής πακέτων σε αυτούς.

Επίσης σημαντικό παράγοντα αποτελεί και ο αλγόριθμος χρονοδρομολόγησης που θα επιλεγεί, όπου φαίνεται να υπερτερεί ο αλγόριθμος PQ έναντι του WFQ με βάση τις παραμέτρους ποιότητας. Όμως παρατηρήθηκε ότι σε μετρήσεις σε σχέση με την διασύνδεση πολλών κόμβων τους οποίους διατρέχουν οι συνενώσεις EF ροών, ο WFQ επιτυγχάνει μικρότερες εκρήξεις για τις διάφορες συνενώσεις. Μια πρόταση που έχει διατυπωθεί είναι η χρησιμοποίηση priority queueing σε συνδυασμό με μορφοποίηση της κίνησης στα κατάλληλα σημεία του δικτύου.

1.9.2. AF based Υπηρεσίες

Οι υπηρεσίες αυτές βασίζονται στην Assured Forwarding PHB. Στόχος τους αποτελεί να εγγυηθούν την εξυπηρέτηση μιας ροής που εναρμονίζεται με το συμφωνηθέν προφίλ με πολύ μεγάλη πιθανότητα και αντίστοιχα κίνηση που βρίσκεται εκτός προφίλ με μικρότερη πιθανότητα. Στο 1.18.3 παρουσιάζεται μια προσπάθεια να καθοριστεί Per Domain Behavior (PDB) με βάση το AF PHB. Γενικά οι υπηρεσίες αυτές βρίσκονται σε πρώιμο στάδιο και αποτελεί θέμα διερεύνησης ο ακριβής καθορισμός τους και η λεπτομερειακή μελέτη της απόδοσης που μπορεί να επιτύχουν. Οι γενικοί κανόνες του AF PHB που χρησιμοποιούνται προκειμένου να οριστούν οι υπηρεσίες αυτές είναι οι ακόλουθοι:

- Δεν πρέπει να γίνεται συνένωση των ροών δύο διαφορετικών AF κλάσεων.
- Σε κάθε κλάση πρέπει να παρέχεται συγκεκριμένη ποσότητα πόρων και η κλάση πρέπει να επιτυγχάνει το ρυθμό της τόσο βραχυπρόθεσμα όσο και μακροπρόθεσμα.
- Η πιθανότητα εξυπηρέτησης ενός πακέτου πρέπει να είναι αντιστρόφως ανάλογη της προτεραιότητας εξυπηρέτησής του.
- Κάθε κόμβος πρέπει να δέχεται πακέτα και όλων των επιπέδων προτεραιότητας και πρέπει να τα εξυπηρετεί με τουλάχιστον δύο διαφορετικά επίπεδα προτεραιότητας.
- Κάθε κόμβος πρέπει να διατηρεί αναλλοίωτη τη σειρά των πακέτων που ανήκουν στην ίδια ροή και στην ίδια κλάση AF.

Για να εφαρμοστεί μια AF-based υπηρεσία, στο σημείο εισόδου του domain γίνεται πρώτα σύγκριση του πραγματικού aggregate κίνησης με τα προκαθορισμένα και συμφωνημένα χαρακτηριστικά που θα έπρεπε να έχει. Έτσι η κίνηση εντός προφίλ μαρκάρεται με προτεραιότητα 1 (πράσινα πακέτα), κίνηση εκτός προφίλ αλλά εντός ενός ορίου σαν προτεραιότητα 2 (κίτρινα πακέτα) και τέλος η υπόλοιπη σαν προτεραιότητα 3 (κόκκινα πακέτα). Η διαδικασία αυτή συνήθως γίνεται με τη χρήση δύο συνεχόμενων leaky buckets που λειτουργούν όμως και σαν μορφοποιητές. Στη συνέχεια μέσα στο δίκτυο κάθε AF κλάση διαχειρίζεται διαφορετικά όσο αφορά το θέμα της απόρριψης πακέτων από τους μηχανισμούς διαχείρισης ουρών.

Μια πρώτη παρατήρηση για τις AF-based υπηρεσίες είναι ότι παρουσιάζουν ορατή βελτίωση στην ποιότητα εξυπηρέτησης μόνο στην περίπτωση που το δίκτυο βρίσκεται σε συμφόρηση. Επίσης ένα μειονέκτημα τους είναι το γεγονός ότι είναι προσανατολισμένες στους αποστολείς και όχι στους παραλήπτες. Έτσι, στην ειδική περίπτωση που ο παραλήπτης πρέπει να λαμβάνει πληροφορία με συγκεκριμένο ρυθμό, αυτό δεν είναι εφικτό διότι η συνένωση ροών που φτάνουν στον παραλήπτη μπορεί να προέρχονται από διαφορετικά σημεία εισόδου στο DiffServ domain.

Γενικά οι AF-based υπηρεσίες δεν παρέχουν συγκεκριμένες εγγυήσεις για την καθυστέρηση και το jitter, αφού κάτι τέτοιο αναιρεί τη δυνατότητα που έχουν οι συνενώσεις ροών να καταλαμβάνουν περισσότερους πόρους από όσους συμφωνήθηκαν, αν αυτοί είναι διαθέσιμοι. Συγκεκριμένα για την παροχή εγγυήσεων από τις AF-based υπηρεσίες ισχύουν 1.18.3:

- Για την εξασφάλιση χωρητικότητας σε over-provisioned δίκτυα είναι εφικτή η εγγύηση ρυθμού που αφορά όμως μόνο την βασική πρόσβαση στους πόρους

του δικτύου και όχι στην παροχή εγγυήσεων για χωρητικότητα που μπορεί να είναι αδιάθετη.

- Η πιθανότητα απόρριψης πακέτων μπορεί να είναι εγγυημένη μόνο όταν πρόκειται για SLAs μεταξύ δύο domains ενώ είναι δύσκολο και χρειάζεται ιδιαίτερη προσοχή η εγγύηση της από άκρο σε άκρο πιθανότητας απόρριψης.
- Τέλος για την καθυστέρηση ισχύουν ακριβώς τα ίδια όπως και στην περίπτωση της απόρριψης πακέτων.

Μια παραλλαγή των AF-based υπηρεσιών είναι οι Assured Data όπου χρησιμοποιούνται AF PHB και PDB όχι για τη διασφάλιση του ρυθμού αλλά για την διασφάλιση μετάδοσης συγκεκριμένων πακέτων. Για παράδειγμα ο μηδενισμός της πιθανότητας απόρριψης πράσινων πακέτων.

ΠΡΟΣΟΜΟΙΩΤΗΣ ΔΙΚΤΥΩΝ NS

1.10. Εισαγωγή

Ένας καλός τρόπος για την ανάπτυξη δικτυακών εφαρμογών αλλά και την κατανόηση της λειτουργίας των επικοινωνιακών πρωτοκόλλων και των διατάξεων των δικτύων, καθώς και για τη μελέτη και εκτίμηση διαφόρων σχεδιαστικών συνδυασμών και επιπτώσεων στην απόδοση είναι η χρησιμοποίηση της προσομοίωσης.

Σε σύγκριση με τις πραγματικές υλοποιήσεις υλικού και λογισμικού, η προσομοίωση παρέχει αρκετά πλεονεκτήματα τόσο κατά την ερευνητική όσο και κατά την εκπαιδευτική της χρήση. Θα πρέπει να αναφερθεί ότι με την προσομοίωση είναι εύκολο να τροποποιηθούν διάφορα στοιχεία ενός δικτύου ή διάφορα χαρακτηριστικά ενός πρωτοκόλλου ή μιας τοπολογίας, με αποτέλεσμα να αλλάξουν τα χαρακτηριστικά απόδοσης, ώστε ο ερευνητής να αναλύσει εύκολα τα αποτελέσματα αυτών των τροποποιήσεων. Έτσι μπορούν να ξεπεραστούν πρακτικές δυσχέρειες που με άλλο τρόπο θα μπορούσαν να αποδειχθούν ανυπέρβλητες σε μία πρώτη φάση για την εξέλιξη και την ανάπτυξη της ιδέας ενός ερευνητή. Θα πρέπει επίσης να αναφερθεί ότι η πραγματική δοκιμή του δικτύου σε ρεαλιστικές συνθήκες είναι αρκετά δαπανηρή αφού θα απαιτούνταν μεγάλες επενδύσεις λογισμικού και πολύ συχνά ανάπτυξη υλικού γεγονός που θα εμπόδιζε τις επιστημονικές μονάδες να ασχοληθούν με την εξέλιξη και την ανάπτυξη των δικτύων. Αντίθετα αντίστοιχες δοκιμές σε ένα περιβάλλον προσομοίωσης είναι φθηνή και ανέξοδη λύση για την αξιολόγηση των μηχανισμών δικτύωσης και πρωτοκόλλων. Τέλος, θα πρέπει να αναφερθεί πως μέσω της προσομοίωσης παρέχονται λεπτομερείς στατιστικά στοιχεία της απόδοσης, τα οποία μπορούν να χρησιμοποιηθούν για την κατανόηση των διαφόρων εναλλακτικών αποδόσεων.

Το πιο ευρέως χρησιμοποιούμενο περιβάλλον προσομοίωσης σε ερευνητικό επίπεδο είναι ο προσομοιωτής ns-2 ο οποίος είναι ένα open source σύστημα που δημιουργήθηκε στο πανεπιστήμιο του Berkeley. Ο ns-2 παρέχει σημαντική υποστήριξη και μια πληθώρα επιπρόσθετων επεκτάσεων από την επιστημονική και την ερευνητική κοινότητα που επιτρέπει στους χρήστες να αναπτύξουν και να μελετήσουν οποιαδήποτε τοπολογία προσομοιώνοντας ρεαλιστικές συνθήκες δικτύου. Ωστόσο, ο ns-2 έχει αρκετές ελλείψεις οι οποίες και οδήγησαν στην ανάγκη για την ανάπτυξη ενός νέου προσομοιωτή του ns-3.

1.11. Περιγραφή του ns-3 (Network Simulator-3)

Ο ns-31.18.3, ο οποίος χρησιμοποιήθηκε για τις ανάγκες της τρέχουσας διπλωματικής εργασίας, είναι ένας προσομοιωτής διακριτών γεγονότων (discrete event simulator) ο οποίος στοχεύει κυρίως στην έρευνα δικτύων αλλά χρησιμοποιείται και για εκπαιδευτικούς σκοπούς. Πρόκειται για ένα ελεύθερο λογισμικό, υπό την άδεια της GNU GPLv2 license 1.18.3, και είναι διαθέσιμο στο κοινό για έρευνα ανάπτυξη και χρήση. Ο στόχος του ns-3 project είναι να αναπτύξει ένα προτεινόμενο, ανοιχτού κώδικα περιβάλλον προσομοίωσης για την έρευνα στα δίκτυα έτσι ώστε να είναι ευθυγραμμισμένο με της ανάγκες της σύγχρονης προσομοίωσης της διαδικτυακής έρευνας και να ενθαρρύνει την συνεισφορά, την αξιολόγηση και την επικύρωση του λογισμικού από την κοινότητα.

Το ns-3 project έχει αναλάβει την δέσμευση να δημιουργήσει ένα σταθερό πυρήνα προσομοίωσης, ο οποίος είναι καλά τεκμηριωμένος, εύκολος στη χρήση και στην αποσφαλμάτωση, καθώς και να ικανοποιήσει τις ανάγκες της συνολικής ροής της προσομοίωσης, ξεκινώντας από τις ρυθμίσεις της προσομοίωσης και φτάνοντας στην συλλογή των ιχνών και την ανάλυση. Επιπλέον, η υποδομή του λογισμικού του ns-3 ενθαρρύνει την ανάπτυξη μοντέλων προσομοίωσης τα οποία είναι αρκετά ρεαλιστικά για να επιτρέψουν στον ns-3 να χρησιμοποιηθεί ως ένας εξομοιωτής δικτύων πραγματικού χρόνου, ο οποίος να διασυνδέεται με τον πραγματικό κόσμο και να επιτρέπει πολλές από τις υφιστάμενες υλοποιήσεις πραγματικών πρωτοκόλλων να επαναχρησιμοποιηθούν εντός του ns-3.

Ο πυρήνας προσομοίωσης του ns-3 υποστηρίζει την έρευνα τόσο σε δίκτυα βασισμένα σε IP όσο και σε δίκτυα που δε στηρίζονται στην IP. Ωστόσο, η μεγάλη πλειοψηφία των χρηστών επικεντρώνονται στην ασύρματη/IP προσομοίωση η οποία συμπεριλαμβάνει μοντέλα για το Wi-Fi, WiMAX, LTE των επιπέδων 1 και 2 και μια ποικιλία από πρωτόκολλα στατικής ή δυναμικής δρομολόγησης όπως OLSR και AODV για IP-based εφαρμογές. Ο ns-3, επίσης, υποστηρίζει χρονοπρογραμματισμό πραγματικού χρόνου διευκολύνοντας περιπτώσεις χρήσης στις οποίες υπάρχει αλληλεπίδραση με πραγματικά δίκτυα. Για παράδειγμα, οι χρήστες μπορούν να εκπέμπουν και να λαμβάνουν πακέτα που δημιουργούνται σε πραγματικές συσκευές δικτύου με τη χρήση του ns-3, ο οποίος μπορεί να χρησιμοποιηθεί ως ένα πλαίσιο διασύνδεσης για την προσθήκη links μεταξύ εικονικών μηχανών.

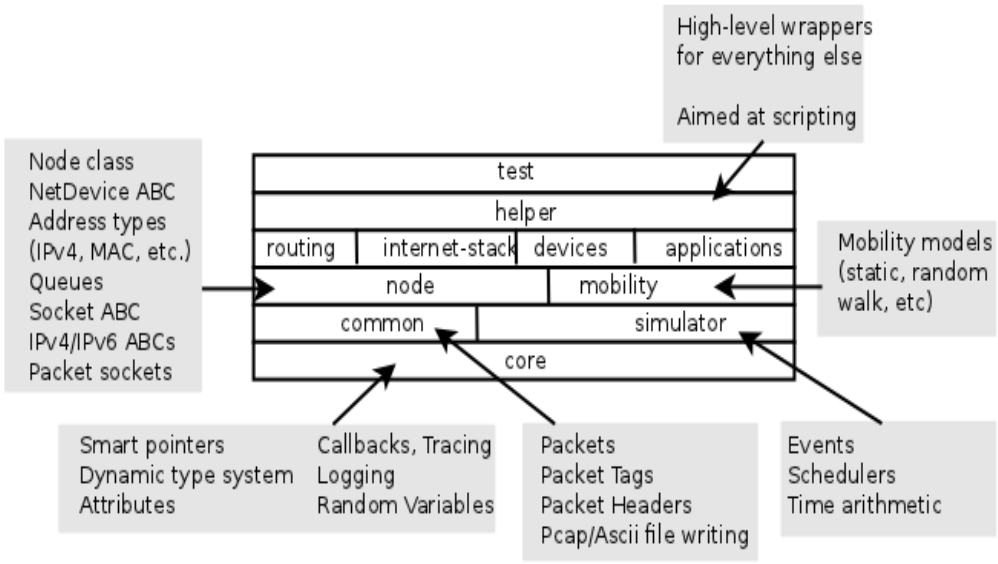
Μια άλλη έμφαση του προσομοιωτή είναι η επαναχρησιμοποίηση των πραγματικών εφαρμογών και του κώδικα του πυρήνα. Τα πλαίσια για την εκτέλεση των μη τροποποιημένων εφαρμογών ή ολόκληρου του πυρήνα των Linux για την δικτύωση εντός του ns-3, προς το παρόν εξετάζονται και αξιολογούνται.

