Χαροκόπειο Πανεπιστήμιο

Τμήμα Πληροφορικής και Τηλεματικής

Πτυχιακή εργασία

Τίτλος: Ανάλυση επίδοσης συστημάτων σε ηλεκτρονικά παιχνίδια σε κινητές συσκευές μέσω διαδικτύου

Βασίλης Νικολόπουλος

Α.Μ.: 20920

Τριμελής Επιτροπή:

κ. Αναγνωστόπουλος Δημοσθένης

κ. Νικολαΐδου Μάρα

κ. Τσερπές Κωνσταντίνος

Αθήνα 2013

Στην μνήμη του μεγάλου Καθηγητή

Δασκάλου

Φίλου

Γεωργίου Καραμπατζού

Περιεχόμενα

Λίστα διαγραμμάτων και πινάκων

Περίληψη

Τα σύγχρονα ηλεκτρονικά παιχνίδια είναι στην πραγματικότητα, στην πλειοψηφία τους, κατανεμημένες διαδραστικές εφαρμογές οι οποίες επιτρέπουν στους συμμετέχοντες να αλληλεπιδρούν μεταξύ τους. Ο μεγάλος αριθμός χρηστών και η μεγάλη κατανομή των τοποθεσιών των χρηστών δημιουργεί την απαίτηση ύπαρξης πολλών εξυπηρετητών, με μεγάλες γεωγραφικές αποστάσεις, οι οποίοι πρέπει να συνεργάζονται μεταξύ τους για την ομαλή διεξαγωγή των παιχνιδιών. Πρέπει λοιπόν πέρα από την επικοινωνία των χρηστών με τους εξυπηρετητές να μελετηθεί και η αρχιτεκτονική της σύνδεσης των εξυπηρετητών μεταξύ τους. Η συνδεσμολογία του εσωτερικού δικτύου επικοινωνίας των εξυπηρετητών, ο αριθμός των εξυπηρετητών και ο τρόπος ανάθεσης χρηστών επηρεάζει τους χρόνους επικοινωνίας και συνεπώς το κόστος. Σε αυτή την εργασία μέσω ενός αριθμού προσομοιώσεων θα εξάγουμε συμπεράσματα για τους διαφορετικούς αλγορίθμους κατανομής χρηστών στους εξυπηρετητές, πώς μπορεί να μεγιστοποιηθεί το θεωρητικό κέρδος από την διεξαγωγή ενός παιχνιδιού καθώς και να συμπεράνουμε μέσω των ανωτέρω αναφερομένων αποτελεσμάτων για το εάν είναι δυνατό να υπάρξουν παιχνίδια με μεγάλες απαιτήσεις πραγματικού χρόνου για κινητά τηλέφωνα.

Abstract

Modern video games are in their majority distributed interactive applications, which allow their participants to interact with each other. The great number of users and the large distribution of users' placement demands the use of many servers, having large geographical distances among them, co operating for the smooth conduct of games. We have to study, not only the way users connect to the servers, but also, the way servers interconnect among themselves. The inter-server network architecture and the way they distribute clients among them, affects the communication delay and therefore the total cost. In this dissertation thesis, through a number of simulations, we will come to a conclusion for a number of client distribution algorithms, how one can achieve a theoretical maximum gain from conducting a match and using the above conclusions, deduct whether it is possible, games with real-time network demands to exist on cellular phones.

I. Εισαγωγή

Τα σύγχρονα ηλεκτρονικά παιχνίδια χωρίζονται σε πολλές κατηγορίες. Μάλιστα τα περισσότερα ηλεκτρονικά παιχνίδια σήμερα βασίζονται στην επικοινωνία με άλλους παίκτες μέσω του διαδικτύου. Η επικοινωνία αυτή, συνήθως, έχει την απαίτηση να είναι πραγματικού χρόνου, δηλαδή να υπάρχει πολύ μικρή χρονοκαθυστέρηση μεταξύ της επικοινωνίας των χρηστών. Τα παιχνίδια αυτά τα οποία έχουν ως απώτερο σκοπό την εξυπηρέτηση ενός μεγάλου αριθμού παιχτών ταυτόχρονα ονομάζονται MMOG, Massively multiplayer online games.

Στα MMOG συνήθως υπάρχει ένας εικονικός κόσμος στον οποίο συνυπάρχουν πολλοί παίχτες μαζί και αλληλεπιδρούν μεταξύ τους. Συνήθως οι πράξεις του ενός χρήστη επηρεάζουν άμεσα τους υπολοίπους. Αυτό δημιουργεί την απαίτηση για διατήρησης της συνοχής του κόσμου. Η διατήρηση της συνεκτικότητας, λοιπόν, μας οδηγεί σε μία αρχιτεκτονική συστήματος στρεφόμενη γύρω από τους εξυπηρετητές, οι χρήστες δηλαδή επικοινωνούν με έναν κεντρικό εξυπηρετητή ο οποίος είναι υπεύθυνος για την διατήρηση μίας κοινής εικόνας του κόσμου.

Όμως, οι αρχιτεκτονικές εφαρμογών που στρέφονται γύρω από τους εξυπηρετητές είναι δύσκολο να διατηρηθούν όσο μεγαλώνει ο αριθμός των χρηστών. Έτσι δημιουργείται η απαίτηση επιπλέον εξυπηρετητών οι οποίοι είναι υπεύθυνοι για την διατήρηση της συνεκτικότητας του ιδίου κόσμου, δηλαδή κατοπτρικοί εξυπηρετητές, ή Mirrored Servers. [1]

Όταν γίνεται χρήση κατοπτρικών εξυπηρετητών, οι εξυπηρετητές επικοινωνούν μεταξύ τους για τη διατήρηση της συνοχής του κόσμου, ενώ ο κάθε χρήστης επικοινωνεί με έναν εξυπηρετητή αποκλειστικά. Ο κάθε χρήστης επικοινωνεί με τον εξυπηρετητή που του έχει προσδιοριστεί, και ύστερα ο εξυπηρετητής είναι υπεύθυνος για την προώθηση της ενέργειας του χρήστη στους υπόλοιπους εξυπηρετητές.

Για να είναι ακριβής η εξομοίωση του παιχνιδιού, ωστόσο, πρέπει να υπάρχει μία σειρά στις ενέργειες των χρηστών. Η χρήση ενός κοινού καθολικού ρολογιού για αυτό τον σκοπό είναι πρακτικά αδύνατη. Τα πιο πρακτικά συστήματα παιχνιδιών υιοθετούν ένα μηχανισμό συγχρονισμού, ο οποίος διαιρεί τον χρόνο σε διακριτές θυρίδες και κάθε εξυπηρετητής περιμένει να έλθουν συμβάντα δημιουργούμενα από απομακρυσμένους χρήστες, πριν να πραγματοποιήσει την εξομοίωση του παιχνιδιού. Ανάλογα με το σχεδιασμό του παιχνιδιού, ασυνέπειες λόγω καθυστερημένων

γεγονότων μπορούν να λυθούν, είτε με το να επιστρέψουν οι εξυπηρετητές σε παλαιότερη κατάσταση παιχνιδιού, ή με το να αγνοηθούν εντελώς. Στη συγκεκριμένη μελέτη δεν θα χρησιμοποιήσουμε κάποιο συγκεκριμένο μηχανισμό συγχρονισμού. Θεωρούμε ότι η καθυστέρηση για τον συγχρονισμό πρέπει να είναι μεγαλύτερη από την καθυστέρηση που υπάρχει για τα συμβάντα από τους χρήστες να φτάσουν στον μακρινότερο εξυπηρετητή, και για τα αποτελέσματα της εξομοίωσης να φτάσουν σε όλους τους τοπικούς χρήστες. [1]

Σκοπός της εργασίας αυτής είναι μέσω ενός αριθμού προσομοιώσεων να ελέγξουμε την απόδοση συγκεκριμένων αλγορίθμων, οι οποίοι περιγράφουν τον τρόπο με τον οποίο συνδέονται οι χρήστες με τους εξυπηρετητές, υπό διαφορετικές συνθήκες. Επίσης θα μελετήσουμε, σε συνδυασμό με τους προηγούμενους αλγορίθμους, την απόδοση δύο αλγορίθμων που περιγράφουν των τρόπο με τον οποίο χειρίζονται οι εξυπηρετητές την ουρά σύνδεσης πελατών.

Θα προσπαθήσουμε μέσω των παραπάνω αποτελεσμάτων να εξάγουμε συμπέρασμα για το ποιοι συνδυασμοί αλγορίθμων είναι οι αποδοτικότεροι και υπό ποιες συνθήκες.

Για να συγκεντρώσουμε τα δεδομένα αυτά θα διεξάγουμε έναν αριθμό προσομοιώσεων. Οι προσομοιώσεις αυτές θα περιγράφουν ένα σύστημα από διασυνδεδεμένους μεταξύ τους εξυπηρετητές και έναν αριθμό χρηστών που προσπαθούν να συνδεθούν στους εξυπηρετητές κατά τη διάρκεια του χρόνου.

Πιστεύουμε ότι τα συμπεράσματα αυτής της εργασίας θα βοηθήσουν στην βελτιστοποίηση των μοντέλων σύνδεσης και εξυπηρέτησης χρηστών σε διαδικτυακές διαδραστικές εφαρμογές τύπου MMOG.

Επιπλέον θα μελετήσουμε εάν είναι δυνατό να υπάρξουν MMOG, με απαιτήσεις δικτύου επιπέδου πραγματικού χρόνου, σε συσκευές κινητής τηλεφωνίας.

- Στο κεφάλαιο II αυτής της εργασίας μελετούμε τις απαιτήσεις ενός διάσημου είδους MMOG, τα MOBA, καθώς και την κατανομή των ταυτόχρονων συνδεδεμένων χρηστών σε συνάρτηση με τον χρόνο.
- Στο κεφάλαιο III θα εξετάσουμε ένα μαθηματικό μοντέλο που μπορεί να περιγράψει τον ρυθμό άφιξης χρηστών.
- Στο κεφάλαιο IV θα δούμε αναλυτικά τους αλγόριθμους που περιγράφουν την εξυπηρέτηση των χρηστών.
- Στο κεφάλαιο V περιγράφουμε τον τρόπο με τον οποίο διεξάγονται οι αγώνες των παιχνιδιών MOBA και αναλύουμε τι απαιτήσεις δημιουργεί αυτό.

- Στο κεφάλαιο VI περιγράφουμε τον τρόπο με τον οποίο διεξήχθησαν οι προσομοιώσεις και παρουσιάζουμε τα αποτελέσματα αυτών.
- Στο κεφάλαιο VII αναλύουμε την τρέχουσα εικόνα της τεχνολογίας των κινητών τηλεφώνων και αναλύουμε εποπτικά τα χαρακτηριστικά των δικτύων κινητής τηλεφωνίας.
- Στο κεφάλαιο VIII έχουμε τα συμπεράσματα της εργασίας.

ΙΙ. Οι τρόποι κατανομής χρηστών

Οι απαιτήσεις δημιουργούν τους αλγορίθμους

Τα MMOG χωρίζονται σε διάφορες κατηγορίες. Κάθε κατηγορία έχει τις δικές της απαιτήσεις και προδιαγραφές, έτσι, κάθε κατηγορία έχει και διαφορετικούς αλγορίθμους για την κατανομή των χρηστών στους εξυπηρετητές.

Σε παιχνίδια First person Shooter, όπως για παράδειγμα το Team Fortress 2 ® και το Counter Strike: Global Offensive ®, έχουμε συνήθως μεγάλο αριθμό εξυπηρετητών, με τη πλειονότητα αυτών να είναι ιδιωτικοί και μη συνυφασμένοι με την εταιρία που έχει δημιουργήσει το παιχνίδι, και σχετικά μικρό αριθμό χρηστών ανά εξυπηρετητή. Οι χρήστες είναι ελεύθεροι να λάβουν μέρος σε όποιο παιχνίδι αυτοί επιθυμούν. Δεν υπάρχει κάποιος συγκεκριμένος αλγόριθμος.

Σε παιχνίδια τύπου MMORPG, Massively Multiplayer Online Role Playing Game, όπως για παράδειγμα το World Of Warcraft ®, η κατασκευάστρια εταιρία έχει ένα σύνολο από εξυπηρετητές οι οποίοι συνεργάζονται μεταξύ τους για την διεξαγωγή και την διατήρηση της συνοχής του κόσμου ενώ δεν υπάρχουν νόμιμα ιδιωτικοί εξυπηρετητές ανεξάρτητοι απ την κατασκευάστρια εταιρία. Υπάρχουν πολλοί ίδιοι μεταξύ τους κόσμοι, με μόνη διαφορά, τους χρήστες που υπάρχουν στο κάθε κόσμο. Οι χρήστες είναι ελεύθεροι να επιλέξουν όποιον κόσμο επιθυμούν για να παίξουν, όμως οι δράσεις τους παραμένουν περιορισμένες στον κόσμο που επέλεξαν. Ο κόσμος είναι χωρισμένος σε περιοχές και κάθε εξυπηρετητής ανήκει σε ένα τουλάχιστον σύνολο εξυπηρετητών που είναι υπεύθυνο για την διατήρηση της συνοχής μίας περιοχής. Όταν ο χρήστης αλλάζει περιοχή, τότε συνήθως πρέπει να μεταφερθεί και σε άλλο εξυπηρετητή. Εδώ λοιπόν υπάρχουν αλγόριθμοι μετανάστευσης χρηστών από έναν εξυπηρετητή σε έναν άλλον. [2]

Ωστόσο, στα παιχνίδια τύπου RTS, real time strategy, και MOBA, multiplayer online battle arena, τα πράγματα είναι λίγο διαφορετικά. [3] [4] Τα παιχνίδια αυτά είναι στρατηγικής και οι παίχτες συχνά έχουν μία βαθμολογία ικανότητας. Όπως και στο σκάκι, όπου υπάρχει το σύστημα ELO, έτσι και σε αυτά τα παιχνίδια υπάρχει ένα σύστημα βαθμολογίας που δείχνει ουσιαστικά τις ικανότητες του παίχτη. Μία υψηλή βαθμολογία καταδεικνύει έναν παίχτη μεγάλων ικανοτήτων ενώ μία χαμηλή βαθμολογία καταδεικνύει έναν άπειρο παίχτη. Ωστόσο είναι πιο πιθανό παίχτες ίδιου επιπέδου ή μικρής διαφοροποίησης να θελήσουν να παίξουν μεταξύ τους. Στα RTS τα παιχνίδια είναι συνήθως μεταξύ δύο παιχτών. Στα MOBA όμως, τα παιχνίδια είναι

μεταξύ τουλάχιστον έξι και συνηθέστερα δέκα παιχτών, χωρισμένων σε δύο ομάδες. Έτσι λοιπόν, θα πρέπει οι παίχτες του ίδιου δυναμικού, να βρίσκονται σε εξυπηρετητές οι οποίοι να μην έχουν μεγάλη δικτυακή απόσταση μεταξύ τους.

Έχει λοιπόν ενδιαφέρον να μελετήσουμε πως γίνεται να βελτιστοποιήσουμε την εμπειρία των χρηστών μειώνοντας τους χρόνους αναμονής για σύνδεση στον εξυπηρετητή καθώς και τους χρόνους για εύρεση συμπαικτών, οι οποίοι ωστόσο να πληρούν κριτήρια που θα θέσουμε αργότερα.

Ανάλυση απαιτήσεων για παιχνίδια MOBA

Για να κατανοήσουμε τις απαιτήσεις για τα παιχνίδια αυτού του τύπου, πρέπει πρώτα να γνωρίζουμε τι είναι ένα MOBA παιχνίδι και πως παίζεται.

Κάθε παιχνίδι MOBA αποτελείται από δύο ομάδες, συνηθέστερα των πέντε ατόμων. Κάθε παίκτης αναλαμβάνει τον χειρισμό ενός ήρωα της επιλογής του, ο καθ' ένας με διαφορετικές ειδικές ικανότητες. Το παιχνίδι διεξάγεται σε έναν χάρτη όπου σε δύο αντίθετες πλευρές βρίσκεται η βάση της κάθε ομάδας.

Ο χάρτης αυτός έχει κάποια μονοπάτια τα οποία είναι ευπρόσιτα και κάποια κομμάτια ζούγκλας που είναι δυσπρόσιτα. Οι παίκτες κάθε ομάδας πρέπει να χωριστούν μέσα στον χάρτη και να αντιμετωπίσουν την αντίπαλη ομάδα, συνεργαζόμενοι με τρόπους, τέτοιους ώστε να ξεγελούν και να εξοντώνουν τους αντίπαλους ήρωες.

