ΜΕΛΕΤΗ ΚΩΔΙΚΟΠΟΙΗΣΕΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΔΙΚΤΥΩΝ ΣΥΜΠΙΕΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΣΕ ΔΙΚΤΥΑΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

ΜΙΧΑΗΛ Ε. ΟΙΚΟΝΟΜΑΚΗΣ

ΤΕΧΝΟΛΟΓΟΣ ΜΗΧΑΝΙΚΟΣ Η/Υ ΣΥΣΤΗΜΑΤΩΝ, ΤΕΙ ΠΕΙΡΑΙΑ, 2000

διατριβή

για τη μερική ικανοποίηση των απαιτήσεων για την απόκτηση του τίτλου

MASTER OF SCIENCE

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

2020

εγκεκριμένη:

Δρ. Ευάγγελος Κ. Μαρκάκης

Σύνοψη:

Ο βασικός στόχος αυτής της διπλωματικής είναι η μελέτη της κωδικοποιημένης υπολογιστικής (coded computing) και δύο τεχνικών κωδικοποίησης δικτύου (network coding) μία με γραμμική κωδικοποίηση και μία στην οποία εφαρμόζουμε στα δεδομένα προς μετάδοση ομομορφική κωδικοποίηση δικτύου.

Τα συστήματα που έχουν κατασκευαστεί για τα δίκτυα (ασύρματα, κινητά και διαδικτυακά) επικοινωνίας ψηφιακών συσκευών, με το πέρασμα του χρόνου φάνηκε ότι δεν είναι αρκετά ισχυρά για αξιόπιστη και αποτελεσματική μεταφορά και επεξεργασία του τύπου των δεδομένων με τα οποία εργαζόμαστε. Επίσης η προσπάθεια επεξεργασίας δεδομένων για μηχανική μάθηση οδηγεί σε προβλήματα συμφόρησης.

Το Coded Computing φαίνεται να δίνει λύση σε αυτά τα προβλήματα εφαρμόζοντας τη θεωρία κωδικοποίησης, τη ραχοκοκαλιά του σχεδιασμού των αρχικών δικτύων επικοινωνίας, στα πληροφοριακά συστήματα. Αυτή η προσέγγιση χρησιμοποιεί θεμελιώδη μαθηματικά που μπορούν να εφαρμοστούν σε ένα ευρύ φάσμα υπολογισμών όπως για παράδειγμα στην εκπαίδευση deep neural networks που είναι το βασικό εργαλείο της μηχανικής μάθησης. Σε αυτήν την εργασία αρχικά θα αναλύσουμε και στη συνέχεια θα εξετάσουμε τις πιθανές ομοιότητες και διαφορές ανάμεσα στην υπολογιστική κωδικοποίηση και την κωδικοποίηση δικτύου. Από τα αποτελέσματα παρατηρούμε για τη μεν υπολογιστική κωδικοποίηση, την βελτιστοποίηση της ταχύτητας μετάδοσης δεδομένων ενώ για τις τεχνικές κωδικοποίησης δικτύου, βλέπουμε αύξηση της παρεχόμενης αξιοπιστίας της μετάδοσης χωρίς να παρατηρούνται μεταβολές στον χρόνο που απαιτείται αυξάνοντας τον αριθμό των κόμβων. Ειδικότερα για την περίπτωση της ομομορφικής κωδικοποίησης έχουμε και αύξηση της παρεχόμενης ασφάλειας, στην περίπτωση προσπάθειας υποκλοπής των δεδομένων. Πρέπει να σημειώσουμε ότι στη μελέτη που επιχειρούμε δεν μας ενδιαφέρουν τόσο οι απόλυτοι χρόνοι επεξεργασίας και μετάδοσης όσο η μεταβολή που παρατηρείται σε αυτούς με την αύξηση των κόμβων, καθώς και η παράλληλη μεταβολή της χρήσης των δικτυακών πόρων των γραμμών μεταφοράς. Μία μελλοντική εργασία θα μπορούσε να παρουσιάσει τα πιθανά επόμενα βήματα στο συνδυασμό των τεχνικών κωδικοποίησης που εξετάζονται, προκειμένου να υπάρξει μία νέα τεχνική η οποία να συνδυάζει τα παρεχόμενα οφέλη.

Abstract:

The main goal of this dissertation is the study of coded computing and two network coding techniques, one with linear coding and one in which we apply homomorphic network coding to the data to be transmitted. The systems that have been built for the networks (wireless, mobile and internet) for the communication of digital devices, over time, seemed that they were not "strong" enough for reliable and efficient transfer and process of the type of data with which we work. Attempting to process data for machine learning also leads to congestion problems. Coded Computing seems to solve these problems by applying coding theory, the backbone of the design of original communication networks, to information systems. This approach uses fundamental mathematics that can be applied to a wide range of calculations such as deep neural networks training, which is the basic tool of machine learning. In this work we will first analyze and then examine the possible similarities and differences between computer coding and network coding. The results concerning computer coding yielded an optimization of the data transmission speed, while the network coding techniques had an increase of the provided transmission reliability without any changes in the time required, by increasing the number of nodes. Especially in the case of homomorphic coding we have an increase in the security provided in the event of an attempt to intercept data. It should be noted that in our trialing examination study we are attempting, we are not so much interested in the absolute processing and transmission times as the change observed in them with the increase of nodes, as well as the parallel change in the use of the network resources of the transmission lines. A future paper could present the possible next steps in combining the coding techniques under consideration, in order to propose a new technique that unites all of the provided benefits.

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

Λίστα Διαγραμμάτων

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε το διάστημα μεταξύ Νοεμβρίου 2019 και Αυγούστου 2020 στα πλαίσια του μεταπτυχιακού προγράμματος στην Πληροφορική του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ελληνικού Μεσογειακού Πανεπιστημίου. Ως την ελάχιστη δυνατή μνεία, με την παρούσα παράγραφο οφείλω να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνησή της και ιδιαίτερα:

Τον επιβλέποντα καθηγητή μου, Δρ. Ευάγγελο Μαρκάκη και τον Δρ. Ηλία Πολίτη, για την πολύτιμη υποστήριξή τους, τις παραγωγικές υποδείξεις τους και το πολύ καλό κλίμα συνεργασίας που διαμόρφωσαν συμβάλλοντας τα μέγιστα για την κατάρτιση της διπλωματικής μου εργασίας, τον ερευνητή του Πανεπιστημίου Πατρών Vipindev Adat ο οποίος μου παρείχε σημαντική βοήθεια στην πραγματοποίηση των μετρήσεων για τις μεθόδους Random Linear Network Coding και Homomorphic Coding και φυσικά τη σύζυγο και τα παιδιά μου που χωρίς την υπομονή τους και τη συμπαράστασή τους τίποτα από όλα αυτά δεν θα ήταν δυνατόν.

Εισαγωγή

Τα τελευταία χρόνια έχει διαπιστωθεί ότι τα διαθέσιμα συστήματα δικτύων και Η/Υ φαίνεται ότι δεν είναι αρκετά ισχυρά ώστε να έχουν τη δυνατότητα να μεταφέρουν γρήγορα και αξιόπιστα τα δεδομένα που χρησιμοποιούμε. Σε αυτό συντελούν και οι τελευταίες εξελίξεις στον τομέα του υπολογιστικού νέφους και ομίχλης (fog) καθώς και του edge computing που δημιουργούν μεγάλους όγκους δεδομένων προς μεταφορά, οι οποίοι με τη σειρά τους δημιουργούν συμφορήσεις στα δίκτυα δεδομένων. Τα παραπάνω οδηγούν σε ένα πρόβλημα στο οποίο καλείται να δώσει λύση το "*coded computing*" [1]. Τα τελευταία δύο χρόνια στη διεθνή βιβλιογραφία και στις αντίστοιχες δημοσιεύσεις φαίνεται μία τάση για ολοένα και περισσότερες σχετικές αναφορές. Θεωρείται δε από πολλούς σαν ζωτικής σημασίας για την εξέλιξη του διαδικτύου και των υπηρεσιών που το χρησιμοποιούν, τόσο σε επίπεδο προσωπικών υπολογιστών όσο και στο επίπεδο των "έξυπνων" φορητών συσκευών (smartphones, tablets κ.λπ.). Χρησιμοποιώντας την *coding theory*, η οποία ήδη έχει εφαρμογή στο σχεδιασμό των δικτύων τηλεπικοινωνιών, πάνω στα πληροφοριακά συστήματα, επιτυγχάνεται, σύμφωνα πάντα με τον *Salman Avestimehr* και την ομάδα του [1], να δοθεί μία πρόταση ως λύση στο πρόβλημα που αναφέρθηκε παραπάνω. Στην ουσία ανέπτυξαν ένα νέο πεδίο έρευνας που ονόμασαν "*polynomial coded computing"* (πολυώνυμος κωδικοποιημένος υπολογισμός) ο οποίος επειδή χρησιμοποιεί βασικές και θεμελιώδης έννοιες των μαθηματικών, μπορεί να χρησιμοποιηθεί σε ένα μεγάλο εύρος εφαρμογών, όπως το *edge* και το *fog computing*. Ένα ακόμα παράδειγμα εφαρμογής είναι τα "*deep neural networks"* που αποτελούν ένα από τους άξονες της μηχανικής μάθησης (*machine learning*). Ο πολυωνυμικός κωδικοποιημένος υπολογισμός μπορεί να διαχειριστεί πολλά, μεγάλης κλίμακας κατανεμημένα υπολογιστικά προβλήματα, στα οποία εμπεριέχονται μεταξύ άλλων και αλγόριθμοι μηχανικής μάθησης και να τα αναγάγει στο πρόβλημα της πολυωνυμικής παρεμβολής [2]. Συγκρινόμενο με το network coding [3], [4] η διαφορά εντοπίζεται στο ότι εδώ έχουμε εφαρμογή στα πληροφοριακά συστήματα αυτά κάθε αυτά και όχι στα δίκτυα τηλεπικοινωνιών. Η βασική αρχή στην κωδικοποίηση δικτύου (*network coding)*, είναι ότι οι ενδιάμεσοι κόμβοι μπορούν να στείλουν πακέτα που είναι γραμμικοί συνδυασμοί των πληροφοριών που έχουν λάβει προηγουμένως. Υπάρχουν δύο βασικά οφέλη αυτής της προσέγγισης: Η πιθανή βελτίωση της απόδοσης του δικτύου στον τομέα της ταχύτητας μετάδοσης δεδομένων καθώς και υψηλός βαθμός ευρωστίας. Με τον όρο ευρωστία εννοούμε

μια μεγαλύτερη ανοχή στις απώλειες. Έτσι γίνεται ευκολότερη η σχεδίαση αλγορίθμων που έχουν καλή απόδοση, ακόμη και αν τα παραγόμενα αποτελέσματα προκύπτουν από ένα σχετικά μικρό όγκο δεδομένων. Η παρούσα εργασία θα προσπαθήσει να εξετάσει και να βρει το πεδίο αλληλοσύνδεσης των δύο μεθόδων "*coding*" και να δώσει μία απάντηση σχετικά με την αποδοτικότητα της καθεμιάς καθώς και τα γενικότερα πλεονεκτήματα και μειονεκτήματά της. Εκτελώντας διαφορετικά πειράματα θα επιχειρήσουμε να δείξουμε τα σημεία υπεροχής της μίας μεθόδου έναντι της άλλης και τα κοινά σημεία. Παρόλο που με μία πρώτη ανάγνωση οι δύο μέθοδοι φαίνεται να μην έχουν κοινό σημείο εφαρμογής, οι σχετικά πρόσφατες εξελίξεις στον τομέα των δικτύων Η/Υ δείχνουν πως αυτό δεν ισχύει. Η αυξημένη υπολογιστική ισχύ στις συσκευές δικτύου οι οποίες πλέον μπορούν και να προγραμματίζονται ανάλογα με τις ανάγκες που καλούνται να εξυπηρετήσουν (*software defined networks*) κάνουν πλέον δυσδιάκριτες τις διαφορές ανάμεσα στο λογισμικό που καλείται να εξυπηρετήσει ένα κατανεμημένο σύστημα από κόμβους Η/Υ και στο λογισμικό που καλείται να εφαρμοστεί από κόμβους που εξυπηρετούν τη λειτουργία ενός δικτύου.

Κεφάλαιο 1 - Coded Computing (κωδικοποιημένη υπολογιστική)

Οι συγγραφείς στο [1] προσπαθούν να αξιοποιήσουν έννοιες και εργαλεία από τη θεωρία κωδικοποίησης και πληροφορίας προκειμένου να δημιουργήσουν πλεόνασμα υπολογιστικής ισχύος καταφέρνοντας έτσι να ξεπεράσουν τα εμπόδια επικοινωνίας και αποδιοργάνωσης, Παραθέτουν σχετικά στοιχεία για την σπατάλη χρόνου που υπάρχει σε εμπορικά κατανεμημένα συστήματα. Για παράδειγμα, παρατηρήθηκε σε ένα Hadoop cluster του Facebook, ότι ένα σχεδόν 33% του συνολικού χρόνου εκτέλεσης σπαταλιέται στο data shuffling ενώ στην περίπτωση ενός Amazon EC2 cluster, ο οποίος τρέχει μία *self-join* εφαρμογή, το ποσοστό αυτό φτάνει στο 70% [1]. Προσπαθώντας να μειώσουν τις συγκεκριμένες χρονικές σπατάλες στα συστήματα κατανεμημένης υπολογιστικής αυτού του τύπου αναρωτήθηκαν αν με την κωδικοποίηση θα μπορούσαν να μειώσουν το φορτίο επικοινωνίας, επιταχύνοντας έτσι την διαδικασία. Θεμελίωσαν έτσι μία σχέση ανταλλαγής ανάμεσα στο υπολογιστικό φορτίο για τη φάση *Map* και το φορτίο επικοινωνίας για τη φάση *Data Shuffling* τα οποία και απέδειξαν σαν αντιστρόφως ανάλογα. Πρότειναν ένα βέλτιστο σχήμα κωδικοποίησης το οποίο και ονόμασαν Κωδικοποιημένη Κατανεμημένη Υπολογιστική *Coded Distributed Computing (CDC)* η οποία δείχνει ότι αυξάνοντας το υπολογιστικό φορτίο στη φάση *Map* κατά ένα συντελεστή r μπορεί να οδηγήσει σε νέες δυνατότητες στη φάση *Data Shuffling*, οι οποίες μειώνουν το φορτίο επικοινωνίας κατά τον ίδιο συντελεστή r .

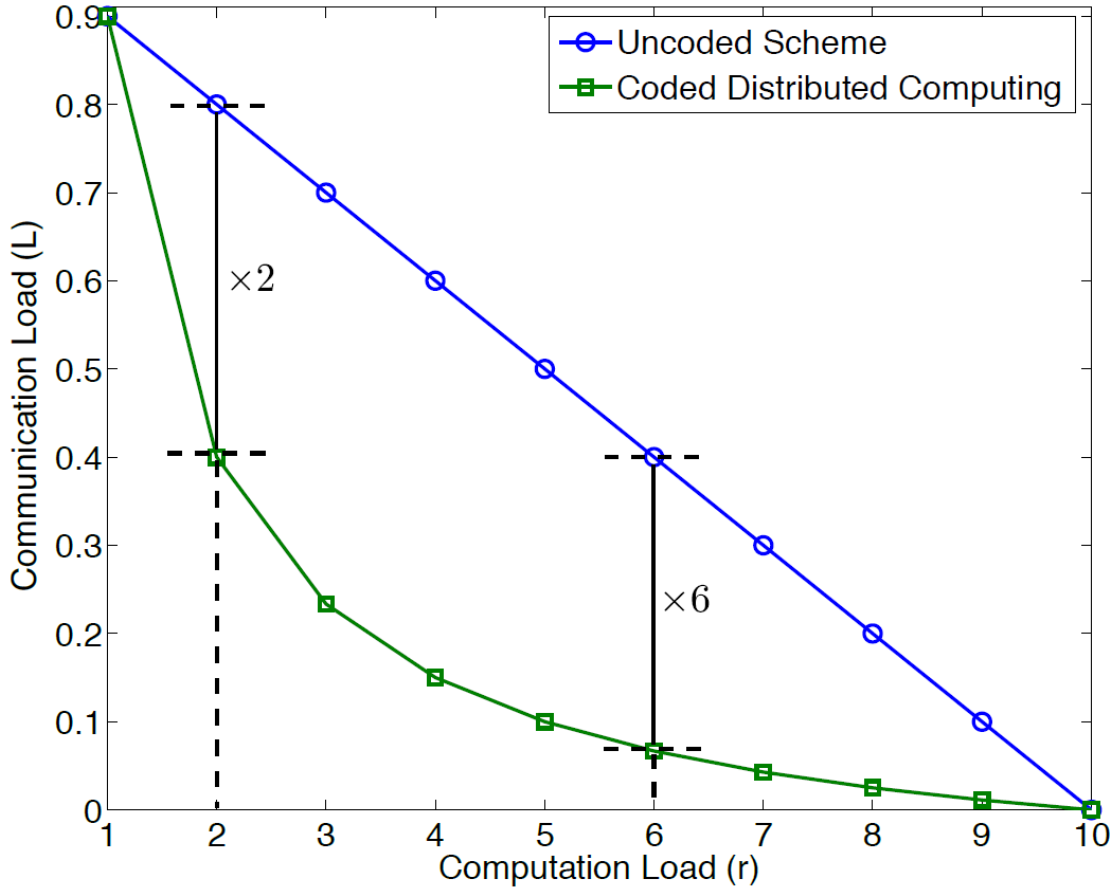
1.1 Εισαγωγή και Σχετικές Δημοσιεύσεις.

Για να εξηγήσουν τα αποτελέσματά τους, θεώρησαν ένα κατανεμημένο υπολογιστικό πλαίσιο το οποίο υπολογίζει Q τυχαίες συναρτήσεις από N αρχεία εισόδου, χρησιμοποιώντας K κατανεμημένους υπολογιστικούς κόμβους. Όπως αναφέραμε προηγουμένως η τελική εκτίμηση γίνεται με την επεξεργασίας και την εξαγωγή αποτελεσμάτων από ένα αριθμό συναρτήσεων *Map* και *Reduce*. Στη φάση *Map* κάθε αρχείο επεξεργάζεται τοπικά σε κάθε κόμβο δημιουργώντας Q ενδιάμεσες τιμές (Q συναρτήσεις θα δώσουν αντίστοιχα Q τιμές) με την καθεμιά από αυτές να προέρχεται από την αντίστοιχη Q συνάρτηση εξόδου. Με την πρόοδο της διαδικασίας έχουμε αποτελέσματα για QN μεταβατικές εξόδους οι οποίες μπορούν να διακριθούν σε Q ξεχωριστά μέρη από N μεταβατικές εξόδους. Το κάθε υποσύνολο είναι

απαραίτητο για τον υπολογισμό κάθε μίας από τις συναρτήσεις εξόδου. Η διαδικασία *Shuffle* μεταφέρει αυτές τις N μεταβατικές εξόδους στους κόμβους που τους αναλογούν προκειμένου να ξεκινήσει η διαδικασία *Reduce*. Για κάθε κόμβο που επιλέγεται, ένα μέρος των μεταβατικών εξόδων που αναφέραμε παραπάνω έχουν προωθηθεί ήδη και δεν χρειάζεται να προωθηθούν στην επόμενη φάση (*Shuffle*). Επειδή η διαδικασία *Map* έχει ήδη «τρέξει» στους ίδιους κόμβους και η παραγόμενη έξοδος παραμένει εκεί μέχρι την εκτέλεση της επόμενης διαδικασίας (*Reduce*). Αυτό μειώνει έστω και λίγο το φορτίο επικοινωνίας. Για να μειωθεί το φορτίο επικοινωνίας ανάμεσα στους κόμβους ακόμα περισσότερο πρότειναν το κάθε αρχείο εισόδου να γίνεται *Map* σε περισσότερους από έναν κόμβους. Προφανώς αυτό αυξάνει το σύνολο των ενδιάμεσων τιμών που κρατάει ο κάθε κόμβος καθώς και το μέγεθός της επεξεργασίας που θα κάνει στα δεδομένα εισόδου αφού πλέον είναι περισσότερα. Παρόλα αυτά, όπως απέδειξαν, θα μπορούσε να δημιουργηθεί μία βελτιωμένη μέθοδος προκειμένου η αύξηση στους υπολογισμούς να βοηθήσει προκειμένου να επιτευχθεί μείωση στο φορτίο που αναπτύσσεται κατά τη μεταφορά των δεδομένων (επικοινωνιακό). Το συμπέρασμα στο οποίοι ήθελαν να καταλήξουν είναι ότι μετά από μια συγκεκριμένη προσπάθεια επανάληψης των *Map* υπολογισμών χρησιμοποιώντας παράλληλα τεχνικές κωδικοποίησης, μπορούν να μειώσουν σημαντικά το φορτίο της επικοινωνίας.

Έθεσαν σαν υπολογιστικό φορτίο r το σύνολο των υπολογιζόμενων *Map* συναρτήσεων το οποίο ορίζεται από των αριθμό των αρχείων N, όπου ισχύει: $1 \leq r \leq K$ όπου K ο αριθμός των κόμβων. Για παράδειγμα αν $r=1$ αυτό σημαίνει ότι η κάθε συνάρτηση υπολογίζεται μόνο σε ένα κόμβο K, ενώ για $r=2$ η κάθε συνάρτηση *Map* υπολογίζεται σε δύο κόμβους. Επίσης όρισαν σαν φορτίο επικοινωνίας L το σύνολο των δεδομένων που μεταφέρεται μεταξύ των κόμβων κατά τη διάρκεια της διαδικασίας *Shuffling*. Το φορτίο επικοινωνίας είναι καθορισμένο από το μέγεθος των μεταβατικών τιμών QN, για να γίνουν οι απαραίτητοι υπολογισμοί από τους οποίους θα προκύψουν οι συναρτήσεις εξόδου Q. Τα αποτελέσματα των συναρτήσεων παράγονται σε κάθε ένα κόμβο από το σύνολο K αυτόνομα. Στη συνέχεια πρότειναν ένα *Coded Distributing Computing (CDC)* σχήμα το οποίο πετυχαίνει ένα επικοινωνιακό φορτίο $L_{coded}(r) = \frac{1}{r} \cdot (1 - \frac{r}{K})$ για r=1,…,K. Ένα *Uncoded Distributed Computing* σχήμα, όπως έδειξαν [1], πετυχαίνει ένα επικοινωνιακό φορτίο $L_{uncoded}(r) = 1 - \frac{r}{K}$ και συνεπώς

έχουμε βελτίωση του επικοινωνιακού φόρτου κατά ένα συντελεστή $\frac{1}{r}$. Δηλαδή το επικοινωνιακό φορτίο καταλήγει να είναι αντιστρόφως ανάλογο με το υπολογιστικό φορτίο, όπως φαίνεται στο παρακάτω διάγραμμα, όπου γίνεται η σύγκριση του φορτίου επικοινωνίας που επιτεύχθηκε από τον κωδικοποιημένο διανεμημένο υπολογιστικό υπολογιστή $L_{coded}(r)$, με αυτό του μη κωδικοποιημένου σχήματος $L_{uncoded}(r)$. Το πείραμα στο οποίο αναφερόμαστε πραγματοποιήθηκε για δέκα συναρτήσεις εξόδου Q και δέκα υπολογιστικούς κόμβους K στους οποίους ανατέθηκαν 2520 αρχεία προς επεξεργασία, N. Για τιμές του r από 1 μέχρι K, το σχήμα της υπολογιστικής κωδικοποίησης καταλήγει να είναι r φορές καλύτερο από το μη κωδικοποιημένο σχήμα.



Διάγραμμα 1: Σύγκριση του τηλεπικοινωνιακού με το υπολογιστικό φορτίο με και χωρίς υπολογιστική κωδικοποίηση

Όπως φαίνεται, στο μη κωδικοποιημένο σχήμα μετάδοσης επιτυγχάνεται ένα φορτίο επικοινωνίας:

$$L_{uncoded}(r) = 1 - \frac{r}{K}$$

Η αύξηση του υπολογιστικού φορτίου r προσφέρει τελικά μόνο μια μέτρια μείωση του φορτίου επικοινωνίας. Στην πραγματικότητα για οποιοδήποτε r (αριθμός των κόμβων όπου γίνεται ο υπολογισμός της κάθε συνάρτησης), όταν ο αριθμός των κόμβων K γίνει πολύ μεγάλος αυτό το κέρδος εξαφανίζεται. Φαίνεται δηλαδή ότι η αύξηση της υπολογιστικής ισχύς στους κόμβους προκειμένου να βελτιστοποιηθεί η ταχύτητα επικοινωνίας, χωρίς τη χρήση τεχνικών κωδικοποίησης δεν οδηγεί σε ουσιαστικό όφελος. Ωστόσο, για το κωδικοποιημένο σχήμα που επιτυγχάνει ένα φορτίο επικοινωνίας ίσο με:

$$L_{coded}(r) = \frac{1}{r} \cdot \left(1 - \frac{r}{K}\right) = \frac{1}{r} \cdot L_{uncoded}(r)$$

είναι φανερό ότι η αύξηση του φορτίου υπολογισμού r θα μειώσει σημαντικά το φορτίο επικοινωνίας και αυτό το κέρδος δεν εξανεμίζεται για μεγάλες τιμές του K (αριθμός κόμβων). Για παράδειγμα, όπως φαίνεται στο Διάγραμμα 1: *Σύγκριση του τηλεπικοινωνιακού με το υπολογιστικό φορτίο με και χωρίς υπολογιστική κωδικοποίηση*, κατά τη χαρτογράφηση κάθε αρχείου σε έναν επιπλέον κόμβο ($r = 2$), το *Coded Distributed Computing* μειώνει το φορτίο επικοινωνίας περίπου κατά 55,55%, ενώ το μη κωδικοποιημένο σχήμα το μειώνει μόνο κατά περίπου 11,11%. Απέδειξαν επίσης την ύπαρξη ενός θεωρητικού κατώτατου ορίου στο ελάχιστο φορτίο επικοινωνίας $L^*(r)$. Υπάρχει ένα βέλτιστο σημείο στο οποίο υπάρχει η μέγιστη απόδοση τόσο από πλευράς υπολογιστικού φορτίου όσο και από πλευράς τηλεπικοινωνιακού φορτίου. Για να αποδειχθεί το κατώτερο όριο, εξήγαγαν ένα κατώτερο όριο στον συνολικό αριθμό bits που μεταδίδονται στο δίκτυο, από οποιοδήποτε υποσύνολο κόμβων, χρησιμοποιώντας επαγωγή στο μέγεθος του υποσυνόλου. Θεωρήθηκε ένα ελάχιστο όριο στον αριθμό των δυαδικών ψηφίων που ο κάθε κόμβος απαιτεί για την ανάκτηση των μεταβατικών τιμών, προκειμένου να υπολογιστεί το κατώτερο όριο για ένα συγκεκριμένο υποσύνολο κόμβων. Αυτό το όριο χρησιμοποιείται προκειμένου να υπολογιστούν τα αποτελέσματα από τις ήδη γνωστές συναρτήσεις εξόδου. Από εκεί και μετά αυτό το ελάχιστο όριο θεωρείται κοινό και για τους υπόλοιπους κόμβους σύμφωνα με την επαγωγική λογική. Το εξαχθέν κατώτερο όριο στο $L^*(r)$ ταιριάζει με το φορτίο επικοινωνίας που επιτεύχθηκε από το σχήμα *Coded Distributed Computing* (*CDC*) για οποιοδήποτε φορτίο υπολογισμού $1 \leq r \leq K$. Σαν αποτέλεσμα,

χαρακτήρισαν ακριβώς τη βέλτιστη αντιστάθμιση μεταξύ του φορτίου υπολογισμού και του φορτίου επικοινωνίας στην παρακάτω παράσταση:

$$L^*(r) = L_{coded}(r) = \frac{1}{r} \cdot \left(1 - \frac{r}{K}\right), r \in \{1, \dots, K\}$$

Γενικά για $1 \leq r \leq K$, το $L^*(r)$ είναι το ελάχιστο σημείο των παραπάνω σημείων $\{r, L_{coded}(r)) \ : r \in \{1, \dots, K\}\}$. Από το παραπάνω είναι φανερό ότι για μεγάλες τιμές του K το κλάσμα $\frac{r}{K}$ τείνει στο 0 οπότε και το $L^*(r)$ τείνει στο $\frac{1}{r}$.

$$\boldsymbol{L^*(r) \approx \frac{1}{r}.}$$

Επιπλέον αυτό το τελευταίο μας δείχνει ότι τα μεγέθη των υπολογιστικών και των τηλεπικοινωνιακών φορτίων είναι αντιστρόφως ανάλογα και ότι το κέρδος $\frac{1}{r}$ που επιτυγχάνεται από το *CDC* είναι το βέλτιστο και δεν μπορεί να βελτιωθεί περεταίρω από κάποια άλλη μέθοδο κωδικοποίησης από τη στιγμή που το $L_{coded}(r)$ είναι το θεωρητικό ελάχιστο της $L^*(r)$ και αυτό ισχύει για κάθε υλοποίηση *Data Shuffling* [1].

