ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΡΥΘΜΟΥ ΑΠΩΛΕΙΑΣ ΠΑΚΕΤΩΝ ΕΛΑΣΤΙΚΗΣ ΚΙΝΗΣΗΣ ΣΕ ΔΙΚΤΥΟ IP ΜΕΣΩ ΑΝΑΔΡΟΜΙΚΟΥ ΤΥΠΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΨΑΡΡΑ ΔΗΜΗΤΡΙΟΥ του ΙΩΑΝΝΗ

ΦΟΙΤΗΤΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΠΡΟΗΓΜΕΝΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΔΙΚΤΥΑ»

Επιβλέπων Καθηγητής: Μοσχολιός Ιωάννης

Σεπτέμβριος 2011

Πίνακας Περιεχομένων

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ, ΠΙΝΑΚΩΝ & ΕΙΚΟΝΩΝ

ΣΧΗΜΑΤΑ

ΠΙΝΑΚΕΣ

ΕΙΚΟΝΕΣ

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας εργασίας, είναι η ανάλυση της πολιτικής διάθεσης εύρους ζώνης balanced fairness και της πιθανότητας απώλειας πακέτων σε δίκτυα IP τεχνολογίας. Παραθέτονται οι σχέσεις υπολογισμού και για τις δύο περιπτώσεις και αναπτύσσονται οι αναδρομικοί τύποι. Τέλος, γίνεται ο μαθηματικός υπολογισμός της διεκπεραιωτικής ικανότητας μιας ροής (flow throughput) μέσω αναδρομικού τύπου ο οποίος έχει δημιουργηθεί σε γλώσσα υψηλού επιπέδου C++.

Η θεωρία της τηλεπικοινωνιακής κίνησης που βασίζεται στον τύπο του Erlang, δεν μπορεί να εφαρμοστεί στα δίκτυα IP, των οποίων οι πόροι μοιράζονται δυναμικά στις ελαστικές ροές κίνησης. Ωστόσο στην βιβλιογραφία έχουν προταθεί αναδρομικοί μαθηματικοί τύποι μέσω των οποίων μπορούν να υπολογιστούν οι βασικές παράμετροι απόδοσης ενός δικτύου (π.χ. διεκπεραιωτική ικανότητα ροής, απώλεια πακέτων).

Σε αυτή την εργασία, εστιάζουμε σε μια γραμμή, η οποία εξυπηρετεί ροές διαφορετικών κατηγοριών ελαστικής κίνησης. Η διάθεση του εύρους ζώνης της γραμμής γίνεται σύμφωνα με την πολιτική balanced fairness. Κάθε ροή έχει μια μέγιστη απαίτηση σε εύρος ζώνης ίση με την χωρητικότητα της γραμμής, ενώ η άφιξη των ροών ακολουθεί μια διαδικασία Poisson. Στο πρώτο κεφάλαιο της εργασίας παρουσιάζεται αρχικά η πολιτική balanced fairness και εν συνεχεία ένας αναδρομικός τύπος υπολογισμού της διεκπεραιωτικής ικανότητας κάθε ροής. Ο αναδρομικός τύπος βασίζεται στον γνωστό αναδρομικό τύπο των Kaufman-Roberts, ο οποίος έχει χρησιμοποιηθεί για τον υπολογισμό της πιθανότητας απώλειας κλήσεων σε δίκτυα circuit-switched. Στο δεύτερο κεφάλαιο παρουσιάζεται ένας αναδρομικός τύπος για τον προσεγγιστικό υπολογισμό της πιθανότητας απώλειας πακέτων, βασισμένος επίσης στον τύπο των Kaufman-Roberts.

ΕΙΣΑΓΩΓΗ

Στα πλαίσια της εκπόνησης αυτής της Διπλωματικής εργασίας κατά το 3^{ο} εξάμηνο του ΠΜΣ «Προηγμένα Τηλεπικοινωνιακά Συστήματα και Δίκτυα», της Σχολής Θετικών Επιστημών, του τμήματος Επιστημών και Τεχνολογίας των Τηλεπικοινωνιών του Πανεπιστημίου Πελοποννήσου, αποφάσισα να ασχοληθώ με τη μελέτη του ρυθμού απώλειας πακέτων ελαστικής κίνησης σε δίκτυα IP. Τα βασικά αίτια που με οδήγησαν στο να ακολουθήσω αυτή την κατεύθυνση επιλογής του θέματος, είναι ότι στο μάθημα του 2^{ου} εξαμήνου «Προχωρημένα Θέματα Δικτύων», οι τεχνολογίες που παρουσιάζονταν ήταν πιο ενδιαφέρουσες και σύγχρονες ως προς την υλοποίησή τους.

Ο σκελετός της εργασίας έχει ως ακολούθως:

Στο **κεφάλαιο 1**, αναφέρουμε και μελετούμε την πολιτική διάθεσης εύρους ζώνης balanced fairness, υπολογίζουμε την διεκπεραιωτική ικανότητα μιας ροής (flow throughput) και παρουσιάζουμε έναν αναδρομικό τύπο για τον υπολογισμό της. Στη συνέχεια του κεφαλαίου παραθέτονται παραδείγματα, ώστε να γίνει πιο κατανοητή η μαθηματική ανάλυση των τύπων που παρουσιάζονται.

Στο **κεφάλαιο 2**, υπολογίζουμε την πιθανότητα απώλειας πακέτων σε δίκτυα IP, καθώς και έναν αναδρομικό τύπο για τον υπολογισμό της.

Στο **κεφάλαιο 3**, παρουσιάζουμε τον κώδικα προγραμματισμού σε C++ που δημιουργήσαμε για τον υπολογισμό της διεκπεραιωτικής ικανότητας μιας ροής (flow throughput) μέσω του αναδρομικού της τύπου και την επεξήγησή του.

Στον επισυναπτόμενο οπτικό δίσκο (CD) περιλαμβάνονται, εκτός από την εργασία σε ηλεκτρονική μορφή, αυτούσιος ο κώδικας του προγράμματος σε αρχείο τύπου *.cpp*, καθώς και αρχείο τύπου *.txt* με όνομα *«main.txt»*, στο οποίο περιέχεται παράδειγμα εισαγωγής τιμών μεταβλητών από αρχείο.

Στο σημείο αυτό, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, τον Λέκτορα του Πανεπιστημίου Πελοποννήσου Δρα Ιωάννη Μοσχολιό, για την αμέριστη βοήθεια που μου προσέφερε για την υλοποίηση αυτής της διπλωματικής εργασίας. Με βοήθησε στο να ανέβω ένα επίπεδο γνώσεων ακόμα σε αυτό που λέμε Τηλεπικοινωνιακή Κίνηση και γενικά Τηλεπικοινωνίες. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, που όλα αυτά τα χρόνια με βοηθά ηθικά και όχι μόνο, ώστε να πετύχω τους στόχους και τα όνειρά μου.

ΚΕΦΑΛΑΙΟ 1

Η πολιτική διάθεσης εύρους ζώνης balanced fairness

1.1 Περιγραφή της πολιτικής balanced fairness

Θεωρούμε μια γραμμή χωρητικότητας *C* η οποία εξυπηρετεί ροές (flows) από *N* κατηγορίες κίνησης. Οι ροές της κατηγορίας *i* (*i*=1,…, *N*) έχουν μέγιστο και μέσο ρυθμό μετάδοσης c_i, σ_i αντίστοιχα, ενώ φθάνουν στην γραμμή ακολουθώντας μια διαδικασία Poisson με ρυθμό λ_i. Η ένταση της κίνησης (traffic intensity) της κατηγορίας *i*, a_i, ορίζεται από την σχέση:

$$a_i = \lambda_i \sigma_i \tag{1}$$

και εκφράζει, σε bit/s, τον μέσο όγκο κίνησης που προσφέρουν στο σύστημα οι ροές της κατηγορίας *i*.

Ορίζουμε επίσης το φορτίο κίνησης στην γραμμή λόγω της κατηγορίας *i*, p_i, ως:

$$p_i = \frac{a_i}{C} \tag{2}$$

Με βάση την σχέση (2), το συνολικό φορτίο κίνησης της γραμμής, *p*, δίνεται από την σχέση:

$$p = \sum_{i=1}^{N} p_i = \frac{1}{C} \sum_{i=1}^{N} a_i \tag{3}$$

Το σύστημα στην κατάσταση ισορροπίας περιγράφεται από το διάνυσμα $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ όπου x_i είναι ο αριθμός των ροών της κατηγορίας *i* που εξυπηρετούνται από την γραμμή χωρητικότητας *C*. Με όμοιο τρόπο μπορούμε να ορίσουμε το διάνυσμα των μέγιστων ρυθμών μετάδοσης $\boldsymbol{c} = (c_1, c_2, ..., c_N)$ και το διάνυσμα της έντασης κίνησης $\boldsymbol{a} = (a_1, a_2, ..., a_N)$. Τέλος ορίζουμε ως e_i το μοναδιαίο διάνυσμα με 1 στο στοιχείο *i* και 0 στα υπόλοιπα στοιχεία (π.χ. $e_3 = (0,0,1,0,...,0)$), καθώς και τα σύμβολα:

$$x! = \prod_{i=1}^{N} x_i ! \tag{4}$$

$$a^x = \prod_{i=1}^{N} a_i^{x_i} \tag{5}$$

$$\boldsymbol{x}.\boldsymbol{c} = \prod_{i=1}^{N} x_i c_i \tag{6}$$

τα οποία θα χρησιμοποιηθούν στην ανάλυση που ακολουθεί.

Η διάθεση του εύρους ζώνης της γραμμής στις ροές όλων των κατηγοριών κίνησης γίνεται βάσει της πολιτικής balanced fairness [1],[2]. Σύμφωνα με την πολιτική αυτή, αν το σύστημα βρίσκεται στην κατάσταση ισορροπίας $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ και $\phi_i(\boldsymbol{x})$ είναι ο συνολικός ρυθμός μετάδοσης όλων των ροών της κατηγορίας i τότε κάθε μία από τις x_i ροές που εξυπηρετούνται από το σύστημα στην κατάσταση $\boldsymbol{x}$ έχει ρυθμό μετάδοσης $\frac{\phi_i(\boldsymbol{x})}{x_i}$. Ως παράδειγμα θεωρούμε δύο κατηγορίες κίνησης που εξυπηρετούνται από μια γραμμή χωρητικότητας C και έστω $\boldsymbol{x} = (x_1, x_2)$ μία κατάσταση ισορροπίας του συστήματος. Στην κατάσταση αυτή, κάθε ροή της πρώτης κατηγορίας έχει ρυθμό μετάδοσης $\frac{\phi_1(\boldsymbol{x})}{x_1}$ ενώ κάθε ροή της δεύτερης κατηγορίας έχει ρυθμό μετάδοσης $\frac{\phi_2(\boldsymbol{x})}{x_2}$. Σε κάθε περίπτωση πρέπει να ισχύει $\phi_1(\boldsymbol{x}) + \phi_2(\boldsymbol{x}) \le C$.