Η δημιουργία ενός προσομοιωτή δικτύου από το μηδέν, απαιτεί μια υψηλής ποιότητας επικύρωσης του λογισμικού του καθώς και αρκετή εργασία για να διατηρηθούν τα μοντέλα του, έτσι ο ns-3 προσπαθεί να μεταφέρει αυτόν τον τεράστιο φόρτο εργασίας σε μια μεγάλη κοινότητα χρηστών και προγραμματιστών καθιστώντας την συμβολή αυτής της κοινότητα ιδιαίτερα μεγάλη για την ανάπτυξη και την εξέλιξη του ns-3.

Προς αυτήν την κατεύθυνση κινείται και η δημιουργία αυτής της διπλωματικής εργασίας, όπου στόχος της ήταν η δημιουργία ενός μηχανισμού που δε θα υπήρχε στον συγκεκριμένο προσομοιωτή. Έτσι, ύστερα από λεπτομερή έλεγχο των υπαρχόντων μηχανισμών προχωρήσαμε στην ανάπτυξη και στην επικύρωση της ορθής λειτουργίας του module leaky bucket ενός μηχανισμού που επεξηγείται παρακάτω.

1.11.1. Δομή του ns-3

Ο ns-3 προσομοιωτής όπως αναφέρθηκε και προηγουμένως είναι ένας προσομοιωτής διακριτών γεγονότων. Αυτό σημαίνει πως ο χρήστης προκειμένου να φτιάξει ένα πρόγραμμα που θα το τρέξει με τη βοήθεια του NS και θα βγάλει τα αποτελέσματα που χρειάζεται πρέπει να δώσει ιδιαίτερη έμφαση στα γεγονότα και κυρίως πώς αυτά εισάγονται στον NS. Επίσης, ο ns-3 είναι ένας προσομοιωτής ο οποίος είναι εξ ολοκλήρου γραμμένος σε C++ , ενώ έχει προαιρετικά κάποιες δομές γραμμένες σε Python. Τα scripts της προσομοίωσης μπορούν και αυτά με την σειρά τους να γραφτούν είτε σε C++ είτε σε Python. Ο προσομοιωτής έχει αναπτυχθεί ως μία βιβλιοθήκη η οποία μπορεί στατικά ή δυναμικά να συνδέεται στο κυρίως πρόγραμμα (όπου είναι γραμμένο στη C++) το οποίο καθορίζει την τοπολογία της προσομοίωσης, την παραγωγή κίνησης πακέτων ενώ ξεκίνα και την προσομοίωση. Ο ns-3, επιπλέον, εξάγει το σύνολο σχεδόν των εφαρμογών σε Python, επιτρέποντας προγράμματα γραμμένα σε Python να εισάγουν κάποιο ns-3 module με τον ίδιο τρόπο που η βιβλιοθήκη του ns-3 συνδέεται με τα εκτελέσιμα στην C++.



Εικόνα 8 Οργάνωση Λογισμικού του ns-3

Ο πηγαίος κώδικας στον ns-3 ως επί τον πλείστον είναι οργανωμένος στον φάκελο *src* και μπορεί να περιγραφεί στη παραπάνω Εικόνα 8 η οποία απεικονίζει την οργάνωση του λογισμικού του ns-3. Αρχικά να αναφερθεί πως όλα τα στοιχεία τα οποία είναι κοινά για όλα τα πρωτόκολλα, για οποιοδήποτε υλικό αλλά και περιβάλλον προσομοίωσης υπάρχουν μέσα στον πυρήνα-*core* του προσομοιωτή ο οποίος υλοποιείται στον φάκελο *src/core*. Επίσης, όσο αφορά στα πακέτα όπου πρόκειται για θεμελιώδη αντικείμενα υλοποιούνται στον φάκελο *src/network*. Αυτές οι δύο ενότητες του προσομοιωτή έχουν ως στόχο από μόνες τους να καταρτίσουν έναν γενικό πυρήνα προσομοίωσης έτσι ώστε να μπορούν να χρησιμοποιηθούν από διαφορετικά είδη δικτύων, όχι μόνο από τα internet-based δίκτυα.

Εκτός από τον πυρήνα που αναφέραμε προηγουμένως, υπάρχουν επιπλέον δύο μονάδες οι οποίες συμπληρώνουν αυτόν τον πυρήνα. Τα προγράμματα του ns-3 μπορούν να έχουν είτε άμεση πρόσβαση σε όλα τα API είτε να μπορούν να χρησιμοποιούν τον λεγόμενο API helper ο οποίος παρέχει έναν εύκολο χειρισμό του χαμηλού-επιπέδου των APIs. Στηριζόμενος σε αυτό δημιουργήθηκε και ο μηχανισμός του leaky bucket όπως περιγράφεται στα επόμενα κεφάλαια. Το γεγονός ότι τα προγράμματα στον ns-3 μπορούν να γραφτούν σε δύο APIs (ή συνδυασμό αυτών) είναι μία θεμελιώδης πτυχή του προσομοιωτή.

Θα πρέπει να σημειωθεί πως το ns-3 project χρησιμοποιεί το Mercurial 1.18.3 για την διαχείριση του πηγαίου κώδικα. Το Mercurial είναι ένα λογισμικό ανοιχτού κώδικα, γρήγορο και ισχυρό όπου χειρίζεται με αποτελεσματικότητα τα projects οποιοδήποτε μεγέθους και είδους. Κάθε «κλώνος» περιέχει ολόκληρη την ιστορία του project, έτσι ώστε οι περισσότερες δράσεις να έχουν τοπικό χαρακτήρα, να είναι γρήγορες και εύκολες. Το Mercurial μπορεί να υποστηρίζει ένα πλήθος ροών εργασίας ενώ μπορεί εύκολα να ενισχύσει την λειτουργικότητά του με διάφορες επεκτάσεις.

Εφόσον ο πηγαίος κώδικας του ns-3 υπάρχει στο τοπικό σύστημα θα πρέπει με κάποιο τρόπο να γίνει compile έτσι ώστε να δημιουργηθούν τα εκτελέσιμα αρχεία και να παραχθούν τα χρήσιμα προγράμματα. Έτσι, όπως και στην περίπτωση της διαχείρισης του πηγαίου κώδικα, υπάρχουν αρκετά εργαλεία που να πραγματοποιούν αυτό. Ένα από αυτά, το οποίο και χρησιμοποιήσαμε στα πλαίσια αυτής της διπλωματικής εργασίας, είναι το Waf 1.18.3. Πρόκειται λοιπόν για ένα framework βασισμένο σε Python με το οποίο μπορούμε να διαμορφώσουμε, να κάνουμε compile αλλά και να εγκαταστήσουμε τον ns-3 καθώς και άλλες εφαρμογές.

Ουσιαστικά για να μπορέσει κάποιος να χρησιμοποιήσει τον ns-3 θα πρέπει να εργάζεται σε Linux περιβάλλοντα ή σε Linux-like περιβάλλοντα. Για την εκτέλεση της διπλωματικής εργασία χρησιμοποιήσαμε λειτουργικό σύστημα Linux και συγκεκριμένα την έκδοση Ubuntu 9.10. Ωστόσο για όσους χρησιμοποιούν ως λειτουργικό σύστημα τα Windows υπάρχουν προγράμματα τα οποία προσομοιώνουν το περιβάλλον των Linux σε σημαντικό βαθμό. Για αυτούς λοιπόν τους χρήστες το ns-3 project υποστηρίζει την ανάπτυξη στο περιβάλλον του Cygwin ενώ μία εναλλακτική είναι να εγκατασταθεί ένα virtual machine περιβάλλον, όπως το VMware server και έπειτα να εγκατασταθεί κάποιο Linux virtual machine.

1.11.2. Λήψη του ns-3

Το σύστημα του ns-3, στο σύνολό του, είναι ένα αρκετά πολύπλοκο σύστημα και έχει μια σειρά εξαρτήσεις από διάφορες άλλες συνιστώσες. Έτσι μαζί με τα συστήματα που πιθανότατα θα ασχολείται ο χρήστης μαζί με τον ns-3 (όπως το GNU toolchain, Mercurial, τον editor για τον προγραμματισμό) θα πρέπει να εξασφαλίσει ότι μια σειρά πρόσθετων βιβλιοθηκών θα είναι παρούσα στο σύστημά του πριν προχωρήσει. Ο ns-3 προσφέρει μια wiki τοποθεσία στο διαδίκτυο που περιλαμβάνει πολλές σελίδες με χρήσιμες συμβουλές και υποδείξεις. Μια τέτοια σελίδα είναι και η "Installation" σελίδα στο 1.18.3 όπου παρουσιάζεται ένας λεπτομερής οδηγός για την εγκατάσταση του ns-3.

Η ενότητα "Prerequisites" (Προϋποθέσεις) αυτής της σελίδας wiki, εξηγεί ποια πακέτα απαιτούνται για την υποστήριξη των διαφόρων επιλογών του ns-3, ενώ παρέχει επίσης τις εντολές που χρησιμοποιούνται για να τις εγκαταστήσετε σε διαφορετικές παραλλαγές των Linux. Οι χρήστες Cygwin θα πρέπει να χρησιμοποιήσουν το Cygwin installer (το οποίο χρησιμοποιείται για την εγκατάσταση του προγράμματος).

Κλείνοντας την ενότητα αυτή θα πρέπει να αναφέρουμε πως ο πηγαίος κώδικας του ns-3 διατίθεται σε Mercurial στο server 1.18.3, ενώ μπορεί επίσης κάποιος να κατεβάσει ένα tarball release από το 1.18.3. Οι ακόλουθες υποενότητες παρουσιάζουν την διαδικασία που ακολουθήθηκε για την λήψη και την εγκατάσταση του ns-3.

1.11.3. Λήψη του ns-3 χρησιμοποιώντας Mercurial

Χρησιμοποιώντας λειτουργικό σύστημα Linux Ubuntu 9.10 και τρέχοντας κάποιο τερματικό του λειτουργικού πληκτρολογούμε τις ακόλουθες εντολές για να δημιουργήσουμε έναν φάκελο που ονομάζεται repos στο home directory μέσα στον οποίο μπορούν να κρατηθούν τα τοπικά Mercurial repositories:

```
cd mkdir repos cd repos hg clone http://code.nsnam.org/ns-3-allinone
```

Καθώς η hg (Mercurial) εντολή εκτελείται, εμφανίζονται τα ακόλουθα στο τερματικό:

```
destination directory: ns-3-allinone
requesting all changes
adding changesets
adding manifests
adding file changes
added 47 changesets with 67 changes to 7 files
updating to branch default
7 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Εφόσον ολοκληρωθεί η εκτέλεση της εντολής, δημιουργείται ένας φάκελος με ονομασία ns-3-allinone, το περιεχόμενο του οποίου είναι το παρακάτω:

```
build.py*  constants.py  dist.py*  download.py*  README  util.py
```

Θα πρέπει να παρατηρήσουμε παραπάνω ότι στα περιεχόμενα του ns-3-allinone υπάρχουν κάποια Python scripts τα οποία θα χρησιμοποιήσουμε για την λήψη και την κατασκευή του ns-3 της επιλογής μας. Πηγαίνοντας λοιπόν στην παρακάτω

τοποθεσία 1.18.3 , θα δούμε έναν αριθμό αρχείων που έχουν να κάνουν με τον ns-3. Πολλά από αυτά είναι ιδιωτικά αρχεία από ομάδες ανάπτυξης του ns-3. Τα αρχεία που μας ενδιαφέρουν θα πρέπει να έχουν το πρόθεμα "ns-3". Οι επίσημες εκδόσεις του ns-3 είναι αριθμημένες ως ns-3. <release>. <hotfix>.

Η τρέχουσα έκδοση της ανάπτυξης του ns-3 (η οποία είναι ακυκλοφόρητη) μπορεί να βρεθεί στο 1.18.3. Οι προγραμματιστές προσπαθούν να κρατήσουν όλες αυτές τις εκδόσεις σε μία συνέπια με συνεχή εργασία, ωστόσο βρισκόμαστε σε μια περιοχή ανάπτυξης λογισμικού με κώδικα ο οποίος είναι ακυκλοφόρητος τουλάχιστον προς το παρόν, έτσι μπορούμε εάν θελήσουμε, εφόσον δεν χρειαζόμαστε κάποια νεοεισαχθέντα χαρακτηριστικά, να παραμείνουμε στην επίσημη έκδοση του ns-3 η οποία και χρησιμοποιείται από πολλούς χρήστες και βρίσκεται ήδη σε κυκλοφορία. Έτσι η έκδοση η οποία και χρησιμοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας ήταν η ns-3.9.

Για να συνεχίσουμε την λήψη και την εγκατάσταση του ns-3 μεταβαίνουμε στον φάκελο ns-3-allinone ο οποίος δημιουργήθηκε με την εντολή που επεξηγήσαμε παραπάνω. Χρησιμοποιούμε τα Python scripts τα οποία και θα κατεβάσουν τα σημαντικότερα κομμάτια για την λειτουργία του ns-3. Για να γίνει αυτό πληκτρολογούμε την ακόλουθη εντολή:

```
./download.py
```

Καθώς ,λοιπόν, η hg (Mercurial) εκτελεί την παραπάνω εντολή εμφανίζονται τα παρακάτω στο τερματικό:

```
    #
    # Get NS-3
    #

Cloning ns-3 branch
 =>  hg clone http://code.nsnam.org/ns-3-dev ns-3-dev
requesting all changes
adding changesets
adding manifests
adding file changes
added 4634 changesets with 16500 changes to 1762 files
```

```
870 files updated, 0 files merged, 0 files removed, 0 files
unresolved
```

Αυτή είναι η έξοδος από το download script, καθώς λαμβάνει τον πραγματικό κώδικα του ns-3 από τον αποθηκευτικό χώρο του server. Εφόσον ολοκληρωθεί η εκτέλεση του download.py script, θα πρέπει να υπάρχουν οι ακόλουθοι φάκελοι κάτω από το `~/repos/ns-3-allinone`:

```
build.py*     constants.pyc  download.py*  nsc/        README      util.pyc
constants.py  dist.py*       ns-3-dev/     pybindgen/  util.py
```

Πηγαίνοντας μέσα στον φάκελο `ns-3-dev/` ο οποίος βρίσκεται κάτω από τον `~/repos/ns-3-allinone` θα εμφανίζονται τα επόμενα:

```
AUTHORS       doc       ns3            scratch   testpy.supp  VERSION   waf-tools
bindings      examples  README         src       utils        waf*      wscript
CHANGES.html  LICENSE   RELEASE_NOTES  test.py*  utils.py     waf.bat*  wutils.py
```

Τέλος, είμαστε έτοιμα να χτίσουμε (build) τον ns-3.

1.11.4. Χτίζοντας τον ns-3

Για την διαμόρφωση και το "χτίσιμο" του ns-3 χρησιμοποιήσαμε το Waf. Θα πρέπει να αναφέρουμε, κάνοντας μια μικρή παράκαμψη, ότι είναι πολύτιμο να εξετάσουμε με ποιον τρόπο μπορούμε να κάνουμε αλλαγές στις ρυθμίσεις του project. Ίσως η ποιο χρήσιμη αλλαγή των ρυθμίσεων που μπορεί κανείς να κάνει είναι να οικοδομήσει την optimized έκδοση του κώδικα. Από προεπιλογή ρυθμίζεται το project να οικοδομηθεί στην debug έκδοση. Εμείς στα πλαίσια αυτής της διπλωματικής διαλέξαμε το project να κατασκευαστεί στην optimized έκδοση. Για να γίνει αυτό χρησιμοποιώντας το Waf εκτελέσαμε την ακόλουθη εντολή βρισκόμενη στον φάκελο `~/repos/ns-3-allinone` :

```
./waf –d optimized configure
```

Η εντολή αυτή τρέχει το Waf έξω από τον τοπικό φάκελο. Όσο η οικοδόμηση του συστήματος του ns-3 πραγματοποιείται και εξετάζονται σημαντικές εξαρτήσεις που μπορούν να υπάρχουν παρουσιάζονται τα ακόλουθα στη οθόνη του τερματικού:

```
Checking for program g++                 : ok /usr/bin/g++
Checking for program cpp                 : ok /usr/bin/cpp
Checking for program ar                  : ok /usr/bin/ar
Checking for program ranlib              : ok /usr/bin/ranlib
Checking for g++                         : ok
Checking for program pkg-config          : ok /usr/bin/pkg-config
Checking for -Wno-error=deprecated-declarations support : yes
Checking for -Wl,--soname=foo support                   : yes
Checking for header stdlib.h                            : ok
Checking for header signal.h                            : ok
Checking for header pthread.h                           : ok
Checking for high precision time implementation         : 128-bit
integer
Checking for header stdint.h                            : ok
Checking for header inttypes.h                          : ok
Checking for header sys/inttypes.h                      : not found
Checking for library rt                                 : ok
```

```
Checking for header netpacket/packet.h                 : ok
Checking for pkg-config flags for GSL                  : ok
Checking for header linux/if_tun.h                     : ok
Checking for pkg-config flags for GTK_CONFIG_STORE     : ok
Checking for pkg-config flags for LIBXML2              : ok
Checking for library sqlite3                           : ok
Checking for NSC location                              : ok ../nsc
(guessed)
Checking for library dl                                : ok
Checking for NSC supported architecture x86_64         : ok
Checking for program python                            : ok
/usr/bin/python
Checking for Python version >= 2.3                     : ok 2.5.2
Checking for library python2.5                         : ok
Checking for program python2.5-config                  : ok
/usr/bin/python2.5-config
Checking for header Python.h                           : ok
Checking for -fvisibility=hidden support               : yes
Checking for pybindgen location                        : ok
../pybindgen (guessed)
Checking for Python module pybindgen                   : ok
Checking for pybindgen version                         : ok
0.10.0.640
Checking for Python module pygccxml                    : ok
Checking for pygccxml version                          : ok 0.9.5
Checking for program gccxml                            : ok
/usr/local/bin/gccxml
Checking for gccxml version                            : ok 0.9.0
Checking for program sudo                              : ok
/usr/bin/sudo
Checking for program hg                                : ok
/usr/bin/hg
Checking for program valgrind                          : ok
/usr/bin/valgrind
---- Summary of optional NS-3 features:
Threading Primitives          : enabled
```

```
Real Time Simulator           : enabled
Emulated Net Device           : enabled
GNU Scientific Library (GSL)  : enabled
Tap Bridge                    : enabled
GtkConfigStore                : enabled
XmlIo                         : enabled
SQlite stats data output      : enabled
Network Simulation Cradle     : enabled
Python Bindings               : enabled
Python API Scanning Support   : enabled
Use sudo to set suid bit      : not enabled (option --enable-sudo not
selected)
Build tests                   : enabled
Build examples                : enabled
Static build                  : not enabled (option --enable-static
not selected)
'configure' finished successfully (2.870s)
```

Παρατηρώντας την έξοδο που τυπώνεται θα πρέπει να αναφέρουμε πως μερικές από τις επιλογές που μπορεί να έχει ο ns-3 κατά την διαμόρφωσή του δεν είναι ενεργοποιημένες ή απαιτούν υποστήριξη από το υποκείμενο σύστημα για να λειτουργήσουν σωστά. Για παράδειγμα για να μπορεί το XmlTo να είναι ενεργοποιημένο θα πρέπει και οι αντίστοιχες βιβλιοθήκες που έχουν να κάνουν με αυτό να βρίσκονται στο σύστημα. Στην προκειμένη περίπτωση πρόκειται για την βιβλιοθήκη libxml-2.0 η οποία και υπάρχει στο λειτουργικό μας γι αυτό και το XmlTo εμφανίζεται ως enable. Σε περίπτωση που η βιβλιοθήκη δεν βρεθεί στο σύστημά μας, το αντίστοιχο χαρακτηριστικό δε θα είναι ενεργοποιημένο και θα εμφανιστεί κατάλληλο μήνυμα. Τέλος να αναφέρουμε ότι υπάρχει η επιλογή ‘Use sudo to set suid bit’ η οποία είναι απενεργοποιημένη από προεπιλογή γι αυτό εμφανίζεται και το αντίστοιχο μήνυμα not enabled (option --enable-sudo not selected.

Έπειτα από την εκτέλεση της παραπάνω εντολής προχωρούμε στην συνέχιση της διαμόρφωσης στην debug έκδοση η οποία περιλαμβάνει παραδείγματα και tests εκτελώντας την ακόλουθη εντολή:

```
./waf -d debug configure
```

Ακολουθώντας αυτήν την διαδικασία το σύστημα του ns-3 είναι διαθέσιμο για έρευνα και πειραματισμό. Έτσι για τη δημιουργία διαφόρων τοπολογιών χρησιμοποιώντας τον ns-3 για να γίνει το απαραίτητο debug αρκεί να πληκτρολογήσουμε απλά την ακόλουθη εντολή:

```
./waf
```

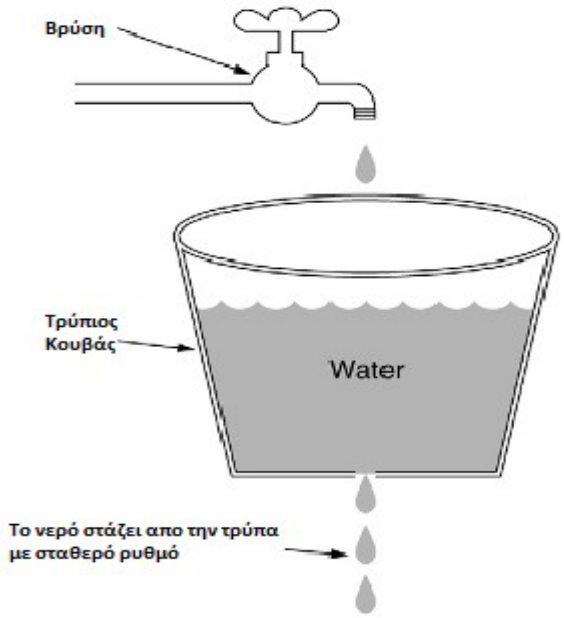
Τέλος, εφόσον δεν υπάρχουν σφάλματα που να αφορούν στην τοπολογία τρέχουμε κάτω από τον έλεγχο του Waf το αρχείο της τοπολογία μας και έχουμε τα αποτελέσματα του πειραματισμού μας. Για να γίνει αυτό χρησιμοποιούμε την επιλογή –run του Waf. Έτσι εάν το script που δημιουργήσαμε έχει ονομασία myLeaky BucketTopology για να το τρέξουμε εκτελούμε την ακόλουθη εντολή:

```
./waf --run myLeakyBucketTopology
```

Ο ΜΗΧΑΝΙΣΜΟΣ ΤΟΥ LEAKY BUCKET

Πριν προχωρήσουμε παρακάτω και συγκεκριμένα στον τρόπο με τον οποίο δημιουργήθηκε και προστέθηκε ο μηχανισμός του leaky bucket στον ns-3 είναι απαραίτητο να προχωρήσουμε πρώτα σε μία λεπτομερή περιγραφή του συγκεκριμένο αλγορίθμου.

Σε ορισμένες βιβλιογραφίες 1.18.3 ο αλγόριθμος του leaky bucket αναφέρεται ως αλγόριθμος του τρύπιου κουβά. Ας φανταστούμε, λοιπόν, έναν κουβά με μια μικρή τρύπα στη βάση του, όπως απεικονίζεται στην Εικόνα 9.



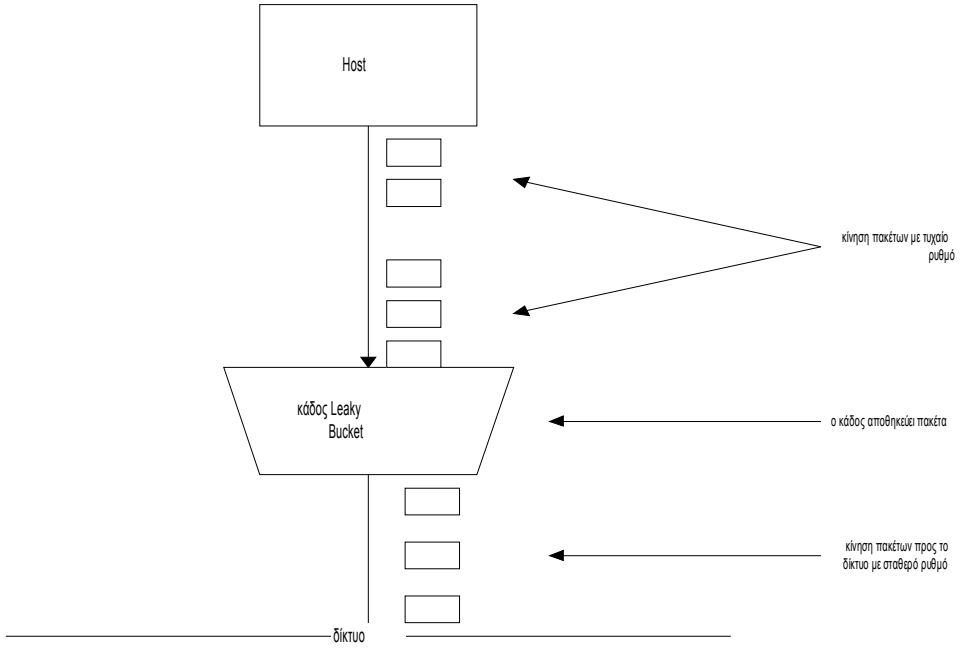
Εικόνα 9 Απεικόνιση του τρύπιου κουβά

Ανεξάρτητα από τον ρυθμό με τον οποίο το νερό εισέρχεται στον κουβά, η εξερχόμενη ροή έχει σταθερό ρυθμό ρ όταν υπάρχει νερό στον κουβά και μηδέν όταν ο κουβάς είναι άδειος. Αφού γεμίσει ο κουβάς, το τυχόν πρόσθετο νερό θα ξεχειλίσει από τις πλευρές και θα χάνεται (δηλαδή, δεν θα εμφανίζεται στη ροή εξόδου κάτω από την τρύπα).

Η ίδια λογική μπορεί να εφαρμοστεί στα πακέτα, όπως παρουσιάζεται στην Εικόνα 10. Από λογική άποψη, κάθε υπολογιστής υπηρεσίας συνδέεται στο δίκτυο με μια διασύνδεση η οποία περιέχει έναν τρύπιο κουβά, δηλαδή μια πεπερασμένου μήκους εσωτερική ουρά. Αν ένα πακέτο φτάσει στην ουρά όταν αυτή είναι γεμάτη, το πακέτο απορρίπτεται. Με άλλα λόγια, αν μία οι περισσότερες διεργασίες του υπολογιστή υπηρεσίας προσπαθήσουν να στείλουν ένα πακέτο όταν η ουρά περιέχει ήδη το μέγιστο πλήθος πακέτων, το νέο πακέτο απορρίπτεται χωρίς χρονοτριβή. Η διάταξη αυτή μπορεί να ενσωματωθεί στο υλικό διασύνδεσης ή να προσομοιωθεί από ένα λειτουργικό σύστημα του υπολογιστή υπηρεσίας. Προτάθηκε για πρώτη φορά

από τον Turner (1986) και ονομάζεται αλγόριθμος του τρύπιου κουβά (leaky bucket algorithm). Στην πραγματικότητα δεν είναι τίποτε άλλο από ένα σύστημα ουράς με ένα διακομιστή, το οποίο παρέχει σταθερό χρόνο εξυπηρέτησης.

Ο υπολογιστής υπηρεσίας επιτρέπεται να εισάγει ένα πακέτο στο δίκτυο σε κάθε χτύπο του ρολογιού. Και πάλι, αυτό μπορεί να επιβάλλεται είτε από κάρτα διασύνδεσης είτε από το λειτουργικό σύστημα. Ο μηχανισμός αυτός μετατρέπει την ακανόνιστη ροή πακέτων από τις διεργασίες των χρηστών που υπάρχουν μέσα στον υπολογιστή υπηρεσίας σε μια ομαλή ροή πακέτων στο δίκτυο, εξομαλύνοντας τις ριπές και μειώνοντας σημαντικά τις πιθανότητες συμφόρησης.



Εικόνα 10 Λειτουργία του Leaky Bucket

Όταν όλα τα πακέτα έχουν το ίδιο μέγεθος (για παράδειγμα τα κελιά ATM), ο αλγόριθμος αυτός μπορεί να χρησιμοποιείται όπως ακριβώς περιγράφηκε παραπάνω. Όταν όμως χρησιμοποιούνται πακέτα μεταβλητού μεγέθους, συχνά είναι καλύτερα να επιτρέπεται η μετάδοση ενός σταθερού πλήθος byte ανά χτύπο ρολογιού, αντί για ένα πακέτο. Έτσι, αν ο κανόνας είναι να επιτρέπονται 1024 byte ανά χτύπο, σε ένα χτύπο ρολογιού μπορεί να γίνει αποδεκτό ένα πακέτο των 1024 byte, δύο πακέτα των 512 byte, τέσσερα πακέτα των 256 byte, και ούτω καθεξής. Αν τα υπολειπόμενα byte είναι πολύ λίγα, το πακέτο θα πρέπει να περιμένει μέχρι τον επόμενο χτύπο.

Η υλοποίηση του αρχικού αλγορίθμου του leaky bucket (του τρύπιου κουβά) είναι εύκολη. Ο τρύπιος κουβάς είναι μια πεπερασμένη ουρά. Όταν φτάνει ένα πακέτο, προστίθεται στην ουρά εφόσον υπάρχει χώρος. Διαφορετικά το πακέτο απορρίπτεται.

Σε κάθε χτύπο του ρολογιού μεταδίδεται ένα πακέτο (εκτός και εάν η ουρά είναι άδεια).

Το leaky bucket που μετράει byte υλοποιείται με τον ίδιο τρόπο. Σε κάθε χτύπο δίνουμε αρχική τιμή *n* σε ένα μετρητή. Αν το πρώτο πακέτο της ουράς έχει λιγότερα byte από την τρέχουσα τιμή του μετρητή, τότε το πακέτο μεταδίδεται και ο μετρητής μειώνεται κατά το αντίστοιχο πλήθος byte. Μπορούν να σταλούν και πρόσθετα πακέτα, εφόσον βέβαια ο μετρητής είναι αρκετά μεγάλος. Όταν ο μετρητής πέσει κάτω από το μήκος του επόμενου πακέτου της ουράς η μετάδοση σταματάει μέχρι τον επόμενο χτύπο ρολογιού, οπότε επαναρυθμίζεται ο μετρητής υπολειπόμενων byte και η ροή μπορεί να συνεχιστεί.

Ως παράδειγμα του leaky bucket, μπορεί να φανταστεί κανείς ότι ένας υπολογιστής μπορεί να παράγει δεδομένα με ρυθμό 25 εκατομμύρια byte/sec (200 Mbps) και ότι το δίκτυο λειτουργεί με την ίδια ταχύτητα. Παρόλα αυτά, οι δρομολογητές μπορούν να δέχονται αυτόν τον ρυθμό δεδομένων μόνο για σύντομα χρονικά διαστήματα (ουσιαστικά, μέχρι να γεμίσουν οι περιοχές προσωρινής αποθήκευσης). Για μεγάλα διαστήματα, οι δρομολογητές λειτουργούν καλύτερα με ταχύτητες οι οποίες δε ξεπερνούν τα 2 εκατομμύρια byte/sec. Ας υποθέσει τώρα κάποιος, ότι τα δεδομένα φτάνουν σε ριπές του 1 εκατομμυρίου byte, με μία ριπή διάρκειας 40 msec ανά δευτερόλεπτο. Για να μειωθεί ο μέσος ρυθμός στα 2MB/sec, θα μπορούσε να χρησιμοποιηθεί ένα leaky bucket με ρ = 2MB/sec και χωρητικότητα *C* ίση με 1 MB. Αυτό σημαίνει ότι μπορούν να αντιμετωπιστούν ριπές μέχρι και 1MB χωρίς απώλεια δεδομένων και ότι οι ριπές αυτές θα στέλνονται κατά τη διάρκεια 500 msec, ανεξάρτητα από το πόσο γρήγορα φτάνουν.

Κλείνοντας, θα πρέπει να αναφερθεί πως ο αλγόριθμος του leaky bucket αναπτύσσεται συνήθως στα σημεία εισόδου του κεντρικού δικτύου, προστατεύοντας έτσι το κυρίως δίκτυο από "εκρήξεις" κινήσεων πακέτων που μπορούν να παραχθούν από τους περιφερικούς κόμβους. Ως εκ τούτου, τα επίπεδα κυκλοφορίας στο δίκτυο κορμού διατηρούνται σε αποδεκτά επίπεδα αποφεύγοντας την συμφόρηση και προσφέροντας ένα είδος ποιότητας στην εξυπηρέτηση. Στην παρακάτω Εικόνα 11 ακολουθεί ο ψευδοκώδικας του αλγορίθμου. Σύμφωνα με αυτόν εάν δεν υπάρχουν διαθέσιμα «κουπόνια» (δηλαδή αν ο κάδος είναι γεμάτος) τότε το πακέτο απορρίπτεται, ενώ σε διαφορετική περίπτωση αποθηκεύεται στην ουρά και αποστέλλεται:

