Διαχείρηση Ουρών (Έλεγχος Κίνησης) χρησιμοποιώντας το OMNeT++

Κωνσταντίνος Λαβδαριάς
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων
Άρτα, Ελλάδα
int01971@uoi.gr

Δημήτριος Κατόπης
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων
Άρτα, Ελλάδα
int02124@uoi.gr

Σπυρίδων Λούντζης
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων
Άρτα, Ελλάδα
int02234@uoi.gr

Δημήτριος Ζήκος
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων
Άρτα, Ελλάδα
int02289@uoi.gr

ΠΕΡΙΛΗΨΗ

Η ανάγκη για μηχανισμούς παρακολούθησης και ρύθμισης της κίνησης σε ένα δίκτυο είναι ζωτικής σημασίας, καθώς η ροή της κίνησης επηρεάζει τη χρήση πόρων και τη συνολική αποτελεσματικότητα του δικτύου. Ο έλεγχος κίνησης επικεντρώνεται στην παροχή ποιότητας υπηρεσίας, τον έλεγχο απόδοσης σύνδεσης και την αντιμετώπιση της συμφόρησης. Η συμφόρηση εμφανίζεται όταν η χωρητικότητα του δικτύου αδυνατεί να ανταποκριθεί στις απαιτήσεις της κίνησης, προκαλώντας προβλήματα στην ποιότητα υπηρεσίας. Ο έλεγχος κίνησης αναζητά την αποδοτική χρήση των πόρων και την αποφυγή συμφόρησης. Διάφοροι παράγοντες, όπως η ικανότητα εξασφάλισης ποιότητας μετάδοσης και η παρακολούθηση της εισερχόμενης κίνησης, παίζουν σημαντικό ρόλο στη βελτιστοποίηση της απόδοσης του δικτύου. Ο έλεγχος κίνησης επικεντρώνεται στις μεθόδους Leaky Bucket και Token Bucket, οι οποίες προσφέρουν αποδοτικούς μηχανισμούς για τον έλεγχο της κίνησης και την πληροφορία ποιότητας υπηρεσίας. Η αποτελεσματική διαχείριση της κίνησης συνιστά κρίσιμη πτυχή για την εξασφάλιση υψηλής απόδοσης και αποφυγής προβλημάτων συμφόρησης στο δίκτυο.

Λέξεις κλειδιά: ***Έλεγχος κίνησης, συμφόρηση, διαχείριση ουρών, token bucket, leaky bucket.***

I. ΕΙΣΑΓΩΓΗ

Σε κάθε δίκτυο, είναι απαραίτητη η ύπαρξη μηχανισμών που να μπορούν να παρακολουθούν και να ρυθμίζουν την κίνησή του. Η ροή κίνησης επηρεάζει άμεσα τις απαιτήσεις σε πόρους του δικτύου (όπως αριθμός και χαρακτηριστικά των προσωρινών καταχωρητών, χρησιμοποιούμενα εύρη ζώνης, υπολογιστική ισχύς των ενεργών διατάξεων κλπ.), καθώς και τη συνολική διεκπεραιωτική ικανότητα και αποτελεσματικότητα του δικτύου. Έτσι, ο έλεγχος κίνησης είναι αναγκαίος τόσο για την προστασία της ποιότητας υπηρεσιών, που οι χρήστες λαμβάνουν από το δίκτυο, όσο και για την επίτευξη αποτελεσματικής διαχείρισης των πόρων του δικτύου. Ο έλεγχος κίνησης εστιάζεται στην παροχή ποιότητας υπηρεσίας, στον έλεγχο απόδοσης σύνδεσης και στον έλεγχο συμφόρησης. Ο συναγωνισμός, όμως, για πόρους του δικτύου μπορεί να οδηγήσει σε καταστάσεις συμφόρησης. Γενικά, η συμφόρηση δημιουργείται όταν η χωρητικότητα του δικτύου αδυνατεί να υποστηρίξει τις απαιτήσεις της προσφερόμενης κίνησης, δηλαδή το προσφερόμενο στο δίκτυο φορτίο πλησιάζει ή και υπερβαίνει τα όριά του. Τα όρια αυτά έχουν σχεδιασθεί για την εγγυημένη παροχή συγκεκριμένης ποιότητας υπηρεσιών, σύμφωνα με το συμβόλαιο κίνησης.

