Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής & Συστημάτων Πληροφορικής

Προσωρινή Αποθήκευση και Συστήματα Συστάσεων σε Δομές Δικτύων

Διπλωματική Εργασία

Άγγελος Καρακούλιας

Επιβλέπων Καθηγητής

Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εργαστήριο NETwork Management & Optimal DEsign
Αθήνα, Ιούνιος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Προσωρινή Αποθήκευση και Συστήματα Συστάσεων σε Δομές Δικτύων

Διπλωματική Εργασία

Άγγελος Καρακούλιας

Επιβλέπων Καθηγητής
Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Ιουνίου 2020

.....	.....	.....
Συμεών Παπαβασιλείου	Θεοδώρα Βαρβαρίγου	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Επίκ. Καθηγήτρια Ε.Μ.Π.

Εργαστήριο NETwork Management & Optimal DEsign
Αθήνα, Ιούνιος 2020

.....

Άγγελος Καρακούλιας
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στη σημερινή κοινωνία της πληροφορίας με την αναδυόμενη **5G** τεχνολογία, ο όγκος των δεδομένων που διακινούνται στο Διαδίκτυο αυξάνεται με ραγδαίο ρυθμό. Ως αποτέλεσμα, παρατηρείται έντονη συμφόρηση σε βασικές συνδέσεις του δικτύου (backhaul). Στην παρούσα διπλωματική, προς αντιμετώπιση του παραπάνω προβλήματος, προτείνεται ένα Σύστημα Συστάσεων σε ένα Δίκτυο Διανομής Δεδομένων (Content Delivery Network), ως εργαλείο για τον καθορισμό του περιεχομένου προς προσωρινή αποθήκευση (Cache). Έτσι, διαλέγοντας φιλικές προς τα ενδιαφέροντα των χρηστών προτάσεις, τοποθετούνται σε κέντρα δεδομένων που βρίσκονται γεωγραφικά κοντά τους, δημοφιλή αρχεία που είναι αρκετά πιθανό να ζητηθούν, μέσω δυναμικού προγραμματισμού. Στη συνέχεια, μέσω της Ανάλυσης Σύνθετων Δικτύων (Complex Network Analysis), οργανώνεται η επικοινωνία και η ανταλλαγή πακέτων μεταξύ των κινητών συσκευών των χρηστών και εντοπίζονται κοινότητες χρηστών (clustering) με κριτήριο τα ενδιαφέροντα και τη γεωγραφική τους θέση. Για κάθε κοινότητα επιλέγεται, βάσει μετρικών της Ανάλυσης Κοινωνικών Δικτύων, ένας κεντρικός κόμβος (Clusterhead), ο οποίος δεσμεύει μνήμη στο κινητό του τηλέφωνο και αποθηκεύει σε αυτήν αρχεία τα οποία είναι δημοφιλή στην κοινότητά του. Αναπτύσσεται ένας αλγόριθμος συστάσεων που συνδυάζει τη λειτουργία προσωρινής αποθήκευσης των τοπικών κέντρων δεδομένων και την επικοινωνία μεταξύ των χρηστών, προκειμένου να ικανοποιούνται όσο το δυνατόν περισσότερα αιτήματα χρηστών γρήγορα, ποιοτικά και τοπικά, μειώνοντας σημαντικά τη χρήση πόρων του δικτύου. Τέλος, για διάφορες τιμές των παραμέτρων που διαμορφώνουν το σύστημα, ελέγχεται με μια σειρά μετρικών, η αποτελεσματικότητα του αλγορίθμου.

Λέξεις Κλειδιά

Προσωρινή αποθήκευση, σύστημα συστάσεων, ανάλυση σύνθετων δικτύων, δίκτυο διανομής δεδομένων, ομαδοποίηση χρηστών, επικοινωνία μεταξύ χρηστών, εντοπισμός κοινοτήτων, αποσυμφόρηση δικτύου, δυναμικός προγραμματισμός, πόροι δικτύου, μνήμη σταθερού ή μεταβλητού μεγέθους.

Abstract

In today's information society and due to the emerging 5G technology, the amount of data transferred over the Internet is growing rapidly. As a result, there is severe congestion in the core links of the network (backhaul). In the present thesis, to address this issue, a Recommendation System in a Content Delivery Network is proposed, as a tool for determining the content eligible for temporary storage (Caching). By selecting user-friendly Recommendations, popular files or items that are quite likely to be requested by a group of users, are placed in Base Stations that are geographically close to them, through dynamic programming. Then, by exploiting features from Complex Network Analysis, the communication and package exchange among the users' mobile devices is organized and user communities are identified (clustering), based on their interests and location. Next, for each community and by using metrics of Social Network Analysis, a central node (Clusterhead) is selected, which node allocates memory to its mobile phone and stores in it files that are popular in its community. A recommendation algorithm is developed that combines the caching function of the Base Stations and the communication between users, in order to satisfy as many user requests as possible locally, qualitatively and quickly, reducing the use of the backhaul. Finally, by giving different values to the parameters that shape the system, the effectiveness of the algorithm is measured with a series of metrics.

Key Words

Caching, Recommender System, Complex Network Analysis, Content Delivery Network, Clustering, Communication between users' mobiles, Community identification, Network decongestion, Dynamic programming, Network resources, cache of stable or non-stable size.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή, κύριο Συμεών Παπαβασιλείου, ο οποίος με καθοδήγησε και με εμπιστεύτηκε μέσω της παρούσας διπλωματικής, προκειμένου να εμβαθύνω και να ανακαλύψω περαιτέρω το πεδίο της Ανάλυσης των Σύνθετων Δικτύων.

Επίσης, θα ήθελα να ευχαριστήσω ξεχωριστά τον αναπληρωτή καθηγητή πληροφορικής Βασίλειο Καρυώτη, και τους υποψήφιους διδάκτορες Μαργαρίτα Βιτοροπούλου και Κωνσταντίνο Τσιτσεκλή που με την καθοδήγηση, τις ιδέες, την υπομονή και τις συνεχείς συναντήσεις που πραγματοποιήσαμε, κατάφερα όχι μόνο να ολοκληρώσω τη διπλωματική εργασία, αλλά και να μάθω χρήσιμες επιστημονικές γνώσεις για το αντικείμενο.

Θα ήθελα να ευχαριστήσω τους γονείς μου και τον αδερφό μου για τη στήριξη, την εμπιστοσύνη και την ελευθερία που μου προσέφεραν αυτά τα τόσο σημαντικά για τη διαμόρφωση του χαρακτήρα ενός ανθρώπου, πανεπιστημιακά χρόνια. Τέλος, να ευχαριστήσω τους φίλους που έκανα και τα άτομα που γνώρισα στο πανεπιστήμιο, οι οποίοι έκαναν τα χρόνια αυτά αξέχαστα, ανέμελα, δημιουργικά και που ήταν το στήριγμά μου στα εύκολα και στα δύσκολα.

Καρακούλιας Άγγελος
Αθήνα, Ιούνιος 2020

Περιεχόμενα

Κατάλογος Σχημάτων

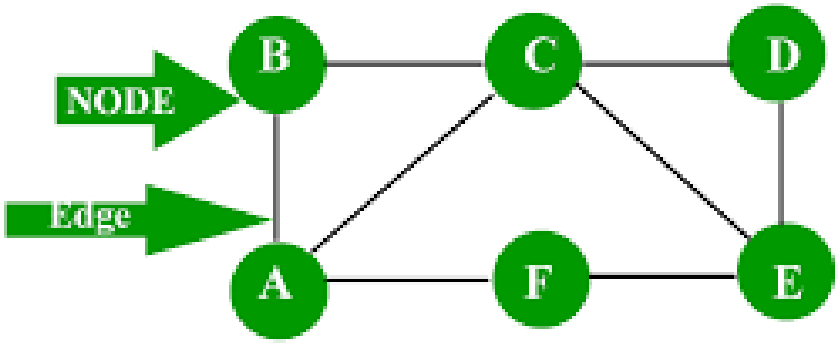
Κατάλογος Πινάκων

1 Θεωρία των γράφων (Complex Networks)

1.1 Γράφος και χαρακτηριστικά

1.1.1 Ορισμός

Ένας γράφος $G = (V, E)$ αποτελείται από ένα σύνολο κόμβων V και ένα σύνολο ακμών (edges) E. Δύο κόμβοι (i, j) ενός γράφου λέμε πως είναι γείτονες εάν συνδέονται μέσω μιας ακμής.



Σχήμα 1.1: Οι κόμβοι και οι ακμές ενός γράφου [1]

Πιο απλά, αναφέρουμε ότι οι γράφοι είναι μοντέλα διασυνδέσεων ανάμεσα σε ένα σύνολο πραγμάτων. Στην πραγματικότητα, πολλές δομές της ζωής μπορούν να χαρακτηριστούν μέσω των γράφων, μεταξύ των οποίων είναι και οι κοινωνικές σχέσεις των ανθρώπων, το διαδίκτυο, τηλεπικοινωνιακά συστήματα, σχέσεις μεταξύ έξυπνων αντικειμένων (Internet of Things).

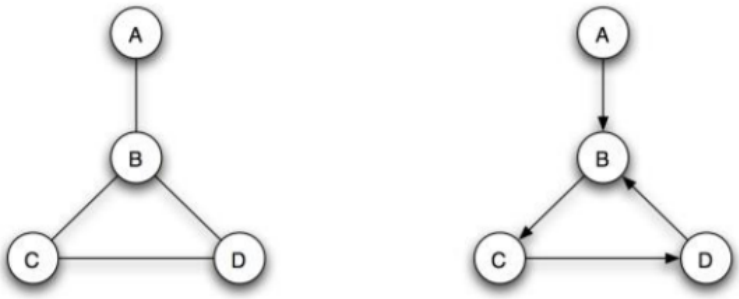
1.1.2 Βασικά Χαρακτηριστικά

Παρακάτω παρουσιάζονται τα βασικότερα χαρακτηριστικά των γράφων τα οποία μας ενδιαφέρουν για την εργασία.

Αρχικά, ως προς την κατεύθυνση των ακμών, αναφέρουμε τα δύο βασικότερα είδη γράφων:

- Μη κατευθυνόμενοι γράφοι: Γράφοι των οποίων οι ακμές δεν έχουν κατεύθυνση.
- Κατευθυνόμενοι γράφοι: Γράφοι των οποίων οι ακμές έχουν κατεύθυνση.

Εμείς θα ασχοληθούμε με τους μη-κατευθυνόμενους γράφους.



Σχήμα 1.2: Μη-Κατευθυνόμενος και Κατευθυνόμενος γράφος [2]

Ο $G' = (V', E')$ είναι ένας *υπογράφος* του $G = (V, E)$ όταν οι κόμβοι και οι ακμές του G' είναι ένα υποσύνολο των κόμβων και των ακμών του G αντίστοιχα, δηλαδή $V' \subset V$, $E' \subset E$ και άρα $G' \subset G$.

Ονομάζουμε **γειτονιά** ενός κόμβου $x \in$ G, συμβολιζόμενη με *Γ(x)* ή *N(x)* ένα σύνολο κόμβων του γράφου G που είναι άμεσα (χωρίς τη μεσολάβηση άλλων κόμβων) συνδεδεμένοι με τον x μέσω ακμής. Σε ένα μη κατευθυνόμενο γράφο, ο **βαθμός (degree)** ενός κόμβου i, $d(i)$ είναι ο αριθμός των γειτόνων που έχει ο συγκεκριμένος κόμβος.

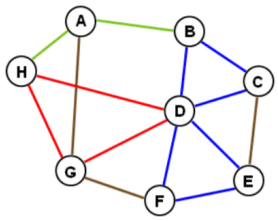
Πίνακα γειτνίασης $\mathbf{A} = [a_{ij}]$ για ένα γράφο G με n κορυφές, ορίζουμε τον $n * n$ πίνακα, ώστε $a_{ij} = 1$ όταν υπάρχει ακμή μεταξύ της κορυφής i και της κορυφής j, με $i, j \in n$.

Διαδρομή (path) ενός γράφου είναι μια ακολουθία κόμβων με την προϋπόθεση πως κάθε ένα διαδοχικό ζεύγος κόμβων συνδέεται μέσω μια ακμής (μπορεί επίσης να οριστεί και ως μια ακολουθία ακμών).

Κύκλος (cycle) ενός γράφου είναι μια κυκλική διαδρομή που ξεκινάει και τελειώνει στον ίδιο κόμβο.

Μήκος Διαδρομής είναι ο αριθμός των βημάτων (hops) που χρειαζόμαστε προκειμένου να μεταβούμε από ένα αρχικό σε ένα τελικό κόμβο μιας διαδρομής.**Απόσταση d(i,j)** δύο κόμβων i, j ονομάζουμε τον αριθμό των βημάτων της συντομότερης διαδρομής μεταξύ του i και του j.

Κλίκα (clique) σε ένα μη κατευθυνόμενο γράφο είναι ένα υποσύνολο του γράφου τέτοιο ώστε κάθε ζεύγος κόμβων αυτού να συνδέονται μέσω μιας ακμής.



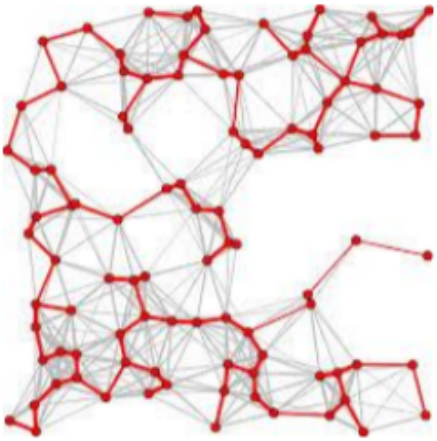
Σχήμα 1.3: Πράσινο: διαδρομή γράφου, Κόκκινο: κύκλος γράφου [2]

1.2 Τυχαίος Γράφος (Random graph)

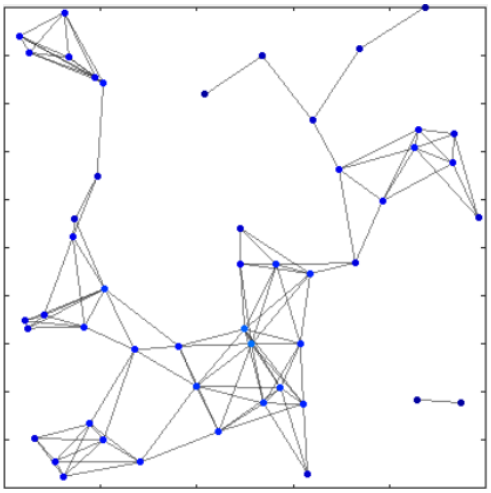
Τυχαίο ονομάζουμε ένα γράφο του οποίου οι κόμβοι συνδέονται τυχαία και ομοιόμορφα σύμφωνα με πιθανοτικές κατανομές με ορισμένα χαρακτηριστικά.

Εμείς θα κάνουμε χρήση του **Τυχαίου Γεωμετρικού Γράφου (Random Geometric Graph, RGG)**. Πρόκειται για μια υποκατηγορία τυχαίου γράφου που λειτουργεί με πιθανοτικά χωρικά μοντέλα. Πιο συγκεκριμένα ο RGG τοποθετεί n κόμβους τυχαία σε ένα χώρο κάποιας διάστασης (π.χ. xy) και τους ενώνει μέσω ακμών αν η ευκλείδεια απόστασή τους είναι μικρότερη ή ίση από την ακτίνα r που ορίζουμε.

Συμβολίζουμε τον RGG με $\boldsymbol{G(n,r)}$ όπου n : αριθμός κόμβων, r : ακτίνα.



Σχήμα 1.4: Τυχαίος Γράφος [3]



Σχήμα 1.5: Τυχαίος Γεωμετρικός Γράφος [3]

1.3 Κεντρικότητες Γράφου (Graph Centrality)

1.3.1 Ορισμός

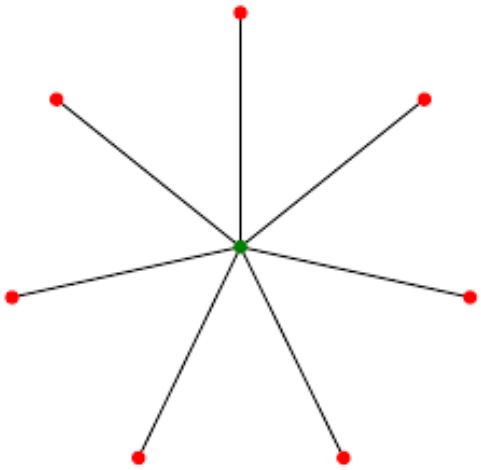
Για την περαιτέρω μελέτη των γράφων και των ιδιοτήτων τους, έχει οριστεί η έννοια της *κεντρικότητας (centrality)*. Πρόκειται για ένα σύνολο μεθόδων μέτρησης της σημαντικότητας

ενός κόμβου ή μιας ομάδας κόμβων σε ένα γράφο. Ωστόσο, ανάλογα με το πρόβλημα που αντιμετωπίζουμε, δίνουμε έμφαση σε διαφορετικά χαρακτηριστικά του γράφου και των κόμβων, προκειμένου να αποφανθούμε τη σημαντικότητά τους. Έτσι, υπάρχουν ποικίλες μέθοδοι μέτρησης της κεντρικότητας, λαμβάνοντας κάθε φορά υπόψη διαφορετικά χαρακτηριστικά (ακμές, γειτονιά κόμβου), δίνοντάς τους διαφορετική βαρύτητα σε κάθε περίπτωση. Οι σημαντικότεροι από αυτούς και για τους οποίους θα συζητήσουμε είναι:

- **Κεντρικότητα Βαθμού (Degree Centrality)**
- **Κεντρικότητα Εγγύτητας (Closeness Centrality)**
- **Ενδιάμεση Κεντρικότητα (Betweenness Centrality)**

1.3.2 Κεντρικότητα Βαθμού (Degree Centrality)

Η *κεντρικότητα βαθμού*, την οποία θα χρησιμοποιήσουμε και στην παρούσα εργασία, είναι η πιο απλή μέθοδος για τη μέτρηση της κεντρικότητας ενός κόμβου σε ένα γράφο. Συγκεκριμένα, συμπίπτει με τον αριθμό των γειτονικών κόμβων, δηλαδή των κόμβων με τους οποίους συνδέεται μέσω ακμής, ένας κόμβος ενός γράφου. Για παράδειγμα, σε μια τοπολογία αστέρα ο κόμβος με την μεγαλύτερη κεντρικότητα βαθμού είναι ο κεντρικός, διότι είναι ο μοναδικός που συνδέεται με όλους τους υπόλοιπους, όπως φαίνεται και στην παρακάτω εικόνα.



Σχήμα 1.6: Γράφος-Αστέρι [4]

Ο υπολογισμός της κεντρικότητας βαθμού $C_D(k)$, καθώς και της αντίστοιχης *σχετικής (relative)* κεντρικότητας βαθμού $C_D'(k)$ για έναν κόμβο k ενός γράφου με πίνακα γειτνίασης $A = [a_{ij}]$ και με συνολικά n κόμβους, δίνεται, αντίστοιχα, από τους παρακάτω τύπους:

$$C_D(k) = \sum_{i=1}^{n} a_{ik} \tag{1.1}$$

$$C_D^{'}(k) = \frac{\sum_{i=1}^{n} a_{ik}}{n-1} \tag{1.2}$$

Η κεντρικότητα βαθμού ήταν η πρώτη μέτρηση που χρησιμοποιήθηκε για την εκτίμηση της σημασίας των κόμβων ενός γράφου. Παρόλα αυτά χρησιμοποιείται λίγο, διότι δεν λαμβάνει υπόψην άλλες παραμέτρους όπως πχ η δυνατότητα διάχυσης πληροφοριών σε απομακρυσμένους κόμβους του δικτύου ή διάχυση πληροφορίας. Για τους λόγους αυτούς αναπτύχθηκαν και οι υπόλοιποι μέθοδοι υπολογισμού της κεντρικότητας. [2]

1.3.3 Κεντρικότητα Εγγύτητας (Closeness Centrality)

Ο συγκεκριμένος τρόπος υπολογισμού της κεντρικότητας εστιάζει στο χωρικό παράγοντα. Πιο συγκεκριμένα, για τον υπολογισμό της κεντρικότητας εγγύτητας υπολογίζονται όλες οι αποστάσεις (αριθμός βημάτων) μεταξύ των κόμβων. Όσο μικρότερη απόσταση έχει ένας κόμβος από τους υπόλοιπους, τόσο μεγαλύτερη κεντρικότητα εγγύτητας έχει. Έτσι, έχουμε μια συνολικότερη εικόνα ενός γράφου και μπορούμε να βγάλουμε συμπέρασμα για τη ταχύτητα διάδοσης της πληροφορίας σε όλο τον γράφο (ακόμα και στους πιο απομακρυσμένους κόμβους).

Ο υπολογισμός της κεντρικότητας εγγύτητας $C_P(k)$, καθώς και της αντίστοιχης σχετικής κεντικότητας εγγύτητας $C_P^{'}(k)$ για έναν κόμβο k με αποστάσεις $d(i,k)$ από τους υπόλοιπους κόμβους του γράφου δίνεται αντίστοιχα από τους παρακάτω τύπους:

$$C_P(k) = \left(\sum_{i=1}^{n} d(i,k) \right)^{-1} \tag{1.3}$$

$$C_P^{'}(k) = \frac{n-1}{\sum_{i=1}^{n} d(i,k)} \tag{1.4}$$

1.3.4 Ενδιάμεση Κεντρικότητα (Betweenness Centrality)

Η ενδιάμεση κεντρικότητα είναι με τέτοιον τρόπο ορισμένη ώστε να αποτυπώνει το μέγεθος της ροής πληροφορίας που περνάει από έναν κόμβο. Πιο συγκεκριμένα μετράει τον αριθμό των φορών που ένας κόμβος βρίσκεται στο μονοπάτι (path) μιας απόστασης μεταξύ δύο κόμβων. Όσο περισσότερες φορές, δηλαδή, ένας γράφος χρησιμοποιεί τον συγκεκριμένο κόμβο, τόσο περισσότερη πληροφορία περνάει από τον τελευταίο.

Έτσι, αν g_{ij} είναι ο αριθμός των πιθανών μονοπατιών μεταξύ των κόμβων i, j και $g_{ij}(k)$ είναι ο αριθμός των μονοπατιών αυτών που περιλαμβάνεται ο κόμβος k, τότε ορίζουμε το μέγεθος $b_{ij}(k)$ όπως παρακάτω:

$$b_{ij}(k) = \frac{g_{ij}(k)}{g_{ij}}$$

Επομένως υπολογισμός της ενδιάμεσης κεντρικότητας $C_B(k)$, καθώς και της αντίστοιχης σχετικής κεντρικότητας εγγύτητας $C_B^{'}(k)$ για έναν κόμβο k με υπολογισμένο $b_{ij}(k)$ δίνεται αντίστοιχα

από τους παρακάτω τύπους:

$$C_B(k) = \sum_{i,j,k}^{n} \sum_{i,j,k}^{n} b_{ij}(k) \quad i \neq j \neq k, i < j \tag{1.5}$$

$$C_B^{'}(k) = \frac{2C_B(k)}{n^2 - 3n + 2} \tag{1.6}$$

1.4 Εντοπισμός Κοινοτήτων (Community Detection)

1.4.1 Ορισμός

Μια *κοινότητα* σε ένα γράφο ή δίκτυο αποτελείται από ένα σύνολο κόμβων (άτομα) τα οποία αλληλεπιδρούν μεταξύ τους πιο συχνά από ότι με τα άλλα άτομα εκτός της κοινότητας [2]. Η χρήση των κοινοτήτων είναι ιδιαίτερα διαδεδομένη και έχει διάφορες εφαρμογές. Για παράδειγμα μπορεί κανείς να φανταστεί τις διαδικτυακές διαφημίσεις οι οποίες απευθύνονται σε κοινό με συγκεκριμένα ενδιαφέροντα και καταναλωτικά προφίλ που ταιριάζουν περισσότερο με το αντίστοιχο προϊόν ή υπηρεσία προς πώληση.

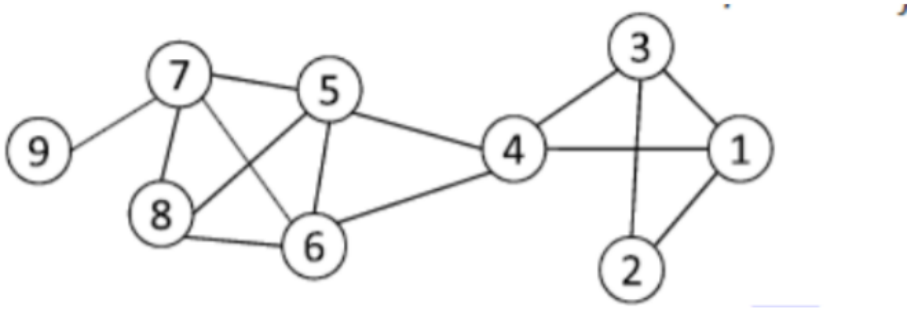
Ο εντοπισμός των κοινοτήτων διαφέρει ανάλογα με το αποτέλεσμα που θέλουμε να επιτύχουμε και επικεντρώνεται σε διαφορετικά χαρακτηριστικά των γράφων. Οι βασικότερες μέθοδοι επικεντρώνονται στα εξής στοιχεία:

- Κομβο-κεντρική Κοινότητα (Node-Centric Community). Στην προκειμένη περίπτωση ένας κόμβος του γράφου ικανοποιεί συγκεκριμένες ιδιότητες [5].
- Ομαδο-κεντρική Κοινότητα (Group-Centric Community). Εδώ οι ομάδες κόμβων, και όχι ατομικά οι κόμβοι, ικανοποιούν ορισμένες ιδιότητες.
- Δικτυο-κεντρική Κοινότητα (Network-Centric Community). Εδώ χωρίζουμε ολόκληρο το δίκτυο (γράφο) σε επιμέρους κοινότητες έτσι ώστε κάθε κόμβος να ανήκει σε μια, μέθοδο την οποία χρησιμοποιούμε στην εργασία και θα αναλύσουμε περαιτέρω [6].
- Ιεραρχική Κοινότητα (Hierarchy-Community). Εδώ με βάση την τοπολογία του δικτύου, δομούνται ιεραρχικά οι κοινότητες. Χαρακτηριστικά παραδείγματα αυτής είναι η Συγκεντρωτική ομαδοποίηση (Agglomerative hierarchical clustering) καθώς και η Διαχωριστική ομαδοποίηση (Divisive Clustering), περισσότερες πληροφορίες για τα οποία συναντούμε στο [7].

1.4.2 Μεγιστοποίηση Αρθρωτότητας (Modularity Maximization)

Ο εντοπισμός κοινοτήτων μέσω *Modularity Maximization* ανήκει στην κατηγορία των δικτυοκεντρικών κοινοτήτων. Πιο συγκεκριμένα χωρίζει το γράφο σε κοινότητες, μεγιστοποιώντας τη δύναμη (strength) και άρα το modularity του διαμελισμού κοινοτήτων που έχει πραγματοποιήσει.[2]

Αν υποθέσουμε ότι ένα δίκτυο έχει m ακμές και ότι κάθε κόμβος έχει βαθμό (degree) d_i, τότε ο αναμενόμενος αριθμός ακμών μεταξύ δύο κόμβων είναι $\frac{d_i d_j}{2m}$, όπως για παράδειγμα φαίνεται στην παρακάτω εικόνα [2].



Σχήμα 1.7: Αναμενόμενος αριθμός ακμών μεταξύ 2 κόμβων [2]

Η δύναμη (strength) μιας κοινότητας C ορίζεται ως:

$$\sum_{i \in C, j \in C} A_{ij} - d_i d_j / 2m \tag{1.7}$$

Τέλος, το modularity Q για ολόκληρο το δίκτυο και για αριθμό κοινοτήτων l είναι:

$$Q = \frac{1}{2m} \sum_{l=1, i \in C_l, j \in C_l}^{k} \sum (A_{ij} - d_i d_j / 2m) \tag{1.8}$$

Όσο πιο μεγάλη είναι η τιμή του Q για ένα γράφο, τόσο το καλύτερο. Ουσιαστικά διαμορφώνουμε με τέτοιο τρόπο τις κοινότητες έτσι ώστε να υπάρχουν πολλές ακμές εντός αυτών. Υπάρχουν διάφοροι αλγόριθμοι οι οποίοι πραγματοποιούν τη συγκεκριμένη διαδικασία. Η βασική τους ιδέα είναι η παρακάτω:

1. Αρχικά κάθε κόμβος ισοδυναμεί με μια κοινότητα (Για N κόμβους, έχω N κοινότητες).

2. Έπειτα, με συγκεκριμένη σειρά επιλογής, οι κόμβοι τοποθετούνται σε γειτονικά community μόνο εφόσον αυτό οδηγεί σε αύξηση του modularity Q.

3. Επαναλαμβάνουμε το προηγούμενο βήμα πολλές φορές και με διαφορετική σειρά, μέχρι να βρεθεί το μέγιστο modularity Q. Όταν αυτό βρεθεί, η διαμέριση του δικτύου με την οποία το πετύχαμε, θα είναι και η τελική μας μορφολογία.

2 Προσωρινή μνήμη δικτύου για κινητά τηλέφωνα (Caching in Mobile Edge Computing)

2.1 Εισαγωγή

Τις τελευταίες δεκαετίες, τα δίκτυα κινητής τηλεφωνίας εξελίσσονται με γρήγορους ρυθμούς, ξεκινώντας από τα συστήματα φωνής της πρώτης γενιάς (1G) και φτάνοντας έως την τέταρτη γενιά (4G). Αυτή η γενεαλογική πρόοδος έχει ως αποτέλεσμα την ολοένα και μεγαλύτερη αύξηση της χωρητικότητας των δικτύων, καθώς και του όγκου των δεδομένων που μεταφέρονται εντός αυτού [8]. Σήμερα, βρισκόμαστε στα πρόθυρα της πέμπτης γενιάς δικτύων κινητής τηλεφωνίας (5G), όπου αναμένονται δραματικές αλλαγές στην αρχιτεκτονική των δικτύων και στην ποσοτική αύξηση της ζήτησης για μεταφορά δεδομένων. Χαρακτηριστικά σύμφωνα με μελέτες, η ζήτηση για δίκτυα κινητής τηλεφωνίας αυξάνεται ετησίως κατά 53% και την τελευταία δεκαετία αυξήθηκε κατά 4000 φορές [8].

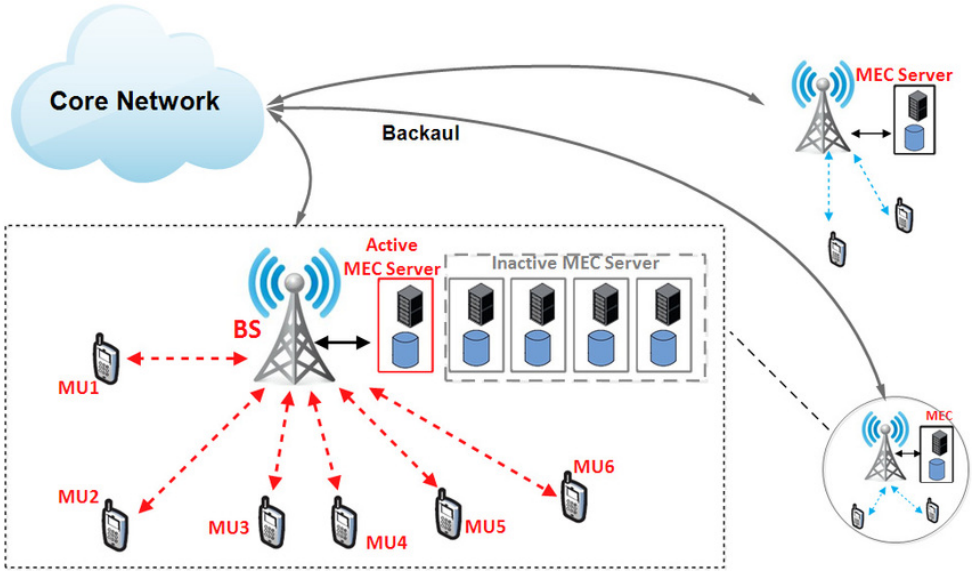
Πιο συγκεκριμένα, η ζήτηση δεδομένων για βίντεο και μουσική καταλαμβάνει πλέον το μεγαλύτερο μέρος της κίνησης στο δίκτυο, φτάνοντας έως και το 81% το 2021 [9]. Αυτό συμβαίνει, διότι τόσο οι πάροχοι αυξάνοντας τη δυνατότητα σε προσφορά δικτύων κινητής τηλεφωνίας, όσο και οι νέες τεχνολογίες των κινητών τηλεφώνων, διαμορφώνουν τις καταναλωτικές συνήθειες των χρηστών οδηγώντας τους προς μεγαλύτερη χρήση αυτών. Το παραπάνω φαινόμενο, έχει δημιουργήσει μια αλληλοτροφοδοτούμενη αλυσίδα προσφοράς και ζήτησης υπηρεσιών δικτύου που έχει σαν αποτέλεσμα να τροποποιήσει ριζικά την αρχιτεκτονική των δικτύων κινητής τηλεφωνίας, προκειμένου να ανταποκριθεί στις νέες προκλήσεις.

