Πανεπιστήμιο Θεσσαλίας

Αλγόριθμοι και τεχνικές για παροχή ποιότητας υπηρεσιών σε Peer-to-Peer δίκτυα

Τμήμα Μηχ/κών Η/Υ Τηλεπικοινωνιών & Δικτύων

Ζαχαρακόπουλος Χρήστος

ΠΡΟΠΤΥΧΙΑΚΟΣ ΦΟΙΤΗΤΗΣ ΤΜΗΜΑΤΟΣ
ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ : ΚΟΥΤΣΟΠΟΥΛΟΣ ΙΟΡΔΑΝΗΣ

ΒΟΛΟΣ 2005

Σύνοψη

Σκοπός της ανάλυσης που ακολουθεί είναι να γίνουν ευρύτερα γνωστές και κατανοητές οι δυνατότητες και οι προοπτικές των Peer-to-Peer δικτύων και να αναλυθούν τεχνικές με τις οποίες μπορεί να βελτιωθεί η ποιότητα υπηρεσιών που παρέχουν. Πραγματοποιείται αρχικά επισκόπηση των χρήσεων των P2P δικτύων, των προκλήσεων που έχουν να αντιμετωπίσουν, καθώς και αναφορά των χαρακτηριστικών μερικών από τα σημαντικότερα. Έπειτα, προσομοιώνουμε ένα δίκτυο P2P τριών κόμβων, με σκοπό να μελετήσουμε τον κυριότερο παράγοντα που επηρεάζει την ποιότητα υπηρεσιών παρόμοιων δικτύων : το χρόνο αναμονής μέχρι την εξυπηρέτηση. Στις προσομοιώσεις που ακολουθούν γίνεται σύγκριση κατάλληλων τεχνικών, με σκοπό να βρεθεί εκείνη που επιτυγχάνει τα καλύτερα αποτελέσματα, ενώ παράλληλα προτείνονται τρόποι βελτιστοποίησης μέσω κατάλληλης ανάθεσης φόρτου εργασίας και προτεραιοτήτων σε κάθε κόμβο.

Περιεχόμενα

Ευρετήριο εικόνων

Ενότητα

1 Εισαγωγικά

Ορισμός

Ο δικτυακός τόπος Wikipedia(γνωστή δωρεάν εγκυκλοπαίδεια στο διαδίκτυο) εξηγεί τον όρο P2P (peer-to-peer), ως ένα δίκτυο συνδεδεμένων υπολογιστών που δεν διαθέτει σταθερούς πελάτες και εξυπηρετητές, αλλά με τη δυνατότητα κάποιοι από αυτούς τους υπολογιστές να παίξουν και τους δύο ρόλους. Σύμφωνα με το Whatis.com, peer-to-peer είναι ένα μοντέλο επικοινωνίας στο οποίο κάθε μέρος μπορεί να αρχικοποιήσει μια επικοινωνιακή σύνδεση. Το μοντέλο αυτό είναι διαφορετικό από αυτό που έχει συνηθίσει ο κόσμος τα τελευταία χρόνια και το οποίο βασίζεται στην ύπαρξη ενός σταθερού εξυπηρετητή και αποτελεί τον κορμό του Internet όπως το γνωρίζουμε σήμερα. Σε γενικές γραμμές, οι χρήστες του Internet έχουν συνηθίσει να ζητούν κάποια σελίδα, ή άλλο αντικείμενο από κάποιον εξυπηρετητή, ο οποίος λειτουργεί ειδικά για το σκοπό αυτό. Σε ένα P2P δίκτυο, οι χρήστες ζητάνε αντικείμενα ο ένας από τον άλλο, καθιστώντας ουσιαστικά τον προσωπικό τους υπολογιστή, πελάτη και εξυπηρετητή ταυτοχρόνως. Το γεγονός αυτό είναι που κάνει τη μορφή αυτή επικοινωνίας τόσο σημαντική : δεν είναι τόσο ότι επικοινωνούν μεταξύ τους δυο υπολογιστές, όσο το ότι αυτοί είναι συνηθισμένοι προσωπικοί υπολογιστές, που συνήθως χρησιμοποιούνται για ανάγνωση ιστοσελίδων στο Διαδίκτυο και γίνονται άμεσα ενεργά μέλη αυτού. Μια πιο τεχνική ερμηνεία που δίνεται, ορίζει ότι για να ονομαστεί ένα σύστημα peer-to-peer πρέπει να διαθέτει τα παρακάτω χαρακτηριστικά :

- Οι προσωπικοί υπολογιστές των χρηστών λειτουργούν τόσο σαν πελάτες, όσο και σαν εξυπηρετητές.
- Το σύστημα παρέχει συνδέσεις με άλλους χρήστες.
- Το σύστημα παρέχει δυνατότητα εύρεσης κάποιας πληροφορίας που ζητείται.
- Κάθε κόμβος μοιράζεται πληροφορίες που μπορεί να είναι σημαντικές για άλλους χρήστες του δικτύου.

Η βασική ανάγκη που επέβαλλε την εμφάνιση της νέας αυτής μορφής επικοινωνίας, είναι τόσο η αδυναμία του κλασσικού client-server μοντέλου να ικανοποιήσει κάποιες βασικές ανάγκες των χρηστών του Internet, όσο και η εν μέρει μη αξιοποίηση των διαθέσιμων πόρων του δικτύου.

Η βασική έννοια που έκανε εφικτή την ιδέα του P2P είναι η λεγόμενη "distributed computing", ή αλλιώς κατανεμημένος υπολογισμός. Σκοπός είναι η αξιοποίηση των πόρων διασυνδεδεμένων συστημάτων, όπως ακριβώς είναι οι προσωπικοί υπολογιστές των χρηστών του Διαδικτύου. Οι πόροι αυτοί, σε γενικές γραμμές περιλαμβάνουν υπολογιστική (επεξεργαστική) ισχύ, αποθηκευτικό χώρο και bandwidth γραμμής. Τα μέρη ενός κατανεμημένου συστήματος, αποτελούμενου από υπολογιστικά συστήματα ανά τον κόσμο, είναι ανομοιόμορφα και με διαφορετικές δυνατότητες, η σύνδεσή τους, όμως, μπορεί να προσφέρει νέες δυνατότητες. Ιδιαίτερα για την περίπτωση των P2P δικτύων, οι διασυνδεδεμένοι υπολογιστές διαμοιράζονται αποθηκευτικό χώρο και bandwidth γραμμής, πόροι οι οποίοι αθροιστικά είναι ασύγκριτα περισσότεροι από όσους μπορεί να προσφέρει το παραδοσιακό client-server μοντέλο.

Ιστορικά στοιχεία

Η ιστορία των P2P δικτύων ξεκινά από αρκετά χρόνια πριν, όταν το 1988 στο πανεπιστήμιο Oulu της Φινλανδίας (Department of Information Processing Science) χρησιμοποιήθηκε για πρώτη φορά το Internet Relay Chat (IRC). Το IRC αποτελείται από ένα μεγάλο αριθμό κατανεμημένων servers οι οποίοι δρομολογούν μεταξύ τους πληροφορίες "chat", δηλαδή online συζητήσεις χρηστών σε πραγματικό χρόνο. Σύμφωνα με απόψεις άλλων ειδικών, η πρώτη εμφάνιση p2p δικτύου αποτελεί η χρήση του Domain Name System. Το DNS, όπως ονομάζεται, αποτελείται από εξειδικευμένους servers που μεταφράζουν τις ονομασίες των λεγόμενων domains σε πραγματικές IP διευθύνσεις και ανταλλάσσουν τις πληροφορίες αυτές μεταξύ τους όπως οι κόμβοι ενός P2P δικτύου. Παρ' όλ' αυτά η μεγάλη έκρηξη στο χώρο των P2P δικτύων πραγματοποιήθηκε το Μάιο του 1999, όταν το Napster, με δημιουργό τον νεαρό φοιτητή Shawn Fanning, ξεκίνησε την online λειτουργία του. Η μοναδική, εκείνη την εποχή, υπηρεσία ανταλλαγής τραγουδιών σε ψηφιακή μορφή και χωρίς

κανένα κόστος, σε συνδυασμό με την εξάπλωση των broadband συνδέσεων στην Ευρώπη και Αμερική, συνέβαλαν στην τεράστια επιτυχία του εγχειρήματος. Η επιτυχία αυτή, όμως, ήταν που ανησύχησε τις μεγάλες δισκογραφικές εταιρείες, οι οποίες έβλεπαν το αναπτυσσόμενο δίκτυο ως απώλεια κερδών, με αποτέλεσμα τελικά το κλείσιμο του Napster τον Ιούλιο του 2001.

Όπως είναι φανερό, το κενό που άφησε το Napster (εκατομμύρια χρηστών που επιθυμούσαν να αποκτήσουν την αγαπημένη τους μουσική) ήταν τεράστιο και αμέσως ανέλαβαν να καλύψουν τη θέση διάφορα άλλα προγράμματα. Το επόμενο σημαντικό γεγονός που αξίζει αναφοράς, είναι η εμφάνιση του πρώτου πραγματικού P2P δικτύου, με την ονομασία Gnutella. Το Gnutella επινοήθηκε το Μάρτιο του 2000 από τους Justin Frankel και Tom Pepper της εταιρείας Nullsoft. Η προώθηση του από την κατασκευάστρια εταιρεία ματαιώθηκε σχεδόν αμέσως, όχι όμως πριν προλάβουν ορισμένοι να ανακαλύψουν τον κρυφό κώδικα (μέσω reverse engineering) και να τον τροποποιήσουν σύμφωνα με τις ανάγκες τους, κάνοντας παράλληλα τον κώδικα ανοιχτό (open source). Ο ανοιχτός αυτός χαρακτήρας του πρωτοκόλλου, οδήγησε μεγάλο αριθμό ανθρώπων να ασχοληθούν μ' αυτό και να το εξελίξουν σε μεγάλο βαθμό.

Παρά τη μεγάλη αποδοχή από το κοινό, όμως, η τεχνολογία αυτή έπασχε από ορισμένα προβλήματα (τα οποία αναλύονται σε επόμενη παράγραφο). Σ' αυτό το σημείο εμφανίστηκε το FastTrack, το οποίο ήταν μια τεχνολογία με σκοπό την ανάπτυξη P2P δικτύων. Με βάση το FastTrack αναπτύχθηκαν τα δημοφιλέστατα KaZaA (της Sharman Networks) και Morpheus (της MusicCity), τα οποία (με κυριότερο εκφραστή το KaZaA) κυριάρχησαν στο χώρο του file sharing για μεγάλο χρονικό διάστημα. Στις μέρες μας υπάρχει μεγάλος αριθμός από P2P δίκτυα, τα οποία στην πλειοψηφία τους χρησιμοποιούν νέες τεχνολογίες (όπως το BitTorrent δίκτυο) με σκοπό όχι μόνο το μοίρασμα μουσικών αρχείων αλλά και πληθώρα άλλων δραστηριοτήτων.

Χρήσεις

Οι πιθανές χρήσεις των σημερινών P2P δικτύων δεν περιορίζονται μόνο στο μοίρασμα μουσικών αρχείων, παρά την μεγάλη δημοτικότητα της χρήσης αυτής. Η καινοτομία που εισάγουν μπορεί να χρησιμοποιηθεί για πληθώρα εφαρμογών που να καλύπτουν πολλές και διαφορετικές ανάγκες των χρηστών. Παρακάτω αναφέρονται μερικές από τις πιθανές χρήσεις, από τις οποίες οι περισσότερες βρίσκουν ήδη εφαρμογή και απολαμβάνουν την αποδοχή του κοινού, ενώ άλλες βρίσκονται ακόμα στο στάδιο της ανάπτυξης :

- **DistributedComputing**
  Το SETI@Home σύστημα κατανέμει την επεξεργασία των ψηφιακών σημάτων για την αναζήτηση εξωγήινης νοημοσύνης σε χιλιάδες χρήστες του δικτύου. Με τον τρόπο αυτό συνδυάζεται η επεξεργαστική ισχύς μεγάλου αριθμού υπολογιστικών συστημάτων που υπό άλλες συνθήκες θα πήγαινε χαμένη. Αν και δεν είναι ένα P2P σύστημα με την αυστηρή έννοια του όρου, δεδομένου ότι οι επιμέρους χρήστες δεν επικοινωνούν μεταξύ τους, εντούτοις μπορούμε να το κατατάξουμε στην κατηγορία αυτή λόγω του μοιράσματος του φόρτου εργασίας που επιτυγχάνει. Επίσης, με τον ίδιο τρόπο μπορεί να επιτευχθεί το μοίρασμα οποιασδήποτε εργασίας η οποία απαιτεί μεγάλη επεξεργαστική δύναμη και μπορεί να χωριστεί σε κομμάτια που μπορούν να δοθούν προς επεξεργασία παράλληλα.

- **DistributedDownloads**
  Περιλαμβάνει όλα τα δημοφιλή δίκτυα P2P που μελετήσαμε και σε προηγούμενη παράγραφο. Χαρακτηριστικότερο παράδειγμα είναι το BitTorrent. Τυπικά σε ένα client-server μοντέλο, όσο περισσότεροι χρήστες «κατεβάζουν» ένα αρχείο, τόσο μικρότερη ταχύτητα απολαμβάνουν, λόγω της εξάντλησης της χωρητικότητας της γραμμής του server. Με το BitTorrent, δεδομένου ότι κάθε χρήστης που κατεβάζει, γίνεται αυτόματα και server, το συνολικό διαθέσιμο bandwidth αυξάνεται όσο περισσότεροι χρήστες κατεβάζουν ένα αρχείο. Με τον τρόπο αυτό μειώνεται η ανάγκη ύπαρξης ακριβού hadware υψηλών δυνατοτήτων, μειώνοντας τα λειτουργικά κόστη.

- **SearchEngine**
  Μια ακόμα σημαντική καινοτομία που δεν έχει βρει ευρεία εφαρμογή είναι η χρήση P2P δικτύων που να χρησιμοποιούνται σαν μηχανές αναζήτησης. Θα ήταν εξαιρετικά χρήσιμο αν οι χρήστες ενός τέτοιου δικτύου μπορούσαν να ανταλλάξουν σημαντικές πληροφορίες, όπως internet bookmarks ή πηγαίο κώδικα κάποιας γλώσσας προγραμματισμού, ώστε να επιτύχουν γρηγορότερη επίλυση κάποια σημαντικής εργασίας. Ιδιαίτερα σε εταιρικά περιβάλλοντα μια τέτοια προσέγγιση είναι ιδιαίτερα προσοδοφόρα και υιοθετείται κατά κύριο λόγο από μεγάλες εταιρείες.

- **Communication**
  P2P συστήματα μπορούν να χρησιμοποιηθούν και για chat και instant messaging. Στις μέρες μας, αυτές οι δύο δραστηριότητες έχουν βρει τεράστια απήχηση στο κοινό, με εφαρμογές ακόμα και σε παροχή απομακρυσμένης βοήθειας (helpdesk & support) όσο και συνεργασία μεταξύ εταιρικών υπαλλήλων. Τα περισσότερα συστήματα, όμως, χρησιμοποιούν κεντρικούς servers που τους διαχειρίζονται οι αντίστοιχοι service providers. Είναι δυνατόν, εντούτοις, οι ίδιοι οι χρήστες να είναι υπεύθυνοι για την ορθή λειτουργία, χωρίς την παρέμβαση κάποιας central authority.

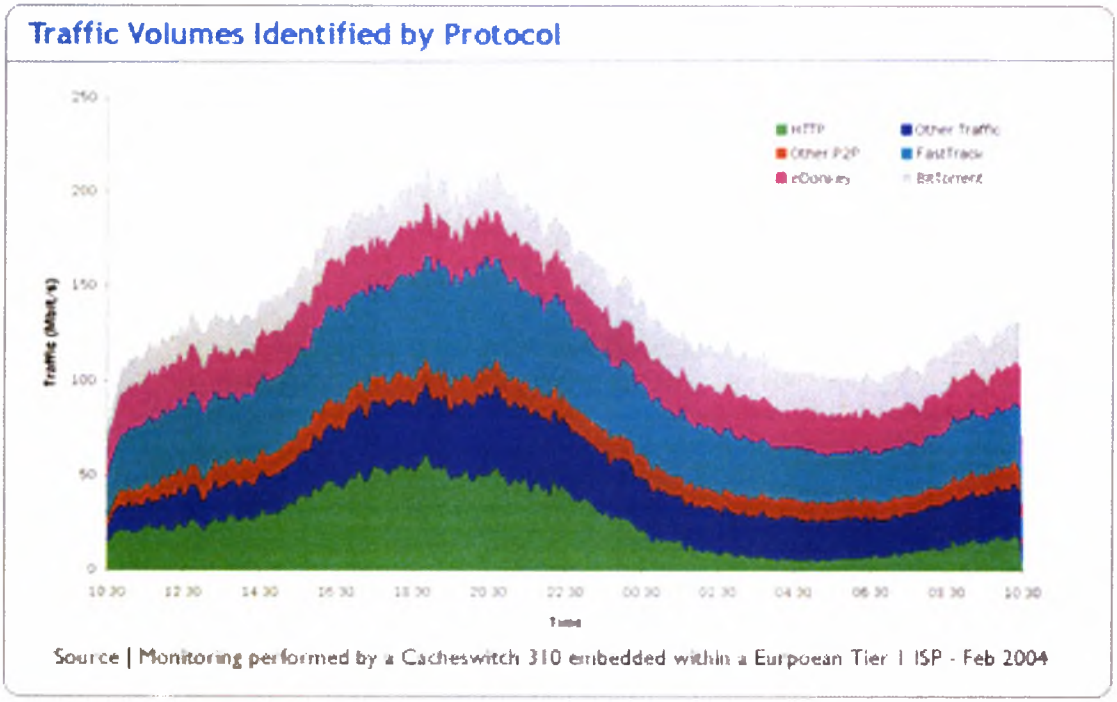
- **Collaboration**
  Με τη χρήση κατάλληλου δικτύου μπορούν δύο απομακρυσμένοι χρήστες να ξεκινήσουν ένα project, να παρακολουθούνε την πρόοδό του, να μοιραστούνε σημαντικά αρχεία, να συνομιλήσουν σε discussion boards, ή ακόμα και να συζητήσουν σε πραγματικό χρόνο μέσω instant messaging.

- **Security**
  Σε περίπτωση που οι κόμβοι π.χ. ενός εταιρικού δικτύου είναι συνδεδεμένοι μεταξύ τους και ανταλλάσσουν πληροφορίες σχετικές με την ασφάλεια, τότε αν κάποιος κόμβος παραβιαστεί, μπορεί να ενημερώσει τους υπόλοιπους, οι οποίοι θα αντιδράσουν για να αποτρέψουν ανεπιθύμητες ενέργειες. Αυτή η αποκεντροποιημένη προσέγγιση προσθέτει σημαντικά στην ασφάλεια του συνολικού δικτύου.

- **ResilientNetworks**
  Γενικώς η χρήση P2P δικτύων σε οποιοδήποτε τομέα, προσθέτει σημαντική διαθεσιμότητα στο δίκτυο, δεδομένου ότι δεν είναι δυνατόν πλέον να καταρρεύσει ένα ολόκληρο δίκτυο λόγω της απώλειας ενός κεντρικού κόμβου. Τα P2P δίκτυα είναι κατά βάση ανθεκτικά σε φυσικές καταστροφές και φθορές.

Σημασία των P2P δικτύων

Τα P2P δίκτυα είναι εξαιρετικά σημαντικά στις μέρες μας, κυρίως για το λόγο ότι απολαμβάνουν την καθολική αποδοχή από τους χρήστες του διαδικτύου. Χαρακτηριστικό είναι ότι στο τέλος του έτους 2004, το BitTorrent ήταν υπεύθυνο για το 30% της συνολικής κίνησης στο Internet (με συνολικό ποσοστό για τα P2P δίκτυα να αγγίζει το 60%), ενώ ανάλογα ποσοστά έχουν κατά καιρούς αναφερθεί και για άλλα δημοφιλή P2P δίκτυα, με πρωταγωνιστές τα eDonkey και KaZaA. Στην παρακάτω γραφική παράσταση φαίνεται η τυπική κίνηση σε ένα μεγάλο ευρωπαϊκό internet service provider κατά τη διάρκεια ενός εικοσιτετραώρου :



Εικόνα 1 Κατανομή internet κίνησης σε γνωστό ευρωπαϊκό ISP

Φαίνεται ξεκάθαρα, ότι η κίνηση που οφείλεται σε P2P δίκτυα καταλαμβάνει το μεγαλύτερο ποσοστό της συνολικής κίνησης, ενώ η http κίνηση είναι η μικρότερη απ' όλες καθ' όλη τη διάρκεια της ημέρας. Είναι επίσης ξεκάθαρο ότι το BitTorrent είναι ο κυρίαρχος από πλευράς χρήσης του bandwidth, ακολουθούμενο από το eDonkey και τα δίκτυα βασισμένα στο FastTrack. Ένα σημαντικό γεγονός είναι επίσης και το ότι μεγάλες εταιρείες, όπως η Microsoft, έχουν αναγνωρίσει τις δυνατότητες των P2P δικτύων και ξοδεύουν όλο και μεγαλύτερα χρηματικά ποσά για την έρευνα στον τομέα αυτό. Πολλές εταιρείες κατασκευής Unix λειτουργικών συστημάτων χρησιμοποιούν P2P προγράμματα για να διανείμουν τις δωρεάν εκδόσεις τους σε όσο

το δυνατόν μεγαλύτερο κοινό, ελαχιστοποιώντας παράλληλα το κόστος διανομής. Μπορεί εύκολα κανείς να αναλογιστεί με βάση τα παραπάνω δεδομένα, τις ανεκμετάλλευτες δυνατότητες που ανοίγονται από τη χρήση P2P δικτύων. Αν συνδυαστούν τα παραπάνω με το γεγονός ότι η εξέλιξη είναι βρίσκεται ακόμα σε πρώιμα στάδια και δεν έχει επιτευχθεί βελτιστοποίηση στον τομέα αυτό, φαίνεται ξεκάθαρα ότι το μέλλον του P2P διαγράφεται εξαιρετικά ευοίωνο.

Ενότητα 2

Δημοφιλή P2P Συστήματα

- Gnutella
- Freenet
- Freehaven
- OpenNap
- Mojo Nation
- Publius, ATT Research and NYU
- Chord, MIT LCS.
- Ohaha, open sourced similar to Chord.
- Tapestry, U.C. Berkeley
- OceanStore , U.C Berkeley
- PAST, Microsoft Research
- Pastry, Microsoft Research, and Rice
- Farsite, Microsoft Research.
- Piazza, Gribble at UWash.
- OpenCola's Swarmcast
- BitTorrent
- ALPINE: Adaptive Large-scale Peer2peer Information NEtworking (at SourceForge)

List of Popular P2P Programs

Implementation	Protocol	Type	Major Use	Licensing Model	Revenue Model	Tech Details	URL
BCDC++	Direct Connect	*Hybrid, hubs and supernodes*	File Sharing	??	??	C++	*
BearShare	*Gnutella*	*Hybrid, hubs*	*File Sharing*	Closed Source	*Advertising*	Windows Only	*
BitTorrent	BItTorrent	Decentral-ized, mixed	File Sharing,	Open Source, *	Voluntary - donations,	Python with	*

		roles	RSS Feed sharing	MIT License	some advertising ?	xWindos client.	
DC++	Direct Connect	Hybrid, hubs and supernodes	File Sharing	Open Source, * GNU GPL	Voluntary	C++, Client only	*
eDonkey	eDonkey	Decentralized	File Sharing	Closed Source	Voluntary, not for profit ??		*
eMule	eDonkey	Decentralized	File Sharing	Closed Source	Voluntary, not for profit	Windows only	*
Freenet	Freenet	Hybrid, nodes	Information dispersal, Anonymous	Open Source, * GNU GPL	Voluntary	Java	*
GnouGat	JXTA	Decentralized	FileSharing	Open Source, * Sun Project JXTA Software License	Voluntary	Java only (JINI)	*
GnuNet-GTK	GnuNet	Decentralized, anonymous	File Sharing	Open Source, * GNU GPL	Voluntary	Source only, UNIX GTK, HTTP, SMTP, UDP over TCP-IP	*
Grokster	FastTrack	Hybrid	File Sharing	Closed Source	Advertising, Content distribution fee	Windows Only	*
Groove	Groove	Decentralized	Groupware	Closed	Fee Based	Windows	*

				Source		Only	
iMesh	FastTrack	Decentral-ized	File Sharing	Closed Source , freeware	Advertising	Windows Client	*
KaZaa	FastTrack	Hybrid	File Sharing	Closed source	Advertising, Spyware, Malware		*
Morpheus	FastTrack, Gnutela2, Direct Connect	Hybrid	File Sharing	Closed Source	Advertising	Windows only	*
Napster	Napster	Napster is now defunct as a P2P network.					*
NeoModus	Direct Connect	Hybrid, hubs and supernodes	File Sharing	Closed Source	Advertising	C++, C#	*
OverNet	Overnet, eDonkey	Decentral-ized	File Sharing (large files)	Closed Source	Voluntary, some advertising ??	Windows client only	*
Piolet / Blubster	Manolito, MP2P	Decentral-ized	File Sharing, MP3 only	Closed Source	??	Windows Only	*
Rebol	Rebol IOS	Decentralized	Internet Operating System	Closed Source	Commercial	Over 40 OSs	*
ShareAza	BitTorrent, eDonkey, Gnutella, Gnutella2	Hybrid	File Sharing	Closed Source, freeware	Advertising	Windows Client	*
Soulseek	Soulseek	Centralized	MP3 File Sharing	Closed Source	Contribution based		*
WinMX	OpenNap, WPNP	Hybrid	File Sharing	?	?	Windows Client	*

Στις μέρες μας, το μοίρασμα αρχείων μεταξύ των χρηστών (file sharing) έχει αναδειχθεί σε μια από τις πιο δημοφιλείς δραστηριότητες στο διαδίκτυο. Η σημαντική αύξηση των broadband συνδέσεων και η έλευση του Napster το 1999, έδωσαν μια μεγάλη ώθηση στο φαινόμενο του file sharing και παρά τους περιορισμούς που επιβάλλουν οι διεθνείς οργανισμοί συνεχίζει να αναπτύσσεται και να βελτιώνεται.

Ιδιαιτερότητες της P2P κίνησης

Η P2P κίνηση έχει κάποια χαρακτηριστικά που την ξεχωρίζουν από την κλασσική http κίνηση. Κατά πρώτο λόγο τα μεγέθη των αρχείων είναι πολύ μεγαλύτερα και κυμαίνονται από 3.5 Mb για ένα κοινό αρχείο ήχου mp3 έως και περισσότερα από 2Gb για μια εικόνα (image) ενός DVD. Οι χρήστες συνήθως ανοίγουν πληθώρα συνδέσεων οι οποίες διαρκούν σε πολλές περιπτώσεις ακόμα και μέρες ολόκληρες. Τα δεδομένα που διακινούνται σε γενικές γραμμές μεταβάλλονται σύμφωνα με τις τάσεις της εποχής, δεδομένου ότι συνεχώς παράγεται νέο οπτικοακουστικό υλικό (μουσική, ταινίες κλπ) και οι χρήστες επιθυμούν να είναι ενημερωμένοι με τις τελευταίες κυκλοφορίες. Τέλος χρησιμοποιείται μεγάλος αριθμός από διαφορετικές network ports, γεγονός που διευκολύνει τον έλεγχο για τη διακίνηση υλικού μέσω P2P, παρ' όλο που είναι δυνατόν να παρακαμφθεί και αυτός ο έλεγχος στέλνοντας την κίνηση μέσω γνωστών ports (π.χ. port 80 για http κίνηση).