Αφού εξοντωθεί ένας ήρωας, ύστερα από ένα χρονικό διάστημα αναγεννάται. Για κάθε εξόντωση, ο παίκτης παίρνει κάποιους πόντους, τους πόντους εμπειρίας, και όταν συμπληρώσει έναν προκαθορισμένο αριθμό από αυτούς, ανεβαίνει επίπεδο ικανότητας. Με κάθε επίπεδο που αποκτά ο παίκτης, ο ήρωάς του αποκτά και πιο δυνατές επιθέσεις. Επίσης με κάθε εξόντωση, οι παίκτες αποκτούν χρυσό, εικονικά νομίσματα δηλαδή, με τα οποία μπορούν να αγοράσουν εξοπλισμό από τη βάση τους είτε για να προκαλούν μεγαλύτερη ζημιά στους αντιπάλους είτε για να μειώνουν τη ζημιά που προκαλείται από τους αντιπάλους. Αντικειμενικός σκοπός του παιχνιδιού είναι η καταστροφή της αντίπαλης βάσης. Σε γενικές γραμμές αυτά συνοψίζουν τι είναι ένα παιχνίδι MOBA και πως αυτό παίζεται.

Η ταχύτητα εκτέλεσης επιθέσεων και ειδικών ικανοτήτων, η ικανότητα διαφυγής και πολλά άλλα κρίνουν την ικανότητα ενός παίχτη. Όπως είναι προφανές, όσο πιο ικανός είναι ένας παίχτης, τόσο πιο πολλές νίκες θα έχει. Θα ήταν άδικο λοιπόν για έναν καινούριο παίχτη να παίξει με έναν έμπειρο

παίχτη ο οποίος έχει πολλά παιχνίδια στο δυναμικό του. Δημιουργείται έτσι η απαίτηση οι παίχτες ίδιων ικανοτήτων να παίζουν μεταξύ τους, για να είναι δίκαιο το παιχνίδι.

Τα παραπάνω υπονοούν ότι θα πρέπει να γίνεται κάποια επιλογή των παικτών που θα συμμετέχουν σε ένα παιχνίδι. Δεν θα πρέπει ο κάθε χρήστης να μπορεί να συμμετέχει σε οποιοδήποτε παιχνίδι, εκτός αν το επιθυμεί συγκεκριμένα και είναι σε συνεννόηση με τους υπόλοιπους παίκτες, το οποίο είναι περίπτωση που δεν θα μελετήσουμε σε αυτή την εργασία. Ο κάθε παίκτης, λοιπόν, όταν θα επιθυμεί να παίξει ένα παιχνίδι, θα πρέπει να βρεθεί σε ένα παιχνίδι δίκαιο, που οι ομάδες να έχουν ίδια επίπεδα ικανότητας. Υπάρχει απαίτηση, συνεπώς, για διαλογή των χρηστών που θα συμμετέχουν σε ένα παιχνίδι, από τον εξυπηρετητή.

Ωστόσο, όσο και αν προτιμούν οι παίχτες να βρίσκονται σε ένα δίκαιο παιχνίδι, θα προκαλούνταν μεγάλη ενόχληση, σε αυτούς, αν ο χρόνος που χρειαζόταν για να βρεθούν συμπαίκτες ήταν πολύ μεγάλος. Πρέπει συνεπώς ο αλγόριθμος επιλογής των συμπαικτών να είναι δυναμικός σε σχέση με τον χρόνο.

Τα παιχνίδια MOBA χαρακτηρίζονται επίσης από μεγάλες ταχύτητες δράσης. Κάθε παίχτης πρέπει να είναι σε επιφυλακή διαρκώς καθώς επίσης πρέπει να μπορεί να λαμβάνει γρήγορα αποφάσεις. Δεδομένου ότι το παιχνίδι είναι πραγματικού χρόνου, δηλαδή, ότι κάνει ένας χρήστης του ίδιου παιχνιδιού, αυτό μεταφέρεται άμεσα και στους υπόλοιπους, πρέπει ο παίχτης να είναι σε ετοιμότητα να αποφεύγει επιθέσεις ή να πραγματοποιεί κινήσεις που πιστεύει ότι θα τον ωφελήσουν και αυτές να μεταφέρονται άμεσα σε όλους τους υπόλοιπους παίχτες.

Αυτή η παράμετρος της άμεσης επικοινωνίας μας βάζει φραγμούς για το πόσο μεγάλη επιτρέπεται να είναι η χρονοκαθυστέρηση συγχρονισμού όπως επίσης και η χρονοκαθυστέρηση μεταφοράς των κινήσεων των πιο απομακρυσμένων χρηστών μεταξύ αυτών. [3]

Τέλος, σημαντικό κομμάτι σε οποιαδήποτε διαδικτυακή εφαρμογή είναι η αναμονή στην ουρά για την χρήση της υπηρεσίας. Έτσι και στα παιχνίδια, η αναμονή στην ουρά για την σύνδεση στον εξυπηρετητή είναι σημαντικός παράγοντας και πρέπει να ληφθεί υπ' όψιν. Κάθε εξυπηρετητής έχει δυνατότητα να εξυπηρετήσει ταυτόχρονα ένα πεπερασμένο αριθμό πελατών, που εξαρτάται από το υλισμικό του καθώς και από τον καλό ή μη σχεδιασμό του προγράμματος εξυπηρέτησης.

Μοντελοποίηση των απαιτήσεων σε αλγορίθμους

Αρχικά θα πρέπει να λάβουμε υπ' όψιν αν θα υπάρχει κάποιος τρόπος διαχωρισμού των χρηστών στους εξυπηρετητές. Η μία επιλογή είναι να μην υπάρχει κάποια διάκριση, όποιος χρήστης επιθυμεί να συνδεθεί, να συνδέεται σε όποιον εξυπηρετητή επιθυμεί. Αν κανένας εξυπηρετητής δεν μπορεί να τον εξυπηρετήσει, αναμένει στην ουρά όποιου εξυπηρετητή επιθυμεί.

Η δεύτερη πρόταση είναι, ο κάθε χρήστης, όταν επιθυμεί να συνδεθεί, να συνδέεται στον εξυπηρετητή ο οποίος έχει την μικρότερη ικανότητα εξυπηρέτησης αλλά μπορεί να εξυπηρετήσει τον χρήστη που επιθυμεί να συνδεθεί. Αν ένας εξυπηρετητής δεν μπορεί να εξυπηρετήσει τον χρήστη, αποκλείεται από την επιλογή. Αν όλοι οι εξυπηρετητές προς επιλογή έχουν την ίδια ικανότητα εξυπηρέτησης, επιλέγεται ο πρώτος που προσπελάστηκε από τον χρήστη. Αν κανένας εξυπηρετητής δεν μπορεί να εξυπηρετήσει τον χρήστη, τότε αναμένει σε αυτόν με τη μικρότερη ουρά αναμονής.

Η τρίτη πρόταση διαχωρισμού των χρηστών σε εξυπηρετητές είναι σύμφωνα με την ικανότητά τους. Κάθε εξυπηρετητής δέχεται να εξυπηρετήσει αποκλειστικά πελάτες ενός συγκεκριμένου φάσματος δυναμικού ικανότητας. Λαμβάνεται υπ' όψιν ότι πρέπει να μπορούν να συνδεθούν όλοι οι χρήστες, οπότε πρέπει οι εξυπηρετητές να καλύπτουν όλο το φάσμα των δυναμικών ικανότητας των χρηστών. Αν ένας χρήστης δεν μπορεί να εξυπηρετηθεί, περιμένει στην ουρά του εξυπηρετητή έως ότου είναι δυνατή η εξυπηρέτησή του.

Η τέταρτη πρόταση είναι παρόμοια με την τρίτη. Κάθε εξυπηρετητής δέχεται να εξυπηρετήσει αποκλειστικά πελάτες ενός συγκεκριμένου φάσματος δυναμικού ικανότητας. Λαμβάνεται υπ' όψιν ότι πρέπει να μπορούν να συνδεθούν όλοι οι χρήστες, οπότε πρέπει οι εξυπηρετητές να καλύπτουν όλο το φάσμα των δυναμικών ικανότητας των χρηστών. Αν ένας χρήστης δεν μπορεί να εξυπηρετηθεί ελέγχει αν μπορεί να εξυπηρετηθεί από γειτονικούς εξυπηρετητές. Θα πρέπει να συνδεθεί ή στον αμέσως επόμενο ή στον αμέσως προηγούμενο. Αν δεν μπορεί να εξυπηρετηθεί ούτε από τους γειτονικούς, τότε μπαίνει στην ουρά του αρχικού εξυπηρετητή.

Φυσικά, εκτός από τον τρόπο σύνδεσης των χρηστών στους εξυπηρετητές, υπάρχει και η παράμετρος, του πως χειρίζονται οι εξυπηρετητές την ουρά αναμονής.

Η μία περίπτωση είναι να έχουμε πρωτόκολλο FIFO, first in first out, ή στην συγκεκριμένη περίπτωση, first come first served, και να έχουμε μία ομαλή κίνηση της ουράς. Αν υπάρχει ουρά, τότε πρέπει όλοι οι νέοι χρήστες

που επιθυμούν να συνδεθούν στον εξυπηρετητή να περιμένουν σε αυτή είτε μπορεί να τους εξυπηρετήσει ο εξυπηρετητής, είτε όχι.

Η άλλη περίπτωση είναι να έχουμε ένα άπληστο αλγόριθμο. Σε αυτόν, ο εξυπηρετητής προσπαθεί να εξυπηρετήσει όλους, ανεξάρτητα από το μέγεθος της ουράς. Αν υπάρχει ουρά λοιπόν, ένας χρήστης που θέλει να συνδεθεί μπαίνει στο τέλος της. Ο εξυπηρετητής εξετάζει με σειρά άφιξης τους χρήστες και επιλέγει τον πρώτο που μπορεί να εξυπηρετήσει και τον εξυπηρετεί.

Η άπληστη εκδοχή του αλγορίθμου, αν και εξυπηρετεί ταχύτερα μεγαλύτερο αριθμό χρηστών και συνεπώς αυξάνει το κέρδος, μπορεί να δημιουργήσει αδικία, καθώς δεν θα εξυπηρετηθεί αυτός που ήρθε πρώτος, αλλά αυτός που μπορεί να εξυπηρετηθεί γρηγορότερα, με αποτέλεσμα να δημιουργείται η πιθανότητα κάποιος «άτυχος» χρήστης να μην μπορεί να εξυπηρετηθεί κατά τη διάρκεια των ωρών αιχμής.

Τους παραπάνω αλγορίθμους θα τους αναλύσουμε σε διαγράμματα ροής και ψευδοκώδικα σε επόμενο κεφάλαιο, ύστερα απ την μοντελοποίηση του προβλήματος.

III. Χρήστες και άφιξη χρηστών σε online services

Ρυθμός άφιξης Χρηστών

Είναι γενικά κοινώς αποδεκτό ότι οι χρόνοι άφιξης πελατών σε ένα σύστημα μπορούν να μοντελοποιηθούν με τη χρήση κατανομής Poisson. [5]Εμείς επιλέξαμε μία κάπως διαφορετική προσέγγιση.

Ερχόμενοι σε επικοινωνία με κατασκευάστριες εταιρίες παιχνιδιών MOBA, μπορέσαμε να μάθουμε κάποιες λεπτομέρειες για το traffic τέτοιων παιχνιδιών. Συγκεκριμένα, για το παιχνίδι MOBA League of Legends ®, μάθαμε τα εξής στοιχεία από την RIOT Games ®:

- Ο μέγιστος αριθμός χρηστών που έχει σημειωθεί ποτέ για το συγκεκριμένο παιχνίδι είναι 5 εκατομμύρια χρήστες ταυτόχρονα. Αυτό αφορά παγκόσμιο πληθυσμό και όχι μίας χώρας. [6]

- Κατά μέσο όρο συνδέονται 12 εκατομμύρια χρήστες ημερησίως. [7]

- Συνολικά σε μία ημέρα παίζονται 10,5 εκατομμύρια ώρες. Αυτό αναλογεί σε περίπου 52 λεπτά και 30 δευτερόλεπτα παιχνιδιού για κάθε χρήστη. [8]

- Την περίοδο διεξαγωγής της εργασίας υπήρχαν 7 ενεργά regions:

    - North America
    - EU West
    - EU Nordic & East
    - Brazil
    - Turkey
    - Russia
    - Korea

Κάθε region πρακτικά είναι ένα σύνολο εξυπηρετητών που εδρεύουν στην περιοχή με το όνομά της, δηλαδή οι εξυπηρετητές του Region North America βρίσκονται κατά κόρον στην Βόρεια Αμερική.

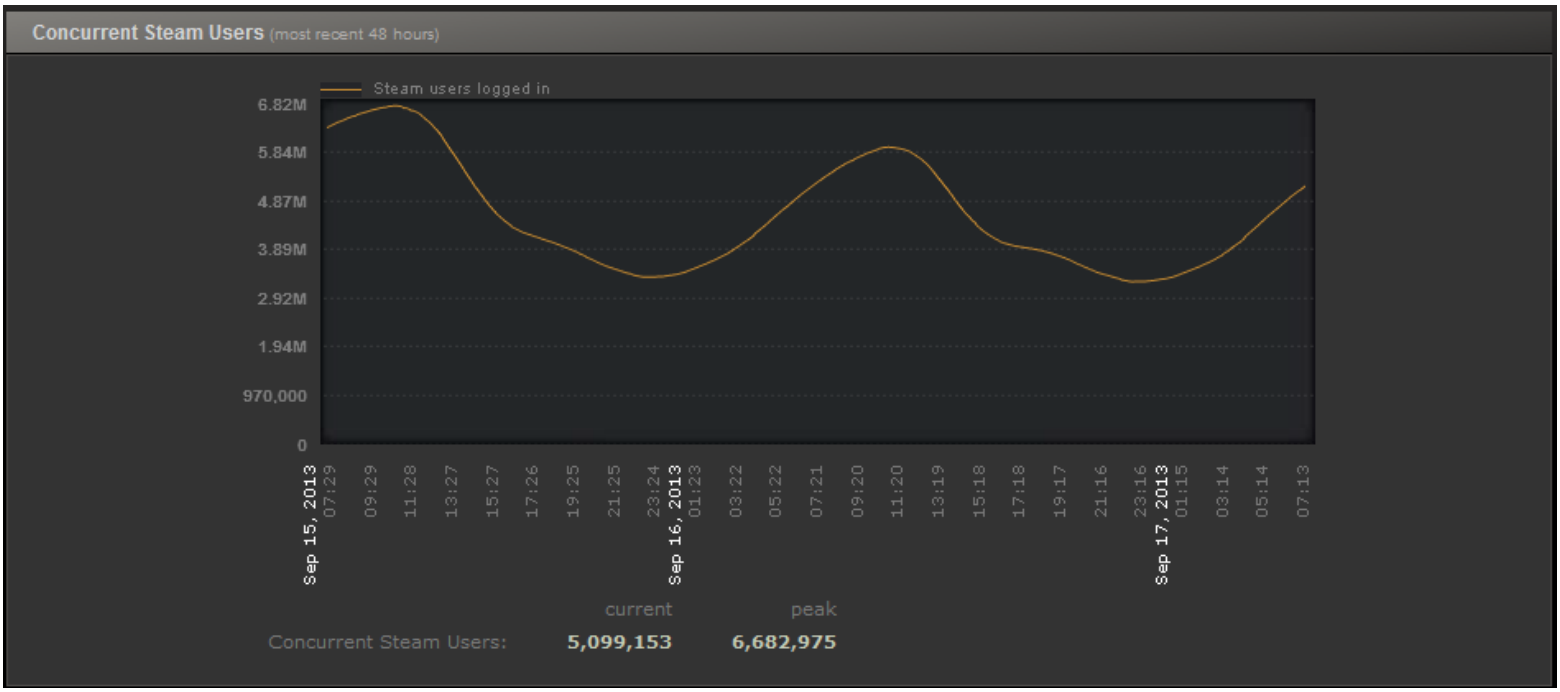
Δεδομένης της ύπαρξης πολλών region και του γεγονότος ότι το σύνολο των χρηστών είναι κατανεμημένο παγκοσμίως, είναι λογική η υπόθεση ότι κάθε region θα έχει το $\frac{1}{Αριθμός\ Region}$ του παγκόσμιου πληθυσμού. Αφού λοιπόν υπάρχουν 7 regions θα πρέπει τα παραπάνω στοιχεία να τα διαιρέσουμε με το 7. Έτσι θα έχουμε λοιπόν:

- Ο μέγιστος αριθμός ταυτόχρονων χρηστών σε ένα region είναι 714.285 χρήστες.
- Κατά μέσο όρο συνδέονται 1.714.285 χρήστες ημερησίως.
- Συνολικά σε μία ημέρα παίζονται 1.500.000 εκατομμύρια ώρες σε κάθε region. Αυτό αναλογεί σε περίπου 52 λεπτά και 30 δευτερόλεπτα παιχνιδιού για κάθε χρήστη.