2.2 Mobile Edge Caching

Από τα παραπάνω συμπεραίνουμε ότι τα δίκτυα κινητής τηλεφωνίας πρέπει να διαμορφωθούν με τέτοιο τρόπο, έτσι ώστε να αποσυμφορίζονται όσο το δυνατόν περισσότερο οι φορτωμένες συνδέσεις του δικτύου (backhaul links). Για το λόγο αυτό αναπτύχθηκαν νέες τεχνικές για mobile edge caching και παράδοσης περιεχομένου (delivery techniques), όπως για παράδειγμα είναι τα **αποκεντρωμένα κέντρα δεδομένων (Base stations).**

Μέχρι τώρα τα κέντρα δεδομένων βρίσκονται σε κεντρικές τοποθεσίες, απομακρυσμένα από τους χρήστες. Έτσι, όταν κάποιος ζητήσει μια υπηρεσία, καλείται να χρησιμοποιήσει κεντρικές συνδέσεις δικτύων που τον ενώνουν με την απομακρυσμένη αυτή δομή, με αποτέλεσμα τη συμφόρηση και τη σπατάλη των πόρων των δικτύων αυτών.

Το *Mobile Edge Computing*, επιχειρώντας να προσφέρει λύση στο παραπάνω πρόβλημα, δομείται από ένα **κατανεμημένο δίκτυο υπολογιστών** που φέρνει τον υπολογισμό και την αποθήκευση δεδομένων πιο κοντά στην τοποθεσία όπου απαιτείται, βελτιώνοντας το χρόνο απόκρισης και αποσυμφορίζοντας το δίκτυο. Πιο συγκεκριμένα, αυτό επιτυγχάνεται τοποθετώντας τα παραδοσιακά κέντρα δεδομένων και τις διαδικτυακές εφαρμογές, γεωγραφικά, πιο κοντά στους χρήστες [8].



Σχήμα 2.1: Προσωρινή μνήμη στα άκρα κινητών τηλεφώνων [10]

Ένα κατατοπιστικό παράδειγμα, απεικονίζεται παραπάνω (2.1), όπου πολλά base stations (BS) επικοινωνούν με τα κινητά τηλέφωνα των χρηστών που βρίσκονται γεωγραφικά κοντά τους. Έτσι, μόνο αυτά τα κέντρα δεδομένων χρησιμοποιούν τις κεντρικές οδούς των δικτύων (Backhaul), προκειμένου να εξυπηρετήσουν τις ανάγκες των χρηστών και ως αποτέλεσμα η δικτυακή κίνηση αποσυμφορίζεται και ελέγχεται ευκολότερα.

Μια άλλη μορφή *Edge Computing* είναι η **μεταφορά δεδομένων μεταξύ συσκευών των χρηστών (device to device connection).** Στη συγκεκριμένη περίπτωση είναι δυνατή η δημιουργία ομάδων (clusters), που διαμορφώνονται λαμβάνοντας υπόψη τόσο τις κοινωνικές σχέσεις

μεταξύ των ατόμων, όσο και την γεωγραφική τους θέση, με σκοπό τη μεταφορά δεδομένων εντός αυτών. Μελέτες δείχνουν ότι το παραπάνω σύστημα, με προσεκτικά οριοθετημένες παραμέτρους, μπορεί να επιφέρει σημαντική βελτίωση ποιότητας εμπειρίας (*Quality of Experience, QoE*) που προσφέρεται στους χρήστες [8].

2.3 Τρόπος Οργάνωσης Αποκεντρωμένων Κέντρων Δεδομένων

Στο σημείο αυτό κρίνεται απαραίτητο να αναφερθεί ο τρόπος οργάνωσης των αποκεντρωμένων κέντρων δεδομένων, προκειμένου οι χρήστες να βρίσκουν αυτό που ζητούν σε αυτά, δηλαδή, να ικανοποιούνται όσο το δυνατόν περισσότερο τα αιτήματα τους τοπικά (cache hit ratio). Είναι γεγονός ότι λίγα "δημοφιλή" αρχεία (π.χ. viral video) καταλαμβάνουν το μεγαλύτερο μέρος της ζήτησης του δικτύου μια χρονική στιγμή. Έτσι, συνηθίζεται τα κέντρα αυτά, να φιλοξενούν τα παραπάνω viral αρχεία (βίντεο, μουσική, δεδομένα που αφορούν Internet Of Things) των οποίων το μέγεθος δεν είναι απαγορευτικά μεγάλο. Με τον τρόπο αυτό, και με μικρή σχετικά εναπόθεση μνήμης, τα τοπικά κέντρα ικανοποιούν πολλά αιτήματα και **αποφεύγεται η χρήση κεντρικών δικτύων (backhaul).**

Οι τρόποι διαμόρφωσης περιεχομένου των τοπικών κέντρων δεδομένων συνοψίζονται στους παρακάτω δύο:

- **Στατικός:** Εδώ κριτήριο για την εναποθέτηση δεδομένων στα κέντρα είναι η μακροπρόθεσμη ζήτησή τους. Αυτό σημαίνει ότι τα περιεχόμενα δεν αλλάζουν συχνά με αποτέλεσμα να υπάρχουν καινούρια *viral βίντεο* που να μην είναι διαθέσιμα στους χρήστες μέσω τοπικών κέντρων.

- **Δυναμικός:** Εδώ κριτήριο για την εναποθέτηση δεδομένων στα κέντρα είναι η βραχυπρόθεσμη ζήτησή τους, μέθοδος που ανταποκρίνεται στις νέες απαιτήσεις που προκύπτουν κάθε χρονική στιγμή, αλλά με συχνή χρήση backhaul [8].

2.4 Γενικά συμπεράσματα και περιορισμοί

Στην πράξη υπάρχουν διάφοροι περιορισμοί που προκύπτουν από τη χρήση των τοπικών κέντρων δεδομένων. Για παράδειγμα, ορισμένα βίντεο υψηλής ποιότητας με υψηλή ζήτηση είναι τόσο μεγάλα σε μέγεθος, που λίγα από αυτά καταναλώνουν όλη τη χωρητικότητα των κέντρων. Παράλληλα, οι προτιμήσεις της πλειοψηφίας των χρηστών αλλάζουν συχνά, με αποτέλεσμα τα viral βίντεο να ανανεώνονται συνεχώς. Έτσι τα local Base stations αναγκάζονται ανα τακτά χρονικά διαστήματα να ανατρέχουν στα κεντρικά κέντρα δεδομένων προκειμένου να τα φορτώσουν. Έτσι, χρησιμοποιούν τις κεντρικές συνδέσεις δικτύου με αποτέλεσμα την περαιτέρω συμφόρησή τους.

Επίσης, οι πάροχοι που διαχειρίζονται τα τοπικά δίκτυα θα μπορούσαν να *προβλέψουν* το τί θα ζητήσουν οι χρήστες, προκειμένου να οργανωθούν εκ των προτέρων. Αυτό επιτυγχάνεται τόσο γνω-

ρίζοντας τις προτιμήσεις των χρηστών, αλλά κυρίως γνωρίζοντας ή διαμορφώνοντας τα συστήματα συστάσεων (Recommender Systems), στα οποία θα αναφερθούμε στο επόμενο κεφάλαιο.

Συμπεραίνουμε, λοιπόν, ότι το φαινόμενο *Caching in Mobile Edge Computing*, με τη σωστή διαχείρησή του, είναι ένα χρήσιμο εργαλείο, προκειμένου να ανταποκριθούμε στις νέες ανάγκες που προκύπτουν από την αναδυόμενη *κοινωνία της πληροφορίας* [11].

3 Συστήματα Συστάσεων (Recommender Systems)

3.1 Ορισμός

Τα *Συστήματα Συστάσεων (Recommender Systems)* είναι συστήματα μέσω των οποίων οι διαχειριστές, διαδικτυακών κυρίως, υπηρεσιών προτείνουν στους χρήστες που τις χρησιμοποιούν, συγκεκριμένα περιεχόμενα προς ζήτηση των υπηρεσιών αυτών. Ένα παράδειγμα είναι τα βίντεο προς παρακολούθηση που προτείνει η εφαρμογή Youtube, κάθε φορά που ολοκληρώνεται οποιοδήποτε βίντεο.

Ο πρωταρχικός σκοπός των *Recommender Systems* είναι η ανακούφιση των χρηστών από την υπερφόρτωση πληροφοριών που μπορεί να έρχονται αντιμέτωποι, προτείνοντάς τους περιεχόμενα που ταιριάζουν με τα ενδιαφέροντα και τις προτιμήσεις τους [12]. Ωστόσο, οι συστάσεις αυτές επηρεάζουν σε τέτοιο βαθμό τους χρήστες, όπου φτάνουν τελικά στο σημείο να διαμορφώνουν τις προτιμήσεις και τα περιεχόμενου που αυτοί θα επιλέξουν. Για παράδειγμα, το σύστημα συστάσεων που χρησιμοποιεί το 'Netflix' είναι υπεύθυνο για το 80% των ωρών ροής του [13], ενώ το αντίστοιχο του 'Youtube' για το 30% [14].

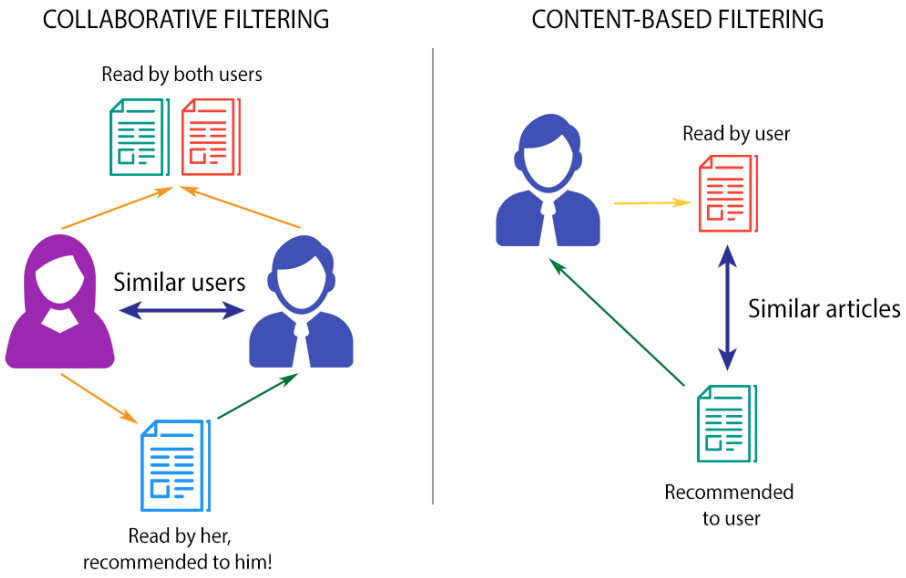
3.2 Κατηγορίες Συστημάτων Σύστασης

Στο σημείο αυτό γίνεται κατανοητό πως, προκειμένου ένα Recommender System να είναι αποτελεσματικό και οι συστάσεις που πραγματοποιεί στους χρήστες να είναι στοχευμένες, πρέπει πρώτα απ' όλα να του γίνει γνωστή η χρησιμότητα των αντικειμένων προς τους χρήστες. Δηλαδή, για κάθε χρήστη $u \in U$ και για κάθε item $i \in I$, να καθοριστεί η τιμή της χρησιμότητας (utility) μεταξύ τους, $f(u,i)$. Η διαδικασία αυτή είναι ιδιαίτερα δύσκολη και απαιτεί την ύπαρξη ενός ευέλικτου και καλά σχεδιασμένου συστήματος. Οι δύο βασικές τεχνικές σύστασης (Recommendation Techniques) για τον υπολογισμό του παραπάνω $f(u,i)$ είναι αυτή που βασίζεται στο περιεχόμενο (Content-Based) και αυτή του συμμετοχικού φιλτραρίσματος (Collaborative Filtering) [15].

- **Content-Based**. Η τεχνική αυτή εστιάζει στη διαμόρφωση του προφίλ ενός χρήστη, αναλύοντας τα χαρακτηριστικά των αντικειμένων τα οποία έχει αξιολογήσει ή χρησιμοποιήσει στο παρελθόν. Έτσι, γνωρίζοντας την χρησιμότητα $f(u,i_j)$ των αντικειμένων $i_j \in I$ που έχει έρθει

σε επαφή ο χρήστης u, υπολογίζεται στη συνέχεια η χρησιμότητα $f(u,i')$ των αντικειμένων $i' \in I$ που δεν έχει έρθει ακόμη σε επαφή, αλλά είναι όμοια με τα αντικείμενα i_j [15].

- **Collaborative Filtering**. Η τεχνική αυτή εστιάζει στην ομοιότητα μεταξύ των χρηστών, προκειμένου να προβλέψει τη χρησιμότητα ενός αντικειμένου σε ένα χρήστη. Πιο συγκεκριμένα, η χρησιμότητα $f(u,i)$ ενός χρήστη $u \in U$ για ένα αντικείμενο $i \in I$, υπολογίζεται από την ήδη διαθέσιμη χρησιμότητα $f(u_j,i)$, όπου $u_j \in U$ είναι χρήστες με όμοια ενδιαφέροντα και προτιμήσεις με τον χρήστη u [15].



Σχήμα 3.1: Οι δύο κατηγορίες Συστημάτων Σύστασης [16]

Η βασική τους ιδέα παρουσιάζεται και στην παραπάνω εικόνα. Συμπερασματικά, οι δύο αυτές τεχνικές, καθώς και συνδυασμός αυτών χρησιμοποιούνται ανάλογα με το σύστημα που έχουμε να διαχειριστούμε, καθώς έχουν διαφορετικά πλεονεκτήματα και μειονεκτήματα. Για παράδειγμα, η content-based τεχνική έχει το μειονέκτημα ότι εξαρτάται από τη διαθεσιμότητα των χαρακτηριστικών των αντικειμένων $i \in I$, ενώ η collaborative filtering τεχνική πολλές φορές αδυνατεί να συνδέσει ένα χρήστη με περισσότερες από μια ομάδες χρηστών (gray sheep problem).

3.3 Ο ρόλος του παρόχου υπηρεσιών

Από τα παραπάνω, γίνεται κατανοητό πως τα Recommender Systems είναι ένα ισχυρό διαχειριστικό εργαλείο που έχουν στη διάθεσή τους οι πάροχοι υπηρεσιών. Ωστόσο, οι πάροχοι αυτοί δεν χρησιμοποιούν τις συστάσεις αποκλειστικά προς όφελος των χρηστών, αλλά και προς το δικό τους. Πιο συγκεκριμένα, τους συμφέρει περισσότερο να γίνεται χρήση των περιεχομένων που έχουν ήδη γίνει cached, για τους λόγους που αναφέραμε στο Κεφάλαιο 2. Έτσι, χρειάζονται πολύ λιγότερους πόρους και χρόνο προκειμένου να ικανοποιήσουν τα αιτήματα των χρηστών. Επομένως, συχνά

προτείνουν περιεχόμενα που έχουν γίνει cached και ταυτόχρονα ταιριάζουν σε κάποιο βαθμό με τα ενδιαφέροντα των χρηστών [17]. Ταυτόχρονα, αν οι πάροχοι προτείνουν παρόμοια περιεχόμενα στους χρήστες και αυτοί τα επιλέγουν, τότε τα συνολικά αιτήματά τους θα είναι ομογενή και έτσι, εύκολα διαχειρίσιμα από τους παρόχους.

Καταλαβαίνουμε, λοιπόν, ότι οι προτάσεις που θα επιλέξουν οι πάροχοι υπηρεσιών να κάνουν, εξαρτάται από πολλούς παράγοντες, όπως οι πόροι που διαθέτουν, τα περιεχόμενα που έχουν κάνει cache, το είδος των χρηστών (χρήστες ειδικής μεταχείρισης). Για την κατανόηση των κατηγοριών των Recommender System παραθέτουμε παρακάτω τα δύο ακραία είδη προτάσεων:

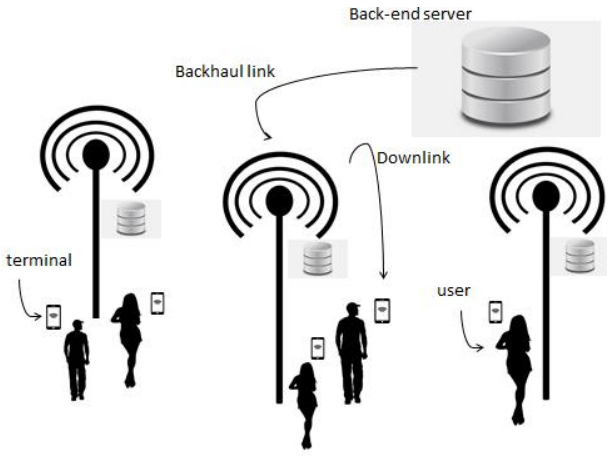
- **Προτάσεις επικεντρωμένες στα ενδιαφέροντα των χρηστών.** Εδώ οι πάροχοι συστήνουν περιεχόμενα που ταιριάζουν απόλυτα με τα ενδιαφέροντα και τις προτιμήσεις των χρηστών. Το είδος των προτάσεων αυτών θα χρησιμοποιήσουμε και στην διπλωματική εργασία.

- **Προτάσεις επικεντρωμένες στη διευκόλυνση του παρόχου.** Εδώ οι πάροχοι συστήνουν περιεχόμενα των οποίων η ζήτηση είναι εύκολο να ικανοποιηθεί ή είναι συμφέρουσα προς τους ίδιους (π.χ. διαφημίσεις).

Είναι σημαντικό να τονιστεί ότι οι δύο παραπάνω περιπτώσεις είναι οι ακραίες κατηγορίες προτάσεων και συνήθως οι πάροχοι επιλέγουν ένα συνδυασμό αυτών.

3.4 Κοινό Πρόβλημα συστάσεων και προσωρινής αποθήκευσης (joint caching and recommendation problem).

Σε συνδυασμό με το Κεφάλαιο 2, συμπεραίνουμε πως για την επίτευξη καλύτερης ποιότητας εμπειρίας (QoE) των χρηστών, με όσο το δυνατόν μικρότερη κατανάλωση πόρων δικτύου, είναι επιτακτική η συνεργασία των διαχειριστών του δικτύου και των παρόχων υπηρεσιών [17].



Σχήμα 3.2: Συστάσεις φιλικές προς το χρήστη [17]

Οι **διαχειριστές δικτύων** είναι υπεύθυνοι για την κίνηση στα δίκτυα, την αποσυμφόρησή τους καθώς και για τα περιεχόμενα που θα γίνουν cached στους τοπικούς σταθμούς δεδομένων.

Ταυτόχρονα, οι **πάροχοι υπηρεσιών** είναι υπεύθυνοι για τα recommendations που θα κάνουν στους χρήστες και για την όσο το δυνατόν καλύτερη εξυπηρέτησή τους προς όφελος της εταιρίας που ανήκουν. Επίσης, επιζητούν την ευνοϊκότερη πρόσβαση σε υπηρεσίες δικτύου.

Γίνεται, λοιπόν, αντιληπτό πως οι δύο αυτές οντότητες που πρέπει να συνεργαστούν, έχουν διαφορετικούς ρόλους και σκοπούς και πολλές φορές η επίτευξη της συνεργασίας τους είναι ιδιαίτερα δύσκολη.

Στα πλαίσια της διπλωματικής εργασίας, ωστόσο, τα κριτήρια με βάση τα οποία οι διαχειριστές επιλέγουν τα περιεχόμενα που θα κάνουν cache στα τοπικά κέντρα δεδομένων, είναι οι προτιμήσεις και τα ενδιαφέροντα των χρηστών (user-friendly). Ο τρόπος με τον οποίο γίνεται αυτό, εξηγείται αναλυτικά στη συνέχεια της εργασίας.

4 Μοντέλο Συστήματος

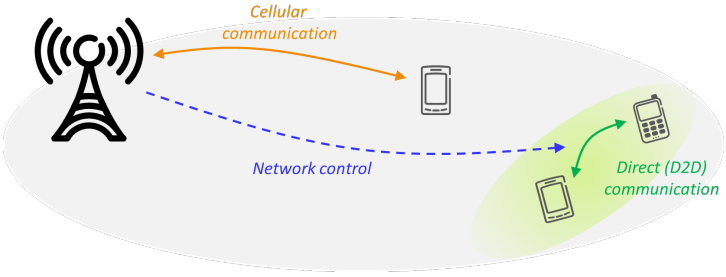
4.1 Περιγραφή συστήματος

Στη διπλωματική εργασία πραγματοποιείται η προσομοίωση ενός υποθετικού σεναρίου. Οι βασικοί παράγοντες που λαμβάνουν μέρος σε αυτή καθώς και η αρχική ιδέα είναι όπως στο [17]. Στο σενάριο αυτό υπάρχουν οι *χρήστες (users)* που βρίσκονται σε μια γεωγραφική περιοχή και έχουν όλοι πρόσβαση στο ίδιο τοπικό *κέντρο δεδομένων (Base station)*. Παράλληλα, υπάρχει ένας σχετικά μεγάλος αριθμός *αντικειμένων (items)*. Τα ενδιαφέροντα του κάθε χρήστη καθώς και οι *συστάσεις (recommendations)* που θα του γίνουν διαμορφώνουν το κατά πόσο πιθανό είναι να ζητήσει ένα αντικείμενο.

Ο αρχικός σκοπός είναι να οργανώσουμε με τέτοιον τρόπο την *cache* στα τοπικά κέντρα δεδομένων, έτσι ώστε οι χρήστες, σαν σύνολο, να βρουν όσο το δυνατόν περισσότερα αντικείμενα ζητήσουν εκεί, να μεγιστοποιηθεί, δηλαδή, το *cache hit ratio* που ορίζεται ως [17]:

$$cache\ hit\ ratio = \frac{cache\ hits}{cache\ hits + cache\ misses}$$

Στη συνέχεια, επιχειρείται η συνεργασία *mobile edge computing* που είναι τα τοπικά κέντρα δεδομένων με *mobile edge network* που, στη συγκεκριμένη περίπτωση είναι, η μεταφορά δεδομένων μεταξύ των κινητών συσκευών των χρηστών (*Device to Device connection, D2D*), συνεργασία η οποία αποτελεί και την καινοτομία της διπλωματικής.



Σχήμα 4.1: Επικοινωνία μεταξύ των κινητών συσκευών των χρηστών [18]

Θεωρούμε τους χρήστες σαν κόμβους ενός *Random Geometric Graph (RGG)*, με τις ακμές μεταξύ αυτών να διαμορφώνονται με κριτήριο τα κοινά τους ενδιαφέροντα και την ευκλείδεια απόστασή τους στο χώρο, με αποτέλεσμα, στο τέλος, να υπάρχουν συνδέσεις μεταξύ όμοιων και κοντινών χρηστών. Κάνοντας χρήση ενός αλγορίθμου *modularity maximization* δημιουργούμε πυκνά *communities* με αποτέλεσμα, κάθε χρήστης να ανήκει σε ένα από αυτά.

Στη συνέχεια, με βάση το *Degree Centrality*, ορίζεται σε κάθε community, ένας αρχηγός κοινότητας (*Clusterhead*) που δεσμεύει μνήμη στην κινητή του συσκευή, η οποία χρησιμοποιείται ως cache εντός του community του. Στην παραπάνω μνήμη μπαίνουν και πάλι αντικείμενα, προκειμένου οι χρήστες εντός του κάθε community, αυτή τη φορά, να βρίσκουν τα αντικείμενα που ζητούν, δίχως να γίνεται κατανάλωση πολύτιμων πόρων δικτύου.

Η συνολικότερη, λοιπόν, εικόνα είναι πως όταν ένας χρήστης ζητάει ένα αντικείμενο, αρχικά διατρέχει στο κοινό για όλους, κέντρο δεδομένων και αναζητά, αν αυτό που επιθυμεί υπάρχει στην cache αυτού. Αν ναι, τότε το αίτημά του ικανοποιείται χωρίς περαιτέρω κατανάλωση πόρων. Αν όχι, τότε διερευνά στο community του, και εφόσον έχει σύνδεση (ακμή) με κάποιον clusterhead, αναζητά το αντικείμενο αυτό στην cache του τελευταίου. Αν το βρει, τότε παίρνει το αντικείμενο από εκεί. Και στις δύο περιπτώσεις αν ένας χρήστης βρει το αντικείμενό του σε κάποια από τις δύο cache, θεωρείται επιτυχία και αυξάνεται το *cache hit ratio* του συστήματος. Αν όχι, τότε αναγκαστικά χρησιμοποιεί τα κεντρικά δίκτυα (backhaul), προκειμένου να ικανοποιήσει το αίτημά του και η περίπτωση θεωρείται σαν *cache miss ratio*.

Τέλος, εκτελούμε διαφορετικές προσομοιώσεις για έναν ή δύο clusterhead σε κάθε community, καθώς και για σταθερή ή μεταβλητή μνήμη cache του εκάστοτε clusterhead. Για κάθε μια από τις παραπάνω περιπτώσεις αλλάζουμε τις βασικές παραμέτρους (χρήστες, αντικείμενα) και βλέπουμε πώς ανταποκρίνεται ο αλγόριθμός μας.

Στις ενότητες που ακολουθούν, περιγράφουμε αναλυτικά τη διαδικασία μέσω της οποίας μοντελοποιούμε και διαμορφώνουμε τις παραμέτρους του συστήματος που οδηγούν τελικά στην ολοκλήρωση του ευριστικού αλγορίθμου. Αναφερόμαστε στα αντικείμενα, στους χρήστες και στις cache του συστήματος, διαμορφώνουμε τις προτιμήσεις των χρηστών, διατυπώνουμε το ρόλο που παίζουν τα recommendation στα τελικά αιτήματα των χρηστών. Έπειτα, δείχνουμε τη διαδικασία τοποθέτησης αντικειμένων στις cache, εξηγούμε τον τόπο που επιτυγάνεται η δημιουργία του γράφου, ο εντοπισμός κοινοτήτων εντός αυτού και η επιλογή clusterhead στις παραπάνω κοινότητες. Τέλος, διαμορφώνουμε τα τελικά αιτήματα των χρηστών.

Ο αλγόριθμος που αναπτύχθηκε για την υλοποίηση της προσομοίωσης γράφτηκε στην γλώσσα ανάπτυξης *Python*.

4.2 Αντικείμενα, Χρήστες, Caches

4.2.1 Αντικείμενα Συστήματος

Αρχικά, τα αντικείμενα (I) του συστήματος έχουν πεπερασμένο μέγεθος L_i, το οποίο παίρνει τιμές στο διάστημα $[1, 10]$ μέσω της συνάρτησης ομοιόμορφης κατανομής και υπόκεινται με δια-

φορετική τιμή σε ορισμένες θεματικές κατηγορίες (π.χ ψυχαγωγία, υγεία κτλ.). Ένα αντικείμενο i ταιριάζει σε περισσότερες από μια κατηγορίες με διαφορετική, όμως, βαρύτητα στην καθεμία. Ορίζουμε M διαφορετικές θεματικές κατηγορίες.

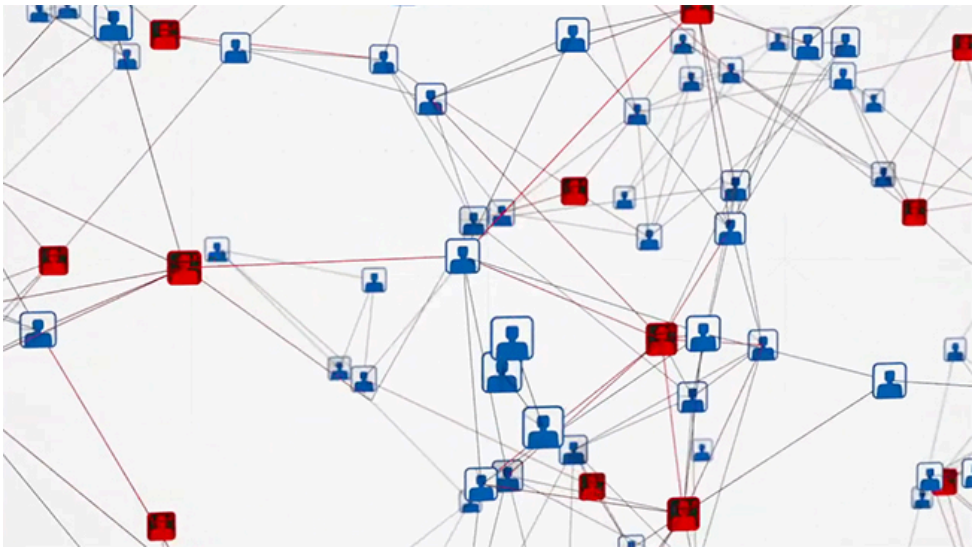
Έτσι, για κάθε αντικείμενο $i \in I$ με μέγεθος L_i, ορίζουμε το διανυσματικό μέγεθος f_i, του οποίου το j^{th} στοιχείο, $f_i(j)$, $j \in [1,..,M]$ αντιστοιχεί στο πόσο ταιριάζει το αντικείμενο i με την κατηγορία j. Όλα τα μεγέθη παίρνουν τιμές στο διάστημα $[0,1]$ και κανονικοποιούνται έτσι ώστε $\sum_{j=1}^{M} f_i(j) = 1, \forall i \in I$ [17].

Σχήμα 4.2: Αντικείμενα Συστήματος [19]

4.2.2 Χρήστες Συστήματος (Network Size)

Ο κάθε χρήστης $u \in U$, βρίσκεται σε διαφορετικό σημείο του δικτύου μας. Επομένως, ο καθένας έχει διαφορετικό τρόπο πρόσβασης, τόσο στα τοπικά κέντρα δεδομένων, όσο και στο κεντρικό δίκτυο (Cell Network).

Για τους χρήστες, όπως και για τα αντικείμενα, ορίζουμε το διανυσματικό μέγεθος f_u, του οποίου το j^{th} στοιχείο, $f_u(j), j \in [1,..,M]$, αντιστοιχεί στο πόσο ταιριάζει ο χρήστης u με την κατηγορία j. Όλα τα μεγέθη παίρνουν τιμές στο διάστημα $[0,1]$ και κανονικοποιούνται έτσι ώστε $\sum_{j=1}^{M} f_u(j) = 1, \forall u \in U$. Στην πραγματικότητα, αυτός ο παράγοντας καθορίζεται από τα cookies, από τις ιστοσελίδες που κάθε χρήστης επισκέπτεται, καθώς και τα αντικείμενα που έχει ζητήσει στο παρελθόν [17].



Σχήμα 4.3: Οι χρήστες ενός δικτύου [20]

4.2.3 Cache

Η cache C είναι ο αποθηκευτικός χώρος του τοπικού κέντρου δεδομένων και έχει καθορισμένο μέγεθος L. Κανονικά, για κάθε κέντρο δεδομένων i, υπάρχει διαφορετική cache, C_i με διαφορετικό περιεχόμενο. Στην προσομοίωσή μας, ωστόσο, θεωρούμε ένα *Base Station* με μία cache στην οποία ανατρέχουν οι χρήστες, προκειμένου να δουν εάν υπάρχει ή όχι το αντικείμενο που επιθυμούν.