Οι βασικές διαφορές σε σύγκριση με συνηθισμένη http κίνηση συνοψίζονται στον παρακάτω πίνακα :

HTTP	P2P
Τα μεγέθη των αρχείων είναι συνήθως μικρά και περιλαμβάνουν μικρά scripts (html, JavaScript, css κλπ) και γραφικά (εικόνες, flash objects) τα οποία δεν ξεπερνούν τα 250 KB.	Τα αρχεία έχουν μεγέθη που ξεκινούν από 2-3 MB για ένα κοινό αρχείο ήχου mp3 και φτάνουν σε μεγέθη μεγαλύτερα των 2-3 GB για ταινίες MPEG4 και images λογισμικού.

Τα δεδομένα μεταφέρονται όλα σε ένα και μοναδικό session.	Ο χρήστης (ή το υπεύθυνο πρόγραμμα) ανοίγουν πολλαπλές συνδέσεις για ταυτόχρονο κατέβασμα των πληροφοριών.
Οι συνδέσεις διαρκούν μερικά δευτερόλεπτα.	Οι συνδέσεις συνήθως παραμένουν ανοικτές για πολλές ώρες, ή ακόμα και μέρες.
Τα δεδομένα είναι πάντα διαθέσιμα και αποθηκευμένα σε ισχυρούς servers με γραμμές μεγάλης χωρητικότητας.	Οι συνδέσεις δεν είναι αξιόπιστες και η διαθεσιμότητα είναι περιορισμένη. Η χωρητικότητα των γραμμών των χρηστών κυμαίνεται από 56kbps μέχρι τυπικά 2mbps.
Οι πληροφορίες που είναι αποθηκευμένες στους servers αλλάζουν με σχετικά βραδύ ρυθμό (με εξαιρέσεις πάντοτε, όπως για παράδειγμα τα news sites, που προσφέρουν συνεχή πληροφόρηση πάνω στις τρέχουσες εξελίξεις).	Δεδομένου ότι οι χρήστες επιθυμούν να κατέχουν ό,τι πιο καινούριο κυκλοφορεί, οι πληροφορίες που διακινούνται μεταβάλλονται με γοργούς ρυθμούς.
Το πρωτόκολλο http είναι αυστηρώς καθορισμένο και δεν αμετάβλητο.	Η ύπαρξη πολλών διαφορετικών P2P δικτύων οφείλεται στην ύπαρξη διαφορετικών πρωτοκόλλων και υλοποιήσεων, άλλων ανοιχτών στο ευρύ κοινό (open source) αλλά και άλλων κλειστών.
Τα δεδομένα διακινούνται δια μέσου γνωστών και προκαθορισμένων θυρών (ports) όπως π.χ. η θύρα 80.	Κάθε υλοποίηση χρησιμοποιεί δική της θύρα, με μερικές από αυτές να προσφέρουν τη δυνατότητα αλλαγής της θύρας (ακόμα και δρομολόγηση μέσω της θύρας 80 ώστε να μην ανιχνεύεται η P2P κίνηση).
Οι τοποθεσίες που είναι αποθηκευμένες οι πληροφορίες είναι σταθερές (αν και συνήθως υπάρχουν mirror sites τα οποία	Οι πληροφορίες είναι διάσπαρτες ανάμεσα στους χρήστες του δικτύου και η αναγνώριση των πληροφοριών γίνεται

και αυτά με τη σειρά τους έχουν σταθερά URL).	είτε μέσω λέξεων-κλειδιών (keywords) είτε μέσω συναρτήσεων κατακερματισμού (hash functions).
Η ροή της κίνησης είναι ασύμμετρη, δηλαδή ο χρήστης κάνει μια μικρή αίτηση και παραλαμβάνει μεγάλη ποσότητα δεδομένων σαν απάντηση.	Η ροή είναι κατά κάποιο τρόπο συμμετρική, δεδομένου ότι ένας χρήστης θα «κατεβάσει» δεδομένα, αλλά θα «ανεβάσει» επίσης σε άλλους χρήστες.

Κλάσεις διαχωρισμού αρχιτεκτονικών P2P

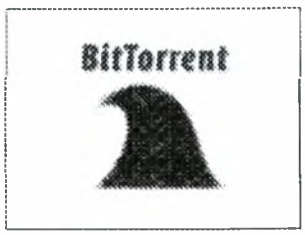
Οι P2P αρχιτεκτονικές μπορούνε να διαχωριστούνε σε δύο βασικές κλάσεις. Η πρώτη ονομάζεται "Pure P2P architecture" και βασικό της χαρακτηριστικό είναι η παντελής έλλειψη κεντρικού server. Η άλλη ονομάζεται "server-mediated P2P architecture" και διαθέτει έναν (ή περισσότερους) κεντρικούς εξυπηρετητές, οι οποίοι καταχωρούν κοινόχρηστες πληροφορίες και απαντούν στις αναζητήσεις των χρηστών.

Η "pure P2P architecture" αποτελείται από κόμβους που έχουν παρόμοια χαρακτηριστικά, χωρίς κάποιος από αυτούς να έχει ξεχωριστές αρμοδιότητες. Ανακαλύπτει δυναμικά νέους κόμβους και αλληλεπιδρά μαζί τους ανταλλάσσοντας μηνύματα. Το βασικότερο πλεονέκτημα της αρχιτεκτονικής αυτής, είναι ότι η διαθεσιμότητα της υπηρεσίας δεν βασίζεται σε κάποιον κεντρικό εξυπηρετητή, μια βλάβη του οποίου θα ήταν αρκετή για να καταρρεύσει ολόκληρο το σύστημα. Από την άλλη, όμως, δεν επιτρέπει την εύρεση όλων των υπαρκτών κόμβων του δικτύου περιορίζοντας κατά κάποιο τρόπο τα αποτελέσματα μιας αναζήτησης, ενώ τέλος πλημμυρίζει το δίκτυο με άχρηστη πληροφορία που αναμεταδίδεται κατά την εκτέλεση ερωτημάτων.

Η "server-mediated" αρχιτεκτονική διαθέτει κεντρικό εξυπηρετητή για την εύρεση χρηστών και εκτέλεση ερωτημάτων. Κατά την εισαγωγή ενός νέου κόμβου στο σύστημα, ενημερώνεται ο κεντρικός server τόσο για την ύπαρξη του νέου κόμβου όσο και για τους πόρους που διαμοιράζεται αυτός. Με τον τρόπο αυτό, κάποιος άλλος κόμβος (χρήστης) αποστέλλει το ερώτημά του στον κεντρικό server αντί για κάθε κόμβο ξεχωριστά και λαμβάνει απάντηση για το ποιος διαθέτει το ζητούμενο.

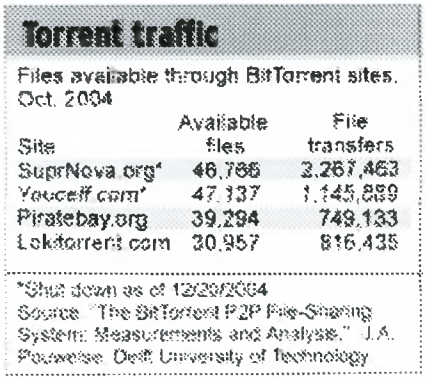
Κυριότερο πλεονέκτημα της αρχιτεκτονικής είναι η ευκολία με την οποία αναπτύσσεται το δίκτυο εισάγοντας νέους κόμβους. Αρκεί ένα μήνυμα από τον νεοεισερχόμενο κόμβο και γίνεται άμεσα γνωστή η ύπαρξή του στους υπόλοιπους. Από την άλλη μεριά, όπως προαναφέρθηκε, μι τέτοια υλοποίηση πάσχει από την εξάρτηση της υπηρεσίας από τη διαθεσιμότητα του κεντρικού εξυπηρετητή. Αν δεν είναι διαθέσιμος, χάνεται κάθε δυνατότητα για ανεύρεση χρηστών και πόρων με συνέπεια την κατάρρευση ολόκληρου του συστήματος.

BitTorrent



Το BitTorrent είναι ένα σύστημα διαμοιρασμού αρχείων το οποίο αναπτύχθηκε από τον Bram Cohen το έτος 2002. Στο σύστημα αυτό, τα αρχεία χωρίζονται σε κομμάτια (της τάξης των χιλιάδων ανά αρχείο) και οι χρήστες της υπηρεσίας διακινούν τα κομμάτια αυτά μεταξύ τους. Όταν ένας κόμβος κατεβάσει ένα κομμάτι του αρχείου, αυτομάτως παρέχει το κομμάτι αυτό σε όσους επιθυμούν να το κατεβάσουν. Με τον τρόπο αυτό, το σύστημα χρησιμοποιεί το upload του χρήστη που κατεβάζει ένα αρχείο και ελαφρώνει το έργο του αρχικού κόμβου από τον οποίο ζητήθηκε το αρχείο. Τα κομμάτια παραλαμβάνονται από τον κόμβο που τα ζήτησε με τυχαία σειρά και συνενώνονται για τη δημιουργία του ζητούμενου αρχείου. Κάθε κόμβος προσπαθεί να μεγιστοποιήσει τη δικιά του ταχύτητα με την οποία κατεβάζει το αρχείο, επικοινωνώντας με τους κατάλληλους κόμβους που προσφέρουν τα κομμάτια που λείπουν με τη μέγιστη ταχύτητα.

Η αναζήτηση στο BitTorrent πραγματοποιείται μέσω δικτυακών τόπων στους οποίους καταχωρούνται τα διαθέσιμα αρχεία. Μερικοί από τους πιο γνωστούς τόπους είναι και αυτοί που φαίνονται στον παρακάτω πίνακα :



Torrent traffic

Files available through BitTorrent sites. Oct. 2004

Site	Available files	File transfers
SuprNova.org*	46,766	3,267,463
Yoyoself.com*	47,137	1,145,889
Piratebay.org	39,294	749,133
Lokotorrent.com	30,957	816,435

*Shut down as of 12/20/2004
Source: "The BitTorrent P2P File-Sharing System: Measurements and Analysis," J.A. Pouwelse, Delft University of Technology

Εικόνα 2 Γνωστοί δικτυακοί τόποι που διαχειρίζονται torrent αρχεία

Στους παραπάνω τόπους οι χρήστες μπορούν να βρούνε πληροφορίες για το όνομα και το μέγεθος του αρχείου που επιθυμούν, καθώς και πληροφορίες για το πόσοι χρήστες το διαθέτουν και πόσοι το κατεβάζουν τη στιγμή εκείνη. Για να ξεκινήσει η διαδικασία απόκτησης, οι χρήστες κατεβάζουν ένα αρχείο τύπου .torrent που αντιστοιχεί στο ζητούμενο αρχείο. Το .torrent περιέχει πληροφορίες για το ζητούμενο αρχείο, είναι δηλαδή ένα meta-data file. Τα .torrent δεν βρίσκονται στους δικτυακούς τόπους που αναφέραμε, αλλά σε εξειδικευμένους servers που φιλοξενούν τα αρχεία αυτά. Τα .torrent με τη σειρά τους επικοινωνούν με ένα πρόγραμμα που ονομάζεται tracker και το οποίο διατηρεί κατάλογο χρηστών που κατέχουν το αρχείο (seeders) και αυτών που το κατεβάζουν εκείνη τη χρονική στιγμή(leechers). Ο tracker μπορεί ανά πάσα στιγμή να δώσει στο χρήστη μια λίστα από κόμβους που διαθέτουν κομμάτια του ζητούμενου αρχείου για κατέβασμα. Γνωρίζοντας αυτή την πληροφορία, το λογισμικό του χρήστη ξεκινάει να κατεβάσει το κομμάτι εκείνο που είναι πιο δυσεύρετο (ανάμεσα σε όσα δεν διαθέτει ήδη), κάνοντας με τον τρόπο αυτό τα κομμάτια αυτά πιο προσβάσιμα. Για το λόγο αυτό, η υψηλή ζήτηση ενός συγκεκριμένου αρχείου, όχι μόνο δεν οδηγεί σε μειωμένη απόδοση του συστήματος, αλλά αντίθετα βελτιώνει το συνολικό throughput, αφού αυξάνεται το συνολικό bandwidth με την αύξηση των seeders για κάθε κομμάτι ξεχωριστά.

Το κυριότερο προτέρημα που διαχωρίζει το BitTorrent από τα υπόλοιπα P2P συστήματα είναι το γεγονός ότι διατηρεί μια λίστα των κόμβων που μοιράζονται ένα αρχείο (seed) και αυτών που το κατεβάζουν (leech), παρακολουθώντας ενεργά το δίκτυο. Για να γίνει πιο εμφανές το πλεονέκτημα της μεθόδου αυτής, θεωρούμε το ακόλουθο τυπικό παράδειγμα : Έστω ότι αρχικά ένας χρήστης μοιράζεται ένα νέο αρχείο. Οι χρήστες που επιθυμούν να το κατεβάσουν, αρχικά ανακτούν το .torrent αρχείο από το δικτυακό τόπο που το παρέχει και το ανοίγουν στο παράθυρο της

εφαρμογής. Το .torrent περιέχει πληροφορίες, όπως το όνομα, μέγεθος και hash του αρχείου αλλά και την IP διεύθυνση του tracker. Το πρόγραμμα έπειτα συνδέεται με τον αντίστοιχο tracker και ξεκινάει το κατέβασμα των κομματιών που συνθέτουν το αρχείο. Όσο περισσότεροι χρήστες διαθέτουν κομμάτια του αρχικού αρχείου στην κατοχή τους, ανταλλάσσουν μεταξύ τους τα κομμάτια αυτά, απαλλάσσοντας ουσιαστικά τον αρχικό κόμβο (seeder) από το να προσφέρει τα ίδια κομμάτια σε όλους τους χρήστες ξεχωριστά και προφανώς ελαχιστοποιώντας την απαίτηση για bandwidth από αυτόν. Όταν κάποιος από τους leechers κατεβάσει ολόκληρο το αρχείο, αυτομάτως μετατρέπεται σε seeder και συνεχίζει να μοιράζεται το αρχείο, χωρίς όμως αυτή τη φορά να το κατεβάζει ταυτοχρόνως. Η μέθοδος αυτή, σε αντίθεση με το κλασσικό μοντέλο client/server, απαιτεί σημαντικά λιγότερο bandwidth από τον αρχικό seeder, καθιστώντας το BitTorrent ιδιαίτερα αποτελεσματικό σε περιβάλλοντα με μεγάλο αριθμό χρηστών και συχνή εισαγωγή νέου υλικού..

Napster

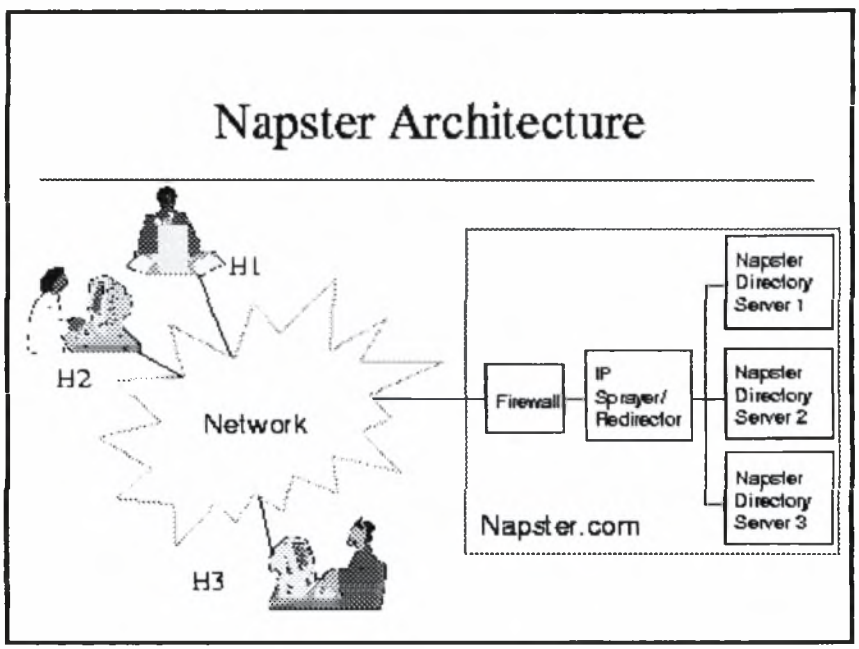
Το αρχικό Napster έγινε ιδιαίτερα δημοφιλές επειδή προσέφερε ένα μοναδικό προϊόν – δωρεάν μουσικά κομμάτια που μπορούσε να αποκτήσει κανείς από μια γιγαντιαία βάση δεδομένων. Δεν υπήρχε πλέον η ανάγκη για απόκτηση ενός άλμπουμ από το τοπικό δισκοπωλείο και το κυριότερο δε χρειαζόταν η αγορά του.

Κατά την εκκίνηση του λογισμικού του Napster για το κατέβασμα ενός τραγουδιού συνέβαιναν τα παρακάτω :

1. Το λογισμικό συνδεόταν με τους κεντρικούς servers, οι οποίοι καταχωρούσαν το χρήστη καθώς και μια λίστα με τα αρχεία που μοιραζόταν, στη βάση δεδομένων των online χρηστών.
2. Ο υπολογιστής του χρήστη μετατρεπόταν σε ένα μικρό server, ικανό να μεταφέρει αρχεία σε άλλους χρήστες που επιθυμούσαν κάποιο από τα διαμοιραζόμενα αρχεία.

3. Ο χρήστης εκτελούσε μια αναζήτηση δίνοντας κάποιες λέξεις-κλειδιά σε ειδικό παράθυρο του λογισμικού και η ερώτησή του (query) προωθούνταν στους κεντρικούς servers προς επεξεργασία.
4. Οι servers επεξεργάζονταν την ερώτηση και συνέτασσαν μια λίστα με τους χρήστες που διέθεταν το ζητούμενο αρχείο, την οποία και απέστειλαν στο χρήστη που τη ζήτησε.
5. Ο χρήστης επέλεγε την έκδοση του αρχείου που επιθυμούσε από τη λίστα
6. Ο υπολογιστής του χρήστη συνδεόταν με εκείνου που κατείχε το αρχείο και εκκινούσε η διαδικασία απόκτησης-κατεβάσματος του αρχείου.

Η αρχιτεκτονική του Napster βασίζεται σε υβριδικό P2P σύστημα, λόγω της ύπαρξης κεντρικού εξυπηρετητή. Η αποθήκευση των αρχείων γίνεται στους κόμβους, ενώ κάθε κόμβος αποστέλλει πληροφορίες για τα αρχεία που μοιράζεται στον κεντρικό server. Ο server αυτός διατηρεί πληροφορίες, όπως το ποιοι χρήστες είναι συνδεδεμένοι (online) και ποια αρχεία έχουν διαθέσιμα. Όταν ένας νέος χρήστης συνδεθεί στο δίκτυο, δηλώνεται στον κεντρικό server σαν ενεργός (online) και αποθηκεύεται μια λίστα των αρχείων που μοιράζεται.



Εικόνα 3 Βασική αρχιτεκτονική του Napster δικτύου

Στη συνέχεια οι ενεργοί χρήστες είναι πιθανό να ζητήσουν κάποιο αρχείο, εκτελώντας ένα query στον κεντρικό server. Ο server κάνει την αναζήτηση μέσα στη

βάση δεδομένων των αρχείων που διαθέτουν οι ενεργοί χρήστες και επιστρέφει κάποια αποτελέσματα στον χρήστη που τα ζήτησε, απεικονίζοντάς τα σε ειδικό παράθυρο του προγράμματος. Ο χρήστης έπειτα έχει τη δυνατότητα να υπολογίσει τον round trip time κάνοντας ping όλους τους κόμβους που έχουν το ζητούμενο αρχείο και να επιλέξει αυτόν με το μικρότερο χρόνο(ή όποιον άλλο επιθυμεί). Με τον τρόπο αυτό αρχικοποιείται η απευθείας σύνδεση ανάμεσα στους χρήστες, χωρίς φυσικά την παρεμβολή του κεντρικού server.

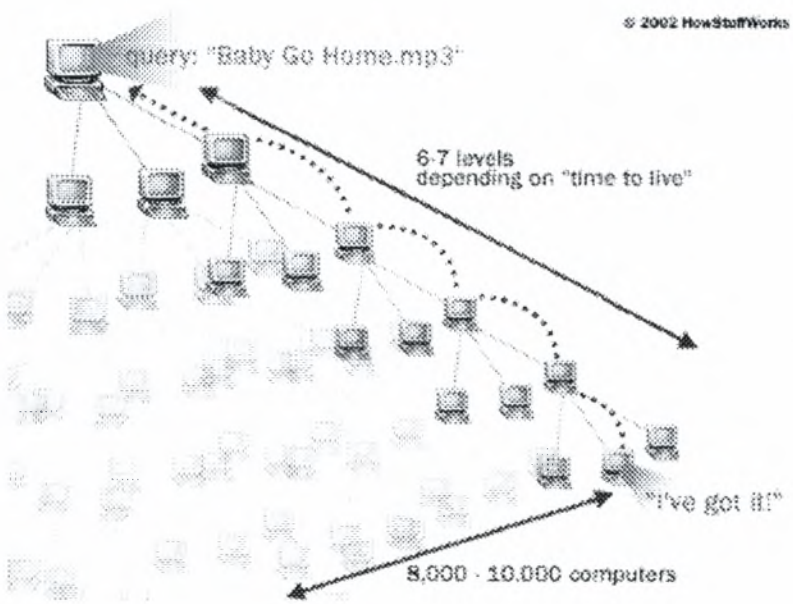
Gnutella

Το δίκτυο γνωστό ως Gnutella αποτελείται από έναν διαρκώς μεταβαλλόμενο αριθμό από κόμβους, οι οποίοι επικοινωνούν μεταξύ τους χρησιμοποιώντας το πρωτόκολλο TCP/IP. Κάθε κόμβος μπορεί να λειτουργήσει ως client ή server, όπως άλλωστε και σε κάθε άλλο P2P δίκτυο υπολογιστών. Όταν ένας νέος κόμβος αρχικοποιείται, προσπαθεί να επικοινωνήσει με παρακείμενους κόμβους ώστε να δηλώσει την παρουσία του και να εισαχθεί στο δίκτυο. Υπάρχουν 5 είδη μηνυμάτων τα οποία ανταλλάσσονται ανάμεσα στους κόμβους κατά τη διάρκεια οποιασδήποτε μορφής επικοινωνίας :

- PING messages : Ο χρήστης δηλώνει την παρουσία του και ζητά από τους παρακείμενους κόμβους να απαντήσουν.
- PONG messages : Η απάντηση σε PING μήνυμα. Ο κόμβος που απαντά δίνει πληροφορίες, όπως την IP διεύθυνσή του, τον αριθμό των αρχείων που μοιράζεται κλπ.
- QUERY messages : αίτηση αναζήτησης αρχείων που πληρούν καθορισμένες προϋποθέσεις.
- HITS messages : η απάντηση στα QUERY μηνύματα, περιλαμβάνουν πληροφορίες για τα αρχεία που πληρούν τις προϋποθέσεις που τέθηκαν, καθώς και για τους χρήστες που τα κατέχουν, όπως IP address.
- GET/PUSH messages : τα μηνύματα που αρχικοποιούν την επικοινωνία ανάμεσα στους χρήστες που πραγματοποιούν μια συναλλαγή.

Σύνδεση : Κάθε χρήστης αρχικά προσπαθεί να συνδεθεί με προκαθορισμένους κόμβους οι οποίοι είναι πιθανότερο να είναι διαθέσιμοι. Η λίστα με τους κόμβους αυτούς είναι αποθηκευμένη μέσα στο λογισμικό της εφαρμογής που χρησιμοποιεί το Gnutella δίκτυο και μπορεί να αλλάξει σύμφωνα με τις εκάστοτε ανάγκες. Γνωρίζοντας, λοιπόν, κάποιους αρχικούς διαθέσιμους κόμβους, ο χρήστης αποστέλλει μηνύματα ping σ' αυτούς, τα οποία με τη σειρά τους προωθούνται σε ολόκληρο το δίκτυο. Με τον τρόπο αυτό γνωρίζουν όλοι οι peers την ύπαρξη του νέου κόμβου, καθώς και τα αρχεία που αυτός μοιράζεται. Για να γνωστοποιήσουν και τη δικιά τους παρουσία, απαντούν στο ping μήνυμα με ένα pong, δηλώνοντας επίσης και τα αρχεία που μοιράζονται. Τελικά ο νέος χρήστης έχει γνωστοποιήσει την παρουσία του στο δίκτυο και γνωρίζει τους άλλους κόμβους που το συνθέτουν.

Ερωτήσεις : Η πραγματοποίηση των ερωτήσεων γίνεται με παρόμοιο τρόπο, όπως και η σύνδεση στο δίκτυο. Ο κόμβος εκτελεί ένα ερώτημα (QUERY) στους γνωστούς του κόμβους, οι οποίοι με τη σειρά τους το προωθούν στους δικούς τους γνωστούς, με αποτέλεσμα το μήνυμα να προωθείται με διαδοχικά hops σε ολόκληρο σχεδόν το δίκτυο. Ο αριθμός των hops καθορίζεται από μια μεταβλητή TTL (Time To Live) που περιέχεται στο μήνυμα. Ενδεικτικά αναφέρουμε πως αν κάθε κόμβος γνωρίζει άλλους τέσσερις και πραγματοποιηθούν 7 hops, τότε το μήνυμα θα φτάσει σε περίπου 8000 υπολογιστές (δεδομένου ότι πολλοί κόμβοι θα λάβουν το ίδιο μήνυμα από διαφορετικά μονοπάτια). Όσοι διαθέτουν το αρχείο που ζητήθηκε απαντούν, στέλνοντας μηνύματα HITS και δηλώνοντας τη διεύθυνση IP τους και το URL του αρχείου.



Εικόνα 4 Πραγματοποίηση ερώτησης στο δίκτυο Gnutella

Λήψη : Κάθε κόμβος που επιθυμεί να «κατεβάσει» ένα αρχείο, ανοίγει μια HTTP σύνδεση με τον κόμβο που το διαθέτει και στέλνει ένα "GET URL" μήνυμα. Κάθε κόμβος επομένως, δεδομένου ότι μπορεί ανά πάσα στιγμή να λειτουργήσει ως server, απαιτείται να υλοποιεί έναν στοιχειώδη web server. Η λειτουργικότητα αυτή παρέχεται φυσικά από το λογισμικό του προγράμματος.