```
while (incoming packet):
     Add tokens to bucket: rate * time_passed
     Check if we have any tokens
     if (no full token available):
          drop packet
     else:
          send packet
          available_tokens = available_tokens - 1
```

Εικόνα 11 Ψευδοκώδικας του leaky bucket

Η ΥΛΟΠΟΙΗΣΗ ΤΟΥ LEAKY BUCKET ΣΤΟΝ NS-3

1.12. Εισαγωγή

Προκειμένου να υλοποιηθεί ο μηχανισμός του leaky bucket και να προστεθεί στον ns-3, πραγματοποιήθηκαν μια σειρά από τροποποιήσεις, έτσι ώστε ο χρήστης του προσομοιωτή να μπορεί να αναπτύξει τον αλγόριθμο στους επιθυμητούς κόμβους. Στη συνέχεια ακολουθεί μία λεπτομερή περιγραφή των αρχείων που δημιουργήθηκαν και τροποποιήθηκαν καθώς και του τρόπου με τον οποίο ο μηχανισμός προστέθηκε στον ns-3.

Ο χρήστης τρέχοντας την προσομοίωση μπορεί να συντονίσει τον αλγόριθμο ρυθμίζοντάς τον σωστά με ένα σύνολο παραμέτρων λειτουργίας, και να το εγκαταστήσει στην επιθυμητή συσκευή. Οι παράμετροι λειτουργίας περιλαμβάνουν την IP του κόμβου ο οποίος στέλνει την κίνηση που θα διαμορφωθεί, τον εξερχόμενων ρυθμό δεδομένων από το bucket, τον τρόπο με τον οποίο μετράτε το μέγεθος ουράς (πακέτα ή bytes) και τον μέγιστο αριθμό πακέτων ή το μέγιστο αριθμό των bytes της ουράς. Τέλος, η DSCP τιμή μπορεί να χρησιμοποιηθεί για την εισαγωγή του αλγορίθμου εντός πλαισίου μιας συνολικής αρχιτεκτονικής DiffServ συμπεριλαμβανομένου του χρονικού προγραμματισμού και της αφαίρεσης πακέτων από την ουρά. Στην τρέχουσα υλοποίηση ο αλγόριθμος του leaky bucket είναι μία drop tail ουρά η οποία αποθηκεύει τα εισερχόμενα πακέτα και τα στέλνει κατά τρόπο FIFO με ένα σταθερό ρυθμό (ρυθμιζόμενο από το χρήστη του προσομοιωτή). Υπάρχει επίσης η δυνατότητα εγκατάστασης περισσότερων από ένα leaky bucket σε έναν ή περισσότερους κόμβους για πιο ακριβή προσομοίωση των πραγματικών δικτύων. Την υλοποίηση μας για τον ns-3 μπορείτε να την κατεβάσετε για πειραματισμό στο 1.18.3.

1.13. Αρχεία που δημιουργήθηκαν

Τα πρώτα αρχεία που δημιουργήθηκαν για τον μηχανισμό του Leaky Bucket είναι τα: leaky-bucket και leaky-bucket-helper. Τα αρχεία αυτά μας επιτρέπουν να ορίσουμε τις ιδιότητες καθώς και να εγκαταστήσουμε ένα Leaky Bucket σε κάποιον Router της τοπολογίας μας για τη μορφοποίηση της κίνησης στο δίκτυό μας. Συγκεκριμένα με την χρησιμοποίηση της κλάσης του Leaky-Bucket (η οποία βρίσκεται ns-3.9/src/node/diffServ/leaky-bucket.cc και .h στον κώδικα του ns-3) μπορούμε να δώσουμε τα ακόλουθα χαρακτηριστικά για ένα Leaky-Bucket:

- Την Ipv4 του κόμβου του οποίου την κίνηση θέλουμε να μορφοποιήσουμε
- Το DataRate σύμφωνα με το οποίο θα εξέρχονται τα πακέτα μέσα από το Bucket του Leaky Bucket. Θα πρέπει να αναφερθεί ότι ο ρυθμός με τον οποίο εξέρχονται είναι σταθερός (constant bit rate)

- Το mode σύμφωνα με το οποίο θα ορίζουμε τον τρόπο που θα μετράμε το μέγεθος της ουράς. Το mode μπορεί να είναι είτε πακέτα είτε Bytes.(Το default είναι πακέτα)
- Το MaxPacket όπου πρόκειται για μέγιστο πλήθος των πακέτων που μπορεί να χωρέσει το Bucket δηλαδή το μέγεθος της ουράς μας.
- Το MaxBytes όπου στην περίπτωση που επιλέξουμε ως Mode του bucket τα bytes δηλώνει το μέγιστο πλήθος των bytes που μπορεί να χωρέσει το bucket μας
- Τέλος την DSCP τιμή. Το πεδίο αυτό προς το παρόν παραμένει αχρησιμοποίητο αλλά σε μεταγενέστερη υλοποίηση των αλγορίθμων απομάκρυνσης των πακέτων από την ουρά μπορεί να χρησιμοποιηθεί.

Θα πρέπει να αναφερθεί ότι η κλάση του Leaky-Bucket είναι υποκλάση της Queue. Αυτό μας επιτρέπει τον χειρισμό του bucket ως μια ουρά αναμονής.

Μετά τον ορισμό των ιδιοτήτων του Leaky Bucket θα πρέπει να εγκατασταθεί σε κάποιον κόμβο ή σε ένα σύνολο κόμβων. Αυτό μπορεί να πραγματοποιηθεί με εύκολο τρόπο με την βοήθεια leaky-bucket-helper (το οποίο βρίσκεται: ns-3.9/src/helper/diffServ/leaky-bucket-helper.cc και .h στον κώδικα του ns-3). Ένα παράδειγμα ορισμού και εγκατάστασης ενός Leaky Bucket είναι:

```
LeakyBucketHelper myBucket;

myBucket.SetAttribute ("DataRate", StringValue("1Mbps"));
myBucket.SetAttribute("Ipv4Shape",Ipv4AddressValue
("10.1.1.1"));

myBucket.SetAttribute ("MaxPackets", UintegerValue (5));

myBucket.Install(p2pDevices.Get(1));
```

Παρατηρούμε ότι το install γίνεται σε έναν συγκεκριμένο netDevice. Ακολουθώντας το προγραμματιστικό στυλ που προϋπήρχε στον ns-3, το οποίο χρησιμοποιεί helpers για να εγκαταστήσει διάφορα χαρακτηριστικά και στοιχεία σε τοπολογίες για τον

πειραματισμό, έτσι και εμείς δημιουργήσαμε τον LeakyBucketHelper ο οποίος και είναι υπεύθυνος για την εύρεση του κόμβο στον οποίον ανήκει το συγκεκριμένο netDevice καθώς και για την εγκατάσταση του leaky bucket με τα χαρακτηριστικά που εμείς ορίσαμε.

1.14. Αρχεία που τροποποιήθηκαν

Η δομή του ns3 περιλαμβάνει μια κλάση net-device η οποία είναι υπερκλάση όλων των devices που μπορούν να χρησιμοποιηθούν στον ns-3. Η κλάση αυτή περιλαμβάνει μόνο virtual συναρτήσεις οι οποίες ορίζονται και υλοποιούνται στις υποκλάσεις της. Έτσι για τη δημιουργία του μηχανισμού του Leaky Bucket ορίσαμε πέντε virtual συναρτήσεις στην net-device.h (η οποία βρίσκεται: ns-3.9/src/node/net-device.h).Οι συναρτήσεις αυτές είναι οι ακόλουθες:

- ✓ `virtual Ptr<Node> GetShapedNode (void) const = 0;`
- ✓ `virtual void SetShapedNode (Ptr<Node> ShapedNode) = 0;`
- ✓ `virtual void EnqueueToLeakyBucket (Ptr<Packet> packet, Ptr<LeakyBucket> EnLeakyBucket) = 0;`
- ✓ `virtual void DoDequeueFromLeakyBucket (Time nextTime, Ptr<LeakyBucket> thisLeakyBucket) = 0;`
- ✓ `virtual void SheduleSendFormLeakyBucket (Ptr<LeakyBucket> currentLeakyBucket) = 0;`

Οι συναρτήσεις αυτές ορίζονται σε όλες της υποκλάσεις της net-device αλλά η ουσιαστική υλοποίηση του μηχανισμού- σε πρώτο στάδιο -πραγματοποιήθηκε μόνο στην point-to-point-net-device. Βέβαια μπορεί ο ίδιος κώδικας που βρίσκεται στην point-to-point-net-device να προστεθεί σε οποιαδήποτε άλλη υποκλάση της net-device και να χρησιμοποιηθεί από την αντίστοιχη συσκευή. Έτσι τα αρχεία που τροποποιήθηκαν και αφορούν την net-device και της υποκλάσεις της είναι:

- ns-3.9/src/node/net-device.cc και .h (πρόκειται για την υπερκλάση)
- ns-3.9/src/devices/wimax/wimax-net-device.cc και .h
- ns-3.9/src/devices/spectrum/aloha-noack-net-device.cc και .h

- ns-3.9/src/devices/spectrum/non-communicating-net-device.cc και .h
- ns-3.9/src/devices/wifi/wifi-net-device.cc και .h
- ns-3.9/src/devices/uan/model/uan-net-device.cc και .h
- ns-3.9/src/devices/mesh/mesh-point-device.cc και .h
- ns-3.9/src/devices/virtual-net-device/virtual-net-device.cc και .h
- ns-3.9/src/devices/bridge/bridge-net-device.cc και .h
- ns-3.9/src/devices/csma/csma-net-device.cc και .h
- ns-3.9/src/devices/tap-bridge/tap-bridge.cc και .h
- ns-3.9/src/devices/point-to-point/point-to-point-net-device.cc και .h
- ns-3.9/src/devices/emu/emu-net-device.cc και .h
- ns-3.9/src/node/simple-net-device.cc και .h
- ns-3.9/src/internet-stack/loopback-net-device.cc και .h

Εκτός από τις τροποποιήσεις στα devices θα έπρεπε κάπου να αποθηκεύονται οι δείκτες από τα leaky buckets καθώς και να δηλώνονται οι κόμβοι στους οποίους έχουν εγκατασταθεί τα leaky buckets. Για το λόγο αυτό η καταλληλότερη κλάση για τροποποίηση ήταν η node.cc (η οποία βρίσκεται: ns-3.9/src/node/node.cc και .h) στην οποία προσθέσαμε πέντε καινούργιες συναρτήσεις οι οποίες είναι:

- ✓ void AddShaper (Ptr<LeakyBucket> leaky, Ptr<NetDevice> ShapedDevice);
- ✓ std::vector<Ptr<LeakyBucket> > GetLeakyBucket(void) const;
- ✓ uint32_t GetNLeakyBucket (void) const;
- ✓ uint32_t GetNLeakyBucketForNode (void);
- ✓ void SetLeakyBucketForNode(Ptr <LeakyBucket> leakyForNode);

Οι συναρτήσεις αυτές χρησιμοποιούνται για να προσθέσουν κάποιο leaky bucket σε ένα συγκεκριμένο net-device του τρέχοντος κόμβου, για να αποθηκεύσουν τους δείκτες των leaky buckets σε διανύσματα και να δηλώσουν τους κόμβους που έχουν τα leaky buckets. Επίσης επιστρέφουν τον συνολικό αριθμό των buckets της τοπολογίας μας καθώς και τον αριθμό των buckets που έχουν προστεθεί σε έναν συγκεκριμένο κόμβο. Οι αριθμοί αυτοί χρησιμοποιούνται από άλλες συναρτήσεις για διάφορους ελέγχους.

Όσο αφορά στον κώδικα που έχει προστεθεί θα πρέπει να τονίσουμε πως σε κάθε γραμμή του περιλαμβάνεται όσο το δυνατόν λεπτομερέστερος σχολιασμό της λειτουργίας που επιτελεί. Επίσης θα πρέπει να αναφερθεί ότι ο κώδικας που προστέθηκε σε κάθε κλάση για τον μηχανισμό βρίσκεται ανάμεσα σε:

//////////////////petros_baltzis_diffserv

…
…
…
…
…
…

//////////////////petros_baltzis_diffserv_end

Πριν προχωρήσουμε στην επόμενη υποενότητα θα πρέπει να αναφέρουμε ότι για να μπορεί να δει ο compiler τα αρχεία που προσθέσαμε, δηλαδή τα leaky-bucket και leaky-bucket-helper θα πρέπει να γίνει κατάλληλη τροποποίηση στα αρχεία:

ns-3.9/src/node/wscript
ns-3.9/src/helper/wscript

Έτσι στο πρώτο αρχείο για να προχωρήσει ο compiler σε ανάγνωση του leaky-bucket.cc καθώς και του αντίστοιχου leaky-bucket.h προσθέσαμε τον ακόλουθο κώδικα στα κατάλληλα σημεία:

```
'diffServ/leaky-bucket.cc',
'diffServ/leaky-bucket.h',
```

Ενώ στο δεύτερο αρχείο για να προχωρήσει ο compiler σε ανάγνωση του leaky-bucket-helper.cc καθώς και του αντίστοιχου leaky-bucket-helper.h προσθέσαμε τον ακόλουθο κώδικα στα κατάλληλα σημεία:

```
'diffServ/leaky-bucket-helper.cc',
'diffServ/leaky-bucket-helper.h',
```

1.15. Προσθήκη του μηχανισμού στον ns-3

Για να γίνει η προσθήκη του μηχανισμού που υλοποιήσαμε στον ns-3 πρέπει, εφόσον κατεβάσει κάποιος από το 1.18.3 τον κώδικα που γράφτηκε, να ακολουθηθούν αυστηρά τα παρακάτω βήματα:

1. Τοποθέτηση των αρχείων leaky-bucket.cc και .h στην διαδρομή: ns-3.9/src/node/diffServ/
   (με δημιουργία του φακέλου diffServ) και αντικατάσταση του ns-3.9/src/node/wscript με το αντίστοιχο που σας παρέχεται στην ίδια διαδρομή. Εκτέλεση της εντολής ./waf configure -d optimized και έπειτα της ./waf
2. Αντικατάσταση των αρχείων node.cc και .h στην διαδρομή: ns-3.9/src/node/ με το αντίστοιχο που σας παρέχεται στην ίδια διαδρομή καθώς και όλων των αρχείων υποκλάσεων της net-device που βρίσκονται στις διαδρομές (που αναφέρονται στις παραπάνω σελίδες) με τα αντίστοιχα που σας παρέχονται στις ίδιες διαδρομές. Εκτέλεση της εντολής ./waf configure -d optimized και έπειτα της ./waf
3. Τοποθέτηση των αρχείων leaky-bucket-helper.cc και .h στην διαδρομή: ns-3.9/src/helper/diffServ/ (με δημιουργία του φακέλου diffServ) και αντικατάσταση του ns-3.9/src/helper/wscript με το αντίστοιχο που σας παρέχεται στην ίδια διαδρομή. Εκτέλεση της εντολής ./waf configure -d optimized και έπειτα της ./waf

Ακολουθώντας τα παραπάνω βήματα ο μηχανισμός τοποθετείτε με επιτυχία. Επομένως μπορούμε στον φάκελο scratch/ να προσθέσουμε αρχεία που περιέχουν τοπολογίες με leaky buckets για να πραγματοποιήσουμε τα πειράματά μας. Στο επόμενο κεφάλαιο παρατίθενται τα πειράματα που πραγματοποιήσαμε μέσω των οποίων αποδείξαμε την ορθή λειτουργία του μηχανισμού. Επίσης παρουσιάζεται και εξηγείται ο κώδικας των τοπολογιών που χρησιμοποιήσαμε για αυτά τα πειράματα. Μέσα σε αυτές τις τοπολογίες φαίνεται και ο τρόπος με τον οποίο ορίζουμε και εγκαθιστούμε κάποιο leaky bucket σε μια συσκευή δικτύου ενός κόμβου.

ΠΕΙΡΑΜΑΤΑ

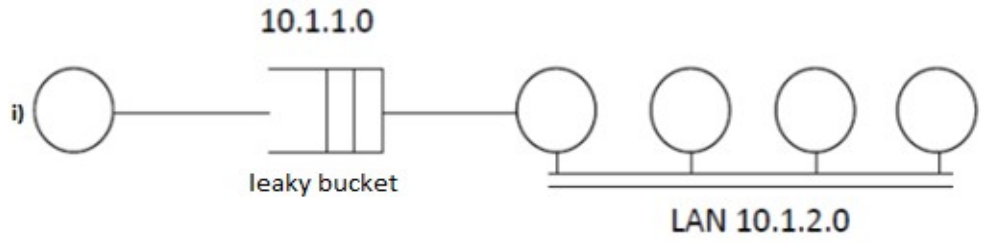
1.16. Εισαγωγή

Για τα πειράματά μας δημιουργήσαμε και χρησιμοποιήσαμε δυο διαφορετικές τοπολογίες ο κώδικας των οποίων παρουσιάζεται παρακάτω. Η πρώτη τοπολογία είναι σχετικά απλή η οποία περιλαμβάνει την σύνδεση ενός κόμβου με ένα δίκτυο LAN (5 κόμβων), ενώ η δεύτερη τοπολογία είναι αρκετά πιο πολύπλοκη προσομοιώνοντας καλύτερα τις συνθήκες που επικρατούν σε ένα πραγματικό δίκτυο. Έπειτα πραγματοποιήσαμε αρκετές μετρήσεις αποδεικνύοντας την ορθότητα του μηχανισμού leaky bucket που υλοποιήσαμε. Οι λεπτομέρειες των δύο αυτών τοπολογιών παρουσιάζονται στην 7.2 και 7.3 .

1.17. Πρώτη τοπολογία πειραμάτων

1.17.1. Παρουσίαση της τοπολογίας

Στην πρώτη τοπολογία η οποία παρουσιάζεται στην Εικόνα 12, προσομοιώνουμε την κίνηση πακέτων που προέρχονται από την αριστερή πλευρά της τοπολογίας και συγκεκριμένα από τον κόμβο *i)* προς την δεξιά πλευρά της τοπολογίας σε έναν από τους κόμβους του δικτύου LAN. Στην είσοδο του LAN έχουμε προσθέσει ένα leaky bucket το οποίο μορφοποιεί την κίνηση που παράγεται από τον κόμβο *i)* αποτρέποντας την συμφόρηση πακέτων εντός του δικτύου LAN.



Εικόνα 12 Απεικόνιση πρώτης τοπολογίας

1.17.2. Πειράματα και αποτελέσματα

Το πρώτο πείραμά μας χρησιμοποιήθηκε για να δοκιμάσει αν τα πακέτα που βρίσκονται μέσα στο bucket φεύγουν τις χρονικές στιγμές που καθορίζονται από το cbr που δώσαμε και η διαδικασία εξαγωγής των πακέτων είναι ανεξάρτητη από την

διαδικασία εισαγωγής των πακέτων στο bucket.Τα χαρακτηριστικά της κίνησης των πακέτων είναι τα ακόλουθα:

- Ο ρυθμός με τον οποίον καταφθάνουν
  τα πακέτα στο bucket είναι 10Mbps
- Ο ρυθμός με τον οποίον εξέρχονται
  τα πακέτα από το bucket είναι 1Mbps
- Μέγεθος του Bucket 5 πακέτα

Τα αποτελέσματα που προκύπτουν παρουσιάζονται στον παρακάτω πίνακα:

Πακέτο	Χρονική στιγμή στην οποία φτάνει το πακέτο	Χρονική στιγμή εισόδου στο Bucket	Χρονική στιγμή εξόδου από Bucket
1. Εισέρχεται	41.598ns	41.598ns	433.597ns
2. Εισέρχεται	82.397ns	82.397ns	825.596ns
3. Εισέρχεται	123.196ns	123.196ns	1.217.595ns
4. Εισέρχεται	163.995ns	163.995ns	1.609.594ns
5. Εισέρχεται	204.794ns	204.794ns	2.001.593ns
6. Απορρίπτεται	245.593ns	-------	-------
7. Απορρίπτεται	286.392ns	-------	-------
8. Απορρίπτεται	327.191ns	-------	-------
9. Απορρίπτεται	367.990ns	-------	-------
10. Απορρίπτεται	408.789ns	-------	-------

Από τα αποτελέσματα παρατηρούμε ότι η διαδικασία εισόδου των πακέτων στο bucket είναι ανεξάρτητη της διαδικασίας εξόδου. Επίσης παρατηρούμε ότι τα πακέτα εξέρχονται από το bucket με σταθερό ρυθμό. Συγκεκριμένα παρατηρούμε ότι:

2.001.593ns - 1.609.594ns = 391.999ns
1.609.594ns - 1.217.595ns = 391.999ns
1.217.595ns - 825.596ns = 391.999ns
825.596ns - 433.597ns = 391.999ns

Οπότε από το bucket εξέρχεται ένα πακέτο κάθε 391.999ns ενώ τα πακέτα που καταφθάνουν στο διάστημα που μεσολαβεί για την αποστολή ενός πακέτου όταν το Bucket είναι γεμάτο απορρίπτονται.

Στη συνέχεια πραγματοποιήθηκαν αρκετές μετρήσεις οι οποίες διακρίνονται σε δύο διαφορετικές κατηγορίες. Στην πρώτη από αυτές διατηρήσαμε όλα τα χαρακτηριστικά της παραγωγής και κίνησης των πακέτων σταθερά τροποποιώντας το

μέγεθος του leaky bucket και καταγράφοντας κάθε φορά τα αποτελέσματα που προέκυπταν. Συγκεκριμένα τα χαρακτηριστικά της κίνησης είναι τα ακόλουθα:

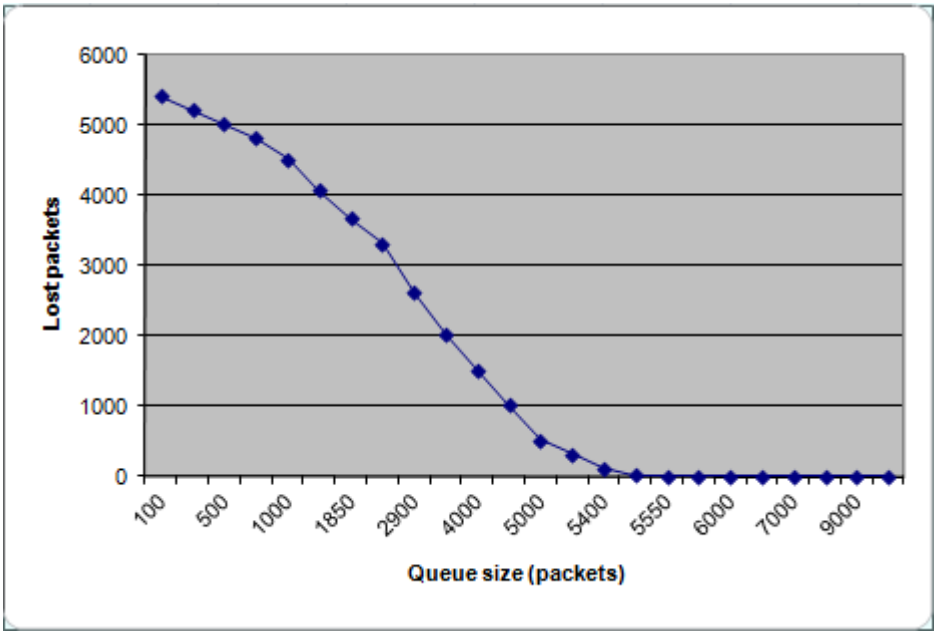
Για την παραγωγή πακέτων χρησιμοποιήσαμε το CBR του OnOffApplication με τα ακόλουθα χαρακτηριστικά:

- CBR (Σταθερός ρυθμός αποστολής bit) στα 10Mbps.
- Τα πακέτα παράγονται με τον παραπάνω ρυθμό για 5 sec και έπειτα για τα επόμενα 5 sec η πηγή παραμένει ανενεργή.
- Το application είναι ενεργό από 0 έως 15 sec

Για το leaky bucket έχουμε τα ακόλουθα χαρακτηριστικά:

- Ο ρυθμός εξερχόμενων πακέτων από τον κάδο είναι 1Mbps.
- Το μέγεθος του bucket ήταν διαφορετικό για την κάθε μέτρηση.

Τα αποτελέσματα παρουσιάζονται στην εικόνα που ακολουθεί (Εικόνα 13):



Εικόνα 13 Γραφική παράσταση Χαμένων πακέτων – Μέγεθος ουράς

Στην παραπάνω εικόνα παρουσιάζεται η γραφική παράσταση των χαμένων πακέτων ως προς το μέγεθος της ουράς. Παρατηρούμε ότι όσο μεγαλώνει το μέγεθος της ουράς του leaky bucket τόσο μικραίνει ο αριθμός των χαμένων πακέτων. Επίσης με δεδομένα τα χαρακτηριστικά της παραγωγής κίνησης των πακέτων υπάρχει ένα όριο στο μέγεθος της ουράς πάνω από το οποίο δεν υπάρχουν απώλειες πακέτων. Εφόσον τα πακέτα εξέρχονται από το bucket με ρυθμό 1Mbps φιλτράροντας τις ριπές από τις πηγές έχουμε ένα είδος μορφοποίησης της κίνησης.

Για τη δεύτερη κατηγορία πειραμάτων τροποποιήσαμε το ρυθμό παραγωγής πακέτων χρησιμοποιώντας για κάθε μέτρηση διαφορετικές τιμές ενώ όλα τα υπόλοιπα χαρακτηριστικά της τοπολογίας μας και του leaky bucket παρέμειναν τα ίδια. Συγκεκριμένα τα χαρακτηριστικά της κίνησης είναι:

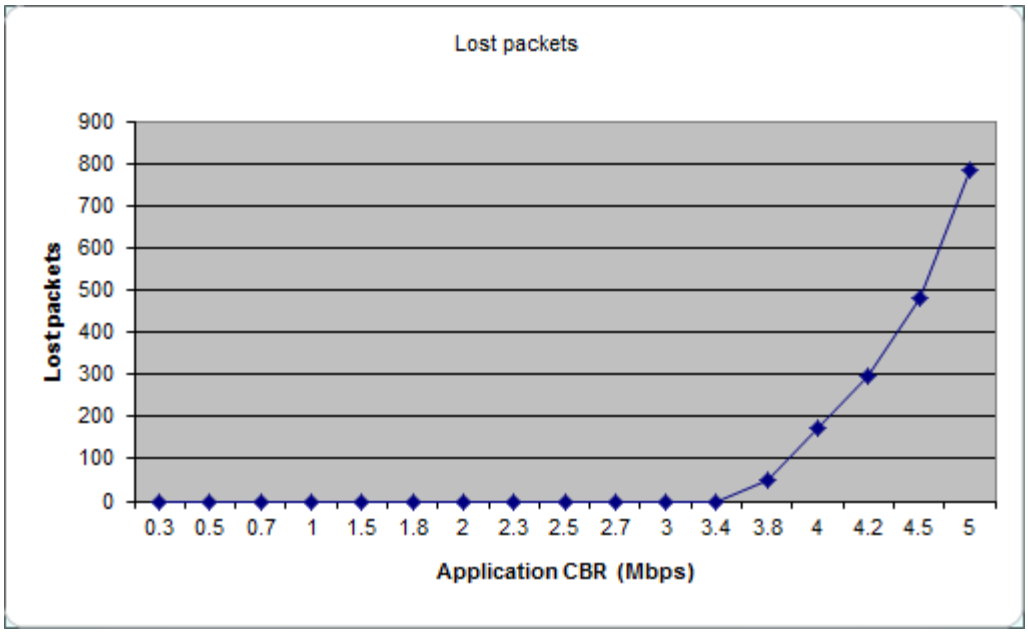
Για την παραγωγή πακέτων χρησιμοποιήσαμε και πάλι το CBR του OnOffApplication με τα ακόλουθα χαρακτηριστικά:

- CBR είναι διαφορετικό για την κάθε μας μέτρηση.
- Τα πακέτα παράγονται με τον ρυθμό που ορίζουμε κάθε φορά σε κάθε νέα μέτρηση για 1 sec και έπειτα για τα επόμενα 1 sec η πηγή παραμένει ανενεργή.
- Το application είναι ενεργό από 0 έως 15 sec

Για το leaky bucket έχουμε τα ακόλουθα χαρακτηριστικά:

- Ο ρυθμός εξερχόμενων πακέτων από τον κάδο είναι 1Mbps.
- Το μέγεθος του leaky bucket είναι σταθερό και ίσο με 1220 πακέτα.

Τα αποτελέσματα παρουσιάζονται στην εικόνα που ακολουθεί (Εικόνα 14):



Εικόνα 14 Χαμένων Πακέτων – Ρυθμό παραγωγής Πακέτων

Στην Εικόνα 14 παρουσιάζεται η γραφική παράσταση των χαμένων πακέτων ως προς το CBR του application. Παρατηρούμε ότι όσο αυξάνεται το CBR δηλαδή ο ρυθμός με τον οποίο παράγονται τα πακέτα και ξεπερνώντας ένα κατώτατο όριο τόσο αυξάνεται ο αριθμός των χαμένων πακέτων εφόσον αυτά πλέον δε χωρούν στο bucket και απορρίπτονται. Εφόσον τα πακέτα εξέρχονται από το bucket με ρυθμό 1Mbps φιλτράροντας τις ριπές της πηγής έχουμε ένα είδος μορφοποίησης της κίνησης.

1.17.3. Κώδικας της τοπολογίας

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/leaky-bucket-helper.h"
#include "ns3/flow-monitor-module.h"

// Script τοπολογίας
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 10.1.2.0


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int
main (int argc, char *argv[])
{
  NodeContainer p2pNodes;
  p2pNodes.Create (2);
  //Δημιουργία δικτύου lan
  NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1));
  csmaNodes.Create (3);

  //Δημιουργία point-to-point σύνδεση μεταξύ δύο κόμβων
  PointToPointHelper pointToPoint;
  //Ορισμός χαρακτηριστικών της point-to-point σύνδεσης
  pointToPoint.SetDeviceAttribute("DataRate",StringValu
  e("10Mbps"));

  NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);


  //Ορισμός χαρακτηριστικών καναλιού για το LAN δίκτυο
  CsmaHelper csma;
  csma.SetChannelAttribute("DataRate",StringValue("100M
  bps"));
  csma.SetChannelAttribute("Delay", TimeValue
  (NanoSeconds (6560)));
```

```
NetDeviceContainer csmaDevices;
  csmaDevices = csma.Install (csmaNodes);

  InternetStackHelper stack;
  stack.Install (p2pNodes.Get (0));
  stack.Install (csmaNodes);

  //Τοποθέτηση IP διευθύνσεων στους κόμβους
  της     //τοπολογίας
  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);
  address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer csmaInterfaces;
  csmaInterfaces = address.Assign (csmaDevices);

  //Χρησιμοποίηση του onoffaplication για την παραγωγή
  //κίνησης πακέτων μέσα στην τοπολογία
  ApplicationContainer apps0;
  OnOffHelper onOffHelper0  ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("10.1.2.4"), 9));

  //Ορισμός των χαρακτηριστικών της παραγωγής και
  //κίνησης των πακέτων
  onOffHelper0.SetAttribute ("DataRate", StringValue
  ("10Mbps"));

  onOffHelper0.SetAttribute ("PacketSize",
  UintegerValue (1));

  onOffHelper0.SetAttribute ("MaxBytes", UintegerValue
  (10));

  onOffHelper0.SetAttribute ("OnTime",
  RandomVariableValue (ConstantVariable (1)));

  onOffHelper0.SetAttribute ("OffTime",
  RandomVariableValue (ConstantVariable (0)));

  //Εγκατάσταση του application στον κόμβο που
  //επιθυμούμε
  apps0.Add(onOffHelper0.Install (p2pNodes.Get (0)));
```

```
//Δημιουργία του leaky bucket χρησιμοποιώντας
το //LeakyBucketHelper
LeakyBucketHelper myBucket;
  //Ορισμός του ρυθμού με τον οποίο θα εξέρχονται
  τα   //πακέτα από τον κάδο
  myBucket.SetAttribute ("DataRate", StringValue
  ("1Mbps"));
  //Ορισμός της IP διεύθυνσης του κόμβου του οποίου την
  //κίνηση επιθυμούμε να μορφοποιήσουμε
  myBucket.SetAttribute ("Ipv4Shape", Ipv4AddressValue
  ("10.1.1.1"));
  //Ορισμός του μεγέθους της ουράς
  myBucket.SetAttribute ("MaxPackets", UintegerValue
  (5));
  //Εγκατάσταση του leaky bucket στην συσκευή
  δικτύου //που επιθυμούμε
  myBucket.Install(p2pDevices.Get(1));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //Εγκατάσταση της flowmonitor σε όλους του κόμβους
  //για την στατιστική καταγραφή των πακέτων που
  //κινούνται μέσα στην τοπολογία
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();

  //Έναρξη της προσομοίωσης
  Simulator::Stop (Seconds (30.0));
  Simulator::Run ();

  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier =
  DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifier
  ());

  std::map<FlowId, FlowMonitor::FlowStats> stats =
  monitor->GetFlowStats ();
```