4.3 Μοντελοποιώντας τις προτιμήσεις των χρηστών

Τα ενδιαφέροντα και οι προτιμήσεις των ατόμων στην πραγματικότητα αλλάζουν με το χρόνο, αν αναλογιστεί κανείς ότι εξαρτώνται από τις εκάστοτε κοινωνικο-οικονομικές συγκυρίες, τη μόδα, τα viral βίντεο που αλλάζουν ανά τακτά χρονικά διαστήματα κτλ. Παρόλα αυτά, για χάρη απλότητας, στην προσομοίωσή μας, θεωρούμε ότι ο καθένας πραγματοποιεί μια φορά ένα συγκεκριμένο αριθμό αιτημάτων. Είναι, δηλαδή, σαν να ικανοποιούμε τα αιτήματα των χρηστών για ένα συγκεκριμένο χρονικό διάστημα μιας ημέρας.

Κρίνεται απαραίτητο να τονιστεί πως οι προτιμήσεις των χρηστών p_u^{pref} δεν είναι ίδιες με τα τελικά αιτήματά τους από το σύστημα p_u^{req}, επειδή επιδρούν σε αυτούς τα recommendations p_u^{rec} και διαμορφώνουν τα τελικά αιτήματά με τρόπο που θα περιγράψουμε παρακάτω [17].

Πιο συγκεκριμένα, για κάθε χρήστη $u \in U$, ορίζουμε την κατανομή των προτιμήσεων την οποία συμβολίζουμε με p_u^{pref} και την κανονικοποιούμε έτσι ώστε $\sum_{i \in I} p_u^{pref}(i) = 1$. Ο όρος $p_u^{pref}(i)$ εκφράζει την αρχική προτίμηση που δείχνει ο χρήστης u στο αντικείμενο i. Άρα για $i \in I$, βλέπουμε την προτίμηση του χρήστη u πάνω σε όλα τα αντικείμενα του συστήματός μας. Στην παρούσα διπλωματική χρησιμοποιούμε τον διανυσματικό όρο a_{ui}, προκειμένου να υπολογίσουμε το $p_u^{pref}(i)$

κάθε χρήστη όπως στο [21].

$$a_{ui} = \frac{\sum_{j=1}^{M} f_u(j) \cdot f_i(j)}{\sqrt{\sum_{j=1}^{M} f_u(j)} \cdot \sqrt{\sum_{j=1}^{M} f_i(j)}} \tag{4.1}$$

Κανονικοποιώντας τον παραπάνω δείκτη προς όλα τα item $i \in I$ για ένα χρήστη u, υπολογίζω όλα τα $p_u^{pref}(i)$ όπως παρακάτω:

$$p_u^{pref}(i) = \frac{a_{ui}}{\sum_{i \in I} a_{ui}} \tag{4.2}$$

4.4 Η επιρροή των συστάσεων στα αιτήματα των χρηστών

Λεπτομέρειες για την επιρροή των συστημάτων συστάσεων συναντάμε στο [22]. Όπως δείξαμε στο Κεφάλαιο 3, όταν γίνεται σε ένα χρήστη u, recommendation για ένα αντικείμενο i, τότε σε συνδυασμό με το κατά πόσο ενδιαφέρει εξαρχής το αντικείμενο αυτό τον χρήστη, αυξάνεται η πιθανότητα να το ζητήσει.

Όταν, όμως, ένας παράγοντας είναι υπεύθυνος τόσο για τα recommendations, όσο και για τη διαχείριση δικτύου, ανοίγει ένας κύκλος συζητήσεων περί ηθικής. Δηλαδή, ποια είναι τα όρια μέχρι τα οποία μπορεί κάποιος πάροχος να φτάσει όταν κάνει recommend ένα αντικείμενο που δεν ταιριάζει κατά πολύ στον χρήστη; Κατά πόσο έχει υποχρέωση να σέβεται τις προτιμήσεις και τα ενδιαφέροντά του; Η απάντηση στο ερώτημα αυτό είναι ιδιαίτερα πολύπλοκη και κάθε πάροχος λαμβάνει υπόψη διαφορετικούς παράγοντες, προκειμένου να την απαντήσει.

Υπάρχουν ενδείξεις ότι τα Recommender Systems δεν επηρεάζουν μόνο εκ των υστέρων τα αιτήματα των χρηστών (p_u^{req}), αλλά επηρεάζουν και το (p_u^{pref}), δηλαδή τις αρχικές τους προτιμήσεις. Πρόκειται για ένα φαινόμενο ψυχολογίας που μπορούμε να το παρομοιάσουμε με την επιρροή των διαφημίσεων στους καταναλωτές. Παρόλα αυτά, δεν έχουμε ακριβή στοιχεία για να το συμπεριλάβουμε στη διπλωματική μας.

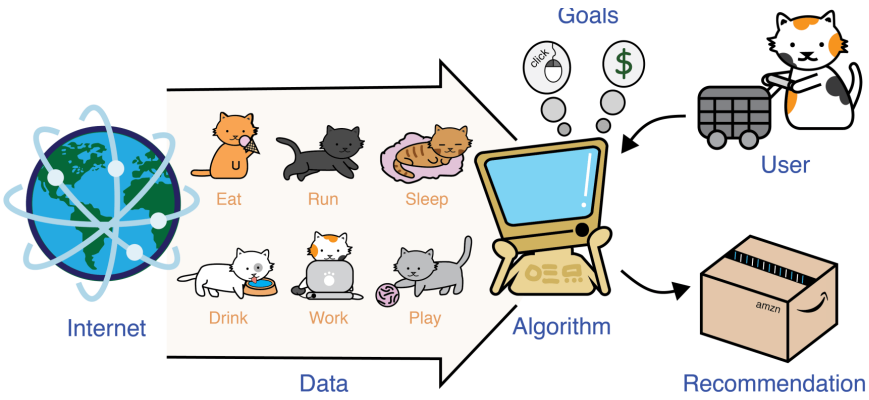
Στο [17] χρησιμοποιείται ο όρος Δ_u που ουσιαστικά εκφράζει την μέγιστη απόκλιση που μπορεί να έχει ένα recommendation από τα ενδιαφέροντα του χρήστη. Έτσι, θα μπορούσε να οριστεί ένα, κοινό για όλους, άνω όριο στο Δ_u, ούτως ώστε κανένας πάροχος υπηρεσιών να μη μπορεί να κάνει recommendation κάτι που συμφέρει μονάχα τον ίδιο και δεν ανταποκρίνεται καθόλου στα ενδιαφέροντα των χρηστών.

Εμείς, υιοθετούμε μια μέθοδο Recommendation που βρίσκεται όσο πιο κοντά γίνεται στα ενδιαφέροντα του εκάστοτε χρήστη u (**User-friendly**). Πιο συγκεκριμένα, το σύστημα συστάσεών μας προτείνει σε ένα χρήστη τα **R** αντικείμενα που αυτός προτιμάει περισσότερο (p_u^{pref}). Έτσι, δίνεται στην πιθανότητα να ζητηθούν τα R αυτά αντικείμενα, μια όμοια ώθηση η οποία ισούται με:

$$p_u^{rec} = 1/R \tag{4.3}$$

Αυτό σημαίνει πως όσο πιο πολλές σε αριθμό είναι οι συστάσεις που κάνουμε, τόσο λιγότερο θα επηρεάζουν το σύστημα. Ωστόσο, το συγκεκριμένο σύστημα συστάσεων επηρεάζει την σχετική πιθανότητα αιτήματος p_u^{req} όχι μόνο των R αντικειμένων που προτείνει, αλλά όλων των αντικειμένων

όπως φαίνεται παρακάτω.



Σχήμα 4.4: Διαδικασία επιλογής συστάσεων [23]

Επίσης, όπως γίνεται και στο [17], ορίζεται ο όρος w_u για κάθε χρήστη $u \in U$, ο οποίος παίρνει τιμές στο διάστημα $[0,1]$ και αντιστοιχεί στο κατά πόσο αυτός επηρεάζεται από τα recommendations που θα του γίνουν. Είναι λογικό να εισαγάγουμε έναν τέτοιο όρο, αφού υπάρχουν χρήστες με αυστηρά κριτήρια επιλογής που μένουν πιστοί στις προτιμήσεις τους και άλλοι που επηρεάζονται σημαντικά από τις συστάσεις. Αν $w_u \to 1$, τότε ένας χρήστης επηρεάζεται πολύ, ενώ αν $w_u \to 0$, τότε τις αγνοεί.

Τελικά ο τύπος υπολογισμού της πιθανότητας ένας χρήστης u να ζητήσει ένα αντικείμενο i, $p_u^{req}(i)$, είναι συνδυασμός των κατανομών p_u^{rec}, p_u^{pref} και του όρου w_u και υπολογίζεται όπως παρακάτω:

$$p_u^{req}(i) = w_u \cdot p_u^{req}(i) + (1 - w_u) \cdot p_u^{pref}(i) \tag{4.4}$$

για τα R αντικείμενα που γίνονται recommended στον u και:

$$p_u^{req}(i) = (1 - w_u) \cdot p_u^{pref}(i) \tag{4.5}$$

για τα $|I| - R$ αντικείμενα που δεν γίνονται recommended.

4.5 Επιλογή αντικειμένων που θα συμπεριληφθούν στην Cache

4.5.1 Κοινό Πρόβλημα συστάσεων και προσωρινής αποθήκευσης

Όπως εξηγήσαμε στο Κεφάλαιο 2, τα αντικείμενα που θα μπουν τελικά στην cache, πρέπει να είναι τέτοια ώστε να ευνοούν τόσο τους χρήστες όσο και την ομαλή λειτουργία των δικτύων.

Όσον αφορά τους χρήστες, σκοπός είναι να βρουν όσο το δυνατόν περισσότερα αντικείμενα αναζητήσουν στην cache, με αποτέλεσμα να ικανοποιηθεί γρηγορότερα το αίτημά τους και έτσι να αυξηθεί η ποιότητα υπηρεσίας (QoE) που τους προσφέρεται.

Από πλευράς δικτύου, όσα περισσότερα αιτήματα ικανοποιηθούν τοπικά στις cache τόσο λιγότερο θα χρησιμοποιηθούν κεντρικές μονάδες δικτύου (backhaul) και τόσο περισσότερο αποσυμφοριζόμενο θα είναι το δίκτυό μας.

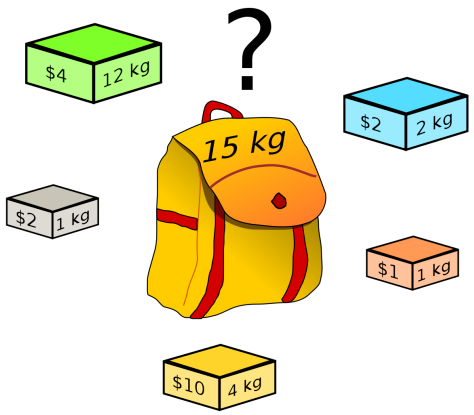
Γίνεται, λοιπόν, αντιληπτό ότι το πρόβλημα που καλούμαστε να λύσουμε είναι η εύρεση ενός αλγορίθμου που θα τοποθετεί σε μια cache συγκεκριμένου μεγέθους L, όσο περισσότερα items μεγέθους L_i γίνεται, που, όμως, είναι πιθανό να τα ζητήσουν οι χρήστες. Αυτό είναι είναι μη-γραμμικό πρόβλημα 0-1 *σακιδίου (knapsack)* και ανήκει στην κατηγορία του **δυναμικού προγραμματισμού** του οποίου τη βασική ιδέα θα εξηγήσουμε παρακάτω.

4.5.2 Δυναμικός Προγραμματισμός

Προκειμένου να καταλάβουμε τον τρόπο με τον οποίο επιλέγονται τα αντικείμενα που μπαίνουν στην cache, κρίνεται απαραίτητη η γνώση της βασικής ιδέας του *δυναμικού προγραμματισμού*.

Στο δυναμικό προγραμματισμό, λοιπόν, η επίλυση ενός προβλήματος γίνεται μέσω της επίλυσης επιμέρους αλληλοεπικαλυπτόμενων υποπροβλημάτων. Καθένα από τα υποπροβλήματα λύνεται μία μόνο φορά και η βέλτιστη λύση αποθηκεύεται και ανακαλείται όταν χρειάζεται.

Η βασική αρχή του δυναμικού προγραμματισμού είναι ότι κάθε υποστρατηγική μιας βέλτιστης στρατηγικής είναι η ίδια βέλτιστη. Για να επιλύσουμε ένα πρόβλημα με δυναμικό προγραμματισμό, το εντάσσουμε σε μια οικογένεια προβλημάτων της ίδιας φύσης και συνδέουμε τις βέλτιστες λύσεις τους με μια αναδρομική σχέση [24]. Ένα διάσημο πρόβλημα δυναμικού προγραμματισμού, του οποίου η λογική είναι παρόμοια με του δικό μας, δηλαδή, η επιλογή αντικειμένων που θα μπουν στη cache όπως θα αναδείξουμε παρακάτω, είναι το λεγόμενο "πρόβλημα του σακιδίου" (knapsack problem).



Σχήμα 4.5: Πρόβλημα σακιδίου [25]

Πιο συγκεκριμένα το πρόβλημα περιλαμβάνει N αντικείμενα $[1,2..,N]$ τα οποία έχουν ένα ορισμένο μέγεθος w και αξία v. Το ζητούμενο είναι να βάλουμε εκείνα τα αντικείμενα στο σακίδιο μεγέθους W, έτσι ώστε να έχει μέσα της, αντικείμενα που αθροιστικά η αξία τους V να είναι η μεγαλύτερη δυνατή.

Μια απλή λύση είναι να δοκιμάζαμε όλους τους πιθανούς συνδυασμούς αντικειμένων και να διαλέγαμε αυτόν που αθροίζει τη μεγαλύτερη αξία V. Αυτό, όμως, θα μας έπαιρνε πάρα πολύ χρόνο.

Μια περιγραφική λύση όπως φαίνεται στο [25] χρησιμοποιώντας δυναμικό προγραμματισμό είναι η παρακάτω :

- Για όλα τα υποσύνολα συνδυασμού αντικειμένων, υπάρχουν δύο εκδοχές για το κάθε αντικείμενο. 1) Το αντικείμενο να συμπεριλαμβάνεται στο υποσύνολο. 2) Το αντικείμενο να μην συμπεριλαμβάνεται στο υποσύνολο.
- Έτσι το μεγαλύτερο άθροισμα σε αξία που μπορούμε να πετύχουμε από n αντικείμενα είναι το μέγιστο από τις παρακάτω 2 περιπτώσεις:
  1. Η μέγιστη τιμή που λαμβάνεται από τα n-1 αντικείμενα με βάρος $\tilde{W}$ (χωρίς να περιλαμβάνεται το αντικείμενο n).
  2. Η τιμή του n αντικειμένου αθροιζόμενη στη μέγιστη τιμή που λαμβάνουμε από τα προηγούμενα n-1 αντικείμενα με βάρος $\tilde{W}$ + w(n) (εδώ περιλαμβάνεται το αντικείμενο n). Αν το $\tilde{W}$ μαζί με το βάρος του n αντικειμένου ξεπερνάει το συνολικό βάρος W που χωράει το σακίδιο, τότε μόνο η πρώτη περίπτωση είναι δυνατή.
- Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να εξετάσουμε όλες οι περιπτώσεις και να βρούμε το συνδυασμό με την μεγαλύτερη αξία.

4.5.3 Caching solution

Συμπεραίνουμε, λοιπόν, ότι το πρόβλημα της επιλογής των αντικειμένων που θα τοποθετήσουμε στην cache είναι παρόμοιο με το παραπάνω πρόβλημα σακιδίου. Η χωρητικότητα της cache είναι L και το κάθε αντικείμενο έχει μέγεθος L_i. Το μόνο που μένει είναι να υπολογίσουμε την αξία του κάθε αντικειμένου. Για το λόγο αυτό και για κάθε item $i \in I$ εισαγάγουμε τον όρο $util(i)$, ο οποίος αντιστοιχεί στο πόσο πιθανό είναι να ζητηθεί ένα αντικείμενο από το σύνολο των χρηστών και υπολογίζεται από τον τύπο:

$$util(i) = \sum_{u \in U} p_u^{req}(i), \qquad i \in I \tag{4.6}$$

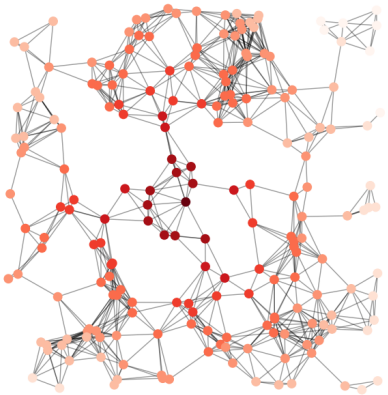
Όπως προαναφέραμε η βέλτιστη τοποθέτηση αντικειμένων είναι ένα πρόβλημα 0-1 *knapsack*. Έτσι, χρησιμοποιούμε τον αλγόριθμο **(DP) FTPAS** όπως αναγράφεται στο ([26], §8.2) για να πάρουμε μια λύση προσεγγιστική κατά 1-ε , ε>0. Ονομάζουμε P τον πίνακα που περιέχει τα στοιχεία της cache [17].

4.6 Δημιουργία Γράφου

4.6.1 RGG

Τα κεφάλαια που ακολουθούν, όπως αναφέραμε, αποτελούν το πρωτότυπο μέρος της διπλωματικής μας. Πιο συγκεκριμένα είναι η προσπάθεια συνδυασμού mobile edge computing (τοπικά κέντρα

δεδομένων) με mobile edge network (μεταφορά δεδομένων μεταξύ συσκευών των χρηστών, Device to Device connection, D2D). Το πρώτο μέρος των τοπικών κέντρων το περιγράψαμε παραπάνω.



Σχήμα 4.6: RGG: Τοποθέτηση χρηστών και ακμών [3]

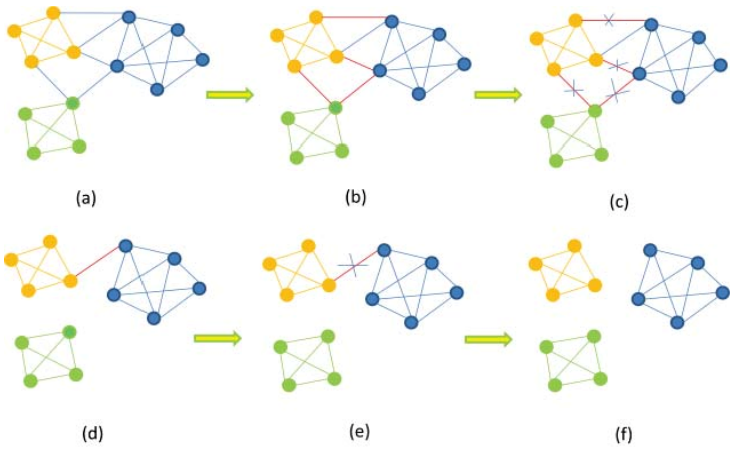
Για να επιτύχουμε **D2D communication** μεταξύ των χρηστών, πρέπει αρχικά να δημιουργήσουμε ένα γράφο. Πιο συγκεκριμένα με χρήση *Random Geometric Graph (RGG)* δημιουργούμε ένα γράφο που οι κόμβοι του αντιστοιχούν στους χρήστες του συστήματος και οι ακμές τους, στις συνδέσεις μεταξύ αυτών. Είναι, δηλαδή, μια προσομοίωση των χρηστών που είναι τοποθετημένοι στο χώρο, με τις ακμές να μας δείχνουν ποιοι από αυτούς θα μπορούσαν να επικοινωνήσουν με ποιους, λόγω της γεωγραφικής τους εγγύτητας.

Για την υλοποίηση του παραπάνω γράφου, έγινε χρήση της βιβλιοθήκης *Networkx* της Python. Μέσα σε αυτή περιλαμβάνεται ο κώδικας δημιουργίας για τη δημιουργία RGG δικτύων[27]. Όπως αναφέραμε στο κεφάλαιο 1, ο αλγόριθμος αυτός τοποθετεί n σε αριθμό κόμβους σε τυχαία σημεία του χώρου και στη συνέχεια τους ενώνει με ακμές αν η ευκλείδια απόσταση δύο κόμβων στο χώρο είναι μικρότερη ή ίση με την *radius* (ακτίνα) που ορίζουμε εμείς.

4.6.2 Κλάδεμα ακμών (edge pruning)

Μέχρι στιγμής, οι ακμές μεταξύ των χρηστών εξαρτώνται μονάχα από τη γεωγραφική τους απόσταση. Ωστόσο, όπως αναφέραμε στο κεφάλαιο **??**, θέλουμε οι συνδέσεις των χρηστών να επηρεάζονται και από τα κοινά τους ενδιαφέροντα, έτσι ώστε, μόνο χρήστες με σχετικά όμοιες προτιμήσεις που βρίσκονται κοντά να έχουν ακμές. Το παραπάνω είναι βασικό κριτήριο, προκειμένου να λειτουργήσει η **επικοινωνία μεταξύ των κινητών συσκευών των χρηστών (D2D)**, καθώς επιβάλλεται να μεγιστοποιήσουμε την πιθανότητα δύο χρήστες συνδεδεμένοι με ακμή να ζητήσουν το ίδιο αντικείμενο και άρα να μπορούν να το ανταλλάξουν μεταξύ τους.

Για το λόγο αυτό δημιουργούμε τον *δείκτη ομοιότητας (similarity number)*. Πρόκειται για έναν αριθμό με βάση τον οποίο δύο χρήστες χαρακτηρίζονται ως όμοιοι ή όχι ακολουθώντας την παρακάτω



Σχήμα 4.7: Κλάδεμα Ακμών [28]

διαδικασία.

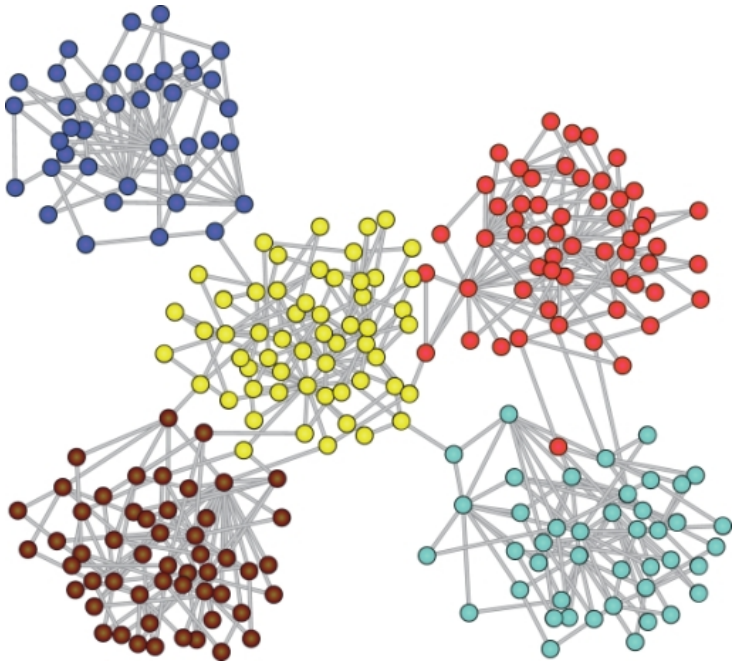
Αρχικά, ταξινομούμε τα p_u^{req} κάθε χρήστη, δηλαδή την πιθανότητα να ζητήσει ένα αντικείμενο, με φθίνουσα σειρά. Αν δύο χρήστες στα πρώτα similarity number ταξινομημένα p_u^{req} στοιχεία τους έχουν έστω και ένα κοινό αντικείμενο, τότε θεωρούνται όμοιοι. Έπειτα κλαδεύουμε τις ακμές των χαρακτηριζόμενων ως "μη όμοιων χρηστών" που κατά τη διάρκεια δημιουργίας του γράφου RGG είχαν δημιουργηθεί ακμές μεταξύ τους. Για τους υπόλοιπους, τα πράγματα παραμένουν ίδια. Στις περισσότερες προσομοιώσεις επιλέγουμε ένα τέτοιο similarity number έτσι ώστε να κλαδεύονται περίπου οι μισές ακμές του αρχικού γράφου.

4.7 Εντοπισμός κοινοτήτων

Στη συνέχεια χρησιμοποιώντας το διαμορφωμένο γράφο, προχωρούμε στον εντοπισμό κοινοτήτων μέσω του αλγορίθμου, modularity maximization, του οποίου τα βασικά χαρακτηριστικά αναφέραμε στο κεφάλαιο 1.4. Έτσι με τη βοήθεια βιβλιοθηκών της Python [29], δημιουργήσαμε τις παρακάτω πυκνές κοινότητες.

Ο αριθμός των κοινοτήτων είναι διαφορετικός κάθε φορά που τρέχουμε την προσομοίωσή μας, διότι εξαρτάται από τον εκάστοτε διαμορφωμένο RGG γράφο.

Έτσι, μετά την εκπόνηση αυτού του σταδίου, όλοι οι χρήστες ανήκουν σε μια κοινότητα και ως αποτέλεσμα η επικοινωνία μεταξύ χρηστών εντός της κοινότητας τους διευκολύνεται σημαντικά. Στην παρούσα διπλωματική κάνουμε την υπόθεση ότι η D2D επικοινωνία πραγματοποιείται μόνο εντός αυτών των κοινοτήτων.



Σχήμα 4.8: Εντοπισμός Κοινοτήτων [29]

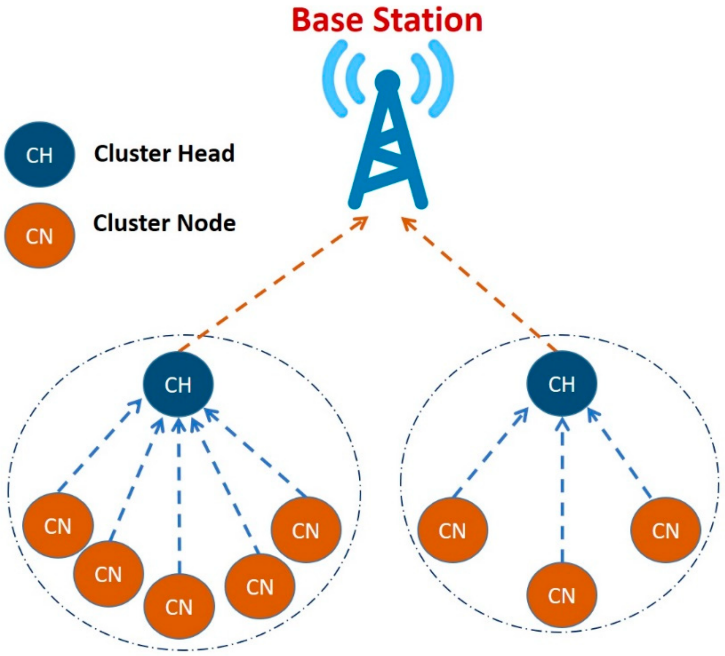
4.8 Επιλογή κεντρικού κόμβου κοινότητας (Clusterhead of Community)

4.8.1 Μέθοδος επιλογής

Η επικοινωνία και η μεταφορά αντικειμένων μεταξύ των κινητών συσκευών των χρηστών (D2D) εντός community γίνεται με τη χρήση ενός clusterhead. Πιο συγκεκριμένα, επιλέγουμε έναν χρήστη ο οποίος είναι δημοφιλής στην κοινότητά του, δηλαδή, έχει το μεγαλύτερο Degree Centrality (περισσότερες ακμές) και τον χρίζουμε Clusterhead. Αφού επιλεγεί, δεσμεύει ένα χώρο στην κινητή του συσκευή ο οποίος χρησιμοποιείται ως cache μόνο για τους χρήστες του community στο οποίο ανήκει. Έτσι, αν κάποιος χρήστης δε βρει ένα αντικείμενο στην cache του τοπικού κέντρου δεδομένων, το αναζητεί στην κοινότητά του και, εφόσον, συνδέεται με τον clusterhead, βλέπει αν περιέχεται στην cache του τελευταίου. Αν το αντικείμενο υπάρχει, τότε ο χρήστης αυτός το αποκτά χωρίς τη χρήση κεντρικών κόμβων (backhaul) και έχουμε αύξηση του cache hit ratio.

Η συγκεκριμένη D2D επικοινωνία, ωστόσο, δεν θεωρείται *peer to peer*, καθώς δεν ανταλλάσσονται αντικείμενα μεταξύ όλων των χρηστών, αλλά μονάχα με τους εκάστοτε clusterhead.

Επίσης, ο λόγος που χρησιμοποιούμε το *Degree Centrality* για την επιλογή clusterhead, είναι επειδή η ανταλλαγή αντικειμένων πραγματοποιείται μόνο αν υπάρχει ακμή μεταξύ χρήστη και clusterhead, δηλαδή μόνο όταν βρίσκονται κοντά και έχουν σχετικά όμοια ενδιαφέροντα. Για το λόγο αυτό, ο χρήστης με τις περισσότερες ακμές και άρα τη μεγαλύτερη πιθανότητα να συνδέεται με κάποιον εντός κοινότητάς του, χρίζεται clusterhead.



Σχήμα 4.9: Εντοπισμός κεντρικού κόμβου κοινότητας [30]

4.8.2 Μέθοδος επιλογής Δευτερεύοντος Clusterhead

Σε ορισμένες προσομοιώσεις τοποθετούμε δύο clusterhead σε κάθε community. Αυτό, διότι μερικοί χρήστες εντός της κοινότητας δεν συνδέονται με τον βασικό clusterhead κι έτσι δεν έχουν πρόσβαση σε καμία cache της κοινότητάς τους. Επομένως, επιλέγουμε έναν δεύτερο clusterhead ο οποίος θα καλύπτει τους χρήστες που δεν έχουν συνδεθεί με τον βασικό και του δεσμεύουμε και πάλι συγκεκριμένο χώρο στο κινητό και τον χρησιμοποιούμε ως cache. Έτσι, κριτήριο επιλογής είναι και πάλι το *Degree Centrality*, αλλά αυτή τη φορά, για τον υπολογισμό, μετράμε μόνο τις ακμές με τους χρήστες που δεν συνδέονται με τον βασικό clusterhead.

Επισημαίνεται ότι ως δευτερεύων clusterhead μπορεί να επιλεγεί και κάποιος που έχει ακμή με το βασικό clusterhead. Επίσης αφού επιλεγεί με το παραπάνω κριτήριο, η cache του χρησιμοποιείται απ'όλους τους χρήστες εντός της κοινότητάς του που συνδέονται με αυτόν.

4.9 Περιεχόμενο της Clusterhead Cache

Όπως αναφέραμε στο 4.8, σε κάθε clusterhead δεσμεύουμε ένα συγκεκριμένο, και κοινό για όλους, χώρο στη μνήμη του κινητού τηλεφώνου τους και τον χρησιμοποιούμε ως cache για τους χρήστες που βρίσκονται εντός του community τους. Προφανώς, το μέγεθος της cache των clusterhead είναι σημαντικά μικρότερο από το μέγεθος της cache του τοπικού κέντρου δεδομένων. Επίσης, στις περιπτώσεις που χρησιμοποιούμε δύο clusterhead σε κάθε community, παρόλο που το κριτήριο επιλογής τους διαφέρει, το κριτήριο τοποθέτησης αντικειμένων στις cache τους είναι όμοιο.

Αρχικά, για την τοποθέτηση αντικειμένων χρησιμοποιούμε τον ίδιο αλγόριθμο δυναμικού προ-

γραμματισμού *FTPAS* όπως στο 4.5.3. Όμως, αυτό που αλλάζει είναι η χωρητικότητα της clusterhead cache Ĺ καθώς και η αξία των αντικειμένων, ενώ το μέγεθος L_i αυτών παραμένει ίδιο.