Το βασικότερο πλεονέκτημα της αρχιτεκτονικής είναι ότι η υπηρεσία δουλεύει αδιάλειπτα. Αρκεί να υπάρχει έστω και ένας υπολογιστής που να τρέχει το κατάλληλο λογισμικό, είναι δυνατόν να γίνουν αναζητήσεις και ερωτήσεις. Σε αντίθεση επομένως, με τη λειτουργία του Napster που βασίζεται στη διαθεσιμότητα του κεντρικού εξυπηρετητή, το Gnutella είναι ικανό να λειτουργήσει μόνο με την ύπαρξη χρηστών στο δίκτυο. Από την άλλη μεριά, όμως, η επιλογή της αρχιτεκτονικής αυτής εισάγει και κάποια μειονεκτήματα ως προς τη λειτουργία του δικτύου. Αν υποθέσουμε ότι μια ερώτηση μπορεί να προωθηθεί σε περίπου 8000 κόμβους του δικτύου, δεν υπάρχει απολύτως καμία εγγύηση ότι θα βρεθεί κάποιο αποτέλεσμα σ' αυτούς, παρ' όλο που το ζητούμενο αρχείο υπάρχει κάπου στο δίκτυο. Επίσης, ένα ακόμα σημαντικό μειονέκτημα, είναι και το γεγονός ότι οι ερωτήσεις παίρνουν αρκετό χρόνο μέχρι να εμφανίσουν τα αποτελέσματα στην οθόνη του χρήστη (φτάνουν ακόμα και ένα λεπτό), δεδομένου ότι πρέπει να προσπελαστούν κόμβοι που βρίσκονται ακόμα και 7 επίπεδα μακριά από τον αρχικό. Τέλος, ένα ακόμα σημαντικό πρόβλημα είναι ότι κάθε κόμβος δέχεται συνεχώς μηνύματα ερωτήσεων (από αναζητήσεις άλλων χρηστών για αρχεία), είτε ο χρήστης κατέχει το ζητούμενο αρχείο είτε όχι και μάλιστα πολλές φορές αναμεταδίδει τα μηνύματα αυτά στο επόμενο επίπεδο (flooding). Η πρακτική αυτή χρησιμοποιεί ένα υπολογίσιμο μέρος του bandwidth του δικτύου και φυσικά μέρος του bandwidth κάθε χρήστη ξεχωριστά.

Διαφορές Napster – Gnutella

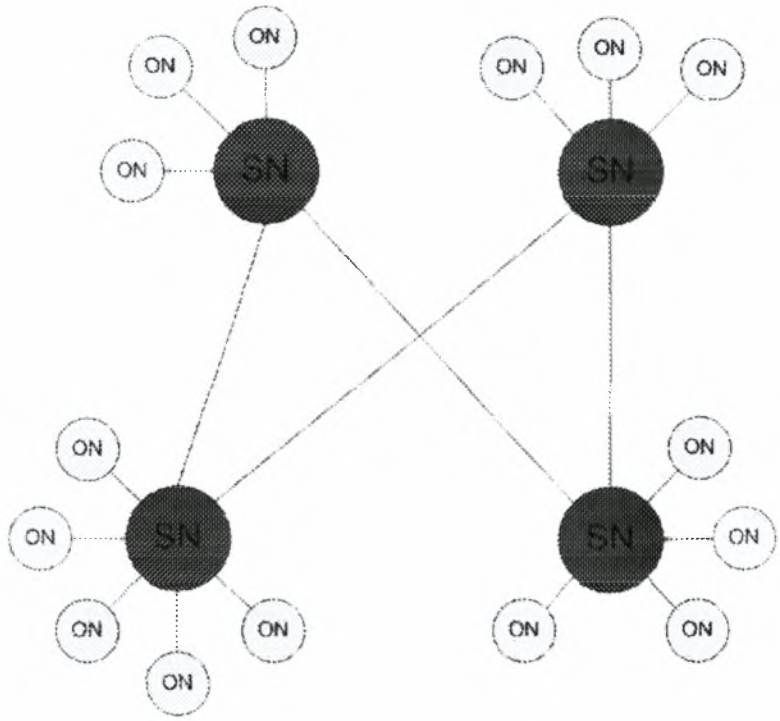
Η βασική διαφορά ανάμεσα στα δύο αυτά P2P δίκτυα έγκειται στην ύπαρξη του κεντρικού server. Το Napster χρησιμοποιεί έναν κεντρικό server για την διαχείριση των χρηστών και των αναζητήσεών τους. Από την άλλη μεριά, το Gnutella είναι πλήρως αυτόνομο δεδομένου ότι λείπει η κεντρική διαχείριση των κόμβων και κάθε χρήστης λειτουργεί σαν client και server ταυτόχρονα.

Το Gnutella σε αντίθεση με το Napster δεν διαθέτει εξειδικευμένο λογισμικό για το δίκτυό του, αλλά μια πληθώρα από αυτά, με κυριότερους εκπροσώπους τα παρακάτω:

- BearShare
- Gnucleus
- LimeWire
- Morpheus
- WinMX
- XoloX

KaZaA

Το δίκτυο του KaZaA είναι παρόμοιο με αυτό του Gnutella, κυρίως στο ότι δεν διαθέτει κεντρικό server για την εξυπηρέτηση των αιτήσεων και των ερωτημάτων. Σε αντίθεση, όμως, με αυτό διαφέρει στο γεγονός ότι δεν είναι όλοι οι κόμβοι ίσοι από άποψη αρμοδιοτήτων. Το δίκτυο του KaZaA διαχωρίζει τους κόμβους σε δύο κατηγορίες, τους Κανονικούς κόμβους (**O**rdinary **N**odes) και στους Υπέρ κόμβους (**S**uper **N**odes). Οι Ordinary Nodes είναι οι κόμβοι που ανήκουν στους κοινούς χρήστες που επιθυμούν να ανταλλάξουν αρχεία με άλλους, ενώ πιο δυνατοί από αυτούς είναι οι Super Nodes, οι οποίοι διαθέτουν πολλαπλάσια υπολογιστική ισχύ, bandwidth γραμμής και πολλαπλάσιες ώρες online λειτουργίας (uptime). Σκοπός των Super Nodes είναι να βοηθούν τους υπόλοιπους κόμβους του δικτύου στην αναζήτησή τους, με τον τρόπο που θα μελετήσουμε στη συνέχεια. Μια τυπική διάταξη του δικτύου είναι αυτή που φαίνεται στην παρακάτω εικόνα :



Εικόνα 5 Βασική αρχιτεκτονική του KaZaA δικτύου

Μεγάλος αριθμός από ordinary nodes συνδέονται με έναν super node και τον χρησιμοποιούν ως μεσάζοντα για την επικοινωνία τους με το υπόλοιπο δίκτυο. Σε κάθε ON αντιστοιχεί ένας SN και κάθε φορά που ένας ON ξεκινάει το λογισμικό του KaZaA, ανοίγει μια TCP σύνδεση με το αντίστοιχο SN. Έπειτα μεταδίδει στο SN πληροφορίες για τα αρχεία που μοιράζεται, με αποτέλεσμα κάθε SN να έχει αποθηκευμένες πληροφορίες στη βάση δεδομένων του για κάθε αρχείο που βρίσκεται σε κάποιο από τα ON «παιδιά» του. Με τον τρόπο αυτό, κάθε SN παίζει το ρόλο του κεντρικού server στο δίκτυο του Napster, μόνο που πρόκειται για έναν απλό κόμβο που ανήκει σε κάποιο χρήστη και όχι για εξειδικευμένο (dedicated) server.

Η αναζήτηση στο δίκτυο του KaZaA είναι παρόμοια με αυτή του Gnutella, με τη διαφορά ότι πραγματοποιείται επικοινωνία μόνο ανάμεσα στους super nodes. Η ερώτηση(query) που κάνει ένας χρήστης μεταδίδεται από το ON που βρίσκεται στο SN με το οποίο είναι συνδεδεμένος. Το SN αρχικά ελέγχει την τοπική του βάση δεδομένων σε περίπτωση που το ζητούμενο αρχείο βρίσκεται στην κατοχή κάποιου από τα παιδιά του. Παράλληλα αποστέλλει την ερώτηση και σε άλλους SN με τους οποίους διατηρεί ανοιχτά TCP connections, ώστε να βρεθούν πιθανώς περισσότερα αποτελέσματα. Τα αποτελέσματα της αναζήτησης, είτε τοπικά για τον SN είτε απομακρυσμένα, αποστέλλονται στον χρήστη που εκτέλεσε την αναζήτηση μαζί φυσικά με τις IP διευθύνσεις των κόμβων που περιέχουν το αρχείο που ζητήθηκε.

Από αυτό το σημείο απομένει μόνο από τον κόμβο του χρήστη να ξεκινήσει η TCP σύνδεση που θα αρχικοποιήσει την επικοινωνία ανάμεσα στα δύο μέρη.

Η υιοθέτηση του παραπάνω σχεδιασμού έχει σημαντικά πλεονεκτήματα έναντι τόσο του Napster, όσο και του Gnutella δικτύου. Από τη μία, αποφεύγεται η χρήση κεντρικού εξυπηρετητή από τον οποίο θα εξαρτιόταν η διαθεσιμότητα της υπηρεσίας και από την άλλη οι αναζητήσεις δεν πλημμυρίζουν το δίκτυο των χρηστών με άχρηστη πληροφορία, παρά μόνο τις υψηλής ταχύτητας συνδέσεις μεταξύ των super nodes. Με άλλα λόγια, το δίκτυο του KaZaA εκμεταλλεύεται την ετερογένεια των κόμβων σε επίπεδο επεξεργαστικής ισχύος, bandwidth γραμμής και χρόνου online λειτουργίας, για να επιτύχει ένα συνδυασμό των πλεονεκτημάτων των δικτύων Napster και Gnutella. Είναι προφανές ότι αν χρησιμοποιηθούν περισσότερες από δύο κλάσεις διαχωρισμού, βελτιώνεται περαιτέρω η αποδοτικότητα του δικτύου, αυξάνοντας, όμως, παράλληλα και την πολυπλοκότητα.

eDonkey

Η βασική αρχή λειτουργίας του eDonkey *είναι* παρόμοια με αυτή του KaZaA. Δεν υπάρχει κεντρικός server όπως στο Napster, ούτε και πλήρης αποκέντρωση όπως στο Gnutella.

Οι κόμβοι του δικτύου διαχωρίζονται σε δύο βαθμίδες, τους servers και τους clients, κατ' αντιστοιχία με τους super nodes και ordinary nodes του KaZaA δικτύου. Οι clients δε διαφέρουν σε τίποτα από τους ordinary nodes του KaZaA δικτύου, με άλλα λόγια απλά πραγματοποιούν ερωτήσεις προς τις ανώτερες βαθμίδες και αρχικοποιούν συνδέσεις για μεταφορές αρχείων. Οι servers, από την άλλη, έχουν ανάλογες αρμοδιότητες με αυτές των super nodes, δηλαδή να διατηρούν ακριβή λίστα με τους online κόμβους, να απαντάνε στα ερωτήματα των χρηστών και να επικοινωνούν μεταξύ τους για να προσφέρουν περισσότερα αποτελέσματα στους χρήστες. Σε αντίθεση, όμως, με τους super nodes, δεν μοιράζονται οι ίδιοι αρχεία στο δίκτυο. Η

μόνη τους χρήση είναι να καθοδηγούν τις αιτήσεις των χρηστών και να διατηρούν συνεπή τη βάση δεδομένων τους.

Δεδομένου ότι οι servers διατηρούν μια δεύτερη λίστα(πέρα από αυτή των online χρηστών) από όλους τους υπόλοιπους online servers και η λίστα αυτή αλλάζει δυναμικά, καθίσταται επιτακτική η ανάγκη να επικοινωνούν μεταξύ τους ανά διαστήματα, ανταλλάσσοντας τις τοπικές τους λίστες. Αυτό γίνεται με μηνύματα τύπου ping/pong, όπου κατά το ping μήνυμα δηλώνεται η παρουσία ενός και κατά την απάντηση με μήνυμα pong, αποστέλλεται η τοπική λίστα των online servers. Αντίθετα, μεταξύ των χρηστών η επικοινωνία περιορίζεται μόνο στην αρχικοποίηση των συνδέσεων και στη μεταφορά δεδομένων.

Ένα ακόμα σημαντικό χαρακτηριστικό του eDonkey, είναι η δυνατότητα για ταυτόχρονο κατέβασμα από διαφορετικές πηγές. Όταν ένας χρήστης ζητήσει κάποιο αρχείο για κατέβασμα, το λογισμικό αποστέλλει την αίτηση του χρήστη (εκτός από τον κόμβο που κατέχει το αρχείο) στον αντίστοιχο server που διαχειρίζεται τις αιτήσεις του. Στο σημείο αυτό, ο server προσπαθεί να βρει άλλους χρήστες που να διαθέτουν το αρχείο αυτό, συγκρίνοντας το hash value του αρχείου με τα υπάρχοντα της βάσης δεδομένων. Η σύγκριση επεκτείνεται και στις λίστες άλλων servers για την εξαγωγή περισσοτέρων αποτελεσμάτων. Μόλις συγκεντρωθούν τα αποτελέσματα, αποστέλλονται πίσω στο λογισμικό του client και αρχικοποιούνται επιπλέον συνδέσεις προς άλλες πηγές του αρχείου. Αυτό οδηγεί σε μικρότερο χρόνο αναμονής για την απόκτηση του αρχείου και φυσικά αυξάνει την αποδοτικότητα του δικτύου.

Πρέπει, επίσης, να αναφερθεί η δυνατότητα του πρωτοκόλλου να κατεβάζει τα αρχεία σε κομμάτια, κατ' αντιστοιχία με το BitTorrent. Τα κομμάτια αυτά (της τάξης των 10Mb το καθένα) γίνονται αμέσως διαθέσιμα στο δίκτυο, προσφέροντας με τον τρόπο αυτό περισσότερες πηγές για το ίδιο αρχείο. Με τον τρόπο αυτό κάθε κόμβος μπορεί να κατεβάζει και να προσφέρει αρχεία την ίδια στιγμή, βελτιστοποιώντας την αποδοτικότητα του δικτύου και ελαχιστοποιώντας το χρόνο απόκτησης.

Αξίζει, τέλος, να σημειωθεί πως ο δημιουργός του eDonkey (MetaMachine) δημιούργησε επίσης και ένα ακόμα δίκτυο, το Overnet. Το λογισμικό του eDonkey δίνει τη δυνατότητα στους χρήστες να συνδεθούν και με τα δύο δίκτυα, αυξάνοντας τον αριθμό των χρηστών που απολαμβάνουν τις υπηρεσίες τους.

Προκλήσεις που καλούνται να αντιμετωπίσουν τα σημερινά P2P δίκτυα

Από τη μελέτη των χαρακτηριστικών των παραπάνω δημοφιλών P2P δικτύων και κατά κύριο λόγο από τη μελέτη των σημείων στα οποία υστερούν, μπορούμε να εξάγουμε σημαντικά συμπεράσματα για τις προκλήσεις και δυσκολίες που έχουν να αντιμετωπίσουν οι σχεδιαστές νέων P2P αρχιτεκτονικών. Χαρακτηριστικό παράδειγμα είναι ο τρόπος με τον οποίο γίνεται η αναζήτηση ενός αρχείου στο δίκτυο. Στο Gnutella δίκτυο, το ερώτημα ενός χρήστη αποστέλλεται στους γνωστούς του κόμβους και αυτοί με τη σειρά τους σε δικούς τους γνωστούς, πλημμυρίζοντας με τον τρόπο αυτό το δίκτυο. Προφανώς, ο τρόπος αυτός δεν εγγυάται την εύρεση του αρχείου ενώ ταυτόχρονα υπερφορτώνει το δίκτυο με άχρηστη πληροφορία. Στο Napster δίκτυο, η εύρεση είναι απλή υπόθεση, δεδομένου ότι η ζητούμενη πληροφορία για τον κόμβο που κατέχει το αρχείο βρίσκεται αποθηκευμένη στον κεντρικό εξυπηρετητή. Ούτε αυτή η λύση, όμως, είναι βέλτιστη, δεδομένου ότι ο κεντρικός server υπόκειται σε κάθε είδους κίνδυνο που διακυβεύει τη διαθεσιμότητα του συστήματος. Το KaZaA δίκτυο από την άλλη μεριά, κατάφερε να ξεπεράσει σε μεγάλο βαθμό το πρόβλημα αυτό υιοθετώντας μια ενδιάμεση αρχιτεκτονική με υπερκόμβους που εξυπηρετούν ως κεντρικοί servers για τους υπόλοιπους. Αν και το πρόβλημα που αντιμετώπισε το Gnutella εξακολουθεί να υφίσταται, εντούτοις έχουν μειωθεί δραστικά οι αρνητικές επιπτώσεις του, καθιστώντας τη λύση αυτή πιο βιώσιμη.

Το παραπάνω πρόβλημα είναι ένα μόνο από τα πολλά που έχουν να αντιμετωπίσουν τα σημερινά P2P δίκτυα ανταλλαγής αρχείων. Ένα ακόμα σημαντικό πρόβλημα είναι αυτό της επιλογής του πιο κατάλληλου κόμβου για λήψη ενός αρχείου. Δεδομένου ότι η ζητούμενη πληροφορία βρέθηκε σε περισσότερους του ενός κόμβους, ζητείται να βρεθεί ένας αποδοτικός τρόπος για να επιλεγεί ο κόμβος από τον οποίο θα ληφθεί τελικά το αρχείο. Η πλειοψηφία των προγραμμάτων που αντιπροσωπεύουν τα γνωστά P2P δίκτυα, απλώς αναφέρουν στο χρήστη τις τοποθεσίες που βρέθηκε η πληροφορία και αφήνεται στο χρήστη η επιλογή της πηγής. Η πρακτική αυτή, φυσικά δεν είναι η βέλτιστη δυνατή καθώς δεν γίνεται δίκαιη κατανομή του φόρτου στους κόμβους. Μάλιστα, στα δίκτυα εκείνα που υποστηρίζουν την ύπαρξη πολλών ταυτόχρονων πηγών, ξεκινά η μεταφορά

δεδομένων από όλες τις πηγές μέχρι ένα συγκεκριμένο όριο, χωρίς, όμως, να υπάρχει εγγύηση ότι επιλέχθηκαν οι γρηγορότερες από αυτές. Επίσης, ένας σημαντικός παράγοντας που πρέπει να λαμβάνεται υπ' όψιν κατά την επιλογή της πηγής είναι η χωρητικότητα των γραμμών διασύνδεσης, δεδομένου ότι διαφορετική ικανότητα έχουν δύο κόμβοι που ο ένας έχει σύνδεση T1 και ο άλλος 56kbps. Στην πλειοψηφία των προγραμμάτων η επιλογή αυτή αφήνεται στο χρήστη (το πρόγραμμα απλά αναγράφει την ταχύτητα της γραμμής κάθε κόμβου όπως έχει δηλωθεί από τον ίδιο το χρήστη και η οποία πολλές φορές, όπως είναι φυσικό, μπορεί να είναι αναληθής ή και ψευδής). Σε προγράμματα που ανοίγουν πολλαπλές συνδέσεις, δεν γίνεται κανένας διαχωρισμός ανάμεσα στις πηγές, με αποτέλεσμα αν επιλεγούν μόνο κόμβοι με γραμμές χαμηλής χωρητικότητας, να μην επιτυγχάνονται οι βέλτιστοι χρόνοι εξυπηρέτησης.

Ένα ακόμα πρόβλημα, από τη μεριά του κόμβου-εξυπηρετητή, είναι η επιλογή της επόμενης αίτησης προς εξυπηρέτηση, ανάμεσα στις διαθέσιμες που έχουν υποβληθεί. Η επιλογή αυτή μπορεί να γίνει με βάση ορισμένα χαρακτηριστικά των αιτήσεων, όπως η απαιτούμενη διάρκεια εξυπηρέτησης, ο χρόνος αναμονής της στην ουρά του εξυπηρετητή, η προέλευση της αίτησης κλπ. Η επιλογή αυτή μπορεί να επηρεαστεί και από το είδος των αρχείων τα οποία διαχειρίζεται το P2P δίκτυο. Όταν πρόκειται για μεικτό δίκτυο (εικόνες, ήχος, βίντεο), ένας χρήστης που επιθυμεί να «κατεβάσει» μια ταινία μεγάλου μεγέθους, αυτομάτως δυσχεραίνει την εξυπηρέτηση άλλων χρηστών που μπορεί να επιθυμούν μια εικόνα μερικών Kb, και οι οποίοι είναι αναγκασμένοι να περιμένουν μεγάλο χρονικό διάστημα. Όταν μάλιστα πρόκειται για P2P δίκτυο που κάνει real-time μετάδοση δεδομένων (φωνής ή εικόνας) , η έννοια της καθυστέρησης εξυπηρέτησης (μετάδοσης) είναι πρωτεύουσας σημασίας.

Όλα τα παραπάνω είναι ανοικτά προβλήματα, τα οποία βρίσκονται υπό συνεχή έρευνα από τους αντίστοιχους επιστημονικούς τομείς (κατανεμημένα συστήματα, βάσεις δεδομένων, δίκτυα υπολογιστών κλπ). Η συνεχής εξέλιξη επιφέρει σημαντικές αλλαγές στα νέα P2P δίκτυα που κάνουν την εμφάνισή τους και περιορίζουν ή εξαλείφουν τα αρνητικά χαρακτηριστικά τους, βελτιστοποιώντας τα σε μεγάλο βαθμό. Παρ' όλ' αυτά η ανάπτυξη βρίσκεται ακόμα σε πρώιμο στάδιο, ιδιαίτερα αν σκεφτεί κανείς ότι τα πρώτα P2P δίκτυα έκαναν την εμφάνισή τους μόλις μερικά χρόνια πριν και δεν έχουν ανακαλυφθεί ακόμα οι τεράστιες δυνατότητες που προσφέρουν τόσο σε προσωπικό επίπεδο σαν αυτόνομοι χρήστες, όσο και σε εταιρικό επίπεδο.

Ενότητα

3

Προσομοίωση P2P δικτύου

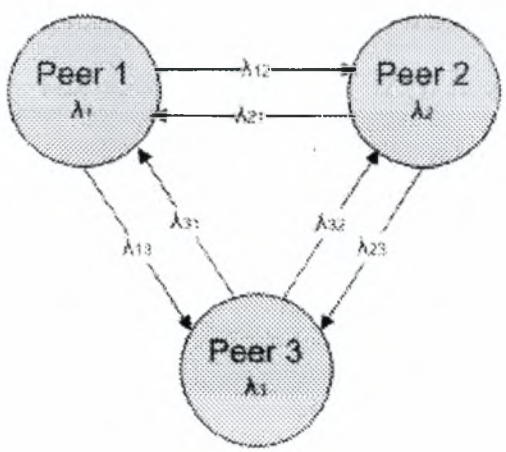
Μοντέλο συστήματος

Θεωρούμε ένα peer-to-peer σύστημα που αποτελείται από 3 κόμβους (peers), οι οποίοι είναι υπολογιστικά συστήματα με πανομοιότυπα χαρακτηριστικά. Σκοπός του συστήματος είναι να δώσει τη δυνατότητα στους χρήστες του (οι οποίοι χρησιμοποιούν τους κόμβους) να διαμοιραστούν τα αρχεία τους. Τα αρχεία αυτά δεν είναι κάποιας συγκεκριμένης μορφής, αλλά θεωρούμε ότι έχουν κυμαινόμενα μεγέθη, από μερικά KB μέχρι GB. Οι κόμβοι του συστήματος είναι πλήρως διασυνδεδεμένοι μεταξύ τους (fully connected) και χωρίς καμία εξωτερική σύνδεση με το περιβάλλον. Καθένας από αυτούς απαιτεί ανά χρονικά διαστήματα αρχεία από τους γειτονικούς κόμβους, παράγοντας με τον τρόπο αυτό κίνηση στο δίκτυο όταν πραγματοποιείται η μεταφορά. Θεωρούμε ότι οι κόμβοι διαθέτουν όλα τα δυνατά αρχεία που μπορεί να ζητηθούν και επομένως πάντα απαντάνε στις αιτήσεις των υπολοίπων.

Η χωρητικότητα των γραμμών διασύνδεσης είναι σταθερή και ίδια για όλες τις γραμμές. Η τιμή που παίρνει δε θα μας απασχολήσει καθώς υπεισέρχεται στον παράγοντα «χρόνος εξυπηρέτησης» (ή service time). Αν θεωρήσουμε m το μέγεθος ενός αρχείου, C τη χωρητικότητα της γραμμής διασύνδεσης και t το χρόνο μετάδοσης, τότε αυτά συνδέονται με τον τύπο $t = \frac{m}{C}$. Επομένως, οποιαδήποτε κατανομή θεωρήσουμε για τα μεγέθη των αρχείων, την ίδια κατανομή θα ακολουθούν και οι χρόνοι μετάδοσης, δηλ. τα service times. Το μόνο που χρειάζεται τελικά να γνωρίζουμε είναι η κατανομή των μεγεθών των αρχείων που ζητάνε συνήθως οι χρήστες σε ένα τυπικό peer-to-peer σύστημα. Συγκεκριμένα, στις προσομοιώσεις που θα ακολουθήσουν θα χρησιμοποιήσουμε την εκθετική κατανομή για το χρόνο ανάμεσα σε διαδοχικές εξυπηρετήσεις αιτήσεων. Με τον τρόπο αυτό, οι εξυπηρετήσεις ακολουθούν κατανομή Poisson, με μέσο ρυθμό εξυπηρέτησης που δίνεται από την παράμετρο της Poisson κατανομής και συμβολίζουμε με "μ". Στο μοντέλο που χρησιμοποιήσαμε, θεωρούμε ρυθμούς εξυπηρέτησης μεταξύ 0 και 1 και

η μονάδα μέτρησης είναι εξυπηρετήσεις ανά λεπτό. Επομένως, όταν αναφερόμαστε σε ρυθμό εξυπηρέτησης 0.5 (μ=0.5), εννοούμε 0.5 εξυπηρετήσεις ανά λεπτό κατά μέσο όρο (ή 1 εξυπηρέτηση ανά 2 λεπτά).

Από την άλλη μεριά, πρέπει να καθορίσουμε τον τρόπο με τον οποίο καταφθάνουν οι αιτήσεις στις ουρές των εξυπηρετητών (οι οποίοι είναι οι ίδιοι οι κόμβοι). Θεωρούμε ότι όλοι οι κόμβοι ζητάνε αρχεία με τον ίδιο τρόπο, δηλαδή ότι οι κατανομές των χρόνων παραγωγής αιτήσεων και των τριών κόμβων είναι οι ίδιες. Στο σημείο αυτό πρέπει να γίνει ο διαχωρισμός ανάμεσα στους όρους "ρυθμός παραγωγής" και "ρυθμός άφιξης" αιτήσεων. Ως ρυθμό παραγωγής αιτήσεων ενός κόμβου εννοούμε το ρυθμό με τον οποίο ο κόμβος αυτός ζητάει αρχεία από οποιουσδήποτε γειτονικούς του. Στο σύστημα που μελετάμε με τρεις κόμβους, καθένας από τους ρυθμούς αυτούς χωρίζεται (ή «σπάζει» όπως θα λέμε απο εδώ και πέρα) σε δύο επιμέρους τμήματα. Το πρώτο τμήμα είναι ο ρυθμός με τον οποίο ο υπό εξέταση κόμβος ζητάει αρχεία από τον ένα γείτονα και ο δεύτερος, ο ρυθμός που ζητάει αρχεία από τον άλλο. Αν για παράδειγμα "λ" είναι ο ρυθμός με τον οποίο ο κόμβος 1 ζητάει αρχεία (που ονομάζεται ρυθμός παραγωγής), τότε αυτός σπάει σε λ_{12} για τις αιτήσεις που γίνονται προς τον 2 και λ_{13} για τις αιτήσεις προς τον 3. Οι λ_{12} και λ_{13} θεωρούνται ως ρυθμοί αφίξεων αιτήσεων για τους 2 και 3 αντίστοιχα, ενώ όπως είναι λογικό, ισχύει ότι $\lambda = \lambda_{12} + \lambda_{13}$. Το σπάσιμο των ρυθμών παραγωγής, με άλλα λόγια εικονίζεται στο παρακάτω σχήμα, από το οποίο συμπεραίνουμε ότι οι ρυθμοί αφίξεων για του κόμβους 1,2 και 3 είναι $\lambda_{21}+\lambda_{31}$, $\lambda_{12}+\lambda_{32}$ και $\lambda_{23}+\lambda_{13}$ αντίστοιχα :



Εικόνα 6 Αναπαράσταση μοντέλου για τρεις κόμβους-peers

Η μονάδα μέτρησης των ρυθμών παραγωγής και άφιξης αιτήσεων είναι κοινή και θεωρούμε ότι είναι "αριθμός αιτήσεων ανά λεπτό". Όταν, για παράδειγμα,

αναφερόμαστε σε ρυθμό 0.5 εννοούμε ότι έχουμε 0.5 αφίξεις κάθε λεπτό (ή 1 άφιξη κάθε 2 λεπτά).