Βασικό πρόβλημα στην πολιτική balanced fairness είναι η εύρεση εκείνων των $\phi_i(\boldsymbol{x})$ (i =1, 2,…,N) που εξασφαλίζουν την ύπαρξη τοπικής ισορροπίας (local balance) μεταξύ γειτονικών καταστάσεων της Μαρκοβιανής αλυσίδας. Ως γειτονικές καταστάσεις ορίζουμε εκείνες που συνδέονται μεταξύ τους μέσω της άφιξης ή της αναχώρησης (τερματισμού) μιας κλήσης. Η ύπαρξη τοπικής ισορροπίας είναι χρήσιμη για τον εύκολο υπολογισμό της πιθανότητας $P(\boldsymbol{x})$ (το σύστημα να βρίσκεται στην κατάσταση $\boldsymbol{x}$) τόσο σε συστήματα απωλειών όσο και σε συστήματα αναμονής [3]. Για την περίπτωση που εξετάζουμε έστω οι γειτονικές καταστάσεις $\boldsymbol{x} - e_i = (x_1, x_2, ..., x_i - 1, ..., x_N)$ και $\boldsymbol{x} = (x_1, x_2, ..., x_i, ..., x_N)$. Τότε η εξίσωση τοπικής ισορροπίας μεταξύ των δύο καταστάσεων είναι της μορφής [2]:

$$a_i P(\boldsymbol{x} - e_i) = \phi_i(\boldsymbol{x}) P(\boldsymbol{x}) \tag{7}$$

Η σχέση (7), λόγω της (1), γράφεται επίσης ως:

$$\lambda_i P(\boldsymbol{x} - e_i) = \frac{\phi_i(\boldsymbol{x})}{\sigma_i} P(\boldsymbol{x}) \tag{8}$$

Προκειμένου λοιπόν να υπάρχει τοπική ισορροπία μεταξύ των γειτονικών καταστάσεων $\boldsymbol{x} - e_i$ και $\boldsymbol{x}$ θα πρέπει ο όρος $\phi_i(\boldsymbol{x})$ να δίνεται από την σχέση:

$$\phi_i(\boldsymbol{x}) = \frac{a_i P(\boldsymbol{x} - e_i)}{P(\boldsymbol{x})} \tag{9}$$

Επιπρόσθετα, αν ορίσουμε την συνάρτηση $\Phi(\boldsymbol{x})$ ως:

$$\Phi(\boldsymbol{x}) = \frac{P(x)}{a_1^{x_1} a_2^{x_2} \ldots a_N^{x_N}} = \frac{P(x)}{\prod_{i=1}^{N} a_i^{x_i}} \tag{10}$$

τότε η (9) λόγω της (10) μπορεί να γραφεί ως:

$$\phi_i(\boldsymbol{x}) = \frac{\Phi(\boldsymbol{x} - e_i)}{\Phi(\boldsymbol{x})} \tag{11}$$

Συμπερασματικά, μπορούμε να επιλέξουμε αυθαίρετες τιμές για τα $P(\boldsymbol{x})$ ή μια αυθαίρετη συνάρτηση $\Phi(\boldsymbol{x})$ και να καταλήξουμε σε ένα σύστημα στο οποίο υπάρχει τοπική ισορροπία αρκεί οι ρυθμοί μετάδοσης $\phi_i(\boldsymbol{x})$ (i=1,…, N) να καθορίζονται από την (11).

Η συνάρτηση $\Phi(\boldsymbol{x})$ μπορεί να υπολογιστεί αναδρομικά μέσω της σχέσης [2]:

$$\Phi(\boldsymbol{x}) = \begin{cases} 0 & ,\alpha\nu\ x_i < 0\ \gamma\iota\alpha\ \kappa\acute{\alpha}\pi\omicron\iota o\ i \\ \dfrac{1}{x!c^x} & ,\alpha\nu\ x_i \geq 0\ \kappa\alpha\iota\ \boldsymbol{x}.\boldsymbol{c} \leq C \\ \dfrac{1}{C}\displaystyle\sum_{i=1}^{N} \Phi(\boldsymbol{x} - e_i) & ,\ \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \tag{12}$$

Σημείωση: Προκειμένου να δείξουμε την χρήση της σχέσης (12), έστω N=2 και i=1. Τότε μέσω της (11) έχουμε: $\phi_1(\boldsymbol{x}) = \phi_1(x_1, x_2) = \dfrac{\Phi(\boldsymbol{x} - e_1)}{\Phi(\boldsymbol{x})} = \dfrac{\Phi(x_1 - 1, x_2)}{\Phi(x_1, x_2)}$ και λόγω της (12) προκύπτει τελικά ότι $\phi_1(\boldsymbol{x}) = \phi_1(x_1, x_2) = \dfrac{\Phi(\boldsymbol{x} - e_1)}{\Phi(\boldsymbol{x})} = \dfrac{\dfrac{1}{(x_1 - 1)! x_2 ! c_1^{x_1 - 1} c_2^{x_2}}}{\dfrac{1}{x_1 ! x_2 ! c_1^{x_1} c_2^{x_2}}} = x_1 c_1$.

Πράγματι, όταν $\boldsymbol{x}.\boldsymbol{c} \leq C$ τότε ο μέγιστος ρυθμός μετάδοσης μιας ροής της πρώτης κατηγορίας δίνεται από την σχέση $\dfrac{\phi_1(\boldsymbol{x})}{x_1} = c_1$.

Μέσω της σχέσης (11) μπορεί να δειχθεί ότι σ' ένα σύστημα στο οποίο εφαρμόζεται η πολιτική balanced fairness ισχύει η σχέση (balance property) [4]:

$$\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x} - e_i) = \phi_j(\boldsymbol{x})\phi_i(\boldsymbol{x} - e_j) \tag{13}$$

αφού και οι δύο όροι της εξίσωσης είναι ίσοι με $\Phi(\boldsymbol{x} - e_i - e_j)/\Phi(\boldsymbol{x})$.

Η σχέση (13) γράφεται και ως:

$$\frac{\phi_j(\boldsymbol{x}\text{-}e_i)}{\phi_j(\boldsymbol{x})}=\frac{\phi_i(\boldsymbol{x}\text{-}e_j)}{\phi_i(\boldsymbol{x})} \tag{14}$$

δηλαδή, η σχετική μεταβολή του ρυθμού μετάδοσης που διανέμεται στην κατηγορία *j* λόγω τερματισμού μιας ροής *i* (αριστερός όρος της (14)) ισούται με την σχετική μεταβολή του ρυθμού μετάδοσης που διανέμεται στην κατηγορία *i* λόγω τερματισμού μιας ροής *j* (δεξιός όρος της (14)).

Σημείωση: Προκειμένου να επαληθεύσουμε την σχέση (14), έστω $N = 2$ και $\boldsymbol{x}=(x_1,x_2)$. Τότε:

$\phi_1(\boldsymbol{x})=\frac{\Phi(x_1-1,x_2)}{\Phi(x_1,x_2)}$ και $\phi_2(\boldsymbol{x})=\frac{\Phi(x_1,x_2-1)}{\Phi(x_1,x_2)}$. Επίσης $\phi_1(x_1,x_2-1)=\frac{\Phi(x_1-1,x_2-1)}{\Phi(x_1,x_2-1)}$ και $\phi_1(x_1-1,x_2)=\frac{\Phi(x_1-1,x_2-1)}{\Phi(x_1-1,x_2)}$. Από τις δύο τελευταίες σχέσεις έχουμε $\phi_1(x_1,x_2-1)\Phi(x_1,x_2-1)=\phi_2(x_1-1,x_2)\Phi(x_1-1,x_2)$ και μέσω των σχέσεων που δίνουν τα $\phi_1(\boldsymbol{x}),\phi_2(\boldsymbol{x})$ προκύπτει τελικά ότι: $\phi_1(x_1,x_2-1)/\phi_1(x)=\phi_2(x_1-1,x_2)/\phi_2(x)$ που είναι η σχέση (14) για δύο κατηγορίες κίνησης.

Η πιθανότητα $P(\boldsymbol{x})$ (το σύστημα να βρίσκεται στην κατάσταση $\boldsymbol{x}$, υπολογίζεται από την σχέση [2]:

$$P(\boldsymbol{x})=\frac{X(\boldsymbol{x})}{\sum_y X(\boldsymbol{y})} \tag{15}$$

όπου

$$X(\boldsymbol{x})=\Phi(\boldsymbol{x})\alpha^x \tag{16}$$

ενώ το άθροισμα στον παρονομαστή της (15) είναι πεπερασμένο όταν $p<1$ $\Rightarrow\sum_{i=1}^{N}p_i<1\Rightarrow\sum_{i=1}^{N}a_i<C$.

1.2 Υπολογισμός της διεκπεραιωτικής ικανότητας μιας ροής (flow throughput)

Στην παράγραφο αυτή παρουσιάζεται ο ορισμός της διεκπεραιωτικής ικανότητας μιας ροής καθώς και ένας μη αναδρομικός τύπος υπολογισμού αυτής.

Η διεκπεραιωτική ικανότητα μιας ροής *i*, γ_i, δίνεται από τον λόγο του μέσου ρυθμού μετάδοσης της ροής προς την μέση διάρκεια εξυπηρέτησης της ροής. Η μέση διάρκεια εξυπηρέτησης της ροής ισούται σύμφωνα με τον νόμο του Little με $E[X_i]/\lambda_i$. Επομένως:

$$\gamma_i = \frac{\sigma_i}{\frac{E[X_i]}{\lambda_i}} \tag{17}$$

Η σχέση (17), βάσει του ορισμού της μέσης τιμής και μέσω της σχέσης (15) γράφεται ως εξής:

$$\gamma_i = \frac{\lambda_i \sigma_i}{E[X_i]} = \frac{a_i}{E[X_i]} = \frac{a_i}{\sum_x x_i P(\boldsymbol{x})} = a_i \frac{\sum_x X(\boldsymbol{x})}{\sum_x x_i X(\boldsymbol{x})} \tag{18}$$

Το γεγονός ότι ο παραπάνω τύπος δεν είναι αναδρομικός, εμποδίζει την εφαρμογή του σε συστήματα με μεγάλη χωρητικότητα και πολλές κατηγορίες κίνησης. Προκειμένου να αντιμετωπιστεί το πρόβλημα αυτό, παρουσιάζεται στην επόμενη παράγραφο ένας αναδρομικός τύπος υπολογισμού της διεκπεραιωτικής ικανότητας.

1.3 Αναδρομικός τύπος υπολογισμού της διεκπεραιωτικής ικανότητας μιας ροής

Υποθέτουμε αρχικά, ότι η χωρητικότητα της γραμμής *C* και οι μέγιστοι ρυθμοί μετάδοσης $c_1, c_2, ..., c_N$ είναι ακέραιοι αριθμοί. Η υπόθεση αυτή δεν είναι δεσμευτική εφόσον το σύστημα που εξετάζουμε είναι ισοδύναμο με ένα σύστημα μοναδιαίας χωρητικότητας και μέγιστων ρυθμών μετάδοσης $c_1/C, c_2/C, ..., c_N/C$.

Για κάθε ακέραιο *n* με *n* = 1,…,*C*, ορίζουμε την πιθανότητα το σύστημα να βρίσκεται στην κατάσταση *n*, *p*(*n*), ως:

$$p(n) = \sum_{\boldsymbol{x}:\boldsymbol{x}.\boldsymbol{c}=n} X(\boldsymbol{x}) \tag{19}$$

Επίσης ορίζουμε την μέση τιμή του πληθυσμού των ροών της κατηγορίας *i* στην κατάσταση *n*, $q_i(n)$, ως:

$$q_i(n) = \sum_{\boldsymbol{x}:\boldsymbol{x}.\boldsymbol{c}=n} x_i X(\boldsymbol{x}) \tag{20}$$

Η σχέση (18) με την βοήθεια των (19), (20) γράφεται ως εξής:

$$\gamma_i = a_i \frac{\sum_{n\geq 0} p(n)}{\sum_{n\geq 0} q_i(n)} \tag{21}$$

Ο αναδρομικός υπολογισμός της διεκπεραιωτικής ικανότητας γ_i γίνεται σε δύο βήματα. Στο πρώτο βήμα υπολογίζουμε τις τιμές των $p(n)$ και $q_i(n)$ για n = 1,…,C σύμφωνα με τις σχέσεις [2]:

$$p(n) = \begin{cases} 1 & , \gamma\iota\alpha\ n = 0 \\ \sum_{j=1}^{N} \frac{a_j}{n} p(n - c_j) & , \gamma\iota\alpha\ n = 1,...,C \\ 0 & , \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \tag{22}$$

$$q_i(n) = \begin{cases} \frac{a_i}{n} p(n - c_i) + \sum_{j=1}^{N} \frac{a_j}{n} q_i(n - c_j) & , \gamma\iota\alpha\ n = 1,...,C \\ 0 & , \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \tag{23}$$

Στο δεύτερο βήμα (περίπτωση όπου $n > C$) έχουμε ότι:

$$\overline{p} = \sum_{i=1}^{N} \frac{p_i \overline{p}_i}{1 - p} \tag{24}$$

όπου

$$\overline{p}_i = \sum_{C - c_i < n \leq C} p(n) \tag{25}$$

$$\overline{q}_i = p_i \frac{\overline{p}_i + \overline{p}}{1 - p} + \sum_{j=1}^{N} \frac{p_j \overline{q}_{ij}}{1 - p} \tag{26}$$

και

$$\overline{q}_{ij} = \sum_{C - c_j < n \leq C} q_i(n) \tag{27}$$

Η απόδειξη των σχέσεων (23)-(26) παραλείπεται. Ο αναγνώστης μπορεί να ανατρέξει στην εργασία [2]. Η απόδειξη της σχέσης (22) (γνωστή στην βιβλιογραφία και ως Kaufman-Roberts formula) παρουσιάζεται στην εργασία [5].