Η αξία κάθε αντικειμένου για κάθε clusterhead που χρησιμοποιούμε είναι διαφορετική. Αυτό, διότι, για την αξία τους υπολογίζονται μόνο οι χρήστες $u \in U$ που συνδέονται με τον εκάστοτε clusterhead και ανήκουν στο community του. Επίσης, η αξία ενός αντικειμένου που βρίσκεται ήδη στην cache του τοπικού κέντρου δεδομένων (P), **μηδενίζεται**, καθώς οι χρήστες πρώτα αναζητούν στα τοπικά Base station για ένα αντικείμενο και στη συνέχεια καταφεύγουν στους clusterhead. Επομένως, δεν υπάρχει λόγος ύπαρξης ενός αντικειμένου και στα δύο είδη cache της προσομοίωσής μας. Πιο συγκεκριμένα για έναν clusterhead A ενός community B η αξία ενός αντικειμένου $i \in I$, $util_A(i)$ υπολογίζεται όπως παρακάτω:

- Για $i \in P$, $util_A(i) = 0$
- Για $i \notin P$, $util_A(i) = \sum_u p_u^{req}(i)$, για $u \in B$ και $u \rightarrow A$
  όπου $u \rightarrow A$ σημαίνει ότι ο u έχει ακμή με τον clusterhead A.

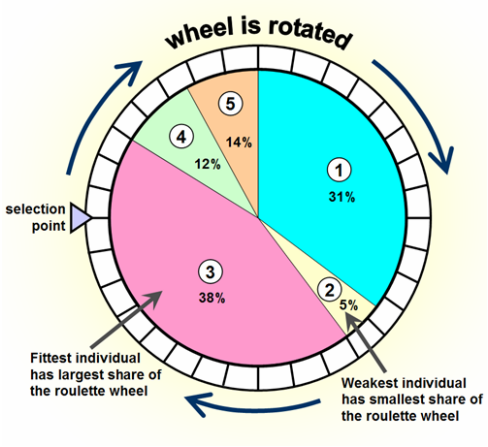
Τέλος, τοποθετώντας στον *FTPAS* αλγόριθμο, για κάθε clusterhead ξεχωριστά, το μέγεθος της cache τους (Ĺ), καθώς και το μέγεθος και την αξία των αντικειμένων $(L_i, util_A(i))$, λαμβάνουμε τα αντικείμενα που θα γίνουν cache στους clusterhead κάθε community. Συμβολίζουμε με P[A] τον πίνακα αντικειμένων της cache του κάθε clusterhead A.

4.10 Αιτήματα χρηστών

Στο κεφάλαιο 4.4, υπολογίσαμε την πιθανότητα $p_u^{req}(i)$ ένας χρήστης $u \in U$ να ζητήσει ένα αντικείμενο $i \in I$. Τώρα πρέπει να καθορίσουμε ποια θα είναι τα R σε αριθμό τελικά αιτήματα αντικειμένων που θα πραγματοποιήσει ο κάθε χρήστης. Στην προσομοίωσή μας θεωρήσαμε και εκτελέσαμε επαναλήψεις για δύο είδη αιτημάτων:

1. Ένας χρήστης $u \in U$ έχει σαν αιτήματα τα R στοιχεία με την υψηλότερη τιμή p_u^{req}. Δηλαδή, ζητάει τα R αντικείμενα που είναι περισσότερο πιθανό να ζητήσει.

2. Εδώ ένας χρήστης $u \in U$ επιλέγει αντικείμενα χρησιμοποιώντας τη λεγόμενη "επιλογή της ρουλέτας" (roullete selection). Πρόκειται για ένα πιθανοτικό μοντέλο επιλογής αντικειμένων που όμως σέβεται την αναλογία πιθανοτήτων όπως έχει οριστεί από το p_u^{req}. Δηλαδή, ένα αντικείμενο i με διπλάσια τιμή $p_u^{req}(i)$ από αυτή ενός άλλου αντικειμένου ί, p_u^{req}(ί) , έχει επίσης διπλάσιες πιθανότητες να επιλεγεί ως τελικό αίτημα ενός χρήστη [31]. Θα μπορούσε κανείς να φανταστεί την ρουλέτα ενός καζίνο με τη μόνο διαφορά ότι κάθε αντικείμενο δεν επιλέγεται ισοπίθανα, αλλά εξαρτάται από την τιμή $p_u^{req}(i)$ που έχει, όπως φαίνεται στην εικόνα 4.10. Κρίνουμε πως το *roullete selection* ανταποκρίνεται περισσότερο στα αιτήματα των χρηστών σε πραγματικό επίπεδο.



Σχήμα 4.10: Καθορισμός αιτημάτων μέσω του τροχού της ρουλέτας [32]

4.11 Ευριστικός Αλγόριθμος (Heuristic Algorithm)

Παρακάτω περιγράφουμε εν συντομία τη βασική ιδέα του μέχρι τώρα ευριστικού αλγορίθμου μας, πριν περάσουμε στην αξιολόγησή της προσομοίωσής μας.

Data: $f_u(j)$,$f_i(j)$, w_u για κάθε χρήστη u.
Result: Οι πίνακες *P, P[A]*, οι κοινότητες, ποιοι χρήστες είναι *clusterhead* και τα *R* αντικείμενα που ζητάει κάθε χρήστης για τις δύο περιπτώσεις.
while *i in I,* $u \in U, j \in M$ **do**
  Υπολογισμός $f_u(j)$, $f_i(j)$, w_u *(4.2.1) ;*
end
Μέσω των παραπάνω, υπολογίζουμε τα p_u^{pref}, $\forall\ u \in U$ *(4.3);*
Με συνδυασμό των p_u^{rec},p_u^{pref}, w_u, *υπολογίζουμε τα* p_u^{req} *(4.4);*
while $i \in I$ **do**
  Υπολογισμός $util(i)$ *(4.5) ;*
end
Υπολογισμός πίνακα P *μέσω FTPAS (4.5);*
Δημιουργία RGG (n,radius) *γράφου (4.6.1);*
Κλάδεμα ακμών (4.6.2);
Δημιουργία κοινοτήτων μέσω αλγορίθμου modularity maximization (4.7);
Εντοπισμός clusterhead (4.8);
Υπολογισμός επιμέρους $util_A(i)$ *και έπειτα υπολογισμός των πινάκων* $P[A]$ *για κάθε clusterhead A (4.9);*
Καθορισμός αιτημάτων των χρηστών για τις δύο περιπτώσεις (4.10);

Algorithm 1: Ευριστικός Αλγόριθμος

5 Αξιολόγηση Προσομοίωσης

5.1 Περιγραφή Κεφαλαίου

Στο κεφάλαιο αυτό, θα περιγράψουμε, αρχικά, τη διαδικασία που ακολουθεί ένας χρήστης u, προκειμένου να ικανοποιήσει ένα αίτημά του για ένα αντικείμενο i τοπικά, ψάχνοντας δηλαδή, στην cache του τοπικού κέντρου δεδομένων και στην cache των clusterhead της κοινότητάς του. Στη συνέχεια, περνάμε στη διαδικασία αξιολόγησης της πειραματικής διάταξης που έχουμε αναπτύξει. Πιο, συγκεκριμένα, υπολογίζουμε τα ποσοστά επιτυχίας των cache του συστήματος, αλλάζοντας τις τιμές των παραμέτρων που συμμετέχουν στον αλγόριθμο.

5.2 Μεθοδολογία αξιολόγησης

5.2.1 Αλγόριθμος

Στο σημείο αυτό περνάμε στο τελικό στάδιο της προσομοίωσής μας, αξιολογώντας τον αλγόριθμο που φτιάξαμε. Ουσιαστικά, θέλουμε να συμπεράνουμε εάν και κατά πόσο ο τρόπος με τον οποίο δομήσαμε τον αλγόριθμό μας αποδίδει. Δηλαδή, εάν οι χρήστες βρίσκουν τα αντικείμενα που επιθυμούν τοπικά, είτε στα κέντρα δεδομένων, είτε στην μεταξύ κοινότητας επικοινωνίας τους. Η δεύτερη περίπτωση (D2D communication) μας ενδιαφέρει περισσότερο, διότι αποτελεί και την καινοτομία της διπλωματικής.

Πιο συγκεκριμένα, ένας χρήστης, έχοντας ολοκληρώσει τα βήματα του ευριστικού αλγορίθμου, γνωρίζει σε ποιο community ανήκει και ποια αντικείμενα θα ζητήσει από το σύστημα. Ταυτόχρονα όλες οι cache της προσομοίωσης είναι γεμάτες αντικείμενα, περιμένοντας να ικανοποιήσουν όσο το δυνατόν περισσότερα αιτήματα χρηστών.

Επομένως, όταν ένας χρήστης $u \in U$ θελήσει ένα αντικείμενο $i \in I$ ακολουθεί την παρακάτω διαδικασία :

1. Αρχικά ψάχνει αν το αντικείμενο i βρίσκεται στην cache του κέντρου δεδομένων $\mathbf{P}$.

- Αν υπάρχει, τότε το αίτημά του ικανοποιείται και η περίπτωση θεωρείται **cache hit**.
- Αν δεν υπάρχει, τότε το αναζητά στο community του.

2. Ο χρήστης μεταβαίνει στην κοινότητά του και ελέγχει αν έχει ακμή με το βασικό clusterhead.

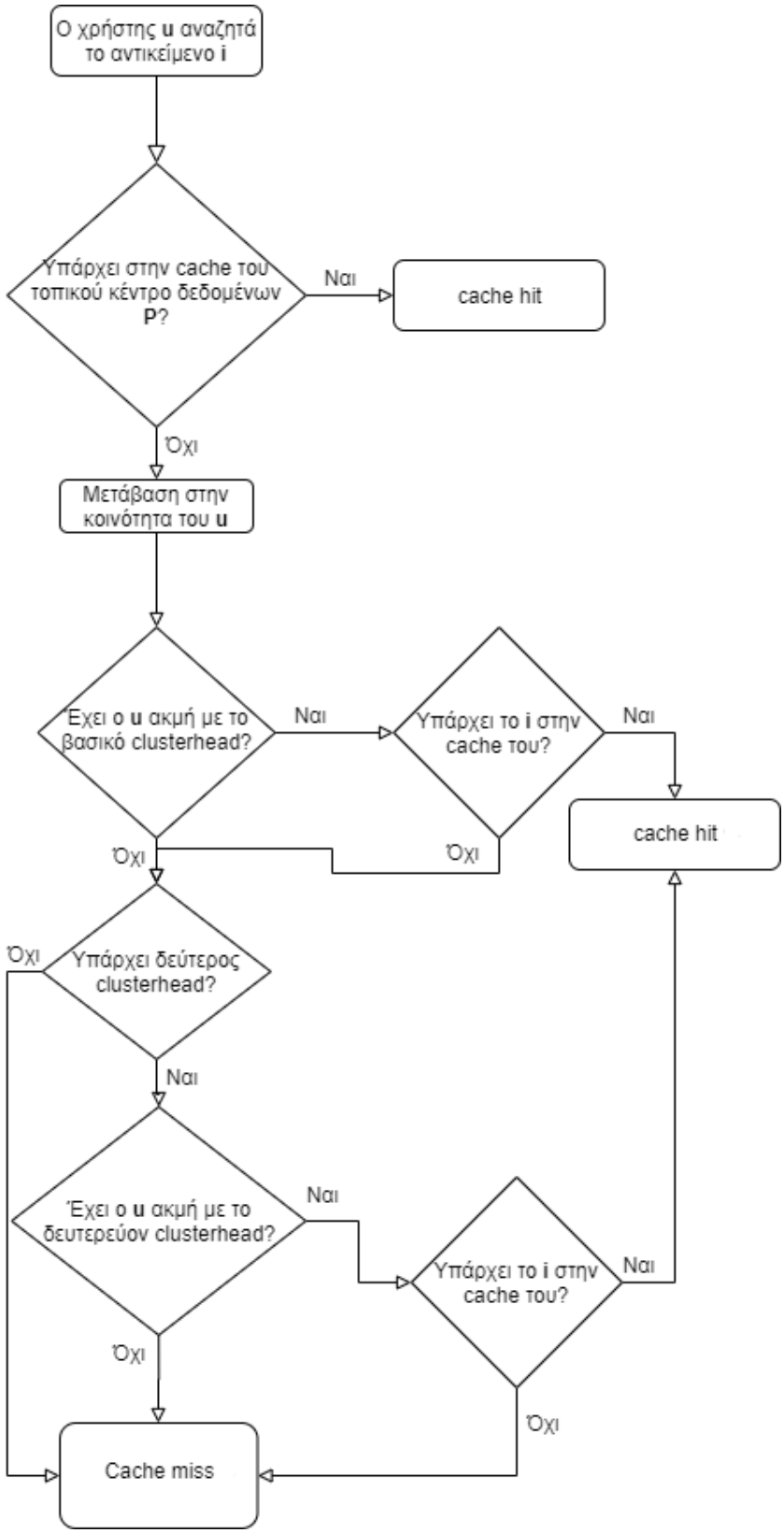
   - Αν υπάρχει η ακμή, τότε ο χρήστης ψάχνει για το αντικείμενο i στην cache του βασικού clusterhead.
     - Αν το βρει, τότε επιτυγχάνεται επικοινωνία από συσκευή σε συσκευή (D2D), το αίτημα του χρήστη ικανοποιείται και η περίπτωση θεωρείται **cache hit**.
     - Αν δεν το βρει, τότε περνάει στο βήμα 3.
   - Αν δεν υπάρχει ακμή, τότε περνάει στο βήμα 3.

3. Ο χρήστης ελέγχει αν έχει ακμή με το δευτερεύον clusterhead.

   - Αν υπάρχει η ακμή, τότε ο χρήστης ψάχνει για το αντικείμενο i στην cache του δευτερεύοντος clusterhead.
     - Αν το βρει, τότε επιτυγχάνεται επικοινωνία από συσκευή σε συσκευή (D2D), το αίτημα το χρήστη ικανοποιείται και η περίπτωση θεωρείται **cache hit ratio**.
     - Αν και πάλι δεν το βρει τότε η περίπτωση θεωρείται **cache miss** και για τη ικανοποίηση του αιτήματος χρησιμοποιούνται κεντρικές δομές δικτύου (*backhaul*).
   - Αν δεν υπάρχει ακμή, τότε η περίπτωση θεωρείται **cache miss** και για τη ικανοποίηση του αιτήματος χρησιμοποιείται το *backhaul*.

Σημειώνεται ότι στον παρακάτω αλγόριθμο θεωρούμε ύπαρξη δύο clusterhead παρόλο που οι μετρήσεις μας πραγματοποιήθηκαν για ύπαρξη ενός ή δύο clusterhead. Στις περιπτώσεις που υπάρχει μονάχα ένας clusterhead, αγνοούμε το βήμα 3 και πηγαίνουμε σε περίπτωση *cache miss ratio*.

Τα παραπάνω, απεικονίζονται και στο παρακάτω διάγραμμα ροής.



Σχήμα 5.1: Διαδικασία εύρεσης ενός αντικειμένου *i* στις cache του συστήματος

5.2.2 Σύνολο δεδομένων (DataSet)

Για την προσομοίωση χρησιμοποιήσαμε συνθετικά σύνολα δεδομένων προκειμένου να καθορίσουμε το περιεχόμενο των αντικειμένων, καθώς και το προφίλ των χρηστών. Αυτό μας δίνει τη δυνατότητα να έχουμε περισσότερο έλεγχο του πειράματος χωρίς, όμως, να απομακρυνόμαστε από ρεαλιστικές συνθήκες.

Πιο συγκεκριμένα, για τις τιμές f_u, f_i, w_u πήραμε τυχαίες τιμές σε ένα ορισμένο και κοινό για όλους διάστημα [0,1] που ακολουθούν την ομοιόμορφη κατανομή. Επίσης οι θεματικές κατηγορίες **M** είναι κοινές και ίσες με 8 τόσο για τους χρήστες u, όσο και για τα αντικείμενα i.

5.2.3 Προεπιλεγμένες παράμετροι (Default Parameters)

Για την πειραματική διαδικασία επιλέξαμε ορισμένες τιμές παραμέτρων που θα χρησιμοποιούμε ως βασικές. Ωστόσο, προκειμένου να διαπιστώσουμε την προσαρμοστικότητα του αλγορίθμου στην εκάστοτε πειραματική διάταξη, σε καθεμία από τις παρακάτω ενότητες μεταβάλλουμε κάθε φορά τη τιμή μιας παραμέτρου, κρατώντας τις υπόλοιπες σταθερές. Οι βασικές τιμές των παραμέτρων είναι οι παρακάτω:

- Users $|U| = 100$
- Items $|I| = 500$
- Όπως προαναφέραμε, το μέγεθος του αντικειμένου i, L_i ακολουθεί την ομοιόμορφη κατανομή στο διάστημα [1,10]. Στη συνέχεια υπολογίζεται το μέσο μέγεθος $L_{average}$ ενός item.
- Το μέγεθος cache του τοπικού κέντρου δεδομένων είναι $L = 40 * L_{average}$
- Το σταθερό και ίσο για όλους τους clusterhead, μέγεθος της cache τους είναι $\acute{L} = 6 * L_{average}$
- *Similarity Number* = 20
- Αριθμός αιτημάτων κάθε χρήστη, $R = 5$

5.3 Αποτελέσματα προσομοιώσεων

5.3.1 Γενική περιγραφή και παραδοχές

Όπως αναφέραμε, πραγματοποιούνται μετρήσεις για διαφορετικές τιμές των παραμέτρων του πειράματος, μετρώντας κάθε φορά το cache hit ratio που αυτές επιφέρουν. Πιο συγκεκριμένα,

μετράμε το συνολικό cache hit ratio (D2D + Base Station) το οποίο ονομάζουμε **overall cache hit ratio**. Επίσης, παραθέτουμε ξεχωριστά τα *cache hit ratio* για mobile edge computing (τοπικά κέντρα δεδομένων) το οποίο ορίζεται ως **base station cache hit ratio** καθώς και για mobile edge Network (D2D communication) που ορίζεται ως **peer caching hit ratio**.

Επιπρόσθετα παραθέτουμε αποτελέσματα για τον αριθμό των φορών που έγινε χρήση του βασικού και του δευτερεύοντος (αν υπάρχει) clusterhead. Τα ονομάζουμε **use of basic clusterhead** και **use of second clusterhead** αντίστοιχα. Τέλος, όταν κρίνουμε ότι υπάρχει λόγος, θα αναγράφουμε και τον αριθμό των στοιχείων που περιέχει η cache του τοπικού κέντρου δεδομένων ως **Items in P**.

Παράλληλα, παραθέτουμε τα αποτελέσματα για τα δύο είδη αιτημάτων των χρηστών όπως αναφέραμε στο 4.10. Τα αιτήματα που αντιστοιχούν στα πρώτα R στοιχεία του p_u^{req} κάθε χρήστη $u \in U$ τα συναντάμε με τον όρο **ντετερμινιστικά αιτήματα (deterministic requests)**, ενώ τα αιτήματα μέσω roullete selection με τον όρο **πιθανολογικά αιτήματα (probabilistic requests)**. Επιπρόσθετα, αναφέρουμε αποτελέσματα άλλοτε με χρήση ενός Clusterhead και άλλοτε με χρήση δύο Clusterhead.

Τα αποτελέσματα που αναγράφονται είναι προϊόντα **μέσου όρου 25 επαναλήψεων**, προκειμένου να εξομαλυνθεί ο παράγοντας της τυχαιότητας.

Τέλος, για την περαιτέρω κατανόηση των παραμέτρων και των αποτελεσμάτων που αυτές επιφέρουν στην προσομοίωση, αναφέρουμε πως για τις σταθερές μας τιμές όπως αναγράφονται στο 5.2.3, όλες οι caches των βασικών clusterhead είναι κατά μέσο όρο προσβάσιμες στο **59.4%** του συνολικού αριθμού των χρηστών, ενώ οι caches των δευτερευόντων clusterhead, όταν αυτοί υπάρχουν, είναι προσβάσιμες στο **46.9%** του συνολικού αριθμού των χρηστών. Ως αποτέλεσμα, περίπου το **90%** των χρηστών έχει πρόσβαση σε τουλάχιστον μια cache ενός clusterhead της κοινότητάς του.

5.4 Αριθμός Χρηστών (Network Size)

5.4.1 Γενική περιγραφή

Η πρώτη παράμετρος που τροποποιούμε, προκειμένου να δούμε τα αποτελέσματα της προσομοίωσης, είναι ο αριθμός των χρηστών *U (Network Size)*. Ορίζοντας τις υπόλοιπες παραμέτρους όπως στο 5.2.3, οι χρήστες παίρνουν τις τιμές:

- Users = 100, 150, 200, 250, 300, 350, 400

Για τις παραπάνω περιπτώσεις ο αριθμός των αντικειμένων που περιέχει ο πίνακας **P**, δηλαδή η cache του τοπικού κέντρου δεδομένων, είναι σχεδόν όμοιος για όλες τις τιμές των χρηστών και ο μέσος όρος του ισούται με **67.4** αντικείμενα.

5.4.2 Ένας Clusterhead ανά κοινότητα

Παρακάτω, λοιπόν, φαίνονται τα πρώτα αποτελέσματα όταν σε κάθε κοινότητα υπάρχει **ένας clusterhead** για **deterministic** και **probabilistic** αιτήματα χρηστών. Τα αποτελέσματα παρα-

τίθενται τόσο σε μορφή πινάκων, όσο και διαγραμμάτων.

1 Clusterhead							
Deterministic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
Overall cache hit ratio (%)	65.49	63.48	61.15	59.89	57.94	58.33	59.3
Base station cache hit ratio (%)	58.11	52.29	56.01	55.05	53.72	54.33	55.19
peer caching hit ratio(%)	7.38	6.20	5.14	4.84	4.22	4.00	4.11
use of basic clusterhead (#)	36.88	46.8	51.4	60.88	63.32	71.4	82.88

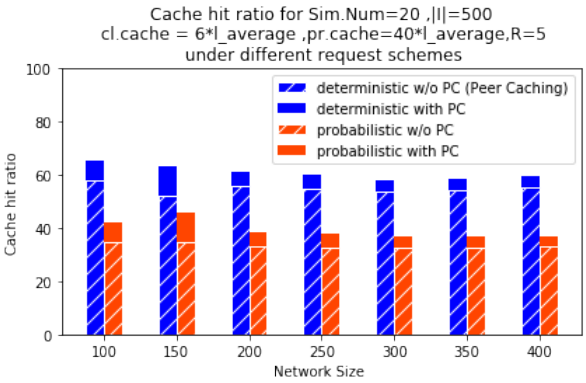
Πίνακας 5.1: Network Size: 1 Clusterhead for Deterministic Requests

1 Clusterhead							
Probabilistic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
Overall cache hit ratio (%)	39.5	38.42	36.66	35.93	35.53	35.27	35.56
Base station cache hit ratio (%)	34.8	34.53	33.34	32.83	32.86	32.72	32.93
peer caching hit ratio(%)	4.69	3.89	3.31	3.1	2.67	2.55	2.62
use of basic clusterhead (#)	23.44	29.44	33.12	38.96	40.08	45.76	53.12

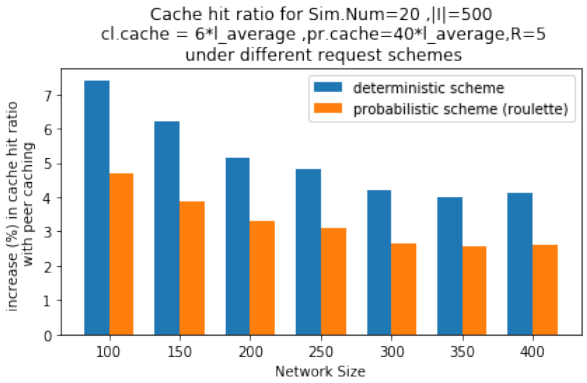
Πίνακας 5.2: Network Size: 1 Clusterhead for Probabilistic Requests

Βλέπουμε πως σε όλες τις περιπτώσεις, πάνω από τα μισά αιτήματα χρηστών ικανοποιούνται τοπικά χωρίς τη χρήστη backhaul δικτύων, το οποίο κρίνεται ιδιαίτερα θετικό. Παράλληλα, οι τιμές των cache hit ratio τόσο για deterministic, όσο και για probabilistic αιτήματα, μειώνονται με μικρό ρυθμό όσο μεγαλώνει ο αριθμός χρηστών. Επίσης, παρά το ότι ο αριθμός των φορών που χρησιμοποιούμε τους clusterhead μεγαλώνει, αυξάνεται και ο συνολικός αριθμός των αιτημάτων λόγω του μεγαλύτερου δικτύου, άρα το cache hit ratio συνολικά πέφτει.

Παρακάτω φαίνονται τα αντίστοιχα διαγράμματα. Το πρώτο αφορά τη συνολική εικόνα της προσομοίωσης και το δεύτερο αφορά αποκλειστικά το cache hit ratio with peer caching, δηλαδή το αποτέλεσμα της D2D επικοινωνίας:



Σχήμα 5.2: 1 Clusterhead Overall Results



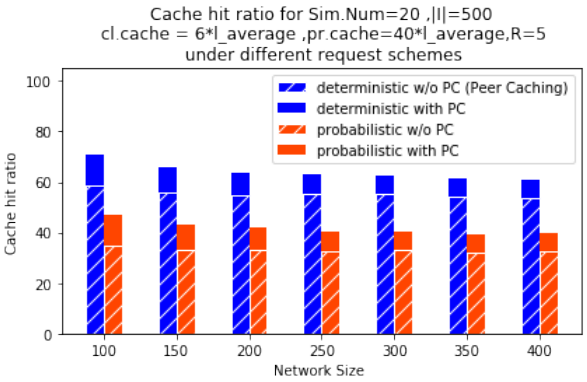
Σχήμα 5.3: Peer caching hit ratio

5.4.3 Δύο clusterhead ανά κοινότητα

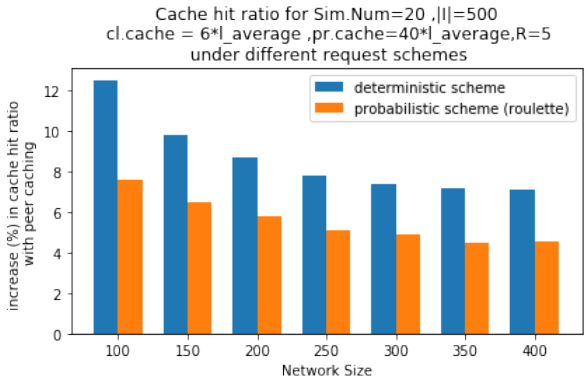
Στη συνέχεια πραγματοποιούμε τις ίδιες μετρήσεις για 2 clusterhead σε κάθε κοινότητα και αναγράφουμε επιπρόσθετα στις μετρήσεις πόσες φορές πραγματοποιήθηκε η χρήση αυτού.

2 Clusterhead							
Deterministic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
Overall cache hit ratio (%)	70.98	65.85	63.45	63.19	62.52	61.51	60.64
Base station cache hit ratio (%)	58.5	56.02	54.73	55.35	55.13	54.31	53.31
peer caching hit ratio(%)	12.47	9.83	8.72	7.83	7.38	7.2	7.13
use of basic clusterhead (#)	37.92	44.08	51.64	56.76	65.44	74.32	82.12
use of second clsuterhead(#)	24.44	29.68	35.56	41.16	45.32	52.4	60.48
Probabilistic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
Overall cache hit ratio (%)	42.32	39.87	39.15	37.8	37.9	36.8	39.09
Base station cache hit ratio (%)	34.7	33.4	33.34	32.72	33.04	32.31	32.56
peer caching hit ratio(%)	7.62	6.46	5.8	5.09	4.86	4.51	4.53
use of basic clusterhead (#)	23.24	29.36	34.96	36.2	41.88	45.68	51.88
use of second clsuterhead(#)	14.88	19.12	23.08	27.4	31.04	33.48	38.64

Πίνακας 5.3: Network Size: 2 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.4: 2 Clusterhead Overall Results



Σχήμα 5.5: Peer caching hit ratio

Παρατηρούμε πως με τη χρήση δύο clusterhead έχουμε μια συνολική αύξηση cache hit ration κατά **4-6%**. Αυτή η αύξηση οφείλεται αποκλειστικά στη χρήση του δεύτερου clusterhead, αφού τώρα πολλοί χρήστες έχουν πρόσβαση σε clusterhead cache που με έναν clusterhead δεν είχαν. Για το λόγο αυτό βλέπουμε και αύξηση μόνο στο *peer caching hit ratio*, ενώ το *base station cache hit ratio* παραμένει στα ίδια επίπεδα με πριν. Τέλος, βλέπουμε πως η χρήση του δεύτερου clusterhead είναι μικρότερη από αυτή του βασικού clusterhead. Διαφορά η οποία οφείλεται στο πλήθος των χρηστών που εξυπηρετεί ο καθένας, καθώς και στο γεγονός ότι ένας χρήστης που βρίσκει το αντικείμενό του στον πρώτο clusterhead (που θα ψάξει αρχικά), δε θα κάνει καθόλου χρήση του δεύτερου clusterhead. Το παραπάνω σχόλιο ισχύει για όλα τα κεφάλαια που ακολουθούν.

5.4.4 Παρατηρήσεις

Όπως προαναφέραμε, τα περισσότερα αιτήματα των χρηστών ικανοποιούνται τοπικά, χωρίς χρήση κεντρικών δικτύων. Επίσης όσο μικρότερος είναι ο αριθμός των χρηστών δικτύου, τόσο μεγαλύτερη επιτυχία έχουμε. Αυτό οφείλεται στο ότι για λιγότερους χρήστες, καλούμαστε να ικανοποιήσουμε λιγότερα αιτήματα. Έτσι, οι caches συγκεκριμένου μεγέθους που έχουμε τόσο στα τοπικά κέντρα δεδομένων όσο και στις κοινότητες μπορούν να οργανωθούν καλύτερα προκειμένου να καλύψουν τα αιτήματα αυτά. Ωστόσο, η διαφορά απόδοσης δεν είναι πολύ μεγάλη.

Παράλληλα παρατηρούμε μεγάλη διαφορά στην απόδοση μεταξύ *Deterministic* και *Probabilistic* αιτημάτων. Αυτό συμβαίνει, διότι οι caches του συστήματός μας οργανώνονται με βάση τα $\boldsymbol{p_u^{req}}$ των χρηστών, δηλαδή τα *Deterministic* αιτήματα που παρατηρούμε μεγαλύτερη επιτυχία. Αντίθετα, με τα *Probabilistic* αιτήματα, παρόλο που ανταποκρίνονται περισσότερο στην πραγματικότητα, ο παράγοντας τυχαιότητα παίζει σημαντικό ρόλο και άρα πολλά από τα τελικά αιτήματα αποκλίνουν από τις προβλέψεις που έχουν γίνει από τις cache.

Τέλος, με τη χρήση δύο clusterhead ανά community, έχουμε σημαντική βελτίωση αποτελεσμάτων, χρησιμοποιώντας λύση που συνδυάζει (κυρίως) "software" και "hardware". Δηλαδή, κάνουμε όλη την έρευνα με τους γράφους για να εντοπίσουμε τις κοινότητες και τα καταλληλότερα άτομα σε αυτές να γίνουν clusterhead και στη συνέχεια τους δεσμεύουμε μια μικρή σχετικά μνήμη στο κινητό τους τηλέφωνο που τη χρησιμοποιούμε ως cache και τοποθετούμε σε αυτή αντικείμενα που είναι πολύ πιθανό να ζητηθούν από χρήστες της κοινότητάς τους.