Δεδομένου ότι η αποσύνθεση μιας Poisson διαδικασίας (παραγωγή αιτήσεων σε κάθε κόμβο) μια δίνει επίσης Poisson διαδικασίες(με ρυθμούς λ_{ij}) και η σύνθεση επιμέρους Poisson διαδικασιών είναι Poisson διαδικασία (άφιξη αιτήσεων σε κάθε κόμβο), συμπεραίνουμε ότι οι χρόνοι μεταξύ δύο διαδοχικών αφίξεων ακολουθούν εκθετική κατανομή παρ' όλο που μπορεί να μην προέρχονται από τον ίδιο κόμβο. Το παραπάνω, σε συνδυασμό με το γεγονός ότι οι εξυπηρετήσεις ακολουθούν επίσης Poisson κατανομή αποδεικνύει ότι οι ουρές σε κάθε κόμβο είναι τύπου M/M/1.

Στα παραδείγματα που θα ακολουθήσουν θεωρούμε πάντα ότι ο ρυθμός παραγωγής αιτήσεων είναι μικρότερος από το ρυθμό εξυπηρέτησής τους σε κάθε κόμβο. Παρ' όλ' αυτά, λόγω του σπασίματος των ρυθμών παραγωγής, είναι δυνατόν σε κάποιο κόμβο να εισέρχονται αιτήσεις με μεγαλύτερο ρυθμό απ' ότι εξυπηρετούνται. Για το λόγο αυτό δεν ισχύει η συνθήκη ευστάθειας της M/M/1 ουράς που εξασφαλίζει ότι η ουρά δεν μεγαλώνει ανεξέλεγκτα. Επειδή, όμως, μελετάμε τη συμπεριφορά του συστήματος για συγκεκριμένο αριθμό πακέτων και όχι επ' αόριστον, τότε αν σε ένα κόμβο ο συνολικός ρυθμός αφίξεων είναι μεγαλύτερος από αυτόν των εξυπηρετήσεων, το μέγεθος της ουράς αυξάνεται μέχρι να σταματήσουν να έρχονται πακέτα στο σύστημα. Το φαινόμενο αυτό είναι συχνό σε πραγματικά P2P συστήματα, ιδιαίτερα σε κόμβους που δεν διαθέτουν μεγάλη χωρητικότητα γραμμής (που συνεπάγεται μικρό ρυθμό εξυπηρέτησης) αλλά δέχονται μεγάλο αριθμό αιτήσεων.

Κάθε προσομοίωση που θα πραγματοποιήσουμε διαρκεί για όσο διάστημα τροφοδοτείται το σύστημα με αιτήσεις. Για το λόγο αυτό καθορίζουμε πριν την εκκίνηση της προσομοίωσης τον αριθμό των αιτήσεων που θα ζητήσουμε από κάθε κόμβο να εξυπηρετήσει, τον οποίο θεωρούμε βασική μεταβλητή του συστήματος. Τη μεταβλητή αυτή συμβολίζουμε με το λατινικό γράμμα "n" και όταν αναφερόμαστε σε "n=1000" εννοούμε ότι κάθε κόμβος θα εξυπηρετήσει συνολικά 1000 αιτήσεις. Από αυτές τις n αιτήσεις, θεωρούμε ότι οι μισές προέρχονται από τον ένα γειτονικό κόμβο και οι άλλες μισές από τον άλλο. Με τον τρόπο αυτό δημιουργείται το εξής παράδοξο: επειδή οι ρυθμοί αφίξεων είναι διαφορετικοί για κάθε γείτονα (έστω λ_1 και λ_2 αντίστοιχα, με $\lambda_1 > \lambda_2$) οι $\frac{n}{2}$ αιτήσεις του πρώτου γείτονα θα φτάσουν στο σύστημα

πιο γρήγορα από αυτές του δεύτερου. Πράγματι, οι $\frac{n}{2}$ αιτήσεις του πρώτου γείτονα κάνουν $t_1 = \frac{n/2}{\lambda_1} = \frac{n}{2\lambda_1}$ για να φτάσουν όλες και του δεύτερου ομοίως $t_2 = \frac{n}{2\lambda_2}$.

Επομένως υπάρχει ένα διάστημα $t_2 - t_1 > 0$ κατά το οποίο πραγματοποιούνται αφίξεις μόνο από τον δεύτερο γειτονικό κόμβο, πράγμα το οποίο οδηγεί (για το συγκεκριμένο διάστημα) σε μείωση του συνολικού ρυθμού αφίξεων, μείωση των αιτήσεων στην ουρά του εξυπηρετητή (αν υπάρχουν αιτήσεις στην ουρά), συνακόλουθη μείωση των χρόνων αναμονής στην ουρά και γενικά συμπεριφορά που δεν συμπίπτει με τα αναμενόμενα. Επειδή, όμως, για $\lambda_1 \approx \lambda_2$ το διάστημα $t_2 - t_1$ είναι αμελητέο και για $\lambda_1 >> \lambda_2$ (ή $\lambda_2 >> \lambda_1$) η μείωση στο συνολικό λ είναι αμελητέα (επειδή $\lambda = \lambda_1 + \lambda_2 \approx \lambda_1$), το φαινόμενο αυτό δεν μας απασχολεί ιδιαίτερα (αν και θα αναλυθεί σε μορφή παραδείγματος κατά την προσομοίωση της FCFS περίπτωσης στη συνέχεια).

Στοιχεία απόδοσης

Για να θεωρηθεί ένα peer-to-peer σύστημα επιτυχημένο, απαιτείται πρώτα απ' όλα να ικανοποιεί τους χρήστες του, δηλαδή να παρέχει υψηλή ποιότητα υπηρεσίας (Quality of Service). Οι μεταβλητές του συστήματος που έχουν να κάνουν με την ποιότητα υπηρεσίας βασίζονται κατά πρώτο λόγο σε ανθρώπινους παράγοντες. Για παράδειγμα, κανείς δεν επιθυμεί να περιμένει μεγάλο χρονικό διάστημα μέχρι να καταφέρει να αποκτήσει ένα πολυπόθητο αρχείο. Στο σύστημα που μελετάμε, ο παράγοντας αυτός αντιστοιχεί (κατά πρώτο λόγο) στο χρόνο αναμονής στην ουρά του κόμβου από τον οποίο ζητήθηκε το αρχείο. Ο χρόνος μέχρι να αποκτήσει ο χρήστης το αρχείο είναι το άθροισμα του χρόνου αναμονής στην ουρά και του χρόνου εξυπηρέτησης. Παρ' όλ' αυτά ο χρόνος εξυπηρέτησης εξαρτάται αποκλειστικά από το μέγεθος του αρχείου, δεδομένου ότι θεωρούμε γραμμές σύνδεσης ίδιας χωρητικότητας. Επομένως, θεωρούμε ως βασικό μέτρο απόδοσης το χρόνο αναμονής στην ουρά, ο οποίος εξαρτάται από το ρυθμό άφιξης αιτήσεων στην ουρά (θεωρούμε ότι όλοι οι κόμβοι παράγουν αιτήσεις με τον ίδιο ρυθμό, παρ' όλο που με το σπάσιμο

του κάθε ρυθμού τελικά κάθε κόμβος έχει διαφορετικό ρυθμό αφίξεων), από το ρυθμό εξυπηρέτησης (επίσης ίδιος για κάθε κόμβο) και τέλος από τον αλγόριθμο εξυπηρέτησης που επιλέγουμε. Μια ακόμα σημαντική παράμετρος που είναι επιθυμητή σε όλα τα peer-to-peer δίκτυα, είναι η δικαιοσύνη. Με τον όρο αυτό εννοούμε το κατά πόσο ένας κόμβος παραμελεί αιτήσεις που έχουν συγκεκριμένη πηγή και ευνοεί άλλες. Σε γενικές γραμμές είναι επιθυμητό, ένας κόμβος που λειτουργεί ως εξυπηρετητής να μην κάνει διακρίσεις ανάμεσα σε κόμβους από τους οποίους δέχεται αιτήσεις, παρά το γεγονός ότι συχνά μπορεί να τους μεταχειρίζεται με διαφορετικές προτεραιότητες. Σε περίπτωση που αιτήσεις από συγκεκριμένες πηγές παραμελούνται και παραγκωνίζονται στην ουρά ενός εξυπηρετητή, οι χρόνοι αναμονής που προκύπτουν δεν είναι ανθρώπινα αποδεκτοί. Με άλλα λόγια η έννοια της δικαιοσύνης είναι άρρηκτα δεμένη με τον παράγοντα «χρόνος καθυστέρησης στην ουρά».

Αλγόριθμοι εξυπηρέτησης

Όσο διαρκεί η εξυπηρέτηση μιας αίτησης σε ένα κόμβο, ανάλογα με το ρυθμό άφιξης συσσωρεύονται στην ουρά επιπλέον αιτήσεις από άλλους κόμβους που περιμένουν να εξυπηρετηθούν. Τη στιγμή που τελειώνει η εξυπηρέτηση της τρέχουσας αίτησης, ο κόμβος πρέπει να επιλέξει την επόμενη αίτηση προς εξυπηρέτηση. Η επιλογή αυτή μπορεί να γίνει με διάφορα χαρακτηριστικά, διαμορφώνοντας διαφορετικές πολιτικές εξυπηρέτησης. Οι κυριότερες πολιτικές εξυπηρέτησης είναι οι παρακάτω:

- First Come First Served (FCFS) : Είναι ο κλασικός αλγόριθμος, όπου επιλέγεται η παλαιότερη από τις αιτήσεις που βρίσκονται στην ουρά (η αίτηση που έφτασε πρώτη στην ουρά). Η ουρά αυτή είναι παρόμοια, για παράδειγμα, με την ουρά ανθρώπων σε μια τράπεζα. Το κυριότερο χαρακτηριστικό της είναι ότι είναι δίκαιη, επειδή δεν δίνει προτεραιότητα σε κάποια συγκεκριμένη αίτηση της ουράς, ενώ δεν απαιτεί καμία απολύτως γνώση για αυτές.
- Round Robin : Δεδομένου ότι σε κάθε κόμβο εισέρχονται αιτήσεις από άλλους δύο με διαφορετικούς ρυθμούς, ο αλγόριθμος αυτός επιλέγει εναλλάξ ανάμεσα σ' αυτούς την επόμενη αίτηση για εξυπηρέτηση. Αν, για παράδειγμα, στον κόμβο 1 εισέρχονται αιτήσεις από τους κόμβους 2 και 3,

τότε ο 1 ακολουθεί την παρακάτω πρακτική : εξυπηρετεί αρχικά την αίτηση που έφτασε πρώτη. Αν αυτή ήταν του κόμβου 2, τότε όταν αυτή ολοκληρωθεί ελέγχει την ουρά για αιτήσεις του κόμβου 3. Σε περίπτωση που βρει αιτήσεις του κόμβου 3, εξυπηρετεί την παλιότερη (για να διατηρείται με αυτόν τον τρόπο η δικαιοσύνη ανάμεσα σε κάθε κλάση αιτήσεων). Έπειτα αναζητεί αιτήσεις του κόμβου 2 στην ουρά και εναλλάσσει συνεχώς ανάμεσα στις δύο κλάσεις. Σε περίπτωση που δεν βρει αίτηση της κλάσης που αναζητεί, τότε εξυπηρετεί αίτηση της ίδιας κλάσης που εξυπηρετούσε πριν. Ο αλγόριθμος αυτός επιτυγχάνει να ικανοποιήσει και τις 2 κλάσεις αιτήσεων, δίνοντας όμως ουσιαστικά περισσότερη βαρύτητα στον κόμβο με το μικρότερο ρυθμό αφίξεων. Έτσι όταν έχουμε δύο κλάσεις αιτήσεων με τη μία κλάση να έχει πολύ μεγαλύτερο ρυθμό αφίξεων από την άλλη, δεν εξυπηρετούνται οι συνεχόμενες αιτήσεις της κλάσεις με το μεγαλύτερο ρυθμό, αλλά εναλλάσσονται με αυτές της κλάσης με τον μικρότερο. Καθυστερεί με άλλα λόγια η εξυπηρέτηση των αιτήσεων της κλάσης με το μεγαλύτερο ρυθμό και επιταχύνεται αυτής με το μικρότερο. Μόνη προϋπόθεση για την ορθή λειτουργία του αλγορίθμου είναι η εκ των προτέρων γνώση της προέλευσης των αιτήσεων στην ουρά. Είναι ανάγκη, δηλαδή, να γνωρίζει ο κόμβος που μελετάμε από ποιον κόμβο προήλθαν οι αιτήσεις που βρίσκονται κάθε στιγμή στην ουρά.

- Shortest Job First : Ο συγκεκριμένος αλγόριθμος επιλέγει από τις αιτήσεις που βρίσκονται στην ουρά, εκείνη με το μικρότερο χρόνο εξυπηρέτησης. Σε αντίθεση με τους προηγούμενους, απαιτεί την εκ των προτέρων γνώση του χρόνου εξυπηρέτησης κάθε αίτησης. Η γνώση αυτή υπάρχει, δεδομένου ότι υπάρχει αναλογία ανάμεσα στο χρόνο εξυπηρέτησης και στο μέγεθος του ζητούμενου αρχείου, το οποίο γνωρίζει ο κόμβος που το διαθέτει. Με τον τρόπο αυτό δεν ευνοούνται οι αιτήσεις της μιας κλάσης ή της άλλης, δεδομένου βέβαια ότι και οι δύο κλάσει ζητάνε αρχεία της ίδιας μορφής (σε περίπτωση που ο ένας κόμβος ζητάει μεγάλα αρχεία και ο άλλος μικρά, μοιραίο είναι να εξυπηρετεί το σύστημα τις αιτήσεις του δεύτερου κόμβου. Παρ' όλ' αυτά ξεκινήσαμε με την παραδοχή ότι οι αιτήσεις που παράγουν οι κόμβοι είναι της ίδιας μορφής και επομένως δεν υφίσταται τέτοιας μορφής αδικία). Χαρακτηριστικό του αλγορίθμου αυτού, είναι ότι δίνει προτεραιότητα σε αιτήσεις που δεν διαρκούν μεγάλο χρονικό διάστημα, ενώ καθυστερεί

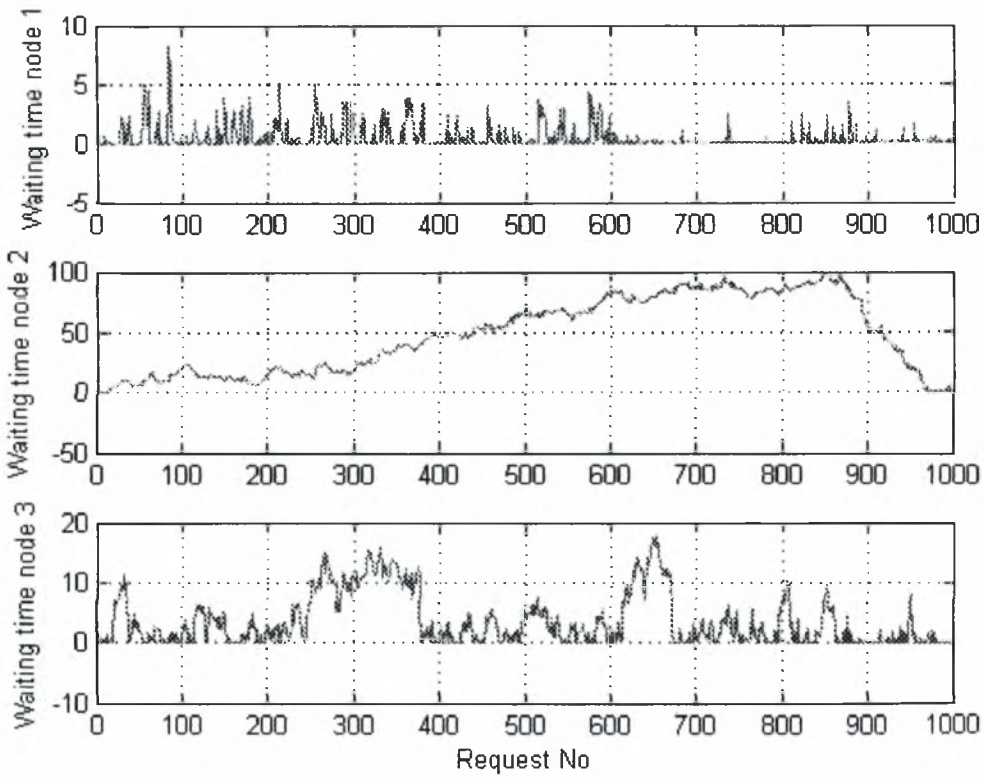
εκείνες που απαιτούν πολύ χρόνο για να ολοκληρωθούν. Η αίτηση ενός χρήστη για ένα μεγάλο αρχείο (π.χ. μια ταινία) θα διαρκέσει μεγάλο χρονικό διάστημα, επειδή θα παραγκωνίζεται από τις αιτήσεις άλλων χρηστών για μικρότερα αρχεία (π.χ. mp3). Μάλιστα, για μεγάλους ρυθμούς αφίξεων (συγκρίσιμους με το ρυθμό εξυπηρέτησης), είναι πιθανό οι αιτήσεις αυτές να μην εξυπηρετηθούν ουσιαστικά ποτέ, αφού πάντα θα υπάρχουν μικρότερες αιτήσεις στην ουρά. Εντούτοις, όπως θα αποδειχθεί στη συνέχεια η πολιτική αυτή επιφέρει σημαντικές βελτιώσεις στην απόδοση του συστήματος και είναι ιδιαίτερα ανταγωνιστική.

- Combined Round Robin-Shortest Job First : Πρόκειται για συνδυασμό των προηγούμενων δύο αλγορίθμων, ο οποίος συνδυάζει τα βασικά χαρακτηριστικά τους. Πιο συγκεκριμένα, ο αλγόριθμος αρχικά επιλέγει από την ουρά την αίτηση με το μικρότερο χρόνο εξυπηρέτησης, σύμφωνα με τον κλασσικό SJF αλγόριθμο που είδαμε προηγουμένως. Παράλληλα, όμως, συγκρατεί στη μνήμη ποιας κλάσης ήταν η αίτηση που επιλέχθηκε και στη συνέχεια επιλέγει από την ουρά την αίτηση εκείνη που ανήκει στη αντίθετη κλάση από αυτή που εξυπηρετήθηκε αμέσως πιο πριν και έχει το μικρότερο χρόνο εξυπηρέτησης. Συνδυάζει με τον τρόπο αυτό τις δύο πολιτικές, αφού επιλέγεται η μικρότερη αίτηση και εναλλάσσονται οι δύο διαφορετικές κλάσεις σε κάθε επανάληψη. Επιτυγχάνεται με τον τρόπο αυτό να γίνει ο αλγόριθμος SJF λίγο πιο δίκαιος κατά την επιλογή της επόμενης αίτησης. Θα παρατηρήσουμε και στη συνέχεια ότι με βάση τους μέσους χρόνους αναμονής στην ουρά, ο αλγόριθμος αυτός τοποθετείται ανάμεσα στους SJF και Round Robin, αφού, όπως φαίνεται, οι παράγοντες δικαιοσύνη και χρόνος αναμονής είναι αντιστρόφως ανάλογοι. Η μείωση του ενός οδηγεί σε αύξηση του άλλου και επομένως σκοπός είναι η εύρεση της χρυσής τομής ανάμεσα σ' αυτούς.
- Fixed Priority : Ο αλγόριθμος αυτός βασίζεται στην απόδοση προτεραιοτήτων σε κάθε κλάση αιτήσεων. Οι προτεραιότητες αυτές είναι γνωστές εξ αρχής από το σύστημα και χρησιμοποιούνται για την επιλογή της επόμενης προς εξυπηρέτηση αίτησης. Για παράδειγμα, όταν ο κόμβος 1 δίνει μεγαλύτερη προτεραιότητα στον κόμβο 2 απ' ότι στον 3, τότε κατά την επιλογή της επόμενης αίτησης προς εξυπηρέτηση, ο 1 θα ελέγξει την ουρά του για αιτήσεις του 2 και θα τις εξυπηρετήσει πριν από αυτές του 3. Όπως είναι

φανερό, αν ο κόμβος 2 έχει μεγάλο ρυθμό αφίξεων, είναι δυνατό να μην εξυπηρετούνται ποτέ οι αιτήσεις του 3, αν και το φαινόμενο αυτό εμφανίζεται μόνο όταν ο συνολικός ρυθμός άφιξης (και των δύο κλάσεων) είναι μεγαλύτερος από το ρυθμό εξυπηρέτησης του κόμβου. Ο αλγόριθμος αυτός, παρά την έλλειψη δικαιοσύνης που διαφαίνεται, είναι ιδιαίτερα σημαντικός γιατί σε ένα σύστημα με πολλούς κόμβους μπορεί να χρησιμοποιηθεί για ανακατανομή της κίνησης και βελτιστοποίηση της απόδοσης.

Στις προσομοιώσεις που πραγματοποιήσαμε σε περιβάλλον Matlab καταλήγουμε σε ενδιαφέροντα συμπεράσματα. Σε κάθε έναν από τους αλγορίθμους θα αναφέρουμε τις τιμές που δώσαμε σαν είσοδο στο σύστημα και πιο συγκεκριμένα το συνολικό ρυθμό αφίξεων (κοινό για όλους τους κόμβους), το ρυθμό εξυπηρέτησης (επίσης κοινό), τον συνολικό αριθμό αιτήσεων που θα διαχειριστεί το σύστημα κατά την προσομοίωση και τέλος για την ειδική περίπτωση του αλγορίθμου Fixed Priority, τον πίνακα με τις προτεραιότητες.

FCFS πολιτική : Είναι η απλούστερη περίπτωση, κατά την οποία δεν γίνεται διαχωρισμός ανάμεσα στις δύο κλάσεις αιτήσεων και δεν δίνεται προτεραιότητα σε κάποια από αυτές. Η παρακάτω γραφικές παραστάσεις μας δείχνουν πως αυξομειώνεται ο χρόνος αναμονής κάθε πακέτου για κάθε έναν από τους τρεις κόμβους.



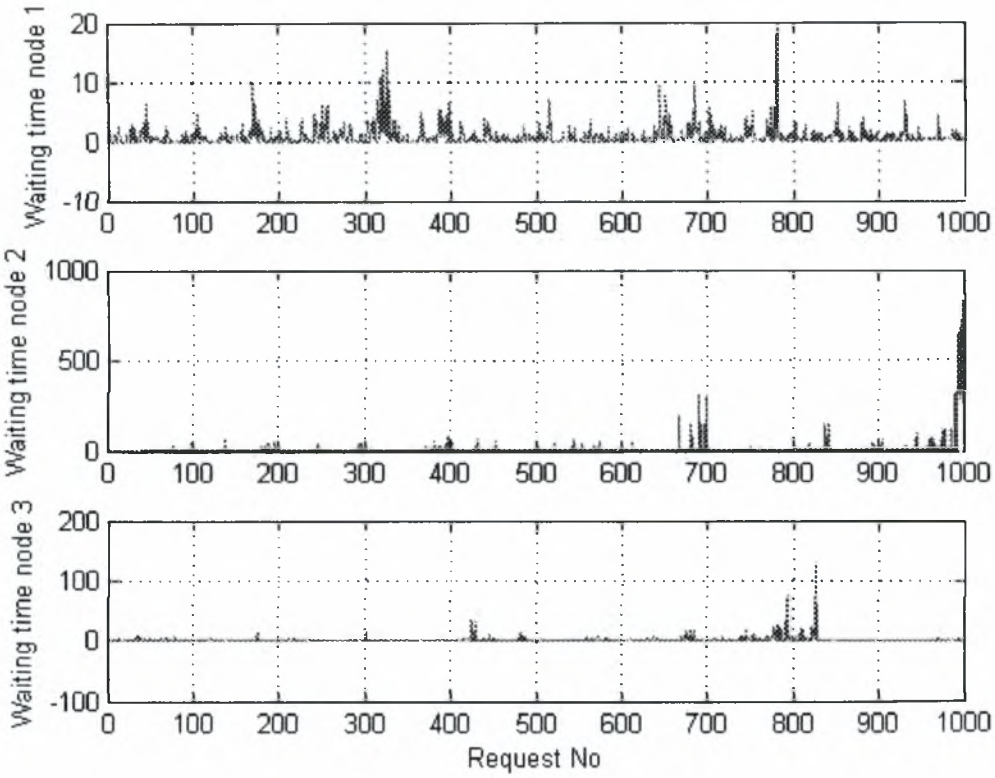
Εικόνα 7 Χρόνος αναμονής στην ουρά, κάθε αίτησης στην FCFS πολιτική

Ο συνολικός ρυθμός παραγωγής αιτήσεων των τριών κόμβων είναι 0.8 αιτ/λεπτό για καθέναν από αυτούς, ενώ μετά το σπάσιμο, οι επιμέρους τιμές αφίξεων σε κάθε κόμβο είναι : στον πρώτο κόμβο από τους άλλους δύο είναι 0.4265, του δεύτερου είναι 1.2187 και του τρίτου 0.7549 (Ισχύει ότι 0.4265+1.2187+0.7549=0.8+0.8+0.8). Ο ρυθμός εξυπηρέτησης είναι ίδιος και για τους τρεις και είναι 1 αιτ/λεπτό, ενώ ο συνολικός αριθμός αιτήσεων που θα μελετήσουμε είναι 1000 αιτήσεις. Παρατηρούμε ότι ο ρυθμός αφίξεων στον δεύτερο κόμβο είναι μεγαλύτερος του ρυθμού εξυπηρέτησης, το οποίο θεωρητικά συνεπάγεται μεγάλο μέγεθος ουράς στον εξυπηρετητή και επομένως μεγάλους χρόνους αναμονής. Τα αποτελέσματα της προσομοίωσης συμφωνούν με τα αναμενόμενα, αφού όπως παρατηρούμε από την δεύτερη γραφική παράσταση, οι χρόνοι αναμονής είναι μεγάλοι, συγκρινόμενοι με αυτούς από τους άλλους κόμβους. Ο πρώτος κόμβος, του οποίου ο ρυθμός αφίξεων είναι περίπου μισός του ρυθμού εξυπηρέτησης, εμφανίζει μικρούς χρόνους αναμονής και μάλιστα στην πλειοψηφία των αιτήσεων η ουρά είναι άδεια και οι χρόνοι αναμονής μηδενικοί. Στον τρίτο κόμβο η ουρά εμφανίζεται πιο ισορροπημένη, με μικρούς χρόνους αναμονής και μικρό αριθμό μηδενικών τιμών, γεγονός το οποίο υποδεικνύει ότι η ουρά ήταν σχετικά σπανίως άδεια. Οι μέσοι χρόνοι εξυπηρέτησης

των αιτήσεων στους τρεις κόμβους είναι 0.4246 min, 47.4396 min και 3.6016 min αντίστοιχα. Από τους χρόνους αυτούς φαίνεται και πάλι η μεγάλη καθυστέρηση στην εξυπηρέτηση των αιτήσεων που καταφθάνουν στην ουρά του δεύτερου κόμβου.