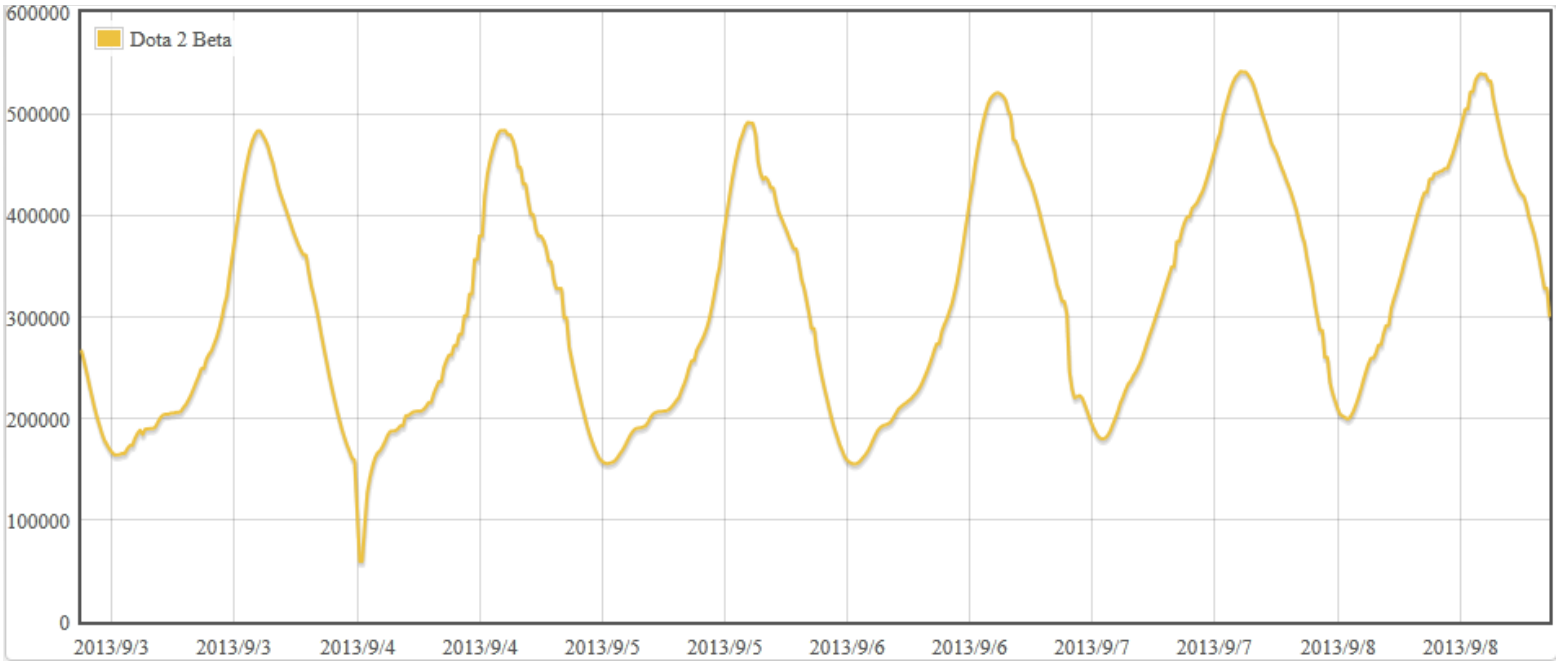
Έχοντας πλέον την δυνατότητα να μελετήσουμε κάθε region αυτόνομα, μπορούμε πολύ πιο εύκολα να εξάγουμε δεδομένα.

Ο αριθμός των ταυτόχρονων χρηστών

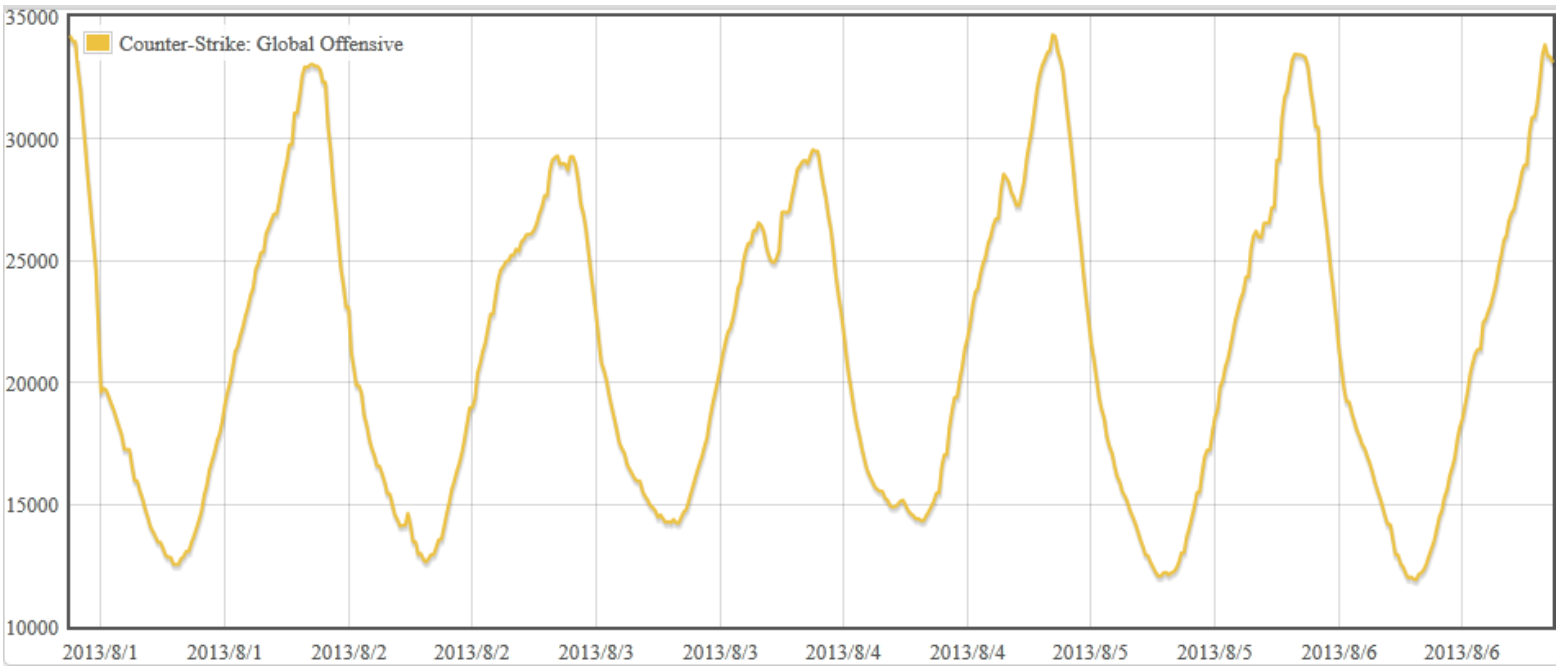
Ύστερα από εμπειρική παρατήρηση των ταυτόχρονων χρηστών, διαφόρων παιχνιδιών κατά τη διάρκεια μερικών ημερών, μπορούμε με ευκολία να παρατηρήσουμε ότι ακολουθούν μία ημιτονοειδή καμπύλη. Παρακάτω θα παραθέσω τα γραφήματα ταυτόχρονων χρηστών της πλατφόρμας παιχνιδιών Steam για το διάστημα 15/09/2013 με 17/09/2013 [9] [10]καθώς και τα γραφήματα ταυτόχρονων χρηστών, γνωστών MMOG σε τυχαία μεταξύ τους διαστήματα.



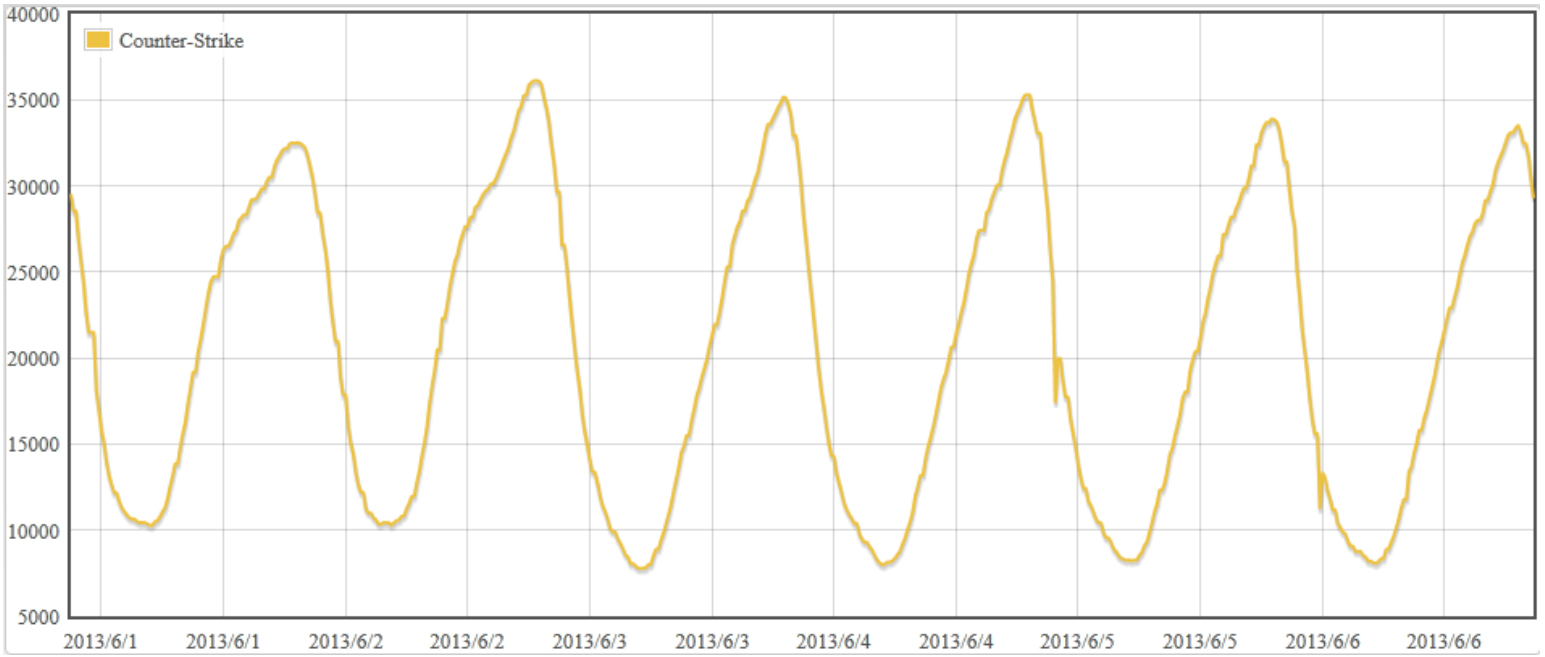
Διάγραμμα 1 - Το γράφημα ταυτόχρονων χρηστών της πλατφόρμας Steam ® για το χρονικό διάστημα 15/09/2013 ως 17/09/2013



Διάγραμμα 2 - Το γράφημα ταυτόχρονων χρηστών του γνωστού MOBA DOTA 2 ® για το χρονικό διάστημα 3/09/2013 ως 8/09/2013



Διάγραμμα 3 - Το γράφημα ταυτόχρονων χρηστών του γνωστού FPS Counter-Strike: Global Offensive ® για το χρονικό διάστημα 1/08/2013 ως 6/08/2013



Διάγραμμα 4 - Το γράφημα ταυτόχρονων χρηστών του γνωστού FPS Counter-Strike ® για το χρονικό διάστημα 1/06/2013 ως 6/06/2013

Παρατηρούμε λοιπόν ότι παρά τις διάφορες μικρομεταβολές στην κλίση και διάφορες απότομες πτώσεις του αριθμού των χρηστών, που είναι πιθανό να οφείλονται σε κάποια αστοχία υλικού των εξυπηρετητών, γενικά υπάρχει μία ημιτονοειδής τάση στον αριθμό των ταυτόχρονων χρηστών.

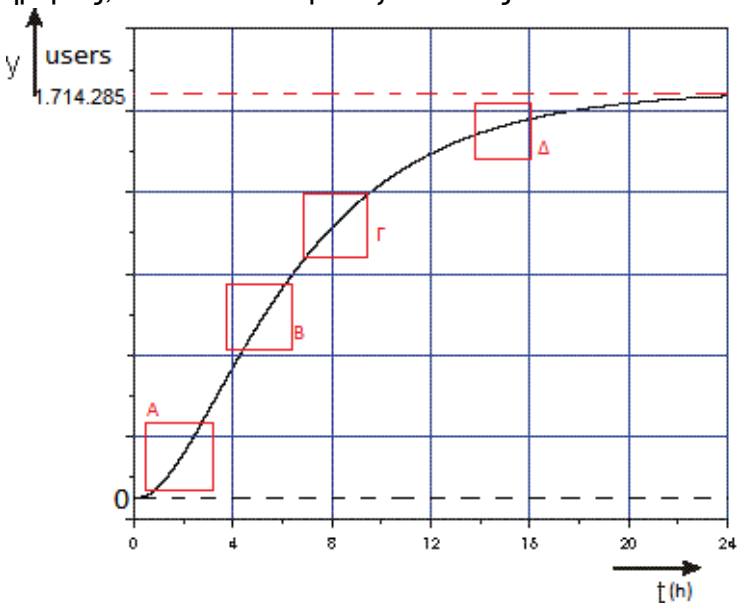
Η παραπάνω υπόθεση επιβεβαιώνεται από το Διάγραμμα 1. Στο Διάγραμμα 1 βλέπουμε τον αριθμό των ταυτόχρονων χρηστών όλων των παιχνιδιών που υποστηρίζει η πλατφόρμα Steam στη διάρκεια δύο ημερών. Παρατηρούμε ότι ακόμα και σε αυτό το διάγραμμα, που είναι αθροιστικό πολλών μικρότερων, υπάρχει η τάση της ημιτονοειδούς κίνησης των χρηστών.

Η σημασία της ημιτονοειδούς συμπεριφοράς

Η ημιτονοειδής αυτή συμπεριφορά μας αποδεικνύει ότι υπάρχουν ώρες αιχμής και συνεπώς ότι μέσα στην ημέρα αλλάζει ο ρυθμός σύνδεσης των χρηστών. Επίσης είναι προφανές ότι δεν είναι ένα γραμμικό σύστημα αλλά ότι μάλλον είναι εκθετικό.

Ας θεωρήσουμε λοιπόν ότι μελετάμε ένα region το οποίο μόλις ξεκίνησε να λειτουργεί, ο αρχικός αριθμός χρηστών του είναι 0 και στο τέλος της ημέρας θα έχει πάλι 0 χρήστες, πράγμα το οποίο φυσικά δεν ισχύει στην πραγματικότητα, αλλά χάριν μελέτης το θεωρούμε έτσι.

Αν είχαμε ένα γράφημα το οποίο μας δείχνει πόσοι χρήστες έχουν συνδεθεί συνολικά από την αρχή λειτουργίας του region μέχρι το τέλος της ημέρας, τότε αυτό θα μοιάζει κάπως έτσι:



Διάγραμμα 5 - Αθροιστικό γράφημα συνδέσεων στη διάρκεια μιας ημέρας για ένα region

Η παράγωγος του Διαγράμματος 5 μας δίνει τον ρυθμό μεταβολής των συνδέσεων, μας δίνει δηλαδή για κάθε t πόσοι χρήστες συνδέονται εκείνη την στιγμή. Τα επισημασμένα σημεία Α,Β,Γ,Δ είναι άξια προσοχής και μας δείχνουν τους διαφορετικούς ρυθμούς σύνδεσης χρηστών.

- Α: Παρατηρούμε ότι υπάρχει μεγάλη κλίση της καμπύλης και μάλιστα δημιουργείται κυρτή καμπύλη, απόδειξη του μεγάλου και θετικού ρυθμού μεταβολής συνδέσεων.
- Β: Ο ρυθμός μεταβολής του αριθμού των συνδέσεων τείνει προς ένα μέγιστο
- Γ: Ο ρυθμός μεταβολής του αριθμού των συνδέσεων είναι πλέον στο μέγιστο και από το Γ και μετά αρχίζει να μειώνεται έως ότου γίνει 0
- Δ: Ο ρυθμός μεταβολής του αριθμού των συνδέσεων πλέον είναι 0 και έχουν συνδεθεί όλοι οι χρήστες.

Να παρατηρηθεί ότι χρησιμοποιούμε τον όρο ρυθμός μεταβολής συνδέσεων και όχι ρυθμός μεταβολής χρηστών. Ο ρυθμός μεταβολής των συνδεδεμένων χρηστών είναι διαφορετικός και θα επεξηγηθεί στο επόμενο κεφάλαιο.

Ρυθμός Μεταβολής Συνδεδεμένων χρηστών ή ρυθμός μεταβολής των συνδέσεων;

Το Διάγραμμα 5 είναι μία οπτική απεικόνιση του συνολικού αριθμού συνδέσεων σε μία ημέρα. Είναι πολύ εύκολο κάποιος να μπερδευτεί και να θεωρήσει ότι η παράγωγος αυτής της καμπύλης είναι και ο ρυθμός μεταβολής των ταυτοχρόνως συνδεδεμένων χρηστών.