Η αποδοτικότητα ενός δικτύου δεν καθορίζεται μόνο από την τεχνολογία που διαθέτει αλλά και από το πως διαχειρίζεται τους δικτυακούς πόρους του με σκοπό την αποδοτικότερη λειτουργία του δικτύου. Σημαντικές παράμετροι της απόδοσης είναι η ικανότητα του να εξασφαλίζει την ασφαλή, μέσα σε προκαθορισμένο χρόνο, ποιότητα μετάδοσης των δεδομένων, αποφεύγοντας τη δημιουργία συμφόρησης στο δίκτυο. Όταν παρατηρείται συμφόρηση στο δίκτυο αυτή επηρεάζει όλους τους χρήστες που χρησιμοποιούν αυτό το τμήμα του δικτύου.

Το πρόβλημα της αποφυγής της συμφόρησης αποτελεί ένα σημαντικό θέμα έρευνας αφού η εμφάνιση του καθορίζεται από πολλούς και μη ντετερμινιστικούς παράγοντες, εφόσον εξαρτάται από την συμπεριφορά των επιμέρους χρηστών.

Δύο διαδεδομένες μέθοδοι για τον έλεγχο της εισερχόμενης κίνησης στο δίκτυο και τη συμμόρφωσή της σύμφωνα με το συμβόλαιο κίνησης είναι τα μοντέλα Leaky Bucket και Token Bucket. [1]

II. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Ένας απλός αλγόριθμος για τον έλεγχο της κίνησης, που διαχωρίζει τα πακέτα σε 2 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. Η λειτουργία τους βασίζεται στην ίδια λογική αλλά επιτυγχάνουν διαφορετικά αποτελέσματα.[2]

A. Token Bucket

Ο αλγόριθμος Token Bucket καθορίζει 2 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 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.

B. *Leaky Bucket*

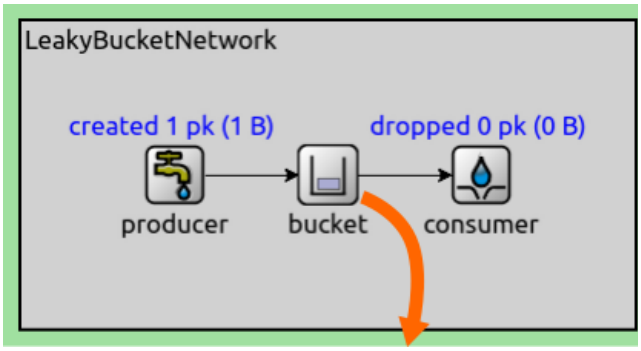
Παρόμοιος σε φιλοσοφία είναι και ο αλγόριθμος leaky bucket που καθορίζει αυστηρά το ρυθμό εξόδων των πακέτων από τον κάδο και είσοδό τους στο δίκτυο. Αν ο ρυθμός με τον οποίο καταφθάνουν τα πακέτα στον κάδο είναι μεγαλύτερα από τον ρυθμό με τον οποίο αυτά εξέρχονται από αυτόν, τότε συσσωρεύονται, μέχρι όμως και μια τιμή που αποτελεί και το μέγιστο μέγεθος του κάδου. Ουσιαστικά λοιπόν ο αλγόριθμος αυτός οδηγεί τα πακέτα να εξέρχονται με σταθερό ρυθμό από τον κάδο εφόσον βέβαια υπάρχει διαθέσιμος χώρος. Τα πακέτα αυτά θεωρούνται ως νόμιμα (εντός προφίλ), ενώ αντίθετα όσα δεν εισέρχονται στον κάδο μαρκάρονται ως εκτός προφίλ και χειρίζονται με ότι προβλέπει η συμφωνία που έχει υπογραφεί. Συμπερασματικά ο αλγόριθμος leaky bucket αποτελεί έναν πολύ καλό αλγόριθμο μορφοποίησης της κίνησης αφού σταθεροποιεί το ρυθμό μετάδοσης των πακέτων εξαλείφοντας εκρήξεις.