Στο σημείο αυτό αξίζει να αναφερθούμε σε ένα φαινόμενο που παρατηρείται στο μοντέλο της προσομοίωσης που υιοθετήσαμε. Παρατηρούμε στις γραφικές παραστάσεις, ότι οι τελευταίες αιτήσεις που εξυπηρετούνται από το σύστημα έχουν μικρούς χρόνους αναμονής. Ιδιαίτερα στη γραφική του δεύτερου κόμβου περιμένουμε ότι επειδή ο ρυθμός αφίξεων είναι μεγαλύτερος αυτού των εξυπηρετήσεων, η ουρά πρέπει να μεγαλώνει απεριόριστα και οι χρόνοι αναμονής να αυξάνονται συνεχώς (γεγονός το οποίο πράγματι συμβαίνει μέχρι λίγο πριν το τέλος της προσομοίωσης). Το γεγονός αυτό συμβαίνει επειδή απαιτούμε από το σύστημα να εξυπηρετήσει ακριβώς 1000 αιτήσεις σε κάθε κόμβο, από τις οποίες οι μισές (500) προέρχονται από τον ένα γειτονικό κόμβο και οι άλλες μισές από τον άλλο. Οι γειτονικοί κόμβοι, όμως, ζητάνε αρχεία με διαφορετικούς ρυθμούς με αποτέλεσμα ο κόμβος με το μεγαλύτερο ρυθμό να τελειώνει την προσομοίωση των μισών αιτήσεων που του αντιστοιχούν πιο γρήγορα απ' ότι αυτός με τον μικρότερο. Για παράδειγμα, αν θεωρήσουμε ότι ο κόμβος 2 εξυπηρετεί αιτήσεις από τους 1 και 3 με ρυθμούς λ_{12} και λ_{32} και ισχύει $\lambda_{12} > \lambda_{32}$, αυτό σημαίνει ότι ο κόμβος 2 θα τελειώσει τις αιτήσεις του 1 πριν από αυτές του 3. Για όσο διάστημα απομένει μέχρι να τελειώσουν και οι αιτήσεις του κόμβου 3, ο κόμβος 2 δέχεται συνολικά μικρότερο εισερχόμενο ρυθμό άφιξης αιτήσεων (και πιο συγκεκριμένα από $\lambda_{12} + \lambda_{32}$ που ήταν αρχικά, πέφτει στο λ_{32}), ο οποίος είναι αναγκαστικά μικρότερος του συνολικού ρυθμού εξυπηρέτησης. Για το λόγο αυτό, οι υπολειπόμενες αιτήσεις στην ουρά και οι χρόνοι αναμονής αρχίζουν να ελαττώνονται.

SJF πολιτική : Με τη χρήση του αλγορίθμου Shortest Job First δεν γίνεται διαχωρισμός ανάμεσα στις δύο κλάσεις, ακριβώς όπως και στην FCFS περίπτωση. Η επιλογή, όμως, της επόμενης αίτησης προς εξυπηρέτηση από την ουρά γίνεται με βάση το χρόνο εξυπηρέτησης και όχι με βάση την παλαιότητα. Στις επόμενες τρεις γραφικές παραστάσεις φαίνονται οι χρόνοι αναμονής στην ουρά κάθε αίτησης για τους τρεις κόμβους :



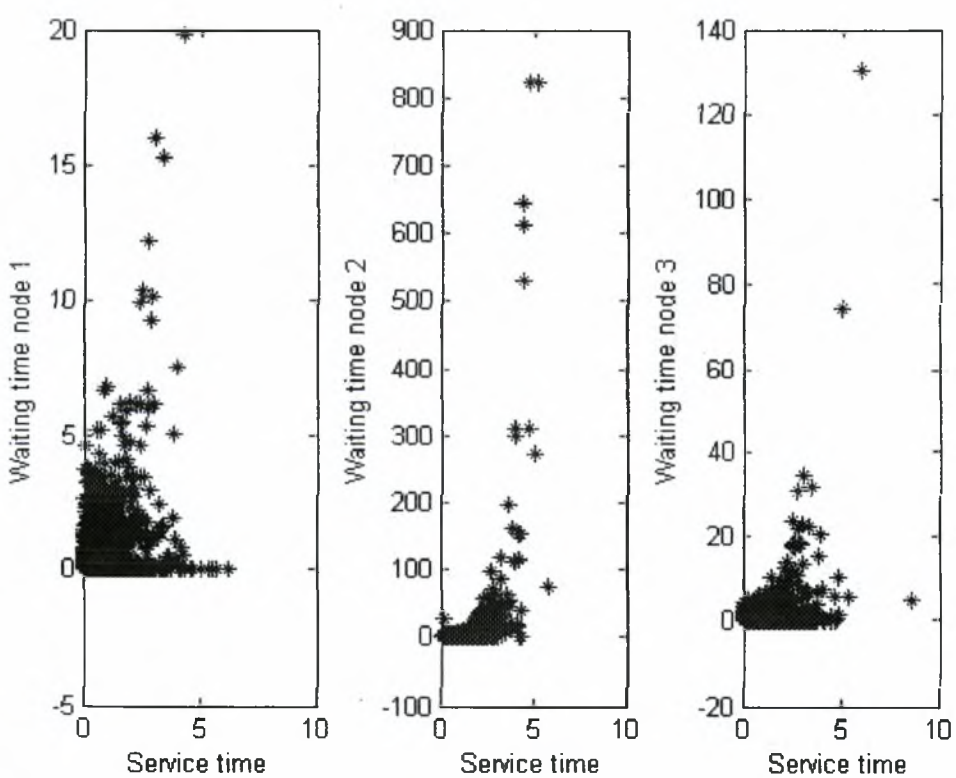
Εικόνα 8 Χρόνος αναμονής στην ουρά, κάθε αίτησης στην SJF πολιτική

Ο συνολικός ρυθμός παραγωγής αιτήσεων είναι 0.8 και για τους τρεις, ενώ μετά το σπάσιμο οι ρυθμοί αφίξεων στους τρεις κόμβους γίνονται : 0.6079 αιτ/λεπτό για τον πρώτο κόμβο, 0.9924 για τον δεύτερο και 0.7997 για τον τρίτο. Δεδομένου ότι ο ρυθμός εξυπηρέτησης και για τους τρεις κόμβους είναι 1, αναμένεται ο κόμβος 2 να εμφανίζει μεγάλους χρόνους αναμονής στην ουρά. Προσομοιώνουμε την εξυπηρέτηση 1000 αιτήσεων στο σύστημα, 500 από κάθε γειτονικό κόμβο. Με μια πρώτη παρατήρηση φαίνεται ότι οι περισσότερες αιτήσεις εξυπηρετήθηκαν αμέσως, χωρίς καθυστέρηση, ακόμα και στον κόμβο 2 όπου αναμενόταν μεγάλη συμφόρηση. Χαρακτηριστικό του αλγορίθμου αυτού είναι ότι όσο υπάρχουν μικρές αιτήσεις στην ουρά, οι μεγαλύτερες παραμελούνται και εξυπηρετούνται μόνο όταν σταματήσουν να έρχονται μικρότερες. Για το λόγο αυτό οι παραπάνω γραφικές παραστάσεις δεν είναι γραμμικές όπως στην FCFS περίπτωση, αλλά εμφανίζουν απότομες κορυφές (peaks). Όσο στο σύστημα έρχονται μικρές αιτήσεις, δίνει προτεραιότητα σ' αυτές με αποτέλεσμα να έχουν μικρό χρόνο αναμονής. Τη στιγμή που δεν θα υπάρχουν άλλες μικρές αιτήσεις, θα εξυπηρετηθούν οι μεγαλύτερες, οι οποίες λόγω του ότι παραγκωνίζονταν για αρκετό διάστημα θα έχουν ήδη μεγάλους χρόνους αναμονής. Στο σημείο αυτό της γραφικής παράστασης εμφανίζεται η απότομη κορυφή (η οποία

είναι χαρακτηριστική στη δεύτερη γραφική γύρω στην αίτηση No 700 και στην τρίτη γυρω στην αίτηση No 800).

Ένα ακόμη φαινόμενο είναι αυτό της απότομης αύξησης των χρόνων αναμονής κατά την εξυπηρέτηση των τελευταίων αιτήσεων σε κόμβους με μεγάλο ρυθμό αφίξεων. Για παράδειγμα, στον δεύτερο κόμβο ενώ οι χρόνοι είναι μικροί καθ' όλη τη διάρκεια σχεδόν της προσομοίωσης, προς το τέλος εμφανίζονται πολύ μεγάλοι. Το γεγονός αυτό συμβαίνει επειδή στον κόμβο αυτό εισέρχονται αιτήσεις με μεγάλο ρυθμό και πάντα θα υπάρχουν στην ουρά μικρές αιτήσεις που να παραγκωνίζουν τις μεγαλύτερες. Επειδή, όμως, εμφανίζεται το φαινόμενο της μείωσης του εισερχόμενου ρυθμού στον κόμβο προς το τέλος της προσομοίωσης (το οποίο εξηγήσαμε σε προηγούμενη παράγραφο), η ουρά μικραίνει κατά πολύ μέχρι να μηδενιστεί. Οι αιτήσεις που έχουν μεγάλο χρόνο εξυπηρέτησης βρίσκουν ευκαιρία να εξυπηρετηθούν προς το τέλος, εμφανίζοντας, όμως, τεράστιους χρόνους αναμονής. Γι' αυτό το λόγο, στον δεύτερο κόμβο ενώ οι χρόνοι είναι σχετικά μικροί συνεχώς, προς το τέλος γίνονται πάρα πολύ μεγάλοι.

Στις επόμενες γραφικές παραστάσεις φαίνεται η σχέση ανάμεσα στους χρόνους αναμονής κάθε αίτησης σε συνάρτηση με τους χρόνους εξυπηρέτησης.

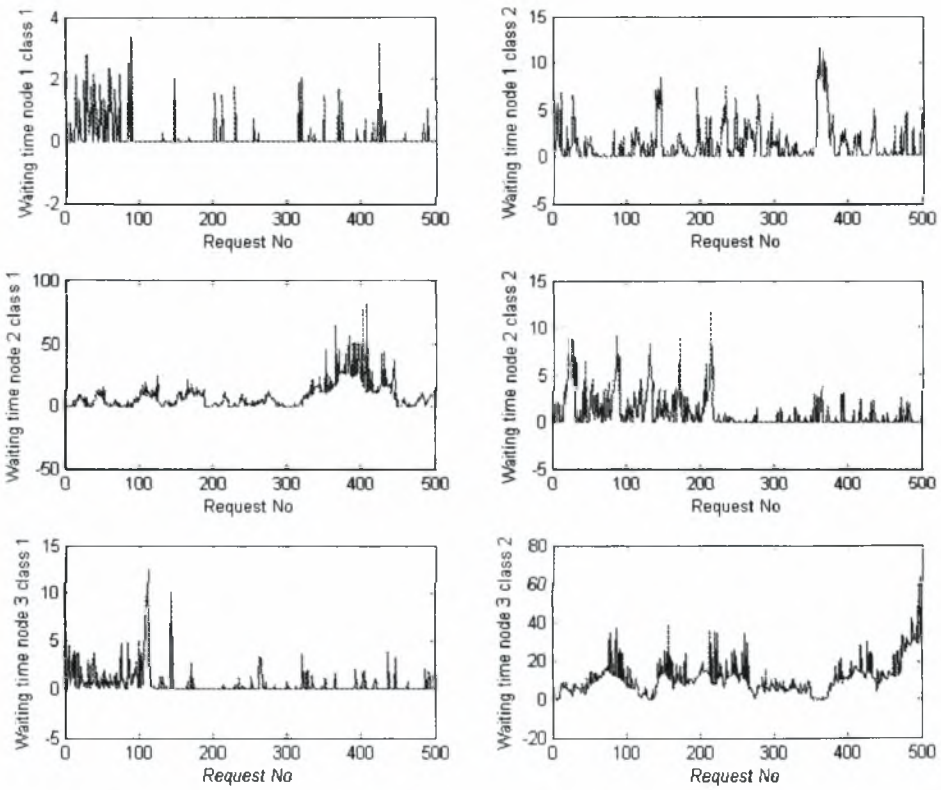


Εικόνα 9 Χρόνος αναμονής στην ουρά σε σχέση με το χρόνο εξυπηρέτησης στην SJF πολιτική

Είναι φανερό ότι οι αιτήσεις με μεγάλο χρόνο εξυπηρέτησης (της τάξης των 5 λεπτών) καθυστερούν πολύ να εξυπηρετηθούν. Στους κόμβους, όμως, όπου ο ρυθμός άφιξης είναι μικρός συγκρινόμενος με το ρυθμό εξυπηρέτησης, η ουρά είναι τον περισσότερο χρόνο άδεια και συχνά μεγάλες αιτήσεις εξυπηρετούνται αμέσως. Για το λόγο αυτό το φαινόμενο είναι πιο εμφανές σε κόμβους όπου ο ρυθμός αφίξεων είναι μεγάλος.

Πιο συγκεκριμένα, οι μέσοι χρόνοι αναμονής για κάθε κόμβο είναι 0.8340 λεπτά για τον πρώτο κόμβο, 9.3236 λεπτά για τον δεύτερο και 1.6745 για τον τρίτο. Παρατηρούμε ότι οι μέσοι χρόνοι αναμονής είναι κατά πολύ μικρότεροι από τη μέγιστη παρατηρούμενη τιμή και αυτό συμβαίνει επειδή η πλειοψηφία των αιτήσεων εξυπηρετείται σε σύντομο χρονικό διάστημα. Μπορούμε, επομένως, να παρατηρήσουμε μια σημαντική διακύμανση από τη μέση τιμή με τη χρήση της πολιτικής αυτής. Εύκολα συμπεραίνουμε ότι η πολιτική αυτή επιτυγχάνει πολύ μικρούς μέσους χρόνους, εις βάρος, όμως, των μεγάλων αιτήσεων οι οποίες καθυστερούν υπερβολικά.

Round Robin πολιτική : Η πολιτική αυτή βασίζεται στην εναλλαγή των κλάσεων κατά την επιλογή της επόμενης αίτησης προς εξυπηρέτηση. Είναι παρόμοια με την πολιτική Fixed Priority που θα μελετήσουμε στη συνέχεια, με μόνη διαφορά ότι οι προτεραιότητες εναλλάσσονται σε κάθε ολοκλήρωση μιας εξυπηρέτησης. Για το λόγο αυτό αναμένουμε η πολιτική αυτή να είναι σχετικά δίκαιη, όσον αφορά τους χρόνους αναμονής. Στις επόμενες γραφικές παραστάσεις φαίνονται οι χρόνοι αναμονής για κάθε κόμβο και για κάθε κλάση αιτήσεων (κάθε γραμμή αντιπροσωπεύει ένα κόμβο και κάθε στήλη διαφορετική κλάση) :



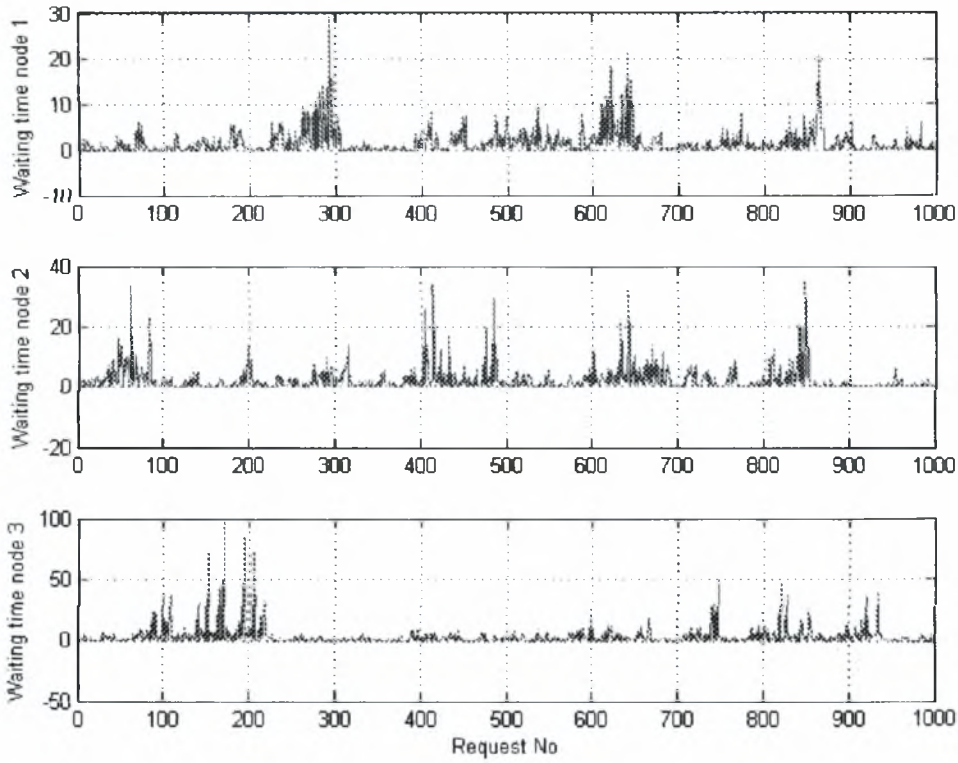
Εικόνα 10 Χρόνος αναμονής στην ουρά, κάθε αίτησης και κάθε κλάσης στην Round Robin πολιτική

Ομοίως και στην περίπτωση αυτή, οι μεταβλητές εισόδου του συστήματος είναι ίδιες με τις προηγούμενες περιπτώσεις. Ο συνολικός ρυθμός παραγωγής αιτήσεων είναι 0.8 και ο ρυθμός εξυπηρέτησης είναι 1 και για τους τρεις κόμβους. Τα σπασίματα των ρυθμών γίνονται ως εξής : στον πρώτο κόμβο εισάγεται ρυθμός 0.0699 αιτ/λεπτό από το δεύτερο και 0.5066 από τον τρίτο (αθροιστικά 0.5765 αιτ/λεπτό), στον δεύτερο κόμβο εισάγεται 0.6294 από τον πρώτο και 0.2934 από τον τρίτο (αθροιστικά 0.9228 αιτ/λεπτό) και τέλος στον τρίτο κόμβο εισέρχονται 0.1706 αιτ/λεπτό από τον πρώτο και 0.7301 από τον δεύτερο (αθροιστικά 0.9007 αιτ/λεπτό). Η προσομοίωση γίνεται για 1000 συνολικά αιτήσεις σε κάθε κόμβο.

Στις γραφικές παραστάσεις βλέπουμε την καθυστέρηση για κάθε κλάση αιτήσεων. Για παράδειγμα, οι δύο πρώτες αντιστοιχούν στην καθυστέρηση που εμφανίζεται στην ουρά του κόμβου 1 και πιο συγκεκριμένα η πρώτη γραφική περιλαμβάνει τις αιτήσεις από τον κόμβο 2, ενώ η δεύτερη αυτές του κόμβου 3. Παρατηρούμε ότι οι αιτήσεις καταφθάνουν από τον κόμβο 2 με ρυθμό 0.0699 που είναι εξαιρετικά μικρός και για το λόγο αυτό οι αιτήσεις αυτές απολαμβάνουν μικρό χρόνο αναμονής στην ουρά. Ο χρόνος αυτός θα ήταν ακόμα μικρότερος, αν από τον κόμβο 3 δεν

ερχόντουσαν αιτήσεις με ρυθμό 0.5066, που είναι αρκετά μεγάλος. Για το λόγο αυτό οι αιτήσεις του 3 παραμένουν μεγαλύτερο διάστημα στην ουρά του εξυπηρετητή. Σε γενικές γραμμές, επειδή ο συνολικός ρυθμός άφιξης του 2 και του 3 είναι 0.5765 αιτ/λεπτό, περίπου μισός του ρυθμού εξυπηρέτησης, η ουρά δεν μεγαλώνει σε μέγεθος και οι χρόνοι παραμονής είναι πολύ μικροί. Για τους κόμβους 2 και 3 η κατάσταση είναι διαφορετική, αφού ο συνολικός ρυθμός εισόδου είναι 0.9228 και 0.9007 αιτ/λεπτό αντίστοιχα (δηλαδή συγκρίσιμος με το ρυθμό εξυπηρέτησης). Στους κόμβους αυτούς παρατηρούμε μεγαλύτερους χρόνους αναμονής, οι οποίοι, όμως, διαφέρουν μεταξύ τους ανάλογα με την κλάση της κάθε αίτησης στην ουρά. Για παράδειγμα στον κόμβο 2, οι αιτήσεις του 1 καθυστερούν πολύ περισσότερο από αυτές του 3, επειδή ο ρυθμός από τον 1 είναι μεγαλύτερος του 3 (0.6294 και 0.2934 αιτ/λεπτό αντίστοιχα). Παρά, όμως, τους μεγάλους χρόνους αναμονής για μερικές αιτήσεις, οι μέσοι χρόνοι δείχνουν ότι σε γενικές γραμμές η αναμονή παραμένει σε επιτρεπτά-ανθρώπινα πλαίσια. Οι τιμές είναι : 0.7481 λεπτά για τον πρώτο κόμβο, 5.0361 για το δεύτερο και 5.8464 λεπτά για τον τρίτο. Παρατηρούμε επομένως και έμπρακτα ότι μια πολιτική τύπου Round Robin επιτρέπει μια τυπική δικαιοσύνη ανάμεσα σε διαφορετικές κλάσεις αιτήσεων, διατηρώντας παράλληλα ικανοποιητικές επιδόσεις. Στο συγκεκριμένο παράδειγμα που μελετάμε (με τρεις κόμβους), το γεγονός αυτό σημαίνει ότι από τις δύο ακολουθίες αιτήσεων που εισέρχονται σε ένα κόμβο, η ακολουθία με το μικρότερο ρυθμό αφίξεων θα απολαμβάνει μικρότερους χρόνους αναμονής στην ουρά. Σε ένα πραγματικό peer-to-peer σύστημα, κάτι τέτοιο είναι επιθυμητό, καθώς αποθαρρύνει χρήστες που ζητάνε συνεχώς αρχεία από κάποιο συγκεκριμένο κόμβο(με μεγάλο ρυθμό) και τους ενθαρρύνει να επιλέξουν κάποιον άλλο, κάνοντας έτσι κατά κάποιο τρόπο εξισορρόπηση του φόρτου εργασίας (load balancing) ανάμεσα στους peers.

Round Robin – Shortest Job First πολιτική : Όπως προαναφέρθηκε, η πολιτική αυτή δανείζεται στοιχεία από τις δύο προηγούμενες, προσπαθώντας να συνδυάσει τα πλεονεκτήματά τους. Η επιλογή της επόμενης αίτησης προς εξυπηρέτηση γίνεται εναλλάξ ανάμεσα στις δύο κλάσεις προτεραιότητας και με βάση το χρόνο εξυπηρέτησης της καθεμιάς. Οι γραφικές παραστάσεις που ακολουθούν είναι παρόμοιες με αυτές της SJF περίπτωσης και από την ανάλυσή τους θα καταλήξουμε σε ενδιαφέροντα συμπεράσματα :

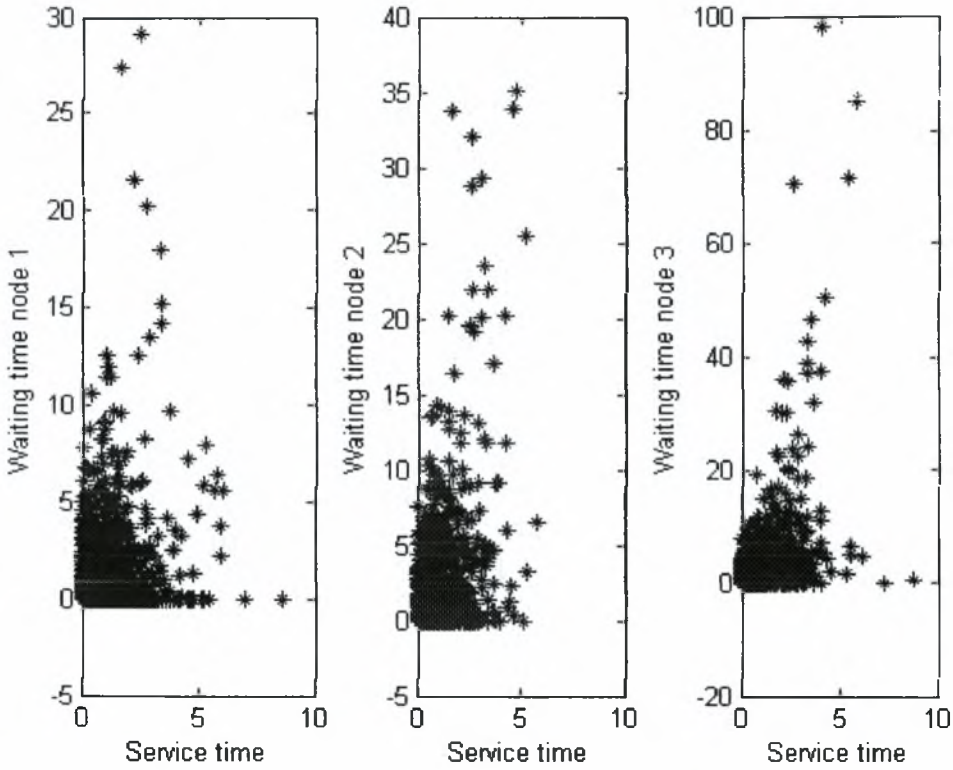


Εικόνα 11 Χρόνος αναμονής στην ουρά κάθε αίτησης στην συνδυασμένη Round Robin - SJF πολιτική

Στην προσομοίωση χρησιμοποιήθηκαν οι ίδιες μεταβλητές εισόδου με αυτές των προηγούμενων, δηλαδή 0.8 αιτ/λεπτό ρυθμός παραγωγής αιτήσεων, 1 αιτ/λεπτό ρυθμός εξυπηρέτησης, 1000 αιτήσεις. Μετά το σπάσιμο των ρυθμών παραγωγής, οι ρυθμοί αφίξεων στις ουρές των εξυπηρετητών γίνονται : 0.6589 αιτ/λεπτό για τον πρώτο κόμβο, 0.8373 για το δεύτερο και 0.9039 αιτ/λεπτό για τον τρίτο. Οι τιμές αυτές είναι παρόμοιες με αυτές που χρησιμοποιήσαμε στην SJF περίπτωση και θα συμβάλλουν στην εξαγωγή συμπερασμάτων σχετικά με την απόδοση του αλγορίθμου.