Όπως προαναφέραμε, κάθε χρήστης μένει στο region περίπου 52 λεπτά και 30 δευτερόλεπτα κατά μέσο όρο. Υπάρχει ένα πανομοιότυπο λοιπόν γράφημα, μετατοπισμένο κατά 52 λεπτά και 30 δευτερόλεπτα αργότερα το οποίο μας δείχνει τον αριθμό των αποσυνδεδεμένων χρηστών αθροιστικά μία δεδομένη στιγμή της ημέρας. Αυτό σημαίνει ο ρυθμός μεταβολής των συνδέσεων μία δεδομένη στιγμή t εξαρτάται άμεσα από τον αριθμό των αποσυνδέσεων την ίδια στιγμή. Αν γι αυτή την έρευνα θεωρήσουμε ότι κάθε χρήστης παραμένει συνδεδεμένος για ακριβώς 52 λεπτά και 30 δευτερόλεπτα τότε μπορούμε εύκολα να συμπεράνουμε τα εξής:

- Αν C(t) είναι η συνάρτηση η οποία μας δείχνει το αθροιστικό αριθμό των χρηστών που συνδέθηκαν ως την στιγμή t με 0<t και με το t να είναι δευτερόλεπτα
- Αν D(t) είναι η συνάρτηση η οποία μας δείχνει το αθροιστικό αριθμό των χρηστών που αποσυνδέθηκαν ως την στιγμή t με 3150<t με το t να είναι δευτερόλεπτα

- Τότε C(t)-D(t)=u(t) μία νέα συνάρτηση η οποία μας δίνει τον αριθμό των ταυτόχρονα συνδεδεμένων χρηστών τη στιγμή t.
  Για t<3150 u(t)=C(t)
  Για t≥3150 u(t)=C(t)-D(t)

- Επίσης D(t)=C(t-m), όπου m ο μέσος χρόνος που μένει κάποιος συνδεδεμένος στο παιχνίδι, και συνεπώς θα έχουμε :
  u(t)=C(t)-C(t-m).

Η u(t) λοιπόν είναι μία πολύ σημαντική συνάρτηση η οποία μας δείχνει για κάθε t τον αριθμό των συνδεδεμένων χρηστών. Μία u(t) συνάρτηση είναι που παράγει τα γραφήματα των Διαγραμμάτων 1-4. Η παράγωγος της u(t) μας δείχνει τον ρυθμό μεταβολής των συνδεδεμένων χρηστών.

Έτσι λοιπόν, δεδομένου ότι είχαμε στα χέρια μας στοιχεία και για τον ρυθμό μεταβολής των συνδέσεων αλλά και για τον ρυθμό μεταβολής των συνδεδεμένων χρηστών ήταν εύκολο να φτιάξουμε μία δική μας συνάρτηση η οποία να μας δείχνει ακριβώς πόσοι χρήστες συνδέθηκαν μία δεδομένη στιγμή στο παιχνίδι χωρίς να χρειάζεται να χρησιμοποιήσουμε θεωρία ουρών και την κατανομή Poisson.

Η συνάρτηση C(t)

Για να μπορέσουμε να έχουμε την παράγωγο της C(t) πρέπει να γνωρίζουμε και την C(t). Από την μελέτη συστημάτων αυτόματου ελέγχου, μπορούμε να δούμε ότι τα συστήματα με χαρακτηριστική καμπύλη σαν αυτή της C(t) ονομάζονται Συστήματα με αναλογική συμπεριφορά και καθυστέρηση ανώτερης τάξης, συγκεκριμένα μοιάζει αρκετά με τρίτης τάξης (P-T_3). Η χαρακτηριστική καμπύλη των συστημάτων αυτών λαμβάνεται από την συνάρτηση:

$$Uout = K * \left(1 - e^{-\frac{t}{T1}}\right) * \left(1 - e^{-\frac{t}{T2}}\right) * \left(1 - e^{-\frac{t}{T3}}\right) * Uin$$

με την εξής επεξήγηση μεταβλητών. K είναι ο μέγιστος αριθμός U_{out} για $u_{in=}$ 1. T_1, T_2, T_3 είναι κάποιες σταθερές χρόνου οι οποίες ορίζονται ως ο χρόνος που χρειάζεται για να γίνει το U_{out} ίσο με το 63% του μέγιστου $U_{out.}$ και συνήθως βρίσκονται πειραματικά. [11]

Στην δική μας περίπτωση όμως δεν έχουμε τάσεις, αλλά χρήστες. Επίσης, δεν έχουμε πολλαπλά T, αλλά ένα, το T1. Στην περίπτωση που υπάρχει μοναδικός συντελεστής χρόνου T, τότε το σύστημα εξισορροπείται,

φτάνει πολύ κοντά στην τελική του τιμή δηλαδή, σε χρόνο 4*T1.Αν αντί για U_{out} πούμε Αριθμός συνδέσεων και για u_{in} έχουμε 1 τότε η συνάρτηση γίνεται .

$$C(t) = K * \left(1 - e^{-\frac{t}{T1}}\right)^3$$

Αν στη θέση του K βάλουμε τον δικό μας μέγιστο αριθμό χρηστών, δηλαδή 1.714.285 και στη θέση του τ_1 βάλουμε το ¼ του χρόνου που χρειάζεται για να γίνουν όλες οι συνδέσεις, δηλαδή 8 ώρες ή 21600 δευτερόλεπτα, τότε έχουμε την εξής συνάρτηση

$$C(t) = 1.714.285 * \left(1 - e^{-\frac{t}{21600}}\right)^3$$

Έχοντας λοιπόν πλέον την συνάρτηση είναι πολύ εύκολο να υπολογίσουμε την παράγωγό της η οποία είναι:

$$C'(t) = \frac{-34285 * (1 - e^{-\frac{x}{21600}})^2 * e^{-\frac{x}{21600}}}{1440}$$

και μας δίνει τον αριθμό συνδέσεων που συμβαίνουν σε κάθε χρονική στιγμή t.

Ωστόσο με μερικούς γρήγορους υπολογισμούς θα παρατηρήσουμε ότι οι παραπάνω συναρτήσεις δεν θα μας δώσουν τα μέγιστα τα οποία εμείς επιζητούμε. Συγκεκριμένα η u(t) δεν θα μας δώσει κάποιο συναρπαστικό μέγιστο αριθμό πόσο μάλλον το 714.285 που επιζητούμε. Γιατί συμβαίνει αυτό;

Υπάρχει μία παράμετρος που μέχρι στιγμής έχουμε αμφισβητήσει αρκετά. Αυτή είναι η τυχαιότητα του πληθυσμού. Για να δημιουργηθεί ένα μέγιστο χρηστών όπως εμείς το επιθυμούμε, πρέπει να υπάρξουν συνθήκες τέτοιες που να το επιφέρουν. Θα πρέπει στιγμιαία να υπάρχει ραγδαία αύξηση των ταυτόχρονων συνδεδεμένων χρηστών και ο μέσος χρόνος παραμονής στο region να αυξηθεί κατά πολύ. Αν και οι παραπάνω συναρτήσεις είναι σωστές, δεν θα μας εξυπηρετήσουν αρκετά στην μελέτη μας. Ωστόσο είναι πολύ σημαντικές και άξιο θέμα για μελλοντική μελέτη.

Παραμένει όμως η απορία του πως θα καθορίσουμε τον ρυθμό σύνδεσης των χρηστών. Δεδομένου ότι μας ενδιαφέρει να μελετήσουμε το σύστημά μας κυρίως υπό μεγάλο φόρτο, μπορούμε να θεωρήσουμε ότι συνδέονται σε πολύ μικρό χρονικό διάστημα της τάξης των 45 λεπτών και 714.285 χρήστες. Ο συνολικός πληθυσμός στη διάρκεια μίας ημέρας δεν μας αφορά ιδιαιτέρως καθώς αυτό που έχει σημασία είναι το πώς θα ανταποκριθεί το σύστημα υπό μεγάλο φόρτο. Έτσι αρκεί να φτιάξουμε μία κατανομή που να συνδέονται περί των 15.873 χρηστών ανά δευτερόλεπτο. Σε 45 λεπτά θα έχουμε ακριβώς 714.285 χρήστες.

Κάποιος θα μπορούσε να ρωτήσει: «Αφού μας ενδιαφέρει να δούμε το σύστημα σε φόρτο, γιατί δεν συνδέουμε και τους 714.285 χρήστες ταυτόχρονα;». Κάτι τέτοιο απορρίφθηκε στη διάρκεια του σχεδιασμού του προβλήματος καθώς δεν θα φαινόταν καθαρά σε εφαρμογή η χρήση των αλγορίθμων. Θεωρήθηκε πιο σωστό να υπάρχει ένας αριθμός χρηστών που να αυξάνεται συνεχώς, για να είναι φανερή και η εξυπηρέτηση μέσα στη διάρκεια του χρόνου.

Επιπλέον, θα τρέξουμε έναν αριθμό προσομοιώσεων που θα προσπαθήσει να συνδέσει τον συνολικό αριθμό χρηστών που συνδέονται σε μία ημέρα, για να δούμε τι συμβαίνει εκτός ωρών αιχμής.

IV. Μοντελοποίηση του προβλήματος σύνδεσης χρηστών

Τι μας ενδιαφέρει πραγματικά;

Υπάρχουν πολλές παράμετροι οι οποίες αλλάζουν τις απαιτήσεις ενός χρήστη από έναν εξυπηρετητή. Από την ικανότητα προώθησης πακέτων του backbone του παρόχου διαδικτύου του ως την ταχύτητα της μνήμης RAM του υπολογιστή του, κάθε μικρή διαφοροποίηση κάνει τις απαιτήσεις κάθε χρήστη διαφορετικές. Μπορεί κάποιος να πει ότι κάθε χρήστης έχει και διαφορετικές απαιτήσεις.

Αυτό που μας ενδιαφέρει από τη μεριά του εξυπηρετητή είναι πόσο συχνά ένας χρήστης θα ανταλλάξει πληροφορία. Υπάρχει ένας μέγιστος ρυθμός ανταλλαγής πληροφοριών και κάθε μικρότερος είναι υποδιαίρεση αυτού. Αν θεωρήσουμε λοιπόν ότι ένας εξυπηρετητής έχει τη δυνατότητα να επικοινωνήσει με X χρήστες ταυτόχρονα όταν είναι στο όριο των δυνατοτήτων του, τότε μπορούμε να πούμε ότι έχει χωρητικότητα X χρηστών. Στη πραγματικότητα ο αριθμός των χρηστών που είναι συνδεδεμένοι σε έναν εξυπηρετητή θα είναι πάντα ίσος ή μεγαλύτερος από X αυτό συμβαίνει για τον εξής λόγο:

- Έστω ότι ο εξυπηρετητής επικοινωνεί κάθε Z δευτερόλεπτα με τους χρήστες στη βέλτιστη περίπτωση.
- Αν δύο χρήστες λόγω των απαιτήσεών τους επικοινωνούν ανά 2*Z δευτερόλεπτα με τον εξυπηρετητή, τότε πιάνουν χώρο ενός χρήστη, καθώς ο καθ' ένας τους απασχολεί τον εξυπηρετητή κατά τον μισό χρόνο και συνεπώς προκαλεί τον μισό φόρτο εργασίας.

Μπορούμε λοιπόν σύμφωνα με τα παραπάνω να πούμε ότι ο εξυπηρετητής έχει ένα σύνολο μονάδων που δείχνουν την δυνατότητά του στο να εξυπηρετεί πελάτες. Οι μονάδες αυτές θα πρέπει να είναι σε αριθμό ίσες με τη μέγιστη χωρητικότητα του εξυπηρετητή. Κάθε πελάτης που απαιτεί να συνδεθεί με τον εξυπηρετητή αφαιρεί από τις μονάδες δυνατότητας του εξυπηρετητή ένα μέρος αυτών που είναι ένας αριθμός μέσα στο συνεχές διάστημα (0,1]. Ο αριθμός αυτός δείχνει πόση από την δυνατότητα εξυπηρετητή που δικαιούται, χρησιμοποιεί πραγματικά.

Αυτό που μας ενδιαφέρει λοιπόν είναι το πόσες μονάδες απομένουν σε έναν εξυπηρετητή, δεδομένου ότι αυτές είναι ένας πεπερασμένος αριθμός

που εξαρτάται από τις δυνατότητές εξυπηρέτησής του. Ας ονομάσουμε λοιπόν αυτές τις μονάδες Shares.

Οι αλγόριθμοι εξυπηρέτησης σε δεύτερη ανάλυση

Στο Δεύτερο μέρος αυτής της εργασίας περιγράψαμε έναν αριθμό αλγορίθμων χωρίς να δώσουμε την αναπαράσταση σε ψευδοκώδικα αυτών. Σε αυτό το κεφάλαιο, θα περιγράψουμε τους εν λόγω αλγορίθμους με τη χρήση ψευδοκώδικα. Θεωρούμε γνωστό σε κάθε περίπτωση το πόσοι εξυπηρετητές ενός region είναι διαθέσιμοι καθώς και το ποιοι είναι αυτοί. Αυτό είναι απολύτως λογικό, καθώς όταν ένα πρόγραμμα επικοινωνεί με έναν εξυπηρετητή, ξέρει με ποιόν θα επικοινωνήσει, ή ενημερώνεται από κάποιο προκαθορισμένο σημείο με ποιόν θα επικοινωνήσει.

Αλγόριθμος πρώτος

Ο πρώτος μας αλγόριθμος, όπως είπαμε είναι να μην υπάρχει κάποια διάκριση μεταξύ των χρηστών η οποία να επηρεάζει τον τρόπο σύνδεσης. Έτσι, όποιος χρήστης επιθυμεί να συνδεθεί, συνδέεται σε όποιον εξυπηρετητή επιθυμεί. Αν κανένας εξυπηρετητής δεν μπορεί να τον εξυπηρετήσει, αναμένει στην ουρά όποιου εξυπηρετητή επιθυμεί.

```
void connectToServer()

{

	Boolean result=false;

	int ammountOfServers;//gnwsto eks arxhs

	server myPrefferenceList[ammountOfServers];
		/*lista me tous eksyphrethtes me seira proteraiothtas*/

	int counter=-1;


	while(result == false&&counter<ammountOfServers){

		counter++;

		result=myPrefferenceList[counter].askForConnection();

		//h askForConnection epistrefei true an mporei na
eksyphreth8ei o xrhsths


	}

	if(result==false)

	{

		myPrefferenceList[0].getInQueue();

	}

}
```

Αλγόριθμος δεύτερος

Ο δεύτερος αλγόριθμος είναι, ο κάθε χρήστης, όταν επιθυμεί να συνδεθεί, να συνδέεται στον εξυπηρετητή ο οποίος έχει την μικρότερη ικανότητα εξυπηρέτησης αλλά μπορεί να εξυπηρετήσει τον χρήστη που επιθυμεί να συνδεθεί. Αν ένας εξυπηρετητής δεν μπορεί να εξυπηρετήσει τον χρήστη, αποκλείεται από την επιλογή. Αν όλοι οι εξυπηρετητές προς επιλογή έχουν την ίδια ικανότητα εξυπηρέτησης, επιλέγεται ο πρώτος που προσπελάστηκε από τον χρήστη. Αν κανένας εξυπηρετητής δεν μπορεί να εξυπηρετήσει τον χρήστη, τότε αναμένει σε αυτόν με τη μικρότερη ουρά.

```
void connectToServer()

{

	Boolean result=false;

	int ammountOfServers;//gnwsto eks arxhs

	server sortedBySharesList[ammountOfServers];
		/*lista me tous eksyphrethtes me f8einousa seira ikanothtas
                                                                eksyphrethshs*/

	server sortedByQueueSizeList[ammountOfServers];

	/*lista me tous eksyphrethtes me f8einousa seira mege8ous listas
                                                                        anamonhs*/

	int counter=0;
	while(result == false&&counter<ammountOfServers){

		result= sortedBySharesList[counter].askForConnection();

		if(result==false){

		counter++;

		}
	}

	if(result==false)

	{

		sortedByQueueSizeList [0].getInQueue();

	}

	else{

	sortedBySharesList[counter].connect();

	}

}
```

Αλγόριθμος τρίτος

Ο τρίτος αλγόριθμος λειτουργεί με διαχωρισμό των χρηστών σε εξυπηρετητές σύμφωνα με την βαθμολογία ικανότητας αυτών. Αυτό προϋποθέτει ένα σύστημα το οποίο να μπορεί να κρίνει την ικανότητα αυτή. Τα πιο πολλά παιχνίδια έχουν έναν δικό τους μηχανισμό ο οποίος το κάνει αυτό αυτόματα. Στην δική μας εργασία δεν μας νοιάζει πολύ και θα θεωρήσουμε ότι είναι ένα γνωστό στον χρήστη νούμερο. Ο αριθμός αυτός είναι, στους περισσότερους μηχανισμούς αξιολόγησης, μεγαλύτερος του 800 χωρίς κάποιο ιδιαίτερο όριο, αλλά πολύ σπάνια βλέπουμε βαθμολογίες μεγαλύτερες του 2.700.