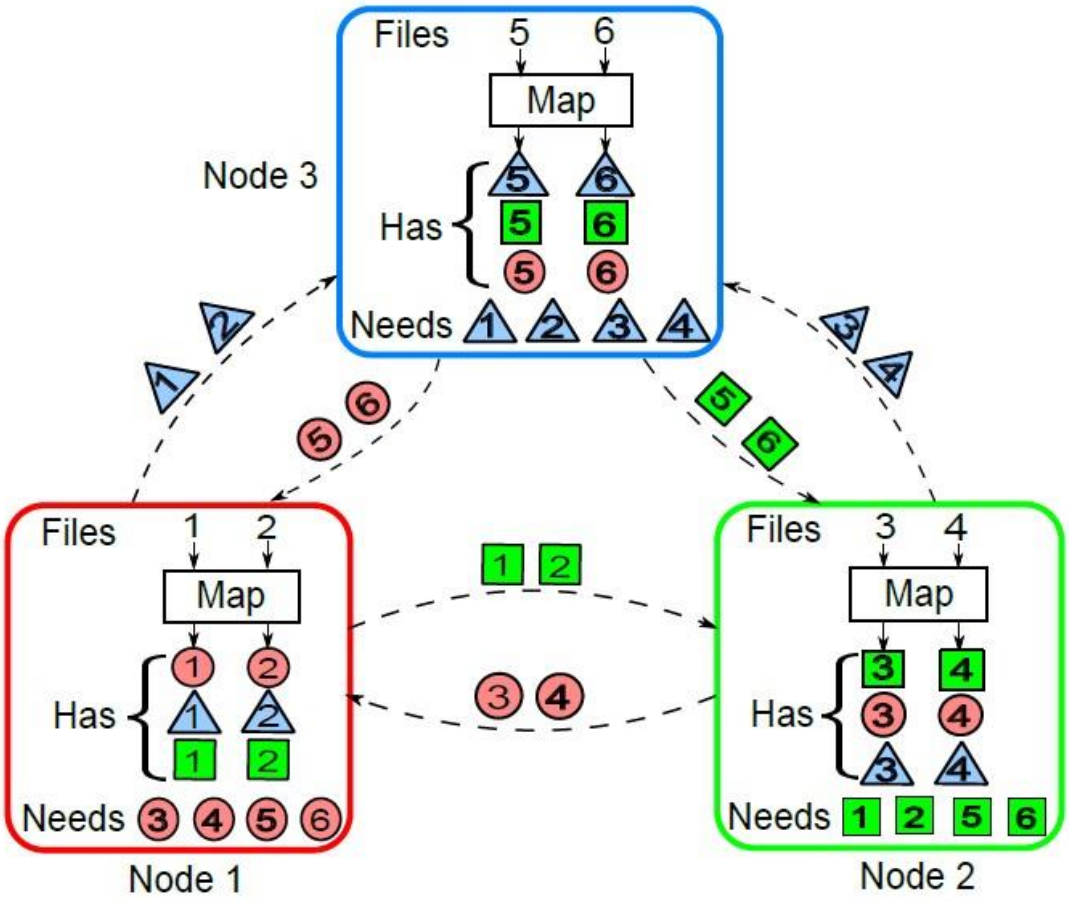
1.1.1 Σχετικές Δημοσιεύσεις

Η εισαγωγή της ιδέας του *Coded Computing* οδήγησε σε πολλές νέες υλοποιήσεις αλγορίθμων βασιζόμενες στη λογική του τόσο από τους αρχικούς συγγραφείς όσο και από άλλους. Για παράδειγμα στο [5] οι συγγραφείς παρουσιάζουν ένα τρόπο με τον οποίο η κωδικοποίηση μπορεί να χρησιμοποιηθεί αποτελεσματικά στο πεδίο της «υπολογιστικής ομίχλης» (*fog computing*) αξιοποιώντας την πλεονάζουσα υπολογιστική ισχύ στις άκρες του δικτύου προκειμένου να μειωθεί η κίνηση στο δίκτυο αλλά και η καθυστέρηση που προκαλείται από τους απαιτούμενους υπολογισμούς. Ο τομέας αυτός ανοίγει πολλές σημαντικές και ενδιαφέρουσες κατευθύνσεις έρευνας. Στο παραπλήσιο πεδίο του *edge computing* συναντήσαμε την έρευνα [6] η οποία εισάγοντας μία νέα μέθοδο δημιουργημένη από τους συγγραφείς επιτυγχάνει να μετριάσει τις καθυστερήσεις που παρατηρούνται σε μία υλοποίηση *edge computing*, στην οποία κάποιοι από τους κόμβους του δικτύου παρουσιάζουν υστέρηση στην επεξεργασία των δεδομένων σε σχέση με τους υπόλοιπους κόμβους. Κατάφεραν να ελαχιστοποιήσουν, κατά το δυνατόν, τις παρατηρούμενες καθυστερήσεις στις συνδέσεις του δικτύου ανάμεσα στους κόμβους και τον διαχειριστή. Παρόμοια μελέτη για τις περιπτώσεις στις οποίες οι κόμβοι του δικτύου δεν είναι ίσης υπολογιστικής ισχύος ή για διάφορους λόγους σε συγκεκριμένες χρονικές στιγμές μπορούν

να παρουσιάσουν καθυστερήσεις, αναπτύχθηκε από τους συγγραφείς στο [7]. Θέτοντας σαν στόχο το ταχύτερο αποτέλεσμα για δεδομένο πολλαπλασιασμό μεγάλων πινάκων κατέληξαν στο συμπέρασμα ότι η προτεινόμενη από αυτούς μέθοδος, την οποία και ονόμασαν *HCMM (Heterogeneous Coded Matrix Multiplication),* δίνει πολύ σημαντικές βελτιώσεις από 39% έως και 71% στο χρόνο που απαιτείται για την εύρεση αποτελέσματος, συγκρινόμενη με άλλες μεθόδους κωδικοποίησης. Οι συγγραφείς στο [8] βασισμένοι στο *coded computing* εισήγαγαν μία νέα τεχνική κωδικοποίησης η οποία βελτιώνει έστω και λίγο την ταχύτητα στα συστήματα που είναι βασισμένα στην *blockchain* κρυπτογράφηση. Πρόσφατα οι συγγραφείς στο [9] πήγαν παραπέρα την αρχική έρευνα από το [1] προτείνοντας ένα μεικτό σχήμα υπολογιστικής κωδικοποίησης στο οποίο χώρισαν τους κόμβους σε δύο ίσες ομάδες εφαρμόζοντας στην πρώτη ομάδα την μέθοδο από το [1] και στη δεύτερη μία καινοτόμο, όπως τη χαρακτηρίζουν, μέθοδο κωδικοποίησης δικής τους έμπνευσης. Απέδειξαν ότι το μεικτό συνδυαστικό σχήμα παρέχει τη βέλτιστη χρονική απόδοση (ελάχιστο χρόνο επικοινωνίας) τόσο για τη διαδοχική όσο και για την παράλληλη υλοποίηση. Πέρα από τα παραπάνω στο [10] οι συγγραφείς, χρησιμοποιώντας το πλαίσιο του *coded computing*, ανέπτυξαν μια νέα τεχνική η οποία εξασφαλίζει ταχύτερες αποκρίσεις σε συστήματα μηχανικής μάθησης, τα οποία αναλύουν δεδομένα προερχόμενα από αισθητήρες, δίνοντας παράλληλα εντολές, εκτελώντας τελικά μια πολυσύνθετη διαδικασία, όπως η αυτόνομη οδήγηση. Παρατηρήθηκε τελικά ότι οι μέθοδοι επανάληψης κωδικοποίησης (*repetition coding schemes*) απαιτούσαν σχεδόν τον δεκαπλάσιο χρόνο συγκρινόμενοι με τις μεθόδους γραμμικής παλινδρόμησης (*linear regression models*). Είναι λοιπόν προφανές ότι υπάρχει ένα πολλά υποσχόμενο πεδίο έρευνας στον τομέα του *coded computing* καθώς είναι φανερό ότι για μεγάλα μεγέθη δεδομένων προς επεξεργασία, ή τη μετάδοσή τους στην περίπτωση χρήσης *coded computing* τεχνικών, επιταχύνεται, ενίοτε σημαντικά, τόσο η ταχύτητα μετάδοσης όσο και η επεξεργασία των δεδομένων. Παρουσιάζεται δε πληθώρα μελετών και ερευνών στο συγκεκριμένο τομέα το όποιο μας δείχνε ότι αξίζει η περεταίρω διερεύνηση.

1.2 Coded Distributed Computing *αναθέτοντας* 2 διαφορετικά αρχεία σε κάθε κόμβο *(r=1).*

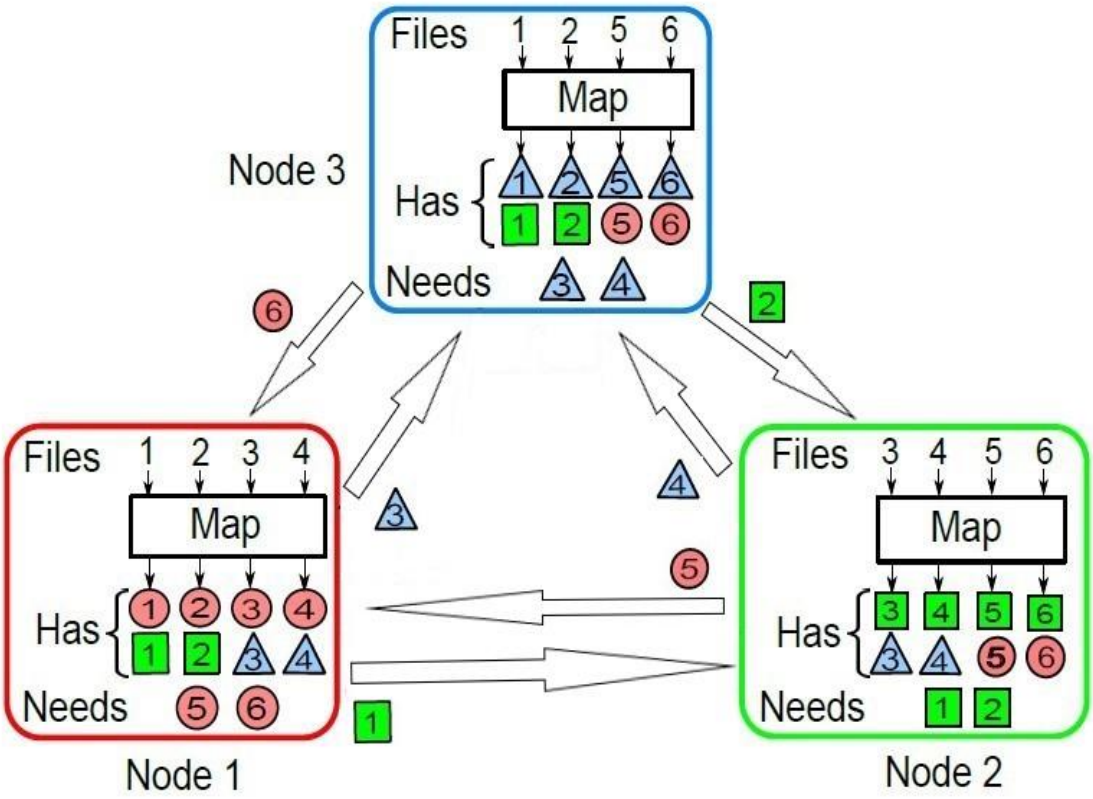
Έχουμε ένα κατανεμημένο δίκτυο που αποτελείται από τρεις διαφορετικούς κόμβους *(K=3)* το οποίο πρέπει να «τρέξει» τρεις διαφορετικές συναρτήσεις εξόδου *(Q=3)* για 6 διαφορετικά αρχεία εισόδου *(N=6)*. Ο κάθε κόμβος λαμβάνει δύο διαφορετικά αρχεία εισόδου (σε σχέση με τους άλλους δύο κόμβους, $r=1$) στα οποία εφαρμόζει τις τρεις συναρτήσεις (κόκκινος κύκλος, μπλε τρίγωνο, πράσινο τετράγωνο) δημιουργώντας έξι νέα αρχεία εξόδου. Γενικά ο κάθε κόμβος $k=(1, 2, 3)$ επεξεργάζεται το αρχείο $2k-1$ και το αρχείο $2k$. Μετά τη φάση *Map* ο κάθε κόμβος πρέπει να λάβει 2 από τα 6 αρχεία εξόδου από τον κάθε άλλο κόμβο (συνολικά 4) τα οποία απαιτούνται προκειμένου να εφαρμόσει τη συνάρτηση *Reduce* για την οποία είναι υπεύθυνος (κόμβος 1 κόκκινος κύκλος, κόμβος δύο πράσινο τετράγωνο και κόμβος 3 μπλε τρίγωνο). Ως εκ τούτου κάθε κόμβος χρειάζεται 4 αρχεία εξόδου, 2 από κάθε κόμβο δημιουργώντας ένα φορτίο επικοινωνίας τεσσάρων αρχείων επί τρεις κόμβους ίσο με 12. Αν όλα τα αρχεία έπρεπε να ληφθούν από όλους τους κόμβους χωρίς καν να εφαρμοστεί κατανεμημένη υπολογιστική το επικοινωνιακό φορτίο θα ήταν ίσο με 6 αρχεία επί 3 κόμβους ίσο με 18. Παίρνοντας τον λόγο των δύο τιμών καταλήγουμε σε ίσο με $\frac{2}{3} = 66{,}667\%$ του αρχικού.



Εικόνα 1: Uncoded Distributed Computing scheme

1.3 Coded Distributed Computing με τα αρχεία να ανατίθενται σε δύο κόμβους. (r=2).

Έχοντας τον ίδιο αριθμό κόμβων, αρχείων και συναρτήσεων εφάρμοσαν την ίδια μέθοδο με μία διαφοροποίηση. Τώρα ο κάθε κόμβος αναλαμβάνει 4 αρχεία αντί για 2.. Αυτό στην ουσία είναι ένας διπλασιασμός του υπολογιστικού φορτίου για τον κάθε κόμβο *(r=2).* Με αυτόν τον τρόπο ο κάθε κόμβος αντί να χρειάζεται τέσσερα αρχεία από τους άλλους δύο, πλέον χρειάζεται μόνο δύο. Έτσι δημιουργείται ένα φορτίο επικοινωνίας δύο αρχείων επί τρεις κόμβους ίσο με 6. Αν όλα τα αρχεία έπρεπε να ληφθούν από όλους τους κόμβους το επικοινωνιακό φορτίο θα ήταν ίσο με 6 αρχεία επί 3 κόμβους ίσο με 18. Με αυτή τη διαφοροποίηση έχουμε τα εξής:



Εικόνα 2: Ανάθεση κάθε αρχείου σε δύο κόμβους

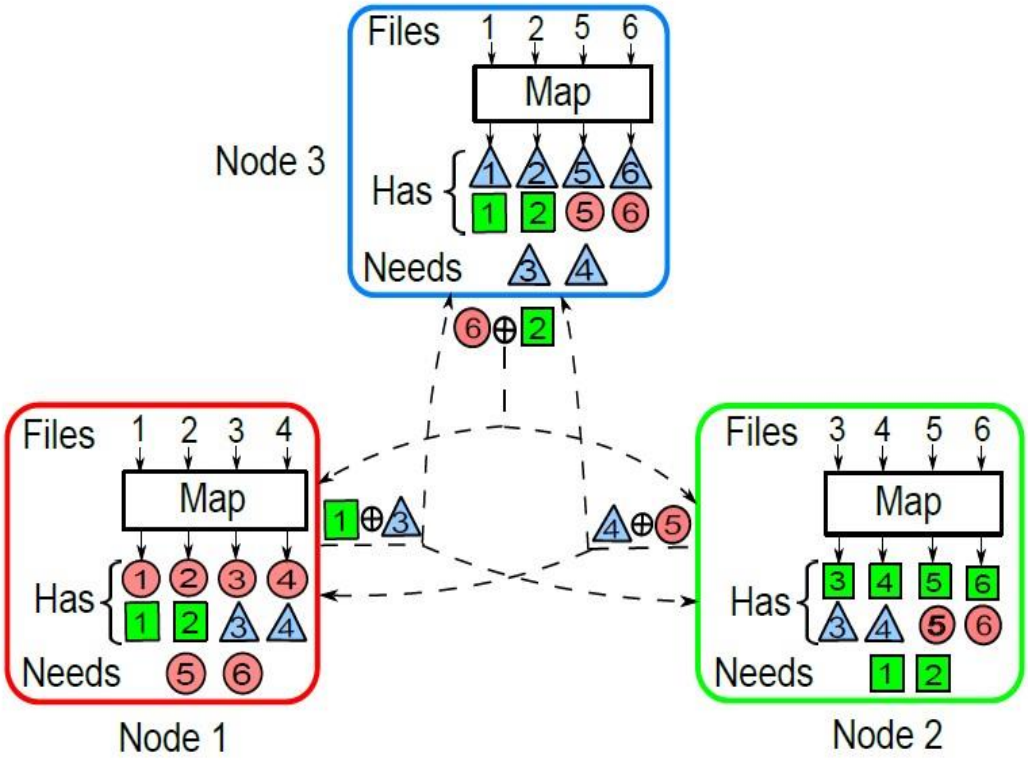
1. Ο κόμβος 1 θα στείλει στον κόμβο 2 το «πράσινο τετράγωνο 1» και στον κόμβο 3 το «μπλε τρίγωνο 3».
2. Ο κόμβος 2 θα στείλει στον κόμβο 1 τον «κόκκινο κύκλο 5» και στον κόμβο 3 το «μπλε τρίγωνο 4» και τέλος,
3. Ο κόμβος 3 θα στείλει στον κόμβο 1 τον «κόκκινο κύκλο 6» και στον κόμβο 2 το «πράσινο τετράγωνο 2».

Η παραπάνω ακολουθία δημιουργεί μεταφορά 6 αρχείων στο δίκτυο και σε σύγκριση με τα 18 αρχικά, καταλήγουμε σε ένα λόγο των δύο τιμών ίσο με $\frac{1}{3} = 33{,}333\%$ του «μέγιστου».

1.3.1 Εισάγοντας την τεχνική Coded Computing

Με την εφαρμογή του *Coded Computing* αυτό μπορεί να βελτιωθεί ακόμα περισσότερο επεμβαίνοντας στο τι εκπέμπει ο κάθε κόμβος. Οι συγγραφείς στο [1] επέλεξαν αντί ο κάθε κόμβος να εκπέμπει δύο ξεχωριστά αρχεία να εκπέμπει στο δίκτυο ένα νέο αρχείο το οποίο είναι αποτέλεσμα μια μαθηματικής πράξης ανάμεσα στα δύο αρχεία που ο κόμβος ήταν έτοιμος να

μεταδώσει αρχικά. Έτσι αντί να στέλνει στο δίκτυο δύο αρχεία, πλέον στέλνει μόνο ένα και το επικοινωνιακό φορτίο μειώνεται ακόμα περισσότερο σε 1 αρχείο επί τρεις κόμβους ίσο με 3. Καταλήγουμε έτσι σε ένα λόγο ίσο με $\frac{3}{18} = \frac{1}{6}$=16,667% του μέγιστου. Για παράδειγμα όπως φαίνεται και στην Εικόνα 3 ο κόμβος 3 επειδή γνωρίζει τον κόκκινο «κύκλο» για το αρχείο 5 μπορεί να το αναπαράγει από το αποτέλεσμα της πράξης *4 XOR 5* και έτσι να αποκτήσει το μπλε «τρίγωνο 4 που χρειάζεται. Όπως φαίνεται βέβαια η μείωση του επικοινωνιακού φορτίου έχει επιτευχθεί με την αύξηση του υπολογιστικού αφού πλέον ο κάθε κόμβος εκτελεί δύο επιπλέον "πράξεις".



Εικόνα 3: Coded Distributed Computing scheme

Γενικεύοντας το παραπάνω παράδειγμα [11] μπορούμε να θεωρήσουμε *K* κόμβους οι οποίοι συνεργάζονται προκειμένου να υπολογίσουν Q τυχαίες συναρτήσεις φ από *N* εισόδους μέσα από ένα πλαίσιο *Map-Reduce* για *q* από 1 έως Q. Να εξηγήσουμε σε αυτό το σημείο ότι ο μέσος όρος του αριθμού των κόμβων οι οποίοι εκτελούν τη διαδικασία *Map* είναι το υπολογιστικό φορτίο του συστήματος που μελετάμε. Ομοίως, το φορτίο επικοινωνίας ορίζεται σαν το σύνολο των ενδιάμεσων τιμών που απαιτούνται να ανταλλαχθούν ανάμεσα στους κόμβους κατά τη φάση του «αναμοιρασμού», προκειμένου να υπολογιστεί το σύνολο Q των συναρτήσεων φ. Στη

γενική περίπτωση, που προτείνεται στο [1] ο υπολογισμός της λειτουργίας *Map* επαναλαμβάνεται σε r επιλεγμένους κόμβους ούτως ώστε να τα αποτελέσματα που θα εκπέμψουν τελικά να είναι ταυτόχρονα χρήσιμα για κάποιους άλλους r κόμβους. Σαν αποτέλεσμα η Coded Map Reduce μειώνει το φορτίο επικοινωνίας κατά ένα παράγοντα ο οποίος είναι πάντα πολλαπλάσιο του υπολογιστικού φορτίου. Τελικά οι συγγραφείς στο [1] κατέληξαν στο ότι υπάρχει μία θεμελιώδης, όπως τη χαρακτηρίζουν, σχέση ανάμεσα στο υπολογιστικό και στο επικοινωνιακό φορτίο η οποία μπορεί να χρησιμοποιηθεί προκειμένου να επιτευχθεί μείωση του απαιτούμενου BW για την επικοινωνία των κόμβων με το να χρησιμοποιηθεί η αδιάθετη επεξεργαστική ισχύς του κάθε κόμβου.

$$\text{επικοινωνιακό φορτίο} \approx \frac{1}{\text{υπολογιστικό φορτίο}}$$

1.4 Πεδίο Εφαρμογής του Coded Computing

Ο TeraSort [12] είναι ένας συμβατικός αλγόριθμος για τη διανεμημένη ταξινόμηση μεγάλου όγκου δεδομένων. Τα δεδομένα που του ανατίθενται προς επεξεργασία είναι μορφοποιημένα σε ζευγάρια που αποτελούνται από κλειδιά και τιμές *(Key Value).* Για παράδειγμα, ο τομέας των κλειδιών μπορεί να είναι ακέραιοι 10 byte και ο τομέας των τιμών μπορεί να είναι τυχαία μηνύματα (*strings).* Ο αλγόριθμός εκτελεί τη λειτουργία τους βασιζόμενος στα κλειδιά τους.

1) Περιγραφή Αλγορίθμου: Ο αλγόριθμός TeraSort εκτελείται σε πέντε βήματα:

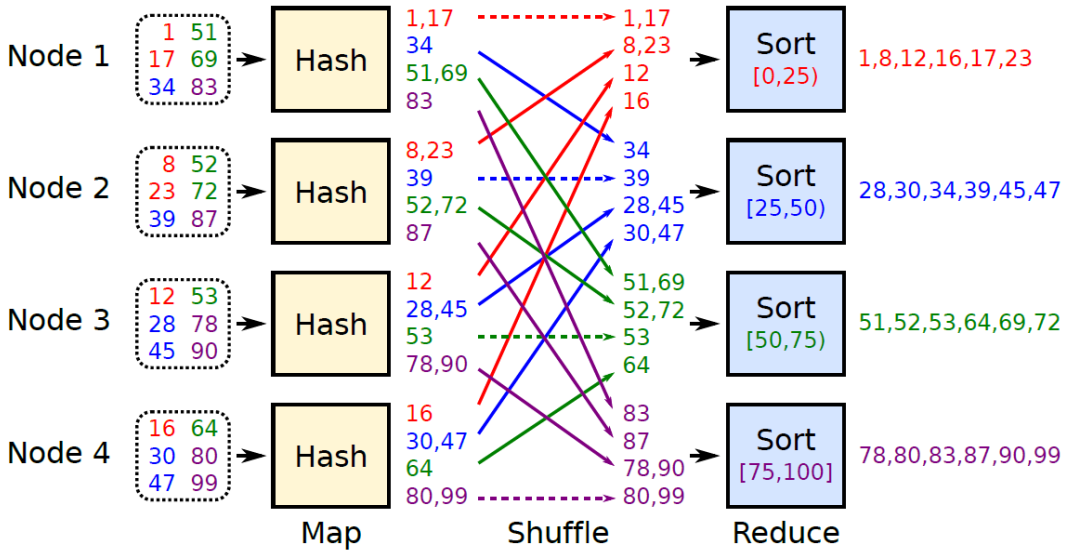
i. Τοποθέτηση/ανάθεση αρχείων,
ii. διαχωρισμός κλειδιών *(key domain partitioning),*
iii. φάση *map,*
iv. φάση αναμοιρασμού (*data shuffle)* και
v. φάσης μείωσης *(reduce).*

Στο πρώτο βήμα (i), χωρίζουμε σε K διαφορετικά μέρη (αρχεία) όλα τα ζευγάρια κλειδιών /τιμών. Κάθε ένας κόμβος K θα αναλάβει ένα από τα αρχεία K. Στην επόμενη φάση (ii), χωρίζουμε σε K τμήματα τα κλειδιά και ο κάθε κόμβος ορίζεται να αναλάβει να ταξινομήσει τα ζευγάρια κλειδιών / τιμών που του έχουν ανατεθεί σύμφωνα με το διαχωρισμό. Κατόπιν (iii) (*Map*), ο κάθε κόμβος ξεχωρίζει τις τιμές που του έχουν ανατεθεί ανάλογα με το διαχωρισμό

που έχει συμφωνηθεί στο ii. Στο τέταρτο βήμα (*Shuffle*), τα ζευγάρια κλειδιών / τιμών που ανήκουν στο ίδιο μέρος μεταφέρονται στον κόμβο που έχει οριστεί προκειμένου να ταξινομήσει το συγκεκριμένο τμήμα δεδομένων. Στη φάση *Reduce*, κάθε κόμβος ταξινομεί τοπικά τα ζεύγη KV που ανήκουν στην κατάτμηση που του έχει ανατεθεί.