1.4 Παραδείγματα

1ο παράδειγμα

Το πρώτο παράδειγμα είναι ιδιαίτερα μικρό και χρησιμοποιείται ως tutorial προκειμένου να δείξουμε αναλυτικά όλες τις επιμέρους πράξεις που απαιτούνται για τον υπολογισμό της διεκπεραιωτικής ικανότητας.

Θεωρούμε μια γραμμή χωρητικότητας $C = 3$, η οποία εξυπηρετεί $N = 2$ κατηγορίες κίνησης. Οι μέγιστοι ρυθμοί μετάδοσης των κατηγοριών κίνησης είναι οι εξής: $c_1=1$ και $c_2=2$. Αντίστοιχα η ένταση της κίνησης για κάθε κατηγορία κίνησης είναι $a_1=1$, $a_2=1$.

Στη συνέχεια παρουσιάζονται οι λύσεις και τα αποτελέσματα των μαθηματικών εξισώσεων για τα $p(n)$, καθώς και οι λύσεις και τα αποτελέσματα της διεκπεραιωτικής ικανότητας γ_i κάθε κατηγορίας κίνησης i ως συνάρτηση του φορτίου κίνησης της γραμμής p.

Άρα έχουμε:

$$p(n) = \sum_{j=1}^{N} \frac{a_j}{n} p(n - c_j)$$

$$p(1) = \frac{a_1}{1} p(1-c_1) + \frac{a_2}{1} p(1-c_2) = \frac{1}{1} p(1-1) + \frac{1}{1} p(1-2) =$$
$$= \frac{1}{1} p(0) + \frac{1}{1} p(-1) = \frac{1}{1} \cdot 1 + 0 = 1$$

$$p(2) = \frac{a_1}{2} p(2-c_1) + \frac{a_2}{2} p(2-c_2) = \frac{1}{2} p(2-1) + \frac{1}{2} p(2-2) =$$
$$= \frac{1}{2} p(1) + \frac{1}{2} p(0) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$$

$$p(3) = \frac{a_1}{3} p(3-c_1) + \frac{a_2}{3} p(3-c_2) = \frac{1}{3} p(3-1) + \frac{1}{3} p(3-2) =$$
$$= \frac{1}{3} p(2) + \frac{1}{3} p(1) = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{2}{3}$$

$$q_i(n) = \frac{a_i}{n}\, p(n-c_i) + \sum_{j=1}^{N} \frac{a_j}{n} q_i(n-c_j)$$

$$q_1(1) = \frac{a_1}{1}\, p(1-c_1) + \sum_{j=1}^{N} \frac{a_j}{1} q_1(1-c_j) = \frac{1}{1}\, p(1-1) + \frac{a_1}{1} q_1(1-c_1) + \frac{a_2}{1} q_1(1-c_2) =$$
$$= \frac{1}{1}\, p(0) + \frac{1}{1} q_1(1-1) + \frac{1}{1} q_1(1-2) = \frac{1}{1}\cdot 1 + \frac{1}{1} q_1(0) + \frac{1}{1} q_1(-1) =$$
$$= \frac{1}{1} + 0 + 0 = 1$$

$$q_1(2) = \frac{a_1}{2}\, p(2-c_1) + \sum_{j=1}^{N} \frac{a_j}{2} q_1(2-c_j) = \frac{1}{2}\, p(2-1) + \frac{a_1}{2} q_1(2-c_1) + \frac{a_2}{2} q_1(2-c_2) =$$
$$= \frac{1}{2}\, p(1) + \frac{1}{2} q_1(2-1) + \frac{1}{2} q_1(2-2) = \frac{1}{2}\cdot 1 + \frac{1}{2} q_1(1) + \frac{1}{2} q_1(0) =$$
$$= \frac{1}{2} + \frac{1}{2}\cdot 1 + 0 = 1$$

$$q_1(3) = \frac{a_1}{3}\, p(3-c_1) + \sum_{j=1}^{N} \frac{a_j}{3} q_1(3-c_j) = \frac{1}{3}\, p(3-1) + \frac{a_1}{3} q_1(3-c_1) + \frac{a_2}{3} q_1(3-c_2) =$$
$$= \frac{1}{3}\, p(2) + \frac{1}{3} q_1(3-1) + \frac{1}{3} q_1(3-2) = \frac{1}{3}\cdot 1 + \frac{1}{3} q_1(2) + \frac{1}{3} q_1(1) =$$
$$= \frac{1}{3} + \frac{1}{3}\cdot 1 + \frac{1}{3}\cdot 1 = 1$$

$$q_2(1) = \frac{a_2}{1}\, p(1-c_2) + \sum_{j=1}^{N} \frac{a_j}{1} q_2(1-c_j) = \frac{1}{1}\, p(1-2) + \frac{a_1}{1} q_2(1-c_1) + \frac{a_2}{1} q_2(1-c_2) =$$
$$= \frac{1}{1}\, p(-1) + \frac{1}{1} q_2(1-1) + \frac{1}{1} q_2(1-2) = 0 + \frac{1}{1} q_2(0) + \frac{1}{2} q_2(-1) =$$
$$= 0 + 0 + 0 = 0$$

$$q_2(2) = \frac{a_2}{2}\, p(2-c_2) + \sum_{j=1}^{N} \frac{a_j}{2} q_2(2-c_j) = \frac{1}{2}\, p(2-2) + \frac{a_1}{2} q_2(2-c_1) + \frac{a_2}{2} q_2(2-c_2) =$$
$$= \frac{1}{2}\, p(0) + \frac{1}{2} q_2(2-1) + \frac{1}{2} q_2(2-2) = \frac{1}{2}\cdot 1 + \frac{1}{2} q_2(1) + \frac{1}{2} q_2(0) =$$
$$= \frac{1}{2} + 0 + 0 = \frac{1}{2}$$

$$q_2(3)=\frac{a_2}{3}p(3-c_2)+\sum_{j=1}^{N}\frac{a_j}{3}q_2(3-c_j)=\frac{1}{3}p(3-2)+\frac{a_1}{3}q_2(3-c_1)+\frac{a_2}{3}q_2(3-c_2)=$$
$$=\frac{1}{3}p(1)+\frac{1}{3}q_2(3-1)+\frac{1}{3}q_2(3-2)=\frac{1}{3}\cdot 1+\frac{1}{3}q_2(2)+\frac{1}{3}q_2(1)=$$
$$=\frac{1}{3}+\frac{1}{3}\cdot\frac{1}{2}+0=\frac{1}{2}$$

Για να υπολογίσουμε το $\overline{p}=\sum_{i=1}^{N}\frac{p_i\overline{p}_i}{1-p}$, θα πρέπει πρώτα να υπολογίσουμε το p_i, το $\overline{p}_i$ και το p.

Άρα έχουμε:

$$p_i=\frac{a_i}{C}$$

$$p_1=\frac{a_1}{C}=\frac{1}{3}$$
$$p_2=\frac{a_2}{C}=\frac{1}{3}$$

$$\overline{p}_i=\sum_{n=C-c_i+1}^{C}p(n)$$

$$\overline{p}_1=\sum_{3-1+1}^{3}p(n)=p(3)=\frac{2}{3}$$

$$\overline{p}_2=\sum_{3-2+1}^{3}p(n)=p(2)+p(3)=1+\frac{2}{3}=\frac{5}{3}$$

$$p=\sum_{i=1}^{N}p_i$$

$$p=\sum_{1}^{2}p_i=p_1+p_2=\frac{1}{3}+\frac{1}{3}=\frac{2}{3}$$

άρα: $\overline{p} = \sum_{i=1}^{N} \frac{p_i \overline{p}_i}{1-p}$

$$\overline{p} = \sum_{i=1}^{2} \frac{p_i \overline{p}_i}{1-p} = \frac{p_1 \overline{p}_1}{1-p} + \frac{p_2 \overline{p}_2}{1-p} = \frac{\frac{1}{3}\cdot\frac{2}{3}}{1-\frac{2}{3}} + \frac{\frac{1}{3}\cdot\frac{5}{3}}{1-\frac{2}{3}} = \frac{6}{9} + \frac{15}{9} = \frac{7}{3}$$

$$\overline{q}_{ij} = \sum_{n=C-c_j+1}^{C} q_i(n)$$

$$\overline{q}_{11} = \sum_{3-1+1}^{3} q_1(n) = \sum_{3}^{3} q_1(n) = q_1(3) = 1$$

$$\overline{q}_{12} = \sum_{3-2+1}^{3} q_1(n) = \sum_{2}^{3} q_1(n) = q_1(2) + q_1(3) = 1+1 = 2$$

$$\overline{q}_{21} = \sum_{3-1+1}^{3} q_2(n) = \sum_{3}^{3} q_2(n) = q_2(3) = \frac{1}{2}$$

$$\overline{q}_{22} = \sum_{3-2+1}^{3} q_2(n) = \sum_{2}^{3} q_2(n) = q_2(2) + q_2(3) = \frac{1}{2} + \frac{1}{2} = 1$$

$$\overline{q}_i = p_i \frac{\overline{p}_i + \overline{p}}{1-p} + \sum_{j=1}^{N} \frac{p_j \overline{q}_{ij}}{1-p}$$

$$\overline{q}_1 = p_1 \frac{\overline{p}_1 + \overline{p}}{1-p} + \sum_{j=1}^{2} \frac{p_j \overline{q}_{1j}}{1-p} = p_1 \frac{\overline{p}_1 + \overline{p}}{1-p} + \frac{p_1 \overline{q}_{11}}{1-p} + \frac{p_2 \overline{q}_{12}}{1-p} =$$

$$= \frac{1}{3}\cdot\frac{\frac{2}{3}+\frac{7}{3}}{1-\frac{2}{3}} + \frac{\frac{1}{3}\cdot 1}{1-\frac{2}{3}} + \frac{\frac{1}{3}\cdot 2}{1-\frac{2}{3}} = \frac{1}{3}\cdot 9 + 1 + 2 = 6$$

$$\bar{q}_2 = p_2 \frac{\bar{p}_2 + \bar{p}}{1-p} + \sum_{j=1}^{2} \frac{p_j \bar{q}_{2j}}{1-p} = p_2 \frac{\bar{p}_2 + \bar{p}}{1-p} + \frac{p_1 \bar{q}_{21}}{1-p} + \frac{p_2 \bar{q}_{22}}{1-p} =$$

$$= \frac{1}{3} \cdot \frac{\frac{5}{3} + \frac{7}{3}}{1-\frac{2}{3}} + \frac{\frac{1}{3} \cdot \frac{1}{2}}{1-\frac{2}{3}} + \frac{\frac{1}{3} \cdot 1}{1-\frac{2}{3}} = \frac{1}{3} \cdot 12 + \frac{3}{6} + 1 = 5,5$$

$$\gamma_i = \alpha_i \frac{\sum_{n \geq 0} p(n)}{\sum_{n \geq 0} q_i(n)}$$

$$\gamma_1 = \alpha_1 \frac{p(0) + p(1) + p(2) + p(3) + \bar{p}}{q_1(0) + q_1(1) + q_1(2) + q_1(3) + \bar{q}_1} =$$

$$= 1 \frac{1+1+1+\frac{2}{3}+\frac{7}{3}}{0+1+1+1+6} = 0,6666$$

$$\gamma_2 = \alpha_2 \frac{p(0) + p(1) + p(2) + p(3) + \bar{p}}{q_2(0) + q_2(1) + q_2(2) + q_2(3) + \bar{q}_2} =$$