5.5 Αριθμός Αντικειμένων (Number of Items)

5.5.1 Γενική περιγραφή

Στη συνέχεια βλέπουμε τα αποτελέσματα της προσομοίωσής μας, αλλάζοντας τον αριθμό των αντικειμένων που συμμετέχουν στο σύστημα και διατηρώντας τις υπόλοιπες τιμές όπως στο 5.2.3. Επομένως, στις παρακάτω περιπτώσεις οι χρήστες θα έχουν μια πολύ μεγαλύτερη "γκάμα" αντικειμένων που μπορούν να επιλέξουν και να ζητήσουν, καθώς το πλήθος αυτών παίρνει τις παρακάτω τιμές:

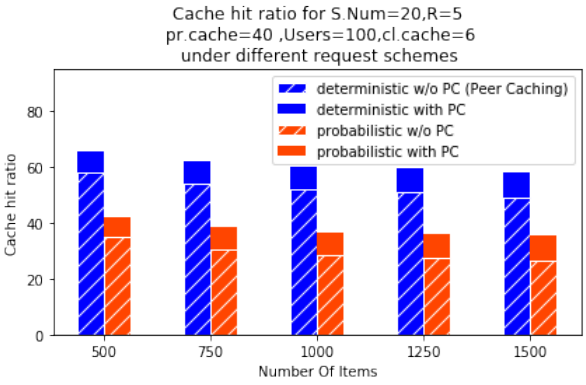
- Items = 500, 750, 1000, 1250, 1500

Και πάλι ο αριθμός των αντικειμένων που περιέχει ο πίνακας **P**, είναι σχεδόν όμοιος για όλες τις τιμές των αντικειμένων και ο μέσος όρος του ισούται με **63.37** αντικείμενα.

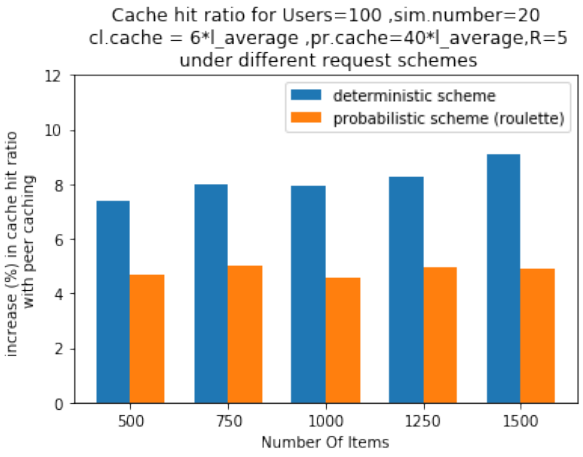
5.5.2 Ένας clusterhead ανά κοινότητα

1 Clusterhead					
Deterministic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
Overall cache hit ratio (%)	65.69	62	60.1	59.5	58.21
Base station cache hit ratio (%)	58.11	54	52.15	51.21	49.13
peer caching hit ratio(%)	7.38	7.99	7.94	8.29	9.07
use of basic clusterhead (#)	36.88	39.96	39.72	41.48	45.36
Probabilistic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
Overall cache hit ratio (%)	39.5	35.7	33.02	32.6	31.46
Base station cache hit ratio (%)	34.8	30.71	28.46	27.61	26.58
peer caching hit ratio(%)	4.69	4.99	4.57	4.98	4.89
use of basic clusterhead (#)	23.44	24.96	22.84	25	24.44

Πίνακας 5.4: Items: 1 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.6: 1 Clusterhead Overall Results



Σχήμα 5.7: Peer caching hit ratio

Βλέπουμε πως και στις δύο περιπτώσεις με την άνοδο του αριθμού των αντικειμένων, πέφτει ελαφρώς η απόδοση του αλγορίθμου μας. Παρόλα αυτα, η μείωση αυτή οφείλεται στο *base station cache hit ratio* και όχι στο *peer caching hit ratio*, το οποίο αυξάνεται ελαφρώς, γεγονός που αποδεικνύει ότι η D2D επικοινωνία που επιδιώκουμε είναι επιτυχής και σταθερή.

5.5.3 Δύο Clusterhead ανά κοινότητα

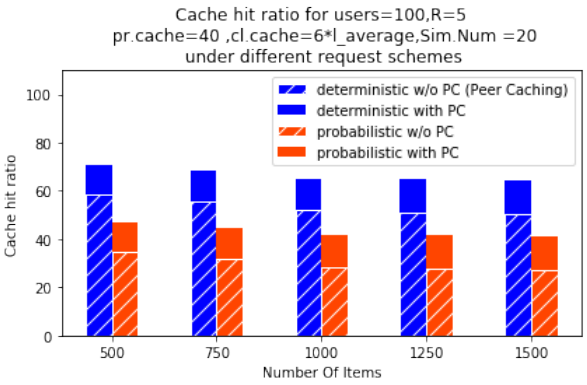
Στη συνέχεια πραγματοποιούμε τις ίδιες μετρήσεις με πριν, τοποθετώντας δύο clusterhead σε κάθε community.

2 Clusterhead					
Deterministic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
Overall cache hit ratio (%)	70.98	68.22	64.74	64.86	64.11
Base station cache hit ratio (%)	58.5	55.66	51.88	51.11	50.19
peer caching hit ratio(%)	12.47	12.57	12.86	13.75	13.92
use of basic clusterhead (#)	37.92	39.96	39.72	41.48	45.36
use of second clsuterhead(#)	24.44	19.12	23.08	27.4	31.04

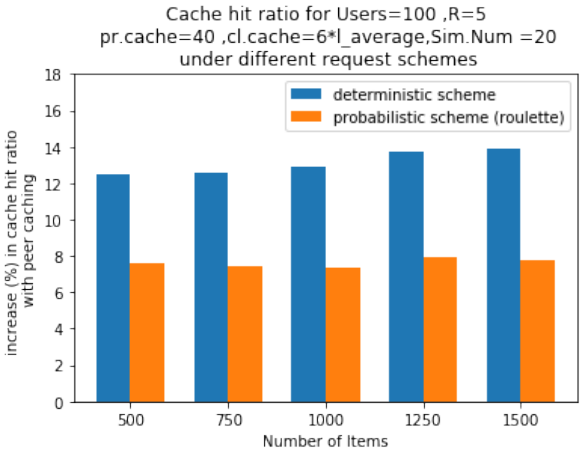
Πίνακας 5.5: Items: 2 Clusterhead for Deterministic Requests

2 Clusterhead					
Probabilistic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
Overall cache hit ratio (%)	42.32	39.3	35.95	35.54	34.73
Base station cache hit ratio (%)	34.7	31.86	28.57	27.66	26.95
peer caching hit ratio(%)	7.62	7.44	7.38	7.89	7.78
use of basic clusterhead (#)	23.24	22.56	21.56	24	24.28
use of second clsuterhead(#)	14.88	15.04	15.52	15.6	14.6

Πίνακας 5.6: Items: 2 Clusterhead for Probabilistic Requests



Σχήμα 5.8: 2 Clusterhead Overall Results



Σχήμα 5.9: Peer caching hit ratio

5.5.4 Παρατηρήσεις

Γενικά και στις δύο περιπτώσεις παρατηρούμε πως με την αύξηση του αριθμού των αντικειμένων το συνολικό *cache hit ratio* μειώνεται ελαφρώς. Ωστόσο, το *peer caching hit ratio*, δηλαδή η επικοινωνία των χρηστών εντός community αυξάνεται. Αυτό είναι ιδιαίτερα θετικό, διότι αποδεικνύει πως ο τρόπος με τον οποίον οργανώνεται η cache των clusterhead είναι ιδιαίτερα αποτελεσματικός και λαμβάνει υπόψη τα p_u^{req} μόνο των χρηστών της κοινότητάς του. Αυτός είναι και ο λόγος που η clusterhead cache δεν "αποπροσανατολίζεται" με την αύξηση των αντικειμένων όπως συμβαίνει με την cache του τοπικού κέντρου δεδομένων **P**, της οποίας η απόδοση *base station cache hit ratio* πέφτει. Αυτό, διότι η P οργανώνεται με βάση τα αιτήματα χρηστών όλου του δικτύου και άρα όσο αυξάνονται τα αντικείμενα, τόσο μοιράζονται σε μεγαλύτερο εύρος αντικειμένων τα αιτήματα των

χρηστών και άρα, λόγω του ορισμένου μεγέθους της, η cache δεν μπορεί πλέον να συμπεριλάβει στη μνήμη της πολλά από αυτά τα αιτήματα.

Ταυτόχρονα, η διαφορά στην επιτυχία *Deterministic*, *Probabilistic* αιτημάτων καθώς και η διαφορά στο πόσο χρησιμοποιούνται οι βασικοί και οι δευτερεύοντες clusterhead εξηγείται όπως στο 5.4.4.

Επομένως, η D2D επικοινωνία εντός community και άρα η αποτελεσματικότητα του αλγορίθμου και της καινοτομίας μας είναι σταθερή και σχετικά ανεξάρτητη από τον αριθμό των αντικειμένων που υπάρχουν στο σύστημα, πράγμα σημαντικό για τη μελλοντική χρήση της σε πραγματικές συνθήκες.

5.6 Αριθμός αιτημάτων χρηστών R

5.6.1 Γενική περιγραφή

Στο επόμενο κομμάτι της διπλωματικής πραγματοποιούμε για άλλη μια φορά μετρήσεις, αλλάζοντας αυτή τη φορά τον αριθμό των αιτημάτων **R** που πραγματοποιούν οι χρήστες και κρατώντας τις τιμές των υπόλοιπων παραμέτρων σταθερές. Περιμένουμε πως όσο μεγαλύτερος είναι ο αριθμός των αιτημάτων των χρηστών, τόσο δυσκολότερο θα είναι για το σύστημα να ανταποκριθεί σε αυτά, χωρίς να γίνει χρήση κεντρικών πόρων του δικτύου (backhaul). Οι τιμές που λαμβάνει το **R** είναι οι παρακάτω:

- R = 2, 5, 8, 11, 14, 17, 20, 23, 26, 29

Στην προκειμένη περίπτωση αξίζει να τονιστεί πως με την αύξηση των αιτημάτων **R**, αυξάνεται σημαντικά και ο αριθμός των αντικειμένων που περιλαμβάνει ο πίνακας **P**, δηλαδή η cache του τοπικού κέντρου δεδομένων. Αυτό συμβαίνει, διότι για περισσότερα αιτήματα χρηστών R, περισσότερα αντικείμενα $i \in I$, αποκτούν υψηλή τιμή $p_u^{req}(i)$ (όπως φαίνεται στο κεφάλαιο 4.4) και άρα υψηλότερες τιμές $util(i)$ (4.5). Επομένως, όταν η cache **P** επιλέγει μέσω δυναμικού προγραμματισμού ποια αντικείμενα θα εμπεριέχει, βλέπει πως πολλά από αυτά, έχουν υψηλή αξία ($util(i)$) και άρα επιλέγει αυτά με το μικρότερο μέγεθος L_i. Επομένως, για περισσότερα αιτήματα χρηστών **R**, η cache του τοπικού Base station χωράει περισσότερα αντικείμενα όπως φαίνεται στον παρακάτω πίνακα.

Number of requests **R**	2	5	8	11	14	17	20	23	26	29
Items in **P**	54.72	62.04	72.2	77.92	80.04	83.52	89.24	90.52	92.88	94.04

Πίνακας 5.7: Items in P with the increase of user's requests **R**

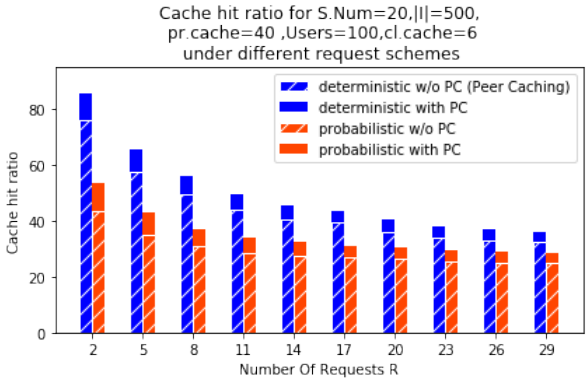
5.6.2 Ένας clusterhead ανά κοινότητα

Τα αποτελέσματα της προσομοίωσης για ένα clusterhead ανά κοινότητα με την αλλαγή των αιτημάτων των χρηστών **R**, τόσο για *Deterministic* όσο και για *Probabilistic* αιτήματα, φαίνονται στον παρακάτω πίνακα.

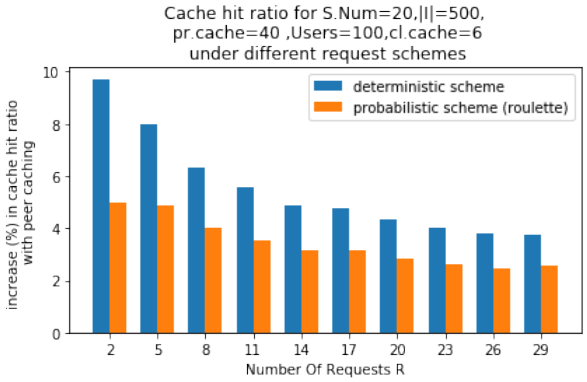
1 Clusterhead										
Deterministic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
Overall cache hit ratio (%)	85.58	65.49	55.88	49.63	45.3	43.38	40.53	38.07	36.88	36.04
Base station cache hit ratio (%)	75.9	58.11	49.58	44.08	40.44	38.6	36.18	34.08	33.09	32.29
peer caching hit ratio(%)	9.68	7.38	6.3	5.56	4.86	4.78	4.35	3.99	3.79	3.76
use of basic clusterhead (#)	19.36	36.88	50.64	61.12	68.12	81.2	87.64	91.84	98.6	108.92
Probabilistic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
Overall cache hit ratio (%)	48.64	39.5	34.88	31.97	30.9	30.12	29.2	27.91	27.69	27.46
Base station cache hit ratio (%)	43.68	34.8	30.87	28.44	27.73	26.97	26.34	25.31	25.21	24.88
peer caching hit ratio(%)	4.96	4.69	4.01	3.53	3.17	3.15	2.86	2.61	2.48	2.58
use of basic clusterhead (#)	9.92	23.44	32.32	38.88	44.64	53.52	57.68	59.96	64.48	74.8

Πίνακας 5.8: Requests: 1 Clusterhead for Deterministic and Probabilistic Requests

Παρατηρούμε πως με την αύξηση των αιτημάτων των χρηστών, πέφτει σημαντικά το συνολικό cache hit ratio του συστήματος. Επίσης, η μείωση του *peer caching hit ratio* παρά την ταυτόχρονη αύξηση των φορών που χρησιμοποιούμε τον βασικό clusterhead είναι λογική, αφού για R=2, έχουμε συνολικά 200 αιτήματα, ενώ για R=29, έχουμε 2900 αιτήματα. Έτσι, παρόλο που αυξάνεται ο απόλυτος αριθμός των φορών που χρησιμοποιούμε τους clusterhead, η ποσοστιαία cache hit ratio που προσφέρεται μέσω αυτών, μειώνεται. Ταυτόχρονα, παρόμοια μείωση λόγω αδυναμίας κάλυψης πολλών αιτημάτων ακολουθεί το *base station cache hit ratio*. Ωστόσο, και στις δύο περιπτώσεις φαίνεται ότι η μείωση επιτυγχάνεται με μειούμενη επιβράδυνση. Τέλος, έχουμε την αναμενόμενη διαφορά απόδοσης ανάμεσα σε *Deterministic* και *Probabilistic* τύπου αιτήματα.



Σχήμα 5.10: 1 Clusterhead Overall Results



Σχήμα 5.11: Peer caching hit ratio

5.6.3 Δύο clusterhead ανά κοινότητα

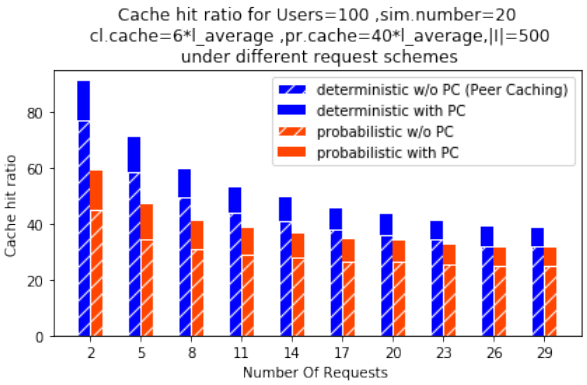
2 Clusterhead										
Deterministic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
Overall cache hit ratio (%)	91.12	70.98	59.53	53.31	49.5	45.68	43.49	41.06	38.78	38.42
Base station cache hit ratio (%)	77.14	58.5	49.31	43.91	40.79	37.9	35.91	34.3	32.2	32.2
peer caching hit ratio(%)	13.98	12.47	10.22	9.39	8.71	7.78	7.58	6.76	6.58	6.22
use of basic clusterhead (#)	17.88	37.92	50.44	62.08	75	80.2	90.36	93.28	103.96	108.24
use of second clusterhead (#)	10.08	24.44	31.32	41.24	46.96	52	61.24	62.2	68.28	72.04

Πίνακας 5.9: Requests: 2 Clusterhead for Deterministic Requests

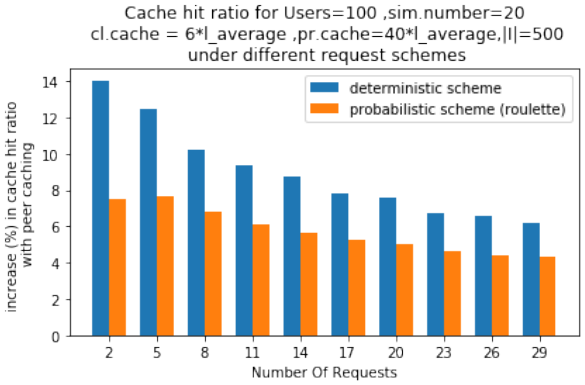
Παρατηρούμε πως για δύο clusterhead ανά κοινότητα παίρνουμε την ίδια μορφή αποτελεσμάτων όπως για έναν clusterhead ανά κοινότητα, αλλά αυτή τη φορά με αυξημένο το *peer caching hit ratio* λόγω της ύπαρξής τους.

2 Clusterhead										
Probabilistic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
Overall cache hit ratio (%)	52.38	42.32	37.8	35.16	33.42	31.96	31.42	30.24	29.54	29.38
Base station cache hit ratio (%)	44.9	34.7	30.97	29.08	27.74	26.68	26.38	25.6	25.11	25.07
peer caching hit ratio(%)	7.48	7.62	6.84	6.07	5.68	5.28	5.04	4.64	4.44	4.31
use of basic clusterhead (#)	10.08	23.24	33.76	40.6	49	54.52	60.04	64.56	69.52	73.56
use of second clusterhead (#)	4.88	11.52	20.96	26.2	30.48	35.2	40.6	42.24	46.56	51.44

Πίνακας 5.10: Requests: 2 Clusterhead for Probabilistic Requests



Σχήμα 5.12: 2 Clusterhead Overall Results



Σχήμα 5.13: Peer caching hit ratio

5.6.4 Παρατηρήσεις

Όπως αναφέραμε προηγουμένως, με την αύξηση των αιτημάτων των χρηστών, έχουμε σημαντική μείωση του συνολικού *cache hit ratio*. Αυτό συμβαίνει, διότι οι caches του συστήματός μας είναι ορισμένου μεγέθους οπότε, όπως είναι λογικό, είναι αδύνατον να χωρέσουν όλα τα αντικείμενα που θα ζητήσουν οι χρήστες. Ωστόσο, η ύπαρξή τους ακόμα και για R>20 είναι ύψιστης σημασίας, αφού αποσυμφορίζει σημαντικά (30-45%) τους πόρους του δικτύου.

Ταυτόχρονα, μέσω των παραπάνω αποτελεσμάτων μπορούμε να βγάλουμε σημαντικά συμπεράσματα για το είδος των cache που είναι αποτελεσματικότερο να χρησιμοποιούμε. Πιο συγκεκριμένα, για μικρό αριθμό αιτημάτων (R<10) είναι σαν να προσομοιώνουμε καταστάσεις που οι cache τόσο του τοπικού κέντρου δεδομένων, όσο και των clusterhead, είναι δυναμικές και αλλάζουν συχνά το περιεχόμενό τους. Αυτό, διότι φανταζόμαστε πως για κάθε **R** αιτήματα αλλάζει το περιεχόμενό τους

και προσαρμόζεται στα καινούρια αιτήματα των χρηστών. Αντίστοιχα, για μεγάλο αριθμό αιτημάτων (R>15), είναι σαν να έχουμε στατικές cache. Δηλαδή, το περιεχόμενό τους αλλάζει κάθε αρκετή ώρα και δεν είναι προσαρμοστικές. Για να ισχύει βέβαια η παραπάνω παραδοχή, υποθέτουμε ότι οι χρήστες δε ζητούν συνέχεια τα ίδια αντικείμενα.

Έτσι, βγάζουμε το ασφαλές συμπέρασμα ότι το σύστημά μας, για το παρόν μέγεθος των cache, είναι αποδοτικότερο όταν το περιεχόμενο τους διαμορφώνεται δυναμικά. Δηλαδή, όταν αλλάζει το περιεχόμενο των cache ανά τακτά χρονικά διαστήματα, ανταποκρινόμενο κάθε φορά στα καινούρια R αιτήματα των χρηστών.

Συμπερασματικά, το πρόβλημα της μείωσης του *cache hit ratio* με την αύξηση των αιτημάτων **R** θα μπορούσε να λυθεί αν δεσμεύαμε περισσότερο χώρο, τόσο στην cache του base station, όσο και στις cache των clusterhead. Για μελλοντική έρευνα, θα μπορούσε να δοκιμαστεί, το μέγεθος των cache του συστήματος να εξαρτάται από το πόσο συχνά ανανεώνονται τα αντικείμενα σε αυτές, δηλαδή του αριθμού **R**. Αν επεξεργάζονται πολλά αιτήματα χρηστών (μεγάλο R) μέχρι την ανανέωσή τους, τότε για να είναι αποδοτικές, οι cache πρέπει να έχουν σχετικά μεγάλο μέγεθος, ενώ αν επεξεργάζονται λίγα αιτήματα, επιτρέπεται το μέγεθός τους να είναι και μικρό.

5.7 Δείκτης ομοιότητας (Similarity Number)

5.7.1 Γενική Περιγραφή

Στο σημείο αυτό πραγματοποιούμε μετρήσεις, αλλάζοντας το *Similarity Number (δείκτη ομοιότητας)*. Όπως προαναφέραμε, ο αριθμός αυτός είναι που καθορίζει αν δύο χρήστες χαρακτηρίζονται ως όμοιοι ή όχι, χαρακτηριστικό το οποίο επηρεάζει το αν θα κοπεί η μεταξύ τους ακμή. Πιο συγκεκριμένα, αν δύο χρήστες έχουν ακμή και χαρακτηριστούν ως μη όμοιοι, τότε αυτή η ακμή θα κλαδευτεί όπως αναφέραμε και στο 4.6.2. Έτσι, όταν το *similarity number* είναι ένας μικρός αριθμός, είναι δυσκολότερο για δύο χρήστες $u \in U$ να χαρακτηρισθούν όμοιοι, αφού πρέπει στα πρώτα *similarity number* στοιχεία των $p_u^{req}(i)$ πινάκων τους να έχουν τουλάχιστον ένα κοινό αντικείμενο i.

Ο βασικός αριθμός που χρησιμοποιούμε για το *similarity number* στις υπόλοιπες περιπτώσεις είναι το **20**. Αριθμός με τον οποίο κόβονται περίπου οι μισές ακμές του αρχικού μας, *RGG* γράφου. Στις μετρήσεις που ακολουθούν, το *similarity number* παίρνει τις τιμές:

- Similarity Number = 10, 20, 30, 50, 70, 90

Άρα αλλάζοντας τις τιμές αυτές, αλλάζουμε και τη δομή του δικτύου, δηλαδή τις ακμές μεταξύ των χρηστών και άρα τα community που τελικά θα δημιουργηθούν μέσω *modularity maximization*.

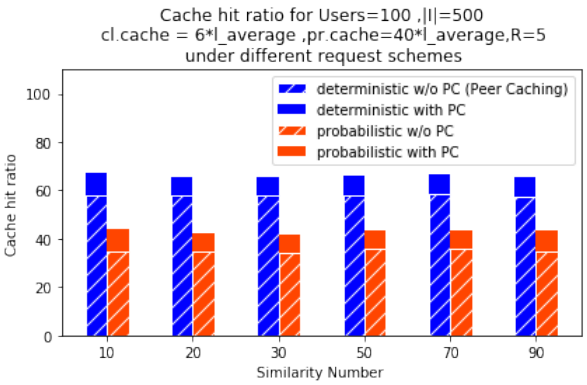
Επίσης, ο αριθμός των αντικειμένων που περιέχει ο πίνακας **P**, είναι σχεδόν όμοιος για όλες τις τιμές των *similarity number* και ο μέσος όρος του ισούται με **63.89** στοιχεία.

5.7.2 Ένας clusterhead ανά κοινότητα

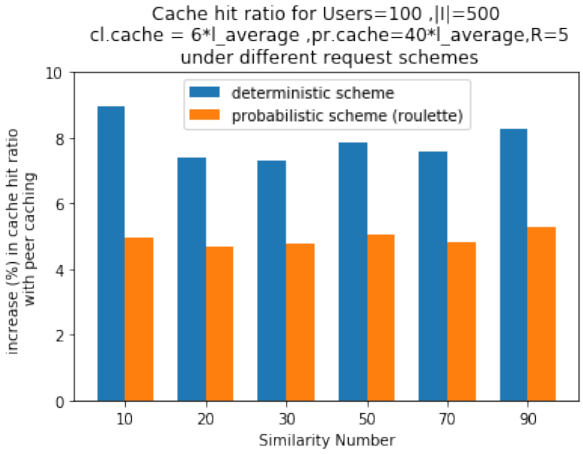
Τα αποτελέσματα για ένα cluster σε κάθε community φαίνονται στους παρακάτω πίνακες.

1 Clusterhead						
Deterministic						
Similarity Number (#)	**10**	**20**	**30**	**50**	**70**	**90**
Overall cache hit ratio (%)	67.4	65.49	65.37	66.06	66.42	65.69
base station cache hit ratio (%)	58.16	58.11	58.07	58.22	58.83	57.41
peer caching hit ratio(%)	8.98	7.38	7.3	7.84	7.58	8.28
use of basic clusterhead (#)	44.88	36.88	36.48	39.2	37.96	41.4
Probabilistic						
Similarity Number (#)	**10**	**20**	**30**	**50**	**70**	**90**
Overall cache hit ratio (%)	38.82	39.5	39.14	40.69	40.46	40.31
Base station cache hit ratio (%)	34.87	34.8	34.35	35.64	35.62	35.02
peer caching hit ratio(%)	4.94	4.69	4.79	5.05	4.84	5.29
use of basic clusterhead (#)	24.72	23.44	23.88	26.96	24.24	26.44

Πίνακας 5.11: Similarity Number: 1 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.14: 1 Clusterhead Overall Results



Σχήμα 5.15: Peer caching hit ratio

Παρατηρούμε ότι το συνολικό *cache hit ratio* όπως και οι υπόλοιπες τιμές παραμένουν σχετι-

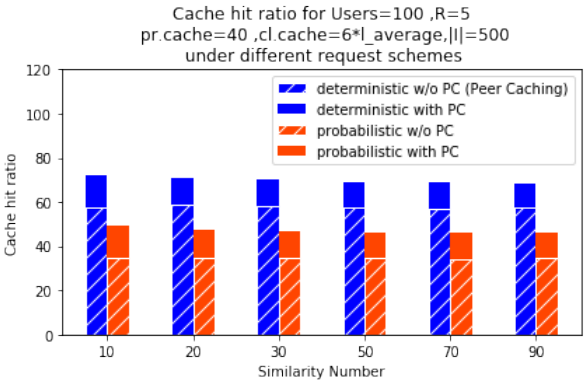
κά σταθερές. Άρα συμπεραίνουμε πως η προσομοίωση μας είναι ανεξάρτητη της μορφολογίας του γράφου που διαχειρίζεται.

5.7.3 Δύο clusterhead ανά κοινότητα

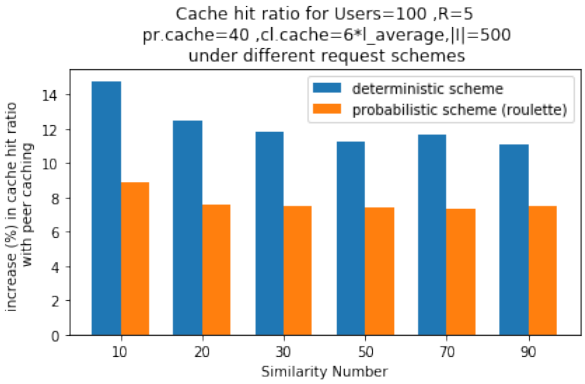
Παρακάτω βλέπουμε τα αποτελέσματα για δύο clusterhead ανά κοινότητα.

2 Clusterhead						
Deterministic						
Similarity Number (#)	**10**	**20**	**30**	**50**	**70**	**90**
Overall cache hit ratio (%)	72.14	70.98	70.06	68.94	68.63	68.4
Base station cache hit ratio (%)	57.4	58.5	58.24	57.65	56.98	57.27
peer caching hit ratio(%)	14.74	12.47	11.82	11.3	11.66	11.13
use of basic clusterhead (#)	44.24	37.92	37.04	39.6	40	38.72
use of second clusterhead (#)	29.44	24.44	22.08	17	18.3	16.92
Probabilistic						
Similarity Number (#)	**10**	**20**	**30**	**50**	**70**	**90**
Overall cache hit ratio (%)	43.74	42.32	42.17	41.92	41.74	42.4
Base station cache hit ratio (%)	34.84	34.7	34.65	34.53	34.44	34.9
peer caching hit ratio(%)	8.9	7.62	7.52	7.39	7.3	7.5
use of basic clusterhead (#)	24.08	23.24	23.96	26.12	27.23	26
use of second clusterhead (#)	17.44	14.88	13.64	11	10.79	11.52

Πίνακας 5.12: Similarity Number: 2 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.16: 2 Clusterhead Overall Results



Σχήμα 5.17: Peer caching hit ratio

5.7.4 Παρατηρήσεις

Παρατηρούμε πως η απόδοση του αλγορίθμου μας είναι σταθερή για τις τιμές του *similarity number*. Όπως προαναφέραμε, για διαφορετικές τιμές, έχουμε διαφορετική δομή δικτύου, δηλαδή, διαφορετικό Average Node Degree γράφου. Επομένως, διαπιστώσαμε πως ο αλγόριθμό μας είναι προσαρμοστικός στην εκάστοτε δομή του δικτύου που θα κληθεί να διαχειριστεί, ακόμα και σε πραγματικές συνθήκες. Αυτή η παρατήρηση είναι ιδιαίτερα καλή, γιατί στην πράξη δεν μπορούμε εύκολα να προβλέψουμε και να διαμορφώσουμε τις κοινωνικές και γεωγραφικές σχέσεις μεταξύ χρηστών (δομή γράφου) και άρα χρειαζόμαστε έναν ελαστικό αλγόριθμο που να μην επηρεάζεται από τη μορφολογία του εκάστοτε δικτύου.

Παράλληλα, παρατηρούμε μια μικρή αύξηση απόδοσης για μικρά similarity number. Αυτό, διότι για μικρές τιμές, έχουμε λιγότερες ακμές μεταξύ χρηστών, καθώς πολλές απο αυτές έχουν κλαδευτεί (4.6.2). Έτσι, όταν εφαρμόζουμε *modularity maximization*, ο αραιός γράφος μας χωρίζεται σε πολλές κοινότητες (communities), προκειμένου οι κοινότητες αυτές να είναι όσο πιο πυκνές γίνεται. Για το λόγο αυτό, περισσότεροι χρήστες, στο σύστημά μας, γίνονται clusterhead και οι cache τους είναι πιο εξειδικευμένες στα αιτήματα των λίγων χρηστών που βρίσκονται στο ίδιο community με αυτούς. Επομένως, έχουμε μια σχετικά αυξημένη απόδοση του *peer caching hit ratio* για μικρά *similarity number* και ταυτόχρονα μεγαλύτερη απόδοση από τη χρήση δύο clusterhead ανά community σχετικά με προηγούμενες προσομοιώσεις (π.χ. 5.4).

Τέλος, η διαφορά στην απόδοση *Deterministic*, *Probabilistic* αιτημάτων δικαιολογείται όπως στο 5.4.4.