Στο παρακάτω σχήμα 3 φαίνεται μία αναπαράσταση του αλγορίθμου *TeraSort* με τέσσερις κόμβους και κύριες κατατμήσεις [0, 25), [25, 50), [50, 75), [75, 100]. Τα τετράγωνα αριστερά αντιπροσωπεύουν τα αρχεία εισόδου. Το αρχείο εισόδου χωρίζεται σε τέσσερα μέρη, ένα για κάθε κόμβο *KV (*κλειδιών, τιμών*)* ένα για κάθε κατάτμηση (Φάση *Map*).

Για κάθε ένα από τα τέσσερα μέρη τα ζεύγη κλειδιών / τιμών που ανήκουν σε συγκεκριμένο τμήμα δεδομένων μεταφέρονται στον αντίστοιχο κόμβο (Φάση *Shuffle*) και πραγματοποιείται η ταξινόμηση (Φάση *Reduce*).



Εικόνα 4: Αλγόριθμος TeraSort

Προκειμένου να έχουν μία εκτίμηση του χρόνου που απαιτείται για μία τέτοια διαδικασία χρησιμοποίησαν ένα αρχείο της τάξης των 12Gbyte προκειμένου να ταξινομηθεί με τη μέθοδο *TeraSort* πάνω σε ένα Amazon EC2 δίκτυο 100MBps, αρχικά με 16 κόμβους και στη συνέχεια με 20,και πήραν τα ακόλουθα αποτελέσματα:

K	*Map (sec)*	*Pack (sec)*	*Shuffle (sec)*	*Unpack (sec)*	*Reduce (sec)*	*Συνολικός χρόνος (sec)*
16	1,86	2,35	945,72	0,85	10,47	961,25
20	1,47	2,00	960,07	0,62	8,29	972,45

Πίνακας 1: Αποτελέσματα TeraSort

Εύκολα παρατηρεί κάποιος ότι το μεγαλύτερο χρονικό διάστημα απαιτείται για τη διαδικασία *Shuffle* και μάλιστα με συντριπτική διαφορά από τις υπόλοιπες. Για παράδειγμα απαιτεί το 98,38% του συνολικού χρόνου ενώ ενδεικτικά η διαδικασία *Map* μόλις το 0,193% του συνολικού χρόνου. Ξεκινώντας από αυτό το πειραματικό αποτέλεσμα οι συγγραφείς του [1] ορθά θεώρησαν ότι υπάρχουν μεγάλα περιθώρια βελτίωσης στον αλγόριθμο *TeraSort* σύμφωνα με την λογική των *Coded Computing*. Έτσι ανέπτυξαν μία νέα εκδοχή του αλγόριθμου *TeraSort την οποία και ονόμασαν Coded TeraSort* [11] και η οποία παρουσιάζεται παρακάτω.

1.4.1 Coded TeraSort

Βασιζόμενο στη λογική που περιγράφεται παραπάνω στο Σχήμα 2. Coded Distributed Computing scheme, το *Coded TeraSort* [11] εκμεταλλεύεται τους περιττούς υπολογισμούς στα αρχεία εισόδου στη φάση *Map*, δημιουργώντας ευκαιρίες κωδικοποίησης για να μειώσει σημαντικά το φορτίο της φάσης ανακατεύθυνσης δεδομένων *(Shuffle).* Συγκεκριμένα, η εκτέλεση του *Coded TeraSort* αποτελείται από 6 στάδια λειτουργίας τα οποία και περιγράφονται συνοπτικά με τα έξι παρακάτω βήματα.

1) Δομημένη περιττή τοποθέτηση αρχείων. (*Structured Redundant File Placement).* Το σύνολο των ζευγών KV εισόδου χωρίζεται σε πολλά μικρά αρχεία, το καθένα από τα οποία τοποθετείται επανειλημμένα σε r *κόμβους με* $1 \leq r \leq K$ όπου K το σύνολο των κόμβων. Έχουμε δηλαδή υπολογιστικό φορτίο ίσο με r, όπως καθορίζεται από το σχήμα CDC.

2) *Map*. [13] Κάθε κόμβος χωρίζει τα δεδομένα που έχει στην κατοχή του κατά αντιστοιχία με τον αλγόριθμό TeraSort.

3) Κωδικοποίηση για τη δημιουργία κωδικοποιημένων πακέτων (*Encoding to Create Coded Packets).* Κάθε κόμβος παράγει κωδικοποιημένα πακέτα τα οποία δημιουργούνται κατά τη διάρκεια της διαδικασίας *Map*.

4) Ανακατεύθυνση «πολυεκπομπής» (*Multicast Shuffling).* Κάθε κόμβος εκπέμπει σε περισσότερους από έναν κόμβους, τα κωδικοποιημένα αρχεία που έχει δημιουργήσει, οι οποίοι είναι προκαθορισμένοι και ίσοι με *r*.

5) Αποκωδικοποίηση (*Decoding).* Τα ζεύγη κλειδιών / τιμών αποκωδικοποιούνται στον κάθε κόμβο με τη βοήθεια των κωδικοποιημένων πακέτων που λαμβάνει.

6) Μείωση (*Reduce*). Κάθε κόμβος ταξινομεί τοπικά τα ζεύγη KV για το τμήμα του αρχείου που του έχει ανατεθεί, με τον ίδιο τρόπο που πραγματοποιείται και στη φάση *Reduce* του απλού *TeraSort*.

Πραγματοποιώντας [1], [11] το ίδιο πείραμα με την περίπτωση του *TeraSort* που περιγράφεται παραπάνω κατέληξαν στα ακόλουθα αποτελέσματα. Οι μετρήσεις πραγματοποιήθηκαν δύο φορές μεταβάλλοντας το υπολογιστικό φορτίο *r* από 3 σε 5. Υπενθυμίζουμε ότι με *r* συμβολίζουμε τον αριθμό των κόμβων οι οποίοι καλούνται να επεξεργαστούν το ίδιο μέρος/κομμάτι του αρχικού αρχείου που ανατίθεται προς ταξινόμηση.

r=3

K	***Code Gen*** *(sec)*	***Map*** *(sec)*	***Pack*** *(sec)*	***Shuffle*** *(sec)*	***Unpack*** *(sec)*	***Reduce*** *(sec)*	***Συνολικός χρόνος*** *(sec)*
16	6,06	6,03	5,79	412,22	2,41	13,05	445,56
20	19,32	4,68	4,89	453,37	1,87	9,73	493,86

Πίνακας 2: Αποτελέσματα Coded TeraSort για r=3

r=5

K	***Code Gen*** *(sec)*	***Map*** *(sec)*	***Pack*** *(sec)*	***Shuffle*** *(sec)*	***Unpack*** *(sec)*	***Reduce*** *(sec)*	***Συνολικός χρόνος*** *(sec)*
16	23,47	10,84	8,10	222,83	3,69	14,40	283,33
20	140,91	8,59	7,51	269,42	3,70	10,97	441,10

Πίνακας 3: <u>Αποτελέσματα Coded TeraSort για r=5</u>

Από τα παραπάνω στοιχεία των πινάκων φαίνεται ξεκάθαρα ότι η χρήση του *Coded Terasort* οδηγεί σε επιτάχυνση της διαδικασίας κατά συντελεστή 2,16 - 1,97 και 3,39 - 2,20 για τις αντίστοιχες περιπτώσεις του *TeraSort*. Το κέρδος από τη διαδικασία του «αναμοιρασμού» *Shuffle* των μερών/κομματιών του αρχείου είναι τόσο μεγάλο που υπερκερνά τις χρονικές αυξήσεις σε όλα τα υπόλοιπα επιμέρους στάδια. Αυτό συμβαίνει παρόλο που απαιτείται σημαντικά επιπλέον χρόνος για αυτές ενώ έχουμε και μία επιπλέον *(Code Gen)* που αφορά στην κωδικοποίηση (πράξη *XOR)* των κομματιών του αρχικού αρχείου. Πέρα όμως από το κέρδος σε χρόνο θα πρέπει να συνεκτιμηθεί και η μείωση της συνολικής κίνησης στο δίκτυο που διασυνδέει όλους τους επιμέρους κόμβους αφού πλέον μεταφέρονται λιγότερα δεδομένα από αυτό. Συμπερασματικά θα μπορούσε κάποιος να πει ότι δεν υπάρχει καμία αμφιβολία για το κέρδος που προκύπτει από τον αλγόριθμο *Coded TeraSort.*

Υπάρχουν όμως ζητήματα τα οποία δεν ελήφθησαν υπόψη όπως η ασφάλεια των προς μετάδοση δεδομένων και η ανοχή στα λάθη ή τις απώλειες στη μετάδοση. Είναι προφανές ότι για το πρώτο, ο αλγόριθμος *Coded TeraSort* δεν προσφέρει καμία λύση ενώ για το δεύτερο αφήνει την επίλυση των ζητημάτων της μετάδοσης στα πρωτόκολλα δικτύου, τα οποία θα δώσουν φυσικά μία αξιόπιστη λύση, όμως θα προσθέσουν περαιτέρω χρονικές καθυστερήσεις και αυξήσεις στην κίνηση στο δίκτυο.

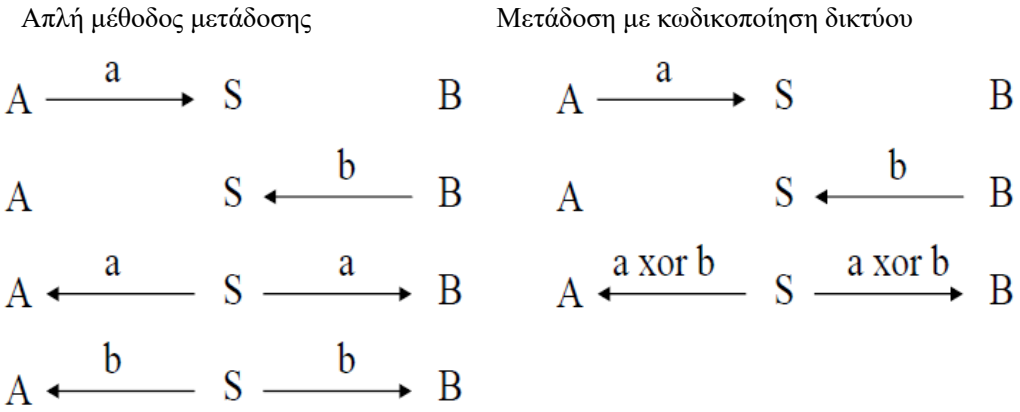
1.5 Συμπέρασμα

Από τα παραπάνω προκύπτει πλέον ξεκάθαρα ότι ο τομέας της Υπολογιστικής Κωδικοποίησης εμφανίζει μεγάλες προοπτικές για περεταίρω διερεύνηση και ως προς τις εφαρμογές του και ως προς τη κατεύθυνση της δημιουργίας νέων αλγορίθμων υλοποίησης. Είναι προφανές ότι έχουμε ένα πολλά υποσχόμενο πεδίο έρευνας στον συγκεκριμένο τομέα καθώς είναι φανερό ότι για μεγάλα μεγέθη δεδομένων προς επεξεργασία, ή μετάδοση, οι απαιτούμενοι χρόνοι, μειώνονται, ενίοτε σε μεγάλο βαθμό, και κατά συνέπεια επιταχύνονται τόσο η ταχύτητα μετάδοσης όσο και η επεξεργασία των δεδομένων.

Κεφάλαιο 2 - Network Coding (κωδικοποίηση δικτύου)

2.1 Εισαγωγή και Σχετικές Δημοσιεύσεις

Η κωδικοποίηση δικτύου είναι ένας τομέας έρευνας που πρωτοεμφανίστηκε σε μία σειρά από *papers* από τα τέλη της δεκαετίας του 90 μέχρι και τα μέσα της δεκαετίας του 2000 και τα οποία βασίστηκαν σε μία αρχική εργασία από το 1978 [14]. Η κωδικοποίηση δικτύου δίνει τη δυνατότητα στους κόμβους που αποτελούν ένα δίκτυο να δημιουργούν πακέτα προς μετάδοση τα οποία προέρχονται από τη σύνθεση δύο ή περισσότερων πακέτων τα οποία έχουν ήδη λάβει. Στη συνέχεια αυτά τα πακέτα προωθούνται σε νέους κόμβους οι οποίοι γνωρίζοντας τον αλγόριθμό συνδυασμού των πακέτων, μπορούν να δημιουργήσουν και πάλι τα αρχικά.. Η λογική υλοποίησης είναι απλή. Ο κάθε κόμβος αντί να προωθεί απλά τα πακέτα που λαμβάνει συνδυάζει διάφορα πακέτα εισόδου σε ένα ή περισσότερα πακέτα εξόδου. Στην παρακάτω Εικόνα 5 αναπαρίσταται σχηματικά μια απλή μέθοδος κωδικοποίησης σχετικά με μία μετάδοση χωρίς κωδικοποίηση [3].



Εικόνα 5: Απλό παράδειγμα επικοινωνίας A,B στους οποίους μεσολαβεί κόμβος S

Έχουμε τρεις κόμβους οι οποίοι ανταλλάσσουν δεδομένα. Ο κόμβος A στέλνει επιθυμεί να στείλει ένα πακέτο a στον κόμβο B ενώ μεσολαβεί ανάμεσά τους ο κόμβος S. Αντιστοίχως και μέσω του κόμβου S, ένα πακέτο b θα αποσταλεί από τον κόμβο B στον κόμβο A. Η διαδικασία απαιτεί για να ολοκληρωθεί 4 βήματα ενώ από και προς τον κόμβο S κυκλοφορούν πακέτα 6 φορές. Χρησιμοποιώντας μία απλή κωδικοποίηση τύπου XOR ο κόμβος S αντί να αποστείλει τα πακέτα a και b ξεχωριστά, τα κωδικοποιεί μαζί χρησιμοποιώντας την πράξη XOR. Έτσι A και

Β αντί να λάβουν το πακέτα a και b ξεχωριστά λαμβάνουν ένα πακέτο: το a XOR b. Από αυτό το συνδυασμό και οι δύο κόμβοι έχουν τη δυνατότητα να αντλήσουν το πακέτο που τους ενδιαφέρει, από τη στιγμή που έχουν ήδη το ένα από τα δύο. Με αυτό τον τρόπο η επικοινωνία ολοκληρώνεται σε τρία βήματα ενώ το δίκτυο χρησιμοποιείται 4 φορές μειώνοντας έτσι την κίνηση στο δίκτυο. Η βασική αρχή λειτουργίας είναι της κωδικοποίησης δικτύου είναι η ίδια, με τη διαφορά ότι αντί της XOR δημιουργούνται γραμμικοί συνδυασμοί των δεδομένων τα οποία τελικά αντιστοιχίζονται με αριθμούς πάνω σε ένα πεπερασμένο σύνολο. Με δεδομένο ότι τα τελευταία χρόνια η υπολογιστική ισχύς έχει γίνει ακόμα φθηνότερη, ακόμα και σε σύγκριση με τα μέσα της δεκαετίας του 2000 οπότε και άρχισε να γίνεται διαδεδομένη η ιδέα του *Network Coding,* είναι προφανές ότι το συγκεκριμένο πεδίο έρευνας έχει ακόμα πολλά να προσφέρει στην εξέλιξη του τομέα της μετάδοσης δεδομένων.

Είναι προφανές ότι τέτοιου είδους συνδυασμοί πέρα από την εξοικονόμηση του διαθέσιμου *BW* ανάμεσα στους κόμβους του δικτύου, μας δίνουν τη δυνατότητα ανασύνθεσης των αρχικών πακέτων ακόμα και αν χαθούν κάποια από αυτά. Επιπλέον η κωδικοποίηση δικτύου μπορεί να χρησιμοποιηθεί και για εφαρμογές κρυπτογράφησης [15]

2.2 Τι είναι η γραμμική κωδικοποίηση δικτύου;

Σε αυτό το σημείο θα προσπαθήσουμε να δώσουμε τις αρχές λειτουργίας του *linear network coding.* Ας θεωρήσουμε ένα κόμβο δικτύου οποίος μπορεί και προωθεί δεδομένα. Χωρίς κωδικοποίηση είναι προφανές ότι όταν ο κόμβος λάβει ένα πακέτο απλά το προωθεί. Με το *network coding* ένας κόμβος Α έχει τη δυνατότητα να συνδυάζει δύο ή περισσότερα πακέτα που λαμβάνει για ένα συγκεκριμένο κόμβο Β, δημιουργώντας ένα νέο πακέτο μήκους *L bits*, ίσο με το μεγαλύτερο από τα αρχικά και να προωθήσει αυτό στον Β. Προκειμένου να μπορεί να υλοποιηθεί ο γραμμικός συνδυασμός των πακέτων, στα πακέτα που έχουν μικρότερο μήκος από *L* προστίθενται μηδενικά στις υπολειπόμενες κενές θέσεις μέχρι του μήκους *L*. Πρέπει να έχει γίνει ήδη κατανοητό ότι η γραμμική κωδικοποίηση δικτύου δεν είναι συνένωση των πακέτων. Η κωδικοποίηση πακέτων μήκους *L*, μας δίδει κωδικοποιημένο πακέτο το οποίο έχει επίσης μέγεθος *L*. Ένα πακέτο που προέρχεται από τη σύνθεση 2 ή περισσότερων πακέτων, όπως είναι φυσικό, περιέχει μόνο μέρος των αρχικών πληροφοριών. Το κωδικοποιημένο πακέτο που

λαμβάνεται δίνει τη δυνατότητα στον παραλήπτη του να επανασυνθέσει στο ακέραιο τα πακέτα που του χρειάζονται.

2.2.1 Κωδικοποίηση

Ας υποθέσουμε ότι ένας αριθμός πρωτότυπων πακέτων M^1, ..., M^n παράγεται από μία ή περισσότερες πηγές. Στην γραμμική κωδικοποίηση δικτύου, κάθε πακέτο μέσω του δικτύου συνδέεται με μια ακολουθία συντελεστών g_1, ..., g_n και είναι ίση με

$$X = \sum_{i=1}^{n} g_i M^i$$

Η άθροιση πρέπει να συμβαίνει για κάθε θέση συμβόλου. Για παράδειγμα $X_\nu = \sum_{i=1}^{n} g_i M_\nu^i$ όπου M_ν^i και X_ν είναι τα νιοστά σύμβολα του M^i και του X αντίστοιχα. Για λόγους απλοποίησης θεωρούμε ότι ένα πακέτο περιέχει τόσο τους συντελεστές $g = g_1, \dots, g_n$ το οποίο και ονομάζουμε διάνυσμα/φορέα κωδικοποίησης όσο και τα κωδικοποιημένα δεδομένα $X = \sum_{i=1}^{n} g_i M^i$ τα οποία και ονομάζουμε διάνυσμα/φορέα πληροφοριών. Για να καταστεί δυνατή η αποκωδικοποίηση των δεδομένων από τους κόμβους/παραλήπτες χρησιμοποιούν το διάνυσμα/φορέας κωδικοποίησης. Για παράδειγμα αν έχουμε ένα διάνυσμα αποκωδικοποίησης $e_\nu = (0, \dots, 0, 1, 0, \dots, 0)$ όπου το 1 βρίσκεται στη νιοστή θέση αυτό σημαίνει ότι το διάνυσμα πληροφορίας είναι ίσο με το M^ν. Η κωδικοποίηση μπορεί να γίνει αναδρομικά δηλαδή σε ήδη κωδικοποιημένα πακέτα και αυτό μπορεί να επαναληφθεί σε διάφορους κόμβους μέσα στο δίκτυο. Θεωρούμε ένα κόμβο ο οποίος έχει λάβει ένα σετ (g^1, X^1),…, (g^m, X^m) κωδικοποιημένων πακέτων όπου g^ν είναι το διάνυσμα κωδικοποίησης (πληροφορίας) του νιοστού πακέτου. Αυτός ο κόμβος μπορεί να δημιουργήσει ένα νέο κωδικοποιημένο πακέτο (g', X') επιλέγοντας ένα σετ συντελεστών $h_1, h_2, \dots, h_m$ και να υπολογίσει το νέο γραμμικό συνδυασμό:

$$X' = \sum_{\nu=1}^{m} g_\nu M^\nu$$

Αυτή η διαδικασία μπορεί να επαναληφθεί σε διάφορους κόμβους στο ίδιο δίκτυο.

2.2.2 Αποκωδικοποίηση

Έστω ότι το σύνολο $(g^1, X^1),\ldots, (g^m, X^m)$ έχει παραληφθεί από ένα κόμβο. Για να ανακτήσει τα αρχικά πακέτα, πρέπει να λύσει το αρχικό σύστημα στο οποίο βασίστηκε η κωδικοποίηση,

$$X^j = \sum_{i=1}^{n} g_i^j \, M^i$$

Όπου άγνωστοι είναι τα M^i. Για να λυθεί αυτό πρέπει να υπολογιστούν n άγνωστοι από το παραπάνω γραμμικό σύστημα m εξισώσεων. Από τα μαθηματικά γνωρίζουμε ότι προκειμένου υπάρχει δυνατότητα επίλυσης θα πρέπει $m \geq n$. Συμπερασματικά το σύνολο των πακέτων που λαμβάνει ο κόμβος θα πρέπει να είναι τουλάχιστον ισάριθμος με τον αριθμό των αρχικών πακέτων. Από την άλλη η σχέση $m \geq n$ πιθανόν να μην είναι αρκετή, μιας και κάποιοι από τους τους συνδυασμούς μπορεί να έχουν γραμμική εξάρτηση μεταξύ τους. Αυτό όμως δεν είναι δύσκολο και όπως θα δούμε παρακάτω αποτελεί πλεονέκτημα της κωδικοποίησης δικτύου.

2.2.3 Επιλογή Γραμμικών Εξισώσεων

Το πρόβλημα του σχεδιασμού της κωδικοποίησης δικτύου είναι η επιλογή των υπολογισμών που πρέπει να κάνει κάθε κόμβος του δικτύου. Ένας απλός αλγόριθμος είναι ο κάθε κόμβος του δικτύου να επιλέγει τυχαία τους συντελεστές μέσα από ένα συγκεκριμένο πεπερασμένο πεδίο με ένα εντελώς ανεξάρτητο και αποκεντρωμένο τρόπο. [16] Υπάρχει συγκεκριμένη πιθανότητα να επιλέξουμε γραμμικά εξαρτημένους συνδυασμούς με την τυχαία κωδικοποίηση δικτύου [16]. Υπάρχει άμεση εξάρτηση αυτής της πιθανότητας με το μέγεθος του πεδίου 2^s. Αποτελέσματα δοκιμών έδειξαν ότι ακόμα και για μικρά σχετικά πεδία (π.χ. s=8) το ενδεχόμενο αυτό είναι εξαιρετικά απίθανο. [17]. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε ντετερμινιστικούς αλγόριθμούς. Ο πολυωνυμικός αλγόριθμός για πολλαπλή εκπομπή που πρότειναν οι συγγραφείς στο [18] εξετάζει με τη σειρά κάθε κόμβο του δικτύου και κατόπιν αποφασίζει τους γραμμικούς συνδυασμούς που θα εκτελέσει κάθε κόμβος στα δεδομένα που είναι προς εκπομπή. Από τη στιγμή που κάθε κόμβος χρησιμοποιεί καθορισμένους γραμμικούς συντελεστές τα πακέτα προς μετάδοση περιέχουν μόνο το διάνυσμα πληροφορίας. Υπάρχουν ακόμα αποκεντρωμένοι ντετερμινιστικοί αλγόριθμοι οι οποίοι εφαρμόζονται σε συγκεκριμένες κατηγορίες δικτύων.

πληροφορίας. Υπάρχουν ακόμα αποκεντρωμένοι ντετερμινιστικοί αλγόριθμοι οι οποίοι εφαρμόζονται σε συγκεκριμένες κατηγορίες δικτύων.

2.2.4 Πρακτικά Ζητήματα

Αποκωδικοποίηση: Για να επιτευχθεί η αποκωδικοποίηση απαιτείται η επίλυση ενός συνόλου γραμμικών εξισώσεων. Η επίλυση υλοποιείται ως εξής. Σε ένα πίνακα αποκωδικοποίησης ο κόμβος αποθηκεύει αυτά που λαμβάνει μαζί με τα δικά του αρχικά πακέτα. Αρχικά, ο πίνακας περιέχει μόνο τα μη κωδικοποιημένα πακέτα που δημιουργούνται από τον συγκεκριμένο κόμβο με τα αντίστοιχα διανύσματα κωδικοποίησης (αν υπάρχουν, διαφορετικά είναι κενός). Το πιο πρόσφατο πακέτο ληφθέν πακέτο, εισάγεται στην τελευταία σειρά του πίνακα. Ο πίνακας αποκωδικοποίησης χρησιμοποιώντας τη μέθοδο Gauss μετασχηματίζεται σε τριγωνικό. Ένα κωδικοποιημένο πακέτο ονομάζεται «καινοτόμο», αν και μόνο αν οδηγεί την τάξη του πίνακα σε αύξηση. Εάν δεν είναι, τότε ο αλγόριθμός το μειώνει σε μια σειρά από 0 (μηδενικά) με τη μέθοδο Gauss και στη συνέχεια το πακέτο αγνοείται. Μόλις ο πίνακας περιέχει μια σειρά της μορφής e_i, τότε ο κόμβος ξέρει ότι το X που μόλις δημιούργησε είναι ίσο με το αρχικό πακέτο M_i. Προκειμένου να ικανοποιηθεί το παραπάνω γεγονός θα πρέπει να ληφθούν n ανεξάρτητα γραμμικά διανύσματα κωδικοποίησης. Η αποκωδικοποίηση εκτελείται μόνος στους κόμβους που λαμβάνουν πακέτα και όχι σε όλους.

Δημιουργία κωδικοποιημένων πακέτων: Προκειμένου να λειτουργήσει αποδοτικά η μέθοδος συνολικά, η αποκωδικοποίηση πρέπει να επιτυγχάνεται σχετικά εύκολα. Με το εύκολα εννοούμε ότι οι υπολογισμοί που απαιτούνται πρέπει να εκτελούνται (σχετικά) σύντομα. Για αυτό το λόγο το μέγεθος των πινάκων αποκωδικοποίησης πρέπει να είναι συγκρατημένο. Αυτό είναι δύσκολο να πραγματοποιηθεί στην περίπτωση που χρησιμοποιούνται αλγόριθμοι τυχαίας κωδικοποίησης δικτύου και ευκολότερο για τους ντετερμινιστικούς στην περίπτωση απλής κωδικοποίησης δικτύου. Για τους πρώτους, τα πακέτα συνήθως ομαδοποιούνται σε ομάδες εκπομπών, 1^{η} ομάδα εκπομπής 2^{η} ομάδα εκπομπής κ.ο.κ. (γενιές πακέτων) και μπορούν να συνδυαστούν μόνο πακέτα από την ίδια ομάδα (γενιά). [19]. Σύμφωνα με τους συγγραφείς στο [20] η «ποιότητα» των ομάδων (μέγεθος και σύνθεση) επηρεάζει το μέγεθος της αποδοτικότητας της κωδικοποίησης δικτύου. Παρόμοια ζητήματα πρέπει να ληφθούν υπόψη και για το μέγεθος του πεπερασμένου πεδίου από το οποίο αντλούνται οι συντελεστές. Και οι δύο αυτοί παράμετροι επιτρέπουν την

μεταβολή της απόδοσης της κωδικοποίησης δικτύου στην περίπτωση που έχουμε κόμβους με μειωμένη μνήμη και μειωμένη υπολογιστική ισχύ.