Κάθε εξυπηρετητής δέχεται να εξυπηρετήσει αποκλειστικά πελάτες ενός συγκεκριμένου συνεχούς φάσματος δυναμικού ικανότητας. Λαμβάνεται υπ' όψιν ότι πρέπει να μπορούν να συνδεθούν όλοι οι χρήστες, οπότε πρέπει οι εξυπηρετητές να καλύπτουν όλο το φάσμα των δυναμικών ικανότητας των χρηστών. Αν ένας χρήστης δεν μπορεί να εξυπηρετηθεί, περιμένει στην ουρά του εξυπηρετητή έως ότου είναι δυνατή η εξυπηρέτησή του.

```
void connectToServer()

{

	Boolean result=false;

	int ammountOfServers;//gnwsto eks arxhs

	server sortedByEloList[ammountOfServers];
		/*lista me tous eksyphrethtes. Einai diatetagmenoi me f8inousa
				seira tou fasmatws dynamikhs pou eksyphretoun */

	int counter=0;

	int serverToConnectTo;

	int myElo,serverTopElo,serverBotElo;


	while(counter<ammountOfServers){

		serverTopElo= sortedByEloList[counter].askForTopElo ();

		serverBotElo= sortedByEloList[counter].askForBotElo ();

		if(myelo≥serverBotElo && myelo<serverTopElo)

		{

			break;

		}
		counter++;
	}// auth h epanalhpsh einai bebaio oti 8a mas dwsei ton Server pros
```

```
syndesh logw ths upo8eshs oti oloi oi xrhstes eksyphretountai apo enan
server

result= sortedBySharesList[counter].askForConnection()

       if(result==false)

       {

              sortedBySharesList [counter].getInQueue();

       }

       else{

              sortedBySharesList [counter].connect();

       }


}
```

Αλγόριθμος τέταρτος

Η τέταρτη πρόταση είναι παρόμοια με την τρίτη. Κάθε εξυπηρετητής δέχεται να εξυπηρετήσει αποκλειστικά πελάτες ενός συγκεκριμένου φάσματος δυναμικού ικανότητας. Λαμβάνεται υπ' όψιν ότι πρέπει να μπορούν να συνδεθούν όλοι οι χρήστες, οπότε πρέπει οι εξυπηρετητές να καλύπτουν όλο το φάσμα των δυναμικών ικανότητας των χρηστών όπως και στην Τρίτη περίπτωση. Αν ένας χρήστης δεν μπορεί να εξυπηρετηθεί ελέγχει αν μπορεί να εξυπηρετηθεί από γειτονικούς εξυπηρετητές. Θα πρέπει να συνδεθεί ή στον αμέσως επόμενο ή στον αμέσως προηγούμενο. Αν δεν μπορεί να εξυπηρετηθεί ούτε από τους γειτονικούς, τότε μπαίνει στην ουρά του αρχικού εξυπηρετητή.

```
void connectToServer()

{

	Boolean result=false;

	int ammountOfServers;//gnwsto eks arxhs

	server sortedByEloList[ammountOfServers];
		/*lista me tous eksyphrethtes. Einai diatetagmenoi me f8inousa
				seira tou fasmatws dynamikhs pou eksyphretoun */

	int counter=0;

	int myElo,serverTopElo,serverBotElo;


	while(counter<ammountOfServers){

		serverTopElo= sortedByEloList[counter].askForTopElo ();

		serverBotElo= sortedByEloList[counter].askForBotElo ();

		if(myelo≥serverBotElo && myelo<serverTopElo)

		{

			break;

		}
		counter++;
	}/* auth h epanalhpsh einai bebaio oti 8a mas dwsei sto counter ton
		Server pros syndesh logw ths upo8eshs oti oloi oi xrhstes
		eksyphretountai apo enan server*/

result= sortedByEloList[counter].askForConnection();
```

```
	if(result==true)
	{
		sortedByEloList[counter].connect();
	}
	else{
		if(counter>0){
		result= sortedByEloList[counter-1].askForConnection();
		}
	}
	if(result==true)
	{
		sortedByEloList[counter-1].connect();
	}
	else{
		if(counter<amountOfServers-1){
		result= sortedByEloList[counter+1].askForConnection();
		}
	}
	if(result==true)
	{
		sortedByEloList[counter+1].connect();
	}
	else{
		sortedByEloList[counter].getInQueue();
	}
}
```

Η αντιμετώπιση της ουράς αναμονής

Όπως είπαμε στο δεύτερο μέρος αυτής της εργασίας, υπάρχουν δύο τρόποι αντιμετώπισης των χρηστών σε ουρά αναμονής. Λόγω του σχεδιασμού των αλγορίθμων σύνδεσης, είναι εύκολο να σχεδιάσουμε τους αλγορίθμους αντιμετώπισης της ουράς, καθώς κάθε εξυπηρετητής έχει δική του ουρά και δεν υπάρχει κάποια συνολική ουρά για όλους τους εξυπηρετητές, που θα μπορούσε να δημιουργήσει διενέξεις στην εξυπηρέτηση των χρηστών (για παράδειγμα, δύο εξυπηρετητές ταυτόχρονα μπορούν να εξυπηρετήσουν το ίδιο άτομο. Ποιος πρέπει να τον εξυπηρετήσει;).

Ο ένας αλγόριθμος αντιμετώπισης της ουράς είναι με μέθοδο FIFO, first in first out ή στην περίπτωσή μας first come first served. Η ιδέα της ουράς FIFO είναι ότι είναι ότι είναι δίκαιος αλγόριθμος. Ο πελάτης που βρίσκεται στην αρχή της ουράς θα είναι ο πρώτος που θα εξυπηρετηθεί. Φυσικά αυτό έρχεται με ένα κόστος, καθώς ο δεύτερος πελάτης, ακόμα και αν μπορεί να εξυπηρετηθεί από τον εξυπηρετητή, θα πρέπει να περιμένει να εξυπηρετηθεί πρώτα ο πρώτος πελάτης, που αναμένει να αδειάσει χώρος στον εξυπηρετητή. Ο αλγόριθμος του daemon που χειρίζεται τη λίστα, μπορεί να αναπαρασταθεί με τον εξής ψευδοκώδικα.

Αλγόριθμος εξυπηρέτησης ουράς FIFO

```
while(true){
	if(queue.isEmpty()){
		wait(10);
	}

	else{
		if(queue.get(0) can be served)
		{
			queue.get(0).connectToThisServer();
			queue.remove(0);
		}
		else{
			wait(10);
		}
	}
}
```

Από την άλλη έχουμε την δεύτερη εκδοχή για τον τρόπο με τον οποίο μπορεί να εξυπηρετείται η ουρά μας. Ο άπληστος αλγόριθμος που προτείναμε στο δεύτερο μέρος αυτής της εργασίας έχει ως σκοπό να προσπαθήσει να εξυπηρετήσει όσο το δυνατόν ταχύτερα μεγαλύτερο αριθμό χρηστών.

Ο εξυπηρετητής έχει έναν daemon ο οποίος συνεχώς κάνει προσπελάσεις της ουράς αναμονής, βρίσκει άτομα τα οποία μπορεί να εξυπηρετήσει και τα συνδέει. Αν υπάρχει ουρά λοιπόν, ένας χρήστης που θέλει να συνδεθεί μπαίνει στο τέλος της. Ο εξυπηρετητής εξετάζει με σειρά άφιξης τους χρήστες και επιλέγει τον πρώτο που μπορεί να εξυπηρετήσει και τον εξυπηρετεί.

Ο παραπάνω αλγόριθμος θα μπορούσε να αναπαρασταθεί με ψευδοκώδικα με τον εξής τρόπο:

Άπληστος αλγόριθμος εξυπηρέτησης ουράς.

```
while(true){
	if(queue.isEmpty()){
		wait(10);
	}

	else{
		for(i=0;i<queue.size();i++){
			if(queue.get(i) can be served){
				queue.get(i).connectToThisServer();
			queue.remove(i)
				break;
			}
		}
	}
}
```

Όπως μπορούμε να δούμε, ο παραπάνω αλγόριθμος, δεν εξετάζει μόνο τον πρώτο στη σειρά, με αυτόν τον τρόπο, αν ο εξυπηρετητής δεν μπορεί να εξυπηρετήσει τον χρήστη με σειρά x στην ουρά, αλλά μπορεί να εξυπηρετήσει κάποιον με σειρά y (x>y), θα το κάνει.

Αυτό δημιουργεί αδικία, καθώς κάποιος χρήστης που μόλις μπήκε στην ουρά μπορεί να εξυπηρετηθεί ενώ κάποιος που περιμένει πολύ ώρα να μην εξυπηρετηθεί ποτέ.

Άλλες προτάσεις εξυπηρέτησης ουράς

Στην παρούσα εργασία θα μελετήσουμε τους παραπάνω αλγορίθμους για την εξαγωγή συμπερασμάτων, θα προτείνουμε ωστόσο μερικούς ακόμα αλγορίθμους οι οποίοι είναι ενδιαφέροντες και καλό υλικό για μελλοντική εργασία.

Τροποποίηση του άπληστου αλγορίθμου

Μία μικρή τροποποίηση του αλγορίθμου που είδαμε στο προηγούμενο κεφάλαιο είναι ότι η ουρά θα μπορούσε να ταξινομείται κάθε φορά που συνδέεται κάποιος χρήστης εκ νέου με τον εξής τρόπο.

- Οι χρήστες με τις μικρότερες απαιτήσεις στη λίστα έρχονται στις πρώτες θέσεις τις λίστας.
- Χρήστες με ίδιο αριθμό απαιτήσεων, κατατάσσονται σύμφωνα με τον χρόνο σύνδεσής τους. Χρήστες οι οποίοι ήταν νωρίτερα στη σειρά έχουν προτεραιότητα σε σχέση με χρήστες οι οποίοι έχουν ίδιες απαιτήσεις και μεταγενέστερο χρόνο άφιξης.

Οι δύο αλγόριθμοι που είδαμε στο προηγούμενο κεφάλαιο δημιουργούν χρονικά θέματα εξυπηρέτησης. Ο αλγόριθμος FIFO δημιουργεί πρόβλημα στους χρήστες με μικρές απαιτήσεις, αν οι απαιτήσεις του χρήστη στην κεφαλή της ουράς είναι πολύ μεγάλες, σε σημείο τέτοιο που σταματάει η εξυπηρέτηση όλης της ουράς εξ αιτίας του.

Ο άπληστος αλγόριθμος που προτείναμε, δημιουργεί χρονική αδικία απέναντι στους χρήστες με μεγάλες απαιτήσεις, καθώς θα πρέπει να περιμένουν ουσιαστικά μέχρι να εξυπηρετηθούν όλοι οι χρήστες με μικρές απαιτήσεις από τον εξυπηρετητή.

Προτείνουμε λοιπόν την ύπαρξη και μίας δεύτερης ουράς που έχει μεγαλύτερη προτεραιότητα από την πρώτη ουρά. Σε αυτή την ουρά, ανάλογα με το μοντέλο εξυπηρέτησής μας, θα μπαίνουν οι χρήστες που αδικούνται έναντι των άλλων ύστερα από ένα χρονικό όριο το οποίο ορίζεται εξ αρχής ή μπορεί να είναι δυναμικό. Θα μπορούσε να είναι ο χρόνος ύστερα από τον οποίο έχει παρατηρηθεί ότι ένας χρήστης αποχωρεί απ την ουρά, ή θα μπορούσε να είναι ο μέσος χρόνος εξυπηρέτησης για παράδειγμα.

Στη συγκεκριμένη εργασία δεν θα εξετάσουμε αυτές τις εκδοχές αλλά είναι καλό αντικείμενο μελέτης για μελλοντική έρευνα.

V. Η διεξαγωγή παιχνιδιών

Τι είναι ο μηχανισμός εύρεσης αγώνα και συμπαικτών

Φυσικά ένας χρήστης που συνδέεται στον εξυπηρετητή δεν έχει ως μοναδικό σκοπό να παραμείνει συνδεδεμένος και να μην κάνει τίποτα για μερική ώρα. Ο λόγος που συνδέονται οι χρήστες στους εξυπηρετητές μας στην προκειμένη περίπτωση είναι για να παίξουν παιχνίδια.

Ένας χρήστης που θέλει να παίξει έναν αγώνα από το παιχνίδι μας, είτε προσκαλεί φίλους του για να σχηματίσει ομάδα, είτε μπορεί να ζητήσει να του βρει ο εξυπηρετητής συμπαίκτες. Όποια και αν είναι η απόφαση του παίκτη, ο εξυπηρετητής φροντίζει, πέρα από το να βρει συμπαίκτες για τον παίκτη μας αν το επιθυμεί, να βρει μία αντίπαλη ομάδα για να διεξαχθεί ο αγώνας.

Είναι πολύ μεγάλης σημασίας οι ομάδες που συνδυάζονται για να διεξαχθεί ένα παιχνίδι να είναι κοντινής δυναμικής ικανότητας. Φυσικά, μέχρι στιγμής γνωρίζουμε ότι βαθμολογία ικανότητας έχουν μόνο οι παίχτες. Όταν γίνεται αναζήτηση για συμπαίκτες λοιπόν, θα ήταν λογικό να επιθυμούμε να βρούμε συμπαίκτες με βαθμολογία ικανότητας κοντινή σε αυτή του χρήστη που αναζητεί. Ύστερα από την συμπλήρωση μίας ομάδας, αυτή η ομάδα έχει έναν μέσο όρο βαθμολογίας ικανότητας, αν αθροίσουμε τις βαθμολογίες ικανότητας των παικτών και διαιρέσουμε το άθροισμα με τον αριθμό των παικτών. Είναι δυνατόν λοιπόν να εξάγουμε έναν αριθμό βαθμολογίας ικανότητας για την κάθε ομάδα.

Παρομοίως συνεπώς, θα πρέπει να συνδυάσουμε ομάδες οι οποίες να έχουν κοντινές βαθμολογίες ικανότητας. Αν δεν το κάνουμε αυτό, είναι πιθανό ομάδες πολύ ικανές να παίξουν με αρχάριους παίκτες και το παιχνίδι να μην είναι διασκεδαστικό, καθώς για τους μεν έμπειρους παίκτες, δε θα υπάρχει πρόκληση ενώ για τους αδύναμους παίκτες, δεν θα είναι διασκεδαστικό.

Τα παραπάνω μας οδηγούν στην απαίτηση ύπαρξης κάποιου μηχανισμού που θα δημιουργεί αγώνες, ή όπως ονομάζεται στα αγγλικά matchmaking mechanism. Ο μηχανισμός αυτός αναζητεί συμπαίκτες για τους χρήστες και συνδυάζει ομάδες που θα αγωνιστούν αντίπαλες μεταξύ τους. Για να γίνει δυνατή η δημιουργία ενός τέτοιου μηχανισμού θα πρέπει να υπάρχουν κάποιες λίστες με χρήστες που αναζητούν συμπαίκτες για να σχηματίσουν ομάδα, ομάδες που αναζητούν αντίπαλες ομάδες και μία λίστα με τα ενεργά παιχνίδια.

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο, οι παίκτες, πέρα από την ύπαρξη δικαιοσύνης στους αγώνες τους, επιθυμούν να μην περιμένουν πολύ ώρα για να παίξουν ένα παιχνίδι, οπότε θα πρέπει αυτός ο μηχανισμός να είναι και γρήγορος.

Πώς δουλεύει ο μηχανισμός εύρεσης συμπαικτών και αγώνα

Αρχικά μπορούμε να χωρίσουμε τη λειτουργία του μηχανισμού σε δύο στάδια. Το πρώτο στάδιο είναι η εύρεση συμπαικτών για έναν χρήστη ή μια ομάδα χρηστών. Το δεύτερο στάδιο είναι η εύρεση μίας ομάδας του ίδιου επιπέδου για την διεξαγωγή ενός αγώνα. Συνεπώς θα μελετήσουμε το κάθε στάδιο ξεχωριστά, καθώς είναι δύο τελείως ξεχωριστές ενέργειες που δεν εμπλέκονται η μία με την άλλη άμεσα.

Το στάδιο εύρεσης συμπαικτών

Όπως είπαμε προηγουμένως ένας παίκτης έχει την δυνατότητα να σχηματίσει μία ομάδα για παιχνίδι με φίλους του, με προσωπικές προσκλήσεις. Συνήθως στα παιχνίδια MOBA οι ομάδες είναι των 5 παικτών, οπότε θα θεωρήσουμε για χάρη παραδείγματος ότι και σε αυτή την περίπτωση ο μέγιστος αριθμός σε μία ομάδα είναι 5 άτομα.