5.8 Μέγεθος Cache Τοπικού Κέντρου Δεδομένων (Provider)

5.8.1 Γενική Περιγραφή

Στο παρόν κεφάλαιο βλέπουμε την απόδοση του αλγορίθμου μας, με την αλλαγή του μεγέθους της cache του τοπικού κέντρου δεδομένων. Με την τροποποίησης αυτή, θα μπορέσουμε να προβλέψουμε τα αποτελέσματα, σε πραγματικό επίπεδο, μιας μεγαλύτερης ή αντίστοιχα μικρότερης επένδυσης στο μέγεθος των τοπικών cache. Όπως είπαμε στο 5.2.3 το παραπάνω μέγεθος μετράται με βάση το μέσο μέγεθος $L_{average}$ ενός αντικειμένου $i \in I$. Συγκεκριμένα, οι τιμές της χωρητικότητας για τις οποίες θα πραγματοποιήσουμε προσομοιώσεις είναι οι παρακάτω:

- Base station's cache size ($^*L_{average}$) = 27.5, 35, 42.5, 50, 57.5, 65, 72.5

Προφανώς στην προκειμένη περίπτωση, αφού αλλάζει και το μέγεθος της cache του τοπικού κέντρου δεδομένων θα αλλάζει και ο αριθμός των αντικειμένων που κάθε φορά περιέχει. Ο συγκεκριμένος αυτός αριθμός ανάλογα με το μέγεθος της cache φαίνεται στον παρακάτω πίνακα.

Base station's cache size ($^*L_{average}$)	27.5	35	42.5	50	57.5	65	72.5
Items in **P**	43.48	52.76	65.2	75.16	83.72	95.64	106.2

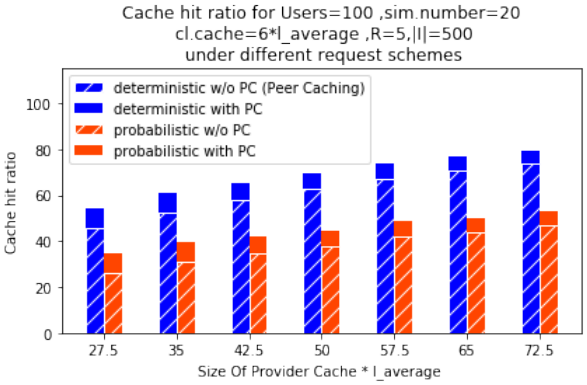
Πίνακας 5.13: Items in P with the increase of Base station's cache size

5.8.2 Ένας clusterhead ανά κοινότητα

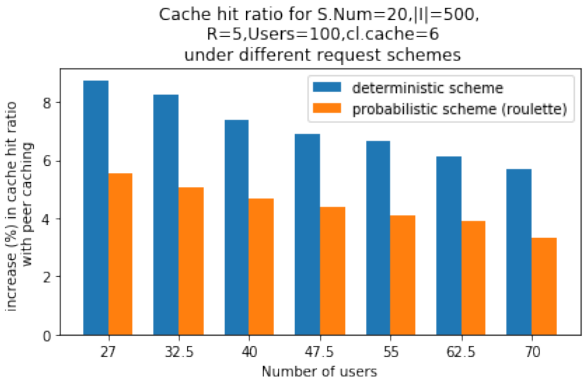
Τα αποτελέσματα του αλγορίθμου για έναν clusterhead ανά κοινότητα τόσο για *Deterministic*, όσο και για *Probabilistic* τύπου αιτήματα φαίνονται στον παρακάτω πίνακα.

1 Clusterhead							
Deterministic							
Base station's cache size (*$L_{average}$)	**27.5**	**35**	**42.5**	**50**	**57.5**	**65**	**72.5**
Overall cache hit ratio (%)	54.18	60.68	65.49	69.77	73.88	76.68	79.37
Base station cache hit ratio (%)	45.46	52.44	58.11	62.85	67.23	70.57	73.67
peer caching hit ratio(%)	8.72	8.24	7.38	6.92	6.65	6.11	5.7
use of basic clusterhead (#)	43.6	41.28	36.88	34.6	33.24	30.56	28.48
Probabilistic							
Base station's cache size (*$L_{average}$)	**27.5**	**35**	**42.5**	**50**	**57.5**	**65**	**72.5**
Overall cache hit ratio (%)	31.46	36.18	39.5	41.9	46.12	47.7	50.42
Base station cache hit ratio (%)	25.9	31.1	34.8	37.5	42.02	43.78	47.1
peer caching hit ratio(%)	5.56	5.08	4.69	4.37	4.1	3.91	3.33
use of basic clusterhead (#)	27.8	25.44	23.44	21.84	20.48	19.56	16.64

Πίνακας 5.14: Base station's cache size: 1 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.18: 1 Clusterhead Overall Results



Σχήμα 5.19: Peer caching hit ratio

Με μία πρώτη ματιά, βλέπουμε πως με την αύξηση του μεγέθους της cache των τοπικών κέντρων δεδομένων έχουμε την ταυτόχρονη αύξηση του *Base station cache hit ratio* και την μικρή μείωση

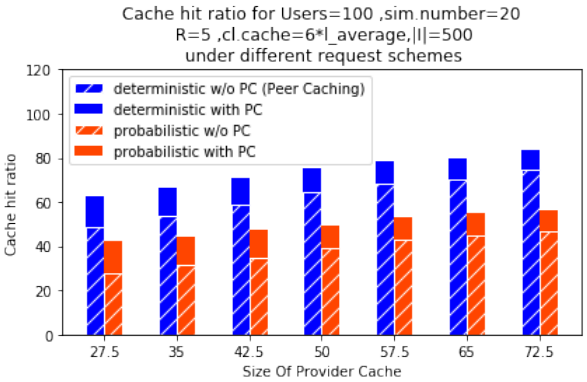
του *peer caching hit ratio*. Συνολικά το *cache hit ratio* του συστήματος αυξάνεται σημαντικά.

5.8.3 Δύο clusterhead ανά κοινότητα

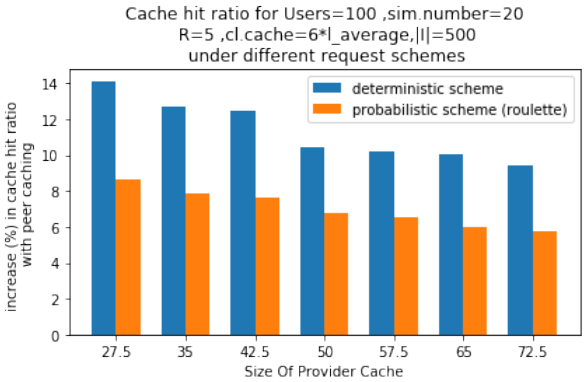
Τα αποτελέσματα του αλγορίθμου για δύο clusterhead ανά κοινότητα τόσο για *Deterministic*, όσο και για *Probabilistic* τύπου αιτήματα φαίνονται στον παρακάτω πίνακα.

2 Clusterhead							
Deterministic							
Base station's cache size (*$L_{average}$)	**27.5**	**35**	**42.5**	**50**	**57.5**	**65**	**72.5**
Overall cache hit ratio (%)	62.58	66.21	70.98	75.18	78.33	79.92	83.82
Base station cache hit ratio (%)	48.51	53.5	58.5	64.77	68.13	69.9	74.36
peer caching hit ratio(%)	14.07	12.71	12.47	10.42	10.2	10.02	9.46
use of basic clusterhead (#)	42.28	38.72	37.92	30.88	31.324	30.12	29.16
use of second clusterhead (#)	28.52	24.84	24.44	21.2	19.8	19.96	18.12
Probabilistic							
Base station's cache size (*$L_{average}$)	**27.5**	**35**	**42.5**	**50**	**57.5**	**65**	**72.5**
Overall cache hit ratio (%)	36.67	39.46	42.32	45.92	49.18	50.7	52.59
Base station cache hit ratio (%)	27.99	31.55	34.7	39.14	42.65	44.7	46.82
peer caching hit ratio(%)	8.68	7.9	7.62	6.78	6.54	6	5.77
use of basic clusterhead (#)	26.48	24.24	23.24	20.12	20.16	18.48	18
use of second clusterhead (#)	17.16	15.28	14.88	13.8	12.68	11.52	10.84

Πίνακας 5.15: Base station's cache size: 2 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.20: 2 Clusterhead Overall Results



Σχήμα 5.21: Peer caching hit ratio

5.8.4 Παρατηρήσεις

Παρατηρούμε πως καθώς αυξάνεται το μέγεθος της cache του τοπικού κέντρου δεδομένων, αυξάνεται το συνολικό *cache hit ratio* του συστήματος, γεγονός το οποίο οφείλεται αποκλειστικά στην αύξηση του *base station cache hit ratio*. Όσο μεγαλύτερο το μέγεθος, τόσο μεγαλύτερη η απόδοση του τοπικού κέντρου δεδομένων, απόδοση που μάλιστα αυξάνεται με σταθερό ρυθμό.

Ωστόσο, παρατηρούμε ότι με την άνοδο του *base station cache hit ratio*, μειώνεται το *peer caching hit ratio*, χωρίς να μεταβάλλεται το μέγεθός της cache των clusterhead. Αυτό συμβαίνει, διότι όσο μεγαλώνει το μέγεθος της cache του τοπικού κέντρου δεδομένων $\mathbf{P}$, τόσο περισσότερο καλύπτει τα αιτήματα των χρηστών τα οποία αντιστοιχούν σε αντικείμενα με σχετικά υψηλό $\boldsymbol{util(i)}$, δηλαδή τα αντικείμενα που μπορούν να ομαδοποιηθούν και να μπουν στην cache με την ασφάλεια ότι θα ζητηθούν. Έτσι, οι χρήστες, στο πρώτο κιόλας στάδιο, ικανοποιούν τα περισσότερα αιτήματά τους για τα σχετικά γνωστά αντικείμενα και όταν μεταβαίνουν στις κοινότητές τους με σκοπό να ικανοποιήσουν τα υπόλοιπα, τους έχουν απομείνει αιτήματα που αφορούν λιγότερο γνωστά αντικείμενα. Επομένως, όταν οι clusterhead κάνουν έρευνα, προκειμένου να βάλουν στις cache τους αντικείμενα που θα ζητηθούν από χρήστες της κοινότητάς τους, που όμως αυτά δεν περιέχονται στην μεγάλη cache $\mathbf{P}$, καταλήγουν σε αντικείμενα $i \in I$ τα οποία δεν έχουν υψηλό $\boldsymbol{util(i)}$, και άρα θα ζητηθούν από λίγους χρήστες. Έτσι, παραμένουν σε αποδόσεις χαμηλού επιπέδου.

Συμπεραίνουμε, λοιπόν, πως με μεγαλύτερη εναπόθεση μνήμης στην cache $\mathbf{P}$ του τοπικού κέντρου δεδομένων, το σύστημά μας αποδίδει περισσότερο και εξοικονομεί σημαντικούς πόρους δικτύου (backhaul). Παρόλα αυτά οδηγεί σε μια μείωση απόδοσης του *peer haching hit ratio*. Επομένως, ανάλογα με το δίκτυο και τους πόρους που διαθέτουμε κάθε φορά, καλούμαστε να εντοπίσουμε τη χρυσή τομή μεταξύ του μεγέθους της cache $\mathbf{P}$ και της clustehead cache, προκειμένου να έχουμε τη μεγαλύτερη δυνατή απόδοση.

Τέλος, τα παραπάνω ισχύουν τόσο για έναν όσο και για δύο clusterhead ανά κοινότητα ενώ η χρήση τους καθώς και η διαφορά στην επιτυχία *Deterministic* και *Probabilistic* αιτημάτων δικαιολογούνται όπως στο 5.4.4.

5.9 Μέγεθος Clusterhead Cache

5.9.1 Γενική Περιγραφή

Στο παρόν κεφάλαιο πραγματοποιούμε μετρήσεις, αλλάζοντας το μέγεθος της cache των clusterhead, Ĺ. Όπως προαναφέραμε, πρόκειται για το πόσο χώρο θα δεσμεύσει ο κάθε χρήστης, που χρίζεται ως clusterhehad (βασικός ή δευτερεύον), στην κινητή του συσκευή, προκειμένου να χρησιμοποιηθεί ως cache για τους χρήστες που βρίσκονται εντός του community του και συνδέονται με αυτόν μέσω ακμής. Όπως προηγουμένως, το παραπάνω μέγεθος μετράται με βάση το μέσο μέγεθος $L_{average}$ ενός αντικειμένου $i \in I$ (5.2.3). Συγκεκριμένα, οι τιμές της χωρητικότητας των clusterhead cache Ĺ, με τις οποίες θα πραγματοποιήσουμε προσομοιώσεις είναι οι παρακάτω:

- Size of Clusterhead cache ($*L_{average}$) = 6, 9, 12, 15, 18, 21

Ταυτόχρονα, ο αριθμός των αντικειμένων που περιέχει ο πίνακας **P**, εφόσον το μέγεθός του δεν μεταβάλλεται, είναι σχεδόν όμοιος για όλες τις τιμές των **Ĺ** και ο μέσος όρος του ισούται με **64.21** στοιχεία.

5.9.2 Ένας clusterhead ανά κοινότητα

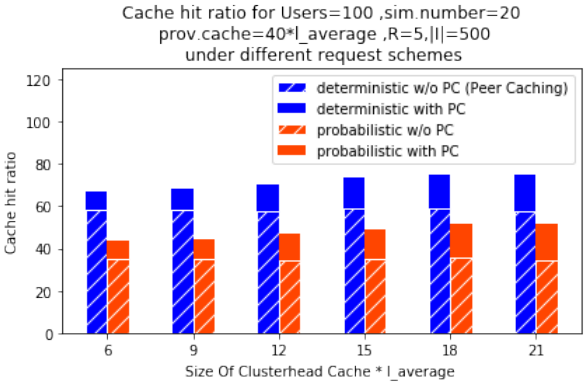
Για έναν clusterhead ανά community, τα αποτελέσματα φαίνονται παρακάτω.

1 Clusterhead						
Deterministic						
Clusterhead cache size ($*L_{average}$)	**6**	**9**	**12**	**15**	**18**	**21**
Overall cache hit ratio (%)	67.14	68.23	70.34	73.33	75.03	74.98
Base station cache hit ratio (%)	58.16	58.4	57.56	59.18	58.9	57.48
peer caching hit ratio(%)	8.98	9.83	12.78	14.15	16.14	17.5
use of basic clusterhead (#)	44.88	49.64	64	71.04	80.68	87.52

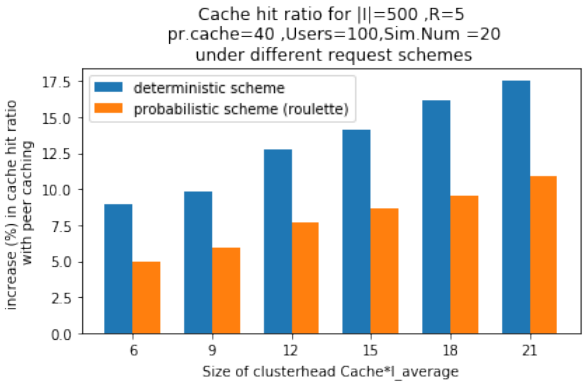
Πίνακας 5.16: Size of Clusterhead cache: 1 Clusterhead for Deterministic Requests

1 Clusterhead						
Probabilistic						
Clusterhead cache size (*$L_{average}$)	**6**	**9**	**12**	**15**	**18**	**21**
Overall cache hit ratio (%)	39.82	40.7	41.74	43.82	45.34	45.12
Base station cache hit ratio (%)	34.87	34.76	34.09	35.16	35.77	34.2
peer caching hit ratio(%)	4.94	5.94	7.65	8.66	9.57	10.91
use of basic clusterhead (#)	24.72	29.88	38.32	43.48	47.84	54.56

Πίνακας 5.17: Size of Clusterhead cache: 1 Clusterhead for Probabilistic Requests



Σχήμα 5.22: 1 Clusterhead Overall Results



Σχήμα 5.23: Peer caching hit ratio

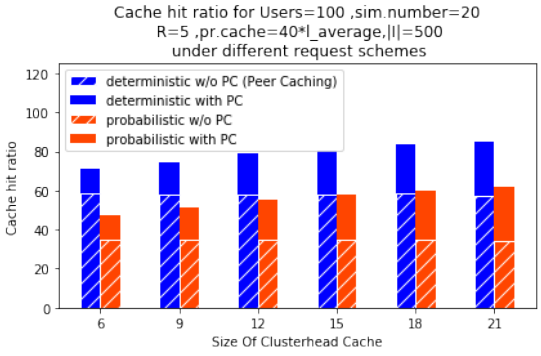
Παρατηρούμε σημαντική βελτίωση του *peer caching hit ratio* και για τις δύο κατηγορίες αιτημάτων. Με την παρουσία ενός μόνο clusterhead, φτάνουμε μέχρι και *17.5%* απόδοση. Έτσι, το συνολικό *cache hit ratio* αυξάνεται και επιτυγχάνεται σημαντική εξοικονόμηση κεντρικών δομών δικτύου (backhaul).

5.9.3 Δύο clusterhead ανά κοινότητα

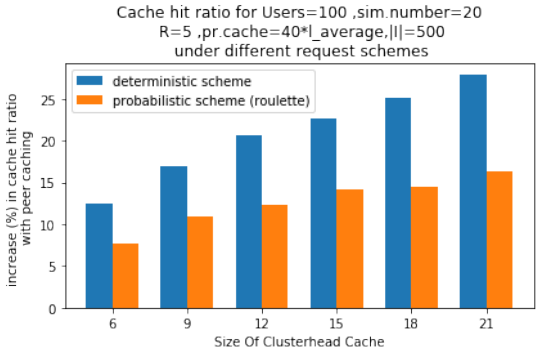
Στη συνέχεια πραγματοποιούμε τις ίδιες μετρήσεις, τοποθετώντας δύο clusterhead για κάθε κοινότητα. Τα αποτελέσματα παρουσιάζονται στους παρακάτω πίνακες και διαγράμματα.

2 Clusterhead						
Deterministic						
Clusterhead cache size (*$L_{average}$)	**6**	**9**	**12**	**15**	**18**	**21**
Overall cache hit ratio (%)	70.98	74.75	78.76	80.5	83.92	85.06
Base station cache hit ratio (%)	58.5	57.87	58.07	57.89	58.81	57.24
peer caching hit ratio(%)	12.47	16.88	20.69	22.61	25.11	27.82
use of basic clusterhead (#)	37.92	53.88	64.24	72.12	82.32	91.8
use of second clusterhead (#)	24.44	30.72	39.2	40.92	43.92	47.4
Probabilistic						
Clusterhead cache size (*$L_{average}$)	**6**	**9**	**12**	**15**	**18**	**21**
Overall cache hit ratio (%)	42.32	45.62	47.12	49.2	49.52	50.56
Base station cache hit ratio (%)	34.7	34.7	34.82	35.08	35.02	34.29
peer caching hit ratio(%)	7.62	10.92	12.3	14.13	14.5	16.27
use of basic clusterhead (#)	23.24	34.84	38.08	45.36	48.64	54.2
use of second clusterhead (#)	14.88	19.84	23.4	25.28	24.24	27.24

Πίνακας 5.18: Size of Cl. cache: 2 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 5.24: 2 Clusterhead Overall Results



Σχήμα 5.25: Peer caching hit ratio

5.9.4 Παρατηρήσεις

Στις παραπάνω μετρήσεις, παρατηρούμε σημαντική αύξηση του *peer caching hit ratio*, τόσο για έναν, όσο και για δύο clusterhead ανά community. Επομένως, η μεγαλύτερη εναπόθεση μνήμης στα κινητά τηλέφωνα των clusterhead, προκειμένου να χρησιμοποιηθούν ως cache, έχει την ανάλογη βελτίωση στην αποτελεσματικότητα τους.

Παράλληλα, βλέπουμε πως όταν οι παραπάνω cache έχουν μεγάλο μέγεθος, η αποτελεσματικότητά τους αρχίζει και φτάνει στα επίπεδα αποτελεσματικότητας των *base stations*. Αυτό σημαίνει ότι ως διαχειριστές δικτύου δεν είναι απαραίτητο να επενδύουμε σε μεγάλα και κοστοβόρα τοπικά κέντρα δεδομένων, αλλά μπορούμε να επικεντρωθούμε στην επικοινωνία μεταξύ χρηστών και να επιτύχουμε παρόμοια αποτελέσματα.

Επίσης, οι τελευταίες τεχνολογίες κινητών τηλεφώνων διαθέτουν μνήμες μεγάλου μεγέθους και άρα μας δίνεται η δυνατότητα να τις χρησιμοποιήσουμε ως cache για τοπικές κοινότητες χρηστών και ταυτόχρονα να χωράνε πολλά αντικείμενα i στη μνήμη τους. Έτσι, με τη συνεργασία "hardware" (τεχνολογία κινητών τηλεφώνων) και "software" (οργάνωση γράφων, εντοπισμός clusterhead, cache κτλ.), μπορούμε να επιτύχουμε σημαντικά επίπεδα *cache hit ratio* και να έρθουμε ένα βήμα πιο κοντά στην εκπλήρωση των απαιτήσεων της νέας *5G τεχνολογίας*.

Επιπρόσθετα, βλέποντας την απόδοση του αλγορίθμου με δύο clusterhead ανά κοινότητα, κρίνουμε απαραίτητη την δοκιμή της επικοινωνίας χρήστη προς χρήστη (D2D communication) με ακόμη περισσότερους clusterhead ανά κοινότητα. Το αποτέλεσμα αυτής της δοκιμής θα μπορούσε να είναι πολύ θετικό και ελπιδοφόρο και ταυτόχρονα θα ερχόμασταν πιο κοντά σε *peer to peer* τεχνολογία με κριτήρια τη γεωγραφική θέση και τα κοινά ενδιαφέροντα χρηστών.

Τέλος, για άλλη μια φορά αποδεικνύουμε ότι ο αλγόριθμός μας είναι ιδιαιτέρα σταθερός και ελαστικός. Αυτό, διότι μεταβάλλοντας τις τιμές μιας μεταβλητής (clusterhead cache) δεν επηρεάζονται οι υπόλοιπες προς το χειρότερο. Δηλαδή η αύξηση της επίδοσης της D2D επικοινωνίας δεν επηρέασε το *base station cache hit ratio*, στο οποίο παρατηρούνται σταθερές τιμές καθόλη τη διάρκεια εκτέλεσης των τελευταίων μετρήσεων.

5.10 Συμπεράσματα και Σχόλια

Στις παραπάνω μετρήσεις, είδαμε τα αποτελέσματα του αλγορίθμου μας για διαφορετικές τιμές των παραμέτρων. Πιο συγκεκριμένα, είδαμε την επίδραση του τοπικού κέντρου δεδομένων, αλλά και της επικοινωνίας μεταξύ χρηστών στην προσπάθεια αποσυμφόρησης σημαντικών πόρων δικτύου. Παρατηρήσαμε πως η συνεργασία αυτών των δύο είναι ιδιαίτερα αποτελεσματική και προσαρμοστική στην εκάστοτε μορφολογία του δικτύου. Ειδικότερα, η καινοτομία της παρούσας διπλωματικής, δηλαδή η ενδοκοινοτική επικοινωνία των χρηστών (D2D communication), μέσω των κινητών τους συσκευών αποδείχτηκε ιδιαίτερα κερδοφόρα, αφού με λύσεις κυρίως *"software"* χαρακτήρα, κατάφερε να επιτύχει σημαντικές τιμές *peer caching hit ratio*. Πρέπει να τονιστεί πως ήδη έχουν αρχίσει να γίνονται πολλές μελέτες πάνω στην υλοποίηση της D2D επικοινωνίας [33], προκειμένου να χρησιμοποιηθεί στην πράξη.

Σε πραγματικό επίπεδο, με τη χρήση ήδη υπαρχόντων τεχνολογιών, μπορούμε να έρθουμε κοντά στην υλοποίησή της διπλωματικής. Για παράδειγμα, τον πίνακα $\boldsymbol{p}_u^{pref}$ για κάθε χρήστη, μπορούμε να τον υπολογίσουμε μέσω των ήδη λειτουργικών cookies. Άρα μπορούμε με βάση αυτά τα ενδιαφέροντα, καθώς και τις σχέσεις των χρηστών στα μέσα κοινωνικής δικτύωσης, να ορίσουμε το κατά πόσο ταιριάζουν δύο χρήστες. Στη συνέχεια με βάση τα GeoData (Data για τον ορισμό της τοποθεσίας ενός ατόμου, βλ. Google Maps) μας γίνεται γνωστό αν δύο χρήστες βρίσκονται γεωγραφικά κοντά. Έτσι, με βάση αυτά τα δύο τοποθετούμε τις ακμές εκεί που κρίνουμε αναγκαίο και στη συνέχεια μπορούμε να φτιάξουμε τα communities και να πραγματοποιήσουμε την πειραματική διαδικασία που περιγράψαμε στη διπλωματική.

Παράλληλα, τα recommendations που πραγματοποιούμε είναι φιλικά προς τους χρήστες, δηλαδή πρόκειται για συστάσεις που ανταποκρίνονται πλήρως στα ενδιαφέροντα των χρηστών. Έτσι, δείξαμε ότι μπορούμε να έχουμε υψηλές τιμές *cache hit ratio*, με User-Friendly τακτικές. Επομένως, με τη σωστή συνεργασία των διαχειριστών του δικτύου και των διαχειριστών των εφαρμογών που χρησιμοποιούν οι χρήστες, μπορούμε να επιτύχουμε αποτελέσματα των οποίων το όφελος να είναι ισοκατανεμημένο προς όλους.

Επιπρόσθετα, παρατηρήσαμε πως ο αλγόριθμός μας είναι εξίσου αποδοτικός για μεγάλο πλήθος χρηστών και αντικειμένων. Παρόλα αυτά, για μικρό πλήθος ατόμων στο δίκτυο, οι cache του πειράματός μας μπορούν να προσαρμοστούν καλύτερα στα αιτήματα των χρηστών και να αποσυμφορήσουν καλύτερα κεντρικούς κόμβους δικτύου. Το ίδιο ισχύει και για τα communities που σχηματίζονται μέσω *modularity maximization*. Έτσι, όσο οργανώνουμε το δίκτυό μας σε περισσότερα communities που το καθένα περιέχει λιγότερους χρήστες, τόσο οι clusterhead θα μπορούν να οργανώνουν τις cache τους με τρόπο που αυτές να ανταποκρίνονται καλύτερα στα αιτήματα των χρηστών, αφού θα υπάρχουν λιγότερα αιτήματα αντικειμένων που θα πρέπει να ικανοποιήσουν. Βλέπουμε, λοιπόν, πως με μία "software" λύση, έχουμε και πάλι αύξηση της αποτελεσματικότητας του πειράματός μας.

Παράλληλα, είδαμε πως όταν οι cache του συστήματος ανανεώνονται ανά τακτά χρονικά διαστήματα, στα κάθε φορά καινούρια $\boldsymbol{p}_u^{pref}$ των χρηστών, έχουν συντριπτικά μεγαλύτερη αποτελεσματικότητα. Αυτό βέβαια έχει ως μειονέκτημα πως με τη συχνή ανανέωση των cache, πρέπει εξίσου συχνά να χρησιμοποιούμε κεντρικούς δρόμους δικτύου, προκειμένου να φορτώσουμε τα συχνά μεταβαλλόμενα αιτήματα των χρηστών στις cache του συστήματος. Αυτό θα μπορούσε να αποφευχθεί, εφόσον οι διαχειριστές δικτύου, αλλά και οι ίδιοι οι χρήστες, δεσμεύουν μεγάλο μέγεθος στη μνήμη τους, προκειμένου να αξιοποιηθεί ως cache. Έτσι, θα μπορούσαν εξ'αρχής να συμπεριλάβουν πολλά αντικείμενα $i \in \boldsymbol{I}$ που πρόκειται να ζητήσουν οι χρήστες σε βάθος χρόνου κι έτσι να αποφευχθεί η χρήση *backhaul*. Συμπεραίνουμε, λοιπόν, πως με βάση τους πόρους που διαθέτουμε και τις εκάστοτε ανάγκες που προκύπτουν, μπορούμε να οργανώσουμε τις cache του δικτύου μας με διαφορετικό και εξίσου αποτελεσματικό τρόπο.

Επιπλέον, παρατηρήσαμε πως όσο περισσότεροι είναι οι clusterhead σε μια κοινότητα, τόσο μεγαλύτερη είναι η αποτελεσματικότητά τους. Αυτό, γιατί, τότε οι χρήστες έχουν τη δυνατότητα να ψάξουν σε πολλές cache, προκειμένου να ικανοποιήσουν κάποιο αίτημά τους για ένα αντικείμενο. Επομένως, με την κατάλληλη τεχνολογία και την προσεκτική ομαδοποίηση (*clustering*) των

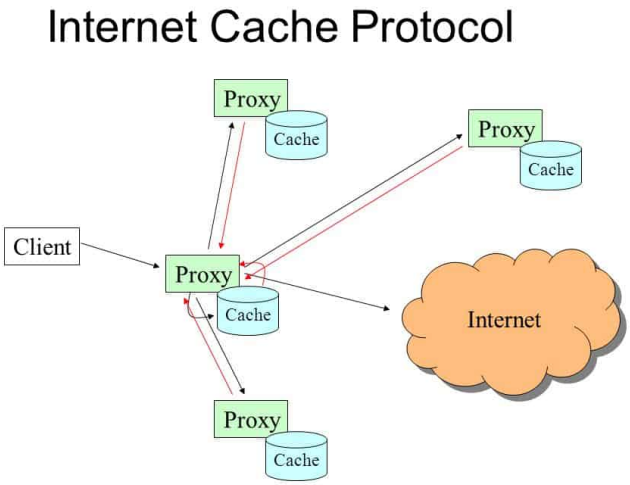
χρηστών θα μπορούσαν πολλοί από αυτούς να είναι clusterhead, και έτσι πολλά αντικείμενα να ανταλλάσσονται τοπικά μέσω των κινητών τους τηλεφώνων. Ωστόσο, προκύπτει το ερώτημα του κατά πόσο είναι διατεθειμένοι οι χρήστες να δεσμεύουν μνήμη στα κινητά τους τηλέφωνα ως cache και κατά πόσο επιτρέπουν ή εμπιστεύονται τους διαχειριστές δικτύων να έχουν πρόσβαση και να πραγματοποιούν μεταφορές δεδομένων σε αυτά. Προς απάντηση αυτών, οι διαχειριστές δικτύων θα μπορούσαν να κρατούν κάθε στιγμή ενήμερους τους χρήστες οι οποίοι είναι clusterhead, για τις πράξεις τους, καθώς και να τους προσφέρουν ορισμένα προνόμια. Πιο συγκεκριμένα, μια προσφορά θα μπορούσε να είναι η ευνοϊκότερη πρόσβασή τους στο διαδίκτυο και στις εφαρμογές του, έτσι ώστε οι μεταφορές που οφείλουν να κάνουν ως clusterhead να μην τους καθυστερούν από τις υπόλοιπες ανάγκες που θέλουν να ικανοποιήσουν. Επίσης, όσο περισσότερος είναι ο χώρος που δεσμεύει ένας clusterhead, τόσο περισσότερα να είναι τα προνόμια δικτύου που απολαμβάνει.

Τέλος, συμπεραίνουμε ότι το πειραματικό μοντέλο που προτείνουμε με cache σταθερού μεγέθους, είναι ιδιαίτερα αποτελεσματικό, αφού αποσυμφορίζει το δίκτυο και τους κεντρικούς του δρόμους. Ταυτόχρονα, ανταποκρίνεται στη αναδυόμενη **5*G*** τεχνολογία η οποία διευκολύνει σημαντικά τις διασυνδέσεις μεταξύ των χρηστών, αλλά και μεταξύ συσκευών που βρίσκονται σε κοντινή, γεωγραφικά, απόσταση (π.χ. IoT).