III. ΜΕΘΟΔΟΛΟΓΙΑ

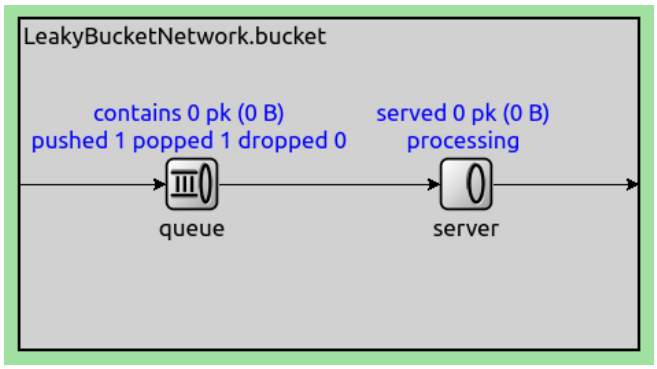
A. *Υλοποίηση του αλγορίθμου Leaky Bucket σε OMNeT++*

Ο αλγόριθμος για την υλοποίηση του Leaky Bucket περιλαμβάνει μια ουρά DropTailQueue και έναν εξυπηρετητή πακέτων (PacketServer). Η χωρητικότητα της ουράς και ο χρόνος επεξεργασίας του server μπορούν να χρησιμοποιηθούν για να παραμετροποιηθεί ο αλγόριθμος.

Στην Εικόνα 3, παρέχεται ένα παράδειγμα δικτύου, όπου τα πακέτα παράγονται από μια ενεργή πηγή πακέτων (ActivePacketSource). Η πηγή πακέτων προωθεί πακέτα στο ένα ενθυλακωτικό σύνολο LeakyBucket (Εικόνα 4), το οποίο τα προωθεί σε ένα παθητικό δέκτη πακέτων (PassivePacketSink). Ο αλγόριθμος ρυθμίζεται με ένα χρόνο επεξεργασίας 1 δευτερολέπτου και μια χωρητικότητα πακέτου 1. [3]



Εικόνα 3. Δίκτυο Leaky Bucket

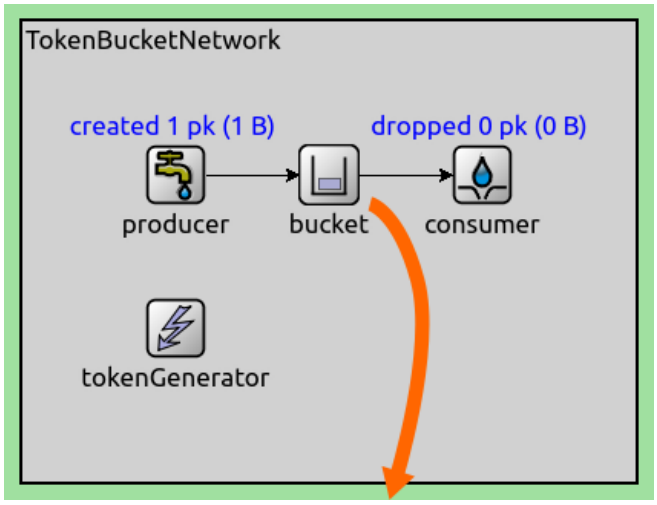


Εικόνα 4. Περιεχόμενο κουβά

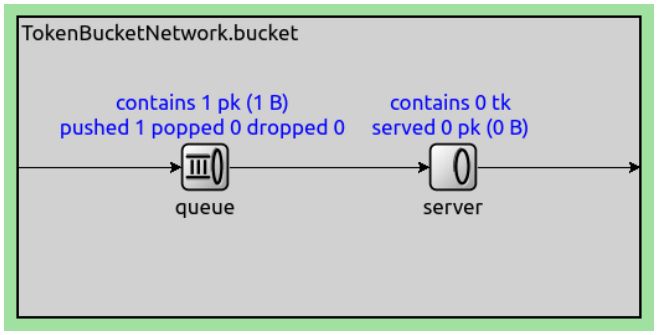
B. *Υλοποίηση του αλγορίθμου Token Bucket σε OMNeT++*