Καθυστέρηση: Η αποκωδικοποίηση των πακέτων επηρεάζει σε μικρό βαθμό την παρατηρούμενη καθυστέρηση. Προκειμένου να είναι δυνατή η αποκωδικοποίηση κάποιων πακέτων (να προκύπτει δηλαδή μία σειρά (e_i, M_i) στον πίνακα αποκωδικοποίησης), ενίοτε δεν είναι απαραίτητο, να ληφθεί το σύνολό τους. Μαζί με τη μείωση του αριθμού των απαιτούμενων μεταδόσεων, η συνολική καθυστέρηση με χρήση κωδικοποίησης δικτύου δεν είναι συνήθως μεγαλύτερη από την κανονική καθυστέρηση σε φυσιολογικές συνθήκες.

Λειτουργίες πεπερασμένων πεδίων: Η κωδικοποίηση δικτύου απαιτεί κάποιες πράξεις στο πεδίο $\mathbb{F}_{2^s}$. Η πρόσθεση π.χ. είναι το τυπικό *XOR*. Για τον πολλαπλασιασμό, ερμηνεύεται μία ακολουθία $b_0, b_1, \dots, b_{s-1}$ από s bit σαν το πολυώνυμο $b_0 + b_1X + b_2X^2 + \cdots + b_{s-1}X^{s-1}$. Έπειτα, ο κόμβος επιλέγει ένα πολυώνυμο s βαθμού που είναι μη αναστρέψιμο στο $\mathbb{F}_2$ (δεν υπάρχει μόνο ένα και καθένα από αυτά δίνει διαφορετική αναπαράσταση των $\mathbb{F}_{2^s}$. για παράδειγμα, η αναπαράσταση του $\mathbb{F}_{2^8}$ του Rijndael χρησιμοποιεί το πολυώνυμο $1 + X + X^3 + X^4 + X^8$. Η διαίρεση υπολογίζεται από τον ευκλείδειο αλγόριθμο.

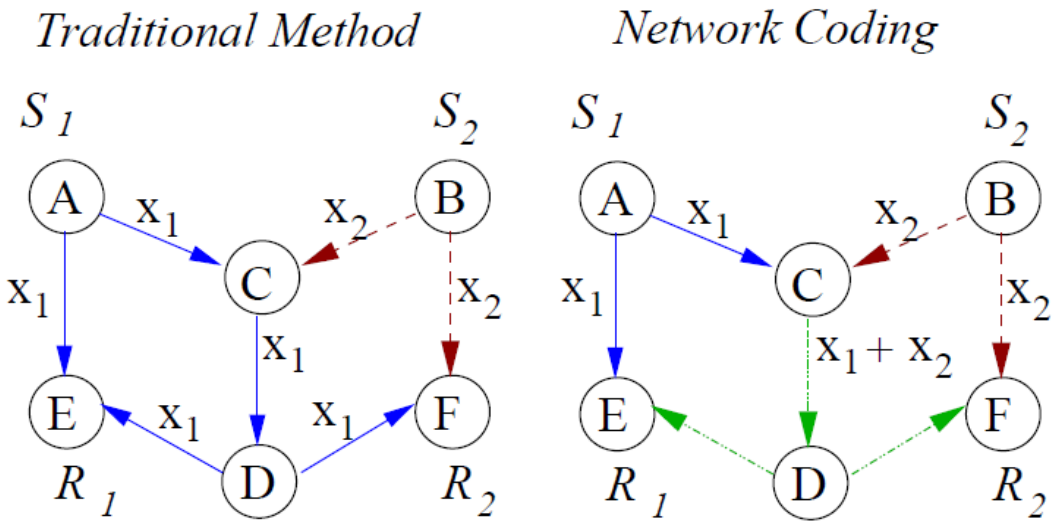
Σε ένα πεπερασμένο πεδίο υπάρχει τουλάχιστον ένα ειδικό στοιχείο α, που ονομάζεται γεννήτρια (για παράδειγμα, στην παράσταση Rijndael του $\mathbb{F}_{2^8}$, για το α ισχύει $\alpha = 0x03 = 1 + X$. Δεδομένου ότι $\ln(xy) = \ln(x) + \ln(y)$, ο πολλαπλασιασμός και η διαίρεση μπορούν να εφαρμοστούν ανατρέχοντας σε δύο πίνακες που αντιστοιχούν το x στο $\ln(x)$ και αντιστρόφως. Αυτό δεν απαιτεί σημαντικό όγκο δεδομένων μιας και για π.χ. $s = 8$ απαιτούνται δύο πίνακες μεγέθους 255 bytes.

2.3 Τα Πλεονεκτήματα της Κωδικοποίησης Δικτύου

Η κωδικοποίηση δικτύου παρουσιάζει βελτιωμένες αποδόσεις, σε θεωρητικό τουλάχιστον επίπεδο, σε περιπτώσεις όπου τα κύρια χαρακτηριστικά των δικτύων μένουν σταθερά. Τα θεωρητικά αποδεδειγμένα αποτελέσματα σχετικά με την κωδικοποίηση δικτύου αφορούν κυρίως τις βελτιώσεις απόδοσης σε στατικές ρυθμίσεις. Θα εξετάσουμε πρώτα αυτές τις περιπτώσεις και στη συνέχεια θα ασχοληθούμε και με τις περιπτώσεις που υλοποιούνται με τυχαία κωδικοποίηση δικτύου.

2.3.1 Βελτίωση της Χωρητικότητας των Δικτύων

Η κωδικοποίηση δικτύου μπορεί να βελτιώσει την χωρητικότητα ενός δικτύου το οποίο μεταδίδει δεδομένα που απευθύνονται ταυτόχρονα σε πολλούς κόμβους. Το θεωρητικό όφελος φτάνει μέχρι το σημείο όπου ο κάθε κόμβος απολαμβάνει το ρυθμό μετάδοσης που θα μπορούσε να έχει ακόμα και αν χρησιμοποιούσε μόνος του το δίκτυο. Για να συμβεί αυτό η κωδικοποίηση δικτύου συμβάλει τα μέγιστα προς αυτή τη κατεύθυνση. Μειώνοντας στην ουσία τον όγκο των δεδομένων που διασχίζουν το δίκτυο επιτυγχάνει πολύ υψηλούς ρυθμούς μετάδοσης για όλα τα μέλη του δικτύου.



Εικόνα 6: Σύγκριση απλής μεθόδου μετάδοσης με μετάδοση κωδικοποίησης δικτύου

Οι κόμβοι S_1 και S_2 κάνουν πολλαπλή εκπομπή στους κόμβους R_1 και R_2. Θεωρούμε ότι όλοι οι σύνδεσμοι που ενώνουν τους κόμβους έχουν χωρητικότητα 1. Με μία απλή μορφή κωδικοποίησης π.χ. XOR δικτύου (εκπέμποντας ο κόμβος C ένα πακέτο το οποίο προκύπτει κάνοντας XOR το πακέτο X_1 του S_1 με το πακέτο X_2 του S_2) μέσω του συνδέσμου CD, στον κόμβο D ο R_1 θα λάβει το X_1 XOR X_2 από το οποίο μπορεί να εξάγει το πακέτο X2 χρησιμοποιώντας το X_1 που έχει λάβει ήδη μέσω του συνδέσμου AE. Οι τιμές που επιτυγχάνονται για το ρυθμό μετάδοσης σε αυτή την περίπτωση είναι 2 για κάθε πηγή, η ίδια με την περίπτωση όπου κάθε δέκτης χρησιμοποιεί το δίκτυο για αποκλειστική χρήση. Από το παραπάνω σχήμα φαίνεται ότι έχουμε μέγιστο ρυθμό μετάδοσης 1,5 χωρίς τη βοήθεια της κωδικοποίησης.

Ένα ενδιαφέρον σημείο είναι ότι η κωδικοποίηση δικτύου επιτρέπει την επίτευξη της μέγιστης ταχύτητας μεταφοράς δεδομένων στις περιπτώσεις πολλαπλής εκπομπής που χρησιμοποιούν αλγόριθμους πολυωνυμικού χρόνου. Αντίθετα, η επίτευξη της βέλτιστης ταχύτητας μεταφοράς δεδομένων με δρομολόγηση είναι NP-πλήρης. Έτσι, [3] ακόμα και όταν τα οφέλη της αναμενόμενης απόδοσης της κωδικοποίησης δικτύου δεν είναι μεγάλα, αναμένουμε ότι θα μπορέσουμε να τα επιτύχουμε χρησιμοποιώντας "απλούστερους" αλγόριθμους.

2.3.2 Ευστάθεια και Προσαρμοστικότητα

Τα πιο σημαντικά οφέλη της κωδικοποίησης δικτύου τελικά αποδείχθηκε ότι ήταν από την μεριά της σταθερότητας και της προσαρμοστικότητας. Κάποιος μπορεί να σκεφτεί ότι στην κωδικοποίηση δικτύου, όπως συμβαίνει με την παραδοσιακή κωδικοποίηση, έχουμε σαν είσοδο πακέτα πληροφοριών και σαν έξοδο κωδικοποιημένα πακέτα, όπου κάθε κωδικοποιημένο πακέτο είναι απαραίτητο για την επανασύσταση της αρχικής πληροφορίας. Η διαφορά είναι όμως ότι αρκεί να λάβουμε επαρκή αριθμό κωδικοποιημένων πακέτων, ανεξάρτητα από το ποια ούτως ώστε να είμαστε σε θέση να αποκωδικοποιήσουμε. Με άλλα λόγια είναι δυνατή η μετάδοση δεδομένων ακόμα και αν έχουμε απώλεια κάποιων πακέτων στο δίκτυο. Από τη στιγμή που οι απαιτούμενες πράξεις και υπολογισμοί που απαιτούνται για την υλοποίηση της κωδικοποίησης δικτύου δεν γίνονται μόνο στον αρχικό κόμβο εκπομπής αλλά και μέσω του δικτύου δίνει ένα νέο χαρακτηριστικό στην συγκεκριμένη τεχνική. Την καθιστά κατάλληλη σε εφαρμογές όπου τα μέλη ενός δικτύου δεν έχουν αρκετή γνώση για την κατάστασή του.

2.4 Πεδίο Εφαρμογών της Κωδικοποίησης Δικτύου

2.4.1 Διανομή Αρχείων σε Συνδέσεις P2P (peer to peer)

Σε αυτή την περίπτωση έχουμε ένα *server* ο οποίος προκειμένου να μεταδώσει ένα μεγάλο αρχείο το τεμαχίζει σε ένα αριθμό πακέτων τα οποία ονομάζονται *blocks* και τα μεταδίδει σε ένα δίκτυο που αποτελείται από ομότιμους κόμβους. Οι κόμβοι λαμβάνουν το αρχείο κατεβάζοντας τα *blocks* από τον κόμβο εξυπηρετητή ενώ παράλληλα προωθούν τα πακέτα που έχουν λάβει στους υπόλοιπους ομότιμους κόμβους του δικτύου που δεν τα έχουν κατεβάσει ακόμα. Ο κάθε κόμβος διατηρεί συνδέσεις με ένα περιορισμένο αριθμό κόμβων οι οποίοι έχουν επιλεχθεί τυχαία. Σε μία υλοποίηση κωδικοποίησης δικτύου η οποία ονομάζεται *Avalanche* [21] τα πακέτα που στέλνει ο εξυπηρετητής στο δίκτυο είναι τυχαίοι γραμμικοί συνδυασμοί όλων των αρχικών πακέτων. Με την ίδια λογική και η κόμβοι πελάτες του υπόλοιπου δικτύου αποστέλλουν στο δίκτυο πακέτα τα οποία είναι τυχαίοι γραμμικοί συνδυασμοί όλων των πακέτων που έχουν τη δεδομένη στιγμή στην κατοχή τους. Ένας κόμβος μπορεί είτε να καθορίσει πόσα καινούρια πακέτα μπορεί να μεταδώσει σε έναν γειτονικό κόμβο, συγκρίνοντας το δικό του πίνακα συντελεστών αποκωδικοποίησης με τον αντίστοιχο πίνακα του γειτονικού κόμβου είτε μπορεί απλά να μεταδώσει κωδικοποιημένα πακέτα έως ότου ο γείτονας λάβει το πρώτο πακέτο που έχει ήδη στην κατοχή του. Η μετάδοση προς τον γειτονικό κόμβο σταματά τότε μέχρι ο κόμβος εκπομπής να παραλάβει νεότερα πακέτα. Οι συντελεστές κωδικοποίησης μεταδίδονται μαζί με τα πακέτα. Αυτό δεν δημιουργεί ιδιαίτερο πρόβλημα, επειδή ο όγκος των δεδομένων στο κάθε πακέτο έχει συνήθως μέγεθος εκατοντάδων Kbyte, ενώ το μέγεθος του πίνακα που περιέχει τους συντελεστές αποκωδικοποίησης είναι σε σχέση με αυτό, αμελητέο. Η κωδικοποίηση δικτύου βοηθά από πολλές απόψεις: α) ελαχιστοποιεί τους χρόνους λήψης. Αν υποθέσουμε ότι θέλουμε να βελτιώσουμε την απόδοση σε ένα μεγάλο κατανεμημένο σύστημα ομότιμων Η/Υ θα πρέπει να γίνει ένας προγραμματισμός μετάδοσης. Κάτι τέτοιο όμως είναι ιδιαίτερα πολύπλοκο και χρονοβόρο. Στην περίπτωση μάλιστα που οι κόμβοι δεν έχουν αρκετές πληροφορίες για την τοπολογία του δικτύου που συμμετέχουν η πολυπλοκότητα και κατά συνέπεια ο χρόνος μεγαλώνουν ακόμα περισσότερο. Χρησιμοποιώντας κωδικοποίηση δικτύου, οι κόμβοι που αποτελούν το δίκτυο αδιαφορούν για την τοπολογία του ή την κατάστασή του.

Κατά συνέπεια, μπορούν να χρησιμοποιηθούν πολύ απλοί μηχανισμοί υλοποίησης δικτύου. β) Λόγω της ποικιλομορφίας των κωδικοποιημένων πακέτων, μια λύση που βασίζεται στη κωδικοποίηση δικτύου είναι πολύ πιο ισχυρή σε περίπτωση που ο διακομιστής εγκαταλείψει νωρίς (προτού όλοι οι ομότιμοι κόμβοι τερματίσουν τη λήψη τους) ή στην περίπτωση όπου οι κόμβοι συμμετέχουν μόνο για ένα σύντομο χρονικό διάστημα ή σταματούν να συμμετέχουν τη διαδικασία αμέσως μετά την ολοκλήρωση της λήψης). γ) Σε αντίθεση με τα πρωτόκολλα που βασίζονται στη διαβίβαση, το πρωτόκολλο κωδικοποίησης δικτύου τους υστερεί μόνο κατ' ελάχιστο όταν υλοποιούνται μηχανισμοί κινήτρου για συνεργασία.

2.4.2 Ασύρματα Δίκτυα

Αμφίδρομη επικοινωνία σε ασύρματο δίκτυο: Όπως δείξαμε και στην ενότητα 2.1 Εισαγωγή και Σχετικές Δημοσιεύσεις(σελ.18), στην κλασσική περίπτωση ενός τοπικού ασύρματου δικτύου όπου δύο κόμβοι επικοινωνούν μέσω μία κοινής βάσης, η κωδικοποίηση δικτύου μπορεί να βελτιώσει την απόδοση του. Αυτή η ρύθμιση μπορεί να επεκταθεί στην περίπτωση δρομολόγησης μέσα από πολλαπλούς κόμβους, σε ένα ασύρματο δίκτυο (ή σε οποιοδήποτε άλλο δίκτυο με τη μετάδοση να λαμβάνει χώρα στο φυσικό επίπεδο) όπου η κυκλοφορία μεταξύ των δύο τελικών κόμβων είναι αμφίδρομη [30] και οι δύο κόμβοι έχουν παρόμοιο αριθμό πακέτων για ανταλλαγή. Κάποια στιγμή και με την πρόοδο της μετάδοσης όλοι οι κόμβοι έχουν στην κατοχή τους πακέτα που πρέπει να μεταδοθούν στο δίκτυο εκατέρωθεν. Όποτε υπάρξει δυνατότητα μετάδοσης, ο κόμβος θα χρησιμοποιήσει δύο από τα πακέτα που έχει στην κατοχή του και θα δημιουργήσει ένα τρίτο που θα προκύπτει από τον συνδυασμό των αρχικών δύο μέσω της λογικής πράξης XOR και το προωθεί στο δίκτυο με τελικό προορισμό τους γειτονικούς προς αυτόν κόμβους. Και οι δύο παραλήπτες γνωρίζουν ήδη ένα από τα πακέτα. Άρα ανά μετάδοση δύο διαφορετικοί δρομολογητές λαμβάνουν ένα νέο πακέτο ενώ κανονικά θα απαιτούνταν δύο μεταδόσεις. Αυτό στην ουσία σημαίνει διπλασιασμό της χωρητικότητας του δικτύου. Οικιακά ασύρματα δίκτυα: Η βελτίωση της απόδοσης ενός οικιακού ασύρματου δικτύου είναι δυνατόν να βελτιωθεί σε μεγάλο βαθμό ακόμα και με μία βασική μορφή κωδικοποίησης [22]. Όλες οι εκπομπές μεταδίδονται και λαμβάνονται από τους γειτονικούς κόμβους. Τα πακέτα περιέχουν συνοπτικές πληροφορίες για όλα τα άλλα πακέτα που έχουν ήδη ληφθεί. Έτσι όλοι οι κόμβοι έχουν γνώση σχετικά με την κατανομή των πακέτων στους γειτονικούς τους κόμβους. Σε

πειράματα με δίκτυα που χρησιμοποιούν το πρωτόκολλο 802.11, οι συγγραφείς απέδειξαν ότι ο μηχανισμός τους σχεδόν διπλασιάζει τη ταχύτητα του δικτύου.

Πολλαπλές εκπομπές: Η μετάδοση σε όλο το δίκτυο χρησιμοποιείται για ορισμένους σκοπούς σε δίκτυα *ad-hoc* (π.χ. εντοπισμός διαδρομών) και μπορεί να υλοποιηθεί πολύ πιο αποτελεσματικά με την κωδικοποίηση δικτύου, [20], [23]. Υπάρχει μάλιστα η δυνατότητα νε μειωθεί ο αριθμός των εκπεμπόμενων πακέτων ακόμα και στο μισό ή περισσότερο. Έτσι επιτυγχάνεται πέρα από την προφανή μείωση της χρήσης του δικτύου αλλά και μείωση της καταναλισκόμενης ενέργειας. Αυτό ίσως φαντάζει απειροελάχιστο αλλά σε μεγάλης κλίμακας συστήματα ή αθροίζοντας τα οφέλη από πολλά δίκτυα το ενεργειακό όφελος είναι σημαντικό. Η ενεργειακή δαπάνη κατανέμεται ομοιόμορφα στους κόμβους ή καλύπτεται μόνο από μερικούς κόμβους (το οποίο μπορεί να προσφέρει χαμηλότερη κατανάλωση ενέργειας από τις φορητές συσκευές). Χρησιμοποιώντας κωδικοποίηση δικτύου, μπορούμε να επιτύχουμε μεγάλη βελτίωση στην απόδοση του δικτύου ακόμα και σε σύγκριση με αυτές που είδη αναφέραμε. Αυτό δείχνει ότι όταν είναι απαραίτητο το δίκτυο να λειτουργεί με σταθερότητα, η κωδικοποίηση δικτύου προβάλει σαν η περισσότερο ταιριαστή επιλογή.

2.4.3 Ασφάλεια Δικτύων

Στο [24], οι συγγραφείς διερευνούν το πρόβλημα του σχεδιασμού ασφαλών κωδίκων δικτύου για δίκτυα καλωδίων, όπου ορισμένοι σύνδεσμοι μπορούν να προσεγγιστούν από εισβολείς. Θεωρούν ότι είναι γνωστό ποιοι σύνδεσμοι έχουν αξιοποιηθεί. Ο κόμβος που κάνει την αρχική εκπομπή μπορεί να χρησιμοποιήσει ένα τυχαίο τρόπο κωδικοποίησης, ο οποίος είναι φυσικά γνωστός στους δέκτες, και να προκύψουν έτσι κρυπτογραφημένα δεδομένα τα οποία και αποστέλλονται στους δέκτες. Επιπλέον, η αμοιβαία πληροφόρηση μεταξύ των πακέτων που λαμβάνονται από τους υποκλοπείς και τα αρχικά πακέτα είναι μηδέν. Θα μπορούσαμε τώρα να πούμε ότι έχουμε εισάγει μία ασφάλεια στα δεδομένα από τη στιγμή που οι δέκτες μπορούν να ανακτήσουν τα αρχικά (χωρίς κωδικοποίηση) πακέτα, μόνο εφόσον πακέτα αποκωδικοποίησης τα οποία πρέπει να έχουν συγκεκριμένο αριθμό και να είναι γραμμικά ανεξάρτητα μεταξύ τους. Τέτοιοι κώδικες είναι πιο αποτελεσματικοί ως προς τη βελτίωση της χρήσης του δικτύου, αλλά σε επίπεδο ασφάλειας υστερούν. Για παράδειγμα ένας κακόβουλος κόμβος που μπορεί να αποκτήσει $n-1$ γραμμικούς συνδυασμούς από ένα σύνολο n είναι πολύ κοντά στο να επιτύχει

να γίνει ισότιμο μέλος του δικτύου αποκωδικοποίησης και να αποκτήσει πρόσβαση στο περιεχόμενο όλων των πακέτων. Με παρόμοιο τρόπο μπορούμε να αποκλείσουμε τα κακόβουλα πακέτα κάνοντας τους κόμβους να μην τα λαμβάνουν υπόψη [25]. Επειδή ο κακόβουλος κόμβος δεν γνωρίζει τον τρόπο κωδικοποίησης, δεν έχει την ικανότητα να εκπέμψει πακέτο στο δίκτυο το οποίο θα γίνει αποδεκτό από τους υπόλοιπους κόμβους. Δεδομένου ότι τα πακέτα δρομολογούνται από πολλά διαφορετικά μονοπάτια, αυτό καθιστά πολύ δύσκολη τόσο την υποκλοπή όσο και την παραποίηση των περιεχόμενων ενός πακέτου.

2.5 Συμπέρασμα

Η κωδικοποίηση δικτύου έχει πλέον μεγάλο αντίκτυπο στο σχεδιασμό πρωτοκόλλων δικτύωσης και διάδοσης πληροφοριών. Επιτρέποντας την καλύτερη διάδοση του περιεχομένου πληροφοριών στο περιβάλλον δικτύου, μπορεί να απλοποιήσει τους κατανεμημένους αλγόριθμους, προσφέροντας βελτίωση της ταχύτητας μετάδοσης στις υπάρχουσες υποδομές και περισσότερη ασφάλεια από υποκλοπές και επιθέσεις ανάλογα με την υλοποίηση της κωδικοποίησης.

Κεφάλαιο 3 - Homomorphic Encryption (ομομορφική κρυπτογράφηση)

3.1 Εισαγωγή και Σχετικές Δημοσιεύσεις

Η απλή κωδικοποίηση δικτύου θεωρεί εξ ορισμού ότι το σύνολο των κόμβων που εμπλέκονται στη διαδικασία κωδικοποίησης και αποκωδικοποίησης είναι «ειλικρινείς» και εκτελούν τις παραπάνω λειτουργίες ορθά και με τον συμφωνημένο, από όλους τους εμπλεκόμενους κόμβους, τρόπο. Αν όμως κάποιοι ή έστω και ένας από αυτούς είναι κακόβουλος θα προωθεί άκυρους γραμμικούς συνδυασμούς των διανυσμάτων που λαμβάνει. Αυτό δημιουργεί πρόβλημα στους παραλήπτες των πακέτων αφού θα έχουν λάβει κάποια πακέτα τα οποία δεν ανταποκρίνονται σε αυτά που έχουν συμφωνηθεί με αποτέλεσμα να μην μπορούν να συνεχίσουν τη διαδικασία σωστά μιας και δεν μπορούν να ξέρουν πια από τα διανύσματα που έχουν λάβει είναι άκυρα και θα πρέπει να τα αγνοήσουν. Αυτού του τύπου οι επιθέσεις οι οποίες χαρακτηρίζονται σαν μολυσματικές δεν είναι δυνατόν να μειωθούν χρησιμοποιώντας συνηθισμένες υπογραφές ή μεθόδους ελέγχου πρόσβασης στο μέσο (MACs). Είναι προφανές ότι η υπογραφή των επαυξημένων διανυσμάτων των μηνυμάτων δεν μπορεί να βοηθήσει αφού οι παραλήπτες δεν έχουν τα αρχικά διανύσματα των μηνυμάτων και επομένως δεν μπορούν να επαληθεύσουν την υπογραφή. Το να προσπαθήσουμε να βάλουμε υπογραφή στο μήνυμα πριν την εκπομπή του στο δίκτυο δεν θα βοηθήσει σε κάτι. Η εξήγηση είναι απλή. Υπάρχει η πιθανότητα από τα διανύσματα που θα ληφθούν μόνο τα μισά να είναι όπως πρέπει Σε αυτή την περίπτωση θα πρέπει να αποκωδικοποιηθούν πολλά m-υποσύνολα από τους κόμβους λήπτες μέχρι να βρεθεί ένα αποκωδικοποιημένο πακέτο που να ταιριάζει με την υπογραφή. [26].

Συνοψίζοντας και όπως εξηγούν και οι συγγραφείς στο [27] νέοι μηχανισμοί ασφάλειας είναι απαραίτητοι προκειμένου να μειωθούν οι επιθέσεις αυτού του τύπου.

3.2 Homomorphic Encryption

Η κρυπτογράφηση homomorphic (ομομορφική) είναι ένας τύπος κρυπτογράφησης ο οποίος έχει το εξής χαρακτηριστικό: Ας υποθέσουμε ότι έχουμε δεδομένα, ήδη κρυπτογραφημένα με τη συγκεκριμένη μέθοδο (A΄) και ένας τρίτος, τους εφαρμόζει μία νέα συνάρτηση/κωδικοποίηση

παράγοντας ένα νέο σύνολο δεδομένων (Α΄΄). Αυτό το νέο σύνολο δεδομένων αν αποκρυπτογραφηθεί με την ανάστροφη διαδικασία *homomorphic* τότε το παραγόμενο αποτέλεσμα (Β) είναι το ίδιο με το αυτό που θα πρόκυπτε αν ο τρίτος είχε εφαρμόσει τη συνάρτησή του στα αρχικά μη κρυπτογραφημένα δεδομένα [28].

Σχηματικά: Αν υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων Α και εφαρμόσουμε τα παρακάτω:

$$\text{A} \rightarrow \text{homomorphic encryption} \rightarrow \text{A΄}$$

$$\text{A} \rightarrow \text{τυχαία κωδικοποίηση δικτύου} \rightarrow \Gamma$$

$$\text{A΄} \rightarrow \text{τυχαία κωδικοποίηση δικτύου} \rightarrow \text{A΄΄}$$

$$\text{A΄΄} \rightarrow \text{αποκωδικοποίηση homomorphic} \rightarrow \text{B}$$

τότε θα ισχύει: $\text{B}=\Gamma$.