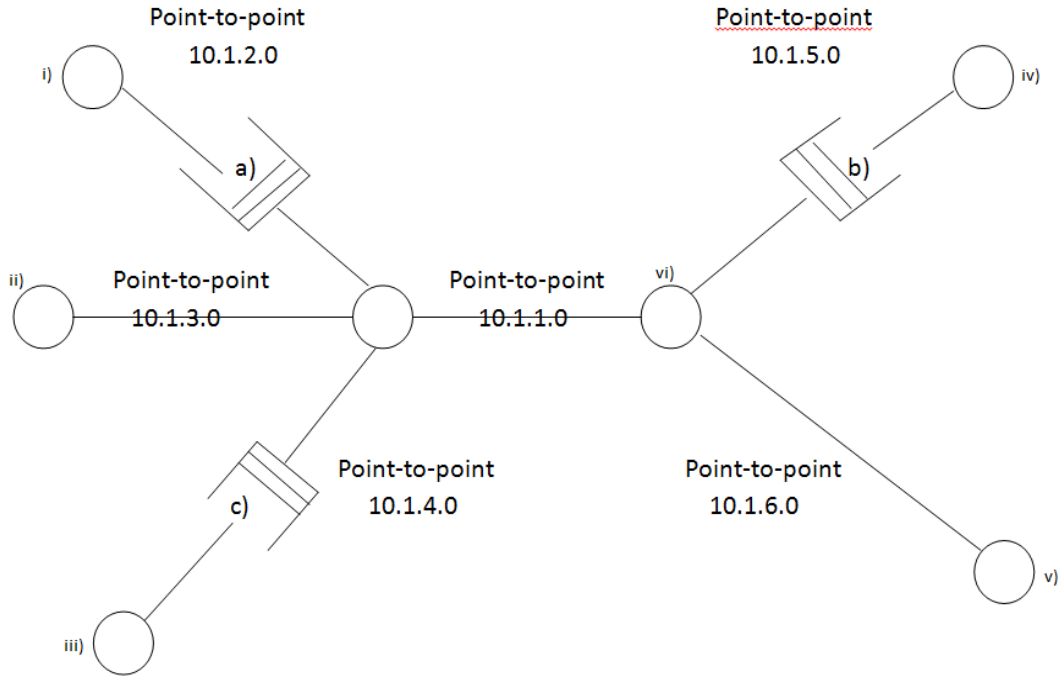
```
//Για κάθε μια από τις ροές που δημιουργήθηκαν
//τυπώνουμε τα στοιχεία που μας ενδιαφέρουν
for (std::map<FlowId,
FlowMonitor::FlowStats>::const_iterator i =
stats.begin (); i != stats.end (); ++i)
{
   Ipv4FlowClassifier::FiveTuple t = classifier-
   >FindFlow (i->first);
   //Τυπώνεται την IP του κόμβου προορισμού
   std::cout << "Flow " << i->first << " (" <<
   t.sourceAddress << " -> " << t.destinationAddress
   << ")\n";
   //Τυπώνεται τα bytes που μεταδόθηκαν
   std::cout << "  Tx Bytes:   " << i->second.txBytes
   << "\n";
   //Τυπώνεται τα bytes που έφτασαν στον προορισμό
   std::cout << "  Rx Bytes:   " << i->second.rxBytes
   << "\n";
   //Τυπώνεται τα πακέτα που μεταδόθηκαν
   std::cout << "  Tx Packets:  " << i-
   >second.txPackets << "\n";
   //Τυπώνεται τα πακέτα που έφτασαν στον προορισμό
   std::cout << "  Rx Packets:  " << i-
   >second.rxPackets << "\n";
   //Τυπώνεται η συνολική καθυστέρηση της ροής
   std::cout << "  delaySum:  " << i->second.delaySum
   << "\n";

}


Simulator::Destroy ();
return 0;
}
```

1.18. Δεύτερη τοπολογία πειραμάτων

1.18.1. Παρουσίαση τοπολογίας

Η δεύτερη τοπολογία είναι αρκετά πιο πολύπλοκη παρουσιάζοντας με αυτόν τον τρόπο ένα πιο ρεαλιστικό δίκτυο. Στην τοπολογία αυτή που παρουσιάζεται στην ακόλουθη εικόνα (Εικόνα 15) προσομοιώνεται ένα πλήθος από ροές κυκλοφορίας. Συγκεκριμένα οι ροές είναι οι ακόλουθες:

1. Από τον κόμβο i) στον κόμβο v)
2. Από τον κόμβο ii) στον κόμβο vi)
3. Από τον κόμβο iii) στον κόμβο iv)
4. Από τον κόμβο iv) στον κόμβο i)



Εικόνα 15 Απεικόνιση δεύτερης τοπολογίας

Όλες οι ροές κυκλοφορίας πακέτων διαμορφώνονται από διαφορετικά leaky buckets με διαφορετικά χαρακτηριστικά εκτός από την ροή πακέτων που παράγεται στον κόμβο ii, στην οποία δεν έχει εγκατασταθεί κάποιο leaky bucket.

1.18.2. Πειράματα και αποτελέσματα

Το πρώτο από τα πειράματα που έγιναν σε αυτήν την τοπολογία πραγματοποιήθηκε για να δοκιμάσει την ορθότητα της λειτουργίας του μηχανισμού, όταν γίνεται εγκατάσταση leaky buckets σε περισσότερους από έναν κόμβους και ο κάθε κόμβος έχει εγκατεστημένα περισσότερα από ένα leaky bucket.

Τα χαρακτηριστικά που αφορούν τα leaky buckets a,b και c (βλέπε Εικόνα 15) του συγκεκριμένου πειράματος είναι τα ακόλουθα:

Για το a) leaky bucket:

- Ρυθμός με τον οποίον καταφθάνουν τα πακέτα στο bucket 10Mbps
- Ρυθμός με τον οποίον εξέρχονται τα πακέτα στο bucket 1Mbps
- Μέγεθος του Bucket 5 πακέτα
- Διεύθυνση του κόμβου που μορφοποιεί την κίνηση είναι 10.1.2.1

Για το b) leaky bucket:

- Ρυθμός με τον οποίον καταφθάνουν τα πακέτα στο bucket 10Mbps
- Ρυθμός με τον οποίον εξέρχονται τα πακέτα στο bucket 5Mbps
- Μέγεθος του Bucket 5 πακέτα
- Διεύθυνση του κόμβου που μορφοποιεί την κίνηση είναι 10.1.4.1

Για το c) leaky bucket:

- Ρυθμός με τον οποίον καταφθάνουν τα πακέτα στο bucket 10Mbps
- Ρυθμός με τον οποίον εξέρχονται τα πακέτα στο bucket 2Mbps
- Μέγεθος του Bucket 2 πακέτα
- Διεύθυνση του κόμβου που μορφοποιεί την κίνηση είναι 10.1.5.1

Θα πρέπει να αναφέρουμε ότι δημιουργούμε τέσσερις διαφορετικές ροές χρησιμοποιώντας την onoff application του ns-3 με τα ίδια χαρακτηριστικά. Συγκεκριμένα και οι τρεις παράγουν πακέτα με ρυθμό 10Mbps και συνολικά 10 πακέτα.

