Η ανάγκη για περισσότερους πόρους υπήρχε και θα υπάρχει πάντα. Με τη πρόοδο της τεχνολογίας καταφέραμε να έχουμε πρόσβαση σε μεγάλη υπολογιστική ισχύ, ενώ ξεπεράσαμε τον περιορισμό και το τεράστιο κόστος, του να πρέπει να αγοράσουμε κάθε τι που χρησιμοποιούμε. Ωστόσο, ακόμα και αυτή η λύση έχει το δικό της ταβάνι και χρειάστηκε αλλαγή ή έστω βελτιώσεις, τις οποίες ήρθε να φέρει το Cloud Computing και στη περίπτωση μας, μια τεχνική καλύτερης διαχείρισης πόρων, το task offloading.

Με τον όρο **task offloading** αναφερόμαστε στην μεταφορά ενός ή περισσοτέρων υπολογιστικών διεργασιών από μια μονάδα επεξεργαστή σε μια άλλη, σε μία εξωτερική πλατφόρμα ή στο cloud. Η μεταφορά αυτή βασίζεται σε μια αλγοριθμική απόφαση, η οποία παίρνεται λογίζοντας διάφορους περιορισμούς καθώς και προκλήσεις που καλούμαστε να αντιμετωπίσουμε. Ειδικότερα, αναφερόμαστε στους εξής άξονες:

* **Διαχείριση διαθέσιμων πόρων:** Είναι προφανές πως οι πόροι είναι περιορισμένοι από άποψη υλικού-οικονομικής ευχέρειας.
* **Καταμερισμός των διεργασιών:** Όσο περισσότερο εκμεταλλευόμαστε την ιδιότητα του κατεμερισμού και χρησιμοποιούμε όλο το φάσμα επιλογών του, από τερματικό χρήστη μέχρι το cloud, πρέπει να υπολογίσουμε πόσο **κόστος** θα προκύψει από τη **πολυπλοκότητα** του απαραίτητου δικτύου αλλά και πόση θα είναι η χρονική **καθυστέρηση** στην εκπλήρωση της υπηρεσίας. Στόχος στη διαχείριση αυτή αποτελεί **η σταθερότητα, η επεκτασιμότητα, η ασφάλεια και η γεωγραφική κάλυψη** όταν μιλάμε για απομακρυσμένες υπηρεσίες.
* **Απόδοση:** Σε διαχειριστικό επίπεδο, είναι αναγκαίο να παίρνουμε μια **καλή απόφαση για τον καταμερισμό των διεργασιών** (τη σειρά εκτέλεσης, τη προτεραιότητα, τη μεταφορά κλπ) καθώς και σε ποιο μέρος του συστήματος θα πραγματοποιηθεί, τοπικά, σε κάποιο ενδιάμεσο κόμβο ή στο σύννεφο. Με μια αποτελεσματική απόφαση κερδίζουμε σε τομείς όπως η **ταχύτητα εκτέλεσης και μεταφοράς**, καθώς και σε **εξοικονόμηση ενέργειας** που είναι λογικό να φαίνεται σε πιο μακροσκοπικό επίπεδο. Τέλος, είναι σημαντικό να εφαρμόσουμε τις κατάλληλες **τεχνικές μοντελοποίησης** ώστε **να μετρήσουμε ορθά την απόδοση** του καταμερισμού που πραγματοποιήσαμε προσφέροντας έτσι ορθά συμπεράσματα και **σταθερότητα.**