Ο αλγόριθμος για την υλοποίηση του Token Bucket περιλαμβάνει μια ουρά DropTailQueue και έναν server βασισμένο σε tokens (TokenBasedServer). Η χωρητικότητα της ουράς και η κατανάλωση των tokens του server μπορούν να χρησιμοποιηθούν για την παραμετροποίηση του αλγορίθμου του token bucket.

Στο παράδειγμα της Εικόνας 6, τα πακέτα παράγονται περιοδικά από μια ενεργή πηγή πακέτων (ActivePacketSource). Τα πακέτα προωθούνται TokenBucket, το οποίο τα προωθεί σε ένα παθητικό δέκτη πακέτων (PassivePacketSink). Η γεννήτρια των tokens (TimeBasedTokenGenerator) παράγει περιοδικά tokens και τα στέλνει στον κουβά (Εικόνα 7). [4]



Εικόνα 6. Δίκτυο Token Bucket

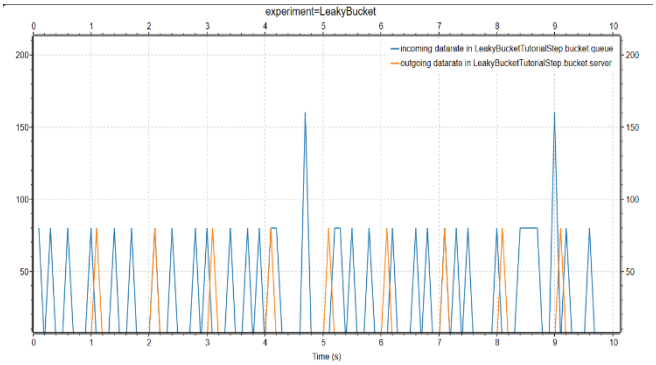


Εικόνα 7. Περιεχόμενο κουβά

Γ. Δοκιμές και Αποτελέσματα

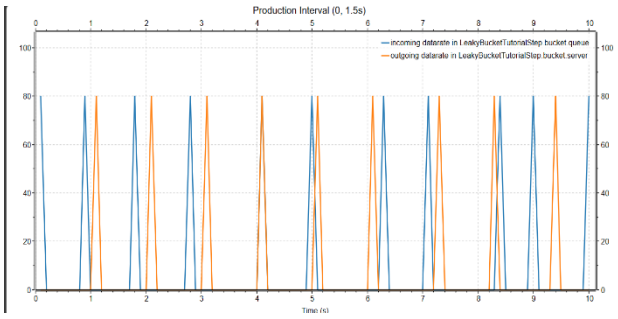
Στην περίπτωση του Leaky Bucket μηχανισμού, δημιουργούμε στο περιβάλλον του Omnet++ μια προσομοίωση διάρκειας 10 δευτερολέπτων. Για την κατανόηση της λειτουργίας και τις διάφορες μετρήσεις και δοκιμές που θέλουμε να κάνουμε, επεξεργαζόμαστε κάποιες από τις παραμέτρους του κώδικα στο αρχείο omnetpp.ini. Στο αρχείο .ini περιλαμβάνεται κώδικας ο οποίος είναι υπεύθυνος για τη λειτουργία του δικτύου.