Με μια πρώτη παρατήρηση φαίνεται ότι οι μέγιστες τιμές χρόνου αναμονής στην ουρά είναι κατά πολύ μικρότερες από αυτές της SJF πολιτικής. Από την άλλη, όμως, είναι λίγες οι περιπτώσεις κατά τις οποίες ο χρόνος αναμονής είναι μηδενικός. Με άλλα λόγια η διασπορά των τιμών γύρω από τη μέση τιμή είναι κατά πολύ μικρότερη από την SJF, παρ' όλο που, όπως θα δούμε, η μέση τιμή είναι αυξημένη. Πράγματι, οι μέσες τιμές των χρόνων αναμονής στην ουρά κάθε κόμβου είναι : 1.4677 λεπτά για τον πρώτο κόμβο, 2.2403 για το δεύτερο και 3.2370 λεπτά για τον τρίτο. Παρά το γεγονός ότι είναι αυξημένοι σε σχέση με την SJF πολιτική, εντούτοις

παραμένουν αποδεκτοί, σε λογικά πλαίσια. Επίσης, οι μέγιστες τιμές είναι πράγματι μειωμένες, γεγονός που αποδεικνύει ότι ακόμα και οι μεγάλες αιτήσεις(που απαιτούν μεγάλους χρόνους εξυπηρέτησης) έχουν δυνατότητα να εξυπηρετηθούν σχετικά γρήγορα. Οι επόμενες γραφικές παραστάσεις δείχνουν τη σχέση ανάμεσα στο χρόνο εξυπηρέτησης και το χρόνο αναμονής στην ουρά :



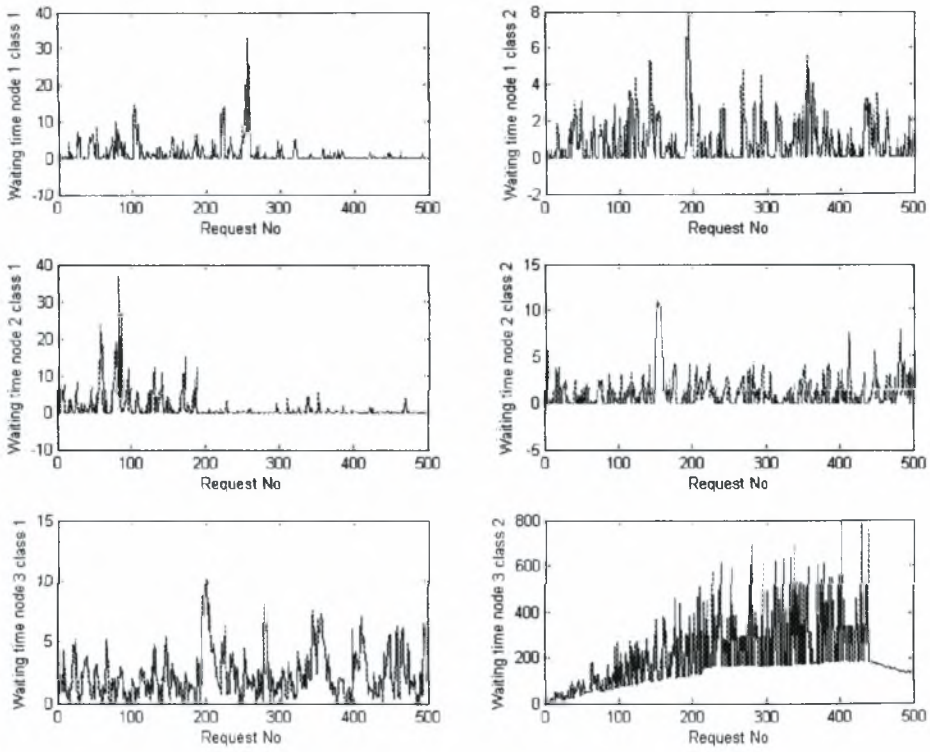
Εικόνα 12 Χρόνος αναμονής στην ουρά σε σχέση με το χρόνο εξυπηρέτησης στην Round Robin-SJF πολιτική

Στις παραστάσεις αυτές παρατηρούμε ότι οι τιμές του χρόνου αναμονής απομακρύνθηκαν από το μηδέν σε σχέση με τις παραστάσεις της SJF πολιτικής, αλλά παράλληλα μειώθηκαν και οι μέγιστες τιμές (έπειτα από παρατήρηση στον άξονα y). Αυτό αποδεικνύει περαιτέρω την παρατήρηση της μείωσης της διασποράς των τιμών. Συμπερασματικά, μπορούμε να πούμε ότι τα σημεία της γραφικής παράστασης, σε σχέση πάντα με τα σημεία των γραφικών της SJF πολιτικής, έχουν «ανοίξει» πάνω στους άξονες και δεν συγκεντρώνονται κοντά στην τιμή μηδέν. Μπορούμε να πούμε, επομένως, ότι τα χαρακτηριστικά της πολιτικής αυτής την κάνουν να υπερτερεί έναντι των προηγούμενων, γεγονός που θα μελετηθεί περισσότερο σε επόμενη παράγραφο.

Fixed priority πολιτική : Με την πολιτική αυτή, καθορίζουμε εκ των προτέρων ποιες ακριβώς θα είναι οι προτεραιότητες ανάμεσα στους κόμβους και για το λόγο αυτό χρειάζεται να τις δώσουμε στο σύστημα. Για το λόγο αυτό χρησιμοποιούμε τρία διανύσματα 2x1 στα οποία δίνεται ακριβώς αυτή η πληροφορία. Κάθε διάνυσμα αναγράφει στην πρώτη στήλη τον κόμβο με τη μεγαλύτερη προτεραιότητα και στη δεύτερη αυτόν με τη μικρότερη. Ένα παράδειγμα τέτοιων διανυσμάτων είναι το παρακάτω :

$$\{\{3,2\} \quad \{3,1\} \quad \{1,3\}\}$$

Στις παρακάτω γραφικές παραστάσεις, φαίνονται οι χρόνοι αναμονής στην ουρά σε κάθε κόμβο και για κάθε κλάση προτεραιότητας. Όμοια με τις προηγούμενες πολιτικές, επιλέγουμε συνολικό ρυθμό παραγωγής αιτήσεων σε κάθε κόμβο 0.8, ρυθμό εξυπηρέτησης 1, προσομοίωση 1000 αιτήσεων στο σύστημα και τέλος χρησιμοποιούμε τα διανύσματα προτεραιοτήτων που είδαμε πιο πάνω (δηλαδή ο κόμβος 1 δίνει μεγαλύτερη προτεραιότητα στον 3, ο 2 στον 3 και ο 3 στον 1). Μετά το τυχαίο σπάσιμο των ρυθμών παραγωγής, οι ρυθμοί αφίξεων γίνονται : 0.5884 για τον πρώτο κόμβο, 0.5928 για το δεύτερο και 1.2188 για τον τρίτο. Παρατηρούμε μεγάλο ρυθμό αφίξεων στον τρίτο κόμβο, γεγονός που μας κάνει να αναμένουμε μεγάλους χρόνους αναμονής στην ουρά. Πράγματι αυτό επιβεβαιώνεται και από τις γραφικές παραστάσεις :



Εικόνα 13 Χρόνος αναμονής στην ουρά, κάθε αίτησης και κάθε κλάσης στην Fixed Priority πολιτική

Πράγματι παρατηρούμε ότι οι τιμές καθυστέρησης των αιτήσεων του κόμβου 2 από τον 1 είναι αρκετά μεγαλύτερες από αυτές του 3 από τον 1 (πρώτες 2 γραφικές παραστάσεις), επειδή ο 1 δίνει μεγαλύτερη προτεραιότητα στον 3 απ' ότι στον 2. Ομοίως και ο 2 εξυπηρετεί τις αιτήσεις του 3 πολύ πιο γρήγορα από αυτές του 1. Τα πιο ενδιαφέροντα συμπεράσματα εξάγουμε από τη μελέτη του τρίτου κόμβου, του οποίου ο ρυθμός αφίξεων ήταν μεγαλύτερος του ρυθμού εξυπηρέτησης (τελευταίες 2 γραφικές παραστάσεις). Ο κόμβος 3 δίνει μεγαλύτερη προτεραιότητα στις αιτήσεις του 1 και γι' αυτό οι αιτήσεις του 1 δεν αναμένουν πολύ στην ουρά (μερικές μάλιστα δεν περιμένουν καθόλου στην ουρά). Αντιθέτως οι αιτήσεις του 2 παραμελούνται και εμφανίζουν μεγάλους χρόνους αναμονής. Οι μεγάλες αυξομειώσεις στους χρόνους των αιτήσεων του κόμβου 2 από τον 3 (τελευταίο διάγραμμα) οφείλεται στο γεγονός ότι συχνά η ουρά δεν περιείχε αιτήσεις του κόμβου 1 και γι' αυτό ο 3 εξυπηρετούσε επανελλειμένως αιτήσεις του 2, μειώνοντας το χρόνο αναμονής τους (όταν η ουρά δεν έχει αιτήσεις από τον κόμβο προτεραιότητας, τότε εξυπηρετούνται οι υπολειπόμενες αιτήσεις).

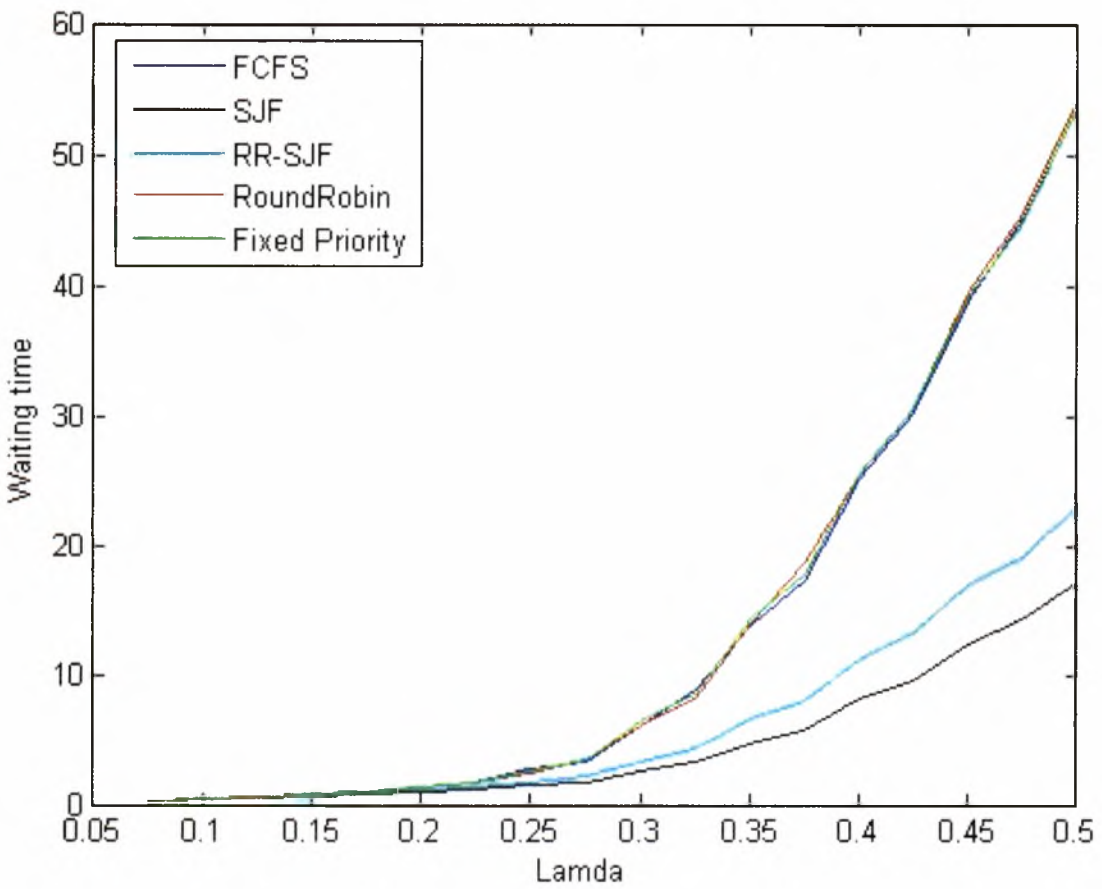
Μια ενδιαφέρουσα παραλλαγή του παραπάνω αλγορίθμου θα μπορούσε να είναι η επιλογή της επόμενης κλάσης αιτήσεων πιθανοτικά και όχι ντετερμινιστικά. Με βάση, δηλαδή, τα διανύσματα προτεραιότητας, αντί να επιλέγονται πάντα οι αιτήσεις του κόμβου 2 έναντι του 3 (όταν κόμβος 1 δίνει μεγαλύτερη προτεραιότητα στον 2 απ' ότι στον 3), να επιλέγεται η παλαιότερη αίτηση του κόμβου 2 με βάση κάποια πιθανότητα (π.χ. 60%) και η παλαιότερη του 3 με τη συμπληρωματική της (40%). Με τον τρόπο αυτό διατηρείται η έννοια της προτεραιότητας (δεδομένου ότι οι αιτήσεις του 2 θα εξυπηρετούνται πιο συχνά από του 3, λόγω μεγαλύτερης πιθανότητας), χωρίς όμως η πολιτική fixed priority να είναι τόσο απόλυτη στην επιλογή της αίτησης. Επιτυγχάνεται με άλλα λόγια, μεγαλύτερη δικαιοσύνη στην επιλογή, αφού οι αιτήσεις του κόμβου 3 δεν θα περιμένουν επ' αόριστον μέχρι να ολοκληρωθούν αυτές του 2.

Συγκριτική Προσομοίωση

Για να μπορέσουμε να αναλογιστούμε καλύτερα τις διαφορές στην απόδοση των αλγορίθμων που μελετήσαμε στις προηγούμενες παραγράφους, είναι απαραίτητη η εκτέλεση μιας ταυτόχρονης προσομοίωσης όλων αυτών. Ως μέτρο σύγκρισης της απόδοσης κάθε αλγορίθμου θεωρούμε τη μέση τιμή της αναμονής στην ουρά, που επιτυγχάνεται για διαφορετικές τιμές του ρυθμού παραγωγής αιτήσεων κάθε κόμβου. Διατηρώντας σταθερό το ρυθμό εξυπηρέτησης αιτήσεων στους κόμβους(και ίδιο για όλους) και αυξάνοντας σταδιακά το ρυθμό παραγωγής (επίσης ίδιο για όλους τους κόμβους), επιτυγχάνουμε να μεταβάλλουμε σταδιακά το φόρτο της ουράς. Για μικρό ρυθμό παραγωγής, οι ουρές παραμένουν το μεγαλύτερο χρόνο άδειες και οι χρόνοι αναμονής μικροί, ενώ όσο μεγαλώνουμε το ρυθμό αυξάνεται ο φόρτος και συνεπακόλουθα η καθυστέρηση των αιτήσεων στις ουρές. Επειδή το σύστημα περιλαμβάνει τρεις κόμβους και κάθε κόμβος έχει το δικό του μέσο χρόνο αναμονής(λόγω του ότι σε κάθε κόμβο εισέρχεται διαφορετική ποσότητα κίνησης) χρησιμοποιούμε το μέσο όρο των τριών αυτών χρόνων, για να χρησιμοποιηθεί σαν τιμή που επιτυγχάνει ο αλγόριθμος. Επίσης, όπως είδαμε σημαντικό ρόλο στην απόδοση παίζει και το σπάσιμο των ρυθμών παραγωγής αιτήσεων, διότι εξαιτίας του σπασίματος αυτού εισάγονται αιτήσεις με διαφορετικό ρυθμό στην ουρά κάθε κόμβου. Για το λόγο αυτό, πραγματοποιούμε όλα τα δυνατά σπασίματα με πολύ

μικρό βήμα και εκτελούμε σε κάθε βήμα προσομοίωση των αλγορίθμων παίρνοντας τις τιμές των χρόνων αναμονής (το μέσο όρο των τριών χρόνων αναμονής κάθε κόμβου όπως εξηγήσαμε πιο πριν). Οι τιμές αποθηκεύονται σε πίνακες, οπότε μετά την ολοκλήρωση της προσομοίωσης για όλα τα δυνατά σπασίματα, προκύπτει ο μέσος όρος όλων των δυνατών σπασιμάτων. Με άλλα λόγια, για κάθε τιμή του ρυθμού παραγωγής αιτήσεων (από μηδέν μέχρι το ρυθμό εξυπηρέτησης) προκύπτει μια τιμή για το μέσο χρόνο αναμονής, που είναι ο μέσος όρος για όλα τα σπασίματα και για τους τρεις κόμβους. Τα ζευγάρια αυτά των τιμών, χρησιμοποιούμε για να παράγουμε τη γραφική παράσταση των χρόνων αναμονής σε συνάρτηση με το συνολικό ρυθμό παραγωγής αιτήσεων.

Πραγματοποιώντας τη συγκριτική προσομοίωση με τον τρόπο που περιγράψαμε καταλήγουμε στην παρακάτω γραφική παράσταση, στην οποία φαίνεται και στην πράξη μερικά από τα συμπεράσματα στα οποία είχαμε καταλήξει από τις επιμέρους προσομοιώσεις των αλγορίθμων :



Εικόνα 14 Χρόνος αναμονής στην ουρά σε συνάρτηση με το ρυθμό παραγωγής αιτήσεων για κάθε πολιτική

Φαίνεται, λοιπόν, με ποιο τρόπο αυξάνεται ο μέσος χρόνος αναμονής των τριών κόμβων, για όλα τα δυνατά σπασίματα σε συνάρτηση με το ρυθμό παραγωγής αιτήσεων, όταν αυτός αυξάνεται από μηδέν μέχρι τη μέγιστη τιμή που έχουμε ορίσει ότι θα είναι ίση με το ρυθμό εξυπηρέτησης. Με τον τρόπο αυτό φαίνεται πως συμπεριφέρεται το σύστημα ακόμα και στην ακραία κατάσταση κατά την οποία οι ρυθμοί παραγωγής αιτήσεων είναι ίσοι με τους ρυθμούς εξυπηρέτησης. Πιο συγκεκριμένα, πραγματοποιήσαμε την προσομοίωση για ρυθμό εξυπηρέτησης ίσο με 0.5 αιτ/λεπτό και 500 πακέτα σε κάθε κόμβο. Ο ρυθμός παραγωγής αιτήσεων αυξανόταν από την τιμή 0.75 (δεν χρησιμοποιήσαμε ακριβώς μηδέν γιατί θα ήταν περιττό) μέχρι το ρυθμό εξυπηρέτησης με βήματα των 0.025. Η χρήση μικρότερου βήματος οδηγεί στον υπολογισμό περισσότερων σημείων στη γραφική παράσταση.

Παρατηρούμε εύκολα ότι για μικρές τιμές του ρυθμού παραγωγής, οι χρόνοι αναμονής είναι ιδιαίτερα μικροί για όλους τους αλγορίθμους. Από τη στιγμή, όμως, που η τιμή φτάνει το μισό του ρυθμού εξυπηρέτησης, οι μέσοι χρόνοι αναμονής για τους Round Robin, First Come First Served και Fixed Priority αυξάνονται πιο γρήγορα απ' ότι των RoundRobin-SJF και Shortest Job First. Το γεγονός αυτό δίνει ένα συγκριτικό πλεονέκτημα στις δύο αυτές τελευταίες πολιτικές, καθώς αποδεικνύει ότι κάτω από συνθήκες φόρτου αποδίδουν καλύτερα. Επίσης, φαίνεται ότι οι τρεις πρώτες πολιτικές (Round Robin, FCFS και Fixed Priority) δεν έχουν διαφορές στην απόδοση, όταν δε μας ενδιαφέρει από ποιο κόμβο προήλθαν οι αιτήσεις. Σε περίπτωση, όμως, που κάτι τέτοιο μας ενδιαφέρει, τότε τα αποτελέσματα είναι διαφορετικά, διότι ιδιαίτερα στην Fixed Priority πολιτική, από τις δυο κλάσεις αιτήσεων που εισέρχονται στον κόμβο, μόνο η μία απολαμβάνει μικρούς χρόνους αναμονής.

Όπως αναφέραμε και προηγουμένως, σκοπός ενός peer-to-peer συστήματος δεν είναι απλά να προσφέρει στους χρήστες του τους μικρότερους δυνατούς χρόνους αναμονής, αλλά να προσφέρει και επαρκή δικαιοσύνη.

Βέλτιστο σπάσιμο ρυθμών παραγωγής αιτήσεων

Μία ακόμα σημαντική προσομοίωση από την οποία μπορούμε να εξάγουμε σημαντικά συμπεράσματα, είναι αυτή που θα εκτελέσουμε στη συνέχεια. Επιθυμούμε

να βρούμε το βέλτιστο σπάσιμο των ρυθμών παραγωγής κάθε κόμβου, ώστε να επιτύχουμε τη μικρότερη δυνατή καθυστέρηση στις ουρές των εξυπηρετητών. Σαν καθυστέρηση θεωρούμε το μέσο όρο των τριών χρόνων αναμονής στις ουρές των κόμβων. Υπολογίζουμε την καθυστέρηση αυτή για διαφορετικά σπασίματα των ρυθμών παραγωγής και παρατηρούμε για ποιο σπάσιμο πήραμε την ελάχιστη τιμή. Για την FCFS πολιτική αρχικά, προσομοιώσαμε την άφιξη 500 αιτήσεων σε κάθε κόμβο, με μεταβαλλόμενο ρυθμό παραγωγής που κυμαινόταν από 0.075 αιτ/λεπτό μέχρι το ρυθμό εξυπηρέτησης, τον οποίο θέσαμε ίσο με 0.55 αιτ/λεπτό (με βήμα αύξησης 0.025). Για κάθε μία από τις τιμές του ρυθμού αυτού, υπολογίζουμε το μέσο χρόνο καθυστέρησης των τριών κόμβων για όλα τα δυνατά σπασίματα και αποθηκεύουμε τη μικρότερη τιμή και το σπάσιμο στο οποίο εμφανίστηκε. Κατά την εκτέλεση της προσομοίωσης προκύπτουν τα παρακάτω μηνύματα :

*lamda=0.075 Minimum waiting time=**0.2415** at* $\lambda_{12}=0.05$, $\lambda_{23}=0.05$, $\lambda_{31}=0.05$

*lamda=0.100 Minimum waiting time=**0.4304** at* $\lambda_{12}=0.05$, $\lambda_{23}=0.05$, $\lambda_{31}=0.05$

*lamda=0.125 Minimum waiting time=**0.3812** at* $\lambda_{12}=0.05$, $\lambda_{23}=0.10$, $\lambda_{31}=0.05$

*lamda=0.150 Minimum waiting time=**0.4397** at* $\lambda_{12}=0.10$, $\lambda_{23}=0.10$, $\lambda_{31}=0.10$

*lamda=0.175 Minimum waiting time=**0.5186** at* $\lambda_{12}=0.05$, $\lambda_{23}=0.05$, $\lambda_{31}=0.05$

*lamda=0.200 Minimum waiting time=**0.6040** at* $\lambda_{12}=0.10$, $\lambda_{23}=0.05$, $\lambda_{31}=0.05$

*lamda=0.225 Minimum waiting time=**0.6564** at* $\lambda_{12}=0.20$, $\lambda_{23}=0.20$, $\lambda_{31}=0.20$

*lamda=0.250 Minimum waiting time=**0.8925** at* $\lambda_{12}=0.20$, $\lambda_{23}=0.20$, $\lambda_{31}=0.15$

*lamda=0.275 Minimum waiting time=**0.8898** at* $\lambda_{12}=0.25$, $\lambda_{23}=0.25$, $\lambda_{31}=0.25$

*lamda=0.300 Minimum waiting time=**1.2174** at* $\lambda_{12}=0.25$, $\lambda_{23}=0.25$, $\lambda_{31}=0.25$

*lamda=0.325 Minimum waiting time=**1.0853** at* $\lambda_{12}=0.05$, $\lambda_{23}=0.05$, $\lambda_{31}=0.10$

*lamda=0.350 Minimum waiting time=**1.2937** at* $\lambda_{12}=0.30$, $\lambda_{23}=0.30$, $\lambda_{31}=0.30$

*lamda=0.375 Minimum waiting time=**1.7652** at* $\lambda_{12}=0.30$, $\lambda_{23}=0.30$, $\lambda_{31}=0.35$

*lamda=0.400 Minimum waiting time=**1.6430** at* $\lambda_{12}=0.35$, $\lambda_{23}=0.35$, $\lambda_{31}=0.35$

*lamda=0.425 Minimum waiting time=**2.3912** at* $\lambda_{12}=0.40$, $\lambda_{23}=0.40$, $\lambda_{31}=0.40$

*lamda=0.450 Minimum waiting time=**2.6192** at* $\lambda_{12}=0.05$, $\lambda_{23}=0.05$, $\lambda_{31}=0.10$

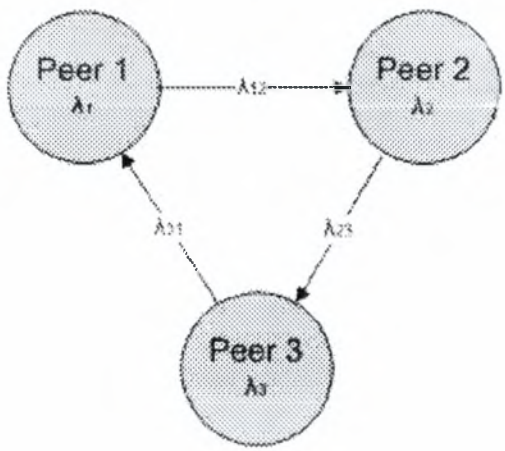
*lamda=0.475 Minimum waiting time=**3.1180** at* $\lambda_{12}=0.45$, $\lambda_{23}=0.40$, $\lambda_{31}=0.40$

*lamda=0.500 Minimum waiting time=**3.6612** at* $\lambda_{12}=0.35$, $\lambda_{23}=0.40$, $\lambda_{31}=0.45$

*lamda=0.525 Minimum waiting time=**4.7108** at* $\lambda_{12}=0.45$, $\lambda_{23}=0.50$, $\lambda_{31}=0.40$

*lamda=0.550 Minimum waiting time=**8.2202** at* $\lambda_{12}=0.40$, $\lambda_{23}=0.40$, $\lambda_{31}=0.45$

όπου με σκούρα γράμματα φαίνονται οι ελάχιστοι χρόνοι αναμονής που επετεύχθησαν και από δίπλα οι τιμές που αντιπροσωπεύουν τα αντίστοιχα σπασίματα. Παρατηρούμε κατ' αρχάς πως αυξάνεται ο ελάχιστος χρόνος αναμονής όσο πλησιάζει ο ρυθμός παραγωγής τον ρυθμό εξυπηρέτησης. Εννοείται πως οι μέσες τιμές βρίσκονται πολύ πιο πάνω από τις ελάχιστες. Μπορούμε, επίσης, να κάνουμε και την εξής παρατήρηση : οι τιμές για τα σπασίματα των ρυθμών παραγωγής για τα οποία επιτυγχάνεται ελάχιστο, είναι λίγο-πολύ ίσες. Οι τιμές λ_{12}, λ_{23} και λ_{31} που επιλέξαμε να εκτυπώσουμε είναι οι τιμές των ρυθμών άφιξης από τον ένα κόμβο στον άλλο σε κυκλική μορφή, όπως στο παρακάτω σχήμα:



Εικόνα 15 Στιγμιότυπο όπου φαίνονται τα υπό μελέτη σπασίματα των ρυθμών παραγωγής

Όταν, επομένως, οι τιμές αυτές είναι ίσες, σημαίνει ότι οι ρυθμοί άφιξης σε κάθε κόμβο είναι ίσες, επειδή οι τιμές λ_{21}, λ_{32} και λ_{13} είναι συμπληρωματικές αυτών που εκτυπώσαμε. Με άλλα λόγια, οι ελάχιστες τιμές των χρόνων αναμονής στην ουρά εμφανίζονται όταν έχουμε εξισορρόπηση των ρυθμών άφιξης σε κάθε κόμβο, ή, δηλαδή, όταν ο φόρτος εργασίας είναι ίδιος και στους τρεις κόμβους.