Μία προτεινόμενη λύση αποτελεί ο «άπληστος» αλγόριθμος SMSEF [1] ( Select Maximum Saved Energy First) ο οποίος, όπως αναφέρει και το όνομα του, επιλέγει κάθε φορά το offloading του εκάστοτε task ή subtask, αυστηρά με βάση την εξοικονόμηση της περισσότερης δυνατής ενέργειας. Ο συγκεκριμένος αλγόριθμος αφορά τη τεχνική MEC (Mobile edge computing) και ουσιαστικά υλοποιεί το offloading κινητών διεργασιών (με δυνατότητα διάσπασης) σε έναν MEC server. Αρχικά, κάνουμε την παραδοχή πως οι διεργασίες είναι διαιρέσιμες. Στη συνέχεια, συνθέτουμε την άπληστη απόφαση με δεδομένο το συγκεκριμένο μοντέλο [1] και τη μαθηματική φόρμουλα [1] που αναλύεται με βάση αυτό. Υλοποιώντας την απόφαση αυτή κάνουμε κάποιες βασικές δεσμεύσεις.

* Τα δεδομένα που στέλνουμε είναι ίσα με αυτά που υπολογίζονται (transmitted data = computed data)
* Τα δεδομένα πρέπει πρώτα να σταλθούν και μετά να υπολογιστούν
* Κάθε κινητό μπορεί να απασχολεί το πολύ ένα κανάλι ανά time slot ( με τον κανάλι εννοούμε ένα κομμάτι του συνολικού φάσματος με εύρος ζώνης Β (bandwidth))

Επιπλέον ο αλγόριθμος μας ακολουθεί τις εξής 2 πολιτικές:

* Η απασχόληση του διαθέσιμου time slot στον MEC server ξεκινά από το deadline, δηλαδή από την αρχή του ήδη δεσμευμένου χώρου, και πάει (δεσμεύει χώρο) προς τα πίσω.
  + Αυτό επιτυγχάνεται υπολογίζοντας το max latency που χρειάζεται το task/subtask που μας αφορά και κατόπιν διαιρώντας το, ώστε το 1ο διαιρεμένο μέρος να καλύπτει με καλύτερη ακρίβεια τον διαθέσιμο χώρο. Όσο πιο κοντά στο deadline, τόσο καλύτερη η διαίρεση που σημαίνει λιγότερες πιθανές συγκρούσεις [1]
* Τα υπόλοιπα διαιρεμένα μέλη (subtasks) ακολουθούν τη λογική επαναλαμβανόμενου water filling μέχρι να σταλθούν και να υπολογιστούν όλα 1 προς 1

Προχωρώντας στο optimization και στον αλγόριθμο: 1) θεωρούμε ως Υi την στρατηγική Y του mobile (χρήστη) i, ώστε να επιλέξει το κατάλληλο κανάλι και το κατάλληλο occupation slot (σε χρόνο) του MEC server. Επομένως για τη στρατηγική που απασχολεί το κανάλι C σε υπολογιστικό χρόνο S έχουμε το μοντελοποιημένο σύστημα Υ = [C,S]. 2) Θέτουμε y\* τη βέλτιστη λύση, όπου y\* = [y\*1,…y\*N] , την επιλέγουμε (έχουμε επιλέξει δηλαδή κατάλληλα task για offloading, κατάλληλα κανάλια και σε συγκεκριμένα time slot) και τέλος την αφαιρούμε από το σύνολο των Υ στοιχείων. 3)Επαναλαμβάνουμε τη διαδικασία για κάθε χρήστη. 4) Θεωρούμε F(i,Y) τη μέγιστη ενέργεια που εξοικονομήθηκε για i tasks στο σύστημα Υ. Η πολυπλοκότητα του αλγορίθμου είναι Ο(Α\*Ν2).

Σημαντική σημείωση: Η λογική αυτή αρχικά παρουσίαζε μικρή διαφορά αποτελεσμάτων αν άλλαζε η αλληλουχία με την οποία γινόντουσαν offload τα tasks. Δηλαδή, Υik != Yki διότι κάθε φορά, ο αλγόριθμος λειτουργεί άπληστα υπέρ της πρώτης διεργασίας. Για την επίλυση του προβλήματος, υιοθετήθηκε η λογική της επαναλαμβανόμενης άπληστης απόφασης, δηλαδή κάθε φορά το υποσύστημα που προκύπτει από μια επιλογή, θεωρείται νέο σύστημα ανεπηρέαστο από το προηγούμενο.