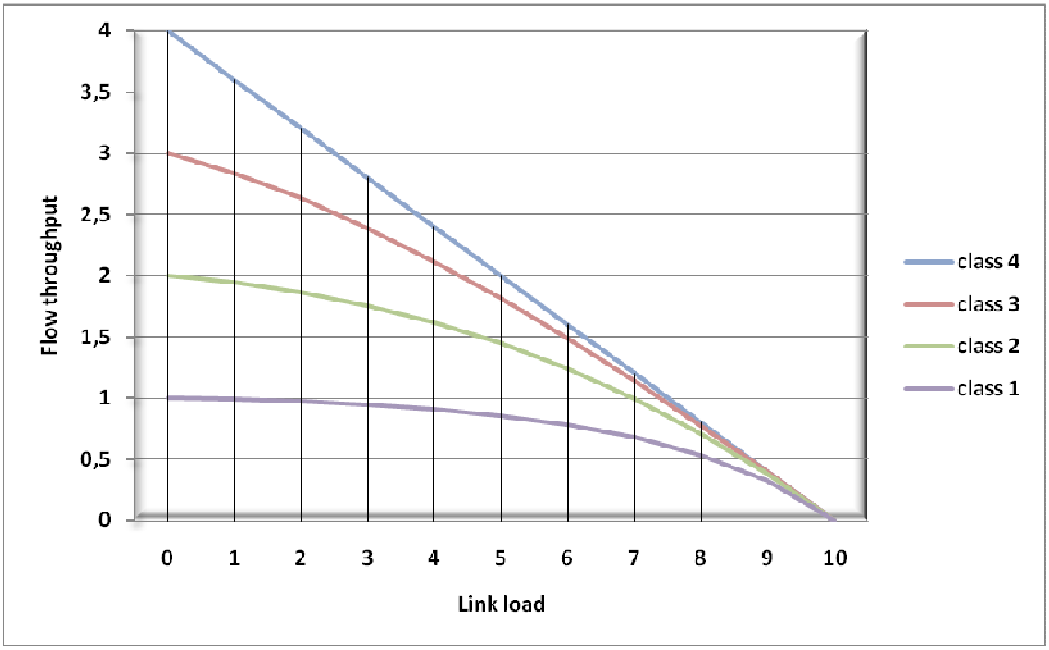
$$= 1 \frac{1+1+1+\frac{2}{3}+\frac{7}{3}}{0+0+\frac{1}{2}+\frac{1}{2}+5,5} = 0,9230$$

2ο παράδειγμα

Θεωρούμε μια γραμμή χωρητικότητας $C = 4$, η οποία εξυπηρετεί $N = 4$ κατηγορίες κίνησης. Οι μέγιστοι ρυθμοί μετάδοσης των κατηγοριών κίνησης είναι οι εξής: $c_1 = 1, c_2 = 2, ..., c_4 = 4$. Αντίστοιχα η ένταση της κίνησης για κάθε κατηγορία κίνησης δίνεται από την σχέση: $a_1 = a_2 = ... = a_4 = \frac{pC}{N}$.

Στο σχήμα 1 παρουσιάζεται η διεκπεραιωτική ικανότητα γ_i κάθε κατηγορίας κίνησης *i* ως συνάρτηση του φορτίου κίνησης της γραμμής *p*. Παρατηρούμε ότι για χαμηλές τιμές του φορτίου *p*, το σύστημα δεν εξυπηρετεί πολλές ροές, επομένως είναι

αναμενόμενο ο στιγμιαίος ρυθμός μετάδοσης κάθε ροής της κατηγορίας *i* να ισούται περίπου με c_i. Στην περίπτωση αυτή, η διεκπεραιωτική ικανότητα της κατηγορίας κίνησης *i* λαμβάνει τιμή κοντά στην τιμή c_i. Όσο το φορτίο *p* αυξάνεται τόσο το σύστημα πρέπει να εξυπηρετήσει πολλές ροές, με αποτέλεσμα ο στιγμιαίος ρυθμός μετάδοσης κάθε ροής να μειώνεται. Η μείωση αυτή παρουσιάζεται στο σχήμα 1, όπου παρατηρούμε ότι για πολύ υψηλό φορτίο *p*, η διεκπεραιωτική ικανότητα τείνει στο μηδέν. Στον πίνακα 1 παρουσιάζονται αναλυτικά τα αποτελέσματα της διεκπεραιωτικής ικανότητας της πρώτης και της τέταρτης κατηγορίας κίνησης.



Σχήμα 1: Διεκπεραιωτική ικανότητα συναρτήσει του φορτίου *p* (παράδειγμα 2)

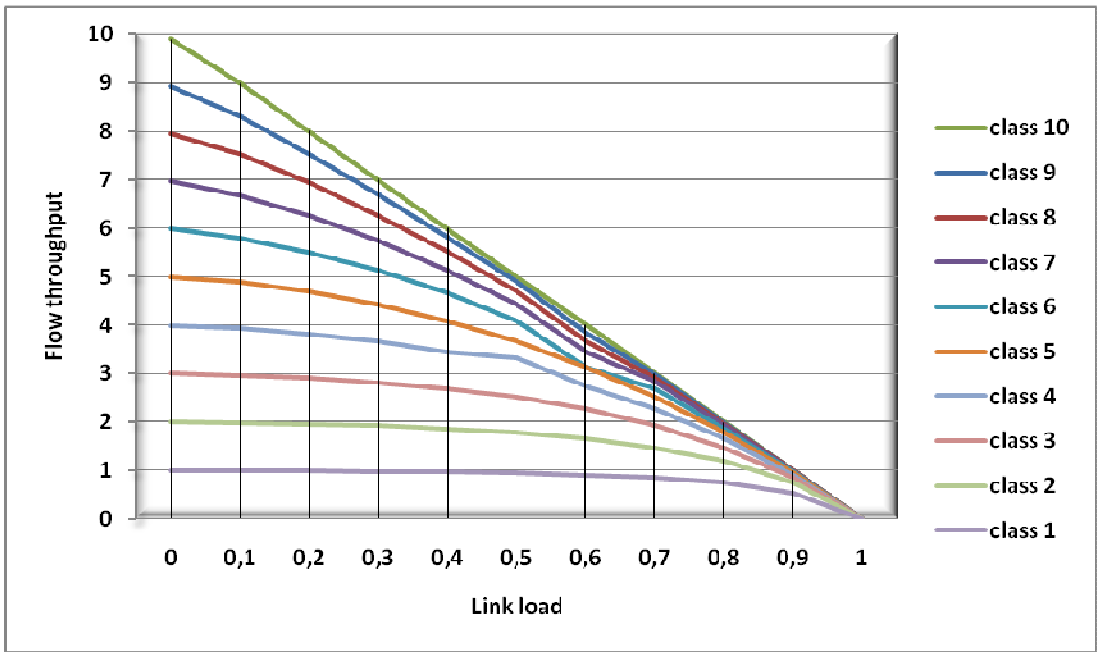
Link load	FlowThroughput (1η κατηγορία)	FlowThroughput (4η κατηγορία)
0	0,999	3,999
0,1	0,990	3,6
0,2	0,973	3,2
0,3	0,948	2,80
0,4	0,910	2,4
0,5	0,857	2
0,6	0,782	1,6
0,7	0,679	1,2
0,8	0,533	0,8
0,9	0,321	0,4
1	0	0

Πίνακας 1: Αναλυτικά αποτελέσματα του 2ου παραδείγματος για τις κατηγορίες 1 και 4

3ο παράδειγμα

Ακολουθούμε την λογική του δεύτερου παραδείγματος, εξετάζοντας ένα μεγαλύτερο παράδειγμα. Θεωρούμε λοιπόν μια γραμμή χωρητικότητας C = 10, η οποία εξυπηρετεί N = 10 κατηγορίες κίνησης. Οι μέγιστοι ρυθμοί μετάδοσης των κατηγοριών κίνησης είναι οι εξής: $c_1 = 1, c_2 = 2, ..., c_{10} = 10$. Αντίστοιχα η ένταση της κίνησης για κάθε κατηγορία κίνησης δίνεται από την σχέση: $a_1 = a_2 = ... = a_{10} = \frac{pC}{N}$.

Στο σχήμα 2 παρουσιάζεται η διεκπεραιωτική ικανότητα γ_i κάθε κατηγορίας κίνησης i ως συνάρτηση του φορτίου κίνησης της γραμμής p. Παρατηρούμε ότι για χαμηλές τιμές του φορτίου p, το σύστημα είναι σχεδόν πάντοτε άδειο, επομένως ο στιγμιαίος ρυθμός μετάδοσης κάθε ροής της κατηγορίας i είναι ίσος με c_i. Αυτό έχει ως αποτέλεσμα, η διεκπεραιωτική ικανότητα της κατηγορίας κίνησης i να βρίσκεται κοντά στην τιμή c_i. Για υψηλές τιμές του φορτίου p, το σύστημα χρειάζεται να εξυπηρετήσει πολλές ροές, με αποτέλεσμα ο στιγμιαίος ρυθμός μετάδοσης κάθε ροής να είναι πολύ μικρός (βλ. σχήμα 2). Στην περίπτωση αυτή, η διεκπεραιωτική ικανότητα όλων των κατηγοριών κίνησης τείνει στο μηδέν. Στον πίνακα 2 παρουσιάζονται αναλυτικά τα αποτελέσματα της διεκπεραιωτικής ικανότητας των τριών πρώτων και της τελευταίας (δέκατης) κατηγορίας κίνησης.



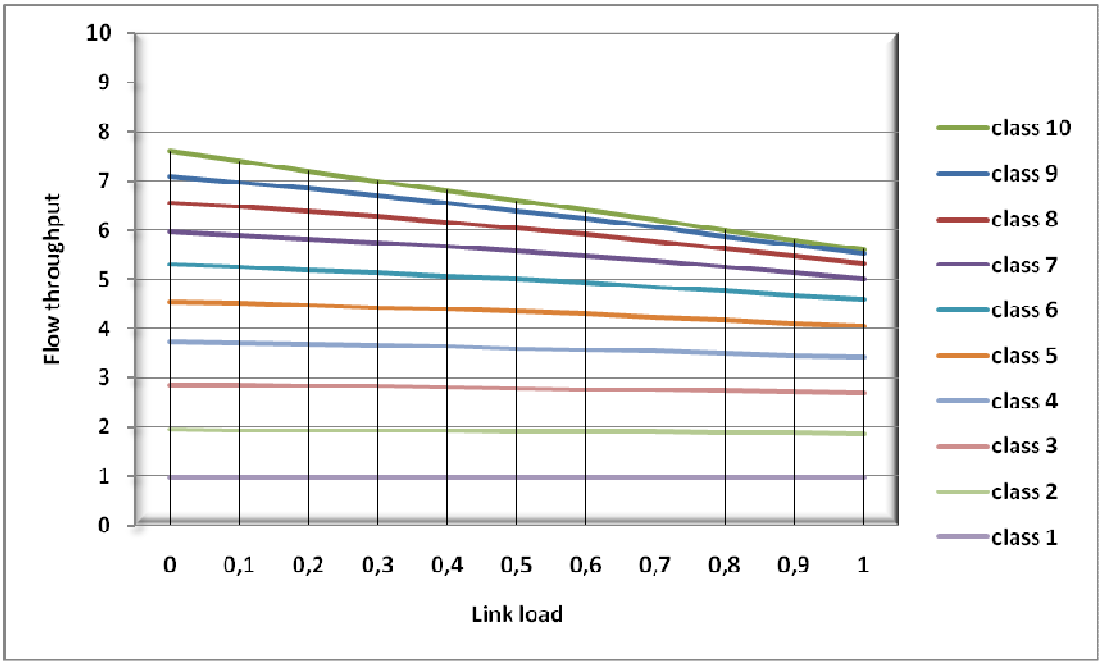
Σχήμα 2: Διεκπεραιωτική ικανότητα συναρτήσει του φορτίου p (παράδειγμα 3)

Link load	FlowThroughput (1η κατηγορία)	FlowThroughput (2η κατηγορία)	FlowThroughput (3η κατηγορία)	FlowThroughput (10η κατηγορία)
0	0,999	1,999	2,997	9,9
0,1	0,997	1,988	2,969	9
0,2	0,992	1,966	2,912	8
0,3	0,983	1,927	2,823	7
0,4	0,968	1,869	2,693	6
0,5	0,945	1,782	2,511	5
0,6	0,909	1,656	2,262	4
0,7	0,851	1,47	1,925	3
0,8	0,75	1,19	1,469	2
0,9	0,547	0,749	0,851	1
1	0	0	0	0

Πίνακας 2: Αναλυτικά αποτελέσματα του 3ου παραδείγματος για τις κατηγορίες κίνησης 1, 2, 3 και 10

4ο παράδειγμα

Θεωρούμε μια επέκταση του τρίτου παραδείγματος με την εξής διαφορά: η ένταση της κίνησης για κάθε κατηγορία δίνεται από την σχέση: $a_3 = a_4 = ... = a_{10} = 0.3$, ενώ $a_1 = a_2 = \frac{pC}{N}$. Με άλλα λόγια, μεταβάλλεται μόνο η ένταση της κίνησης των δύο πρώτων κατηγοριών. Στο σχήμα 3 παρουσιάζεται η διεκπεραιωτική ικανότητα γ_i κάθε κατηγορίας κίνησης *i* ως συνάρτηση του φορτίου κίνησης της γραμμής *p*. Παρατηρούμε ότι για σχεδόν όλες τις τιμές του *p*, το σύστημα είναι σχεδόν άδειο, επομένως ο στιγμιαίος ρυθμός μετάδοσης κάθε ροής της κατηγορίας *i* είναι σχεδόν ίσος με c_i. Αυτό έχει ως αποτέλεσμα, η διεκπεραιωτική ικανότητα της κατηγορίας κίνησης *i* να βρίσκεται κοντά στην τιμή c_i. Το παραπάνω συμπέρασμα είναι πιο έντονο για χαμηλές τιμές του *p*. Στον πίνακα 3 παρουσιάζονται αναλυτικά τα αποτελέσματα της διεκπεραιωτικής ικανότητας των τριών πρώτων και της τελευταίας (δέκατης) κατηγορίας κίνησης.