Αρχικά πραγματοποιούμε δοκιμή με τον υπάρχων κώδικα που διαθέτει το Omnet. Καθοριστική σημασία για το μηχανισμό LB έχουν οι παράμετροι producer.productionInterval(διάστημα παραγωγής πακέτων της πηγής), bucket.server.processingTime (χρονική λειτουργία του κάδου) και bucket.queue.packetCapacity (χωρητικότητα του κάδου). Έτσι, σε 10 δευτερόλεπτα παρήχθησαν 36 πακέτα από τα οποία τα 10 πέρασαν από τον κάδο, δηλαδή τόσα δευτερόλεπτα όσα είναι και αυτά της προσομοίωσης αφού ο κάδος έχει χρόνο λειτουργίας κάθε ένα δευτερόλεπτο. Τα υπόλοιπα από τα 36 πακέτα χαθήκανε και δεν έμειναν ούτε στον κάδο αφού η χωρητικότητά του είναι ίση με ένα πακέτο. Σε δοκιμή που έγινε αλλάζοντας την χωρητικότητά του σε 100 πακέτα προφανώς δεν απορρίφθηκε κανένα πακέτο και θα σταλθούν και αυτά με τη σειρά τους όσο λειτουργεί το δίκτυο.



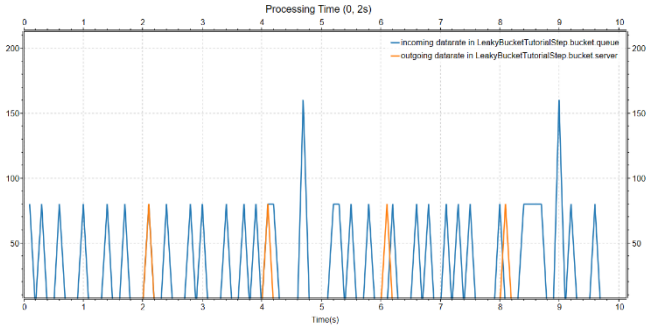
Εικόνα 8. Το διάγραμμα μας δείχνει την κίνηση των πακέτων στον κάδο. Με μπλε χρώμα είναι τα πακέτα που λαμβάνει και με πορτοκαλί τα πακέτα που εξάγει. Κάθε ένα δευτερόλεπτο προωθείται και από ένα πακέτο.

Αυξάνοντας το διάστημα παραγωγής του παραγωγού και έχοντας ίδιες συνθήκες με την προηγούμενη δοκιμή , παρατηρούμε ότι θα παραχθούν λιγότερα πακέτα αλλά η λειτουργία του κάδου είναι η ίδια.



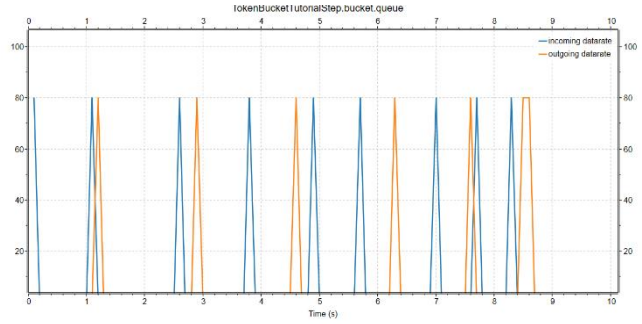
Εικόνα 9. Αύξηση του διαστήματος παραγωγής.

Προφανώς , αυξάνοντας το χρόνο λειτουργίας του κάδου θα εξαχθούν λιγότερα πακέτα στον ίδιο χρόνο λειτουργίας με την αρχική δοκιμή. Από τα 36 πακέτα εξάγονται τελικά τα 5.



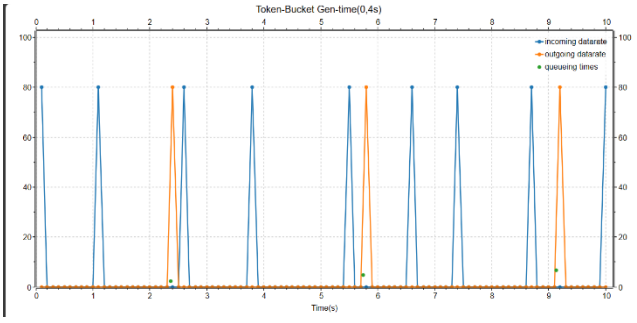
Εικόνα 10.Αυξημένος χρόνος λειτουργίας του κάδου.