Πραγματοποιώντας την ίδια προσομοίωση, με τις ίδιες παραμέτρους εισόδου, χρησιμοποιώντας την Shortest Job First πολιτική καταλήγουμε στα ίδια ακριβώς συμπεράσματα, με μόνη ίσως διαφορά το γεγονός ότι οι ελάχιστες τιμές είναι μικρότερες από εκείνες της FCFS πολιτικής.

*lamda=0.075 Minimum waiting time=**0.2176** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*
*lamda=0.100 Minimum waiting time=**0.3618** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*
*lamda=0.125 Minimum waiting time=**0.3001** at* λ_{12}*=0.10,* λ_{23}*=0.10,* λ_{31}*=0.05*
*lamda=0.150 Minimum waiting time=**0.4590** at* λ_{12}*=0.10,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.175 Minimum waiting time=**0.4419** at* λ_{12}*=0.15,* λ_{23}*=0.15,* λ_{31}*=0.15*

*lamda=0.200 Minimum waiting time=**0.5993** at* λ_{12}*=0.15,* λ_{23}*=0.15,* λ_{31}*=0.15*

*lamda=0.225 Minimum waiting time=**0.5919** at* λ_{12}*=0.20,* λ_{23}*=0.20,* λ_{31}*=0.20*

*lamda=0.250 Minimum waiting time=**0.6110** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.275 Minimum waiting time=**0.7408** at* λ_{12}*=0.25,* λ_{23}*=0.20,* λ_{31}*=0.25*

*lamda=0.300 Minimum waiting time=**0.7397** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.325 Minimum waiting time=**0.9697** at* λ_{12}*=0.30,* λ_{23}*=0.30,* λ_{31}*=0.30*

*lamda=0.350 Minimum waiting time=**0.9937** at* λ_{12}*=0.10,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.375 Minimum waiting time=**0.9819** at* λ_{12}*=0.35,* λ_{23}*=0.35,* λ_{31}*=0.35*

*lamda=0.400 Minimum waiting time=**1.2161** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.425 Minimum waiting time=**1.2607** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.450 Minimum waiting time=**1.6623** at* λ_{12}*=0.05,* λ_{23}*=0.05,* λ_{31}*=0.05*

*lamda=0.475 Minimum waiting time=**1.8122** at* λ_{12}*=0.40,* λ_{23}*=0.45,* λ_{31}*=0.45*

*lamda=0.500 Minimum waiting time=**1.9279** at* λ_{12}*=0.05,* λ_{23}*=0.10,* λ_{31}*=0.05*

*lamda=0.525 Minimum waiting time=**2.0585** at* λ_{12}*=0.45,* λ_{23}*=0.45,* λ_{31}*=0.40*

*lamda=0.550 Minimum waiting time=**2.5494** at* λ_{12}*=0.10,* λ_{23}*=0.05,* λ_{31}*=0.05*

Μπορούμε, επομένως, με ασφάλεια να καταλήξουμε στο συμπέρασμα ότι το σύστημα συμπεριφέρεται καλύτερα και επιτυγχάνει καλύτερους(μικρότερους) χρόνους αναμονής στην ουρά, όταν ο φόρτος είναι εξίσου μοιρασμένος στους κόμβους που το αποτελούν. Μια ακόμα ενδιαφέρουσα παρατήρηση είναι η παρακάτω: όπως είδαμε, οι βέλτιστες τιμές αναμονής προκύπτουν όταν οι ρυθμοί αφίξεων σε κάθε κόμβο είναι ίσοι. Επειδή, όμως, το άθροισμα των ρυθμών αφίξεων πρέπει να είναι ίσο με το άθροισμα των ρυθμών παραγωγής, προκύπτει το παρακάτω σύστημα :

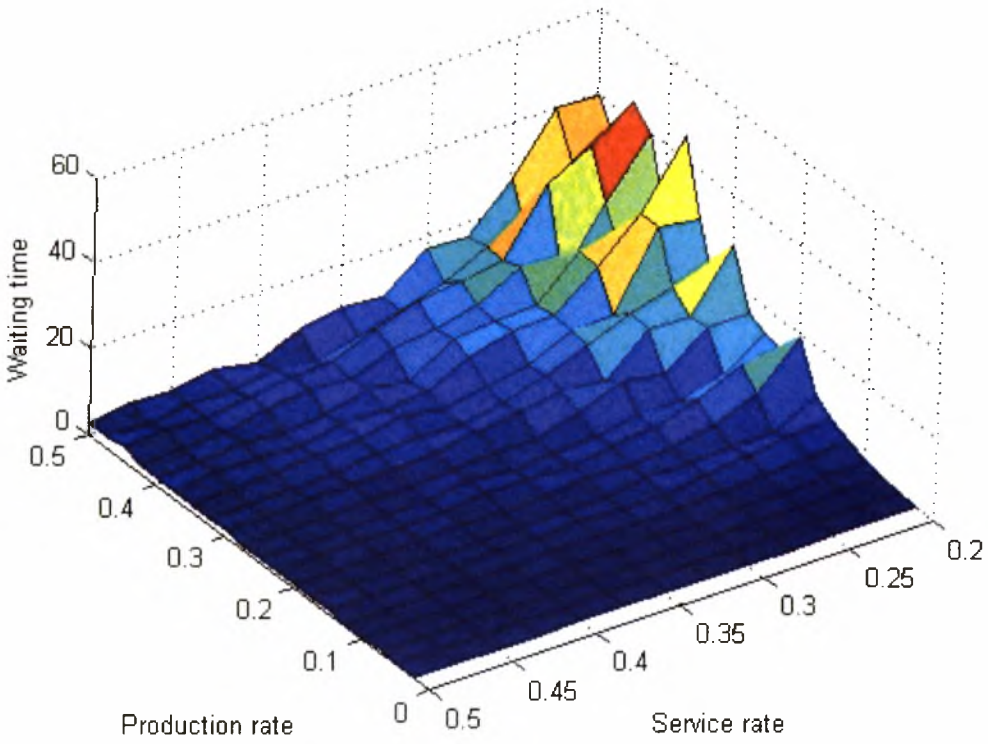
x + y + z = 3a (αρχή διατήρησης) και x = y = z (ίσος φόρτος)

όπου x, y, z οι ρυθμοί άφιξης σε κάθε κόμβο και a ο κοινός ρυθμός παραγωγής αιτήσεων. Προκύπτει, δηλαδή, ότι x = y = z = a, οπότε στην περίπτωσή μας που χρησιμοποιήσαμε ρυθμό παραγωγής 0.55, πρέπει το άθροισμα των ρυθμών αφίξεων σε κάθε κόμβο να είναι ίσο με το ρυθμό παραγωγής αιτήσεων του κόμβου αυτού. Το γεγονός αυτό συμβαίνει και στα δύο παραπάνω παραδείγματα που χρησιμοποιήσαμε, με σχετικά μεγάλη ακρίβεια. Για παράδειγμα στην πρώτη γραμμή του τελευταίου παραδείγματος(κόκκινου χρώματος), έχουμε λ_{12}=0.05 και λ_{31}=0.05. Επειδή ο ρυθμός παραγωγής είναι 0.55 αιτ/λεπτό, έχουμε ότι λ_{32}= 0.55-λ_{31}=0.5 και επομένως ο ρυθμός

με τον οποίο καταφθάνουν αιτήσεις στον κομβο 2 είναι $\lambda_{12} + \lambda_{32} = 0.05 + 0.5 = 0.55$ αιτ/λεπτό, που είναι ίσο με το ρυθμό παραγωγής, για το λόγο που εξηγήσαμε. Με μικρές αποκλίσεις, το ίδιο ισχύει για κάθε γραμμή των παραδειγμάτων, δηλαδή για κάθε τιμή του ρυθμού παραγωγής.

Σχέση χρόνου αναμονής με ρυθμό εξυπηρέτησης και ρυθμό παραγωγής αιτήσεων

Μια ακόμα ενδιαφέρουσα προσομοίωση είναι και αυτή που μας δείχνει με ποιο τρόπο σχετίζεται ο χρόνος αναμονής στην ουρά σε συνάρτηση με το ρυθμό εξυπηρέτησης και το ρυθμό παραγωγής αιτήσεων. Για το λόγο αυτό θεωρήσαμε το σύστημα που χρησιμοποιήσαμε στις προηγούμενες προσομοιώσεις, μόνο που αυτή τη φορά δεν διατηρούσαμε σταθερό το ρυθμό εξυπηρέτησης, αλλά τον μεταβάλλαμε σταδιακά ανάμεσα σε δύο προκαθορισμένες τιμές. Ξεκινώντας, λοιπόν, από μια αρχική τιμή (0.2 αιτ/λεπτό στο παράδειγμά μας) αυξήσαμε το ρυθμό εξυπηρέτησης μέχρι την τιμή 0.5, με βήμα 0.025. Για κάθε μία από τις τιμές της μεταβλητής αυτής, πήραμε όλες τις δυνατές τιμές του ρυθμού παραγωγής από 0.025 μέχρι 0.5 αιτ/λεπτό(με βήμα 0.025) και υπολογίσαμε τις τιμές του χρόνου αναμονής στην ουρά για κάθε ζευγάρι τιμών ρυθμού εξυπηρέτησης και παραγωγής αιτήσεων. Με τον τρόπο αυτό πήραμε τριάδες τιμών, τις οποίες μπορούμε να αναπαραστήσουμε σε τρισδιάστατη γραφική παράσταση, όπως φαίνεται παρακάτω (χρησιμοποιήσαμε ουρές SJF πολιτικής):



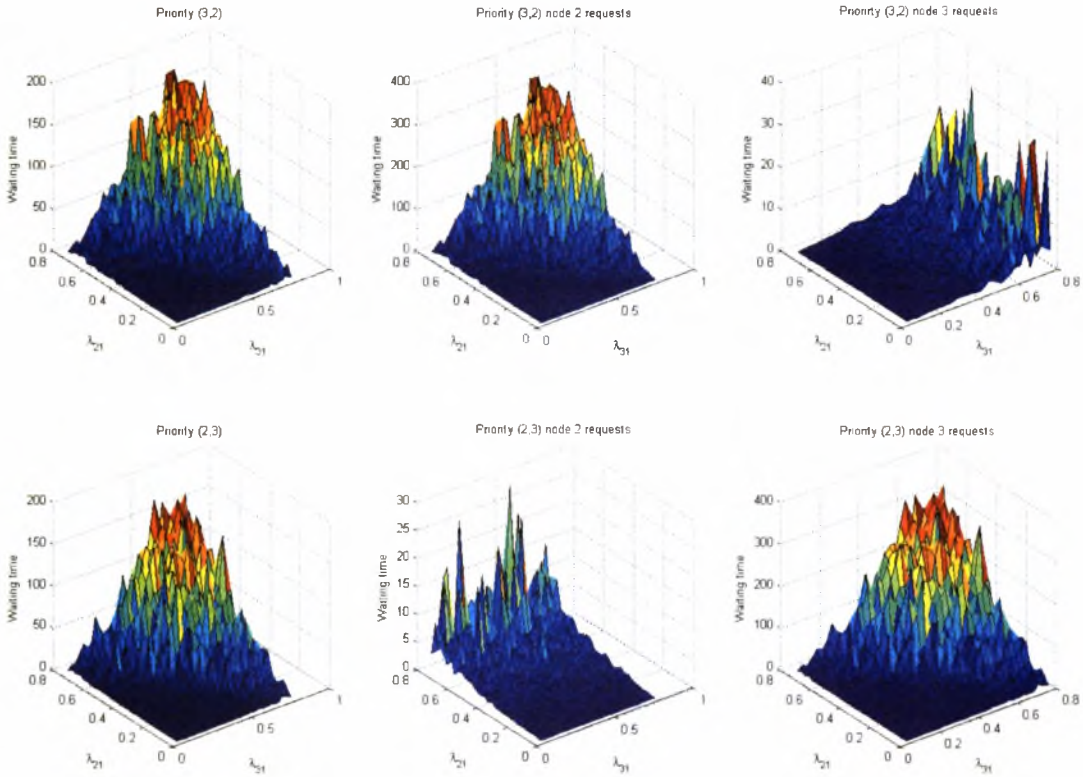
Εικόνα 16 Αναπαράσταση της σχέσης ανάμεσα σε ρυθμό παραγωγής-, ρυθμό εξυπηρέτησης και χρόνο αναμονής στην ουρά ενός κόμβου

Στη γραφική αυτή παρατηρούμε πως μεταβάλλεται ο μέσος χρόνος αναμονής στην ουρά των τριών εξυπηρετητών, καθώς μεταβάλλουμε το ρυθμό εξυπηρέτησης και το ρυθμό παραγωγής αιτήσεων. Εύκολα φαίνεται ότι για μικρό ρυθμό εξυπηρέτησης και μεγάλο ρυθμό παραγωγής, η καθυστέρηση στην ουρά είναι εξαιρετικά μεγάλη (στο βάθος του διαγράμματος), ενώ αν οι τιμές αυτές είναι αντίθετες, η καθυστέρηση είναι σχεδόν μηδενική. Γενικά παρατηρείται μεγάλη αύξηση στο χρόνο αναμονής, για τις τιμές εκείνες του ρυθμού παραγωγής που ξεπερνάνε τις αντίστοιχες του ρυθμού εξυπηρέτησης. Το γεγονός αυτό συμφωνεί με τη θεωρία ουρών, όπου αναφέρεται ότι το μέγεθος της ουράς ενός εξυπηρετητή(και επομένως η καθυστέρηση στην ουρά) αυξάνεται ανεξέλεγκτα όταν ο ρυθμός αφίξεων είναι πολύ μεγαλύτερος του ρυθμού εξυπηρέτησης. Για το λόγο αυτό δε χρησιμοποιήσαμε τιμές του χρόνου εξυπηρέτησης μικρότερες των 0.2 αιτ/λεπτό, διότι για ζευγάρια τιμών π.χ. (0.5 αιτ/λεπτό ρυθμός παραγωγής, 0.1 αιτ/λεπτό ρυθμός εξυπηρέτησης), οι τιμές των χρόνων αναμονής στην ουρά θα ήταν εξωπραγματικά μεγάλες, με αποτέλεσμα να μην ήταν εμφανής η συμπεριφορά του συστήματος στις τιμές που μας ενδιαφέρει περισσότερο λόγω της

κλίμακας που θα έπρεπε να χρησιμοποιήσει το Matlab για να εμφανίσει όλα τα σημεία της γραφικής παράστασης. Τέλος, αξίζει να αναφέρουμε ότι στο συγκεκριμένο παράδειγμα χρησιμοποιήσαμε την Shortest Job First πολιτική, αλλά την ίδια συμπεριφορά επιδεικνύουν και οι υπόλοιπες, με μόνη διαφορά φυσικά τις τιμές που θα πάρει ο χρόνος αναμονής στην ουρά για τις συγκεκριμένες τιμές του ρυθμού εξυπηρέτησης και παραγωγής.

Σχέση χρόνου αναμονής με επιμέρους ρυθμούς αφίξεων

Η επόμενη προσομοίωση που θα εκτελέσουμε, βασίζεται πάνω σε ένα μόνο κόμβο και σκοπό έχει να δείξει με ποιο τρόπο συνδέεται ο χρόνος αναμονής στην ουρά με τους δύο εισερχόμενους ρυθμούς άφιξης αιτήσεων, από τους δύο γειτονικούς κόμβους. Για το λόγο αυτό, θα εκτελέσουμε την προσομοίωση σε ένα μόνο κόμβο και θα σχεδιάσουμε τις 3 αυτές μεταβλητές σε τρισδιάστατους άξονες. Χρησιμοποιούμε, λοιπόν, τον κόμβο 1 στον οποίο εισέρχεται κίνηση από τον κόμβο 2 με ρυθμό λ_{21} και από τον 3 με ρυθμό λ_{31}. Θεωρούμε ότι η ουρά του κόμβου 1 ακολουθεί την Fixed Priority πολιτική, οπότε και θα εκτελέσουμε την προσομοίωση και για τις δύο περιπτώσεις προτεραιότητας (ο κόμβος 2 μεγαλύτερη προτεραιότητα από τον 3 και αντίθετα). Συγκεκριμένα, χρησιμοποιούμε διάφορες τιμές των ρυθμών εισόδου λ_{21} και λ_{31} που κυμαίνονται από 0.05 μέχρι 0.8 αιτ/λεπτό με βήμα 0.025. Ο ρυθμός εξυπηρέτησης αιτήσεων είναι 0.8 και εξυπηρετούνται συνολικά 500 αιτήσεις. Για κάθε ένα από τα ζευγάρια αυτά τιμών των δύο ρυθμών αφίξεων, η προσομοίωση μας δίνει μια τιμή για το χρόνο αναμονής, οπότε μπορούμε πλέον να παραστήσουμε γραφικά αυτές τις τριάδες τιμών σε τρισδιάστατο χώρο, όπως φαίνεται στη συνέχεια :



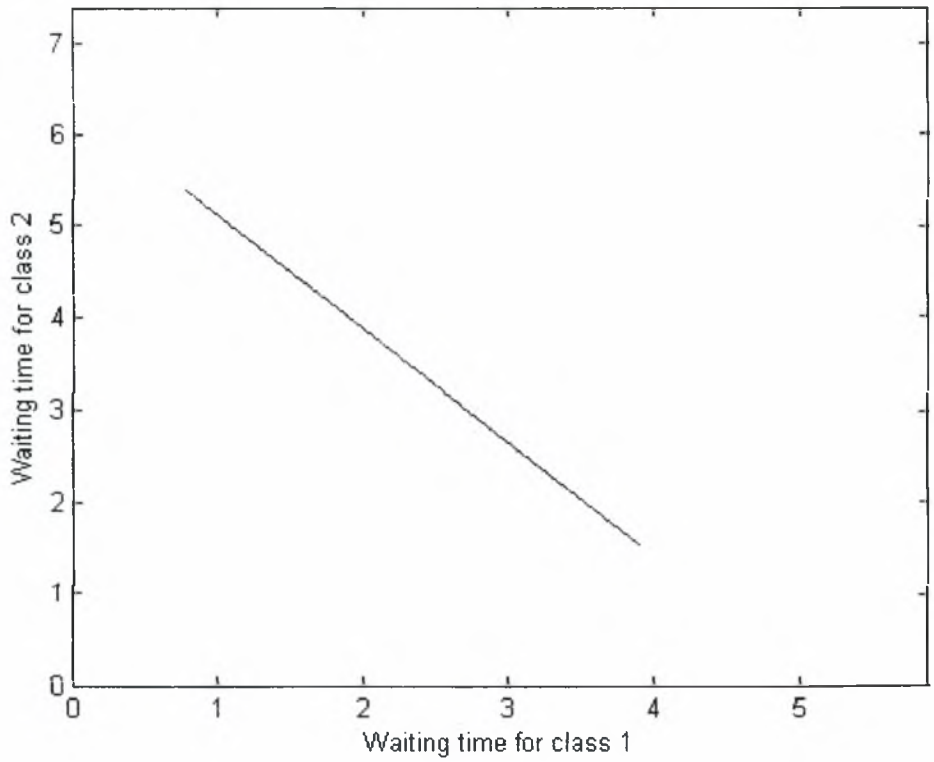
Εικόνα 17 Σχέση των επιμέρους ρυθμών αφίξεων κάθε κλάσης με το χρόνο αναμονής

Στην πρώτη γραφική παράσταση φαίνονται οι χρόνοι αναμονής όλων των αιτήσεων για όλες τις δυνατές τιμές των λ_{21} και λ_{31}, όταν οι αιτήσεις του κόμβου 3 έχουν προτεραιότητα αυτών του 2. Από την παράσταση αυτή είναι εμφανές ότι όσο πιο μικρές τιμές ρυθμού αφίξεων χρησιμοποιήσουμε, τόσο πιο μικρό χρόνο αναμονής δίνει το σύστημα. Όσο το άθροισμα των δύο ρυθμών παραμένει κάτω από ένα συγκεκριμένο όριο (στο παραπάνω παράδειγμα γύρω στις 0.8 αφίξεις/λεπτό) οι χρόνοι παραμένουν σε χαμηλά επίπεδα. Φαίνεται, δηλαδή, για μια ακόμα φορά ότι αν ο συνολικός ρυθμός αφίξεων στην είσοδο μιας ουράς ξεπεράσει το ρυθμό εξυπηρέτησης, η ουρά μεγαλώνει ανεξέλεγκτα και οι χρόνοι αναμονής εκτοξεύονται στα ύψη. Στη δεύτερη γραφική παράσταση φαίνονται οι αιτήσεις του κόμβου 2 αποκλειστικά, για όλες τις δυνατές τιμές των λ_{21} και λ_{31}, με προτεραιότητα του κόμβου 3 (σε ολόκληρη την πρώτη γραμμή των γραφικών ο κόμβος 3 έχει προτεραιότητα έναντι του 2). Παρατηρούμε και σ᾽ αυτή την περίπτωση ότι η αύξηση των δύο ρυθμών επιβαρύνει τους χρόνους αναμονής. Στην τρίτη, όμως, γραφική παράσταση που αναπαριστά τους χρόνους αναμονής των αιτήσεων του κόμβου 3 που έχει τη μεγαλύτερη προτεραιότητα, τα αποτελέσματα είναι ελαφρώς διαφορετικά.

Βλέπουμε ότι η αύξηση του ρυθμού λ_{21} δεν επηρεάζει καθόλου τους χρόνους αναμονής των αιτήσεων του 3. Το γεγονός αυτό αποδεικνύει ότι η πολιτική Fixed Priority δίνει πλήρη προτεραιότητα στον κόμβο 3 έναντι του 2 και οι αιτήσεις του 3 εξυπηρετούνται άμεσα παρά την πιθανή ύπαρξη μεγάλου ρυθμού αφίξεων από τον κόμβο 2. Φαίνεται, με άλλα λόγια, η ικανότητα της πολιτικής αυτής να ικανοποιεί τις αιτήσεις του κόμβου που έχει προτεραιότητα σαν να ήταν αυτός ο μόνος κόμβος που έστελνε αιτήσεις. Στις γραφικές παραστάσεις της κάτω γραμμής φαίνεται αντίστοιχα η κατάσταση σε περίπτωση όπου ο κόμβος 2 έχει προτεραιότητα έναντι του 3, οπότε και παρατηρούμε ανάλογα φαινόμενα.

Σχέση μεταξύ των δύο μέσων χρόνων αναμονής κάθε κλάσης προτεραιοτήτων στον ίδιο κόμβο

Στη συνέχεια, θα εκτελέσουμε μια προσομοίωση, η οποία μας δείχνει με ποιο τρόπο συνδέονται οι δύο μέσοι χρόνοι αναμονής στην ουρά για κάθε μια από τις κλάσεις προτεραιοτήτων σε ένα συγκεκριμένο κόμβο. Θεωρούμε, λοιπόν, τον κόμβο 1 από τους τρεις συνολικά του συστήματος και εισάγουμε σ' αυτόν αιτήσεις από τους άλλους με τυχαίο ρυθμό. Ο ρυθμός εξυπηρέτησης σε όλους τους κόμβους είναι 1 αιτ/λεπτό, ο ρυθμός παραγωγής αιτήσεων 0.75 αιτ/λεπτό και η πολιτική της ουράς είναι η Fixed Priority. Προσομοιώνουμε την άφιξη 1000 αιτήσεων συνολικά και υπολογίζουμε τους μέσους χρόνους αναμονής για καθεμιά από τις δύο προτεραιότητες (προτεραιότητα στον κόμβο 2 έναντι του 3 και αντίστροφα). Τους χρόνους που προκύπτουν μπορούμε να τους παραστήσουμε γραφικά, όπως φαίνεται στο παρακάτω διάγραμμα :



Εικόνα 18 Σχέση μεταξύ των δύο χρόνων αναμονής κάθε κλάσης

Από αυτή τη γραφική παράσταση μπορούμε να παρατηρήσουμε ότι οι χρόνοι αναμονής των αιτήσεων διαφορετικών κλάσεων σε μια ουρά πολιτικής Fixed Priority μεταβάλλονται αντίστροφα : όσο μειώνεται ο ένας χρόνος τόσο περισσότερο αυξάνεται ο άλλος. Δίνοντας, για παράδειγμα, προτεραιότητα στον κόμβο 2 ($1^{η}$ κλάση προτεραιότητας), οι χρόνοι αναμονής των αιτήσεων του κόμβου 3 ($2^{η}$ κλάση προτεραιότητας) αυξάνονται, όπως δείχνει το πάνω αριστερά σημείο της γραφικής. Αντίστροφα, όταν δώσουμε προτεραιότητα στις αιτήσεις του κόμβου 3, αυξάνεται κατά πολύ ο μέσος χρόνος αναμονής των αιτήσεων του 2 και συγκεκριμένα από 0.778 λεπτά που ήταν, αυξάνεται σε 3.896 λεπτά (κάτω δεξιά σημείο). Τα σημεία της γραφικής παράστασης είναι αυτά που φαίνονται παρακάτω(με σκούρο χρώμα), όπως προκύπτουν από την έξοδο του προγράμματος (με p=0.1 συμβολίζουμε την προτεραιότητα στην κλάση 2):

*time1= **0.778** and time2= **5.389***

*time1= **3.896** and time2= **1.545***

minimum sum=5.4418 for p=0.1

lamda21=0.27234, lamda31=0.51

Simulation time= 5.768 seconds

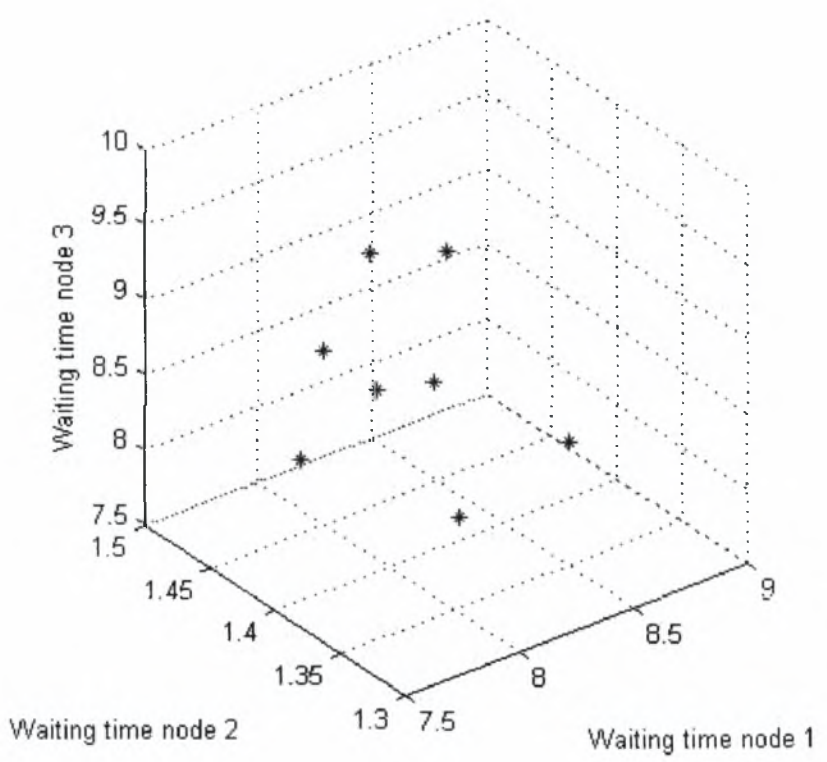
Για να αποφασίσουμε ποια από τις δύο προτεραιότητες είναι συμφέρουσα (προτεραιότητα στον κόμβο 2 ή στον 3), αρκεί να υπολογίσουμε το άθροισμα των χρόνων αναμονής για κάθε μία από αυτές. Έτσι, αν δώσουμε προτεραιότητα στον κόμβο 2 (κλάση 1) το άθροισμα είναι *0.778 + 5.389 = 6.1670* λεπτά, ενώ αν δώσουμε προτεραιότητα στον 3 (κλάση 2) το άθροισμα είναι *3.896 + 1.545 = 5.4418*. Ο δεύτερος τρόπος, δηλαδή, επιτυγχάνει αθροιστικά μικρότερο χρόνο αναμονής στην ουρά για τις δύο κλάσεις. Στο ίδιο συμπέρασμα καταλήγει και η προσομοίωση, όπως φαίνεται από την έξοδο του προγράμματος που παραθέσαμε παραπάνω. Με τον τρόπο αυτό, επομένως, το πρόγραμμα βρίσκει όχι μόνο τη σχέση ανάμεσα στους δύο χρόνους αναμονής, αλλά και τη βέλτιστη σειρά προτεραιότητας ώστε να ελαχιστοποιείται ο μέσος χρόνος και για τις δύο κλάσεις.