Σχήμα 3: Διεκπεραιωτική ικανότητα συναρτήσει του φορτίου *p* (παράδειγμα 4)

Link load	FlowThroughput (1η κατηγορία)	FlowThroughput (2η κατηγορία)	FlowThroughput (3η κατηγορία)	FlowThroughput (10η κατηγορία)
0	0,9881	1,9462	2,8650	7,5998
0,1	0,9867	1,9406	2,8522	7,4
0,2	0,9852	1,9345	2,8384	7,2
0,3	0,9836	1,9279	2,8235	7,0
0,4	0,9818	1,9208	2,8075	6,8
0,5	0,9799	1,9131	2,7902	6,6
0,6	0,9778	1,9047	2,7715	6,4
0,7	0,9755	1,8956	2,7513	6,2
0,8	0,9730	1,8857	2,7293	6,0
0,9	0,9702	1,8748	2,7056	5,8
1	0,9672	1,8630	2,6798	5,6002

Πίνακας 3: Αναλυτικά αποτελέσματα του 4ου παραδείγματος για τις κατηγορίες κίνησης 1, 2, 3 και 10

ΚΕΦΑΛΑΙΟ 2

Η πιθανότητα απώλειας πακέτων σε IP δίκτυα

2.1 Εισαγωγή

Στόχος αυτού του κεφαλαίου είναι η παρουσίαση ενός προσεγγιστικού και αναδρομικού τύπου για τον υπολογισμό της πιθανότητας απώλειας πακέτων σε δίκτυο IP τεχνολογίας. Η ανάλυση που ακολουθεί βασίζεται σε μια τηλεπικοινωνιακή γραμμή, της οποίας το διαθέσιμο εύρος ζώνης μοιράζεται σε ροές πολλών κατηγοριών κίνησης σύμφωνα με την πολιτική balanced fairness που παρουσιάστηκε στο πρώτο κεφάλαιο. Ο αναδρομικός τύπος βασίζεται στον ακριβή αναδρομικό τύπο των Kaufman-Roberts, ο οποίος αποτελεί γενίκευση της Erlang B formula και χρησιμοποιείται για τον υπολογισμό της πιθανότητας απώλειας κλήσεων σε ενσύρματα δίκτυα.

2.2 Περιγραφή του μοντέλου

Θεωρούμε εκ νέου μια γραμμή χωρητικότητας C η οποία εξυπηρετεί ροές προερχόμενες από N κατηγορίες κίνησης. Οι ροές της κατηγορίας i (i=1,…, N) έχουν μέγιστο και μέσο ρυθμό μετάδοσης c_i, σ_i αντίστοιχα, ενώ η άφιξη των ροών γίνεται μέσω μιας διαδικασίας Poisson με ρυθμό λ_i. Η ένταση της κίνησης της κατηγορίας i, a_i, ορίζεται από την σχέση (1) ($a_i = \lambda_i \sigma_i$) ενώ το φορτίο κίνησης στην γραμμή λόγω της κατηγορίας i, p_i, από την σχέση (2) ($p_i = \frac{a_i}{C}$). Βασιζόμενοι στην σχέση (2), μπορούμε να υπολογίσουμε το συνολικό φορτίο κίνησης της γραμμής, p, μέσω της σχέσης (3) ($p = \sum_{i=1}^{N} p_i$).

Όπως και στην περίπτωση του πρώτου κεφαλαίου, ορίζουμε το διάνυσμα $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ όπου x_i είναι ο αριθμός των ροών της κατηγορίας i που εξυπηρετούνται από την γραμμή καθώς και το διάνυσμα: α) των μέγιστων ρυθμών μετάδοσης $\boldsymbol{c} = (c_1, c_2, ..., c_N)$, β) της έντασης κίνησης $\boldsymbol{a} = (a_1, a_2, ..., a_N)$.

Η πιθανότητα $P(\boldsymbol{x})$ (το σύστημα να βρίσκεται στην κατάσταση $\boldsymbol{x}$) υπολογίζεται από την σχέσεις (15)-(16), ενώ η συνάρτηση $\Phi(\boldsymbol{x})$ μπορεί να υπολογιστεί αναδρομικά μέσω της σχέσης (12) (και των βοηθητικών σχέσεων (4)-(6)).

2.3 Υπολογισμός της πιθανότητας απώλειας πακέτων

Ο υπολογισμός της πιθανότητας απώλειας πακέτων βασίζεται στην υπόθεση (προσέγγιση) ότι απώλειες πακέτων συμβαίνουν κατά κύριο λόγο όταν η γραμμή είναι υπερφορτωμένη. Δηλαδή, όταν η γραμμή προσπαθεί να εξυπηρετήσει πολλές ροές πολλών κατηγοριών κίνησης.

Σ' αυτή την περίπτωση και θεωρώντας επίσης ότι οι ροές κίνησης εξυπηρετούνται με τον μέγιστο ρυθμό μετάδοσης, η κίνηση που χάνεται (lost traffic) σε μια κατάσταση $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ είναι ίση με $\boldsymbol{x.c} - C$ όταν $\boldsymbol{x.c} > C$ (οπότε έχουμε ξεπεράσει την χωρητικότητα της γραμμής), διαφορετικά αν $\boldsymbol{x.c} \le C$ τότε όλη η κίνηση εξυπηρετείται.

Προκειμένου να υπολογιστεί η πιθανότητα απώλειας πακέτων προτείνεται η ακόλουθη προσεγγιστική σχέση [6]:

$$L = \frac{\sum_{x:x.c>C} (\boldsymbol{x.c} - C)\, X(\boldsymbol{x})}{\sum_{x} (\boldsymbol{x.c})\, X(\boldsymbol{x})} \tag{28}$$

Η σχέση (28) υπολογίζεται εύκολα για μικρά συστήματα που εξυπηρετούν λίγες κατηγορίες κίνησης. Στην περίπτωση όμως που ένα σύστημα έχει υψηλή χωρητικότητα και εξυπηρετεί πολλές κατηγορίες κίνησης τότε ο υπολογισμός του L είναι χρονοβόρος. Προκειμένου να αντιμετωπιστεί το πρόβλημα αυτό, προτείνεται στην επόμενη ενότητα ένας προσεγγιστικός αλλά αναδρομικός τύπος υπολογισμού της πιθανότητας απώλειας πακέτων L.

2.4 Αναδρομικός τύπος υπολογισμού της πιθανότητας απώλειας πακέτων

Υποθέτουμε ότι η χωρητικότητα της γραμμής C και οι μέγιστοι ρυθμοί μετάδοσης $c_1, c_2, ..., c_N$ είναι ακέραιοι αριθμοί.

Για κάθε ακέραιο n με $n = 1,\ldots,C$, ορίζουμε την πιθανότητα το σύστημα να βρίσκεται στην κατάσταση n, $p(n)$, ως:

$$p(n) = \sum_{\boldsymbol{x}:\boldsymbol{x.c}=n} X(\boldsymbol{x}) \tag{29}$$

Ορίζουμε επίσης ως:

$$\overline{p} = \sum_{n>C} p(n) \tag{30}$$

και

$$\overline{q} = \sum_{n>C} np(n) \tag{31}$$

Μέσω των σχέσεων (29), (30) και (31) η σχέση (28) γράφεται ως εξής:

$$L = \frac{\overline{q} - C\,\overline{p}}{\sum_{n=1}^{C} np(n) + \overline{q}} \tag{32}$$

όπου ο υπολογισμός του L βασίζεται στην εύρεση των τιμών $p(n)$ για $n = 0,1,\ldots,C$ καθώς και στην εύρεση των $\overline{p}$, $\overline{q}$ για $n>C$.

Προκειμένου να υπολογίσουμε τις τιμές των $p(n)$ χρησιμοποιούμε την σχέση [6]:

$$p(n) = \begin{cases} 1 & , \gamma\iota\alpha\ n = 0 \\ \sum_{j=1}^{N} \frac{a_j}{n} p(n - c_j) & , \gamma\iota\alpha\ n = 1,\ldots,C \\ 0 & , \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \tag{33}$$

Στην περίπτωση όπου $n > C$ έχουμε ότι [6]:

$$\overline{p} = \sum_{i=1}^{N} \frac{p_i\,\overline{p}_i}{1-p} \tag{34}$$

όπου

$$\overline{p}_i = \sum_{C-c_i<n\le C} p(n) \tag{35}$$

και

$$\overline{q} = \sum_{i=1}^{N} \frac{p_i(q_i + c_i(p + \overline{p}_i))}{1-p} \tag{36}$$

όπου

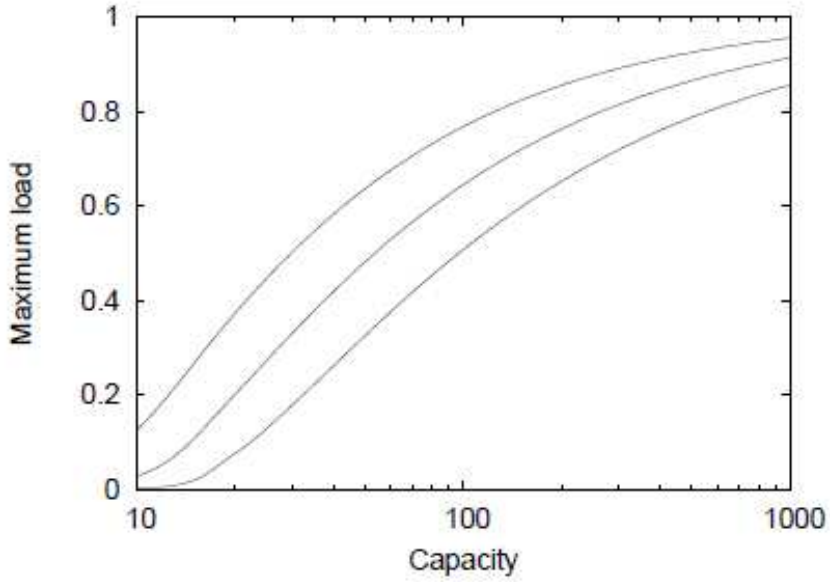
$$q_i = \sum_{C-c_i<n\le C} np(n) \tag{37}$$

Η απόδειξη των σχέσεων (34)-(37) παραλείπεται. Ο αναγνώστης μπορεί να ανατρέξει στην εργασία [6].