Η ομομορφική κρυπτογράφηση (homomorphic encryption) μπορεί να χρησιμοποιηθεί προκειμένου να παραμείνουν κρυπτογραφημένα δεδομένα στα οποία είναι αναγκαίο να υποστούν διάφορες επεξεργασίες, χωρίς ο τρίτος που θα τις εφαρμόσει να γνωρίζει ή να έχει πρόσβαση στο περιεχόμενό τους. Αυτό επιτρέπει την κρυπτογράφηση των δεδομένων και να μεταφερθούν/αποθηκευτούν τους σε εμπορικά περιβάλλοντα σύννεφων για επεξεργασία, ενώ είναι κρυπτογραφημένα. Σε βιομηχανίες με υψηλό βαθμό ευαισθησίας σχετικά με τη διαφύλαξη του απορρήτου δεδομένων, όπως η υγειονομική περίθαλψη, μπορεί να χρησιμοποιηθεί η ομομορφική κρυπτογράφηση για να ενεργοποιηθούν οι νέες υπηρεσίες, αφαιρώντας τα εμπόδια απορρήτου που εμποδίζουν, με βάση την ισχύουσα νομοθεσία, την ανταλλαγή δεδομένων. Για παράδειγμα, οι προγνωστικές αναλύσεις στην υγειονομική περίθαλψη μπορεί να είναι δύσκολο να εφαρμοστούν λόγω ανησυχιών για την προστασία της ιδιωτικής ζωής των ασθενών, αλλά εάν ο πάροχος υπηρεσιών μπορεί να λειτουργήσει με κρυπτογραφημένα δεδομένα, τα προβλήματα προστασίας της ιδιωτικότητας μειώνονται. Υπάρχουν αρκετοί διαφορετικοί τύποι κρυπτογράφησης οι οποίοι πραγματοποιούν διάφορους υπολογισμούς πάνω στα δεδομένα που πρέπει να κρυπτογραφηθούν. Για την πραγματοποίηση του δικού μας πειράματος χρησιμοποιήσαμε ένα τύπο ομομορφικής κρυπτογράφησης ο οποίος ονομάζεται *Homomorphic MACs* και περιγράφεται στη δημοσίευση [26].

3.3 Ομομορφική Κωδικοποίηση με Έλεγχο Πρόσβασης στο Μέσο. (Homomorphic MAC's)

3.3.1 Σύντομη Περιγραφή

Προκειμένου να σχεδιαστεί ένας τρόπος πρόσβασης στο μέσο (MAC) που μπορεί να χρησιμοποιηθεί για την μείωση των μολυσματικών επιθέσεων οι συγγραφείς στο [26] ακολούθησαν την παρακάτω μεθοδολογία. Με δεδομένα δύο ζευγάρια (v_1, t_1) και (v_2, t_2), όπου $v1, v2 \in \mathbb{F}_q^{n+m}$, μπορεί να δημιουργηθεί μία ετικέτα t για το διάνυσμα $y = a_1 v_1 + a_2 v_2 \ \ \forall \ a_1, a_2 \ \in \ \mathbb{F}_q$. Γενικά, η συγκεκριμένη απόδειξη ασφαλείας δείχνει ότι, ακόμη και στην περίπτωση μιας επίθεση μηνύματος, είναι δύσκολη η δημιουργία μιας έγκυρης ετικέτας για έναν διάνυσμα που δεν θα προέρχεται από μη έγκυρα αρχικά διανύσματα μηνυμάτων. Στις περιπτώσεις όπου αποστολέας και παραλήπτης μοιράζονται ένα κλειδί το οποίο και γνωρίζουν μόνο αυτοί η ομομορφική κωδικοποίηση με έλεγχο πρόσβασης στο μέσο μπορεί να χρησιμοποιηθεί προκειμένου να περιοριστούν οι επιθέσεις από τρίτους κόμβους οι οποίοι έχουν σκοπό να αλλοιώσουν τα περιεχόμενα των μεταδιδόμενων πακέτων. Ο κόμβος εκπομπής, με τη βοήθεια του κλειδιού δημιουργεί μία ετικέτα η οποία ανατίθεται στα αρχικά διανύσματα των πακέτων που θα στείλει στο δίκτυο προς μετάδοση. Οι κόμβοι που παραλαμβάνουν το πακέτο, χωρίς να είναι οι ίδιοι παραλήπτες αλλά πρέπει να προωθήσουν, μπορούν να εκμεταλλευτούν την ομομορφική ιδιότητα δημιουργώντας νέες ετικέτες χωρίς όμως να γνωρίζουν το περιεχόμενο του πακέτου που δεν απευθύνεται σε αυτούς έτσι κι αλλιώς.

Ο παραλήπτης επαληθεύει τις ετικέτες των διανυσμάτων που λαμβάνει (από τους ενδιάμεσους κόμβους) ενώ παράλληλα απορρίπτει όλα τα διανύσματα τα οποία φέρουν μη έγκυρη ετικέτα (*tag*). Η χρησιμότητα της παραπάνω τεχνικής είναι προφανής στις περιπτώσεις όπου αφενός υπάρχουν πολλοί παραλήπτες για κάθε μήνυμα, αφετέρου το μήνυμα θα περάσει από πολλούς ενδιάμεσους κόμβους. Αυτό πέρα από την ασφάλεια που προσφέρει δημιουργεί όμως ένα πρόβλημα: κάθε πακέτο θα πρέπει να περιλαμβάνει μια ετικέτα ανά κόμβο δικτύου, το οποίο φυσικά δεν μπορεί να εφαρμοστεί. Στη δεύτερη φάση συμβαίνει μία μετατροπή από το ομομορφικό MAC που έχουμε ήδη σε ένα ομομορφικό MAC που δύναται να αποσταλεί προς όλους τους κόμβους του δικτύου [29]. Με αυτό τον τρόπο τα παραληφθέντα διανύσματα είναι δυνατόν να επικυρωθούν από όλους τους κόμβους του δικτύου οι οποίοι γνωρίζουν το αρχικό κλειδί εκπομπής. Ο περιορισμός αυτής της μεθόδου είναι ότι λειτουργεί αποδοτικά

αποκλείοντας τα κακόβουλα πακέτα για ένα συγκεκριμένο προκαθορισμένο αριθμό κόμβων c. Αν δηλαδή ο αριθμός των κόμβων αυξηθεί έστω και κατά έναν, η μέθοδος δεν θα έχει τα αναμενόμενα αποτελέσματα και δεν θα είναι πλέον ασφαλής. Το τρίτο βήμα μετατρέπει την ομόμορφη MAC σε ένα έμπιστο σύστημα όπου υπάρχουν πολλοί αποστολείς και πολλαπλοί επαληθευτές. Το τελικό αποτέλεσμα είναι ένα δίκτυο κωδικοποίησης MAC όπου ο κάθε κόμβος μπορεί να είναι αποστολέας, παραλήπτης ή ενδιάμεσος κόμβος.

3.3.2 Ορισμοί

Πρώτα πρέπει να ορίσουμε την ομομορφική MAC και την ασφάλειά της. Ένα διάνυσμα *(q, n, m)* ομομορφικής κωδικοποίησης με έλεγχο πρόσβασης στο μέσο MAC ορίζεται από τρεις πιθανοτικούς αλγόριθμους πολυωνύμου χρόνου, (Sign, Verify, Combine). Ο αλγόριθμος «Sign» (υπογραφή) υπολογίζει μια ετικέτα για ένα διανυσματικό χώρο $V = span\ (v_1, v_2, \dots, v_m)$ υπολογίζοντας την τιμή μιας ετικέτας για ένα βασικό διάνυσμα κάθε φορά. Ο «Combine» (συνδυασμός) είναι ο αλγόριθμός που αναλαμβάνει να υλοποιήσει την ομομορφική ιδιότητα και στον «Verify» απομένει να αναλάβει τα παραλαμβανόμενα ζεύγη και να επαληθεύσει την εγκυρότητά τους. Παρακάτω εξηγούμε πιο αναλυτικά τις λειτουργίες των παραπάνω αλγορίθμων:

- Υπογραφή *Sign (k, id, v, i)*: Αυτή τη συνάρτηση θα μας δώσει σαν έξοδο την ετικέτα *t* με είσοδο ένα μυστικό κλειδί k, το αναγνωριστικό id που ορίζει το διανυσματικό χώρο, το επαυξημένο διάνυσμα $V \in \mathbb{F}_q^{n+m}$ και το $i \in [m]$ το οποίο i μας δείχνει ότι το v είναι το *i-οστό* βασικό διάνυσμα του διανυσματικού χώρου που ορίζεται από το *id*. Με άλλα λόγια ο αλγόριθμος $sign$ υπογράφει ένα διανυσματικό χώρο V που περιέχει m διανύσματα, τρέχοντας τη συνάρτηση $sign$ για i από 1 μέχρι m. Αυτή δημιουργεί μία ετικέτα t για κάθε ένα από τα διανύσματα (t_i). Το αναγνωριστικό id ορίζει τον διανυσματικό χώρο. Όταν μεταδίδεται ένα διάνυσμα v_i στο δίκτυο ο αποστολέας μεταδίδει (id, v_i, t_i). Ο κόμβος παραλήπτης θα συγκεντρώσει όλα τα v_i τα οποία περνούν τη διαδικασία επαλήθευσης και θα ξεκινήσει την αποκωδικοποίηση συνολικά προκειμένου να ανακτηθούν τα αρχικά σύμφωνα με το οποία θα καταλήξει στο αρχικό μήνυμα.
- Συνδυασμός *Combine* $((v_1, t_1, a_1), \dots, (v_m, t_m, a_m))$ Η συνάρτηση $combine$ θα πάρει σαν είσοδο τα m διανύσματα $(v_1, v_m) \in \mathbb{F}_q^{n+m}$ και τις ετικέτες τους t_1, t_m κάτω από

το κλειδί k μαζί με τις m σταθερές $(a_1, \dots, a_m)$ και θα μας δώσει στην έξοδο μία ετικέτα t για το διάνυσμα $y := \sum_{i=1}^{m} a_i v_i \in \mathbb{F}_q^{n+m}$.

- Επαλήθευση *Verify (k, id, y ,t)* Η συνάρτηση της επαλήθευσης παίρνει σαν είσοδο την τιμή του κλειδιού *k* και παράγει έξοδο με δύο πιθανές τιμές: *0* για την απόρριψη του πακέτου και *1* για την αποδοχή του.

Για να καταλήξει σε αυτό το συμπέρασμα παίρνει σαν είσοδο το κλειδί k, το αναγνωριστικό id, ένα διάνυσμα $\boldsymbol{y} \in \mathbb{F}_q^{n+m}$ και την ετικέτα t. Σύμφωνα με όλα τα παραπάνω αυτό που πρέπει να ισχύει προκειμένου να επαληθευτεί σωστά το πακέτο είναι:

$$Verify\left(k, id, \sum_{i=1}^{m} a_i\, v_i, Combine\big((v_1, t_1, a_1), \dots, (v_m, t_m, a_m)\big)\right) = 1$$

- Ασφάλεια. *(Security)* Δίνουμε τη δυνατότητα στον κακόβουλο κόμβο να γνωρίζει υπογραφή σε τυχαίους διανυσματικούς χώρους της επιλογής του (αντίστοιχα με μια επιλεγμένη επίθεση μηνυμάτων σε MAC). Κάθε διανυσματικός χώρος V_i που αποστέλλεται στο δίκτυο από κάθε κόμβο έχει ένα αναγνωριστικό id_i. Ο κακόβουλος κόμβος δεν θα μπορέσει να παραγάγει μία έγκυρη τριάδα *(id, y ,t) στην οποία* το id είτε θα είναι νέο είτε θα ισχύει $id = id_i$ οπότε και τελικά στην τριάδα που στέλνει το $y \notin V_i$.

Σενάριο επίθεσης 1. Έστω ;ότι το $\mathrm{T} = (Sign, Combine, Verify)$ είναι μία τριάδα *(q, n ,m)* τύπου homomorphic MAC. Ορίζουμε την ασφάλεια *T* χρησιμοποιώντας το παρακάτω παράδειγμα ανάμεσα σε ένα κακόβουλο κόμβο B και έναν A που θα πρέπει να απορρίψει τα πακέτα προερχόμενα από τον B.

Ο B δημιουργεί ένα τυχαίο κλειδί $k \overset{R}{\leftarrow} K$. Για να απαντήσει ο κόμβος B σε ένα *(V_i , id_i)* που λαμβάνει από τον A κάνει τα εξής: Έστω ότι τα V_1, $V_2, \dots, V_m \in \mathbb{F}_q^{n+m}$ είναι η βάση για όλα τα V_i. Ο B τώρα πρέπει να υπολογίσει τα MAC για όλα τα βασικά διανύσματα όπως ορίζουμε στον παρακάτω ψευδοκώδικα:

Για j *από* 1 *μέχρι* m

$t_j \overset{R}{\leftarrow} Sign\left(k, id_i, v_j, j\right)$

στείλε $\left(t_1, t_2, \dots, t_m\right)$ *στον* A

Τέλος_επανάληψης

Ο Α υπολογίζει ένα αναγνωριστικό id^* μία ετικέτα t^* και ένα διάνυσμα $y^* \in \mathbb{F}_q^{n+m}$.

3.3.3 Δημιουργία Homomorphic MAC

Η κατασκευή ενός ομομορφικού MAC προέρχεται από ένα κλασικό σύστημα MAC που βασίζεται στην δουλειά των Carter και Wagman [30].

Το MAC λειτουργεί ως εξής:

– Sign(k, id, v, i): Για να δημιουργηθεί μία ετικέτα για το i-οστό βασικό διάνυσμα $v \in \mathbb{F}_q^{n+m}$ με τη χρήση του κλειδιού $k = (k1, k2)$ πρέπει να γίνουν τα παρακάτω:

1. $u \leftarrow G(k_1) \in \mathbb{F}_q^{n+m}$
2. $b \leftarrow F\big(k_2, (id, i)\big) \in \mathbb{F}_q$
3. $t \leftarrow F(u \cdot v) + b \in \mathbb{F}_q$

Για την έξοδο t πρέπει να σημειωθεί ότι είναι ένα απλό στοιχείο από το σύνολο $\mathbb{F}_q$.

– $Combine\big((v_1, t_1, a_1), \dots, (v_m, t_m, a_m)\big)$ έτσι ώστε η έξοδος να προκύπτει από τη σχέση $t \leftarrow \sum_{j=1}^{m} a_j t_j \in \mathbb{F}_q$.

– $Verify(k, id, y, t)$ όπου το $k = (k_1, k_2)$ είναι το μυστικό κλειδί και $y = (y_1, \dots, y_{n+m}) \in \mathbb{F}_q^{n+m}$.

Συνεχίζοντας:

- $u \leftarrow G(k_1) \in \mathbb{F}_q^{n+m}$ $a \leftarrow (u \cdot y) \in \mathbb{F}_q$
- $b \leftarrow \sum_{i=1}^{m} [y_{n+1} \cdot F(k_2, (id, i))] \in \mathbb{F}_q$
- $Αν\ a + b = t\ έξοδος\ 1, αλλιώς\ έξοδος\ 0$

Ολοκληρώνεται έτσι η περιγραφή της ομομορφικής MAC κρυπτογράφησης. Προκειμένου να επαληθευτεί η ορθότητα της μεθόδους θεωρούμε ότι

$$y = \sum_{i=1}^{m} a_i v_i$$

όπου $(v_1, \dots, v_m)$ είναι τα αρχικά ενισχυμένα διανύσματα βάσης και $(t_1, \dots, t_m)$ είναι οι ετικέτες τους. Επομένως, το $(a + b)$ που υπολογίζεται στην συνάρτηση επαλήθευσης Verify ικανοποιεί την

$$a + b = u \cdot y + b = \sum_{i=1}^{m} a_i \cdot \Big((u \cdot v_i) + F\left(k_2, \quad (\mathrm{id}, i)\right)\Big) = \sum_{i=1}^{m} a_i \cdot t_i$$

που είναι ακριβώς η έξοδος της συνάρτησης συνδυασμού $Combine((v_1, t_1, a_1, \dots, v_m, t_m, a_m))$ όπως και είναι προαπαιτούμενο.

Ασφάλεια. Οι συγγραφείς στο [26] απέδειξαν ότι η ασφάλεια, για μία ασφαλής γεννήτρια ψευδοτυχαίων αριθμών G και για μία εξίσου ασφαλής γεννήτρια ψευδοτυχαίων συναρτήσεων F, είναι δεδομένη. Ο επίδοξος εισβολέας δεν μπορεί να επικυρώσει τα δικά του πακέτα και να γίνουν αποδεκτά από τους υπόλοιπους κόμβους χρησιμοποιώντας δικές του συναρτήσεις και ο μόνος τρόπος να τα καταφέρει είναι να αποκτήσει πρόσβαση στις G, F.

Βελτίωση της Ασφάλειας. Κρατώντας υπόψη ότι από το σύνολο $\mathbb{F}_q$ μία ετικέτα στο διάνυσμα v είναι μοναδική, ένας κακόβουλος κόμβος θα μπορούσε να αποκωδικοποιήσει με επιτυχία την συγκεκριμένη ομομορφική μέθοδο με πιθανότητα $\frac{1}{q}$. Όταν το $q = 2^8$ η πιθανότητα να παραβιαστεί η ασφάλεια είναι $\frac{1}{2^8}$. Μπορούμε να έχουμε περεταίρω βελτίωση της ασφάλειας αν για κάθε φορέα δεδομένων, υπολογιστούν πολλαπλά MAC. Για παράδειγμα, με 8 ετικέτες ανά φορέα ασφαλείας η πιθανότητα παραβίασης γίνεται $\frac{1}{q^8}$.. Το μέγεθος των ετικετών διατηρείται επίσης σε σχετικά χαμηλά επίπεδα. Οι συγγραφείς επισημαίνουν ότι ένα ομομορφικό MAC με ασφάλεια $\frac{1}{2^8}$ είναι αρκετό για την εφαρμογή κωδικοποίησης δικτύου από τη στιγμή που χρησιμοποιείται μόνο για την απόρριψη κακόβουλων πακέτων. Ο αποστολέας μπορεί, επιπλέον, να υπολογίσει ένα συνηθισμένο MAC (όπως το HMAC - hash based message authentication code) για το μεταδιδόμενο μήνυμα, πριν την κωδικοποίησή του μηνύματος με τη μέθοδο της κωδικοποίησης δικτύου. Οι παραλήπτες, αφού αποκωδικοποιήσουν ένα πίνακα διανυσμάτων με έγκυρα ομομορφικά MAC, θα προχωρήσουν και στην επικύρωση σύμφωνα με το HMAC και θα απορρίψουν το μήνυμα εάν το αποτέλεσμα από την αποκωδικοποίηση HMAC δεν είναι έγκυρο. Όπως γίνεται αντιληπτό η χρησιμότητα του ομομορφικού MAC δεν αφορά στην απόρριψη όλων των παραπλανητικών μηνυμάτων από τους παραλήπτες. Αυτό που συμβαίνει τελικά είναι ότι απλά οι παραλήπτες θα χρειαστεί να αφιερώσουν περισσότερο υπολογιστικό χρόνο προκειμένου να γίνει σωστή η αποκωδικοποίηση του πακέτου.

3.3.4 Εκπέμποντας Homomorphic MACs: Ορισμοί

Στη συνέχεια περιγράφεται [26] ο τρόπος με τον οποίο το ομόμορφο MAC της προηγούμενης ενότητας θα μετατραπεί σε ένα ομόμορφο MAC εκπομπής. Αυτό θα επιτρέψει σε όλους τους κόμβους του δικτύου (τόσο παραλήπτες όσο και δρομολογητές) να επαληθεύσουν τις ετικέτες στα μεταδιδόμενα πακέτα. Η διαδικασία ξεκινάει ορίζοντας την ασφάλεια για μια ομόμορφη ετικέτα εκπομπής, η οποία θεωρεί πιθανό το γεγονός ότι ένα σύνολο κόμβων θα προσπαθήσει να ξεγελάσει κάποιον άλλο κόμβο. Ένα πακέτο ομομορφικής κωδικοποίησης αποτελείται από μηδενικά με μία πεντάδα (q, n, m, μ, c). Το μ αντιστοιχεί στον αριθμό των κόμβων στο εξεταζόμενο δίκτυο και το c αντιστοιχεί στο μέγιστο αριθμό των κόμβων που μπορεί να παρουσιάσουν κακόβουλη συμπεριφορά και τελικά να προσπαθήσουν όλοι μαζί προκειμένου να οδηγήσουν κάποιο κόμβο προς την παραλαβή ενός κακόβουλου πακέτου. Για τη δημιουργία του εκπεμπόμενου πακέτου σύμφωνα με την ομομορφική κωδικοποίηση με έλεγχο πρόσβασης στο μέσο (q, n, m, μ, c), απαιτούνται τέσσερις αλγόριθμοί. Εξηγούμε παρακάτω τον καθένα από αυτούς ξεχωριστά:

- *Setup* (λ, μ, c): (ρύθμιση) Ο συγκεκριμένος αλγόριθμός παίζει το ρόλο της αρχικής ρύθμισης. Παίρνει σαν εισόδους μία παράμετρο ασφάλειας λ, τον αριθμό των κόμβων μ, και c όπως αναφέραμε και πριν είναι το όριο που θέτουμε για τον αριθμό των κακόβουλων κόμβων που μπορεί να συνεργαστούν προκειμένου να ξεγελάσουν ένα παραλήπτη. Η παραγόμενη έξοδος από αυτή τη συνάρτηση είναι ένα σύνολο $\mu + 1$ κλειδιών $k, k_1, \ldots, k_\mu$. Τα κλειδιά ανατίθενται στους παραλήπτες κόμβους που θα επαληθεύσουν τα πακέτα καθώς και στον αρχικό αποστολέα.
- Οι συναρτήσεις *Sign, Combine, Verify* δημιουργούνται με τον ίδιο ακριβώς τρόπο, όπως και προηγουμένως (3.3.2 Ορισμοί), με τη μοναδική διαφορά ότι πλέον πρέπει να συμπεριληφθούν και τα κλειδιά στις συναρτήσεις υπογραφής «*Sign*» (k) και επαλήθευσης «*Verify*» k_i . Ένα για κάθε κόμβοι i που φυσικά αποτελεί μέρος του σύνολο μ των κόμβων.

Ασφάλεια: Στη συνέχεια, ορίζεται η ασφάλεια έναντι των κόμβων c. Στον A κόμβο δίνεται ένας αριθμός κλειδιών c και ο στόχος του είναι να δημιουργήσει ένα ζεύγος μηνύματος ετικέτας που θα επαληθεύει το πακέτο χρησιμοποιώντας ένα άλλο γνήσιο κλειδί επαληθευτή το οποίο ο *Α δεν θα το γνωρίζει*.

Έστω ότι το σύνολο από το αποτέλεσμα των αλγορίθμων (*Setup, Sign, Combine, Verify*) είναι (q, n, m, μ, c) το οποίο είναι και η εκπομπή ομόμορφων MAC. ΘΑ εξετάσουμε τώρα τη

συμπεριφορά που θα παρουσιάσει συνολικά η τεχνική όταν ένα κακόβουλος κόμβος C βρεθεί σε «αντιπαράθεση» με τον κόμβο A.

Ο A στέλνει στον C τους δείκτες των κόμβων c που λειτουργούν ως επαληθευτές $\{ i_1,\ldots, i_c \}$. Ο C εκτελεί τον αλγόριθμο ρύθμισης $Setup\ (\lambda, \mu, c)$ προκειμένου να αποκτήσει τα κλειδιά $k, k_1,\ldots, k_\mu$ και στέλνει τα κλειδιά $\{ k_{i1},\ldots, k_{i\mu} \}$ στον A.

Ο A υποβάλλει ερωτήματα MAC στον C και αυτός απαντά χρησιμοποιώντας το κλειδί k του αποστολέα

Ο A δημιουργεί ένα δείκτη $j^* \in [\mu] \setminus \{i_1, \ldots, i_c\}$, ένα αναγνωριστικό id^*, μια ετικέτα t^* και ένα διάνυσμα $y^* \in \mathbb{F}_q^{n+m}$.

Ο A "κερδίζει" το παιχνίδι ασφαλείας εάν $Verify(k, id^*, y^*, t^*) = 1$ και ισχύουν τα παρακάτω:

1. Αν για όλα τα i ίσχύει ότι $id^* \neq id_i$ τότε έχουμε πλαστογραφία τύπου 1
2. Αν για κάποια από τα i και $y^* \notin \Omega$ι έχουμε $id^* = id_i$ τότε έχουμε πλαστογραφία τύπου 2).
3. Αν $y^* = (y^*_1, \ldots, y^*_{n+m})$ τότε η $(y^*_{n+1}, \ldots, y^*_{n+m})$ στο y^* δεν είναι το διάνυσμα που αποτελείται μόνο από μηδενικά. Αυτό σημαίνει ότι η πλαστογραφία που εντοπίστηκε δεν δημιουργεί πρόβλημα στη μετάδοση.

Το πλεονέκτημα BNC-Adv [A, T] του A σε σχέση με το T ορίζεται ως η πιθανότητα να κερδίσει το A αυτό το παιχνίδι ασφαλείας. Άρα μία (q, n, m, μ, c) ομομορφική εκπομπή T *με τη μέθοδο* MAC είναι ασφαλής εάν για όλους τους πολυωνύμου χρόνου A, αν το μέγεθος BNC-Adv $[A, T]$ είναι αμελητέο.

3.3.5 Βασική Διαχείριση για Ομομορφικά MAC Πολλαπλών Αποστολέων

Το σχήμα διάδοσης κλειδιών που περιγράφεται στην προηγούμενη ενότητα υποστηρίζει μόνο έναν αποστολέα στο δίκτυο. Στην πραγματικότητα όμως σε ένα δίκτυο, μπορεί να υπάρχουν περισσότεροι από έναν κόμβοι οι οποίοι αποστέλλουν πακέτα. Επιπλέον κάποιος κόμβος για παράδειγμα μπορεί να έχει και τους τρεις ρόλους. Προκειμένου να είναι δυνατή η εφαρμογή της

συγκεκριμένης μεθόδου οφείλουμε να δημιουργήσουμε ένα σύνολο κλειδιών το οποίο και θα ανατεθεί σε κάθε αποστολέα.

Για να επιτευχθεί αυτό, το βασικό σχήμα διάδοσης, σύμφωνα με την δουλειά των συγγραφέων στο [26] τροποποιείται ως εξής: Πλέον χρειαζόμαστε δύο διαφορετικά σύνολο από κλειδιά. Το ένα θα χρησιμοποιείται από τον κόμβο όποτε θέλει να αποστείλει δεδομένα στο δίκτυο και το δεύτερο θα χρησιμοποιείται από τον κόμβο προκειμένου να έχει τη δυνατότητα να επαληθεύει τα δεδομένα που λαμβάνει. Όλοι οι κόμβοι του δικτύου αποκτούν ένα μοναδικό αναγνωριστικό. Υποδηλώνουμε το αναγνωριστικό αποστολέα του κόμβου i από το αναγνωριστικό του αποστολέα (sender) id_i (sid_i).