Έτσι σύμφωνα με τα αποτελέσματα της flow Stat έχουμε:

```
Flow 1 (10.1.2.1 -> 10.1.6.2)
-----------------------------------
  Tx Bytes:   10720
  Rx Bytes:   5360
  Tx Packets:  10
  Rx Packets:  5
  delaySum:  132.524.766ns
```

```
Flow 2 (10.1.3.1 -> 10.1.1.2)
-------------------------------------
  Tx Bytes:   10720
  Rx Bytes:   10720
  Tx Packets:  10
  Rx Packets:  10
  delaySum:  36.167.960ns


Flow 3 (10.1.4.1 -> 10.1.5.2)
-------------------------------------
  Tx Bytes:   10720
  Rx Bytes:   9648
  Tx Packets:  10
  Rx Packets:  9
  delaySum:  81.345.507ns

Flow 4 (10.1.5.1 -> 10.1.2.1)
-------------------------------------
  Tx Bytes:   10720
  Rx Bytes:   3216
  Tx Packets:  10
  Rx Packets:  3
  delaySum:  28.545.597ns
```

Παρατηρούμε λοιπόν ότι καταγράφεται απώλεια πακέτων στις ροές στις οποίες εφαρμόζεται κάποιο Leaky-Bucket ενώ αντίθετα στη ροή 2 στην οποία δεν εφαρμόζεται κανένα leaky Bucket δεν υπάρχει καμία απώλεια πακέτων. Αυτό είναι φυσιολογικό καθώς και ο ρυθμός αποστολής των πακέτων από το Bucket είναι μικρότερος σε σχέση με τον ρυθμό με τον οποίο καταφθάνουν τα πακέτα στο bucket αλλά και το μέγεθος του bucket είναι μικρότερο σε σχέση με το αριθμό των πακέτων που παράγονται από τις ροές. Έτσι στη ροή 1 απορρίπτονται 5 πακέτα καθώς το μέγεθος της ουράς είναι 5 πακέτα ενώ ο ρυθμός αποστολής από το bucket είναι 1Mbps. Στη ροή 3 απορρίπτεται 1 πακέτο καθώς το μέγεθος της ουράς είναι 5 πακέτα ενώ ο ρυθμός αποστολής από το bucket είναι 5Mbps. Τέλος στην ροή 4 απορρίπτονται 7 πακέτα καθώς το μέγεθος της ουράς είναι 2 πακέτα ενώ ο ρυθμός αποστολής από το bucket είναι 2Mbps. Έτσι μπορούμε να εφαρμόσουμε όσα leaky-buckets επιθυμούμε σε έναν ή και περισσότερους κόμβους χωρίς να επηρεάζεται η λειτουργία του ενός bucket από την λειτουργία του άλλου.

Έπειτα από την επιβεβαίωση της ορθής λειτουργίας πολλαπλών leaky buckets σε μια τοπολογία πραγματοποιήσαμε και εδώ δύο διαφορετικά σετ πειραμάτων. Για την πρώτη σειρά πειραμάτων τα χαρακτηριστικά είναι τα εξής:

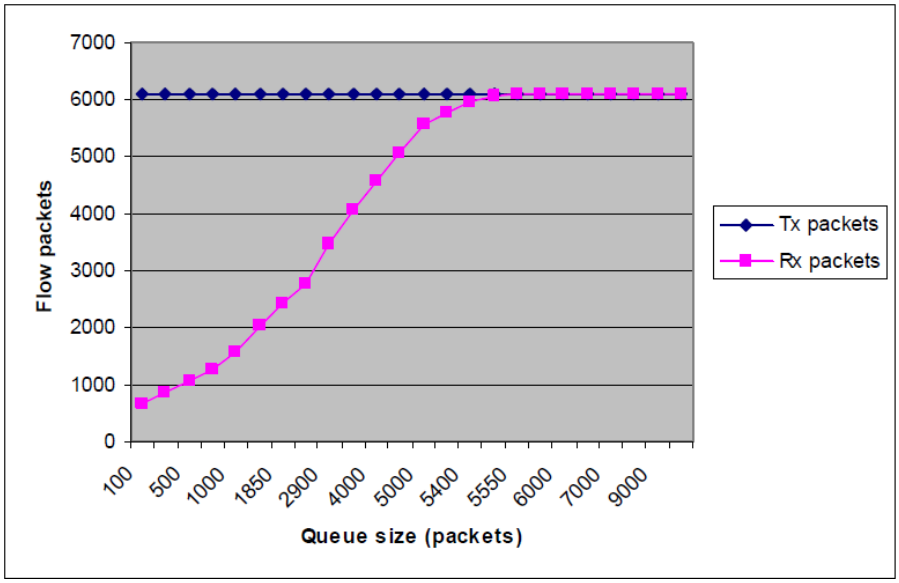
- Ο ρυθμός με τον οποίο εξέρχονται τα πακέτα από τα leaky buckets είναι στα 1Mbps
- Το μέγεθος των leaky buckets είναι 6103 πακέτα.
- Το CBR (χρησιμοποιώντας το onoffAplication) είναι στα 10Mbps
- Τα πακέτα παράγονται με τον παραπάνω ρυθμό για 5 sec και έπειτα για τα επόμενα 5 sec η πηγή παραμένει ανενεργή.
- Η εφαρμογή είναι ενεργή από 0 έως 15 sec.

Για τη δεύτερη σειρά πειραμάτων, τα χαρακτηριστικά είναι τα εξής:

- Ο ρυθμός με τον οποίο εξέρχονται τα πακέτα από τα leaky buckets είναι στα 1Mbps
- Το μέγεθος των leaky buckets είναι: 800 πακέτα για το a), 6103 πακέτα για το b), 3000 πακέτα για το c).
- Το CBR (χρησιμοποιώντας το onoffAplication) ήταν στα 10Mbps
- Τα πακέτα παράγονται με τον παραπάνω ρυθμό για 5 sec και έπειτα για τα επόμενα 5 sec η πηγή παραμένει ανενεργή.
- Η εφαρμογή είναι ενεργή από 0 έως 15 sec.

Οι μετρήσεις οι οποίες θα καθορίσουν κατά πόσο ο leaky buckets αλγόριθμος λειτουργεί με τον αναμενόμενο τρόπο μπορούν να συνοψιστούν σε δύο τομείς: Πρώτον, ένας μεγαλύτερος κάδος θα πρέπει να έχει την ικανότητα να χειριστεί μεγαλύτερες ροές πακέτων χωρίς να ξεχειλίζει και ως εκ τούτου να μην χάνονται πακέτα. Δεύτερον, όσο μια ροή πακέτων μεγαλώνει, θα πρέπει να υπάρχει ένα κατώτατο όριο ρυθμού όπου ο ρυθμός ροής πακέτων συναντά τον ρυθμό αποστολής των πακέτων από το bucket.

Θα πρέπει να τονίσουμε πως παρόμοια είναι τα αποτελέσματα και στους δύο τύπους τοπολογιών που περιγράφονται ανωτέρω, και ένα από αυτά παρουσιάζεται στην Εικόνα 16. Η εικόνα αυτή δείχνει πώς ο αριθμός των ληφθέντων πακέτων επηρεάζεται από το μέγεθος του leaky bucket. Όπως μπορούμε να δούμε, υπάρχει ένα όριο κάτω από το οποίο τα πακέτα χάνονται, και το όριο αυτό αντιστοιχεί στα πακέτα που συγκεντρώθηκαν στην ουρά εξαιτίας της εξομάλυνσης των καταιγιστικών κινήσεων που εκτελείται από τον μηχανισμό του leaky bucket. Εάν το μέγεθος του leaky bucket είναι αρκετά μεγάλο για να κρατήσει όλα τα εκκρεμεί πακέτα, αυτά προωθούνται αργότερα και έτσι δε χάνονται πακέτα.



Εικόνα 16 Πακέτα ροής – Μέγεθος ουράς

1.18.3. Κώδικας τοπολογίας

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/leaky-bucket-helper.h"
#include "ns3/flow-monitor-module.h"

// Default Network Topology
/*
//
//       n2                                  n5
//         \ 10.1.2.0                      / 10.1.5.0
//          \                             /
//           \                           /
//            \                         /
//   10.1.3.0  \       10.1.1.0       /
// n3 --------- n0 -------------- n1
//             /                   \
//            /                     \
//           /                       \
//          /                         \
//         /10.1.4.0                   \ 10.1.6.0
//       n4                             n6
//
*/
```

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int
main (int argc, char *argv[])
{
/////////////////////////////////////////////////////////
///
//Ορισμός των κόμβων που βρίσκονται αριστερά του n0
//
/////////////////////////////////////////////////////////
//
//Δημιουργία του n2
  NodeContainer p2pNodes2;
  p2pNodes2.Create (2);
  //Ορισμός των χαρακτηριστικών της συσκευής σύνδεσης
  PointToPointHelper pointToPoint2;
  pointToPoint2.SetDeviceAttribute ("DataRate",
  StringValue ("10Mbps"));

  //Εγκατάσταση του net device στον επιθυμητό κόμβο n2
  NetDeviceContainer p2pDevices2;
  p2pDevices2 = pointToPoint2.Install (p2pNodes2);

//Δημιουργία του n3
  NodeContainer p2pNodes3;
  p2pNodes3.Create (1);
  p2pNodes3.Add (p2pNodes2.Get (1));

  //Ορισμός των χαρακτηριστικών της συσκευής σύνδεσης
  PointToPointHelper pointToPoint3;
  pointToPoint3.SetDeviceAttribute ("DataRate",
  StringValue ("10Mbps"));

  //Εγκατάσταση του net device στον επιθυμητό κόμβο n3
  NetDeviceContainer p2pDevices3;
  p2pDevices3 = pointToPoint3.Install (p2pNodes3);

//Δημιουργία του n4
  NodeContainer p2pNodes4;
  p2pNodes4.Create (1);
  p2pNodes4.Add (p2pNodes2.Get (1));

  //Ορισμός των χαρακτηριστικών της συσκευής σύνδεσης
  PointToPointHelper pointToPoint4;
  pointToPoint4.SetDeviceAttribute ("DataRate",
  StringValue ("10Mbps"));
```

```
  //Εγκατάσταση του net device στον επιθυμητό κόμβο n4
  NetDeviceContainer p2pDevices4;
  p2pDevices4 = pointToPoint4.Install (p2pNodes4);

////////////////////////////////////////////////////
//
//Ορισμός των helper και των ιδιοτήτων των router n0,n1
//
////////////////////////////////////////////////////

  NodeContainer p2pNodes01;
  p2pNodes01.Add (p2pNodes2.Get (1));
  p2pNodes01.Create (1);

  //Ορισμός των χαρακτηριστικών της συσκευής σύνδεσης
  PointToPointHelper pointToPoint01;
  pointToPoint01.SetDeviceAttribute ("DataRate",
  StringValue ("10Mbps"));

  //Εγκατάσταση του net device στον επιθυμητό κόμβο n1
  NetDeviceContainer p2pDevices01;
  p2pDevices01 = pointToPoint01.Install (p2pNodes01);

//
//
////////////////////////////////////////////////////
//
//Ορισμός των κόμβων που βρίσκονται δεξιά του n1
//
////////////////////////////////////////////////////
//
//Δημιουργία του n5
  NodeContainer p2pNodes5;
  p2pNodes5.Create (1);
  p2pNodes5.Add (p2pNodes01.Get (1));

  //Ορισμός των χαρακτηριστικών της συσκευής σύνδεσης
  PointToPointHelper pointToPoint5;
  pointToPoint5.SetDeviceAttribute ("DataRate",
  StringValue ("10Mbps"));

  //Εγκατάσταση του net device στον επιθυμητό κόμβο n5
  NetDeviceContainer p2pDevices5;
  p2pDevices5 = pointToPoint5.Install (p2pNodes5);
```

```
//Δημιουργία του n6
  NodeContainer p2pNodes6;
  p2pNodes6.Create (1);
  p2pNodes6.Add (p2pNodes01.Get (1));

  //Ορισμός των χαρακτηριστικών της συσκευής σύνδεσης
  PointToPointHelper pointToPoint6;
  pointToPoint6.SetDeviceAttribute ("DataRate",
  StringValue ("10Mbps"));

  //Εγκατάσταση του net device στον επιθυμητό κόμβο n6
  NetDeviceContainer p2pDevices6;
  p2pDevices6 = pointToPoint6.Install (p2pNodes6);
////////////////////////////////////////////////////
///
//Ορισμός και αντιστοίχιση IP διευθύνσεων στις
//συσκευές δικτύου των κόμβων της τοπολογίας
///
////////////////////////////////////////////////////
InternetStackHelper stack;
  stack.Install (p2pNodes2.Get (0));
  stack.Install (p2pNodes3.Get (0));
  stack.Install (p2pNodes4.Get (0));
  stack.Install (p2pNodes01);
  stack.Install (p2pNodes5.Get (0));
  stack.Install (p2pNodes6.Get (0));

  Ipv4AddressHelper address;
  address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces2;
  p2pInterfaces2 = address.Assign (p2pDevices2);

  address.SetBase ("10.1.3.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces3;
  p2pInterfaces3 = address.Assign (p2pDevices3);

  address.SetBase ("10.1.4.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces4;
  p2pInterfaces4 = address.Assign (p2pDevices4);

  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces01;
  p2pInterfaces01 = address.Assign (p2pDevices01);

  address.SetBase ("10.1.5.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces5;
  p2pInterfaces5 = address.Assign (p2pDevices5);

  address.SetBase ("10.1.6.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces6;
  p2pInterfaces6 = address.Assign (p2pDevices6);
```

```
//
////////////////////////////////////////////////////
//
//Δημιουργία κίνησης πακέτων στο δίκτυο μέσω της
OnOff //application του ns-3
//
////////////////////////////////////////////////////
//
//Δημιουργία κίνησης από τον n2 στον n6
ApplicationContainer apps1;
  //Ορισμός διεύθυνσης προορισμού της κίνησης
  OnOffHelper onOffHelper1  ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("10.1.6.2"), 9));

  //Ορισμός του ρυθμού παραγωγής κίνησης
  onOffHelper1.SetAttribute ("DataRate", StringValue
  ("10Mbps"));

  //Ορισμός μεγέθους πακέτων
  onOffHelper1.SetAttribute ("PacketSize",
  UintegerValue (1024));

  //Ορισμός μέγιστου αριθμού πακέτων
  onOffHelper1.SetAttribute ("MaxBytes", UintegerValue
  (10240));

  //Ορισμός διαστημάτων όπου η πηγή είναι ενεργή
  και //ανενεργή
  onOffHelper1.SetAttribute ("OnTime",
  RandomVariableValue (ConstantVariable (1)));
  onOffHelper1.SetAttribute ("OffTime",
  RandomVariableValue (ConstantVariable (0)));
  //Εγκατάσταση του application στον επιθυμητό κόμβο
  apps1.Add(onOffHelper1.Install (p2pNodes2.Get (0)));

//
//Δημιουργία κίνησης από τον n3 στον n1
ApplicationContainer apps2;
  //Ορισμός διεύθυνσης προορισμού της κίνησης
  OnOffHelper onOffHelper2  ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("10.1.1.2"), 9));

  //Ορισμός του ρυθμού παραγωγής κίνησης
  onOffHelper2.SetAttribute ("DataRate", StringValue
  ("10Mbps"));

  //Ορισμός μεγέθους πακέτων
  onOffHelper2.SetAttribute ("PacketSize",
  UintegerValue (1024));
```

```
  //Ορισμός μέγιστου αριθμού πακέτων
  onOffHelper2.SetAttribute ("MaxBytes", UintegerValue
  (10240));

  //Ορισμός διαστημάτων όπου η πηγή είναι ενεργή
  και //ανενεργή
  onOffHelper2.SetAttribute ("OnTime",
  RandomVariableValue (ConstantVariable (1)));
  onOffHelper2.SetAttribute ("OffTime",
  RandomVariableValue (ConstantVariable (0)));
  //Εγκατάσταση του application στον επιθυμητό κόμβο
  apps2.Add(onOffHelper2.Install (p2pNodes3.Get (0)));

//
//Δημιουργία κίνησης από τον n4 στον n5
ApplicationContainer apps0;
  //Ορισμός διεύθυνσης προορισμού της κίνησης
  OnOffHelper onOffHelper0  ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("10.1.5.2"), 9));

  //Ορισμός του ρυθμού παραγωγής κίνησης
  onOffHelper0.SetAttribute ("DataRate", StringValue
  ("10Mbps"));

  //Ορισμός μεγέθους πακέτων
  onOffHelper0.SetAttribute ("PacketSize",
  UintegerValue (1024));

  //Ορισμός μέγιστου αριθμού πακέτων
  onOffHelper0.SetAttribute ("MaxBytes", UintegerValue
  (10240));

  //Ορισμός διαστημάτων όπου η πηγή είναι ενεργή
  και //ανενεργή
  onOffHelper0.SetAttribute ("OnTime",
  RandomVariableValue (ConstantVariable (1)));
  onOffHelper0.SetAttribute ("OffTime",
  RandomVariableValue (ConstantVariable (0)));
  //Εγκατάσταση του application στον επιθυμητό κόμβο
  apps0.Add(onOffHelper0.Install (p2pNodes4.Get (0)));
```

```
//
//Δημιουργία κίνησης από τον n5 στον n2
ApplicationContainer apps3;
  //Ορισμός διεύθυνσης προορισμού της κίνησης
  OnOffHelper onOffHelper0  ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("10.1.2.1"), 9));

  //Ορισμός του ρυθμού παραγωγής κίνησης
  onOffHelper0.SetAttribute ("DataRate", StringValue
  ("10Mbps"));

  //Ορισμός μεγέθους πακέτων
  onOffHelper0.SetAttribute ("PacketSize",
  UintegerValue (1024));

  //Ορισμός μέγιστου αριθμού πακέτων
  onOffHelper0.SetAttribute ("MaxBytes", UintegerValue
  (10240));

  //Ορισμός διαστημάτων όπου η πηγή είναι ενεργή
  και //ανενεργή
  onOffHelper0.SetAttribute ("OnTime",
  RandomVariableValue (ConstantVariable (1)));
  onOffHelper0.SetAttribute ("OffTime",
  RandomVariableValue (ConstantVariable (0)));
  //Εγκατάσταση του application στον επιθυμητό κόμβο
  apps0.Add(onOffHelper0.Install (p2pNodes5.Get (0)));

//
////////////////////////////////////////////////////////
//
//Δημιουργία και εγκατάσταση των leaky buckets
στους //κόμβους n0 και n1
//
////////////////////////////////////////////////////////
//
//Δημιουργία leaky bucket στον κόμβο n0 το οποίο
//μορφοποιεί την κίνηση που εισέρχεται στο δίκτυο
από //το κόμβο n2 με ip:10.1.2.1
LeakyBucketHelper myBucket;
  //Ορισμός του ρυθμού εξόδου δεδομένων από το bucket
  myBucket.SetAttribute ("DataRate", StringValue
  ("1Mbps"));

  //Ορισμός της IP διεύθυνσης του κόμβου του οποίου
  η //κίνηση επιθυμούμε να μορφοποιηθεί
  myBucket.SetAttribute ("Ipv4Shape", Ipv4AddressValue
  ("10.1.2.1"));
```

```
  //Ορισμός του μέγιστου πλήθους πακέτων της ουράς
  myBucket.SetAttribute ("MaxPackets", UintegerValue
  (5));

  //Εγκατάσταση του leaky bucket στον κόμβο n0
  myBucket.Install(p2pDevices2.Get(1));


//Δημιουργία leaky bucket στον κόμβο n0 το οποίο
//μορφοποιεί την κίνηση που εισέρχεται στο δίκτυο
από //το κόμβο n4 με ip:10.1.4.1
LeakyBucketHelper myBucket2;
  //Ορισμός του ρυθμού εξόδου δεδομένων από το bucket
  myBucket2.SetAttribute ("DataRate", StringValue
  ("5Mbps"));

  //Ορισμός της IP διεύθυνσης του κόμβου του οποίου
  η //κίνηση επιθυμούμε να μορφοποιηθεί
  myBucket2.SetAttribute ("Ipv4Shape", Ipv4AddressValue
  ("10.1.4.1"));

  //Ορισμός του μέγιστου πλήθους πακέτων της ουράς
  myBucket2.SetAttribute ("MaxPackets", UintegerValue
  (5));

  //Εγκατάσταση του leaky bucket στον κόμβο n0
  myBucket2.Install(p2pDevices4.Get(1));


//Δημιουργία leaky bucket στον κόμβο n1 το οποίο
//μορφοποιεί την κίνηση που εισέρχεται στο δίκτυο
από //το κόμβο n5 με ip:10.1.5.1
LeakyBucketHelper myBucket3;
  //Ορισμός του ρυθμού εξόδου δεδομένων από το bucket
  myBucket3.SetAttribute ("DataRate", StringValue
  ("2Mbps"));

  //Ορισμός της IP διεύθυνσης του κόμβου του οποίου
  η //κίνηση επιθυμούμε να μορφοποιηθεί
  myBucket3.SetAttribute ("Ipv4Shape", Ipv4AddressValue
  ("10.1.5.1"));

  //Ορισμός του μέγιστου πλήθους πακέτων της ουράς
  myBucket3.SetAttribute ("MaxPackets", UintegerValue
  (2));

  //Εγκατάσταση του leaky bucket στον κόμβο n0
  myBucket3.Install(p2pDevices5.Get(1));
```