2.5 Παραδείγματα

5ο παράδειγμα

Θεωρούμε λοιπόν μια γραμμή χωρητικότητας *C*, η οποία εξυπηρετεί *N* = 10 κατηγορίες κίνησης. Οι μέγιστοι ρυθμοί μετάδοσης των κατηγοριών κίνησης είναι οι εξής: $c_1 = 1, c_2 = 2, ..., c_{10} = 10$. Αντίστοιχα η ένταση της κίνησης είναι ίδια για όλες τις κατηγορίες κίνησης. Στο σχήμα 4, παρουσιάζεται το μέγιστο φορτίο που μπορεί να μεταφέρει η γραμμή για διάφορες τιμές της απώλειας πακέτων *L* (5%, 1% και 0.1% από πάνω προς τα κάτω στην γραφική παράσταση) και διάφορες τιμές της χωρητικότητας *C* [6].



Σχήμα 4: Μέγιστο φορτίο γραμμής συναρτήσει της χωρητικότητας *C*

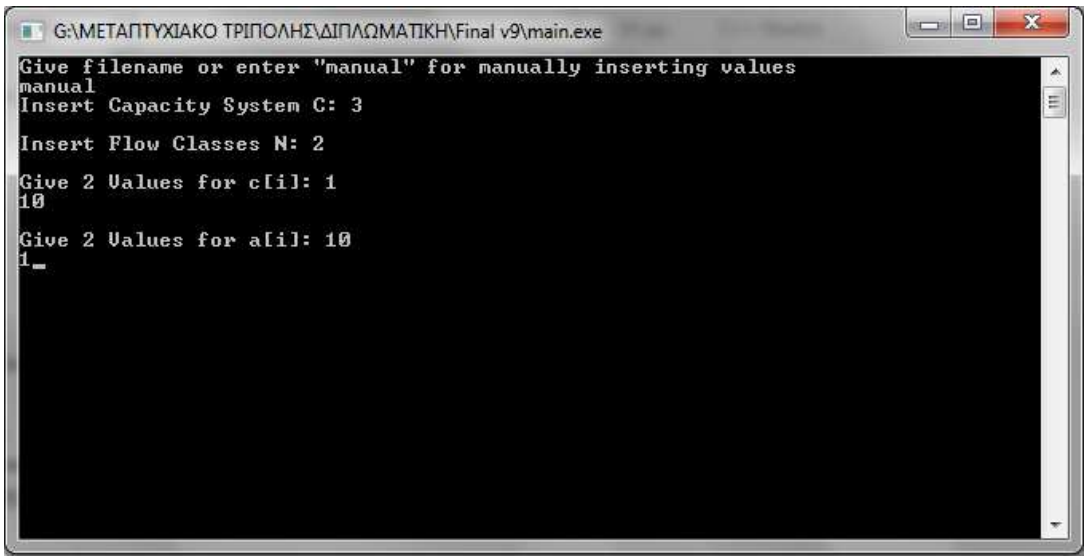
ΚΕΦΑΛΑΙΟ 3

Υλοποίηση προγράμματος

3.1 Τρόποι εισαγωγής δεδομένων

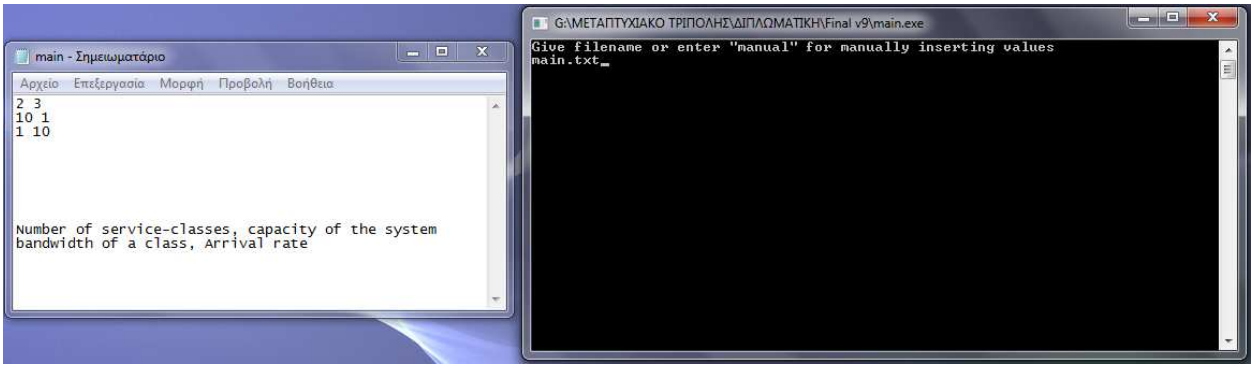
Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της εργασίας είναι εξολοκλήρου η γλώσσα υψηλού επιπέδου C++, με περιβάλλοντα μεταγλώττισης το **Microsoft Visual Studio 2010** και το **Bloodshed Dev-C++**.

Το συγκεκριμένο πρόγραμμα είναι προγραμματισμένο, έτσι ώστε να δέχεται δύο τρόπους εισόδου τιμών μεταβλητών, οι οποίοι διαχωρίζονται με μια εντολή απόφασης **«if»**. Ο πρώτος τρόπος, είναι είσοδος των τιμών δεδομένων από το χρήστη. Για να επιτευχθεί αυτό, τρέχουμε το εκτελέσιμο πρόγραμμα με τη βοήθεια της γραμμής εντολών (command promp) και στην πρώτη απόφαση του προγράμματος γράφουμε απλά την λέξη **«manual»**. Στη συνέχεια εισάγουμε τις τιμές που θέλουμε στα μηνύματα εκτύπωσης, όπως φαίνεται στην παρακάτω εικόνα 1:



Εικόνα 1: Εισαγωγή δεδομένων από το χρήστη

Ο δεύτερος τρόπος, είναι η εισαγωγή των τιμών από αρχείο. Για να επιτευχθεί αυτό, τρέχουμε πάλι το εκτελέσιμο πρόγραμμα με την βοήθεια της γραμμής εντολών (command promp) και στην πρώτη απόφαση του προγράμματος γράφουμε το όνομα και τον τύπο του αρχείου, έχοντας εισάγει και αποθηκεύσει πιο πριν τις τιμές των μεταβλητών στο συγκεκριμένο αρχείο, όπως φαίνεται στην παρακάτω εικόνα 2:



Εικόνα 2: Εισαγωγή δεδομένων από αρχείο

3.2 Πρόγραμμα - Κώδικας

Πιο κάτω παρατίθεται ο κώδικας του συγκεκριμένου προγράμματος σε C++.

```
001 #include <iostream>
002 #include <iomanip>
003 #include <fstream>
004 #include <sstream>
005
006 #define CLEAN_STREAM(inOp) if (!(inOp)) { cin.clear(); cin.ignore(0x7FFF, '\n'); }
007
008 using namespace std;
009
010 long C;
011 long N;
012 long double* b;
013 long double* a;
014 long double G=1.0;
015
016 int ignoreBlank(ifstream&);
017 bool loadFData(const char*);
018 long double Formula(long);
019 long double FormulaEx(long, long);
020 int loadFile(char *f, long double *, long double *);
021
022 int main(int argc, char* argv[ ])
023 {
024     int i, j, k;
025     bool fileLoaded = false;
026
027     b = new long double[1000];
028     a = new long double[1000];
```

```
029
030     cout << setprecision(15);
031
032     char *fname;
033     cout<<"Give filename or enter \"manual\" for manually inserting values"<<endl;
034     CLEAN_STREAM(cin >> fname);
035     if(!strstr(fname, "manual"))
036         loadFile(fname, a, b);
037     Else
038     {
039         cout << "Insert Capacity System C: ";
040         CLEAN_STREAM(cin >> C);
041
042         while (C <= 0)
043         {
044             cout << "Please try again: ";
045             CLEAN_STREAM(cin >> C);
046         }
047
048         cout << "\nInsert Flow Classes N: ";
049         CLEAN_STREAM(cin >> N);
050
051         while (N <= 0)
052         {
053             cout << "Please try again: ";
054             CLEAN_STREAM(cin >> N);
055         }
056         Do
057         {
058             cout << "\nGive " << N << " Values for c[i]: ";
059             for (i = 0; i < N; i++)
060             {
061                 if (!(cin >> b[i]))
062                    break;
063             }
064
065             if (i < N)
066             {
067                cout << "\nPlease try again.";
068                CLEAN_STREAM(false);
069             }
070         }
071         while (i < N);
072
073         Do
074         {
075             cout << "\nGive " << N << " Values for a[i]: ";
```

```
076             for (i = 0; i < N; i++){
077                 if (!(cin >> a[i]))
078                    break;
079             }
080 
081             if (i < N)
082             {
083                cout << "\nPlease try again.";
084                CLEAN_STREAM(false);
085             }
086         }
087         while (i < N);
088     }
089 
090     cout<<"\n**************************************************"<<endl;
091     cout<<"**************************************************\n"<<endl;
092     cout<<"C: "<<C<<endl;
093     cout<<"N: "<<N<<endl;
094     cout<<"\n**************************************************\n"<<endl;
095     cout<<"---> Control of Entry c[i]\n"<<endl;
096     for(int i=0; i<10;i++)
097         cout<<"c["<<i+1<<"]="<<b[i]<<endl;
098         cout<<"\n**************************************************\n"<<endl;
099     cout<<"---> Control of Entry a[i]\n"<<endl;
100     for(int i=0; i<10;i++)
101         cout<<"a["<<i+1<<"]="<<a[i]<<endl;
102         cout<<"\n**************************************************"<<endl;
103 
104     cout<<"\n_____________________________________________________"<<endl;
105     cout<<"Unnormalized Values p(n)"<<endl;
106     cout<<"_____________________________________________________\n"<<endl;
107     for (i = 1; i <= C; i++)
108     {
109         cout <<"p(" << i << ")\t= " << Formula(i) << '\n';
110     }
111 
112     for (i = 1; i <= C; i++)
113     {
114         G=G+Formula(i);
115     }
116     cout<<"\nThe Normalization Constant is G = " << G <<endl;
117 
118     cout<<"\n_____________________________________________________"<<endl;
119     cout<<"Normalized Values p(n)"<<endl;
120     cout<<"_____________________________________________________\n"<<endl;
121     for (i = 1; i <= C; i++)
122     {
```

```
123         cout <<"p(" << i << ")\t= " << Formula(i)/G << '\n';
124     }
125 
126     cout<<"\n_____________________________________________________"<<endl;
127     cout<<"Unnormalized Values q[i](n)"<<endl;
128     cout<<"_____________________________________________________\n"<<endl;
129     for (k = 0; k < N; k++)
130     {
131         for (j = 1; j <= C; j++)
132         {
133             cout << "q[" << k+1 << "](" << j << ")\t= " << FormulaEx(k, j) << '\n';
134         }
135     }
136 cout<<"\n\n*************************************************\n"<<endl;
137 
138 long double pmeso=0.0;
139 long double pe=0.0;
140 for (int i=0; i<N; i++)
141 {
142     pe+=a[i]/C;
143 }
144 long double qmesosglobal=0;
145 for (int i=0; i<N; i++)
146 {
147     long double pform=0;
148     long double ptemp=a[i]/C;
149     for(int l=C-b[i]+1;l<=C;l++)
150             pform+=Formula(l);
151 
152     pmeso+=(ptemp*pform)/(1-pe);
153 }
154 
155 cout << "pbar:\t\t\t" << pmeso << endl;
156 cout<<"\n*************************************************\n"<<endl;
157 
158 long double ff=1;
159 for (int i = 1; i <= C; i++)
160 {
161     ff+=Formula(i);
162 }
163 
164 for(int i=0; i<N; i++)
165 {
166         long double qmesos;
167         long double ptemp=a[i]/C;
168         long double pform=0;
169         for(int l=C-b[i]+1;l<=C;l++)
```

```
                pform+=Formula(l);

        long double seiraq=0;
        for (int ii=0; ii<N; ii++)
        {
            long double qtemp=a[ii]/C;
            long double qform=0;
            for(int l=C-b[ii]+1;l<=C;l++)
                    qform+=FormulaEx(i,l);

            cout<<i+1<<"\t"<<ii+1<<"\t"<<"Qbarij:\t\t"<<qform<<endl;

            seiraq+=(qtemp*qform)/(1-pe);

       }
cout<<"\n*************************************************\n"<<endl;
        qmesos=ptemp*(pform+pmeso)/(1-pe) + seiraq;

        cout<<i+1<<"qbar:\t\t\t\t"<<qmesos<<endl;

        long double ffex=0;
        for (int j = 1; j <= C; j++)
        {
            ffex+= FormulaEx(i, j) ;
        }
        long double gama1=a[i]*(ff+pmeso)/(ffex+qmesos);

        cout<<i+1<<"throughput:\t\t\t"<<gama1<<endl;

}
cout<<"\n*************************************************\n"<<endl;
cout<<"\n\n\n------------Press [Enter] to terminate...------------\n\n"<<endl;
cout<<"_________________Psarras Dimitrios_________________\n\n"<<endl;

CLEAN_STREAM(false);
cin.get();
delete [] b;
delete [] a;

return 0;
}

long double FormulaEx(long k, long j)
{
    long double reciprocal_j = 1.0L/j;
    long double result = reciprocal_j*a[k]*Formula(j - b[k]);
    long i;
```

```
217
218     if (j <= 0)
219        result = 0.0L;
220     Else
221     {
222         for (i = 0; i < N; i++)
223             result += reciprocal_j*a[i]*FormulaEx(k, j - b[i]);
224     }
225
226     return result;
227 }
228
229 long double Formula(long j)
230 {
231     long double result = 0.0L;
232     long double reciprocal_j = 1.0L/j;
233     long k;
234
235     if (j < 0)
236        result = 0.0L;
237     else if (j == 0)
238        result = 1.0L;
239     Else
240     {
241         for (k = 0; k < N; k++)
242             result += reciprocal_j*a[k]*Formula(j - b[k]);
243     }
244
245     return result;
246 }
247
248 bool loadFData(const char* filepath)
249 {
250     ifstream infile;
251     char temp;
252     bool bad = false;
253     int i;
254
255     infile.open(filepath);
256
257     if (infile.is_open())
258     {
259         while (!bad)
260         {
261             temp = ignoreBlank(infile);
262             if (temp == EOF)
263                break;
```

```
264
265             switch (temp)
266             {
267                 case 'C':
268                    temp = ignoreBlank(infile);
269
270                    if (temp != '=' || !(infile >> C))
271                       bad = true;
272                 break;
273                 case 'N':
274                    temp = ignoreBlank(infile);
275
276                    if (temp != '=' || !(infile >> N))
277                       bad = true;
278                 break;
279                 case 'b':
280                    if (b != NULL)
281                    {
282                       delete [] b;
283                       b = NULL;
284                    }
285
286                    if (N <= 0)
287                    {
288                        cout << "N is uninitialised or has an invalid value\n";
289                        bad = true;
290                    }
291                    Else
292                    {
293                        b = new long double[N];
294
295                        temp = ignoreBlank(infile);
296
297                        if (temp == '=')
298                        {
299                            temp = ignoreBlank(infile);
300
301                            if (temp == '{')
302                            {
303                                for (i = 0; i < N; i++){
304                                    if (!(infile >> b[i]))
305                                    {
306                                        bad = true;
307                                        break;
308                                    }
309                                    temp = ignoreBlank(infile);
310
```

```
311                                 if (temp != ',' && (i == N-1 && temp != '}'))
312                                 {
313                                     bad = true;
314                                     break;
315                                 }
316                             }
317                         }
318                         Else
319                            bad = true;
320                     }
321                     Else
322                        bad = true;
323                 }
324             break;
325             case 'a':
326                 if (a != NULL)
327                 {
328                    delete [] a;
329                    a = NULL;
330                 }
331 
332                 if (N <= 0)
333                 {
334                     printf("N is uninitialised or has an invalid value\n");
335                     bad = true;
336                 }
337                 Else
338                 {
339                     a = new long double[N];
340 
341                     temp = ignoreBlank(infile);
342 
343                     if (temp == '=')
344                     {
345                         temp = ignoreBlank(infile);
346 
347                         if (temp == '{')
348                         {
349                             for (i = 0; i < N; i++){
350                                 if (!(infile >> a[i]))
351                                 {
352                                     bad = true;
353                                     break;
354                                 }
355                                 temp = ignoreBlank(infile);
356 
357                                 if (temp != ',' && (i == N-1 && temp != '}'))
```

```
                                    {
                                        bad = true;
                                        break;
                                    }
                                }
                            }
                            Else
                               bad = true;
                        }
                        Else
                           bad = true;
                    }
                break;
                default:
                   cout << "Unknown symbol \'" << temp << "\', ignoring...\n";
            }
        }

        if (!bad)
           return true;
        Else
        {
            cout << "Corrupted file! Ignoring...\n";
            return false;
        }
    }

    cout << '\'' << filepath << "\' doesn't exist or it's already locked by another program.\n";

    return false;
}

int ignoreBlank(ifstream& infile)
{
    char temp;

    Do
    {
        if (!infile.get(temp))
           return EOF;

    }while (temp == ' '  || temp == '\t' ||
            temp == '\v' || temp == '\r' ||
            temp == '\n');

    return temp;
}
```

```
405
406 int loadFile(char *f, long double *a, long double *b)
407 {
408     ifstream ifile (f);
409     string line;
410     int cou=0;
411     if (ifile.is_open())
412     {
413         getline (ifile,line);
414         long double item;
415         std::stringstream os(line);
416
417         os>>item;
418         N=item;
419         os>>item;
420         C=item;
421
422         while (!ifile.eof())
423         {
424             getline (ifile,line);
425             std::stringstream os(line);
426
427             cout<<"cou "<<cou<<endl;
428             os>>item;
429             a[cou]=item;
430             os>>item;
431             b[cou]=item;
432
433             cou++;
434         }
435     }
436     Else
437         return -1;
438     ifile.close();
439     cout<<"Exiting normally"<<endl;
440     return 1;
441 }
```