Ορίζουμε σαν $\mathbb{X}$ το σύνολο των κλειδιών και το $\mathbb{B}$ είναι ένα σύνολο το οποίο περιέχει το σύνολο των υποομάδων κλειδιών του $\mathbb{X}$. Τα μέλη του $\mathbb{X}$ συμβολίζονται φυσικά με $x_1, x_{2\ ,} \dots, x_l$. Η ομάδα συνόλων $(\mathbb{X}, \mathbb{B})$ συσχετίζεται με το δίκτυο. Στον κόμβο i με αναγνωριστικό αποστολέα sid_i δίδονται δύο σύνολα πλήκτρων $K_{1,i}$ και $K_{2.i}$ με το πρώτο να χρησιμοποιείται προκειμένου να «υπογράψει» ένα μήνυμα και το δεύτερο να χρησιμοποιείται προκειμένου να επικυρώσει την αυθεντικότητα ενός παραληφθέντος μηνύματος.

Ισχύει πάντα ότι $\left|K_{1,i}\right| = \ell \ \forall \ i = 1, \dots, \mu$. Για το κλειδί $K_{2.i}$ ισχύει ότι: $K_{2.i} \in \mathbb{B}$.

Τα παραπάνω είναι αρκετά προκειμένου ο κόμβος να μπορεί να ανταπεξέλθει τόσο στο ρόλο του αποστολέα (κωδικοποίηση των πακέτων) όσο και του παραλήπτη (επαλήθευση). Ένας κόμβος δημιουργεί ℓ αριθμό ετικετών για το πακέτο επειδή ο αριθμός των κλειδιών που του έχουν ανατεθεί από το σύνολο $K_{1,i}$, υπενθυμίζουμε ότι είναι ℓ. Με τον παραπάνω τρόπο ο κόμβος καθίσταται ικανός προκειμένου να «υπογράψει» ένα συγκεκριμένο πακέτο p. Ένας άλλος κόμβος, έστω i, ο οποίος λαμβάνει ένα πακέτο p, χρειάζεται να το επαληθεύσει. Ο κόμβος θα διαβάσει το αναγνωριστικό αποστολέα (id) του πακέτου, sid_p. Από αυτά τα δεδομένα ο κόμβος θα κάνεις τις κατάλληλες πράξεις και θα υπολογίσει τα b κλειδιά που χρειάζεται προκειμένου να προχωρήσει στην επαλήθευση του πακέτου. Προφανώς τα b κλειδιά υπολογίζονται κάθε φορά που ο κόμβος λαμβάνει ένα πακέτο.

Άρα ακόμα και αν γνωρίζει κάποια κλειδιά ένας κακόβουλος κόμβος αυτό δεν είναι αρκετό προκειμένου να καταφέρει κάποιον άλλο κόμβο να αναγνωρίσει το πακέτο του σαν γνήσιο. Τα κλειδιά που μπορεί να γνωρίζει είναι μόνο «απλά» κλειδιά μέσα από σύνολο κλειδιών $\mathbb{X}$ και όχι

αυτά που χρησιμοποιούν τελικά οι κόμβοι προκειμένου να επαληθεύσουν τα πακέτα και τα οποία υπολογίζονται από τον κόμβο όποτε υπάρξει λήψη. Επειδή η συνάρτηση F μέσω της οποίας γίνονται οι υπολογισμοί είναι μία ασφαλή ψευδοτυχαία συνάρτηση, η τιμή $F(x, sid_i)$ δεν αποκαλύπτει καμία πληροφορία σχετικά με το $F(x, sid_j)$ αν το $sid_i \neq sid_j$ και έτσι δεν μπορεί να υπάρξει συνεννόηση ανάμεσα στους κόμβους.

3.3.6 Πεδίο Εφαρμογών της Homomorphic Encryption

Οι συγγραφείς [26] ανέπτυξαν σε κώδικα την ομομορφική εκπομπή MAC που περιγράφεται παραπάνω προκειμένου να αξιολογηθεί η απόδοσή της. Τα μηνύματά επιλέχθηκαν ως διανύσματα μήκους 1024 πάνω από το πεδίο $\mathbb{F}_{2^8}$ και οι συντελεστές κωδικοποίησης δικτύου επιλέχθηκαν τυχαία από το $\mathbb{F}_{2^8}$. Στα δίκτυα που πραγματοποιήθηκα τα πειράματα επιλέχθηκε να υπάρχει μόνο ένας κόμβος ο οποίος θα αποστέλλει πακέτα στο δίκτυο. Για την παραπάνω ρύθμιση πραγματοποιήθηκαν δύο δοκιμές με τα παρακάτω χαρακτηριστικά:

- Μια όπου το σύνολο $\mathbb{X}$ περιέχει 49 κλειδιά που προκύπτουν από 7 μπλοκ. Σε αυτή την περίπτωση ο αριθμός των κόμβων που μπορούν με επιτυχία να επαληθεύσουν τα πακέτα είναι $\mu = 7^4 = 2401$.
- Μια όπου το σύνολο $\mathbb{X}$ περιέχει 121 κλειδιά που προκύπτουν από 11 μπλοκ. Σε αυτή την περίπτωση ο αριθμός των κόμβων που μπορούν με επιτυχία να επαληθεύσουν τα πακέτα είναι $\mu = 7^{11} = 14641$.

Ο κόμβος που έχει το ρόλο του αποστολέα στέλνει 5 μηνύματα (m=5), με το καθένα να είναι ένα 1 kilobyte. Κάθε μήνυμα «υπογράφεται» στην μία περίπτωση με 49 (7^2) κλειδιά από τον αποστολέα και στην δεύτερη με 121 (11^2).

Δεδομένου ότι το ομομορφικό MAC που παρουσίασαν απαιτεί γρήγορους πολλαπλασιασμούς στο $\mathbb{F}_{2^8}$, θεώρησαν σωστό να δημιουργήσουν έναν πίνακα πολλαπλασιασμού εκτός σύνδεσης, ο οποίος αποθηκεύει και τα 216 γινόμενα των ζευγών ζεύγους στοιχείων $\mathbb{F}_{2^8}$ (look up table). Ο πίνακας αυτός παρέχει έτοιμα τα γινόμενα, κάνοντας απλά μια αναζήτηση σε ένα μικρό πίνακα. Εφάρμοσαν την ψευδοτυχαία συνάρτηση F και την ψευδοτυχαία γεννήτρια G

χρησιμοποιώντας Advanced Encryption Standard (AES) από την ανοιχτή βιβλιοθήκη OpenSSL. Χρονομέτρησαν τις ακόλουθες λειτουργίες:

1. Υπογραφή: Η κόμβος εκπομπής δημιουργεί τις απαιτούμενες ετικέτες για το μήνυμα που θα εκπέμψει.
2. Συνδυασμός και επαλήθευση: Ο κόμβος που δέχεται το πακέτο θα λάβει 5 ζευγάρια από (μήνυμα, ετικέτα) και θα αναλάβει να υπολογίσει από τα παραλαμβανόμενα 5 διανύσματα και τις ετικέτες τους, τα κλειδιά που θα τον οδηγήσουν στην επαλήθευση του πακέτου.

Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα και για τις δύο περιπτώσεις. Οι χρόνοι είναι αναγραμμένοι σε μικροδευτερόλεπτα. Τα αποτελέσματα προέκυψαν από τους μέσους όρους των χρόνων που λήφθηκαν [26].

Αριθμός κλειδιών	Υπογραφή	Συνδυασμός και Επαλήθευση	Μέγεθος Ετικέτας (bytes)	Ασφάλεια
p=7	430,3	88,5	49	$(0,5)^{8}$
p=11	1329,3	161,5	121	$(0,5)^{40}$

Πίνακας 4: <u>Αποτελέσματα Εξομοίωσης</u>

Αυτά τα πειράματα διεξήχθησαν σε ένα σύστημα GNU / Linux με 4 επεξεργαστές Intel Xeon 3 Ghz με συμμετρική υποστήριξη πολλαπλής επεξεργασίας.

3.3.7 Συμπεράσματα

Παραπάνω παρουσιάστηκε ένα ομομορφικό MAC κατάλληλο για δίκτυα που χρησιμοποιούν κωδικοποίηση δικτύου και το οποίο χρησιμοποιήθηκε για τη δική μας εργασία σχετικά με τη σύγκριση των δύο τεχνικών και την εξαγωγή συμπερασμάτων σχετικά με την απόδοσή τους. Η ομομορφική κωδικοποίηση με έλεγχο πρόσβασης στο μέσο μπορεί να μετατραπεί σε ομομορφική κρυπτογράφηση με έλεγχο πρόσβασης στο μέσω προκειμένου να γίνει εκπομπή πακέτων. Η παραγόμενη εκπομπή MAC είναι ανθεκτική σε συμπαιγνία έως ένα προκαθορισμένο όριο κακόβουλων κόμβων c. Το μέγεθος της ετικέτας προκύπτει υψώνοντας το c στο τετράγωνο (c^2). Τα πειραματικά αποτελέσματα δείχνουν ότι το MAC αποδίδει καλά

για συνδέσεις από σημείο προς σημείο. Ως εκπομπή (από έναν κόμβο προς όλους) MAC λειτουργεί καλά αρκεί ο αριθμός των κακόβουλων κόμβων c να παραμένει σχετικά μικρός.

Κεφάλαιο 4 - Πραγματοποίηση Πειραμάτων

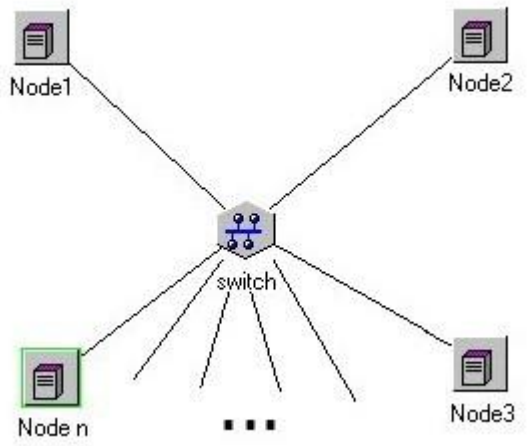
4.1 Εισαγωγή

Στα πειράματα που ακολουθούν θα προσπαθήσουμε να επαληθεύσουμε τα συμπεράσματα των αρχικών δημοσιεύσεων στις οποίες τα βασίσαμε. Η ανακάλυψη πιθανών συσχετισμών ανάμεσα στις μεθόδους είναι το δεύτερο ζητούμενο. Τα πλεονεκτήματα και τα μειονεκτήματα που εμφανίζονται στην κάθε υλοποίηση θα επισημανθούν και φυσικά ο βασικός στόχος είναι να δούμε αν υπάρχει η δυνατότητα του συνδυασμού της υπολογιστικής κωδικοποίησης με κάθε μία ξεχωριστά από της μεθόδους κωδικοποίησης δικτύου προκειμένου να δημιουργηθεί ένα νέο σχήμα το οποίο θα παρέχει βελτιώσεις σε επιμέρους τομείς όπως η ασφάλεια και η ταχύτητα.
Προκειμένου να επαληθεύσουμε την βελτιστοποίηση που προκύπτει από τη χρήση του αλγόριθμου *Coded TeraSort* πραγματοποιήσαμε μετρήσεις χρησιμοποιώντας τον ίδιο αριθμό κόμβων και στη συνέχεια συγκρίναμε τα παραγόμενα αποτελέσματα. Αυτό που μας ενδιαφέρει είναι τόσο ο συνολικός χρόνος εκτέλεσης της διαδικασίας όσο και η κίνηση που δημιουργείται στο δίκτυο *(BandWidth).*

4.2 Πείραμα *Coded Computing*

Για την πραγματοποίηση του δικού μας πειράματος εξομοιώσαμε ένα κατανεμημένο δίκτυο στο οποίο δεν υπάρχουν λάθη στη μετάδοση ή παρεμβολές από τρίτους. Οι συγγραφείς από τα [1] και [11] μοιράστηκαν τον κώδικα σε C++ που ανέπτυξαν χρησιμοποιώντας την ανοιχτού κώδικα βιβλιοθήκη *MPI* [31], μέσω της διαδικτυακής πλατφόρμας *GitHub* [32]. Μέσω αυτής μπορέσαμε και πραγματοποιήσαμε τις δικές μας μετρήσεις προκειμένου να πάρουμε κάποιες ενδεικτικές τιμές σχετικά με την ταχύτητα και τις απαιτήσεις σε BW χρησιμοποιώντας τον αλγόριθμο *Coded TeraSort*. Επειδή δεν ήταν δυνατόν να επαναλάβουμε το πείραμα με τις ίδιες ακριβώς συνθήκες αρκεστήκαμε στο αρχείο που διέθεσαν οι ίδιοι μαζί με τον πηγαίο κώδικα. Το ίδιο αρχείο χρησιμοποιήσαμε και στις επόμενες μετρήσεις όπου προσπαθήσαμε να εφαρμόσουμε τον αλγόριθμο *Random Linear Network Coding* [3] [4]. Τα προγράμματα εκτελέστηκαν σε μία εικονική μηχανή στην οποία φορτώσαμε ένα Λειτουργικό Σύστημα fedora. Στην πειραματική διαδικασία δοκιμές αυξάναμε κάθε φορά κατά ένα, το φορτίο *r*. Αυτό σημαίνει ότι:

1. Κάθε φορά αυξάναμε κατά ένα τον αριθμό των κόμβων που επεξεργάζονταν το κάθε κομμάτι του αρχείου ενώ
2. παράλληλα αυξάναμε τον αριθμό των αρχείων στα οποία «χωρίζαμε» το αρχικό αρχείο, μεγέθους 0,33MB. Η εξομοίωση μετράει τους χρόνους που απαιτούνται για κάθε φάση καθώς και την κίνηση που δημιουργείται στο δίκτυο.



Εικόνα 7: Σχήμα εικονικής πειραματικής διάταξης

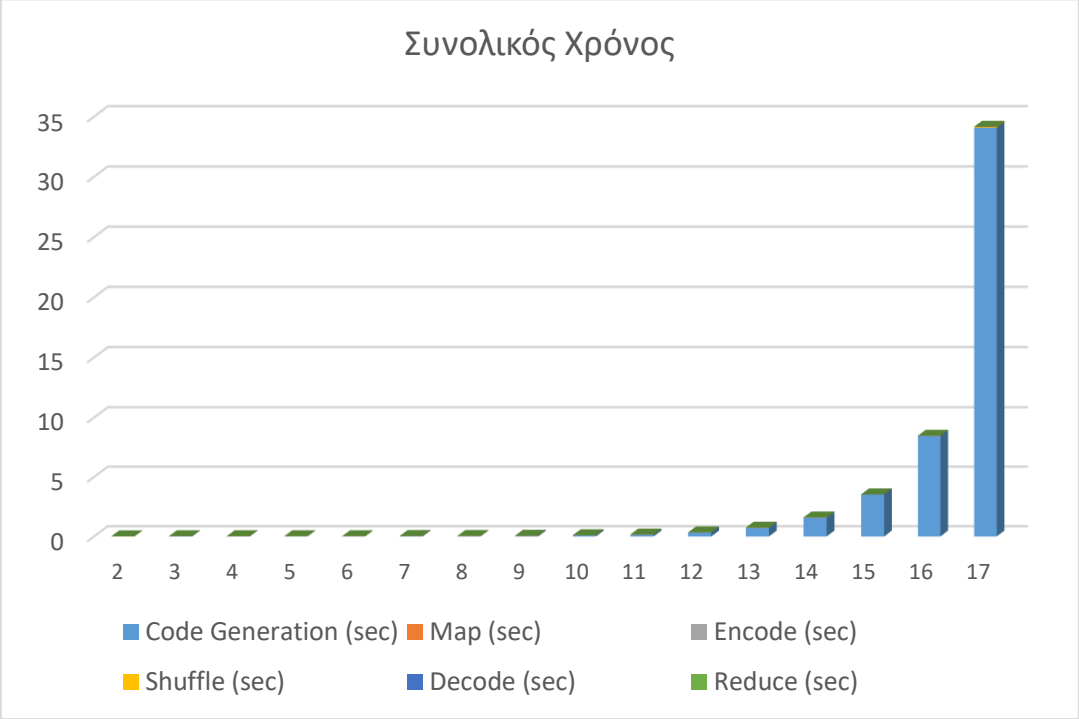
Ακολουθούν πίνακες μετρήσεων στους οποίους αναγράφονται οι τιμές που καταγράφηκαν κατά την εκτέλεση του πειράματος. Στην περίπτωση των δύο κόμβων δεν υπήρξε στην ουσία ανταλλαγής αρχείων στο δίκτυο μιας και ο κάθε κόμβος είχε εξαρχής και τα δύο απαιτούμενα αρχεία. Για αυτό και ο χρόνος που απαιτείται για το Code Generation και το Shuffle είναι στην ουσία μηδενικός. Το ίδιο φυσικά ισχύει και για το απαιτούμενο BW. Η εξομοίωση που πραγματοποιήσαμε έδινε αποτελέσματα για μέσους και μέγιστους χρόνους εκτός από τη διαδικασία Shuffle όπου ο χρόνος ήταν κοινός και δεν είχε διάφορες τιμές.

Αριθμός Κόμβων	Code Generation (sec)	Map (sec)	Encode (sec)	Shuffle (sec)	Decode (sec)	Reduce (sec)	Συνολικός χρόνος (sec)
2	0,0000675	0,010126	0,0000035	0,000022	0,002237	0,0083525	0,0208085
3	0,0121083	**0,006868**	0,001401	0,001852	0,00147467	0,005441	0,02914497
4	**0,0106125**	0,0080005	0,00119075	0,001701	0,00127525	0,003867	**0,026647**
5	0,0127236	0,0089782	0,0010468	0,002009	0,0010004	0,003061	0,028819
6	0,0114407	0,0103208	0,001103733	**0,001042**	0,000983667	0,00257367	0,02746457
7	0,026926	0,0102433	**0,000998429**	0,001434	**0,000926**	0,00206857	0,042596299
8	0,031203	0,0105699	0,001033	0,002304	0,0009845	0,00169575	0,04779015
9	0,0523986	0,0119216	0,00119578	0,00225	0,001106	0,00159722	0,0704692
10	0,0884795	0,0110529	0,0012294	0,001682	0,0011808	0,0013989	0,1050235
11	0,152766	0,0117023	0,00132091	0,002302	0,00118527	0,001232	0,17050848
12	0,330895	0,010126	0,00146083	0,002088	0,00132817	0,00109933	0,34699733
13	0,725137	0,012435	0,00169177	0,002551	0,00148638	0,00102662	0,74432777
14	1,5816	0,013308	0,00192521	0,002901	0,00167571	0,000927071	1,602336991
15	3,5092	0,0129812	0,00202853	0,002828	0,0187233	0,000849867	3,546610897
16	8,44654	0,0154096	0,00237181	0,003872	0,00211831	**0,0007525**	8,47106422
17	34,0722	0,0212271	0,00299547	0,054569	0,00284053	0,000810706	34,15464281
Ελάχιστες τιμές	0,0106125	0,006868	0,000998429	0,001042	0,000926	0,0007525	0,026647

Πίνακας 5: <u>Αποτελέσματα Εξομοίωσης</u>

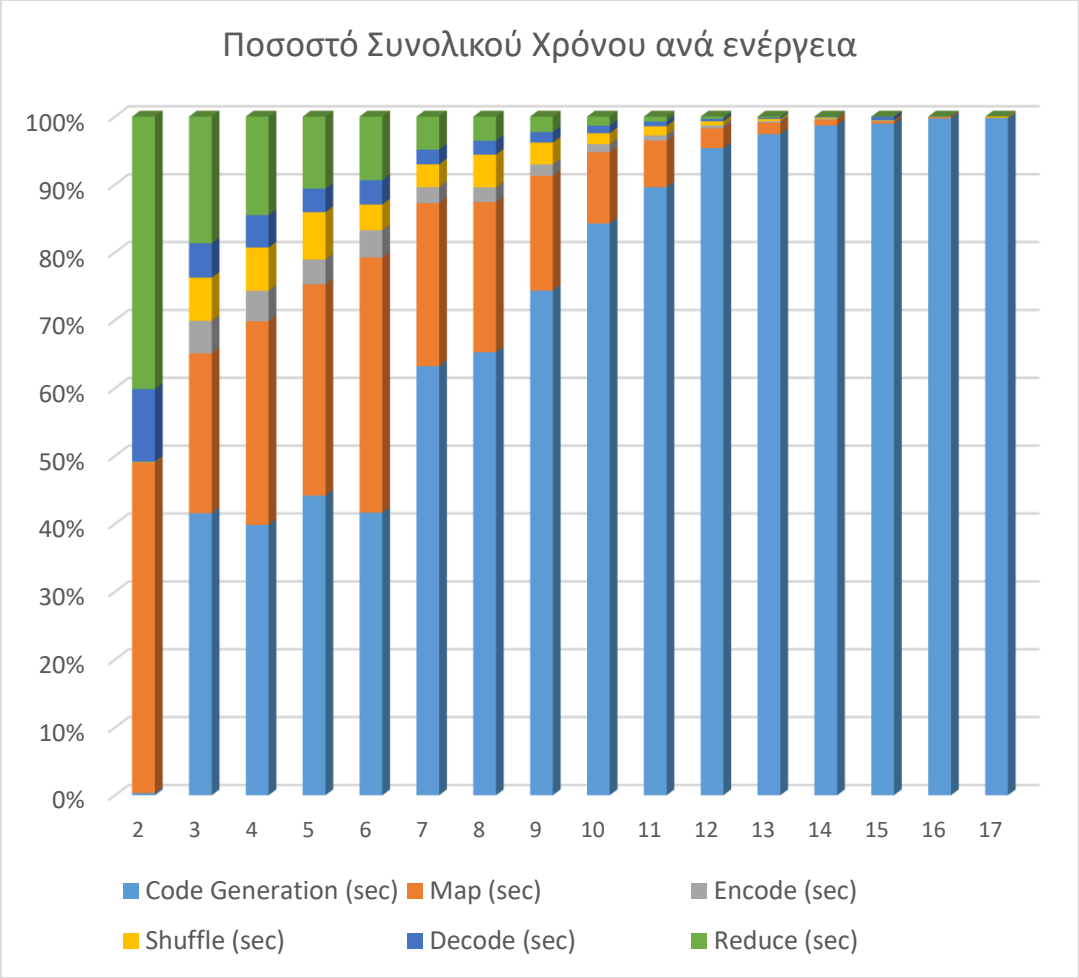
Αριθμός Κόμβων	Code Generation (sec)	Map (sec)	Encode (sec)	Decode (sec)	Reduce (sec)
2	0,000068	0,010313	0,000004	0,002244	0,008532
3	0,013284	0,006933	0,001515	0,001643	0,005646
4	0,011818	0,008222	0,001386	0,001607	0,004236
5	0,014372	0,009213	0,001073	0,001018	0,00328
6	0,012048	0,01085	0,001123	0,001056	0,002604
7	0,028658	0,010617	0,001113	0,000977	0,002275
8	0,033193	0,010863	0,001076	0,001044	0,001743
9	0,055302	0,01232	0,001301	0,001338	0,001676
10	0,090123	0,011365	0,001319	0,001344	0,001445
11	0,155107	0,015057	0,001428	0,001225	0,001351
12	0,333421	0,011922	0,001497	0,001357	0,001193
13	0,728261	0,012715	0,001753	0,001564	0,001086
14	1,58725	0,016547	0,002105	0,001754	0,00102
15	3,51408	0,015796	0,0021	0,002105	0,001066
16	8,45053	0,021504	0,00265	0,002215	0,000795
17	105,491	0,031228	0,004032	0,00378	0,001102

Πίνακας 6: <u>Αποτελέσματα Εξομοίωσης (μέγιστοι χρόνοι)</u>



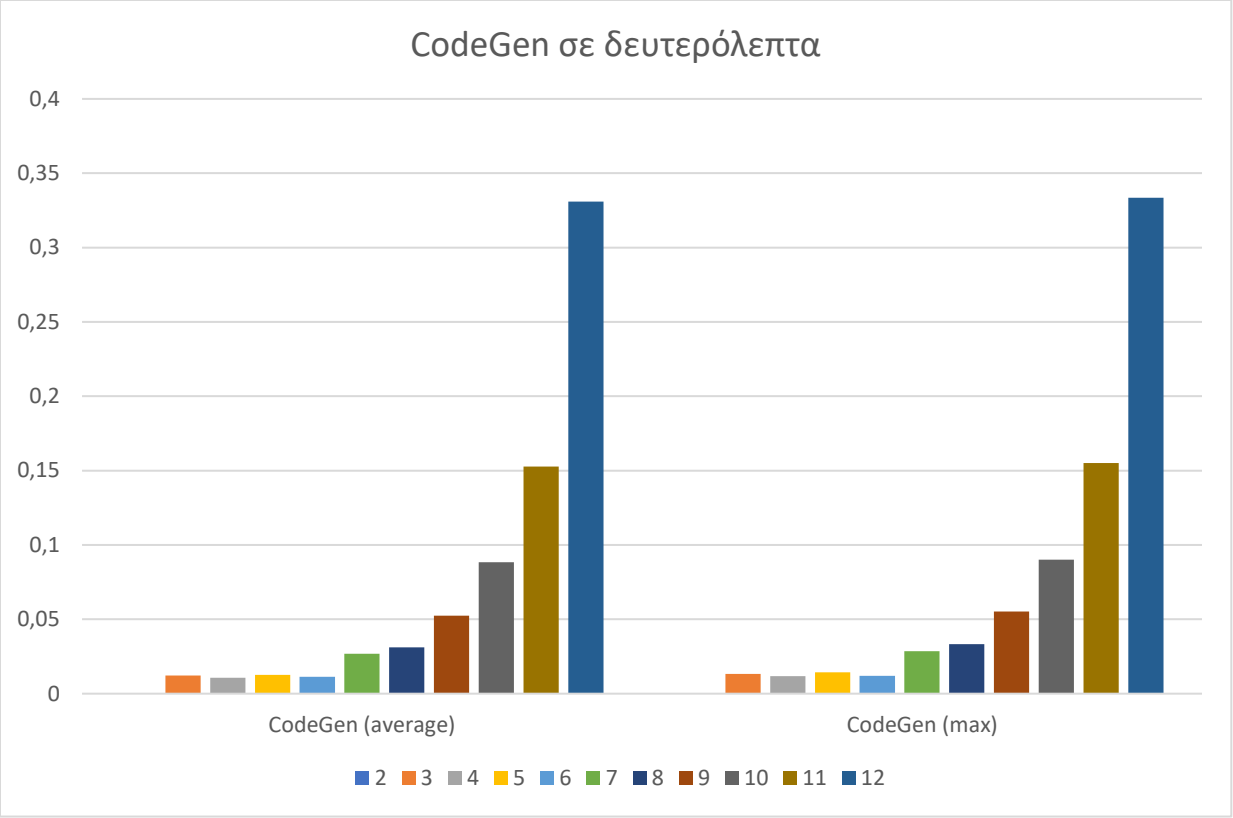
Διάγραμμα 2: <u>Συνολικός χρόνος</u>

Στο παραπάνω διάγραμμα φαίνεται η εξέλιξη του συν ολικού χρόνου που απαιτείται για να ολοκληρωθεί η διαδικασία σε σχέση με την αύξηση του αριθμού του φορτίου *r*. Για τιμές του *r* από 11 και πάνω ο απαιτούμενος χρόνος για την ολοκλήρωση της διαδικασίας αυξάνεται πάρα πολύ με αποτέλεσμα να μην είναι δυνατό να συμπεριληφθούν στο ίδιο διάγραμμα. Σύμφωνα με τα αποτελέσματα παρατηρούμε μία βελτίωση στου συνολικού χρόνου μέχρι τον αριθμό των επτά κόμβων όπου όπως θα φανεί και παρακάτω η επεξεργασία των δεδομένων απαιτεί ολοένα και περισσότερο χρόνο οδηγώντας σε μειωμένη απόδοση τον αλγόριθμό.