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //Εγκατάσταση της flowmonitor σε όλους του κόμβους
  //για την στατιστική καταγραφή των πακέτων που
  //κινούνται μέσα στην τοπολογία
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();

  //Έναρξη της προσομοίωσης
  Simulator::Stop (Seconds (30.0));
  Simulator::Run ();

  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier =
  DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifie
  r ());

  std::map<FlowId, FlowMonitor::FlowStats> stats =
  monitor->GetFlowStats ();
//Για κάθε μια από τις ροές που δημιουργήθηκαν
//τυπώνουμε τα στοιχεία που μας ενδιαφέρουν
for (std::map<FlowId,
FlowMonitor::FlowStats>::const_iterator i =
stats.begin (); i != stats.end (); ++i)
{
   Ipv4FlowClassifier::FiveTuple t = classifier-
   >FindFlow (i->first);
   //Τυπώνεται την IP του κόμβου προορισμού
   std::cout << "Flow " << i->first << " (" <<
   t.sourceAddress << " -> " << t.destinationAddress
   << ")\n";
   //Τυπώνεται τα bytes που μεταδόθηκαν
   std::cout << "  Tx Bytes:   " << i->second.txBytes
   << "\n";
   //Τυπώνεται τα bytes που έφτασαν στον προορισμό
   std::cout << "  Rx Bytes:   " << i->second.rxBytes
   << "\n";
   //Τυπώνεται τα πακέτα που μεταδόθηκαν
   std::cout << "  Tx Packets:  " << i-
   >second.txPackets << "\n";
   //Τυπώνεται τα πακέτα που έφτασαν στον προορισμό
   std::cout << "  Rx Packets:  " << i-
   >second.rxPackets << "\n";
   //Τυπώνεται η συνολική καθυστέρηση της ροής
   std::cout << "  delaySum:  " << i->second.delaySum
   << "\n";
}
Simulator::Destroy ();
return 0;
}
```

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Στην εργασία αυτή παρουσιάσαμε την υλοποίηση του μηχανισμού leaky bucket για το δημοφιλές περιβάλλον προσομοίωσης ns-3 και δείξαμε πως αυτός ο αλγόριθμος μπορεί να χρησιμοποιηθεί για να διαμορφώσει μια κυκλοφορία στο πλαίσιο μιας συνολικής DiffServ αρχιτεκτονικής. Έχουμε επίσης παρουσιάσει μέσω διαφόρων πειραμάτων χρησιμοποιώντας διαφορετικές τοπολογίες τη χρήση αυτής της επέκτασης και την έχουμε συγκρίνει με μεταδόσεις που δεν ήταν μορφοποιημένες. Θα πρέπει να αναφερθεί πως το σημαντικότερο έργο της εργασίας αυτής έγκειται στο γεγονός ότι η λειτουργία σε DiffServ περιβάλλοντα είναι δύσκολη χρησιμοποιώντας πραγματικά δίκτυα με πραγματικές συνθήκες. Για να επιτευχθεί κάτι τέτοιο θα ήταν πολύ πιθανό να απαιτούνταν ανάπτυξη τόσο σε υλικό όσο και σε λογισμικό, ενώ για την διεξαγωγή ευρείας κλίμακας πειραμάτων θα ήταν απαραίτητο ένα μεγάλο τμήμα εξοπλισμού ανεβάζοντας πολύ υψηλά το κόστος. Ως εκ τούτου, η χρήση της προσομοίωσης είναι απαραίτητη και ιδιαίτερα χρήσιμη ως ένα πρώτο βήμα πριν από την εφαρμογή και τη δοκιμή σε μια πραγματική τοπολογία δικτύου.

Παρατηρούμε, λοιπόν, ότι με την προσθήκη του μηχανισμού που υλοποιήσαμε, μπορούμε να προσομοιώσουμε μία οποιαδήποτε τοπολογία και να προσθέσουμε ένα ή και περισσότερα leaky buckets με τα επιθυμητά χαρακτηριστικά σε έναν ή και περισσότερους κόμβους επιτυγχάνοντας ένα είδος μορφοποίηση της κίνησης απαλείφοντας τις ριπές και τις εκρήξεις οι οποίες μπορούν να τύχουν σε ένα δίκτυο.

Μελλοντικές εργασίες μας στον τομέα αυτό περιλαμβάνουν την ενίσχυση του περιβάλλοντος προσομοίωσης με μηχανισμούς χρονοπρογραμματισμού όπως ο Modified Deficit Round Robin (MDRR) και άλλες εναλλακτικές λύσεις, την ανάπτυξη εργαλείων για τη μέτρηση και ταξινόμηση των πακέτων, και την εφαρμογή των φορέων που χειρίζονται τη διαχείριση των υπηρεσιών DiffServ, όπως το εύρος ζώνης. Σκοπεύουμε επίσης να συγκρίνουμε την ακρίβεια προσομοίωσης μεταξύ των ns-2 και ns-3 και να αξιολογήσουμε τις πιθανές βελτιώσεις της νεότερης αρχιτεκτονικής προσομοίωσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] http://www.isi.edu/nsnam/ns/

[2] Nino Kubinidze, Ivan Ganchev and Máirtin O'Droma: "Network Simulator NS2: Shortcomings, Potential Development and Enhancement Strategies", Springer US, 2006, p.p. 263-277.

[3] http://www.nsnam.org/

[4] Jianli Pan, Raj Jain "A Survey of Network Simulation Tools: Current Status and Future Developments", 2008, http://www1.cse.wustl.edu/~jain/cse567-08/ftp/simtools.pdf

[5] Εισαγωγή στις νέες τεχνολογίες επικοινωνιών, Ανδρέας Πομπόρτσης, εκδόσεις Α. Τζιόλα Ε

[6] Mauro Campanella, "Premium IP service", presentation at the APM meeting, Paris, 21 September 2001

[7] V. Jacobson, K. Nicholsm K.Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999

[8] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, 1999

[9] D.Clark and W.Feng. "Explicit allocation of the best effort packet delivery service". IEEE/ACM Transactions on Networking, 6(4):362-374, 1998

[10] Δίκτυα Υπολογιστών, Τέταρτη Αμερικανική Έκδοση, Andrew Tanenbaum

[11] K.Ramakrishman and S.Floyd, "A proposal to Add Explicit Congestion Notification (ECN) to IP", RFC 2481, January 1999

[12] B.Braden, D.Clark, J.Crowfort, B.Davie, S.Deering, D.Estrin, S.Floyd, V.Jacobson, G.Minshall, C.Partridge, L.Peterson, K.Ramakrishman, S.Shenker, J.Wroclawski and L.Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, January 1998

[13] M.Shreedhar and G.Varghese, "Efficient Fair Queueing Using Deficit Round Robin", Proceedings of ACM SIGCOMM 95, October 1995

[14] E.Hahne and R.Gallager, "Round Robin Scheduling for Fair Flow Control in Data Communication Networks", IEEE International Conference on Communications, June 1986

[15] Cisco 12000 Series Internet Router: Frequently Asked Questions, http://www.cisco.com/warp/public/63/gsrfaq_11085.shtml

[16] Mauro Campanella, "Premium IP service", presentation at the APM meeting, Paris, 21 September 2001

[17] M. May, J-C. Bolot, A. Jean-Marie, C. Diot, “Simple performance models of tagging schemes for service differentiation in the internet”, Proc. Infocom ’99, New York, NY, March 1999

[18] TF.TANT Task Force, Tiziana Ferrari, Editor, “Differentiated Services: Experiment Report, Phase 2, May 15, 2000

[19] N.Seddigh, B.Nandy, J.Heinanen, “An Asured Rate Per-Domain Behaviour for Differentiated services”, Internet Draft (draft-ietf-diffserv-pdb-ar-00.txt), 2000

[20] B. Nandy, N. Seddigh, P. Pieda, “DiffServ’s Assured Forwarding PHB What Assurance does the costumer Have ? ,” The 9th International Workshop on Network and Operating Systems support for Digital Audio and Video (NOSSDAV’99), New Jersey, June 1999

[21] http://www.isi.edu/nsnam/ns/

[22] Nino Kubinidze, Ivan Ganchev and Máirtin O’Droma: “Network Simulator NS2: Shortcomings, Potential Development and Enhancement Strategies”, Springer US, 2006, p.p. 263-277.

[23] http://www.gnu.org/copyleft/gpl.html

[24] http://mercurial.selenic.com/

[25] http://code.google.com/p/waf/

[26] http://www.cygwin.com/

[27] http://www.nsnam.org/wiki/index.php/Installation

[28] http://code.nsnam.org/

[29] http://www.nsnam.org/releases/

[30] http://code.nsnam.org/ns-3-dev/

[31] http://ru6.cti.gr/ru6/ns3.tar/