Ένας παίκτης, λοιπόν, μπορεί να προσκαλέσει έως 4 φίλους του για να συμπληρώσει μία ομάδα και ύστερα να αναζητήσει αντίπαλο. Ωστόσο, ένας παίκτης δεν είναι υποχρεωμένος να σχηματίσει μόνος του ομάδα με 5 άτομα. Μπορεί να σχηματίσει μία ομάδα με λιγότερα άτομα ή ακόμα και να ζητήσει να παίξει έναν αγώνα μόνος του χωρίς να έχει προσκαλέσει κάποιον συμπαίκτη. Σε αυτή τη περίπτωση ο μηχανισμός εύρεσης συμπαικτών πρέπει να συμπληρώσει την ομάδα με άτομα έως ότου η ομάδα να έχει 5 άτομα.

Πρέπει συνεπώς να υπάρξει μία μοντελοποίηση του προβλήματος εύρεσης συμπαικτών. Μπορούμε να θεωρήσουμε ότι μία ομάδα που αναζητεί παίκτες μπορεί να έχει από έναν έως 4 παίκτες. Αν έχει πέντε παίκτες, τότε έχει συμπληρώσει τον μέγιστο αριθμό ατόμων και πλέον περνάει στην δεύτερη φάση του μηχανισμού, την αναζήτηση αντίπαλης ομάδας.

Η αναζήτηση παικτών πρέπει να ικανοποιεί δύο απαιτήσεις, οι συμπαίκτες που συνδυάζονται μεταξύ τους να μην έχουν μεγάλη διαφορά βαθμολογίας δυναμικού και να μην διαρκεί πολύ ώρα η αναζήτηση. Για τον λόγο αυτό χρειαζόμαστε έναν αλγόριθμο ο οποίος να είναι δυναμικός με τον χρόνο και όσο πιο πολύς χρόνος περνάει, τόσο περισσότερο ελαστικός να γίνεται στην απαίτηση οι συμπαίκτες να έχουν κοντινές βαθμολογίες ικανότητας.

Φυσικά, η ελαστικότητα είναι θέμα υποκειμενικό και θα μπορούσε να την ορίζει ή να την περιορίζει ο κατασκευαστής του παιχνιδιού ή να την ορίζει ο χρήστης ως έναν βαθμό. Για παράδειγμα ένας χρήστης που δεν τον ενδιαφέρει ο χρόνος αναμονής, θα μπορούσε να επιλέξει να μην υπάρχει ελαστικότητα στην επιλογή παικτών και να συνδυαστεί μόνο με παίκτες με μικρή απόκλιση βαθμολογίας ικανότητας. Από την άλλη, ο κατασκευαστής του παιχνιδιού μπορεί να θέλει να έχει πολύ μικρές ελαστικότητες, γιατί μπορεί να έχει τους χρήστες συνδεδεμένους σε εξυπηρετητές σύμφωνα με την βαθμολογία ικανότητάς τους και να μην επιθυμεί να γίνονται συνδέσεις με πολλούς άλλους εξυπηρετητές. Στη πραγματικότητα η ελαστικότητα είναι ένας καθαρός αριθμός ο οποίος δείχνει πόσους βαθμούς ικανότητας διαφορά επιτρέπεται να έχει μία ομάδα από μία άλλη για να συνδυαστούν.

Αφού οριστεί ο βαθμός ικανότητας μίας μη συμπληρωμένης ομάδας, που δηλαδή έχει από έναν έως τέσσερεις παίκτες, αυτή μπαίνει στην ουρά εύρεσης συμπαικτών. Ανάλογα με το πόση ώρα βρίσκεται στην ουρά η συγκεκριμένη ομάδα, ορίζεται και η ελαστικότητά της. Το εάν ο χρόνος μετράται από την στιγμή που έγινε η τελευταία προσθήκη χρήστη ή από την στιγμή που περιμένει ο χρήστης με τον μεγαλύτερο χρόνο αναμονής ή αν βγαίνει κάποιος μέσος όρος αναμονής όλων των χρηστών της ομάδας ή ακόμα και εάν υπάρχει ελαστικότητα, εξαρτάται από τον κατασκευαστή. Εμείς στη συγκεκριμένη εργασία, θα θεωρήσουμε ότι κάθε φορά που προστίθενται καινούρια μέλη σε μία ομάδα, ο χρόνος αναμονής μηδενίζεται.

Ο μηχανισμός ελέγχει μία μία τις ομάδες και αναζητεί άλλες ομάδες οι οποίες να μπορούν να προστεθούν στην ήδη υπάρχουσα. Αυτό εξαρτάται από τον αριθμό παικτών που υπάρχουν στις δύο ομάδες, καθώς πρέπει ο συνολικός αριθμός χρηστών σε μία ομάδα να είναι κάτω από ή ακριβώς πέντε, από την βαθμολογία της κάθε ομάδας και από την ελαστικότητα της κάθε ομάδας. Αν οι δύο ομάδες έχουν αθροιστικό αριθμό ατόμων ίσο με πέντε, ή λιγότερο από αυτό, και η απόλυτη τιμή του αποτελέσματος της διαφοράς των βαθμολογιών των δύο ομάδων είναι μικρότερη από την ελαστικότητα και των δύο ομάδων, τότε στην πρώτη ομάδα προστίθενται τα άτομα της δεύτερης και η δεύτερη βγαίνει από την ουρά αναμονής. Επαναϋπολογίζεται η βαθμολογία δυναμικότητας της ομάδας και εάν έχει πέντε ακριβώς άτομα, τότε αφαιρείται από την λίστα αναζήτησης συμπαικτών και προστίθεται στην λίστα αναζήτησης αντίπαλης ομάδας. Σε αντίθετη περίπτωση περιμένει στην λίστα εύρεσης συμπαικτών έως ότου συμπληρωθούν και οι πέντε θέσεις.

Το στάδιο εύρεσης αντίπαλης ομάδας

Σε αυτό το στάδιο, ομάδες που έχουν ήδη συμπληρώσει τον απαιτούμενο αριθμό παικτών, στην περίπτωσή μας πέντε παίκτες, μπαίνουν σε μία λίστα διαφορετική από την προηγούμενη, στην οποία βρίσκονται όλες οι ομάδες που έχουν τον απαιτούμενο αριθμό παικτών.

Όπως και πριν, έτσι και τώρα, η κάθε ομάδα έχει μία βαθμολογία ικανότητας, έναν χρόνο αναμονής στη λίστα και μία ελαστικότητα. Ο μηχανισμός εξετάζει μία μία τις ομάδες και σε περίπτωση που δύο ομάδες ταιριάζουν για να παίξουν ένα παιχνίδι, τότε τις συνδυάζει και τις αφαιρεί από τη λίστα εύρεσης αντίπαλης ομάδας, δημιουργείται ένα παιχνίδι στο οποίο συμμετέχουν οι 2 ομάδες που συνδυάστηκαν και το παιχνίδι αυτό προστίθεται σε μία λίστα ενεργών παιχνιδιών.

Στην επιλογή του ποιες ομάδες ταιριάζουν μεταξύ τους ακολουθείται περίπου η ίδια διαδικασία με πριν. Αν η απόλυτη τιμή του αποτελέσματος της διαφοράς των βαθμολογιών ικανότητας των δύο ομάδων είναι μικρότερη από την τιμή της ελαστικότητας και των δύο ομάδων, τότε αυτές οι δύο ομάδες ταιριάζουν και μπορούν να συνδυαστούν για να διεξαχθεί ένας αγώνας μεταξύ τους.

Παρατηρήσεις για τον μηχανισμό

Όπως είναι προφανές, ο μηχανισμός αυτός πρέπει να λειτουργεί σε ένα σύνολο εξυπηρετητών, δεδομένου ότι ένα region αποτελείται από πολλούς εξυπηρετητές και πρέπει να υπάρχει η δυνατότητα να διεξαχθούν αγώνες μεταξύ κάθε χρήστη. Από την άλλη, δεν διεξάγονται αγώνες μεταξύ regions ούτε υπάρχει δυνατότητα ένας χρήστης να επικοινωνήσει με έναν άλλον χρήστη που ανήκει σε διαφορετικό region. Έτσι λοιπόν ο μηχανισμός πρέπει να λαμβάνει πληροφορίες για τους παίκτες που επιθυμούν να συμμετάσχουν σε έναν αγώνα από όλους τους εξυπηρετητές. Μπορεί να πει κάποιος ότι ο μηχανισμός ανήκει στο region και ότι οι εξυπηρετητές υπάρχουν μόνο για να μεταφέρουν δεδομένα μεταξύ των παικτών, χωρίς να εμπλέκονται καθόλου στην επιλογή των χρηστών.

Ωστόσο, ο μηχανισμός έχει κάποια ελαττώματα. Δεν λαμβάνει υπ' όψιν του, σε ποιόν εξυπηρετητή βρίσκεται ο κάθε παίκτης, ούτε τι απόσταση υπάρχει μεταξύ των παικτών, σύμφωνα με το σε ποιόν εξυπηρετητή βρίσκονται. Με αυτόν τον τρόπο, μπορεί να συνδυαστούν σε ένα παιχνίδι παίκτες για τους οποίους ο χρόνος επικοινωνίας να είναι μεγαλύτερος από τον προκαθορισμένο χρόνο συγχρονισμού του παιχνιδιού, σε τέτοιο βαθμό που το παιχνίδι να είναι αδύνατο να παιχτεί. Σε μία τέτοια περίπτωση υπάρχουν μία σειρά από μέτρα που μπορούμε να λάβουμε.

Η μία επιλογή μας είναι να συμπεριλάβουμε κατά την δράση του μηχανισμού τον έλεγχο για την απόσταση του κάθε παίκτη από τον άλλον και να μην επιτρέπεται να συνδυαστούν ομάδες στις οποίες η μέγιστη απόσταση μεταξύ των παικτών να ξεπερνάει ένα όριο.

Μία δεύτερη επιλογή είναι να μην ασχοληθούμε με την απόσταση μεταξύ των παικτών κατά την διάρκεια της εύρεσης συμπαικτών και αντίπαλης ομάδας. Αφού δημιουργηθεί ένα παιχνίδι, πριν διεξαχθεί, εξετάζονται οι αποστάσεις των παικτών. Αν υπάρχει πρόβλημα με τις αποστάσεις μεταξύ των παικτών τότε ακολουθείται μία διαδικασία μετανάστευσης χρηστών που θα αναλύσουμε στο επόμενο κεφάλαιο.

Τι είναι η μετανάστευση χρηστών μεταξύ εξυπηρετητών

Υπάρχουν φορές που χρειάζεται να υπάρξει μετακίνηση χρηστών μεταξύ των εξυπηρετητών. Η μετακίνηση αυτή ονομάζεται μετανάστευση. Είτε αυτό είναι επειδή κάποιος εξυπηρετητής πρέπει να σταματήσει να λειτουργεί για συντήρηση, ή γνωρίζει ότι είναι απαραίτητο να σταματήσει να λειτουργεί λόγω τεχνικού προβλήματος είτε επειδή κάποιος χρήστης πρέπει να μετακινηθεί λόγω των περιορισμών που περιγράψαμε στο προηγούμενο κεφάλαιο, η μετανάστευση είναι κάτι αναπόφευκτο και χρονοβόρο. [1]

Υπάρχουν διάφοροι μηχανισμοί για την πραγματοποίηση της μετανάστευσης, αλλά έχει μεγαλύτερο ενδιαφέρον το πώς γίνεται η επιλογή του εξυπηρετητή προς τον οποίο θα γίνει η μετανάστευση και ο τρόπος αντίδρασης σε μία υποχρεωτική μετανάστευση.

Στην εργασία αυτή, δεν θα εξετάσουμε τις περιπτώσεις μετανάστευσης λόγω κακής λειτουργίας των εξυπηρετητών, καθώς δεν είναι μέρος του αντικειμένου της μελέτης μας. Θα εξετάσουμε ωστόσο την περίπτωση μετανάστευσης λόγω των περιορισμών που τέθηκαν στο προηγούμενο κεφάλαιο. Συγκεκριμένα, στην περίπτωση που υπάρχει ένας αγώνας προς διεξαγωγή και υπάρχουν αποστάσεις μεταξύ των χρηστών τέτοιες ώστε να εμποδίζουν την ομαλή διεξαγωγή του παιχνιδιού.

Για να επιλέξουμε τον εξυπηρετητή που θα δεχθεί τους χρήστες της μετανάστευσης υπάρχουν κάποιες παράμετροι που πρέπει να λάβουμε υπ' όψιν μας:

- Λύνει ο εξυπηρετητής αυτός το πρόβλημά μας;
- Μπορεί να εξυπηρετήσει τους χρήστες;
- Αν δεν μπορεί να εξυπηρετήσει τους χρήστες στην τρέχουσα κατάστασή του, μπορεί να μεταναστεύσει άλλους χρήστες που δεν

βρίσκονται ήδη σε παιχνίδι ώστε να μπορεί να εξυπηρετήσει τους χρήστες μας;

Αν εν τέλει υπάρχει κάποιος εξυπηρετητής στον οποίο μπορούμε να μεταφέρουμε τους χρήστες προς μετανάστευση, τότε μεταφέρουμε σε αυτόν τους χρήστες μας. Εάν δεν είναι δυνατή η μετανάστευση, τότε οι χρήστες θα παραμείνουν στον αρχικό εξυπηρετητή τους.

Έχοντας πλέον τα βήματα για την πραγματοποίηση της μετανάστευσης, είναι ευκαιρία να μιλήσουμε για τον τρόπο επιλογής του εξυπηρετητή. Υπάρχουν πολλές απόψεις στο γι αυτό το θέμα. Στη συγκεκριμένη εργασία θα εφαρμόσουμε τα εξής:

- Έστω ότι το δίκτυο που ενώνει τους εξυπηρετητές είναι ένας γράφος G με κόμβους N τους εξυπηρετητές και ακμές E τις συνδέσεις μεταξύ των εξυπηρετητών
- Κάθε ακμή E(N1,N2) έχει μία απόσταση L ίση με τη χρονική καθυστέρηση για την επικοινωνία μεταξύ των κόμβων N1 και N2
- Δημιούργησε μία λίστα F και βάλε σε αυτή με αύξουσα σειρά εκκεντρικότητας τους κόμβους

Πλέον έχουμε μία λίστα στην οποία έχουμε διατεταγμένους τους κόμβους με σειρά εκκεντρικότητας. Η εκκεντρικότητα ενός κόμβου είναι η μεγαλύτερη απόσταση που έχει ο κόμβος από τους υπολοίπους, ή αλλιώς, η απόσταση του κόμβου προς τον πιο μακρινό απ' αυτόν κόμβο. Ο κόμβος με την μικρότερη εκκεντρικότητα είναι το κέντρο του γράφου. Η εύρεση του κέντρου του γράφου είναι μία δύσκολη διαδικασία σε μεγάλους γράφους, όμως στην περίπτωσή μας, στην χειρότερη περίπτωση θα έχουμε 10 κόμβους, οπότε δεν θα είναι δύσκολο να βρούμε το κέντρο του.

Από την λίστα F θα πάρουμε με σειρά τους εξυπηρετητές και θα επιχειρήσουμε μετανάστευση προς αυτούς.

Στη περίπτωση που μετά από προσπάθειες επίλυσης της χρονικής διένεξης δεν έχει δοθεί λύση τότε το παιχνίδι πρέπει να διεξαχθεί με χρονική καθυστέρηση μεταξύ των παικτών. Αυτό θα δημιουργήσει πολλά προβλήματα και είναι προτιμότερο να σταματήσει η διεξαγωγή του παιχνιδιού παρά να διεξαχθεί ένα κατώτερης ποιότητας παιχνίδι.

Ένα παράδειγμα μετανάστευσης

Έστω λοιπόν ότι έχουμε ένα παιχνίδι που διεξάγεται μεταξύ 10 χρηστών χωρισμένοι σε ομάδες των 5 ατόμων. Τυχαίνει να υπάρχει διένεξη μεταξύ 5 από αυτών των χρηστών, από διαφορετικές ομάδες. Συγκεκριμένα τρείς από τους χρήστες βρίσκονται στον εξυπηρετητή X, δύο στον

εξυπηρετητή Y και ένας στον εξυπηρετητή Z ενώ οι υπόλοιποι βρίσκονται στον εξυπηρετητή L. Το όριο χρονικής καθυστέρησης μεταξύ των εξυπηρετητών είναι Δ. Οι χρονικές αποστάσεις μεταξύ των εξυπηρετητών είναι οι ακόλουθες:

FROM/TO	L	X	Y	Z
L	0	10	12	9
X	10	0	22	19
Y	12	22	0	21
Z	9	19	21	0

Πίνακας 1 - Χρόνος σε ms επικοινωνίας μεταξύ των εξυπηρετητών L,X,Y,Z

- Αν το Δ είναι μεγαλύτερο από 22ms τότε δεν χρειάζεται να γίνει κάποια μετανάστευση.
- Αν το Δ είναι 21 τότε υπάρχει πρόβλημα στην επικοινωνία μεταξύ των εξυπηρετητών Y και X καθώς έχουν χρόνο επικοινωνίας 22 και το όριο είναι 21.