Σχέση μεταξύ των τριών μέσων χρόνων αναμονής κάθε κόμβου της προσομοίωσης

Με τρόπο αντίστοιχο με αυτό της προηγούμενης προσομοίωσης, μπορούμε να δείξουμε τη σχέση ανάμεσα στους τρεις χρόνους αναμονής στους κόμβους του συστήματος που μελετάμε καθώς και τον τρόπο που αυτοί οι χρόνοι μεταβάλλονται με την αλλαγή στις προτεραιότητες. Συγκεκριμένα θα δείξουμε ότι όσο μεγαλύτερος είναι κάποιος από τους τρεις χρόνους, τόσο περισσότερο μικραίνουν οι υπόλοιποι.

Ξεκινάμε ορίζοντας τις μεταβλητές του συστήματος : ο συνολικός ρυθμός παραγωγής αιτήσεων είναι 0.35 αιτ/λεπτό και ο ρυθμός εξυπηρέτησης 0.5 αιτ/λεπτό, τιμές σταθερές και για τους τρεις κόμβους καθ' όλη τη διάρκεια της προσομοίωσης. Προσομοιώνουμε την άφιξη 1000 αιτήσεων στην ουρά κάθε κόμβου, ενώ θεωρούμε τυχαία αλλά σταθερά σπασίματα των ρυθμών παραγωγής. Παρατηρούμε ότι υπάρχουν 8 (2^3) διαφορετικοί τρόποι να δώσουμε προτεραιότητα, δεδομένου ότι υπάρχουν 3 κόμβοι και 2 διαφορετικοί τρόποι για κεθέναν να δώσει προτεραιότητα στους υπόλοιπους (ο κόμβος 1 να δώσει προτεραιότητα στον 2 έναντι του 3 ή στον 3 έναντι του 2 κ.ο.κ.). Για κάθε έναν από αυτούς τους διαφορετικούς τρόπους, εκτελούμε την προσομοίωση με τις μεταβλητές που θέσαμε και παίρνουμε τιμές για τους μέσους χρόνους αναμονής στην ουρά κάθε κόμβου. Έπειτα σχηματίζουμε τις τριάδες (waiting time for node1, waiting time for node2, waiting time for node3) για καθεμιά από τις 8 περιπτώσεις των διαφορετικών προτεραιοτήτων και τοποθετούμε

τις τριάδες αυτές στον τρισδιάστατο χώρο, όπου κάθε άξονας αντιπροσωπεύει το μέσο χρόνο αναμονής σε καθέναν από τους τρεις κόμβους. Η γραφική αυτή παράσταση φαίνεται παρακάτω, με τους μπλε αστερίσκους να αντιπροσωπεύουν τις προαναφερόμενες τριάδες τιμών :



Εικόνα 19 Σχέση ανάμεσα στους χρόνους αναμονής των τριών κόμβων

Φαίνεται εκ πρώτης όψεως σωστή η πρώτη παρατήρηση που είχαμε κάνει, ότι δηλαδή, όσο περισσότερο αυξάνεται η τιμή της αναμονής σε ένα κόμβο, τόσο περισσότερο μειωμένη είναι στους άλλους (αν και λόγω της μικρής κλίμακας ιδιαίτερα στον άξονα του κόμβου 2 και της τυχαιότητας δεν είναι ξεκάθαρα εμφανής). Για παράδειγμα, το δεξιότερο σημείο της γραφικής παράστασης, έχει χρόνο αναμονής στον κόμβο 1 περίπου ίσο με 9 λεπτά, που είναι ο μεγαλύτερος απ' όλους για τον κόμβο αυτό, ενώ στον 3 έχει τη μικρότερη τιμή απ' όλες (περίπου 7.5 λεπτά) και στον 2 έχει ενδιάμεση τιμή (περίπου 1.45 λεπτά). Σε παρόμοιο συμπέρασμα καταλήγουμε, με μικρές αποκλίσεις, για όλα τα σημεία της γραφικής παράστασης, δηλαδή για όλες τις δυνατές προτεραιότητες. Η αιτία που συμβαίνει το φαινόμενο αυτό είναι εξαιρετικά απλή : όταν σε κάποιο κόμβο ο ρυθμός άφιξης είναι μεγάλος, τότε είναι μεγάλοι και οι χρόνοι αναμονής στην ουρά του. Επειδή, όμως, οι ρυθμοί άφιξης σε κάθε κόμβο συνδέονται μεταξύ τους με τη σχέση :

Άθροισμα ρυθμών άφιξης = άθροισμα ρυθμών παραγωγής

τότε όταν ένας εμφανίζει υψηλό ρυθμό αφίξεων, κάποιος από τους γειτονικούς θα εμφανίζει αντίστοιχα σχετικά μικρό ρυθμό αφίξεων και επομένως μικρό μέσο χρόνο αναμονής.

Από την έξοδο του προγράμματος, που φαίνεται παρακάτω, μπορούμε μάλιστα να υπολογίσουμε τις βέλτιστες προτεραιότητες που επιτυγχάνουν συνολικά τους βέλτιστους χρόνους αναμονής στην ουρά κάθε κόμβου:

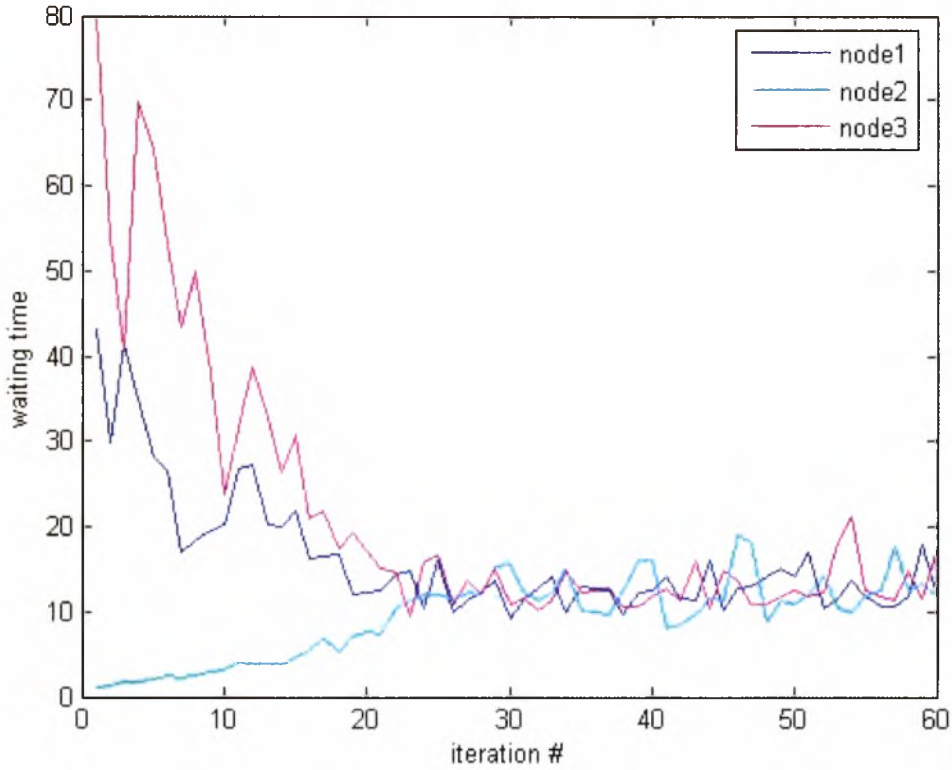
Waiting time1 = 7.843, Waiting time2 = 1.319, Waiting time3 = 8.392
Waiting time1 = 7.820, Waiting time2 = 1.419, Waiting time3 = 8.935
Waiting time1 = 8.095, Waiting time2 = 1.372, Waiting time3 = 9.703
Waiting time1 = 7.937, Waiting time2 = 1.456, Waiting time3 = 7.931
Waiting time1 = 8.278, Waiting time2 = 1.463, Waiting time3 = 9.060
Waiting time1 = 8.176, Waiting time2 = 1.396, Waiting time3 = 8.642
Waiting time1 = 8.010, Waiting time2 = 1.411, Waiting time3 = 8.604
Waiting time1 = 8.409, Waiting time2 = 1.334, Waiting time3 = 8.475
node1 rate=0.417, node2 rate=0.219, node3 rate=0.414
Minimum sum=17.324 for node1 giving priority to 2, node 2 to 3 and node 3 to 2
Simulation time=66.626 second

Αναπροσαρμογή φόρτου κατά την εκτέλεση της προσομοίωσης

Ένα ακόμα πρόβλημα το οποίο αξίζει να αναλυθεί, είναι εκείνο της αναπροσαρμογής του φόρτου ώστε να επιτευχθούν βέλτιστοι χρόνοι αναμονής. Τίθεται, αρχικά, το ερώτημα, αν είναι δυνατόν το σύστημα να αναγνωρίσει ότι κάποιος κόμβος είναι υπερβολικά φορτωμένος με αιτήσεις και να μοιράσει τον επιπλέον αυτό φόρτο στους παρακείμενους κόμβους. Για το λόγο αυτό εκτελούμε την επόμενη προσομοίωση, κατά την οποία υλοποιείται αυτός ακριβώς ο αλγόριθμος.

Θεωρούμε το ίδιο σύστημα, όπως στα προηγούμενα παραδείγματα, με τα ίδια χαρακτηριστικά. Ο ρυθμός παραγωγής αιτήσεων είναι ίδιος και για τους τρεις κόμβους και έχει τιμή 0.45 αιτ/λεπτό, ενώ ο ρυθμός εξυπηρέτησης είναι 0.5 αιτ/λεπτό. Προσομοιώνουμε την άφιξη 1000 αιτήσεων σε κάθε κόμβο και θεωρούμε αρχικά τυχαία σπασίματα των ρυθμών παραγωγής. Κατά την εκκίνηση της

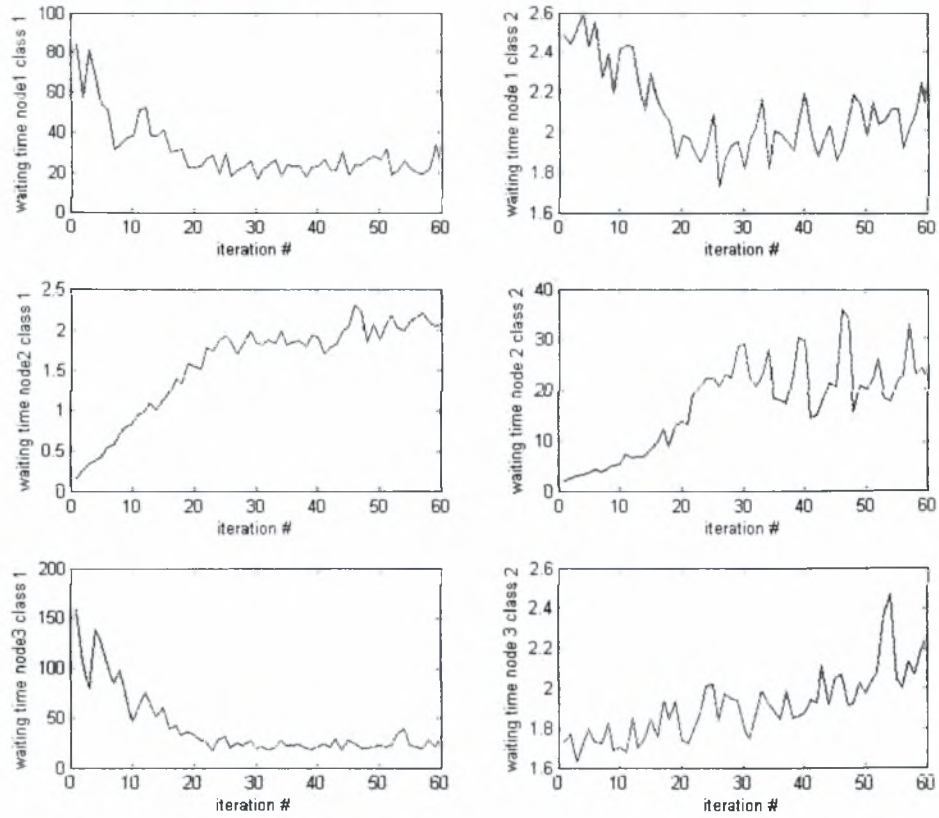
προσομοίωσης και για την πρώτη επανάληψη, προκύπτουν κάποιες τιμές για τους μέσους χρόνους αναμονής, οι οποίοι αντιστοιχούν στις τιμές των ρυθμών άφιξης σε κάθε κόμβο (μεγάλος ρυθμός άφιξης έχει ως αποτέλεσμα μεγάλο μέσο χρόνο αναμονής των αιτήσεων στην ουρά). Πριν ξεκινήσει η δεύτερη επανάληψη, το σύστημα ενημερώνει κάθε κόμβο για τους χρόνους που επιτεύχθηκαν στους γειτονικούς τους. Σε περίπτωση που ο μέσος χρόνος αναμονής σε κάποιο κόμβο ξεπερνάει κατά 30% το μέσο χρόνο κάποιου από τους γειτονικούς του, τότε γίνεται αναπροσαρμογή του φόρτου εργασίας, μειώνοντας κατά 1% το ρυθμό άφιξης αιτήσεων καθενός από τους επιμέρους ρυθμούς που εισέρχονται στον φορτωμένο κόμβο. Αν, δηλαδή, ο κόμβος 1 έχει μέση καθυστέρηση στην ουρά που ξεπερνάει κατά 30% τη μέση καθυστέρηση των άλλων, τότε οι ρυθμοί λ_{21} και λ_{31} που εισέρχονται σ' αυτόν μειώνονται κατά 1% (και αντίστοιχα αυξάνονται οι λ_{23} και λ_{32} λόγω διατήρησης της ροής) για να αμβλυνθεί η παρατηρούμενη διαφορά. Παράλληλα, γίνεται αναπροσαρμογή και των προτεραιοτήτων με βάση τις νέες τιμές. Δεδομένου ότι επιθυμούμε να πετύχουμε ισόποσο μοίρασμα του φόρτου σε κάθε κόμβο, δίνουμε μεγαλύτερη προτεραιότητα στον κόμβο εκείνο με το **μικρότερο** ρυθμό αφίξεων. Το σκεπτικό είναι ότι θέλουμε να αποθαρρύνουμε τους κόμβους που ζητάνε ήδη πολλά αρχεία από το να ζητήσουν κι άλλα, για να μην μονοπωλήσουν τη δυνατότητα του εξυπηρετητή, ενώ παράλληλα να ενθαρρύνουμε τους κόμβους που δεν ζητάνε πολλά, ώστε τελικά ένας κόμβος-server να εξυπηρετεί μεγάλο αριθμό κόμβων-clients. Κατά την δεύτερη επανάληψη, υπολογίζονται οι νέες τιμές καθυστέρησης στις ουρές των κόμβων και ανάλογα με τα αποτελέσματα επαναλαμβάνεται η διαδικασία αναπροσαρμογής που περιγράφηκε παραπάνω. Με τον τρόπο αυτό επιτυγχάνουμε να μειώσουμε τους μέσους χρόνους αναμονής στους φορτωμένους κόμβους και παράλληλα να αυξήσουμε τους αντίστοιχους χρόνους στους κόμβους που δεν έχουν μεγάλο φόρτο. Το σύστημα οδηγείται τελικά σε μια εξισορροπημένη κατάσταση, στην οποία όπως φάνηκε και παραπάνω οι ρυθμοί αφίξεων σε κάθε κόμβο είναι ίσοι. Τα αποτελέσματα μιας ενδεικτικής προσομοίωσης φαίνονται στις επόμενες γραφικές παραστάσεις :



Εικόνα 20 Σύγκλιση αλγορίθμου αναπροσαρμογής φόρτου

Στην πρώτη αυτή γραφική παράσταση καταγράφεται η μεταβολή των μέσων χρόνων αναμονής κάθε κόμβου, σε κάθε επανάληψη. Παρατηρούμε ότι αρχικά, ο μέσος χρόνος αναμονής στην ουρά του κόμβου 3 είναι πολύ μεγαλύτερος από αυτόν του κόμβου 1 και ακόμα περισσότερο από αυτόν του κόμβου 2. Μετά τη δεύτερη επανάληψη, όμως, οι χρόνοι αρχίζουν να ελαττώνονται για τον 3 και για τον 1 δεδομένου ότι και οι δύο είναι κατά 30% μεγαλύτεροι από τον μέσο χρόνο του 2. Αντιθέτως ο μέσος χρόνος αναμονής του 2 αυξάνεται μέχρι τελικά να συναντηθούν οι προαναφερόμενες μεταβλητές σε μία κοινή τιμή. Παρατηρούμε ότι ο αλγόριθμος συγκλίνει μέσα σε περίπου 25 επαναλήψεις, ενώ από εκεί και πέρα οι τιμές παραμένουν ως έχουν (οι μεταβολές γύρω από την τιμή αυτή οφείλονται στην τυχαιότητα του αλγορίθμου Fixed Priority).

Στην επόμενη γραφική παράσταση φαίνεται η αυξομείωση των τιμών με τον τρόπο που περιγράφηκε για κάθε κόμβο και κάθε κλάση προτεραιότητας ξεχωριστά :



Εικόνα 21 Σύγκλιση των επιμέρους χρόνων αναμονής κάθε κλάσης

Είναι ακόμα πιο εμφανής η μείωση της καθυστέρησης στον κόμβο 3 (τελευταίες δύο γραφικές) και ιδιαίτερα η καθυστέρηση των αιτήσεων λόγω του κόμβου 1 (αριστερή γραφική παράσταση). Παρατηρούμε, επίσης, ότι οι αιτήσεις του κόμβου 2 στον 3 (τελευταία γραφική) δεν επηρεάστηκαν ιδιαίτερα από τις αυξομειώσεις των ρυθμών άφιξης και αυτό λόγω του γεγονότος ότι είχαν προτεραιότητα στην εξυπηρέτηση έναντι των αιτήσεων του κόμβου 1. Το ίδιο, βέβαια, σκηνικό επικρατεί και στους άλλους κόμβους : οι αιτήσεις στον κόμβο 2 λόγω του 1 και 3 καθυστερούν περισσότερο απ' ότι πριν και ιδιαίτερα αυτές του κόμβου 3 (μεσαία-δεξιά γραφική), λόγω της χαμηλότερης προτεραιότητας. Στον κόμβο 1, τέλος, οι αιτήσεις του κόμβου 3 (δεξιά) έχουν μεγαλύτερη προτεραιότητα από αυτές του κόμβου 2 και γι' αυτό η καθυστέρηση παραμένει σχεδόν σταθερή (κυμαίνεται μεταξύ 2 και 2.6 λεπτά), ενώ αντίθετα οι αιτήσεις του 3 (αριστερά) έχουν εμφανή μείωση στις τιμές καθυστέρησης (από 80 λεπτά αρχικά σε 20 λεπτά περίπου). Με άλλα λόγια παρατηρούμε πως οι αιτήσεις των κόμβων που έχουν προτεραιότητα, δεν επηρεάζονται ιδιαίτερα από τον αλγόριθμο που υλοποιήσαμε, αλλά η μείωση στην καθυστέρηση των αιτήσεων που

δεν έχουν προτεραιότητα είναι εξαιρετικά μεγάλη και μπορούμε να μιλάμε για σημαντική βελτίωση των επιδόσεων του συστήματος.

Η έξοδος του προγράμματος της προσομοίωσης μας εμφανίζει (εκτός φυσικά από τις γραφικές παραστάσεις) τα παρακάτω αποτελέσματα, όπου είναι πλέον προφανές ότι ο φόρτος εργασίας μοιράστηκε εξίσου σε κάθε κόμβο. Φαίνεται με άλλα λόγια ότι για να λειτουργήσει το σύστημα με βέλτιστο τρόπο, αρκεί να χρεώσουμε σε κάθε κόμβο ίσο φόρτο εργασίας. Μάλιστα, επειδή θεωρούμε τους ρυθμούς παραγωγής αιτήσεων ίσους, παρατηρούμε ότι ο βέλτιστος ρυθμός αφίξεων ισούται με το ρυθμό παραγωγής κάθε κόμβου.

Initial request rates : node1=0.500, node2=0.270, node3=0.580
Final request rates : node1=0.444, node2=0.457, node3=0.448
Simulation time=114.555 seconds

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Ο κυριότερος παράγοντας που επηρεάζει την ποιότητα υπηρεσιών σε ένα P2P δίκτυο είναι αναμφισβήτητα ο χρόνος αναμονής μέχρι να εξυπηρετηθεί κάποιος χρήστης. Στην ανάλυση που προηγήθηκε, μελετήσαμε τρόπους με τους οποίους μπορούμε να επιτύχουμε την ελαχιστοποίηση του χρόνου αυτού. Για να πετύχουμε το σκοπό αυτό χρησιμοποιήσαμε τεχνικές που βρίσκουν εφαρμογή στη θεωρία ουρών, με πιο αποτελεσματική την τεχνική που επιλέγει την μικρότερη εργασία από την ουρά (Shortest Job First). Ένα ακόμα σημαντικό πρόβλημα που επιλύσαμε, είναι αυτό της βέλτιστης ανάθεσης εργασιών ανάμεσα στους κόμβους, με το οποίο καταφέραμε την περαιτέρω αύξηση της απόδοσης του συστήματος. Τέλος, αναθέτοντας προτεραιότητες ανάμεσα στους κόμβους, επιλύσαμε το πρόβλημα της βέλτιστης ανάθεσης προτεραιοτήτων, ώστε και πάλι να επιτυγχάνονται οι καλύτερες δυνατές τιμές αναμονής. Αν και ασχοληθήκαμε κατά κύριο λόγο με την ελαχιστοποίηση του χρόνου αναμονής, υπάρχουν και άλλοι παράγοντες που επηρεάζουν την ποιότητα υπηρεσιών ενός P2P δικτύου, όπως ο τρόπος διεξαγωγής ερωτημάτων, η αποτελεσματικότητά τους και η επιβάρυνση του δικτύου με άχρηστη πληροφορία. Η μελέτη αυτών των παραμέτρων και ο τρόπος που επηρεάζουν την απόδοση του συστήματος μπορούν να γίνουν αντικείμενο μελλοντικής έρευνας και επέκτασης της παρούσας ανάλυσης.

ΠΑΡΑΡΤΗΜΑ

Στο σημείο αυτό αξίζει να αναφέρουμε τον τρόπο με τον οποίο πραγματοποιήσαμε το μετασχηματισμό των τυχαίων μεταβλητών. Δεδομένου ότι το Matlab διαθέτει συνάρτηση που παράγει τυχαίους αριθμούς που ακολουθούν ομοιόμορφη κατανομή, ήταν αναγκαίος ο μετασχηματισμός τους σε αριθμούς που ακολουθούν άλλες κατανομές. Για το λόγο αυτό ακολουθήθηκε η μέθοδος του μετασχηματισμού που θα εξηγήσουμε αμέσως :

Έστω X είναι η τυχαία μεταβλητή που μας ενδιαφέρει και F(x) είναι η συνάρτηση κατανομής της, ώστε:

$$F(x) = P[X \leq x]$$

Θέτουμε $F(x) = Y$. Θεωρούμε ότι η Y ορίζεται στο διάστημα 0 έως 1. Αν η Y είναι τυχαία μεταβλητή με ομοιόμορφη κατανομή στο 0 έως 1, τότε η X που ορίζεται ώς $X = F^{-1}(Y)$ έχει συνάρτηση κατανομής F(x):

$$P[X \leq x] = P[Y \leq F(x)] = F(x)$$

Παράδειγμα: Εκθετική Κατανομή

$$F(x) = 1 - e^{-\mu x}, \quad x \geq 0$$

$$Y = 1 - e^{-\mu x}$$

$$X = F^{-1}(Y) = -\frac{(\ln(1-Y))}{\mu}$$

Εφόσον η (1-Y) είναι τυχαίος αριθμός μεταξύ 0 και 1, άρα μπορούμε να τον αντικαταστήσουμε με την Y.

$$X = F^{-1}(1-Y) = -\frac{\ln Y}{\mu}$$

Με τον τύπο αυτό, δίνοντας στην τυχαία μεταβλητή Y τιμές που ακολουθούν ομοιόμορφη κατανομή, μπορούμε να πάρουμε μέσω της τυχαίας μεταβλητής X τιμές που ακολουθούν εκθετική. Με παρόμοιο τρόπο μπορούμε να πάρουμε τιμές οποιασδήποτε γνωστής κατανομής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Jay Sethuraman, Mark S. Squillante. Optimal Stochastic Scheduling in Multiclass Parallel Queues

[2] Keith W. Ross, Dan Rubenstein. P2P Tutorial

[3] Magnus Kolweyh. Towards Next-Generation Peer-to-Peer Systems

[4] Neil Daswani, Hector Garcia-Molina, and Beverly Yang. Open Problems in Data-Sharing Peer-to-Peer Systems

[5] Sem Borst, G.M. Koole, L.C.M. Kallenberg. Stochastic Dynamic Programming & Control of Queues

[6] Don Towsley, Peer-to-peer and application-level networking

[7] http://www.cachelogic.com

[8] http://www.atm.tut.fi/delco/literature.html

[9] http://www.bsdg.org/Jim/Peer2Peer/Paper/3214.html

[10] http://wiki.cs.uiuc.edu/cs427/p2p

[11] http://nislab.bu.edu/

[12] Oliver Heckmann, Axel Bock. The eDonkey 2000 Protocol

[13] B. Cohen. Incentives build robustness in bittorrent, May 2003. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf

[14] Jeremy Reyes. Cutting Edge Peer-to-Peer File Sharing: BitTorrent

[15] Lee, J. Peer-to-Peer File Sharing Systems: What Matters to the End. MIT Center for Coordination Science

[16] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, M. Faloutsos, File-sharing in the Internet: A characterization of P2P traffic in the backbone

[17] The Gnutella protocol specification www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[18] Alberto Leon-Garcia. Harnessing the power of P2P

[19] Jian Liang, Rakesh Kumar, Keith W. Ross. Understanding KaZaA

[20] http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/Intro.html

[21] http://compnetworking.about.com/cs/peertopeer/

[22] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips. The BitTorrent P2P File-Sharing System: Measurements and Analysis

[23] Xiangying Yand, Gustavo de Veciana. Performance of P2P Networks : Service capacity and role of resource sharing policies

[24] Zihui Ge, Daniel Figueiredo. Modeling P2P File Sharing Systems

[25] Dongyu Qiu, R. Srikant. Modeling and Performance Analysis of BitTorrent-like P2P Networks