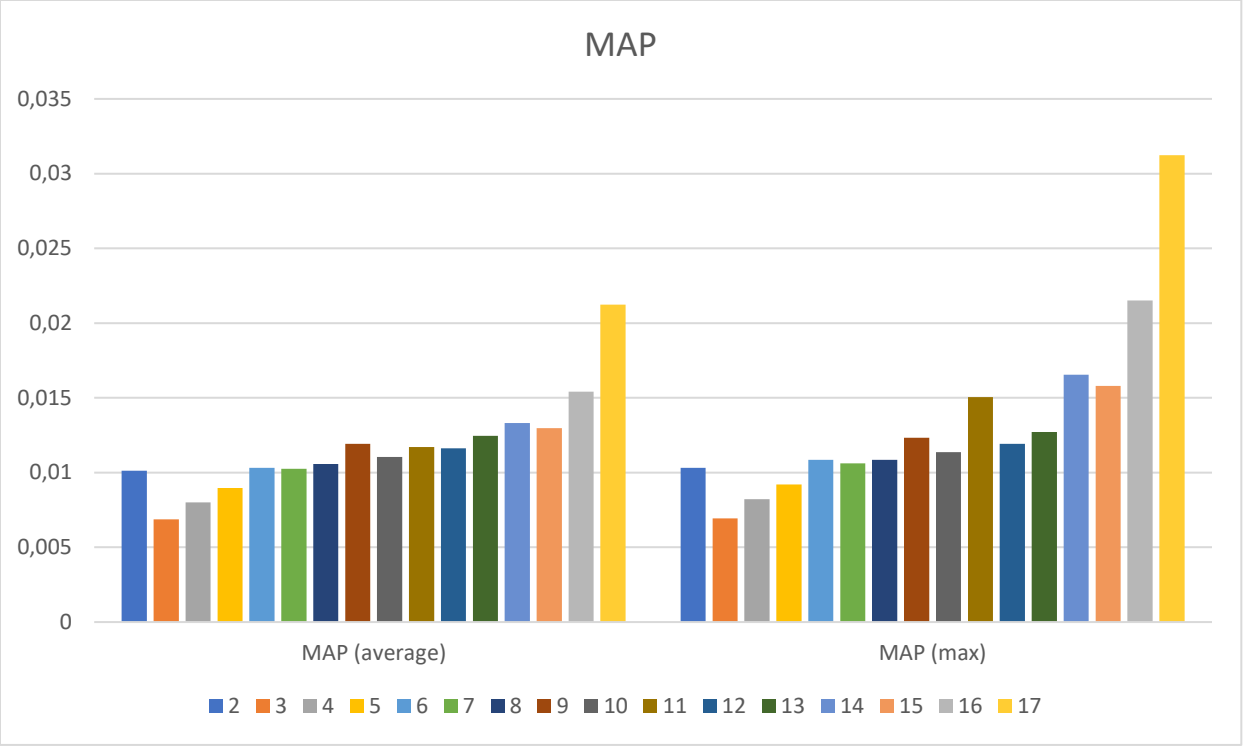
Διάγραμμα 3: <u>Κατανομή συνολικού χρόνου</u>

Όπως είναι αναμενόμενο με την αύξηση του αριθμού των κόμβων και το τεμάχισμα των δεδομένων προς επεξεργασία απαιτεί ολοένα και περισσότερο χρόνο σε αντίθεση με την τελική τους επεξεργασία. Αντιθέτως η επεξεργασία των τεμαχισμένων δεδομένων μειώνεται με την αύξηση του φορτίου που ανατίθεται στον κάθε κόμβο μιας και έχουμε μικρότερα τεμάχια από το αρχικό αρχείο να ανατίθενται στους κόμβους.



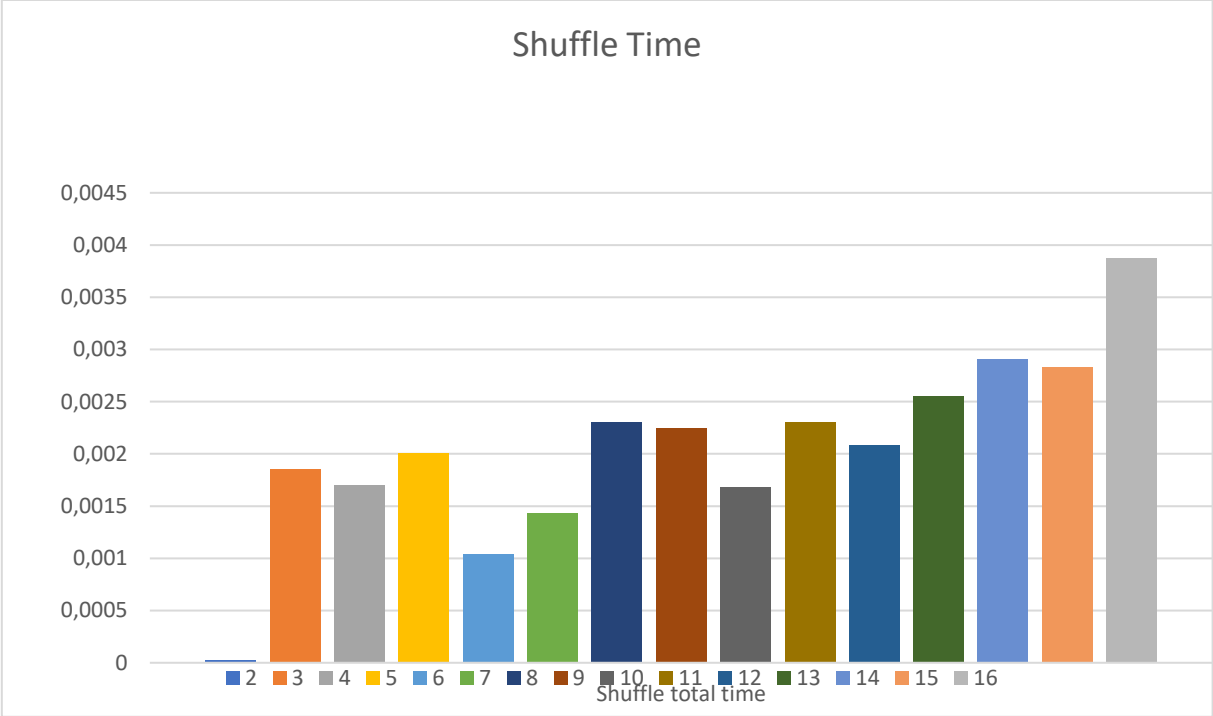
Διάγραμμα 4: <u>Δημιουργία κώδικα</u>

Στο διάγραμμα 3 φαίνεται ακόμα πιο ξεκάθαρα η αύξηση του απαιτούμενου χρόνου για το σπάσιμο των αρχικών δεδομένων σε μικρότερα κομμάτια προκειμένου να ανατεθούν στους κόμβους.

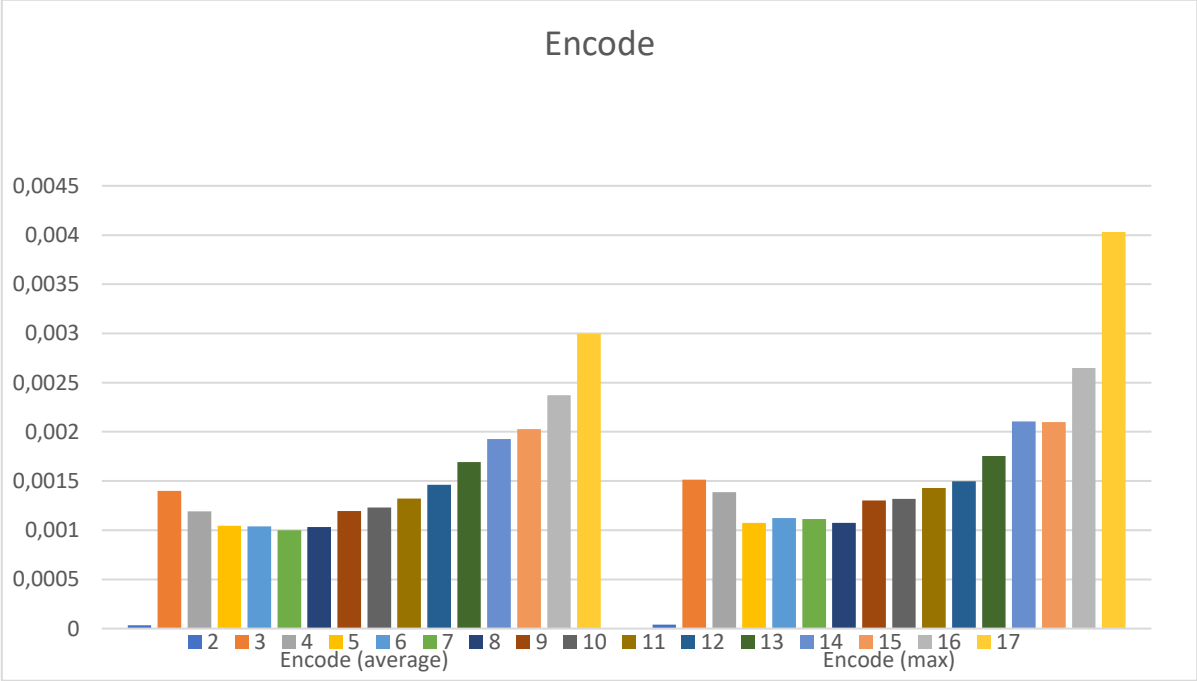


Διάγραμμα 5: <u>Διαδικασία map</u>

Για τη διαδικασία *MAP* τα αποτελέσματα είναι αναμενόμενα. Με την αύξηση του αριθμού του φορτίου *r* των κόμβων έχουμε αύξηση του συνολικού χρόνου που απαιτείται για τη διαδικασία διαχωρισμού των δεδομένων. Το ίδιο ισχύει και για τη διαδικασία *SHUFFLE* που τα αποτελέσματά της απεικονίζονται στον παρακάτω πίνακα.



Διάγραμμα 6: Διαδικασία shuffle



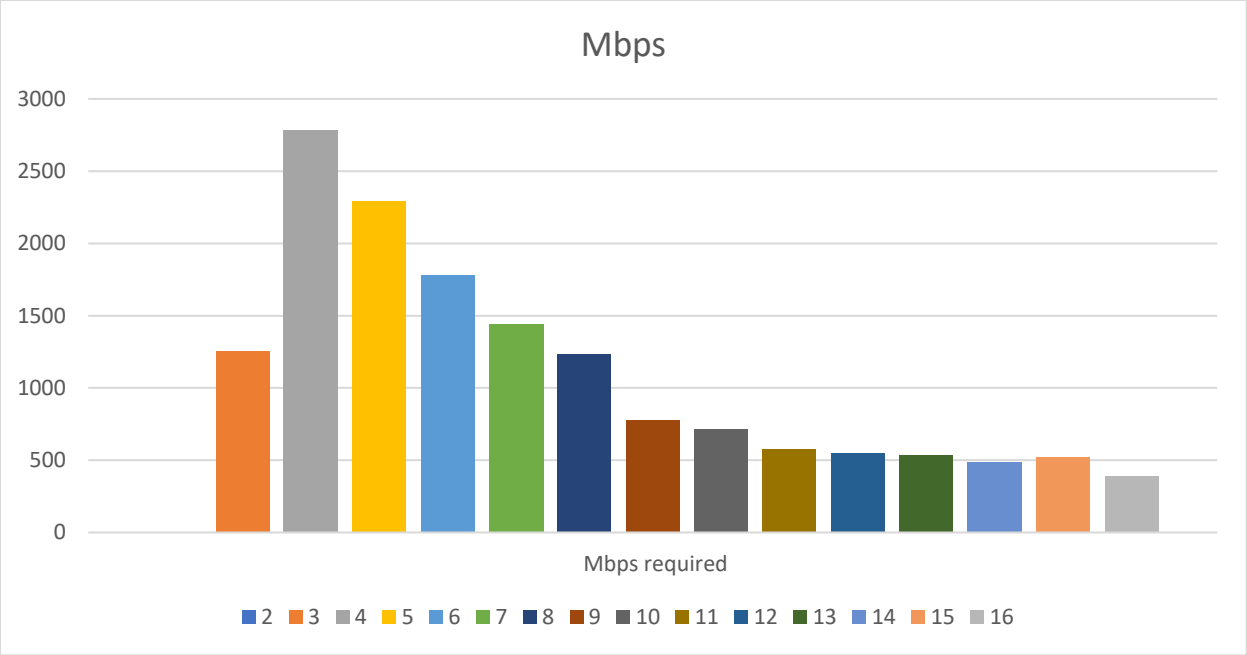
Διάγραμμα 7: Διαδικασία encode

Με τον όρο *ENCODE* εννοούμε τον χρόνο που απαιτείται προκειμένου να δημιουργήσουν τον πακέτο που θα στείλουν στους κόμβους που απαιτείται. Λογικά στην περίπτωση όπου το αρχείο γίνεται δύο κομμάτια και στους κόμβου; αναθέτουμε δύο αρχεία οι κόμβοι έχουν ήδη όλα τα δεδομένα στην κατοχή τους οπότε και δεν απαιτείται μετάδοση δεδομένων και για αυτό η τιμή του encode για αυτή την περίπτωση είναι πρακτικά μηδενική. Από εκεί και πέρα όμως η αυξάνοντας τον αριθμό του φορτίου επιτυγχάνουμε τη βέλτιστη μέση απόδοση για r ίσο με το 7. Παρόλα αυτά για r=8 έχουμε χαμηλότερη μέγιστη τιμή για τη διαδικασία *ENCODE*.

Αριθμός κόμβων	BW(Mbps)
2	0
3	1252,15
4	2782,10
5	2296,77
6	1782,94
7	1445,93
8	1238,10
9	775,29

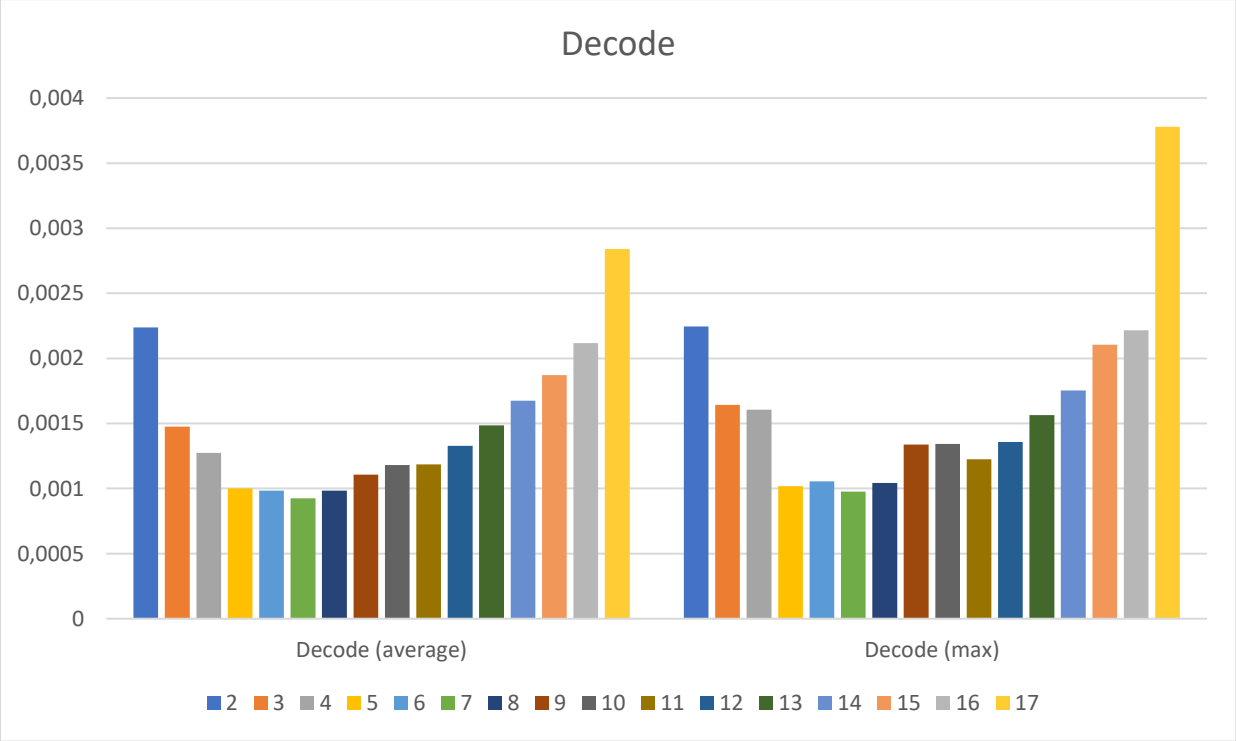
Αριθμός Κόμβων	BW(Mbps)
10	717,62
11	580,28
12	551,66
13	532,75
14	485,41
15	522,28
16	391,34
17	16,99

Πίνακας 7: <u>Χρήση BW</u>



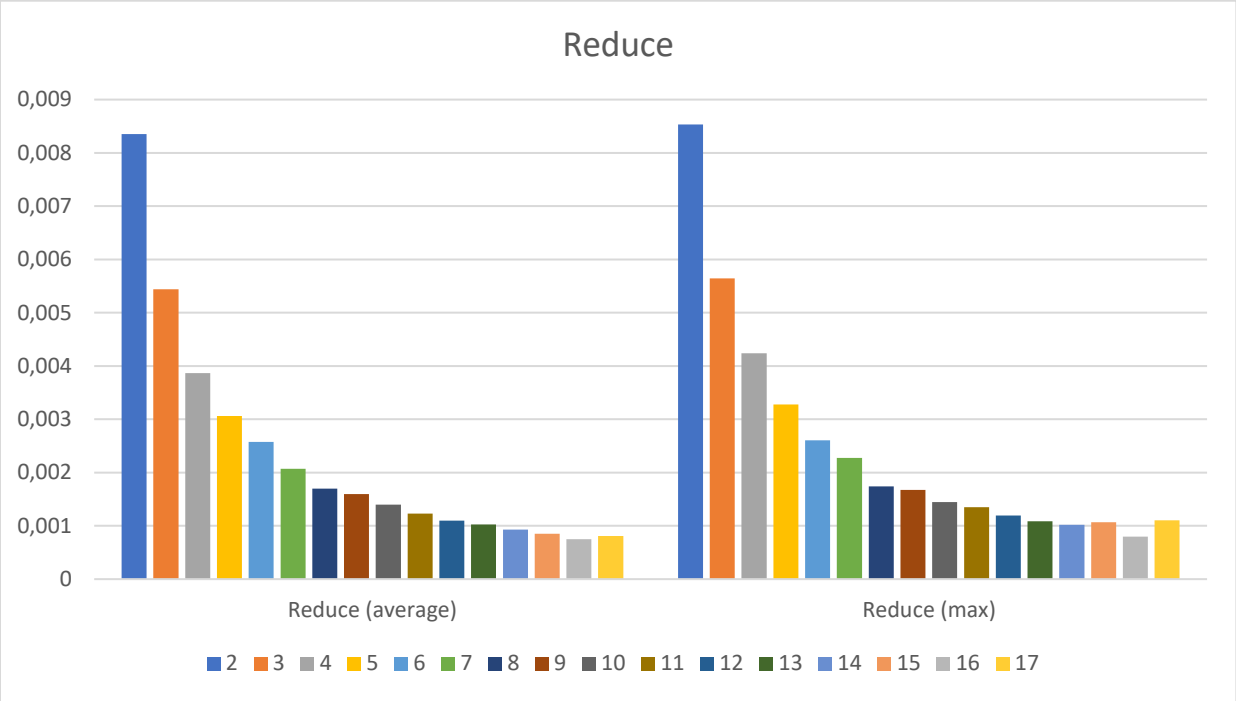
Διάγραμμα 8: <u>Απαιτούμενο BW</u>

Για το απαιτούμενο *BW* τα αποτελέσματα είναι επίσης αναμενόμενα. Για φορτίο $r = 2$ και αριθμό κόμβων ίσο με 2 είναι 0 μιας και δεν απαιτείται ανταλλαγή δεδομένων ενώ με την αύξηση των κόμβων και τον περεταίρω κατακερματισμό των αρχείων η χρήση του δικτύου μειώνεται.



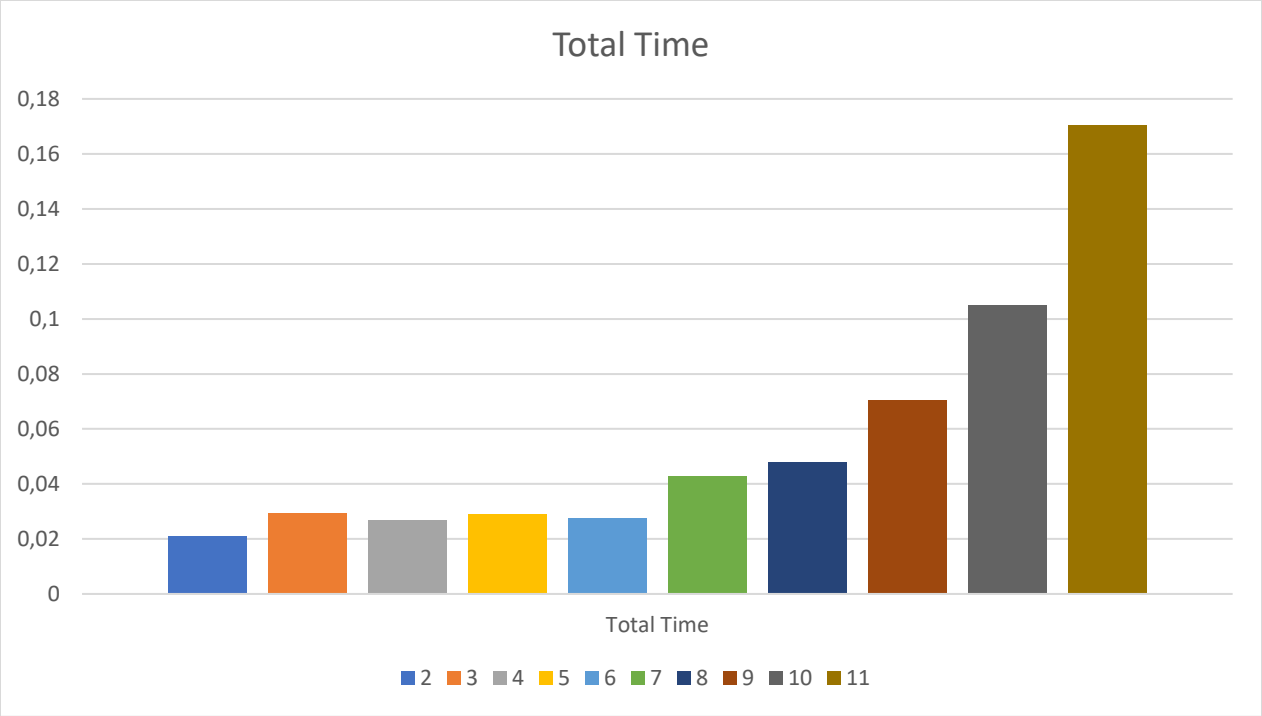
Διάγραμμα 9: <u>Διαδικασία αποκωδικοποίησης</u>

Η διαδικασία της αποκωδικοποίησης (decode) έχει να κάνει με την δημιουργία των δύο πακέτων που αφορούν τον κόμβο από το ένα πακέτο που θα λάβει. Αυξάνοντας το r ο κόμβος λαμβάνει περισσότερα πακέτα (προϊόντα της πράξης XOR) και για αυτό ο χρόνος αποκωδικοποίησης αυξάνεται. Οι ελάχιστες απαιτήσεις σε χρόνο παρατηρούνται για την περιοχή με αριθμό κόμβων και φορτίο από 6 μέχρι και 8.



Διάγραμμα 10: <u>Διαδικασία reduce</u>

Η διαδικασία *REDUCE* είναι αυτή η οποία οδηγεί και στα τελικά αποτελέσματα και είναι λογικό να μειώνεται ο χρόνος που απαιτεί, με την αύξηση του φορτίου και του αριθμού των κόμβων, μιας και τα δεδομένα προς ταξινόμηση για τον κάθε κόμβο μειώνονται.



Διάγραμμα 11: Συνολικός χρόνος

Ο συνολικός χρόνος που απαιτεί η διαδικασία παρουσιάζει ελάχιστο στην περίπτωση που δεν απαιτείται καμία αποστολή και λήψη δεδομένων (δύο κόμβοι και r=2). Το σημαντικό όμως το οποίο είναι και το ζητούμενο είναι ότι για r=6 έχουμε την ταχύτερη ολοκλήρωση της διαδικασίας διάρκειας ίση με 0,02746457 δευτερόλεπτα. Παρεμφερείς τιμές έχουμε για r ίσο με 3,4 και 5 όμως από το 8 και μετά ο συνολικός χρόνος αρχικά διπλασιάζεται, για r=11 γίνεται 7 φορές μεγαλύτερος και από εκεί και πάνω οι τιμές που παίρνει είναι πολύ μεγάλες για να έχει νόημα η απεικόνισή τους. Ενδεικτικά για 14 απαιτούνται περίπου 1,6 δευτερόλεπτα, 3,5*sec* για 15 και 8*sec* για r=16.

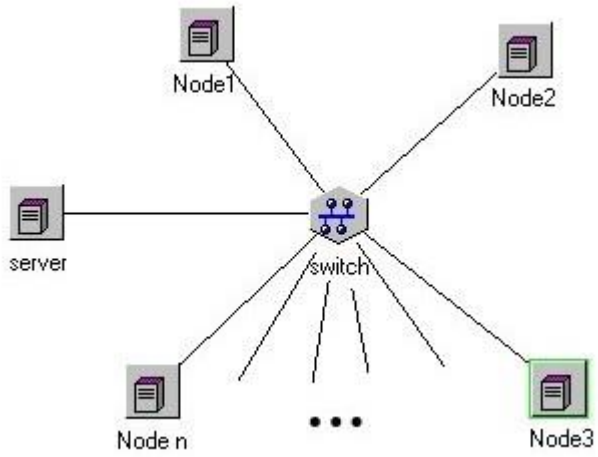
4.3 Τελικό Συμπέρασμα *Coded Computing*

Το μοντέλο που δοκιμάσαμε παρουσιάζει βέλτιστη συμπεριφορά για τις τιμές του *r* ίσες με 5, 6 και 7 επαληθεύοντας έστω και κατά προσέγγιση την αρχική υπόθεση της σχέσης ανάμεσα στο υπολογιστικό και το επικοινωνιακό φορτίο που δείξαμε στο Διάγραμμα 1 (σελ.5) (υπενθυμίζουμε ότι από το συγκεκριμένο διάγραμμα πρόκυπτε σαν βέλτιστη τιμή απόδοσης, η περίπτωση κατά την οποία το υπολογιστικό φορτίο ήταν ίσο με 6). Από εκεί και μετά η απόδοση του συστήματος κάμπτεται οδηγώντας τη διαδικασία σε μεγάλες καθυστερήσεις. Το όφελος που

προκύπτει από τη μείωση του χρησιμοποιούμενου εύρους ζώνης δεν φαίνεται να μπορεί να υπερκεράσει τις χρονικές καθυστερήσεις και η χρήση της συγκεκριμένης μεθόδου θα ήταν αποδοτική και κατά συνέπεια δικαιολογημένη μόνο σε περιπτώσεις όπου είτε το διαθέσιμο BW είναι μικρό, είτε το κόστος χρήσης του είναι υψηλό. Εδώ δίδεται μία ευκαιρία για συνέχεια στην ερευνητική προσπάθεια στην κατεύθυνση για την ανάπτυξη ενός καλύτερα σχεδιασμένου αλγόριθμου ο οποίος θα απέδιδε καλύτερα (και) για μεγαλύτερες τιμές του r.