  Στη συγκεκριμένη περίπτωση πρέπει οι χρήστες από τον εξυπηρετητή X ή από τον εξυπηρετητή Y να μετακινηθούν. Καθώς η μετανάστευση είναι μία χρονοβόρα διαδικασία, και συνεπώς κοστίζει, είναι επιθυμητό να γίνονται όσο το δυνατόν λιγότερες μεταναστεύσεις χρηστών. Έτσι λοιπόν, θα επιλέξουμε να μετακινήσουμε τους χρήστες από τον εξυπηρετητή που έχει τους λιγότερους χρήστες που συμβάλλουν στην διένεξη. Εδώ, στον εξυπηρετητή X έχουμε τρείς χρήστες ενώ στον εξυπηρετητή Y έχουμε δύο χρήστες. Θα προτιμήσουμε να μετακινήσουμε τους χρήστες από τον εξυπηρετητή Y λοιπόν. Η λίστα F που περιγράψαμε παραπάνω θα περιέχει με αυτή τη σειρά τα εξής στοιχεία F={L,Z,X,Y}. Θα επιχειρήσουμε συνεπώς μετανάστευση πρώτα προς τον L μετά προς τον Z και τέλος προς τον X. Αν δεν είναι επιτυχής η μετανάστευση, τότε σταματάμε την προσπάθεια και διεξάγουμε το παιχνίδι.
- Αν το Δ είναι 20, τότε πρέπει να μεταναστεύσουν και οι χρήστες από τον X και από τον Y, ακολουθούμε την ίδια διαδικασία όπως παραπάνω. Η σειρά με την οποία θα γίνει η μετανάστευση των χρηστών δεν έχει ιδιαίτερη σημασία.
- Παρομοίως για τις διάφορες τιμές του Δ αναγκάζονται να μεταναστεύσουν όλο και περισσότεροι ή λιγότεροι χρήστες.

Το κόστος της μετανάστευσης

Όπως αναφέραμε προηγουμένως, το κόστος της μετανάστευσης των χρηστών είναι αρκετά μεγάλο, ειδικά στην περίπτωση που η μετανάστευση ενός χρήστη δημιουργεί αλυσιδωτή αντίδραση μετανάστευσης περισσότερων

χρηστών [1]. Η μετανάστευση περιλαμβάνει την επικοινωνία του εξυπηρετητή από τον οποίο αποστέλλεται ο χρήστης με τους υπόλοιπους μέχρι την εύρεση ενός εξυπηρετητή που να μπορεί να εξυπηρετήσει τον καινούριο χρήστη και την μετακίνηση των δεδομένων του χρήστη. Όπως είναι προφανές, δεν είναι καλό να γίνονται πολλές μεταναστεύσεις καθώς τα δίκτυα επικοινωνίας των εξυπηρετητών θα επιβαρύνονται πέρα από τα δεδομένα των αγώνων που παίζονται εκείνη τη στιγμή και με τα δεδομένα των μεταναστεύσεων.

VI. Προσομοίωση για εξαγωγή δεδομένων

Δεδομένα προς συλλογή

Στην έρευνά μας θέλουμε να δούμε ποιος συνδυασμός των αλγορίθμων που ήδη περιγράψαμε είναι ο πιο αποδοτικός και υπό ποιες συνθήκες. Η απόδοση των αλγορίθμων θα κριθεί από τον αριθμό αναμονής των χρηστών σε λίστες αναμονής και τον αριθμό των παιχνιδιών που χρειάστηκαν να γίνουν μεταναστεύσεις για να διεξαχθούν.

Τρόπος διεξαγωγής της προσομοίωσης

Για να εξάγουμε τα δεδομένα που θέλουμε σχεδιάσαμε μία προσομοίωση και την προγραμματίσαμε σε Java. Η προσομοίωση έτρεξε τοπικά, σε έναν υπολογιστή, χωρίς την μεσολάβηση βάσεων δεδομένων.

Δυστυχώς δεν ήταν δυνατό να διεξάγουμε την προσομοίωση σε πλήρη κλίμακα, δηλαδή να έχουμε 714.285 χρήστες ταυτόχρονα συνδεδεμένους, καθώς υπήρχαν προβλήματα. Συγκεκριμένα, και η υπολογιστικής ισχύς όπως και η προσωρινή μνήμη RAM του υπολογιστή δεν επαρκούσαν για να υπάρξει διεξαγωγή πλήρους κλίμακας. Κάθε χρήστης που δημιουργούσε το πρόγραμμά μας απαιτούσε περίπου 97 KB μνήμης, το οποίο σημαίνει ότι για να τρέξουμε το πρόγραμμα σε πλήρη κλίμακα θα χρειαζόμασταν 66 Gigabyte Ram, πέρα από την τεράστια υπολογιστική ισχύ.

Λόγω αυτών των συνθηκών, αναγκαστήκαμε να εκτελέσουμε την προσομοίωση υπό κλίμακα 1/50. Αυτό πρακτικά σημαίνει ότι όλα τα δεδομένα μας θα έπρεπε να τα διαιρέσουμε με το 50. Σίγουρα το πρόβλημα που μελετάμε δεν είναι γραμμικό, αλλά θα μπορέσουμε να δούμε κάποια κυρίαρχα μοτίβα.

Σχεδιασμός του προγράμματος της προσομοίωσης σε Java

Για να δημιουργήσουμε μία προσομοίωση που να μπορεί να δείξει τον τρόπο με τον οποίο οι χρήστες συνδέονται με ένα MOBA παιχνίδι εξετάσαμε τη λειτουργία ενός region. Μοντελοποιήσαμε την ύπαρξή του στις εξής βασικές κλάσεις:

- Χρήστης
- Εξυπηρετητής
- Ομάδα
- Παιχνίδι
- Σύνδεση

Επιπλέον αυτών των βασικών μονάδων, δημιουργήσαμε κάποιες κλάσεις ελέγχου:

- Καθολικό ρολόι
- Δημιουργός Χρηστών
- Ελεγκτής Μηχανισμών

Πριν προχωρήσουμε στην ανάλυση των κλάσεων και την προβολή κάποιων σημαντικών μεθόδων τους, είναι σημαντικό να αναφέρουμε κάποια πράγματα για τον τρόπο παραγωγής ψευδοτυχαίων αριθμών. Όπου χρησιμοποιούμε ψευδοτυχαίους αριθμούς, στη πραγματικότητα χρησιμοποιούμε ένα στιγμιότυπο της κλάσης Java.util.random και παράγουμε αριθμούς με την χρήση των συναρτήσεων που παρέχει.

Κλάση χρήστη

Είναι η κλάση που περιγράφει έναν χρήστη του συστήματός μας. Τον περιγράφει ένα μοναδικό όνομα χρήστη, ένας ψευδοτυχαίος δεκαδικός αριθμός στο διάστημα (0,1] που περιγράφει τις απαιτήσεις του από τον εξυπηρετητή καθώς και ο αριθμός νικών και ηττών του.

Ο αριθμός νικών και ηττών του παράγεται ψευδοτυχαίως με την χρήση των ενσωματωμένων κλάσεων της Java για παραγωγή ψευδοτυχαίων αριθμών. Θα τους χρησιμοποιήσουμε για να παράξουμε την βαθμολογία ικανότητας του χρήστη. Στη πραγματικότητα, ο αλγόριθμος που θα χρησιμοποιήσουμε για την παραγωγή της βαθμολογίας ικανότητας δεν είναι κάποιος από αυτούς που χρησιμοποιούν τα παιχνίδια MOBA καθώς αυτοί είναι κρυφοί. Άλλωστε, δεν μας απασχολεί ιδιαιτέρως να έχουμε έναν αλγόριθμο ο οποίος να δίνει πραγματική βαθμολογία, αρκεί να μας βγάζει μία βαθμολογία για να μπορούμε να την αξιοποιήσουμε όπου χρειάζεται και να είναι αληθοφανής. Η συνάρτηση που θα χρησιμοποιήσουμε λοιπόν είναι η εξής:

```
public void calculateElo() {
   int elo = 800;


        if (loses != 0) {
            if (loses > wins) {
                elo += ((loses - wins) * 1);
            } else {
                elo += ((wins - loses) * 3);
            }
        } else {
            elo = 800;
        }
        myElo = elo;
    }
```

Πρακτικά δίνει σε κάθε χρήστη 800 βαθμούς βαθμολογίας ως κατώτατο όριο βαθμολογίας και επ' αυτού προσθέτει πόντους ανάλογα με τον αριθμό νικών ή ηττών του. Ο λόγος που επιλέξαμε ως ελάχιστο όριο τους 800 βαθμούς είναι επειδή οι πιο πολλοί μηχανισμοί βαθμολόγησης ξεκινούν με βάση το 800, το οποίο είναι ένας αυθαίρετος αριθμός. Στην περίπτωση της εργασίας μας, όποια και αν ήταν η βάση της βαθμολογίας, δεν θα επηρέαζε τα αποτελέσματα.

Άλλη συνάρτηση ενδιαφέροντος για τους χρήστες είναι η συνάρτηση που περιγράφει την συμπεριφορά του. Φτιάξαμε λοιπόν μία μικρή συνάρτηση τεχνητής νοημοσύνης.

```
public void myAI() {
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                Random rand = new Random();
                while (new GregorianCalendar().getTimeInMillis() -
connection <= millisToBeOnline) {
```

```
                int what = rand.nextInt(10);
                if (what < 7) {
                    idle();
                } else {
                    playGame();
                }
            }
            disconnect();
        }
    });
    t.start();
}
```

Αυτό που κάνει αυτή η συνάρτηση είναι ότι δημιουργεί ένα νήμα για τον τρέχοντα χρήστη ο οποίος όταν είναι συνδεδεμένος στην υπηρεσία, είτε θα παραμένει ανενεργός είτε θα θελήσει να παίξει ένα παιχνίδι. ύστερα από το πέρας κάποιου χρόνου αυτός θα αποσυνδέεται. Υπάρχει περίπτωση αυτός ο χρήστης να μην παίξει ποτέ κάποιον αγώνα όπως είναι προφανές, αλλά αυτό δε μας ενοχλεί, καθώς δεν είναι σπάνιο το φαινόμενο των χρηστών που χρησιμοποιούν την εφαρμογή για ένα μικρό χρονικό διάστημα σε μία ημέρα για να επικοινωνήσουν με φίλους τους.

Η κλάση του εξυπηρετητή

Ο εξυπηρετητής είναι μία κλάση της οποίας τα στιγμιότυπα έχουν ένα μοναδικό όνομα, μία λίστα με συνδέσεις προς άλλους εξυπηρετητές, έναν αριθμό Shares τα οποία μοιράζονται οι χρήστες, λίστες για την επεξεργασία των χρηστών και αν το απαιτεί ο αλγόριθμος, ένα άνω και ένα κάτω όριο βαθμολογίας ικανότητας.

Η κλάση Δημιουργός Χρηστών

Στην προσομοίωσή μας δημιουργούμε ένα αντικείμενο βασισμένο σε αυτή τη κλάση το οποίο έχει ως σκοπό την δημιουργία χρηστών. Χρησιμοποιούμε τον τρόπο που περιγράψαμε στο κεφάλαιο 3 για το πόσοι χρήστες συνδέονται. Ο έλεγχος είναι ανά ms και δημιουργεί ανά ms όσους χρήστες είναι απαραίτητοι για την εκπλήρωση των απαιτήσεων της προσομοίωσής μας.

Οι υπόλοιπες κλάσεις δεν είναι κάποιου ιδιαίτερου ενδιαφέροντος και είναι αρκετά αυτονόητο το τι κάνει η κάθε μία. Επίσης, πέρα από τις προαναφερθείσες κλάσεις, δημιουργήσαμε κάποιες επιπλέον κλάσεις οι οποίες ήταν βοηθητικές. Είχαν σαν σκοπό την εκτύπωση μηνυμάτων και την εκτύπωση αποτελεσμάτων/δεδομένων που θα χρησιμοποιήσουμε για την εξαγωγή συμπερασμάτων

Τρόπος συλλογής δεδομένων

Για κάθε αλγόριθμο έγιναν 10 εκτελέσεις της προσομοίωσης και τα δεδομένα που έχουμε είναι το αποτέλεσμα του μέσου όρου των 10 αυτών εκτελέσεων. Όπως αναφέραμε προηγουμένως, η προσομοίωσή μας θα είναι σε κλίμακα 1/50 λόγω των μεγάλων υπολογιστικών απαιτήσεων που έχει η 1/1 εκτέλεση της προσομοίωσης. Αυτό πρακτικά σημαίνει ότι:

- Ο μέγιστος αριθμός ταυτόχρονων χρηστών σε ένα region είναι 714.285/20= 14286 χρήστες.
- Κατά μέσο όρο συνδέονται 1.714.285/50=34286 χρήστες ημερησίως.
- Συνολικά σε μία ημέρα παίζονται 1.500.000 εκατομμύρια ώρες σε κάθε region. Αυτό αναλογεί σε περίπου 52 λεπτά και 30 δευτερόλεπτα παιχνιδιού για κάθε χρήστη, ή στην 1/50 μορφή, 63 δευτερόλεπτα.

Στην πραγματικότητα, ο χρόνος δεν μας αφορά να είναι σε κλίμακα, ούτε να είναι στην ίδια κλίμακα με αυτή των αριθμών χρηστών. Ο χρόνος είναι ανεξάρτητη μεταβλητή και μπορούμε να χρησιμοποιήσουμε όποια κλίμακα θέλουμε. Παρ' όλα αυτά, θα χρησιμοποιήσουμε κλίμακα 1/50 και για τον χρόνο για να μην υπάρξουν απρόβλεπτες διαφοροποιήσεις και διενέξεις.

Συνδεσμολογία Εξυπηρετητών

Για την συνδεσμολογία των εξυπηρετητών χρησιμοποιήσαμε 10 εξυπηρετητές συνδεδεμένους μεταξύ τους με τυχαίες αληθοφανείς καθυστερήσεις.

Ως τυχαίες αληθοφανείς καθυστερήσεις εννοούμε έναν ακέραιο αριθμό στο σύνολο [10,30] που παράγεται ψευδοτυχαίως. Αυτός ο αριθμός δείχνει τον χρόνο σε ms που χρειάζεται η πληροφορία για να ταξιδέψει μεταξύ των εξυπηρετητών που είναι συνδεδεμένοι με τη συγκεκριμένη σύνδεση.

Στους αλγορίθμους με διαχωρισμό των χρηστών σύμφωνα με την βαθμολογία ικανότητάς τους, οι εξυπηρετητές ήταν συνδεδεμένοι άμεσα με τους διπλανούς τους μόνο, ενώ σε περίπτωση που έπρεπε να επικοινωνήσουν με εξυπηρετητή μακρινότερο, θεωρούσαμε ότι οι ενδιάμεσοι εξυπηρετητές έκαναν μεταφορά (relay) της πληροφορίας.

Στους υπόλοιπους αλγορίθμους δημιουργείτο μία τυχαία συνδεσμολογία μεταξύ των εξυπηρετητών με μέγιστο αριθμό συνδέσεων για κάθε εξυπηρετητή 2, μέγιστο αριθμό αθροιστικών συνδέσεων 10, και την απαίτηση κάθε εξυπηρετητής να μπορεί να επικοινωνήσει με κάθε άλλο εξυπηρετητή.

Τα δεδομένα από την προσομοίωση

Όπως αναφέραμε εκτελέσαμε δέκα επαναλήψεις για κάθε συνδυασμό αλγορίθμων σύνδεσης και εκμετάλλευσης ουράς. Ο παρακάτω πίνακας μας δείχνει τα δεδομένα που συλλέξαμε μέσω αυτών των προσομοιώσεων:

αλγόριθμος	Σύνολο παιχνιδιών	Παιχνίδια που δεν χρειάστηκαν μετανάστευση	Παιχνίδια που χρειαστήκαν μετανάστευση	Μέσος αριθμός μεταναστεύσεων ανά παιχνίδι που απαιτήθηκαν μεταναστεύσεις	Μέσος χρόνος αναμονής σε ουρά για σύνδεση	Μέσος χρόνος αναμονής για εύρεση παιχνιδιού
Elo, non visiting/FIFO	26123	26089	34	1,1	5996 ms	3027 ms
Elo, non visiting/Greedy	27356	27327	29	1,2	7238 ms	2983 ms
Elo, visiting/FIFO	26484	25913	487	1,9	4358 ms	3251 ms
Elo, visiting/Greedy	27103	26677	423	2,1	4783 ms	3014 ms
Free Choice/FIFO	26980	13382	13598	3,6	2543 ms	3215 ms
Free Choice/Greedy	27321	14067	13254	3,9	2679 ms	2986 ms
Fill Servers/FIFO	28646	2606	26340	4,7	1853 ms	3125 ms
Fill servers/Greedy	28734	2794	25940	4,8	1535 ms	3098 ms

Πίνακας 2 - Τα αποτελέσματα της προσομοίωσης μας

Τα δεδομένα κάθε κελιού είναι το αποτέλεσμα του μέσου όρου από τις 10 εκτελέσεις για τον κάθε αλγορίθμων. Είναι σημαντικό να σημειώσουμε ότι τα αποτελέσματά μας είναι σε κλίμακα 1/50. Φυσικά ο πολλαπλασιασμός με το 50 κάθε αποτελέσματος ΔΕΝ θα μας έδινε το αποτέλεσμα μίας πλήρους κλίμακας, καθώς το πρόβλημά μας ΔΕΝ είναι γραμμικό.