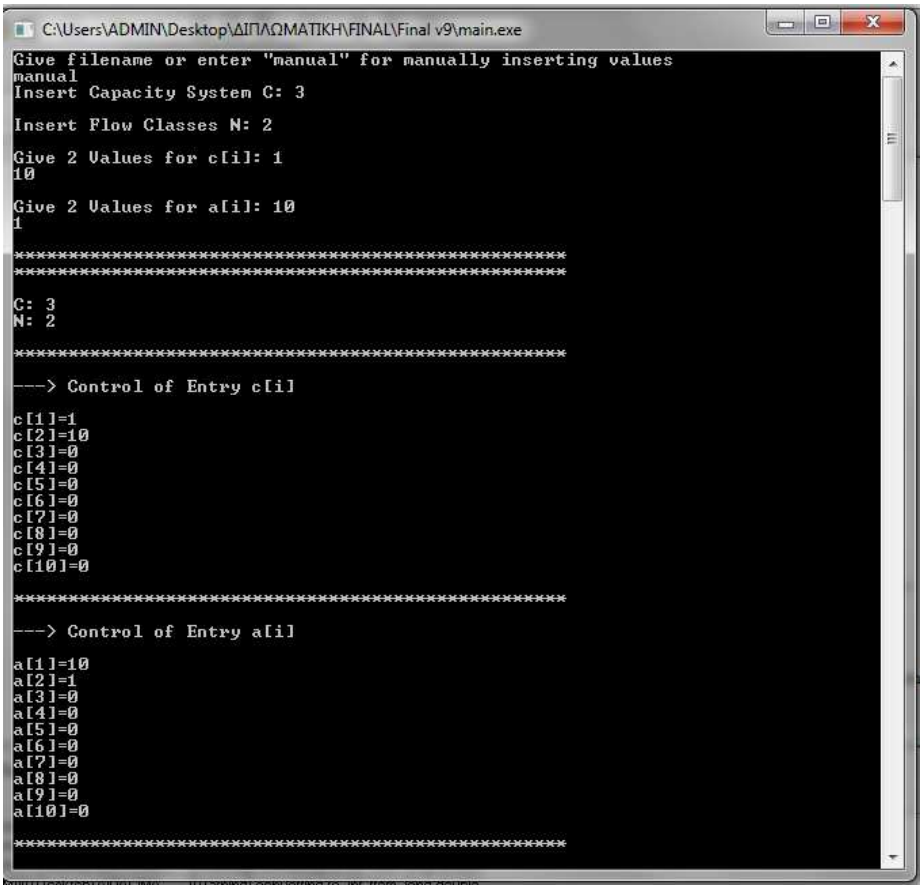
3.3 Επεξήγηση Προγράμματος

Στις γραμμές **(001-004)** εκτελούμε με την "#" μια εντολή προεπεξεργασίας, ώστε να προετοιμάσουμε το πρόγραμμά μας. Πιο συγκεκριμένα καλούμε τον προεπεξεργαστή του προγράμματος μεταγλώττισης, ώστε να συμπεριλάβει τις εξής βιβλιοθήκες:

- **iostream**
- **iomanip**
- **fstream**
- **sstream**

Οι βιβλιοθήκες αυτές, είναι αναγκαίες να δηλωθούν στην αρχή του προγράμματος και περιέχουν όλες αυτές τις εντολές, οι οποίες χρησιμοποιήθηκαν στο παραπάνω πρόγραμμα. Μόνο έτσι θα μπορεί να καταλάβει ο μεταγλωττιστής την χρήση της κάθε εντολής. Στη γραμμή **(005)** δηλώνουμε έναν «δυναμικό» πίνακα με όνομα «CLEAN_STREAM», ο οποίος δεσμεύει κελία και πόρους από την μνήμη RAM του υπολογιστή. Σε αυτά τα κελία γίνεται όλη η καταχώρηση τιμών του προγράμματος και όταν αυτό σταματήσει να τρέχει, τότε τα κελιά αδειάζουν αυτόματα. Επειδή τα δεδομένα που μπορεί να εισάγουμε από το συγκεκριμένο πρόγραμμα, μπορεί να μην είναι μικροί αριθμοί σε κλίμακα και σε μέγεθος και ακόμα για να πάρουμε όσο το δυνατόν πιο ακριβή αποτελέσματα, χρησιμοποιούμε μεταβλητές τύπου *«long»* για ακέραιους τύπους μεταβλητών και μεταβλητές τύπου *«long double»* για δεκαδικούς τύπους μεταβλητών (διπλού ελέγχου). Πιο σπάνια χρησιμοποιούμε μεταβλητές τύπου *«int»* μόνο για δηλώσεις μετρητών και δεικτών. Στις γραμμές **(010-011)** δηλώνω τις μεταβλητές **C** και **N**, οι οποίες είναι η χωρητικότητα και οι κατηγορίες κίνησης του συστήματος αντίστοιχα. Στις γραμμές **(012-013)** δηλώνω δύο πίνακες a και b, στους οποίους θα εισάγονται οι τιμές των μεταβλητών C, N, a[i], b[i] και θα ελέγχονται τα περιεχόμενα των πινάκων και οι διευθύνσεις των κελιών τους στο τέλος του προγράμματος και σε ξεχωριστή συνάρτηση με όνομα *loadFData*. Στη γραμμή **(014)** δηλώνω και αρχικοποιώ την μεταβλητή G, με την οποία θα υπολογίζω τις κανονικοποιημένες τιμές του *p(n)*. Στις γραμμές **(016-020)** δηλώνω τις συναρτήσεις με τις εισόδους και τους τύπους αποτελεσμάτων που θα δώσουν. Αυτές οι συναρτήσεις έχουν δημιουργηθεί έξω από την κεντρική συνάρτηση του προγράμματος με όνομα *«main»* και είναι υπεύθυνες, καλώντας τες, για τον υπολογισμό των σχέσεων (22) και (23), την διαδικασία εισαγωγής δεδομένων από αρχείο, καθώς και τον έλεγχο εισαγωγής των μεταβλητών. Στη γραμμή **(022)** καλώ την κεντρική συνάρτηση του προγράμματος και στην γραμμή **(024)** δηλώνω τους μετρητές που θα χρησιμοποιηθούν για τις επαναλήψεις εισαγωγής των μεταβλητών **C** και **N**. Οι πίνακες *a, b* που είχαμε δηλώσει πιο πάνω, στις γραμμές **(027-028)** δεσμεύουν χώρο στην μνήμη RAM από 1000 κελιά ο καθένας (δίνω συγκεκριμένο μέγεθος) για καταχώρηση.