Τέλος, για την προσομοίωση και τη μέτρηση απόδοσης, χρησιμοποιούμε άλλους 2 αλγορίθμους: τον Priority-based Algorithm (Prio) και τον Non-divisable SMSEF (N-SMSEF) του οποίου η διαφορά με τον SMSEF είναι πως οι διεργασίες δεν διαιρούνται.

***Συμπέρασμα:***  Υλοποιώντας προσομοιώσεις και συγκρίνοντας τα αντίστοιχα figures, φαίνεται ότι για τυχαίο αριθμό χρηστών και διεργασιών ανά χρήστη, ο αλγόριθμος SMSEF υπερτερεί εμφανώς των άλλων 2 σε ένα MEC σύστημα και η υπεροχή του δεν επηρεάζεται καθόλου από τον αριθμό των χρηστών και των διεργασιών σε κάθε δυνατή περίπτωση.

***Συμπεράσματα/ Μελλοντικές κατευθύνσεις***

Με βάση τις επιστημονικές μελέτες που αναλύσαμε, αλλά παρατηρώντας παράλληλα και τις τεχνολογικές εξελίξεις της πληροφορικής, γίνεται αντιληπτό πως ο ίδιος ο κλάδος καταφέρνει να σχεδιάσει και να προτείνει λύσεις σχετικά με τη βελτιστοποίηση ήδη υπαρχουσών τεχνικών ή ακόμα και την δημιουργία νέων. Όπως φαίνεται οι λύσεις αυτές απαντούν όλο και περισσότερο και με ποικίλους τρόπους, στα προβλήματα που συναντάμε πολύ συχνά μπροστά μας στο τομέα του cloud computing και συγκεκριμένα του task offloading. Συγκεκριμένα στις παραπάνω μελέτες, έχουμε προτεινόμενες λύσεις ως προς την εξοικονόμηση ενέργειας, την σημαντική μείωση χρόνου ασφαλούς εκτέλεσης διεργασιών και τη καλύτερη διαχείριση αυτών ως προς του πόρους που διαθέτουμε, αντίστοιχα. Είναι σαφές πως κάθε μία από τις προαναφερθείσες μελέτες αφορούν ένα πιο ειδικό πλαίσιο, πχ συγκεκριμένη τοπολογία δικτύου ή ειδικευμένη εφαρμογή για επικοινωνία ανθρώπου οχήματος, όμως έχει σημασία να κοιτάξουμε την ευρύτερη εικόνα. Ο αλγόριθμος εξοικονόμησης ενέργειας διαθέτει ιδιαίτερα μεγάλη επεκτασιμότητα. Ως αυτόνομος μηχανισμός, μπορεί να υιοθετηθεί αυτούσιος ή παραμετροποιημένος για κάθε μελλοντικό παρόμοιο σύστημα σε ένα κόσμο cloud που όσο εξελίσσεται είναι επιτακτική ανάγκη να μειώνει αναλόγως τη κατανάλωση του. Στη περίπτωση του MANET, προκύπτει πως μια απλή προσθήκη εξελιγμένου και υποστηριζόμενου από αυτή τη τεχνολογία interface, μπορεί αποδοτικότερα να αντικαταστήσει τη χρήση Bluetooth ως μέσο σύνδεσης σε αυτό. Οι κύριες ιδέες πίσω από τις προτεινόμενες λύσεις είναι αυτές που τους χαρίζουν μια ιδιαίτερη δυναμική, καθώς μελλοντικές παρεμφερείς τεχνολογίες που θα αντιμετωπίζουν το ίδιο πρόβλημα, μπορούν να τις θέσουν ως βάση και με τη σειρά τους καλούνται να λάβουν και να κινήσουν περαιτέρω τη σκυτάλη.