Με την ίδια ακριβώς διαδικασία δοκιμάζουμε και τον μηχανισμό token bucket. Εδώ σημασία έχουν οι παράμετροι producer.productionInterval , bucket.server.maxNumTokens (μέγιστος αριθμός των token του κάδου) και το tokenGenerator.generationInterval (διάστημα παραγωγής token). Σε χρόνο 10 δευτερολέπτων παράγονται 9 πακέτα και εξάγονται 7.



Εικόνα 11.Λειτουργία token bucket ανά 2 δευτερόλεπτα.

Αν μεγαλώσουμε το χρόνο παραγωγής των token θα σταλθούν λιγότερα πακέτα και το αντίθετο.



Εικόνα 12.Λειτουργία token bucket ανά 4 δευτερόλεπτα.

Στον μηχανισμό token bucket δημιουργούνται tokens με ρυθμό r σε ένα κάδο με χωρητικότητα b. Για να σταλούν πακέτα στο δίκτυο θα πρέπει να υπάρχουν token στον κάδο.

Η διαφορά ανάμεσα στον κάδο leaky και τον κάδο token μπορεί να διατυπωθεί ως εξής: το token bucket επιτρέπει την

κυκλοφορία να είναι εκρηκτική αλλά την περιορίζει με ένα ρυθμό r. Από την άλλη ο μηχανισμός leaky bucket επιτρέπει την ομαλή κυκλοφορία των πακέτων.

Στον παρακάτω πίνακα παρατηρούμε ορισμένες διαφορές μεταξύ Leaky και Token Bucket

Παράμετροι	***Leaky Bucket***	***Token Bucket***
Εξάρτηση από τα token	*Ανεξάρτητο*	*Βασισμένο σε token*
Γεμάτος κουβάς με token	*Όταν ο κάδος είναι γεμάτος τα πακέτα απορρίπτονται*	*Όταν ο κάδος είναι γεμάτος τα token απορρίπτονται και όχι τα πακέτα*
Μεταφορά πακέτων	*Στέλνει πακέτο σε σταθερό ρυθμό*	*Μπορεί να στείλει μεγάλη ριπή πακέτων με ταχύτερο ρυθμό*
Προϋπόθεση για τη μετάδοση πακέτων	*Τα πακέτα μεταδίδονται συνέχεια*	*Τα πακέτα μεταδίδονται μόνο όταν υπάρχουν διαθέσιμα token*
Αποθήκευση token	*Δεν αποθηκεύει κανένα token*	*Αποθηκεύει token για εκρηκτική μετάδοση πακέτων*
Περιορισμένος αλγόριθμος	*Είναι πιο περιορισμένος σε σχέση με τον token bucket αλγόριθμο*	*Είναι λιγότερο περιορισμένος*

IV. ΣΥΖΗΤΗΣΗ

Μπορούμε να συνδυάσουμε τον κάδο leaky και τον κάδο token. Ιδανική συνθήκη είναι να λειτουργεί πρώτα ο κάδος token και μετά ο leaky για την ρύθμιση της κυκλοφορίας. Ο ρυθμός μετάδοσης του leaky bucket θα πρέπει να είναι μεγαλύτερος από το ρυθμό των token που παράγονται στον κάδο. Με τον συνδυασμό αυτών των μηχανισμών μπορούμε να αποφύγουμε την απώλεια πακέτων και πληροφορίας και δίνουμε στο δίκτυο τον επιθυμητό ρυθμό μετάδοσης.

[1] Μ. Δήμου, « Πολιτικές Κοστολόγησης και Τεχνικές Διαπραγμάτευσης στην Διαχείριση Τερματικών σε Περιβάλλοντα Adaptive Δικτύων Υψηλών Ταχυτήτων », Πανεπιστήμιο Πειραιά.

[2] Λ. Πουλόπουλος, « Μελέτη Παροχής Υπηρεσιών σε Ενοποιημένα L2 και MPLS Δίκτυα », Πανεπιστήμιο Πατρών.

[3] https://inet.omnetpp.org/docs/tutorials/queueing/doc/LeakyBucket.html

[4] https://inet.omnetpp.org/docs/tutorials/queueing/doc/TokenBucket.html