6 Cache μη σταθερού μεγέθους

6.1 Γενική Περιγραφή

Στο προηγούμενο κεφάλαιο πραγματοποιήσαμε μετρήσεις κάνοντας χρήση cache σταθερού μεγέθους. Στο παρόν κεφάλαιο, προκειμένου να κατανοήσουμε σε μεγαλύτερο βάθος τη χρήση της cache και τα αποτελέσματα που αυτή επιφέρει ανάλογα με τον τρόπο που οργανώνεται, πραγματοποιούμε μετρήσεις χρησιμοποιώντας cache μη σταθερού μεγέθους. Ο τρόπος δόμησης όλου του υπόλοιπου πειράματος παραμένει ίδιος, το μόνο που μεταβάλλεται είναι το μέγεθος και ο τρόπος οργάνωσης της cache τόσο του τοπικού κέντρου δεδομένων, όσο και των clusterhead. Τονίζεται πως η σταθερού μεγέθους cache του προηγούμενου κεφαλαίου, ανταποκρίνεται πολύ περισσότερο στην πραγματικότητα και είναι πιο εύκολο να υλοποιηθεί, παρόλα αυτά δοκιμάζουμε την προσομοίωση και με μη σταθερές cache για λόγους σύγκρισης και περαιτέρω εμβάθυνσης.



Σχήμα 6.1: Προσωρινή αποθήκευση στο διαδίκτυο [34]

6.1.1 Cache τοπικού κέντρου δεδομένων

Όπως αναφέραμε παραπάνω, στο παρόν κεφάλαιο το μέγεθος της cache του τοπικού κέντρου δεδομένων δεν είναι σταθερό. Η τιμή του εξαρτάται από το πλήθος των αντικειμένων που υπάρχουν στο σύστημα, καθώς και το συνολικό τους μέγεθος.

Συγκεκριμένα, υπολογίζεται το l_{total} το οποίο αντιστοιχεί στο άθροισμα όλων των μεγεθών L_i των αντικειμένων $i \in I$ που συμμετέχουν στο πείραμα όπως φαίνεται παρακάτω:

$$l_{total} = \sum_{i \in I} L_i \tag{6.1}$$

Τότε το μέγεθος της cache του τοπικού κέντρου L αντιστοιχεί σε μία τιμή στο διάστημα $[\frac{l_{total}}{16}, \frac{l_{total}}{14}]$, τιμή την οποία λαμβάνουμε μέσω της συνάρτησης *ομοιόμορφης κατανομής*. Αυτό σημαίνει ότι το μέγεθος αυτό εξαρτάται από τον αριθμό των αντικειμένων i που συμμετέχουν στην προσομοίωση. Όσο περισσότερα είναι τα αντικείμενα αυτά, τόσο μεγαλύτερο το μέγεθος της cache του τοπικού κέντρου δεδομένων $\mathbf{P}$.

Στις περισσότερες περιπτώσεις που ο αριθμός των αντικειμένων έχει τη βασική του τιμή ($|I| = 500$), η cache είναι σχετικά σταθερή και χωράει κατά μέσο όρο 57.7 στοιχεία.

Ωστόσο, από τη στιγμή που καθοριστεί το μέγεθος της cache, η επιλογή των αντικειμένων που θα συμπεριλάβει στην μνήμη της είναι και πάλι μέσω του δυναμικού προγραμματισμού, όπως είδαμε στο 4.5.

6.1.2 Clusterhead Cache

Από την άλλη πλευρά η cache των clusterhead οργανώνεται τελείως διαφορετικά, ενώ ο τρόπος που επιλέγονται οι ίδιοι οι clusterhead παραμένει ίδιος. Πιο συγκεκριμένα, όταν αρχίζει η διεξαγωγή του πειράματος και οι χρήστες $u \in U$, αρχίζουν να ψάχνουν στις κοινότητές τους για αντικείμενα, όλες οι cache των clusterhead είναι άδειες. Η μέθοδος που τοποθετούνται αντικείμενα στις cache περιγράφεται από τις παρακάτω οδηγίες.

1. Ο χρήστης u θέλει το αντικείμενο i το οποίο όμως δεν υπάρχει στην αρχική cache του κέντρου δεδομένων $\mathbf{P}$. Τότε ο χρήστης θα ψάξει στην cache του clusterhead (βασικός ή δευτερεύον) της κοινότητάς του.

    - Αν το αντικείμενο υπάρχει στην cache, τότε ο χρήστης το παίρνει, το αίτημά του ικανοποιείται και η περίπτωση θεωρείται *cache hit*.
    - Αν δεν το βρει, το οποίο στην αρχή είναι πολύ πιθανό, αφού η cache είναι αρχικά άδεια, τότε παίρνει το αντικείμενο i που επιθυμεί χρησιμοποιώντας κεντρικές οδούς του δικτύου και η περίπτωση θεωρείται *cache miss*. Στη συνέχεια αυτό το αντικείμενο προστίθεται στην cache του clusterhead την οποία μόλις είχε ψάξει.

Με τον τρόπο αυτό, η clusterhead cache γεμίζει με τα αντικείμενα που δεν βρίσκουν αρχικά σε αυτήν οι χρήστες, ελπίζοντας πως οι χρήστες που ακολουθούν και βρίσκονται στην ίδια κοινότητα,

εφόσον έχουν όμοια ενδιαφέροντα, θα ξαναζητήσουν το ίδιο αντικείμενο και έτσι η περίπτωση θα θεωρηθεί cache hit.

Έτσι, όταν στην κοινότητά μας υπάρχουν πολλοί χρήστες, τα αιτήματα θα είναι επίσης πολλά και οι clusterhead θα καταλήξουν να έχουν cache με πολλά στοιχεία. Στο σημείο αυτό τονίζεται ότι, εφόσον η cache είναι μη σταθερού μεγέθους, δεν έχει άνω όριο στο πόσα αντικείμενα χωράει και προφανώς όσο περισσότερα αντικείμενα περιέχει, τόσο το πιθανότερο ένας χρήστης να βρει αυτό που ψάχνει.

Τέλος, ένας χρήστης ψάχνει στην cache του clusterhead εντός της κοινότητάς του με τον οποίον συνδέεται μέσω ακμής. Αν κάποιος συνδέεται και με τους δύο clusterhead, τότε ψάχνει στην cache μονάχα του βασικού.

6.1.3 Παράμετροι

Παρακάτω παραθέτουμε ορισμένους πίνακες και διαγράμματα, προκειμένου να κατανοήσουμε περαιτέρω τη συμπεριφορά των cache μη σταθερού μεγέθους και να συγκρίνουμε τα αποτελέσματά τους με τις σταθερού μεγέθους. Οι τιμές των παραμέτρων όπως και στην προηγούμενη ενότητα (εκτός μεγέθους cache) είναι όπως παρακάτω:

- Users $|U| = 100$
- Items $|I| = 500$
- *Similarity Number* = 20
- Αριθμός αιτημάτων κάθε χρήστη, $R = 5$
- Το μέγεθος αντικειμένου L_i παίρνει τιμές μέσω της συνάρτησης ομοιόμορφης κατανομής στο διάστημα [1,10]. Στη συνέχεια υπολογίζεται το άθροισμα των μεγεθών τους, l_{total}.
- Το μέγεθος cache του τοπικού κέντρου δεδομένων είναι $L = (\frac{l_{total}}{16}, \frac{l_{total}}{14})$

Στη συνέχεια όπως και στο κεφάλαιο 5 αλλάζουμε τις τιμές ορισμένων παραμέτρων, προκειμένου να δούμε πως ανταποκρίνεται ο αλγόριθμος. Οι παράμετροι αυτοί που θα τροποποιηθούν είναι:

1. Ο αριθμός των χρηστών U (Network Size).
2. Ο αριθμός των αντικειμένων I που συμμετέχουν στο σύστημα.
3. Ο αριθμός των αιτημάτων R που θα πραγματοποιήσει κάθε χρήστης.
4. Το *similarity number*.

Όπως αναφέραμε, οι μετρήσεις θα πραγματοποιηθούν με μοναδικό σκοπό να δούμε πως συμπεριφέρονται οι cache του συστήματος και ιδιαίτερα οι cache των clusterhead. Αυτό, διότι, σύμφωνα με το 6.1.1, το μέγεθος της cache του τοπικού κέντρου δεδομένων, παραμένει σταθερό σε όλες τις περιπτώσεις εκτός από αυτή που αλλάζει ο αριθμός των αντικειμένων I, στην οποία περίπτωση

θα γίνει ειδική αναφορά στον τρόπο απόκρισης. Τέλος, όπως και στο κεφάλαιο 5, οι μετρήσεις αφορούν τα δύο είδη αιτημάτων, **Deterministic** και **Probabilistic** και επίσης για **έναν ή δύο** clsuterhead σε κάθε κοινότητα με ξεχωριστή cache ο καθένας.

6.2 Αριθμός Χρηστών (Network Size)

6.2.1 Γενική περιγραφή

Στην πρώτη περίπτωση, η παράμετρος που τροποποιούμε είναι ο αριθμός των χρηστών *U (Network Size)*. Ορίζοντας τις υπόλοιπες παραμέτρους όπως στο 6.1.3, οι χρήστες παίρνουν τις τιμές:

- Users = 100, 150, 200, 250, 300, 350, 400

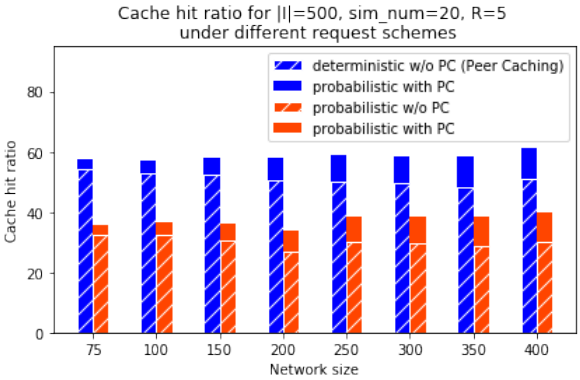
Στην προκειμένη περίπτωση ο πίνακας **P** είναι σχετικά σταθερός, άρα η μόνη παράμετρος της οποίας τα αποτελέσματα θα αναδείξουμε είναι το *peer caching hit ratio*, δηλαδή η αποτελεσματικότητα των clusterhead cache.

6.2.2 Ένας Clusterhead ανά κοινότητα

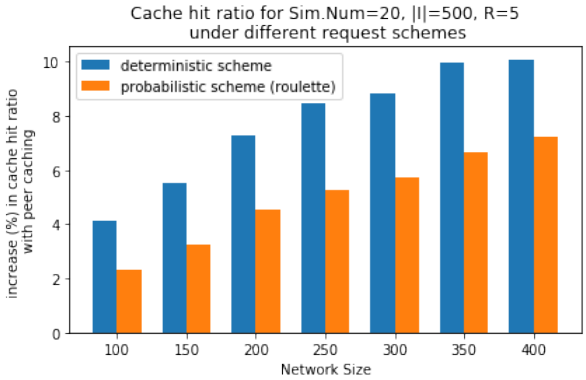
1 Clusterhead							
Deterministic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
peer caching hit ratio(%)	4.15	5.51	7.3	8.49	8.84	9.98	10.07
use of basic clusterhead (#)	20.76	41.48	73.08	106.8	132.6	174.72	201.32
Probabilistic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
peer caching hit ratio(%)	2.33	3.27	4.56	5.25	5.72	6.67	7.25
use of basic clusterhead (#)	11.64	24.6	45.56	65.64	85.76	116.64	145

Πίνακας 6.1: Network Size: 1 Clusterhead for Deterministic and Probabilistic Requests

Με μια πρώτη ματιά βλέπουμε πως όσο περισσότεροι χρήστες συμμετέχουν στο πείραμα και όσο περισσότερα αιτήματα έχουμε, τόσο αποτελεσματικότερος είναι ο αλγόριθμός μας. Περισσότερες λεπτομέρειες φαίνονται και στις εικόνες που ακολουθούν.



Σχήμα 6.2: 1 Clusterhead Overall Results



Σχήμα 6.3: Peer caching hit ratio

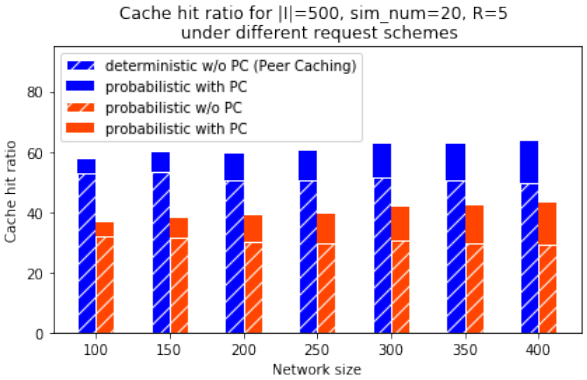
6.2.3 Δύο Clusterhead ανά κοινότητα

Για την ύπαρξη δύο Clusterhead σε κάθε κοινότητα παίρνουμε τα παρακάτω αποτελέσματα.

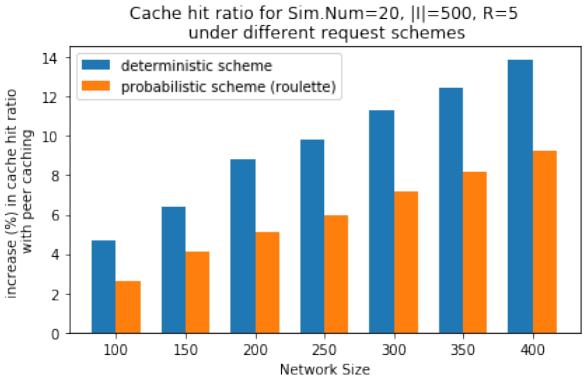
2 Clusterhead							
Deterministic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
peer caching hit ratio(%)	4.65	6.42	8.82	9.78	11.32	12.47	13.86
use of basic clusterhead (#)	17.88	36.84	68.2	90.4	126.96	160.44	199.68
use of Second clusterhead (#)	5.36	11.32	20.68	31.8	42.92	61.12	77.68
Probabilistic							
Network Size (# Users)	**100**	**150**	**200**	**250**	**300**	**350**	**400**
peer caching hit ratio(%)	2.62	4.14	5.12	5.98	7.16	8.19	9.24
use of basic clusterhead (#)	10.6	25.16	40.64	57.84	83.96	110.12	139.32
use of Second clusterhead (#)	2.48	5.92	10.96	16.88	23.48	34.64	45.64

Πίνακας 6.2: Network Size: 2 Clusterhead for Deterministic and Probabilistic Requests

Παρατηρούμε πως υπάρχει αύξηση του *peer caching hit ratio* με την τοποθέτηση δύο clusterhead.



Σχήμα 6.4: 2 Clusterhead Overall Results



Σχήμα 6.5: Peer caching hit ratio

6.2.4 Παρατηρήσεις

Παρατηρήσαμε και στις δύο περιπτώσεις πως με την αύξηση των χρηστών, υπήρξε και αύξηση του *peer caching hit ratio*, δηλαδή της αποτελεσματικότητας των clusterhead cache. Αυτό συμβαίνει, διότι, με την αύξηση των χρηστών, έχουμε περισσότερα άτομα για κάθε κοινότητα που δημιουργείται. Ως αποτέλεσμα, αυξάνονται και οι χρήστες που ψάχνουν την clusterhead cache της κοινότητάς τους, προκειμένου να ικανοποιήσουν ένα αίτημα αντικειμένου που δεν ικανοποιήθηκε, προηγουμένως, από την cache του τοπικού κέντρου δεδομένων. Έτσι, η αρχικά άδεια cache των clusterhead γεμίζει γρήγορα από τα αρχικά ανικανοποίητα αιτήματα των χρηστών, όπως περιγράψαμε στο 6.1.2. Ως αποτέλεσμα, οι χρήστες έχουν πολλές πιθανότητες να βρουν, τελικά, το αντικείμενό τους στην μεγάλη cache του clusterhead της κοινότητάς τους.

Αντίθετα, όταν έχουμε λίγους χρήστες σε κάθε community, η αποτελεσματικότητα του αλγορίθμου είναι πολύ μικρότερη, αφού η clusterhead cache δεν προλαβαίνει να γίνει μεγάλη και έτσι, να είναι πιθανό για τους χρήστες να βρουν τα αντικείμενα που επιθυμούν σε αυτή.

Συγκρίνοντας την αποτελεσματικότητα της *non stable cache* με αυτή της *stable cache* του κεφαλαίου 5, παρατηρούμε πως στην προκειμένη περίπτωση (μη σταθερού μεγέθους cache), με την αύξηση του αριθμού των χρηστών η αποτελεσματικότητα αυξάνεται, ενώ προηγουμένως (σταθερού μεγέθους cache) σχετικά μειώνεται. Βέβαια, για λίγα αιτήματα ή χρήστες, η cache σταθερού μεγέθους είναι πολύ πιο αποτελεσματική, αφού δεν περιμένει να γεμίσει από τους χρήστες που δεν ικανοποιούν τα αιτήματά τους, προκειμένου να γίνει χρήσιμη, αλλά έχει από την αρχή και καθόλη τη διάρκεια τα ίδια αντικείμενα που έχουν μεγάλη πιθανότητα να ζητηθούν. Επίσης όπως προαναφέραμε, η σταθερού μεγέθους cache του κεφαλαίου 5 είναι πολύ πιο οικονομική και εύκολη στην υλοποίηση από μία μη σταθερού μεγέθους.

Παράλληλα, στα παραπάνω αποτελέσματα, βλέπουμε πως η ύπαρξη δύο clusterhead, δεν αυξάνει το *peer caching hit ratio* το ίδιο αποτελεσματικά με την cache σταθερού μεγέθους. Αυτό, διότι όπως και ο βασικός, ο δευτερεύον clusterhead αρχικά έχει άδεια cache και, εφόσον, χρησιμοποιείται λιγότερο από τον βασικό, δεν γεμίζει ικανοποιητικά η cache του, προκειμένου να υπάρχουν αρκετές πιθανότητες ένας χρήστης να ικανοποιήσει σε αυτή το αίτημά του.

Τέλος, η διαφορά στην αποτελεσματικότητα μεταξύ *Deterministic* και *Probabilistic* αιτημάτων εξηγείται όπως στο 5.4.4.

6.3 Αριθμός Αντικειμένων (Number of Items)

6.3.1 Γενική Περιγραφή

Στο κεφάλαιο αυτό θα πραγματοποιήσουμε μετρήσεις, αλλάζοντας αυτή τη φορά τον αριθμό των αντικειμένων I που συμμετέχουν στην προσομοίωση. Οι υπόλοιπες τιμές είναι όπως στο 6.1.3 και ο αριθμός των I ορίζεται όπως παρακάτω:

- Number of Items |I| = 500, 750, 1000, 1250, 1500

Επίσης, στην προκειμένη περίπτωση που αλλάζει ο αριθμός των αντικειμένων $|I|$, αλλάζει και το μέγεθος της cache του τοπικού κέντρου δεδομένων **P**. Έτσι, θα συμπεριλάβουμε στα αποτελέσματά μας το πεδίο *base station cache hit ratio* που αντιστοιχεί στην αποτελεσματικότητά της παραπάνω cache.

6.3.2 Ένας clsuterhead ανά κοινότητα

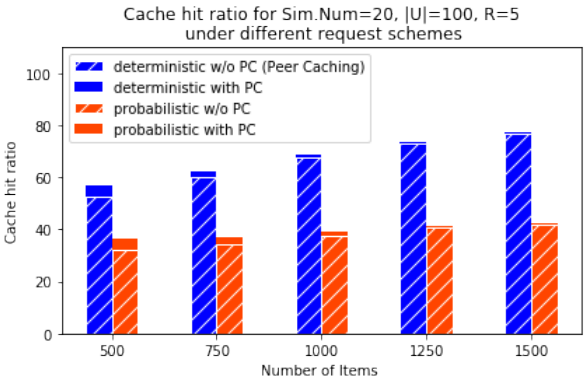
Τα αποτελέσματα της προσομοίωσης για την ύπαρξη ενός clusterhead σε κάθε κοινότητα φαίνονται στους παρακάτω πίνακες και διαγράμματα.

1 Clusterhead					
Deterministic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
base station cache hit ratio (%)	52.85	60.15	67.46	72.92	76.78
peer caching hit ratio(%)	4.15	2.33	1.3	0.7	0.6
use of basic clusterhead (#)	12.16	11.64	6.48	3.48	3

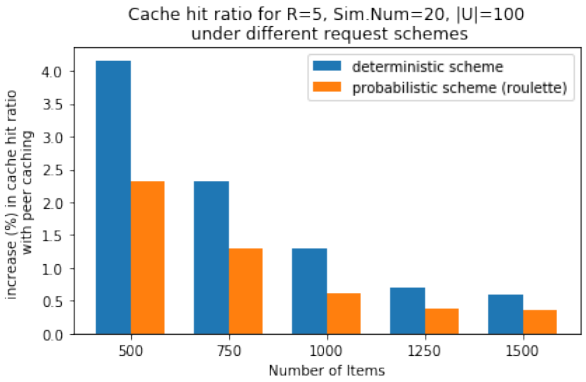
Πίνακας 6.3: Items: 1 Clusterhead for Deterministic Requests

1 Clusterhead					
Probabilistic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
base station cache hit ratio (%)	32.31	34.54	37.78	40.59	41.92
peer caching hit ratio(%)	2.33	1.29	0.62	0.37	0.36
use of basic clusterhead (#)	11.64	6.44	3.12	1.84	1.8

Πίνακας 6.4: Items: 1 Clusterhead for Probabilistic Requests



Σχήμα 6.6: 1 Clusterhead Overall Results



Σχήμα 6.7: Peer caching hit ratio

Βλέπουμε πως με την αύξηση του αριθμού των αντικειμένων $|I|$, αυξάνεται σημαντικά το *base station cache hit ratio*, ενώ το *peer caching hit ratio*, οριακά μηδενίζεται.

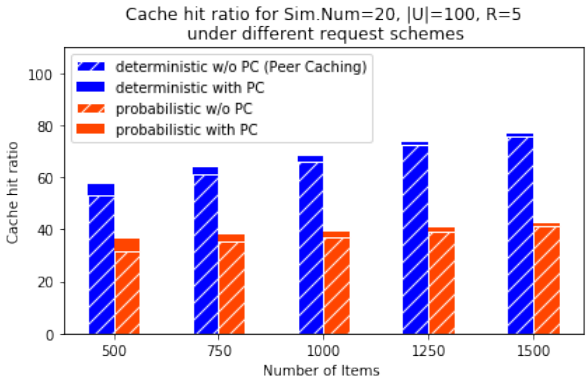
6.3.3 Δύο clusterhead ανά κοινότητα

2 Clusterhead					
Deterministic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
base station cache hit ratio (%)	53.02	61.44	66.31	72.58	75.96
peer caching hit ratio(%)	4.65	2.58	1.77	1.18	0.68
use of basic clusterhead (#)	17.88	10	6.96	4.68	2.28
use of Sexond clusterhead (#)	5.36	2.88	1.92	1.2	1.04

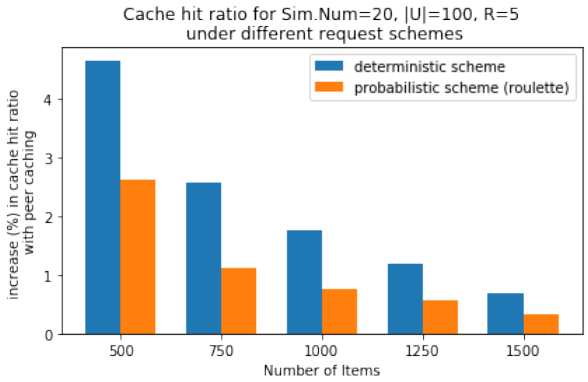
Πίνακας 6.5: Items: 2 Clusterhead for Deterministic Requests

2 Clusterhead					
Probabilistic					
Network Size (# Users)	**500**	**750**	**1000**	**1250**	**1500**
base station cache hit ratio (%)	31.84	35.35	37.29	39.42	41.59
peer caching hit ratio(%)	2.62	1.12	0.77	0.58	0.32
use of basic clusterhead (#)	10.6	4.36	3.04	2.44	1.28
use of Sexond clusterhead (#)	2.48	1.24	0.72	0.44	0.32

Πίνακας 6.6: Items: 2 Clusterhead for Probabilistic Requests



Σχήμα 6.8: 2 Clusterhead Overall Results



Σχήμα 6.9: Peer caching hit ratio

Βλέπουμε πως και με δύο clusterhead ανά κοινότητα το *peer caching hit ratio* παραμένει σε πολύ χαμηλά επίπεδα.

6.3.4 Παρατηρήσεις

Παρατηρούμε πως και στις δύο περιπτώσεις το *peer caching hit ratio* φτάνει σε πολύ χαμηλά επίπεδα. Αυτό συμβαίνει, διότι, η χωρητικότητα της αρχικής cache του τοπικού κέντρου δεδομένων αυξάνεται με την αύξηση των αντικειμένων I και άρα μεγαλώνει και η αποτελεσματικότητά της. Έτσι, τα περισσότερα αιτήματα χρηστών ικανοποιούνται στο πρώτο στάδιο και η clusterhead cache δεν προλαβαίνει να γίνει μεγάλη, προκειμένου να ικανοποιήσει τα υπόλοιπα. Ακόμα, όμως, κι αν η cache $\mathbf{P}$ παρέμενε σταθερή σε μέγεθος, η αύξηση των αντικειμένων I θα οδηγούσε και πάλι σε μείωση της αποτελεσματικότητας των clusterhead cache, αφού τότε αυτές θα γέμιζαν με αντικείμενα τα οποία όμως λόγω της μεγάλης ποικιλίας τους δεν θα ζητούνταν αναλογικά από πολλούς χρήστες, αφού τα αιτήματα θα μοιράζονταν σε πληθώρα αντικειμένων. Επίσης, η ύπαρξη δεύτερου clusterhead σε κάθε κοινότητα είναι άσκοπη, καθώς αυξάνει ελάχιστα την αποτελεσματικότητα του *peer caching hit ratio*.

Συγκρίνοντας, λοιπόν, την αποτελεσματικότητα των *non stable cache* με των *stable cache* ως προς τα πόσα αντικείμενα υπάρχουν στο σύστημα, παρατηρούμε πως η σταθερού μεγέθους cache είναι πολύ πιο αποτελεσματικές, σταθερές και ανεξάρτητες του πλήθους αντικειμένων, λόγω της δομής τους, όπως παρατηρήσαμε και στο 5.5. Ωστόσο, το συμπέρασμα αυτό είναι αντιφατικό, αφού στην προκειμένη περίπτωση δεσμεύσαμε περισσότερη μνήμη στα κινητά τηλέφωνα των clusterhead για να χρησιμοποιηθούν ως cache και το αποτέλεσμα ήταν χειρότερο.

Ως αποτέλεσμα του τελευταίου, γίνεται κατανοητό ότι για την αποσυμφόρηση των δικτύων, δεν επαρκεί απλά η επένδυση σε υλικό (hardware, μνήμη), αλλά εξίσου απαραίτητη (ίσως και σημαντικότερη) κρίνεται η προσεκτική σχεδίαση του συστήματός, δηλαδή οι λύσεις "software" χαρακτήρα.

6.4 Αριθμός αιτημάτων χρηστών

6.4.1 Γενική Περιγραφή

Παρακάτω πραγματοποιούμε μετρήσεις, αλλάζοντας αυτή τη φορά τον αριθμό των αιτημάτων των χρηστών **R**, κρατώντας τις υπόλοιπες τιμές όπως στο 6.1.3. Οι τιμές που λαμβάνει το **R** είναι οι παρακάτω:

- R = 2, 5, 8, 11, 14, 17, 20, 23, 26, 29

Στην προκειμένη περίπτωση και για τους ίδιους λόγους που αναλύσαμε στο 5.6, ο αριθμός των αντικειμένων που περιλαμβάνει ο πίνακας **P** αυξάνεται με την αύξηση των αιτημάτων **R** των χρηστών, παρόλο που το μέγεθος του πίνακα παραμένει σταθερό. Για το λόγο αυτό, παρατίθεται ο παρακάτω πίνακας στον οποίο αναγράφεται ο μέσος αριθμός αντικειμένων του πίνακα **P** που αντιστοιχούν στα **R** αιτήματα χρηστών και ταυτόχρονα, συμπεριλαμβάνεται η μέτρηση της αποτελεσματικότητάς του **P**, *base station cache hit ratio*.

Number of requests **R**	2	5	8	11	14	17	20	23	26	29
Items in **P**	46.2	53.32	62.4	65.88	71.4	74.68	79.4	78.68	81.2	81.8

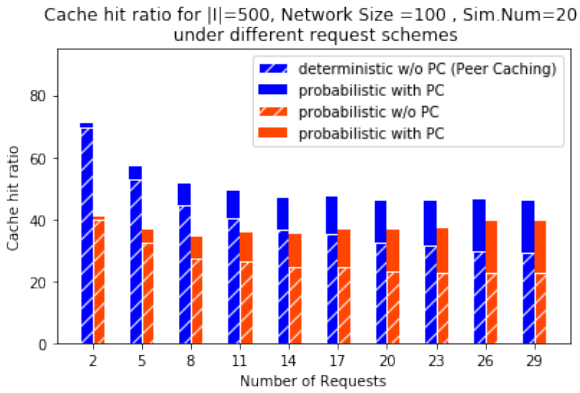
Πίνακας 6.7: Items in P with the increase of user's requests **R**

6.4.2 Ένας clusterhead ανά κοινότητα

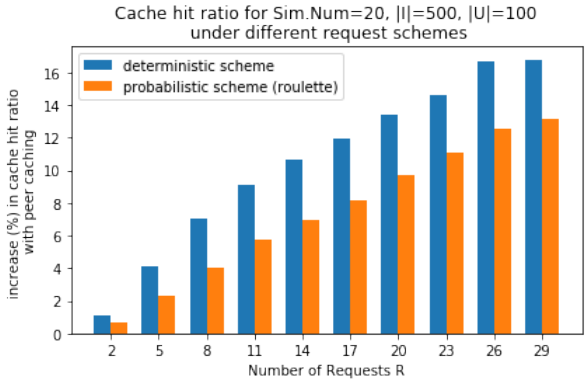
Στους παρακάτω πίνακες και διαγράμματα βλέπουμε τα αποτελέσματα της προσομοίωσης με την ύπαρξη ενός clusterhead σε κάθε κοινότητα.

1 Clusterhead										
Deterministic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
base station cache hit ratio (%)	69.82	52.85	44.54	40.12	36.46	35.26	32.38	31.28	29.82	29.12
peer caching hit ratio(%)	1.14	4.15	7.05	9.11	10.62	11.92	13.41	14.58	16.66	16.75
use of basic clusterhead (#)	2.52	20.76	56.36	100.48	148.68	203.6	268.28	335.44	433.28	486.32
Probabilistic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
base station cache hit ratio (%)	39.74	32.31	27.41	26.42	24.59	24.61	23.08	22.62	22.66	22.51
peer caching hit ratio(%)	0.68	2.33	4.04	5.76	6.94	8.17	9.72	11.09	12.53	13.19
use of basic clusterhead (#)	1.44	11.64	32.32	63.6	97.2	139.12	194.32	255.08	325.76	382.76

Πίνακας 6.8: Requests: 1 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 6.10: 1 Clusterhead Overall Results



Σχήμα 6.11: Peer caching hit ratio

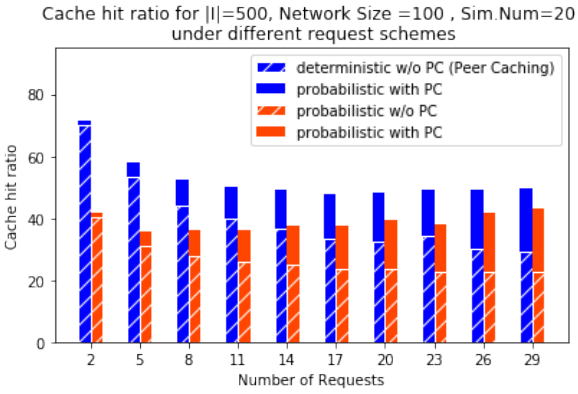
Βλέπουμε πως με την αύξηση των αιτημάτων των χρηστών, αυξάνεται ραγδαία η χρήση των clusterhead και το *peer caching hit ratio* φτάνει σε πολύ υψηλά επίπεδα.