Σχολιασμός των αποτελεσμάτων

Με μία πρώτη ματιά μπορούμε να δούμε ότι λίγο πολύ σε κάθε αλγόριθμο παίζονταν περίπου 26~28 χιλιάδες παιχνίδια. Επίσης, σε κάθε

αλγόριθμο, ο χρόνος αναμονής για εύρεση παιχνιδιού ήταν περίπου 3 δευτερόλεπτα.

Οι σημαντικές διαφοροποιήσεις μεταξύ των αλγορίθμων εμφανίζονται στον αριθμό των παιχνιδιών που χρειάστηκαν μετανάστευση, στον αριθμό των ατόμων που χρειάστηκε να μεταναστεύσουν και στον χρόνο αναμονής σε ουρά για σύνδεση στον εξυπηρετητή.

Παρατηρούμε ότι όσο πιο αυστηρός είναι ο αλγόριθμος τόσο περισσότερο χρόνο αναμονής στην ουρά έχουμε, ωστόσο και τόσο λιγότερες μεταναστεύσεις χρειάζονται. Αντιθέτως, όσο πιο χαλαρός στην επιλογή εξυπηρετητή προς σύνδεση είναι ο αλγόριθμος, τόσο μικρότερος είναι ο χρόνος αναμονής για σύνδεση στον εξυπηρετητή, αλλά τόσο περισσότερες μεταναστεύσεις χρειάζονται.

Μεταξύ ίδιων αλγορίθμων σύνδεσης στον εξυπηρετητή, ο FIFO τρόπος εξυπηρέτησης της ουράς έδωσε μικρότερο μέσο όρο χρόνου αναμονής αλλά περισσότερα παιχνίδια είχαν απαίτηση για μετανάστευση χρηστών. Ωστόσο, ο αριθμός των ατόμων που χρειάζονταν μετανάστευση ήταν ελαφρώς μεγαλύτερος στην εξυπηρέτηση ουράς με άπλητο τρόπο.

Προτείνεται η εκ νέου διεξαγωγή της παραπάνω προσομοίωσης σε πλήρη κλίμακα για την επιβεβαίωση των αποτελεσμάτων ή την κατάρριψη αυτών.

VII. Χαρακτηριστικά παιχνιδιών σε κινητά τηλέφωνα

Τα διαδικτυακά ηλεκτρονικά παιχνίδια στα κινητά τηλέφωνα

Με την ραγδαία ανάπτυξη της τεχνολογίας, τα κινητά τηλέφωνα έχουν μεταμορφωθεί σε μικρούς υπολογιστές τσέπης, που μπορούν να κάνουν, σχεδόν, όσα μπορεί να κάνει και ένας κανονικός υπολογιστής. Από την αγορά των κινητών τηλεφώνων λοιπόν δεν θα μπορούσαν να λείπουν και τα ηλεκτρονικά παιχνίδια

Έως τώρα, ελάχιστα παιχνίδια για κινητό απαιτούν επικοινωνία πραγματικού χρόνου για την διεξαγωγή ενός παιχνιδιού. Συνήθως είναι παιχνίδια τα οποία διεξάγονται σε γύρους. Πρώτα παίζει ένας παίκτης, ύστερα ένας άλλος και ούτω καθ' εξής, χωρίς να υπάρχει πρόβλημα στο πόσο μεγάλη θα είναι η χρονική διάρκεια μεταξύ των γύρων των παικτών. Σε ένα επιτραπέζιο παιχνίδι για παράδειγμα, ένας χρήστης μπορεί να κάνει μία κίνηση την Τρίτη και ο επόμενος να κάνει μία κίνηση την Τετάρτη, χωρίς αυτό να επηρεάζει το αποτέλεσμα του παιχνιδιού

Τα παιχνίδια MOBA ωστόσο απαιτούν επικοινωνία πραγματικού χρόνου, και μάλιστα γρήγορη επικοινωνία. Κάθε δευτερόλεπτο μετράει και μπορεί να αλλάξει το αποτέλεσμα ενός παιχνιδιού. Αυτό σημαίνει συνεπώς ότι όλοι οι παίκτες θα πρέπει να είναι συνδεδεμένοι ταυτόχρονα για ένα αδιευκρίνιστα μεγάλο χρονικό διάστημα.

Χαρακτηριστικά δικτύων κινητής επικοινωνίας

Η ποιότητα του σήματος ενός κινητού είναι σχετική με την θέση του στον χώρο, την απόστασή του από την κεραία επικοινωνίας, τυχόν παρεμβολές και πολλά άλλα. Είναι προφανές λοιπόν ότι δεν είναι εύκολο να έχει ένας χρήστης καλή ποιότητα σήματος συνεχώς.

Τα σύγχρονα δίκτυα κινητής τηλεφωνίας προσφέρουν μεγάλες ταχύτητες επικοινωνίας. Το πρότυπο που εφαρμόζεται σήμερα είναι το 4G, είτε LTE Advanced είτε WiMAX 2 (802.16m), ανάλογα με τον πάροχο. Οι θεωρητικές μέγιστες ταχύτητες που προσφέρουν και τα 2 πρωτόκολλα είναι 100Mbps καταφόρτωσης και 60Mbps ανωφόρτωσης, σε κίνηση. [12]

Χαρακτηριστικά κινητών τηλεφώνων

Όπως είπαμε και προηγουμένως, ο ρυθμός ανάπτυξης της τεχνολογίας των κινητών τηλεφώνων είναι ραγδαίος. Χαρακτηριστικό παράδειγμα είναι το γεγονός ότι τα κινητά πλέον έχουν επεξεργαστές με 4 πυρήνες και έχουν επεξεργαστική ισχύ μεγαλύτερη του 1 Gigahertz. [13] Επίσης, τα περισσότερα κινητά τηλέφωνα πλέον λειτουργούν με οθόνες αφής και ελάχιστα πλήκτρα.

VIII. Συμπεράσματα

Ποιόν αλγόριθμο και σε ποια περίπτωση

Όπως είδαμε, οι αλγόριθμοί μας, εξυπηρετούν διαφορετικά πράγματα.

Οι μεν αυστηροί αλγόριθμοι επιλογής εξυπηρετητή προς σύνδεση, μας δίνουν πιο καλή κατανομή των χρηστών στους εξυπηρετητές, τέτοια ώστε να μην χρειάζονται συχνά, έως και καθόλου, μεταναστεύσεις μεταξύ εξυπηρετητών. Αυτό βέβαια συνεπάγεται το γεγονός ότι οι χρήστες κατά μέσο όρο, στις ώρες αιχμής, συναντούν σχετικά μεγάλη ουρά αναμονής.

Κατά τις ώρες αιχμής που παίζονται και τα περισσότερα παιχνίδια, αν τα εσωτερικά δίκτυα επικοινωνίας των εξυπηρετητών επιβαρύνονταν επιπλέον με πολλές μεταναστεύσεις, πέρα από τα δεδομένα παιχνιδιών, θα υπήρχε πρόβλημα. Ο κατασκευαστής του παιχνιδιού θα έπρεπε είτε να επενδύσει σε ακριβότερες γραμμές επικοινωνίας, μεγαλύτερης ταχύτητας, είτε να θυσιάσει ποιότητα παιχνιδιού. Αυτό σημαίνει ότι οι αυστηροί αλγόριθμοι μειώνουν το κόστος της επικοινωνίας κατά τις ώρες αιχμής.

Από την άλλη, οι αυστηροί αλγόριθμοι, αν και προτιμότεροι για τις ώρες αιχμής, δημιουργούσαν ουρά σύνδεσης καθ' όλη σχεδόν τη διάρκεια της μέρας, με αποτέλεσμα όλοι οι χρήστες σχεδόν να πρέπει να αναμείνουν σε σειρά πριν συνδεθούν.

Οι ελαστικότεροι αλγόριθμοι έδωσαν πολύ μικρότερους χρόνους αναμονής σε ουρά και δημιουργούσαν ουρά σύνδεσης μόνο κατά τις ώρες αιχμής. Ωστόσο, κατά τις ώρες αιχμής χρειάζονταν πολλές μεταναστεύσεις ανά παιχνίδι. Πάνω από τα μισά παιχνίδια στον αλγόριθμο ελεύθερης επιλογής και σχεδόν όλα στον αλγόριθμο γεμίσματος εξυπηρετητών απαιτούσαν μεταναστεύσεις. Μάλιστα, ο αριθμός μεταναστεύσεων ήταν πολύ συχνά πάνω του τέσσερα.

Ένα τέτοιο φορτίο επικοινωνίας κατά τις ώρες αιχμής θα ήταν καταστροφικό για το δίκτυο των εξυπηρετητών, όπως είναι προφανές, ωστόσο δίνει στον χρήστη πολύ μικρότερους χρόνους αναμονής στην ουρά για σύνδεση στον εξυπηρετητή.

Πιστεύουμε λοιπόν ότι η πιθανή λύση για τον βέλτιστο τρόπο εξυπηρέτησης είναι μία συνδυαστική χρήση των αλγορίθμων. Θα ήταν δυνατό μία εταιρία να χρησιμοποιεί πολλούς αλγορίθμους εξυπηρέτησης σε μία ημέρα, ανάλογα με το φόρτο και τον αριθμό χρηστών.

Δεδομένου ότι είναι σχετικά σταθερά τα μοτίβα των ωρών αιχμής, θα μπορούσε η κατασκευάστρια εταιρία, κατά τη διάρκεια των ωρών αιχμής, και λίγο πριν απ αυτές, να χρησιμοποιεί αυστηρούς αλγόριθμους σύνδεσης, για να μην υπάρχουν πολλές μεταναστεύσεις, ενώ κατά τις υπόλοιπες ώρες θα μπορούσε να χρησιμοποιεί τους λιγότερο αυστηρούς αλγορίθμους, δεδομένου ότι ο αριθμός των παιχνιδιών είναι μικρός και μπορεί να προσφέρει στους χρήστες μικρότερους χρόνους σύνδεσης με την χρήση των αλγορίθμων αυτών.

Παιχνίδια MOBA σε κινητά τηλέφωνα

Όπως είδαμε στο κεφάλαιο επτά, αν και οι ταχύτητες που υπάρχουν σήμερα είναι υπεραρκετές για να καλύψουν τις ανάγκες των MOBA παιχνιδιών, τα δίκτυα κινητής τηλεφωνίας δεν παρέχουν πάντα καλή ποιότητα σήματος. Η αστάθεια της ποιότητας του σήματος είναι τροχοπέδη για την δημιουργία παιχνιδιών με απαιτήσεις πραγματικού χρόνου.

Το υλισμικό των κινητών τηλεφώνων φαίνεται πως είναι ικανό να καλύψει τις απαιτήσεις ενός παιχνιδιού πραγματικού χρόνου. Οι επεξεργαστές που χρησιμοποιούνται σήμερα στα κινητά είναι πολύ ισχυροί καθώς και η μνήμη RAM που διαθέτουν είναι μεγάλη. [13]

Εν κατακλείδι, λόγω της αστάθειας της ποιότητας του σήματος των κινητών τηλεφώνων, φαίνεται να μην είναι δυνατή η υλοποίηση παιχνιδιών με απαιτήσεις δικτύου πραγματικού χρόνου για κινητά με την μεταφορά δεδομένων να γίνεται μέσω του δικτύου κινητής τηλεφωνίας. Ωστόσο, το γεγονός ότι το υλισμικό των κινητών είναι ικανό να υποστηρίξει τέτοιες εφαρμογές, μας δίνει την ικανότητα να αναπτυχθούν παιχνίδια με απαιτήσεις πραγματικού χρόνου, αλλά με την απαίτηση σύνδεσης Wi-Fi ή στο μέλλον, αν βελτιωθεί η ποιότητα του σήματος με τη μείωση των ασταθειών, και μέσω των δικτύων της κινητής τηλεφωνίας.

Βιβλιογραφία

[1] X. T. Lu Zhang, "Optimizing Client Assignment for Enhancing Interactivity in Distributed Interactive Applications," *IEEE/ACM TRANSACTIONS ON NETWORKING,* vol. 20, no. 6, p. 13, 2012.

[2] J. C. D. C. T. K. Y. C. E. Y. Jaecheol Kim, «Traffic Characteristics of a Massively Multi-Player Online Role Playing Gmae».

[3] E. G. S. B. M. C. E. A. Nathan Sheldon, «The effects of latency on user performance in Warcraft III,» CS Dept., Worcester Polyethnic Institute, Worcester, MA, USA.

[4] M. T. Paul Bettner, «Network Programming in Age of Empires and Beyond,» Gamasutra.com, 2001.

[5] W. Stallings, Queueing Analysis, 2000.

[6] R. Games, «League of Legends players summit a new peak,» Riot Games, 12 3 2013. [Ηλεκτρονικό]. Available: http://www.riotgames.com/articles/20130312/700/league-legends-players-summit-new-peak. [Πρόσβαση 23 9 2013].

[7] R. Games, «League of Legends' Growth Spells Bad News for Teemo,» Riot Games, 15 10 2012. [Ηλεκτρονικό]. Available: http://www.riotgames.com/articles/20121015/138/league-legends-growth-spells-bad-news-teemo. [Πρόσβαση 23 9 2013].

[8] R. Games, «11.5 million playing LoL each month,» Riot Games, 18 11 11. [Ηλεκτρονικό]. Available: http://www.riotgames.com/articles/20111118/508/11-million-playing-lol. [Πρόσβαση 23 9 2013].

[9] Steam2Stranger, «SteamGraph,» SteamGraph, [Ηλεκτρονικό]. Available: http://steamgraph.net/. [Πρόσβαση 18 9 2013].

[10] Steam, «Steam & Game Stats,» Steam, [Ηλεκτρονικό]. Available: http://store.steampowered.com/stats/. [Πρόσβαση 18 9 2013].

[11] Ν. Σ. -. Λ. Θ. Ζαρογιάννης, Συστήματα Αυτόματου Ελέγχου.

[12] J. Vilches, «Everything You Need To Know About 4G Wireless Technology,» TechSpot, 29 4 2010. [Ηλεκτρονικό]. Available:

http://www.techspot.com/guides/272-everything-about-4g/. [Πρόσβαση 23 9 2013].

[13] phoneArena, «Samsung Galaxy S III,» phoneArena.com, [Ηλεκτρονικό]. Available: http://www.phonearena.com/phones/Samsung-Galaxy-S-III_id6330. [Πρόσβαση 23 9 2013].

Ευχαριστίες

Η πτυχιακή εργασία αυτή θα ήταν ένα όνειρο θερινής νυχτός χωρίς την συμβολή των παρακάτω ατόμων. Θα ήθελα να τους αφιερώσω λοιπόν αυτή τη σελίδα για να τους πω ένα μεγάλο ευχαριστώ.

1. Ευχαριστώ τον καθηγητή μου, κύριο Κωνσταντίνο Τσερπέ για την εξαιρετική του βοήθεια όποτε την χρειάστηκα
2. Ευχαριστώ όλους τους καθηγητές μου, που με δίδαξαν και με έκαναν τον προγραμματιστή που είμαι
3. Ευχαριστώ τον αδερφό μου Σταμάτη Νικολόπουλο, Φυσικό Α.Π.Θ., για την στήριξή του και την διορατικότητά του
4. Ευχαριστώ την οικογένειά μου που με στήριξε καθ' όλη τη διάρκεια των σπουδών μου
5. Ευχαριστώ την κοπέλα μου για την στήριξη και τη συμπαράστασή της.
6. Ευχαριστώ τους Riot Moonleaf καιRiot Rincewind για την εξαιρετικής ποιότητας και ταχύτητας βοήθεια που μου προσέφεραν
7. Ευχαριστώ όλη την παρέα από το Χαροκόπειο που με τη τρέλα τους με έφεραν ως εδώ.
8. Τέλος ευχαριστώ όλους εσάς που διαβάσατε αυτό το πόνημα ως εδώ και διαβάζετε και τις ευχαριστίες. Είστε τέλειοι. Μην αφήσετε ποτέ κανέναν να σας πει το αντίθετο.