4.4 Πείραμα Κωδικοποίησης Δικτύου και Ομομορφικής Κωδικοποίησης Δικτύου

Σύμφωνα με τις μετρήσεις που έγιναν παραπάνω χρησιμοποιώντας την τεχνική της υπολογιστικής κωδικοποίησης, κάναμε κάποιες δοκιμές στο περιβάλλον κωδικοποίησης δικτύου για να δούμε πώς μπορεί να «συγκριθεί» με το σενάριο της υπολογιστικής κωδικοποίησης. Αυτό που κάναμε είναι ότι πραγματοποιήσαμε κάποιες συνδέσεις διακομιστή-πελάτη σε ένα δίκτυο στο οποίο, όπως και στην περίπτωση του coded computing, δεν υπάρχουν λάθη στη μετάδοση ή παρεμβολές από τρίτους. Στείλαμε τα αρχεία από τον διακομιστή (server) στους πελάτες (clients). Εάν υπάρχουν πολλοί δέκτες, τότε όλοι λαμβάνουν το πλήρες αρχείο επειδή μεταδίδουμε το αρχείο. Αυτό το δοκιμάσαμε για διάφορους συνδυασμούς πελατών διακομιστή, καθώς και για δύο διαφορετικά αρχεία. Το ένα μεγέθους 0,33MB είναι το ίδιο αρχείο που χρησιμοποιήθηκε στις μετρήσεις coded terasot στην ενότητα 4.2 Πείραμα *Coded* (σελ.43) και ένα ακόμα αρχείο μεγέθους 1MB. Το πείραμα επαναλήφθηκε χρησιμοποιώντας ένα ασφαλές περιβάλλον κωδικοποίησης δικτύου όπου εφαρμόζεται μια ομομορφική κρυπτογράφηση όπως περιγράφεται στην ενότητα 3.8.



Εικόνα 8: Σχήμα εικονικής πειραματικής διάταξης

4.5 Κωδικοποίηση Δικτύου Random Linear Network Coding

4.5.1 Μέγεθος Αρχείου 0,33MByte

Αριθμός Client(s)	*Coding Time (sec)*	*Transmission Time (sec)*	*Total Time (sec)*
1	0,00008	0,33492	0,335
2	0,00008	0,33492	0,335
3	0,00008	0,33492	0,335
4	0,00008	0,33492	0,335
5	0,00008	0,33492	0,335
6	0,00008	0,33492	0,335
7	0,00008	0,33492	0,335
8	0,00008	0,33492	0,335

Πίνακας 8: RLNC coding time για αρχείο 0,33 MB

4.5.2 Μέγεθος Αρχείου 1MByte

Αριθμός Client(s)	*Coding Time (sec)*	*Transmission Time (sec)*	*Total Time (sec)*
1	0,00009	1,04991	1,05
2	0,00009	1,04991	1,05
3	0,00009	1,04991	1,05
4	0,00009	1,04991	1,05

Πίνακας 9: RLNC coding time για αρχείο 1 MB

Και για τα δύο μεγέθη αρχείων ο αλγόριθμος Random Linear Network Coding επιδεικνύει σταθερή συμπεριφορά και τελικά δεν υπάρχουν διαφορές ως προς τον αριθμό των πελατών

καθώς μεταδίδουμε το μήνυμα. Αυτό είναι λογικό άλλωστε από τη στιγμή που στο συγκεκριμένο αλγόριθμο όλοι οι κόμβοι πρέπει να πάρουν όλοι την πληροφορία.

4.6 Ομομορφική Κωδικοποίηση Δικτύου (ασφαλής-secure)

4.6.1 Μέγεθος Αρχείου 0,33MByte

Αριθμός Client(s)	*Coding Time (sec)*	*Transmission Time (sec)*	*Total Time (sec)*
1	0,00009	0,33491	0,335
2	0,00009	0,33491	0,335
3	0,00009	0,33491	0,335
4	0,00009	0,33491	0,335
5	0,00009	0,33491	0,335
6	0,00009	0,33491	0,335
7	0,00009	0,33491	0,335
8	0,00009	0,33491	0,335

Πίνακας 10: Homomorphic coding time για αρχείο 0,33 MB

4.6.2 Μέγεθος Αρχείου 1MByte

Αριθμός Client(s)	*Coding Time (sec)*	*Transmission Time (sec)*	*Total Time (sec)*
1	0,0001	1,0499	1,050
2	0,0001	1,0499	1,050
3	0,0001	1,0499	1,050
4	0,0001	1,0499	1,050

Πίνακας 11: Homomorphic coding time για αρχείο 1 MB

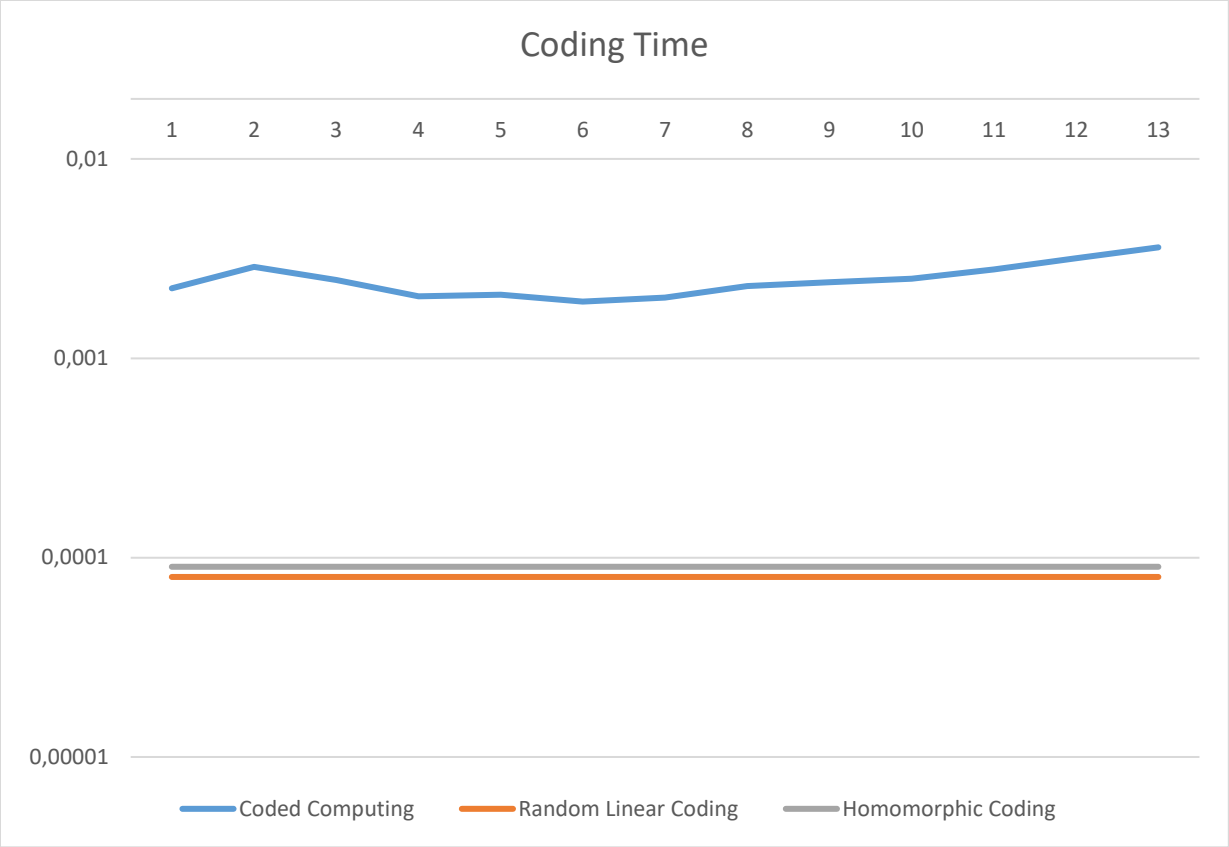
Από τις προσομοιώσεις μας φάνηκε, ότι ο χρόνος κωδικοποίησης και αποκωδικοποίησης στο περιβάλλον κωδικοποίησης δικτύου είναι αρκετά χαμηλός και σχεδόν ο ίδιος και για τα δύο μεγέθη αρχείων. Επίσης όπως και προηγουμένως στην περίπτωση του *Random Linear Network Coding* δεν υπάρχουν διαφορές με την αύξηση του αριθμού των πελατών (clients) καθώς μεταδίδουμε το μήνυμα. Το κύριο μέρος του χρόνου που απαιτείται αφορά στην καθυστέρηση μετάδοσης. Υποθέτουμε ότι η καθυστέρηση κωδικοποίησης στο κωδικοποιημένο υπολογιστικό περιβάλλον μπορεί να διαφέρει ανάλογα με το μέγεθος του αρχείου και σε αυτήν την περίπτωση, έχουμε ένα σημείο σύγκρισης.

Κεφάλαιο 5 - Αποτελέσματα

Στον παρακάτω πίνακα βλέπουμε συγκεντρωτικά τις τιμές για τις τρεις μεθόδους (Coded Computing, Random Linear Coding και Homomorphic Coding που χρησιμοποιήσαμε πειραματικά για το ίδιο αρχείο των 0,33Mbyte. Όλοι οι χρόνοι είναι σε *sec.*

	Total Coding Time (sec)		
Αριθμός Client(s)	***Coded Computing***	***Random Linear Coding***	***Homomorphic Coding***
2	0,0022405	0,00008	0,00009
3	0,00287567	0,00008	0,00009
4	0,002466	0,00008	0,00009
5	0,0020472	0,00008	0,00009
6	0,0020874	0,00008	0,00009
7	0,001924429	0,00008	0,00009
8	0,0020175	0,00008	0,00009

Πίνακας 12: Συγκεντρωτικός πίνακας coding time για αρχείο 0,33 MB



Διάγραμμα 12: <u>Γραφική απεικόνιση χρόνων</u>

Η ειδοποιός διαφορά ανάμεσα στο Coded Computing και στα Random Linear Coding, Homomorphic Coding είναι ότι στην πρώτη περίπτωση μόνο ένα μέρος της πληροφορίας ανατίθεται σε κάθε κόμβο. Αυτό από μόνο του χαρίζει στο *Coded Computing* ένα πλεονέκτημα σε σύγκριση με τις υπόλοιπες απλές (χωρίς κρυπτογράφηση), μεθόδους *Network Coding*: Από τη στιγμή που κανένας κόμβος δεν έχει ολοκληρωμένο το σύνολο των προς μετάδοση δεδομένων, η μετάδοση καθίσταται προστατευμένη από πιθανές προσπάθειες υποκλοπής. Όμως επειδή ακριβώς μόνο ένα μέρος της πληροφορίας ανατίθεται σε κάθε κόμβο, αυτό έχει σαν αποτέλεσμα ότι με την αύξηση του αριθμού (των κόμβων) ο χρόνος επεξεργασίας φυσιολογικά αυξάνεται, μιας και πλέον οι κόμβοι καλούνται να κάνουν περισσότερους υπολογισμούς, προκειμένου να ανακτήσουν το σύνολο της πληροφορίας που τους έχει ανατεθεί, όπως φαίνεται και στο Διάγραμμα 11 (σελ.55). Στην περίπτωση των δύο άλλων μεθόδων (RLNC – Homomorphic Coding) ο χρόνος επεξεργασίας στον κάθε κόμβο μένει σταθερός και ανεξάρτητος από το πλήθος τους πράγμα που επίσης επιβεβαιώνεται από τις μετρήσεις που

απεικονίζονται σχηματικά στο Διάγραμμα 12 (σελ.62). Είναι φανερό επίσης ότι ο χρόνος που απαιτείται για τον υπολογισμό στην περίπτωση του coded computing είναι πολλαπλάσιος ακόμα και για μικρό αριθμό κόμβων εξαιτίας της μεγαλύτερης πολυπλοκότητας των υπολογισμών. Όπως μάλιστα φαίνεται και στις μετρήσεις στον Πίνακας 5 (σελ.45), για μεγαλύτερο αριθμό κόμβων αυξάνεται σημαντικά. Αυτό οφείλεται στο γεγονός ότι ο κατακερματισμός του αρχικού αρχείου σε ολοένα και περισσότερα κομμάτια οδηγεί αναπόφευκτα στη χρήση ολοένα και περισσότερων υπολογιστικών πόρων. Αυτό δεν συμβαίνει στις δύο άλλες περιπτώσεις διότι όλοι οι κόμβοι λαμβάνουν όλο το αρχείο το οποίο, κωδικοποιείται στην περίπτωση του RLNC, κρυπτογραφείται στην περίπτωση της homomorphic κωδικοποίησης έχοντας σταθερό χρονικό διάστημα *coding time*.

Εξετάζοντας τις αποδόσεις των τριών μεθόδων στον τομέα της χρήσης δικτύου, στις περιπτώσεις της Γραμμικής Κωδικοποίησης Δικτύου και της Ομομορφικής Κρυπτογράφησης η χρήση του δικτύου μένει σταθερή, ανεξάρτητα από τον αριθμό των κόμβων. Στην περίπτωση του Coded Computing όμως έχουμε μείωση της χρησιμοποίησης του δικτύου με την αύξηση των κόμβων αφού η μέθοδος μπορεί, αυξάνοντας τους κόμβους, να περιορίσει το μέγεθος των δεδομένων που διασχίζουν το δίκτυο όπως περιγράψαμε στο Κεφάλαιο 1 - (σελ3) και ο κάθε κόμβος μπορεί να συνθέσει το ολικό πακέτο κάνοντας συνδυασμό των δεδομένων που έχει ήδη λάβει. Το τελικό ισοζύγιο είναι θετικό, αφού η καθυστέρηση στους κόμβους, εξαιτίας της επιπλέον απαιτούμενης επεξεργασίας που απαιτούν τα δεδομένα προκειμένου να δημιουργήσουν το αρχικό αρχείο, είναι ασήμαντη, σχετικά με το όφελος που προκύπτει από τη μείωση των καθυστερήσεων στο δίκτυο. Ένα βασικό μειονέκτημα της μεθόδου *coded computing* που αναλύσαμε σε αυτή την εργασία, είναι ότι με τη χρήση της Μαθηματικής πράξης XOR, από την άλγεβρα Boole, στην ουσία μηδενίζονται οι ανοχές ως προς την ορθότητα των δεδομένων που λαμβάνουν οι κόμβοι. Αν ένα κανάλι μετάδοσης δεν είναι ιδανικό (αξιόπιστη μετάδοση χωρίς μεγάλες καθυστερήσεις) τότε η μέθοδος παύει να είναι αποδοτική και στην περίπτωση μάλιστα που ο κόμβος λάβει ένα πακέτο με λανθασμένο έστω και 1 bit, τότε το τελικό αρχείο που θα δημιουργήσει δεν θα ανταποκρίνεται στην πραγματικότητα, καθιστώντας έτσι τη μετάδοση αναξιόπιστη. Επιπλέον όπως παρατηρήθηκε και στις μετρήσεις μας η ραγδαία αύξηση του αριθμού των κόμβων οδηγεί σε μεγάλη αύξηση του χρόνου επεξεργασίας και έτσι το τελικό χρονικό όφελος από τη μείωση της χρήσης του δικτύου, όχι μόνο μηδενίζεται αλλά παίρνει και αρνητικές τιμές, μιας και ο συνολικός χρόνος που απαιτείται για τη μετάδοση κωδικοποιημένων δεδομένων με τη μέθοδο

coded computing, ξεπερνάει τον αντίστοιχο χρόνο μετάδοσης όταν δεν χρησιμοποιείται η κωδικοποίηση.

Από την άλλη, οι άλλες δύο μέθοδοι μπορούν να υποστηρίξουν μεταδόσεις σε οποιοδήποτε μέσο, προσφέροντας είτε συμπίεση, είτε κρυπτογράφηση των προς μετάδοση δεδομένων και είναι κατάλληλες για μεταδόσεις που έχουν ευαισθησία στις χρονικές καθυστερήσεις.

Συνοψίζοντας το συμπέρασμα που βγαίνει είναι ότι η μέθοδος coded computing που εξετάσαμε εδώ είναι εξαιρετικά αποδοτική αλλά για να συμβεί αυτό απαιτούνται αξιόπιστα κανάλια μετάδοσης ανάμεσα στους κόμβους όπως επίσης και ο αριθμός των κόμβων να παραμένει σχετικά μικρός. Επιπλέον θα λέγαμε ότι η αποδοτικότερη χρήση της αφορά σε εφαρμογές τύπου cloud computing όπου η προς μετάδοση πληροφορία πρέπει να ανατεθεί εκ νέου σε νέους κόμβους όπου θα υποστεί περεταίρω επεξεργασία. Από την άλλη το network coding δεν πρέπει να θεωρηθεί σε καμία περίπτωση ξεπερασμένο μιας και ειδικά με την ανάπτυξη των 5G δικτύων ανοίγει ένα νέο πεδίο εφαρμογών, με διάφορες υλοποιήσεις, για τη συγκεκριμένη τεχνολογία προκειμένου να επιτευχθούν με βελτιωμένους τρόπους η ασφαλέστερη και ταχύτερη η μετάδοση δεδομένων μέσα από τα ασύρματα δίκτυα νέας τεχνολογίας [33].

Κεφάλαιο 6 - Τελικά Συμπεράσματα

Με δεδομένη τη δυσκολία του εγχειρήματος καταφέραμε παρόλα αυτά να υλοποιήσουμε το πείραμα που αναφέραμε στην εισαγωγή. Το ζητούμενο, πρέπει να υπενθυμίσουμε, δεν ήταν η απευθείας «αντιπαράθεση» των δύο μεθόδων σε απόλυτα νούμερα και χρόνους αλλά η σύγκριση των δύο διαφορετικών «φιλοσοφιών» κωδικοποίησης. Σε ποιες περιπτώσεις εξυπηρετεί καλύτερα η μία και σε ποιες η άλλη. Ποια είναι τα οφέλη που προκύπτουν στην κάθε περίπτωση και ποιες οι αδυναμίες. Είναι φυσικά αυτονόητο πως και οι δύο μέθοδοι μπορούν να χρησιμοποιηθούν και συνδυαστικά προκειμένου να δώσουν λύση σε περισσότερο σύνθετα προβλήματα.

6.1 Μελλοντικές Σκέψεις

Με δεδομένο ότι το αρχείο προς μετάδοση χωρίζεται σε πολλά μπλοκ τα οποία μεταδίδονται ανεξάρτητα στο δίκτυο μία σκέψη θα ήταν σε ένα περιβάλλον κωδικοποίησης δικτύου, να δημιουργήσουμε πολλές διαδρομές και προορισμούς και να προσπαθήσουμε να στείλουμε μέρος του αρχείου σε κάθε προορισμό.

Θα μπορούσαμε να προσπαθήσουμε να συνδυάσουμε τη μέθοδο της κρυπτογράφησης με το *coded computing* κάνοντας έτσι την επικοινωνία ανάμεσα στους κόμβους ασφαλή χρησιμοποιώντας την ομομορφική κρυπτογράφηση κάνοντας εκτός από ασφαλή και περισσότερο αξιόπιστη τη μετάδοση. Να σημειώσουμε εδώ ότι με την ανάπτυξη των δικτύων 5G σε βιομηχανικά και αστικά περιβάλλοντα δημιουργούνται δίκτυα με μεγάλο αριθμό κόμβων οι οποίοι καλούνται να επεξεργαστούν μεγάλο όγκο δεδομένων προερχόμενο από διάφορους αισθητήρες. Οι συγκεκριμένοι αισθητήρες μπορεί να παρατηρούν για παράδειγμα είτε την εξέλιξη της παραγωγής σε μία βιομηχανία είτε την κυκλοφορία των οχημάτων σε ένα αστικό περιβάλλον. Τα δεδομένα αυτά θα πρέπει να επεξεργαστούν κατάλληλα ώστε να δοθούν οι κατάλληλες εντολές και οι διαδικασίες διαχείρισης των συστημάτων που επιτηρούν να συνεχίσουν να εκτελούνται ομαλά. Ήδη υπάρχει πλήθος ερευνών πάνω στα συγκεκριμένα αντικείμενα το οποίο όπως δείχνουν τα πράγματα και με την πρόοδο των εξελίξεων θα αυξάνεται ακόμα περισσότερο.

Κεφάλαιο 7 - Αναφορές

[1] S. Li, M. A. Maddah-Ali, Q. Yu και S. A. Avestimehr , «A Fundamental Tradeoff between Computation and Communication in Distributed Computing,» *IEEE Transactions on Information Theory,* τόμ. 64, αρ. 1, pp. 109-128, 26 09 2017.

[2] «Polynomial interpolation,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Polynomial_interpolation. [Πρόσβαση 18 12 2019].

[3] C. Fragouli, J.-Y. Le Boudec και J. Widmer, «Network coding: an instant primer,» *ACM SIGCOMM Computer Communication Review,* τόμ. 36, αρ. 1, pp. 63-68, 01 01 2006.

[4] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi και B. Leong, «A Random Linear Network Coding Approach to Multicast,» τόμ. 52, αρ. 10, pp. 4413 - 4430, 25 09 2006.

[5] S. Li, M. A. Maddah-Ali και A. S. Avestimehr, «Coding for Distributed Fog Computing,» *IEEE Communications Magazine,* τόμ. 55, αρ. 4, pp. 34-40, 17 04 2017.

[6] S. Prakash, S. Dhakal, Y. Yona, S. Talwar, S. Avestimehr, N. Himayat και M. Akdeniz, «Coded Computing for Low-Latency Federated Learning over Wireless Edge Networks,» 2020.

[7] A. Reisizadeh, S. Prakash, R. Pedarsani και A. S. Avestimehr, «Coded Computation over Heterogeneous Clusters,» *IEEE Transactions on Information Theory,* τόμ. 65, αρ. 7, pp. 4227 - 4242, 08 03 2019.

[8] S. Li, M. Yu, S. Avestimehr, S. Kannan και P. Viswanath, «PolyShard: Coded Sharding Achieves Linearly Scaling Efficiency and Security Simultaneously,» 27 09 2019. [Ηλεκτρονικό]. Available: https://arxiv.org/abs/1809.10361. [Πρόσβαση 08 12 2019].

[9] H. Chen και Y. Wu, «Coded Computing for Master-Aided Distributed Computing Systems,» arXiv:2010.10718v1, 2020.

[10] S. Dhakal, S. Prakash, Y. Yona, S. Talwar και N. Himayat, «Coded Computing for Distributed Machine Learning in Wireless Edge Network,» σε *IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Honolulu, 2019.

[11] S. Li, S. Supittayapornpong, M. A. Maddah-Ali και S. Avestimehr, «Coded TeraSort,» σε *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Lake Buena Vista, FL, USA , 2017.

[12] O. O'Malley, «Terabyte sort on Apache Hadoop,» 2008.

[13] S. Li, . M. A. Maddah-Ali και A. S. Avestimehr, «Coded MapReduce,» σε *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015.

[14] M. Celebiler και G. Stette, «On increasing the down-link capacity of a regenerative satellite repeater in point-to-point communications,» *Proceedings of the IEEE,* τόμ. 66, αρ. 1, pp. 98-100, 1 1978.

[15] P. Zhang, C. Lin , Y. Jiang , Y. Fan και X. Shen , «A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks,» *IEEE Transactions on Parallel and Distributed Systems,* τόμ. 25, αρ. 9, pp. 2211 - 2221, 09 2014.

[16] T. Ho, R. Koetter, M. Medard, D. R. Karger and M. Effros, "The benefits of coding over routing in a randomized setting," in *The benefits of coding over routing in a randomized setting*, Yokohama, 2003.

[17] Y. Wu, P. Chou και K. Jain, «A comparison of network coding and tree packing,» σε *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings*, Chicago, 2004.

[18] P. Sanders, S. Egner και L. Tolhuizen, «Polynomial Time Algorithms for Network Information Flow,» 2003.

[19] P. A. Chou, Y. Wu και K. Jain, «Practcal Network Coding,» σε *In Proc. Allerton*, 2003.

[20] C. Fragouli, J. Widmer και J.-Y. LeBoudec, «A network coding approach to energy efficient broadcasting: from theory to practice,» σε *Infocom 2006*, 2005.

[21] R. W. YEUNG, «AVALANCHE: A NETWORK CODING ANALYSIS,» *COMMUNICATIONS IN INFORMATION AND SYSTEMS,* τόμ. 7, αρ. 4, pp. 353-358, 2007.

[22] S. Katti, D. Katabi, W. Hu, H. Rahul και M. Medard, «The importance of being opportunistic: Practical network coding for wireless environments.,» Allerton, 2005.

[23] J. Widmer, C. Fragouli και J. Y. LeBoudec, «Energy efficient broadcasting in wireless ad hoc networks,» σε *First Workshop on Network Coding*, 2005.

[24] N. Cai και R. Yeung, «Secure network coding,» σε *Proceedings IEEE International Symposium on Information Theory*, Lausanne, Switzerland, 2002.

[25] T. Ho, B. Leong, R. Koetter και M. Médard, «Byzantine Modification Detection in Multicast Networks with Random Network Coding,» 2004.

[26] S. Agrawal και D. Boneh, «Homomorphic MACs: MAC-Based Integrity for Network Coding,» σε *International Conference on Applied Cryptography and Network Security*, Paris, 2009.

[27] D. Boneh, D. Freeman, J. Katz και B. Water, «Irvine: Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography:,» σε *Signing a Linear Subspace: Signature Schemes for Network Coding*, Irvine, 2009.

[28] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter και M. Strand, «A Guide to Fully Homomorphic Encryption,» 2015. [Ηλεκτρονικό]. Available: https://www.semanticscholar.org/paper/A-Guide-to-Fully-Homomorphic-Encryption-Armknecht-Boyd/7ee670d05930c034d2224a42b37db8862a566810. [Πρόσβαση 04 02 2020].

[29] R. Canetti, J. Garay, G. Itkis και D. Micciancio, «Multicast security: A taxonomy and some efficient constructions,» σε *INFOCOM*, 1999.

[30] J. L. Carter και M. N. Wegman, *Journal of Computer and Science,* αρ. 18, pp. 143-154, 1979.

[31] «Open MPI: Open Source High Performance Computing,» [Ηλεκτρονικό]. Available: https://www.open-mpi.org/. [Πρόσβαση 7 12 2019].

[32] «GitHub - AvestimehrResearchGroup/Coded-TeraSort,» [Ηλεκτρονικό]. Available: https://github.com/AvestimehrResearchGroup/Coded-TeraSort. [Πρόσβαση 12 3 2019].

[33] V. Adat, I. Politis, C. Tselios, P. Galiotos και S. Kotsopoulos, «On blockchain enhanced secure network coding for 5G deployments,» σε *2018 IEEE Global Communications Conference (GLOBECOM), , 2018,* , Abu Dhabi, United Arab Emirates, 2018.