Με το που τρέχουμε το πρόγραμμα, εμφανίζεται το μήνυμα «`Give filename or enter \"manual\" for manually inserting values`», το οποίο μας δίνει την δυνατότητα με τις εντολές απόφασης που υπάρχουν στις γραμμές **(035-088)**, να επιλέξουμε τον τρόπο εισαγωγής των δεδομένων. Οι δύο τρόποι εισαγωγής των δεδομένων αναφέρθηκαν στην ενότητα 2.1. Εάν θέλουμε να εισάγουμε τα δεδομένα από αρχείο, τότε θα εκτελεστούν οι γραμμές **(034-036)** μαζί με την δήλωση του ονόματος του αρχείου την γραμμή **(032)**. Εάν θέλουμε να γίνει εισαγωγή δεδομένων από το χρήστη θα ακολουθήσει το πρόγραμμα τις υπόλοιπες γραμμές **(037-088)**. Εάν τα δεδομένα που εισαχθούν για τις μεταβλητές **C** και **N** είναι αρνητικά, θα εμφανιστούν τα κατάλληλα μηνύματα προς τον χρήστη, ώστε να τον ενημερώσουν για την λανθασμένη εισαγωγή και να εισάγει νέα. Στις γραμμές **(089-103)** θα γίνει μια επαλήθευση και εκτύπωση των τιμών εισαγωγής των δεδομένων, όπως φαίνεται στην εικόνα 3:



Εικόνα 3: Έλεγχος εισαγωγής από το πρόγραμμα

Στις γραμμές **(104-111)**, έχουμε την εκτύπωση των αποτελεσμάτων των μη κανονικοποιημένων τιμών για το *p(n)*. Τα αποτελέσματα του *p(n)* βρίσκονται μέσα σε ένα πίνακα με όνομα *Formula(i)*, του οποίου ο υπολογισμός γίνεται μέσα σε μια άλλη

συνάρτηση του προγράμματος στις γραμμές **(229-246)**. Όμοια, ισχύει και για το $q_i(n)$, το οποίο υπολογίζεται στις γραμμές **(126-136)** και τα αποτελέσματά του αποθηκεύονται σε έναν πίνακα με όνομα *FormulaEx(k, j)*. Ο υπολογισμός αυτού του πίνακα γίνεται και αυτός εκτός κεντρικής συνάρτησης, σε μια άλλη συνάρτηση στις γραμμές **(212-227)**.

Συμπερασματικά, τα αποτελέσματα της σχέσης (22) $p(n)=\sum_{j=1}^{N}\frac{a_j}{n}p(n-c_j)$ βρίσκονται στον πίνακα *Formula(i)* και τα αποτελέσματα της σχέσης $q_i(n)=\frac{a_i}{n}p(n-c_i)+\sum_{j=1}^{N}\frac{a_j}{n}q_i(n-c_j)$ βρίσκονται στον πίνακα *FormulaEx(k, j)*.

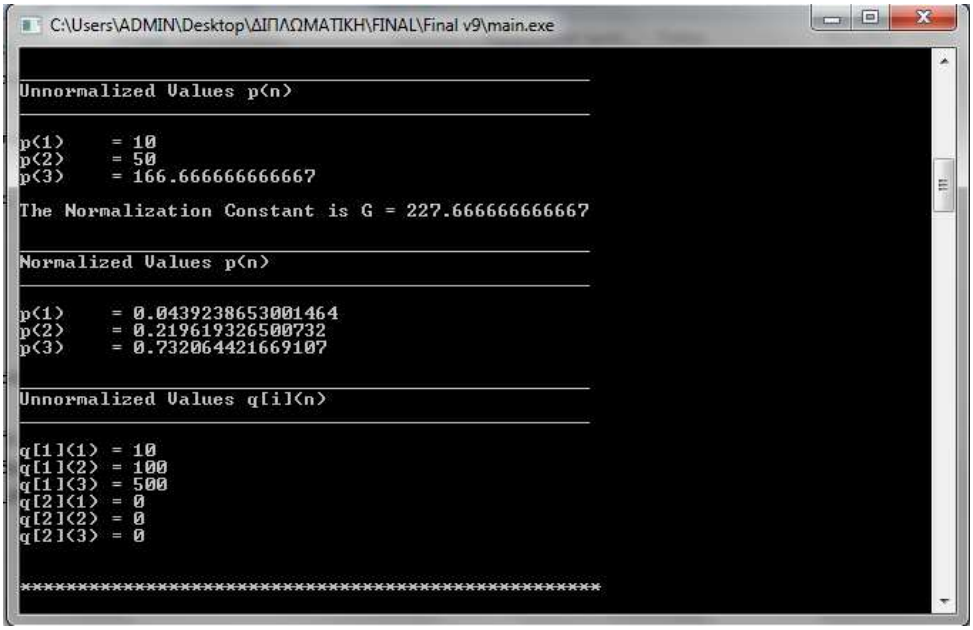
Στον τύπο του *p(n)* στη γραμμή **(242)** έχουμε:

Μαθηματική σχέση	**Πρόγραμμα**
α_j	a[k]
n	j
c_j	b[k]

Στον τύπο του $q_i(n)$, στις γραμμές **(215),(223)** έχουμε:

Μαθηματική σχέση	**Πρόγραμμα**
α_i	a[k]
n	j
c_i	b[k]
α_j	a[i]
c_j	b[i]

Τα αποτελέσματα του *p(n)* και του $q_i(n)$ εμφανίζονται με τον εξής τρόπο στο πρόγραμμα:



Εικόνα 4: Εμφάνιση αποτελεσμάτων για *p(n) και $q_i(n)$*

Στη συνέχεια, στη γραμμές **(140-143)** υπολογίζουμε το *p*, όπου:

.

Μαθηματική σχέση	**Πρόγραμμα**
α_i	a[i]
P	pe

αφού πιο πριν έχω αρχικοποιήσει το *p* στη γραμμή **(139).**

Στις ακόλουθες γραμμές **(145-153)** υπολογίζουμε το $\overline{p}$, αφού πιο πριν έχω υπολογίσει το p_i και το $\overline{p}_i$. Το $\overline{p}$ στον κώδικα αναπαριστάται ως *pmeso*, το p_i ως *ptemp* και το $\overline{p}_i$ ως *pform*.

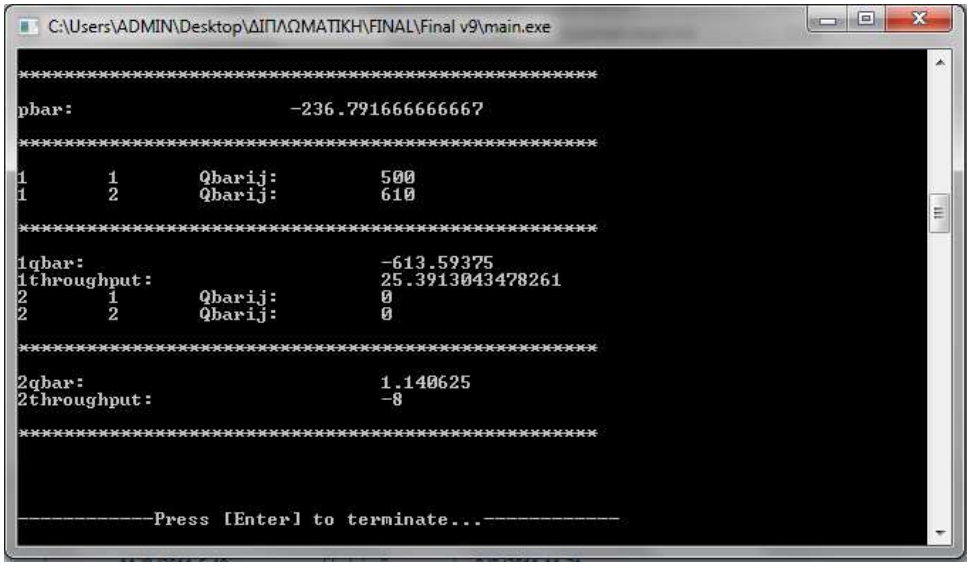
Μαθηματική σχέση	**Πρόγραμμα**
$\overline{p}$	pmeso
p_i	ptemp
$\overline{p}_i$	pform

Στην γραμμή **(155)** γίνεται η εκτύπωση του $\overline{p}$.

Στις γραμμές **(164-189)** υπολογίζονται και τυπώνονται τα αποτελέσματα για τα $\overline{q}_i$ και $\overline{q}_{ij}$. Πιο συγκεκριμένα, έχω ορίσει το $\sum_{j=1}^{N} \frac{p_j \overline{q}_{ij}}{1-p}$, δηλαδή το δεύτερο μέρος της σχέσης

(26), ως μία μεταβλητή με όνομα *seiraq*. Τα αποτελέσματα της μεταβλητής *seiraq* τα τυπώνω στη γραμμή **(182)**. Το p_j αναπαριστάται ως *qtemp* και το $\bar{q}_{ij}$ αναπαριστάται ως *qform*. Το $\bar{q}_{ij}$ υπολογίζεται και τυπώνεται στις γραμμές **(177-180).** Αφού έχω υπολογίσει το $\bar{q}_{ij}$ και δια μέσω αυτού, το άθροισμα σχέσης (26), είμαι σε θέση να υπολογίσω και το $\bar{q}_i$, γνωρίζοντας από τους παραπάνω υπολογισμούς το p_i, το $\bar{p}_i$, το $\bar{p}$ και το p. Το $\bar{q}_i$ είναι η μεταβλητή *qmesos* στο πρόγραμμα και τα αποτελέσματα τυπώνονται στη γραμμή **(186)**. Οι τελευταίοι υπολογισμοί στο πρόγραμμα γίνονται στις γραμμές **(190-199)**. Σε αυτές τις γραμμές υπολογίζεται το *flowthroughput* και η μεταβλητή που το αναπαριστά ονομάζεται *gamma1*. Τα αποτελέσματα του *Flowthroughput* τυπώνονται στην γραμμή **(197)**.

Τα αποτελέσματα του εκτελέσιμου αρχείου για τα $\bar{p}$, $\bar{q}_{ij}$, $\bar{q}_i$ και *flowthroughput* απεικονίζονται στο πρόγραμμα όπως φαίνεται στην εικόνα 5:



Εικόνα 5: Εμφάνιση αποτελεσμάτων για $\bar{p}$, $\bar{q}_{ij}$, $\bar{q}_i$ και *flowthroughput*

Τέλος, στις γραμμές (248-441), έχουν δημιουργηθεί συναρτήσεις με περισσότερους ελέγχους για τις τιμές των μεταβλητών που εισάγουμε, καθώς και τον κώδικα για την εισαγωγή των μεταβλητών από αρχείο και όχι από το χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] T. Bonald and A. Proutiere, “Insensitive bandwidth sharing in data networks”, Queueing Systems, Vol. 44, pp. 69–100, May 2003.

[2] T. Bonald and J. Virtamo, “A Recursive Formula for Multirate Systems with Elastic Traffic”, IEEE Communications Letters, Vol. 9, No. 8, pp. 753-755, August 2005.

[3] Μ. Δ. Λογοθέτης, «ΘΕΩΡΙΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΗΣ ΚΙΝΗΣΕΩΣ ΚΑΙ ΕΦΑΡΜΟΓΕΣ», Εκδόσεις Παπασωτηρίου, Αθήνα 2001, ISBN 960-7510-84-4.

[4] T. Bonald, A. Proutiere, J. Roberts, and J. Virtamo, “Computational aspects of balanced fairness”, in *Proc. ITC 18*, 2003, pp. 801–810.

[5] J. S. Kaufman, “Blocking in a shared resource environment”, IEEE Trans. Commun. Vol. 29, No. 10, pp. 1474-1481, October 1981.

[6] T. Bonald, "A Recursive Formula for Estimating the Packet Loss Rate in IP Networks", in Proc. VALUETOOLS 2009, Pisa, Italy, October 2009.

[7] J. W. Roberts, A service system with heterogeneous user requirement, in: *Performance of Data Communications Systems and Their Applications*, G. Pujolle, Ed. Amsterdam, The Netherlands: North-Holland, 1981, pp. 423–431.