6.4.3 Δύο clusterhead ανά κοινότητα

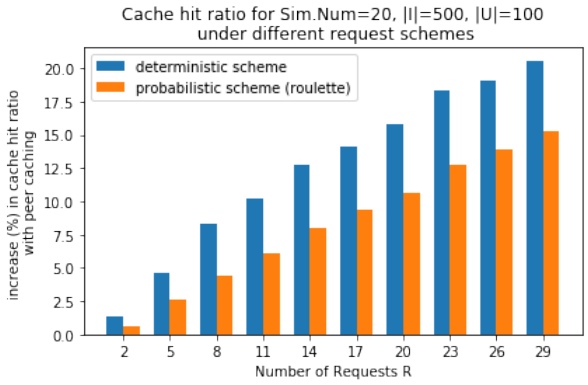
Παρακάτω παρατίθενται τα αποτελέσματα με την ύπαρξη δύο clusterhead σε κάθε κοινότητα.

2 Clusterhead										
Deterministic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
base station cache hit ratio (%)	70.12	53.02	44.04	39.95	36.71	33.58	32.32	34.14	29.9	29.06
peer caching hit ratio(%)	1.37	4.65	8.3	10.2	12.7	14.08	15.76	18.3	19.06	20.54
use of basic clusterhead (#)	2.2	17.88	52	85.4	134.72	185.2	248.16	331.24	384.64	451.76
use of second clusterhead (#)	0.64	5.36	14.44	26.92	42.84	54.04	67.08	88.16	110.28	143.12
Probabilistic										
User's requests R (#)	**2**	**5**	**8**	**11**	**14**	**17**	**20**	**23**	**26**	**29**
base station cache hit ratio (%)	40.3	31.84	27.87	25.9	24.95	23.52	23.68	22.81	22.51	22.54
peer caching hit ratio(%)	0.62	2.62	4.38	6.13	8.05	9.4	10.6	12.72	13.89	15.32
use of basic clusterhead (#)	1.08	10.6	29	52.36	89.28	127	171.76	235	287	349.48
use of second clusterhead (#)	0.24	2.48	6.08	15.12	23.52	32.8	40.2	57.32	73.72	94.72

Πίνακας 6.9: Requests: 2 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 6.12: 2 Clusterhead Overall Results



Σχήμα 6.13: Peer caching hit ratio

Στην περίπτωση των δύο clusterhead, παρατηρούμε ακόμα μεγαλύτερη βελτίωση του *peer caching hit ratio*.

6.4.4 Παρατηρήσεις

Στα παραπάνω αποτελέσματα βλέπουμε πως με την αύξηση των αιτημάτων των χρηστών **R**, έχουμε πολύ μεγάλη αύξηση του *peer caching hit ratio*. Αυτό συμβαίνει, διότι, οι clusterhead που διαθέτουν cache χωρίς άνω όριο μεγέθους, με πολλά αιτήματα, καταλήγουν να έχουν cache με πάρα πολλά στοιχεία και άρα να είναι πολύ πιθανό κάποιος χρήστης να ικανοποιήσει ένα αίτημά του σε αυτές. Επίσης, η cache του τοπικού κέντρου δεδομένων έχει σταθερό μέγεθος και άρα δεν μπορεί να ανταποκριθεί στα αυξημένα αιτήματα των χρηστών κι έτσι το *base station cache hit ratio* πέφτει. Ως αποτέλεσμα, περισσότερα αιτήματα χρηστών καταλήγουν στις κοινότητές τους (clusterhead) για να ικανοποιηθούν.

Συγκρίνοντας το παραπάνω αποτέλεσμα με αυτό των σταθερού μεγέθους cache, βλέπουμε πως στην προκειμένη περίπτωση έχουμε καλύτερη απόκριση του αλγορίθμου και ως εκ τούτου καλύτερα αποτελέσματα *peer caching hit ratio*. Στην πράξη, όμως, θα ήταν αδύνατο να κάνουμε χρήση τόσο μεγάλων cache στα κινητά τηλέφωνα των clusterhead. Παράλληλα, επιβεβαιώνουμε το συμπέρασμα του κεφαλαίου 6, ότι αν οι cache του συστήματός μας δεν ανανεώνουν το περιεχόμενό τους ανά τακτά χρονικά διαστήματα (R>10), πρέπει να είναι μεγάλου μεγέθους, προκειμένου να είναι αποτελεσματικές. Επίσης, όπως και στο 6.2, παρατηρούμε ότι η ύπαρξη δύο clusterhead, δεν έχει την ίδια αποτελεσματικότητα, όπως είχε με cache σταθερού μεγέθους.

6.5 Similarity Number

6.5.1 Γενική Περιγραφή

Στο σημείο αυτό πραγματοποιούμε μετρήσεις, αλλάζοντας το *Similarity Number (δείκτη ομοιότητας)*. Όπως προαναφέραμε στο 5.7, ο αριθμός αυτός είναι που καθορίζει αν δύο χρήστες χαρακτηρίζονται ως όμοιοι ή όχι, χαρακτηριστικό το οποίο επηρεάζει το αν θα κοπεί η μεταξύ τους ακμή. Άρα, αλλάζοντας τον αριθμό αυτό, αλλάζουμε την τροπολογία του δικτύου και τη δόμηση των κοινοτήτων. Συγκεκριμένα το *similarity number* παίρνει τις τιμές:

- Similarity Number = 10, 20, 30, 40, 50, 65, 80, 95

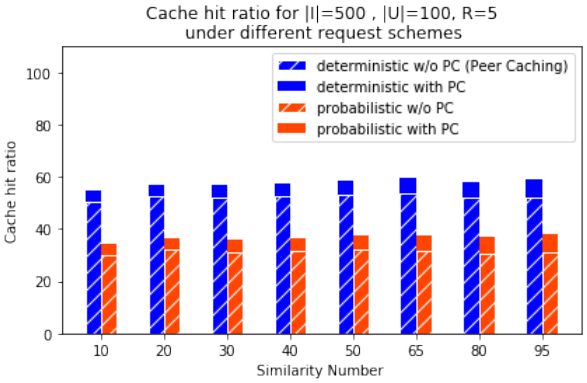
Επίσης, ο αριθμός των αντικειμένων που περιέχει ο πίνακας **P**, είναι σχεδόν όμοιος για όλες τις τιμές των *similarity number* (αφού ο αριθμός των ***I*** μένει σταθερός) και ο μέσος όρος του ισούται με **55.18** στοιχεία.

6.5.2 Ένας clusterhead ανά κοινότητα

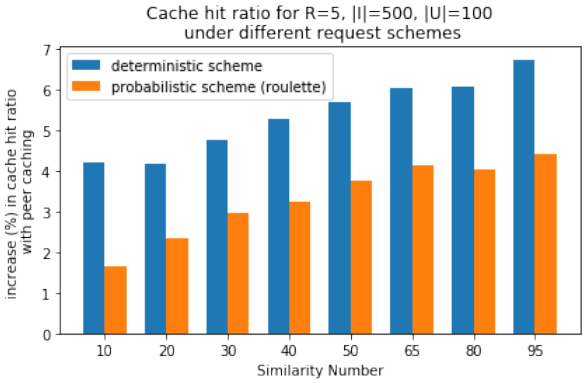
Παρακάτω φαίνονται τα αποτελέσματα της προσομοίωσης για την D2D επικοινωνία, έχοντας έναν clusterhead σε κάθε κοινότητα.

1 Clusterhead								
Deterministic								
Similarity Number (#)	**10**	**20**	**30**	**40**	**50**	**65**	**80**	**95**
peer caching hit ratio(%)	4.19	4.15	4.76	5.27	5.67	6.03	6.07	6.71
use of basic clusterhead (#)	20.96	20.76	23.8	26.36	28.36	30.16	30.36	33.56
Probabilistic								
Similarity Number (#)	**10**	**20**	**30**	**40**	**50**	**65**	**80**	**95**
peer caching hit ratio(%)	1.66	1.89	2.96	3.22	3.76	4.14	4.03	4.42
use of basic clusterhead (#)	8.32	7.12	14.8	16.08	18.8	20.68	20.16	22.08

Πίνακας 6.10: Similarity Number: 1 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 6.14: 1 Clusterhead Overall Results



Σχήμα 6.15: Peer caching hit ratio

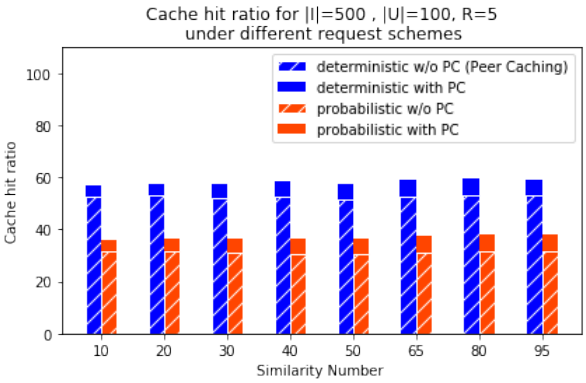
Με μία πρώτη ματιά, παρατηρούμε πως ο αλγόριθμός μας παρουσιάζει μικρή αύξηση του *peer caching hit ratio* με την αύξηση του similarity number.

6.5.3 Δύο clusterhead ανά κοινότητα

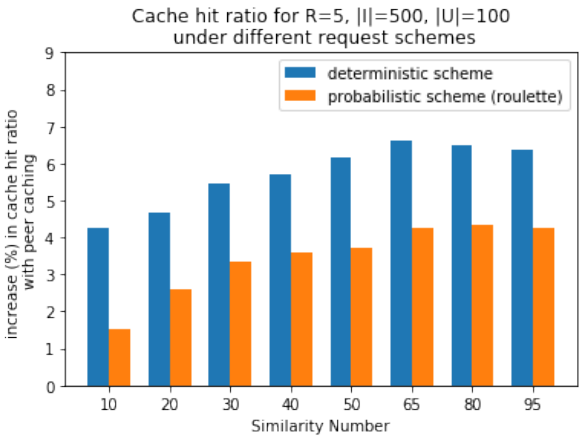
Στη συνέχεια παρουσιάζονται τα αποτελέσματα της προσομοίωσης τοποθετώντας δύο clusterhead σε κάθε κοινότητα, τροποποιώντας και πάλι την παράμετρο του *similarity number*.

2 Clusterhead								
Deterministic								
Similarity Number (#)	**10**	**20**	**30**	**40**	**50**	**65**	**80**	**95**
peer caching hit ratio(%)	4.24	4.65	5.44	5.69	6.15	6.6	6.5	6.38
use of basic clusterhead (#)	15.24	17.88	23.28	25.16	28.48	30.48	29.68	29.64
use of Second clusterhead (#)	6.2	5.36	4.24	3.52	2.2	2.48	2.52	1.96
Probabilistic								
Similarity Number (#)	**10**	**20**	**30**	**40**	**50**	**65**	**80**	**95**
peer caching hit ratio(%)	1.53	2.62	3.36	3.6	3.71	4.26	4.35	4.26
use of basic clusterhead (#)	5.96	10.6	14.68	16.08	17.04	19.8	20.28	19.6
use of Second clusterhead (#)	1.96	2.48	2.32	1.96	1.48	1.64	1.4	1.48

Πίνακας 6.11: Similarity Number: 2 Clusterhead for Deterministic and Probabilistic Requests



Σχήμα 6.16: 2 Clusterhead Overall Results



Σχήμα 6.17: Peer caching hit ratio

6.5.4 Παρατηρήσεις

Παρατηρούμε πως με την αύξηση του *similarity number*, το *peer caching hit ratio* αυξάνεται με μικρό ρυθμό. Αυτό, διότι όσο μεγαλύτερο είναι, τόσο περισσότερες συνδέσεις θα έχει ο clusterhead και άρα τόσο περισσότεροι χρήστες θα ανατρέξουν στην cache του, προκειμένου να ικανοποιήσουν ένα αίτημά τους. Έτσι, οι clusterhead καταλήγουν να έχουν μεγάλες cache γεμάτες αντικείμενα και άρα είναι πιθανότερο να φανούν χρήσιμες. Ωστόσο, όσο μεγαλύτερο είναι το *similarity number*, τόσο περισσότερο υπάρχουν συνδέσεις μεταξύ μη όμοιων χρηστών και άρα οι cache των clusterhead θα

έχουν πολλά αντικείμενα χωρίς, όμως, να είναι αρκετά πιθανό να ζητηθούν από τους χρήστες, αφού πλέον οι κοινότητες δεν είναι μεταξύ χρηστών με όμοια ενδιαφέροντα, αλλά μόνο μεταξύ χρηστών που βρίσκονται γεωγραφικά κοντά. Συμπεραίνουμε, λοιπόν, πως με τον τρόπο αυτό αφιερώνεται υπερβολικά πολύ χώρος μνήμης στα κινητά τηλέφωνα των clusterhead ως cache, χωρίς όμως, να υπάρχει ιδιαίτερο αποτέλεσμα. Επίσης, για ακόμα μια φορά, η ύπαρξη δεύτερου clusterhead δεν βελτιστοποιεί σε σημαντικό βαθμό το πρόβλημα.

Άρα, είναι σημαντικό για εξοικονόμηση υλικού, να κρατήσουμε το *similarity number* σε χαμηλά επίπεδα, προκειμένου οι κοινότητες να περιέχουν χρήστες με κοινά ενδιαφέροντα και άρα οι cache να οργανώνονται με πιο αποτελεσματικό τρόπο. Θυμίζουμε, ότι σύμφωνα με τα αποτελέσματα του 5.7, οι cache σταθερού μεγέθους είναι σχετικά ανεξάρτητη από το *similarity number* και άρα από την τελική τροπολογία του δικτύου.

6.6 Συμπεράσματα και Σχόλια

Στο παρόν κεφάλαιο εξετάσαμε τη λειτουργία cache, των οποίων το μέγεθος αυξάνεται δυναμικά, χωρίς την ύπαρξη άνω ορίου. Παρατηρήσαμε πως οι εν λόγω clusterhead cache έχουν μεγαλύτερη αποτελεσματικότητα όταν ανήκουν σε ένα μεγάλο δίκτυο με πολλούς χρήστες και με πολλά αιτήματα χρηστών $\boldsymbol{R}$, πριν την ανανέωσή τους. Αυτό, διότι έτσι γίνονται μεγάλες με πολλά αντικείμενα και έτσι πολλοί χρήστες θα ικανοποιήσουν τελικά κάποιο αίτημά τους, δηλαδή, θα βρούν στο περιεχόμενό των cache, το αντικείμενο που επιθυμούν.

Ωστόσο, για να πραγματοποιηθεί το παραπάνω απαιτείται οι χρήστες που είναι clusterhead, να δεσμεύσουν μεγάλο χώρο στα κινητά τους τηλέφωνα, προκειμένου αυτός να χρησιμοποιηθεί ως cache, πράγμα το οποίο πρακτικά είναι αδύνατο. Παράλληλα, ένα ακόμη μειονέκτημα της παραπάνω υλοποίησης, είναι πως αρχικά οι clusterhead cache είναι άδειες και μέχρι να γεμίσουν με αντικείμενα, η ύπαρξή τους είναι αναποτελεσματική. Επίσης, βάζοντας στην cache όλα τα αντικείμενα που ζητούν και δε βρίσκουν οι χρήστες, καταλήγουν να υπάρχουν πολλά αντικείμενα, πιθανότατα μεγάλου μεγέθους, που δε θα ζητηθούν ποτέ από άλλο χρήστη και δεσμεύουν μνήμη χωρίς λόγο.

Παράλληλα, με τη συγκεκριμένη προσομοίωση, επιβεβαιώσαμε τον ισχυρισμό που κάναμε στο κεφάλαιο 5, ότι δηλαδή, με τη συχνή ανανέωση των clusterhead cache (μικρό $\boldsymbol{R}$), δεν απαιτείται η δέσμευση μεγάλης μνήμης, προκειμένου αυτές να είναι αποτελεσματικές (υψηλό *peer caching hit ratio*). Αντίθετα, για λιγότερο συχνή ανανέωση των cache (μεγάλο $\boldsymbol{R}$), απαιτείται μεγάλη μνήμη, 6.4. Επίσης, η ύπαρξη παραπάνω clusterhead σε κάθε κοινότητα για την περίπτωση της cache μη σταθερού μεγέθους κρίθηκε άσκοπη σε αντίθεση με το κεφάλαιο 5.

Σε συνδυασμό όλων των παραπάνω και για την πρακτική υλοποίηση της D2D επικοινωνίας, επιβάλλεται να επικεντρωθούμε σε cache σταθερού μεγέθους, διότι οι χρήστες πρέπει να ξέρουν πόση μνήμη του κινητού τους τηλεφώνου θα χρησιμοποιηθεί από τους διαχειριστές δικτύων ως cache. Παρόλα αυτά θα ήταν χρήσιμη μια υλοποίηση συνδυαστικού χαρακτήρα όπου για παράδειγμα στο ένα κομμάτι της μνήμης μιας clusterhead cache να μπαίνουν αντικείμενα με βάση το δυναμικό προγραμματισμό όπως στο 4.5 και στο άλλο κομμάτι να μπαίνουν αντικείμενα δυναμικά, με βάση πολυζήτητα νέα αιτήματα χρηστών. Έτσι, οι διαχειριστές δικτύων, ανάλογα με την τοπολογία του

δικτύου, τους πόρους που διαθέτουν, τα ενδιαφέροντα των χρηστών και την εκάστοτε συγκυρία, να είναι σε θέση να αποφασίσουν τον ακριβή συνδυασμό τεχνολογίας προς υλοποίηση.

Συμπερασματικά, η προσεκτική οργάνωση των cache κρίνεται ιδιαίτερα σημαντική, αφού λύσεις "software" χαρακτήρα μπορούν να δώσουν πιο αποτελεσματικές λύσεις με την προσαρμογή τους στα εκάστοτε δίκτυα, από ότι θα έδιναν οι λύσεις "hardware" χαρακτήρα (υλικό, μνήμη). Συνολικά, είδαμε πως η clusterhead cache και η ενδοκοινοτική επικοινωνία, είναι ένα χρήσιμο εργαλείο, με σκοπό την αποσυμφόρηση πολυσύχναστων δρόμων δικτύου και το διαμοιρασμό πακέτων, των οποίων η χρήση θα διευκολυνθεί περαιτέρω στην αναδυόμενη $5G$ τεχνολογία.

7 Συμπεράσματα και Μελλοντική Εργασία

Στην παρούσα διπλωματική, προτείναμε μια μέθοδο ενοποίησης της διαδικασίας συστάσεων (Re7ommender Systems) που αφορά υπηρεσίες (αντικείμενα) προς ζήτηση από τους χρήστες και της διαδικασίας προσωρινής αποθήκευσης (cache) ορισμένων υπηρεσιών τόσο στα τοπικά κέντρα δεδομένων (Base stations, BS), όσο και στα κινητά τηλέφωνα των χρηστών, με σκοπό την εξυπηρέτηση των παραπάνω αιτημάτων αποτελεσματικά (Quality of Experience) και τοπικά, χρησιμοποιώντας όσο το δυνατόν λιγότερο κεντρικές οδούς του δικτύου (backhaul). Πιο συγκεκριμένα, αφού ορίσαμε τα ενδιαφέροντα των χρηστών u που συμμετέχουν στο σύστημα για αντικείμενα i, $p_u^{pref}(i)$, πραγματοποιήσαμε ένα σύστημα συστάσεων, στο οποίο προτείναμε σε αυτούς αντικείμενα προς ζήτηση τα οποία ταιριάζουν αρκετά με τα ενδιαφέροντά τους (user-friendly) και διαμορφώσαμε τα τελικά αιτήματά τους $p_u^{req}(i)$. Στη συνέχεια, με βάση τα παραπάνω, οργανώσαμε τις cache του συστήματός μας, προκειμένου να ικανοποιούνται όσο το δυνατόν περισσότερα αιτήματα τοπικά. Έτσι, τοποθετήσαμε μέσω δυναμικού προγραμματισμού *FTPAS* στην, συγκεκριμένου μεγέθους, μνήμη της cache του τοπικού κέντρου δεδομένων (BS), αντικείμενα i τα οποία είναι αρκετά πιθανό να ζητηθούν (υψηλό *utility*) από τους χρήστες που συμμετέχουν στο σύστημα και συνδέονται με το συγκεκριμένο BS. Όμως, παρόλο που η cache αυτή είναι αποτελεσματική, οι δυνατότητές της είναι περιορισμένες καθώς απευθύνεται σε πολλούς χρήστες και έτσι δε μπορεί συγκεκριμενοποιήσει τα αντικείμενά της για ομάδες χρηστών με όμοια ενδιαφέροντα. Για το λόγο αυτό, υλοποιήσαμε την επικοινωνία μεταξύ χρηστών (D2D communication) οι οποίοι έχουν όμοια ενδιαφέροντα και βρίσκονται γεωγραφικά κοντά, υλοποίηση η οποία αποτελεί και την πρωτοτυπία τούτης της διπλωματικής. Έτσι, κάνοντας χρήση της τεχνολογίας των γράφων και συγκεκριμένα μέσω του *Random Geometric Graph (RGG)* δημιουργούμε ένα τυχαίο γράφο με κόμβους που αντιστοιχούν στους χρήστες του συστήματος. Στη συνέχεια, τοποθετούμε ακμές μεταξύ χρηστών που βρίσκονται κοντά (κοντινοί κόμβοι) και ταυτόχρονα έχουν όμοια ενδιαφέροντα και μέσω *modularity maximization* ομαδοποιούμε (clustering) τους κόμβους σε όσο το δυνατόν πυκνότερες κοινότητες (communities). Έπειτα, με βάση την κεντρικότητα βαθμού (Degree Centrality), ορίζουμε clusterhead (ένα ή δύο) για κάθε κοινότητα, οι οποίοι συνδέονται με πολλούς χρήστες στην κοινότητα που ανήκουν. Οι clusterhead αυτοί δεσμεύουν στην κινητή τους συσκευή συγκεκριμένο μέγεθος μνήμης που χρησιμοποιείται ως cache και τοποθετούν σε αυτή αντικείμενα που δεν υπάρχουν στην αρχική cache του BS και ταιριά-

ζουν στα ενδιαφέροντα των χρηστών εντός της κοινότητάς τους. Μετά, ορίσαμε την πειραματική μας διάταξη όπου όλοι οι χρήστες $u \in U$ ζητούν από το σύστημα R αντικείμενα $i \in I$ και μετρήσαμε την επιτυχία που είχαν οι cache της διάταξης (BS και clusterhead), προκειμένου να ικανοποιήσουν τα αιτήματα των χρηστών τοπικά. Οι χρήστες ψάχνουν για το αντικείμενό τους αρχικά στην cache του τοπικού κέντρου δεδομένων και στη συνέχεια στους clusterhead της κοινότητάς τους. Στη συνέχεια βλέπουμε τα αποτελέσματα του αλγόριθμού μας, τροποποιώντας τις παραμέτρους του συστήματος (users, items, number of requests (R), similarity number, clusterhead cache size, provider cache size).

Παρατηρήσαμε πως ο αλγόριθμός μας είναι ιδιαίτερα αποτελεσματικός και ιδιαίτερα η D2D επικοινωνία μεταξύ χρηστών της ίδιας κοινότητας, αφού αναλογικά με το μέγεθος της μνήμης που δεσμεύουν οι clusterhead στα κινητά τους τηλέφωνα, προσφέρουν υψηλό cache hit ratio στο σύστημα. Παράλληλα, ο αλγόριθμος είναι ιδιαίτερα ελαστικός και προσαρμοστικός στη δομή του δικτύου. Στη συνέχεια, αλλάξαμε τις cache σε μη σταθερού μεγέθους, προκειμένου να εμβαθύνουμε στη συμπεριφορά των cache και να δοκιμάσουμε καινούργια συστήματα βελτιστοποίησης του αλγορίθμου. Είδαμε πως παρόλο που οι cache μη σταθερού μεγέθους είναι δύσκολες στην υλοποίηση, μπορούν να χρησιμεύσουν σε λύσεις συνδυαστικού τύπου.

Συνολικά, λοιπόν, παρουσιάσαμε έναν αποτελεσματικό τρόπο ικανοποίησης των αιτημάτων των χρηστών που αφορούν υπηρεσίες δικτύου (QoE), αποσυμφορίζοντας όσο το δυνατόν περισσότερο βασικούς πόρους δικτύων. Το καταφέραμε συνδυάζοντας τεχνικές *mobile edge computing* που είναι η οργάνωση της cache στα τοπικά κέντρα δεδομένων (BS) και *mobile edge networking* που είναι η επικοινωνία μεταξύ των κινητών συσκευών των χρηστών (D2D communication) μέσω γράφων και κοινοτήτων. Ταυτόχρονα, συμπεράναμε πως με την εποικοδομητική συνεργασία των διαχειριστών δικτύου, δηλαδή, αυτών που οργανώνουν τις cache του συστήματος (BS, clusterhead) και των διαχειριστών των εφαρμογών, δηλαδή, αυτών που πραγματοποιούν τα recommendations σε χρήστες, μπορούμε να προσφέρουμε καλύτερες υπηρεσίες, να οργανώνουμε καλύτερα το σύστημά μας και ταυτόχρονα να σεβόμαστε τις επιθυμίες και τα ενδιαφέροντα των χρηστών.

7.1 Μελλοντική Εργασία

Παρακάτω παραθέτουμε ορισμένα πεδία έρευνας στην κατεύθυνση της διπλωματικής που αποσκοπούν στην υλοποίησή της σε πραγματικό επίπεδο ή στην περαιτέρω εμβάθυνση.

- Έρευνα, προκειμένου σε κάθε κοινότητα που δημιουργείται, κάθε χρήστης να είναι εν δυνάμει clusterhead και έτσι να πλησιάσουμε σε μια *peer to peer* τεχνολογία μέσα στις κοινότητες, όπου όλοι οι συμμετέχοντες να ανταλλάσουν πακέτα, δεσμεύοντας ο καθένας λίγη μνήμη που αθροιστικά, όμως, να είναι ικανοποιητική.

- Να έρθουν σε συμφωνία οι διαχειριστές δικτύου με τους χρήστες-clusterhead, έτσι ώστε οι τελευταίοι να είναι σε θέση να δεσμεύουν μνήμη στην κινητή του συσκευή χωρίς το φόβο της παραβίασης των προσωπικών τους δεδομένων και της υπερφόρτωσης της συσκευής τους.

- Ανάπτυξη τεχνολογίας τεχνητής νοημοσύνης (artificial intelligence), έτσι ώστε το σύστημα (cache, μέγεθος μνήμης κτλ) να προσαρμόζεται αυτόματα με βάση την εκάστοτε δομή του δικτύου που διαχειρίζεται και τους υλικούς πόρους που διαθέτει.

- Ανάπτυξη πρωτοκόλλου συνεργασίας ανάμεσα σε διαχειριστές δικτύου και διαχειριστές εφαρμογών (joint caching and recommendation problem) που να αποσκοπεί σε ένα σύστημα λειτουργικότερο και εύκολα διαχειρίσιμο. Επίσης, πρέπει να υπάρχει ίση αντιμετώπιση όσον αφορά την ευνοϊκότερη πρόσβαση σε παροχές δικτύου για όλες τις εφαρμογές του διαδικτύου που συμμετέχουν στο σύστημα και όχι μονάχα για τις διασημότερες.

Bibliography

[1] geeks. *Node.edge*. https://www.geeksforgeeks.org/mathematics-graph-theory-basics-set-1/.

[2] Vasileios Karyotis, Eleni Stai και Symeon Papavassiliou. *Evolutionary dynamics of complex communications networks*. CRC Press, 2013.

[3] dlpng. *Random.Graphs*. https://dlpng.com/tag/graph?q=&pageNo=2.

[4] wikipedia. *Star.Graph*. https://en.wikipedia.org/wiki/Star_(graph_theory).

[5] Yehonatan Cohen, Danny Hendler και Amir Rubin. «Node-centric detection of overlapping communities in social networks». Στο: *International Conference and School on Network Science*. Springer. 2017, σσ. 1–10.

[6] Irwin King, Baichuan Li και Tom Chao Zhou. «Social Network Analysis». Στο: (2012).

[7] Hichem Frigui και Raghu Krishnapuram. «Clustering by competitive agglomeration». Στο: *Pattern recognition* 30.7 (1997), σσ. 1109–1119.

[8] Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang και Wenbo Wang. «A survey on mobile edge networks: Convergence of computing, caching and communications». Στο: *IEEE Access* 5 (2017), σσ. 6757–6779.

[9] Cisco Visual Networking. «Cisco global cloud index: Forecast and methodology, 2015-2020. white paper». Στο: *Cisco Public, San Jose* (2016).

[10] researchgate. *Mobile.Edge*. https://www.researchgate.net/figure/Architecture-of-Mobile-Edge-Computing-MEC-servers-caching-system_fig1_317255787.

[11] Wikipedia. *Information Society*. https://en.wikipedia.org/wiki/Information_society.

[12] Dong Liu και Chenyang Yang. «A learning-based approach to joint content caching and recommendation at base stations». Στο: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, σσ. 1–7.

[13] Carlos A Gomez-Uribe και Neil Hunt. «The netflix recommender system: Algorithms, business value, and innovation». Στο: *ACM Transactions on Management Information Systems (TMIS)* 6.4 (2015), σσ. 1–19.

[14] Renjie Zhou, Samamon Khemmarat και Lixin Gao. «The impact of YouTube recommendation system on video views». Στο: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 2010, σσ. 404–410.

[15] Osman Khalid, Samee U Khan και Albert Y Zomaya. *Big Data Recommender Systems: Algorithms, Architectures, Big Data, Security and Trust*. Computing και Networks, 2019.

[16] datascience. *Recommender.Syst*. https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd.

[17] Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos και Iordanis Koutsopoulos. «Caching-aware recommendations: Nudging user preferences towards better caching performance». Στο: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, σσ. 1–9.

[18] ericsson. *D2D.comm*. https://www.ericsson.com/en/blog/2014/7/d2d-communications---what-part-will-it-play-in-5g.

[19] vectorstock. *Items.Network*. https://www.vectorstock.com/royalty-free-vector/allowed-and-prohibited-items-in-carry-on-baggage-vector-24245368.

[20] britannica. *Users.Network*. https://www.britannica.com/technology/telecommunications-network.

[21] Anand Rajaraman, Jure Leskovec και Jeffrey D Ullmann. «Mining data streams». Στο: *Mining of Massive Datasets, 2nd ed., Cambridge University Press*. 2014, σσ. 165–173.

[22] Dilip Kumar Krishnappa, Michael Zink, Carsten Griwodz και Pål Halvorsen. «Cache-centric video recommendation: an approach to improve the efficiency of youtube caches». Στο: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11.4 (2015), σσ. 1–20.

[23] Harvard. *Recommender.System*. http://sitn.hms.harvard.edu/flash/2017/recommended-machine-learning-helps-choose-consume-next/.

[24] B. Zisimopoulos. *Combined Optimization*.

[25] geeksforgeeks. *ksanpsack problem*. https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/.

[26] V. V. Vazirani. *Approximation algorithms*.

[27] Github. *Random Geometric Graph Python*. https://networkx.github.io/documentation/stable/reference/generated/networkx.generators.geometric.random_geometric_graph.html.

[28] semantics. *Edge.pruning*. https://www.semanticscholar.org/paper/Edge-pruning-based-community-detection-He-Xu/53687d0e7ed6964c8ecfc860c9183d1938ce839c.

[29] Github. *Community Detection with Modularity Maximization Python*. https://zhiyzuo.github.io/python-modularity-maximization/doc/quick-start.html.

[30] mdpi. *Clusterhead Detection*. https://www.mdpi.com/1424-8220/20/5/1375/htm.

[31] Wikipedia. *Fitness proportionate selection*. https://en.wikipedia.org/wiki/Fitness_proportionate_selection.

[32] edc.ncl. *roullete.sel*. http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php.

[33] B. Bai, L. Wang, Z. Han, W. Chen και T. Svensson. «Caching based socially-aware D2D communications in wireless content delivery networks: a hypergraph framework». Στο: *IEEE Wireless Communications* 23.4 (2016), σσ. 74–81.

[34] slideplayer. *Web.cache*. https://slideplayer.com/slide/4942272/.