ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

«ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ»

ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Επιδημική Διάδοση Δεδομένων και Κωδικοποίηση Ελέγχου Σφαλμάτων»

ΖΑΪΜΙΔΗΣ ΕΥΡΙΠΙΔΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΧΑΤΖΗΕΥΘΥΜΙΑΔΗΣ ΕΥΣΤΑΘΙΟΣ

ΠΑΤΡΑ
ΣΕΠΤΕΜΒΡΙΟΣ, 2010

Διπλωματική Εργασία

Επιδημική Διάδοση Δεδομένων και Κωδικοποίηση Ελέγχου Σφαλμάτων

ΖΑΪΜΙΔΗΣ ΕΥΡΙΠΙΔΗΣ

ΣΕΠΤΕΜΒΡΙΟΣ, 2010

Τίτλος Διπλωματικής Εργασίας

Επιδημική Διάδοση Δεδομένων και Κωδικοποίηση Ελέγχου Σφαλμάτων

Ζαϊμίδης Ευριπίδης

Ονοματεπώνυμο Επιβλέποντα	Ονοματεπώνυμο Μέλους 1	Ονοματεπώνυμο Μέλους 2
Χατζηευθυμιάδης Ευστάθιος	Γλεντής Γιώργος	Κορμέντζας Γιώργος

Περίληψη:

Στην εργασία αυτή διερευνώνται οι δυνατότητες βελτιστοποίησης της εξάπλωσης της πληροφορίας σε ένα μοντέλο επιδημικής διάδοσης σε ασύρματο ad hoc δίκτυο.
Αυτό επιτυγχάνεται μέσω της χρήσης μηχανισμών προσαρμογής της πιθανότητας εκπομπής β και του μηχανισμού κωδικοποίησης ελέγχου σφάλματος. Πιο συγκεκριμένα, χρησιμοποιείται συνελικτική κωδικοποίηση ελέγχου σφάλματος η οποία εισάγει και αντίστοιχο overhead. Με βάση μετρικές, όπως το πλήθος σφαλμάτων και το πλήθος διπλοτύπων, που πηγάζουν από την ποιότητα του ασυρματικού καναλιού και την πολυδιαδρομικότητα αντίστοιχα, προσαρμόζουμε δυναμικά την πιθανότητα εκπομπής β από κάθε κόμβο και το μηχανισμό κωδικοποίησης. Ένας καταλυτικός παράγοντας στη μελέτη μας είναι και η κινητικότητα των κόμβων.
Καταλήγουμε σε ένα μηχανισμό δυναμικής προσαρμογής του β και του ρυθμού κωδικοποίησης, ο οποίος επιτυγχάνει μεγάλη αποτελεσματικότητα, αποφεύγοντας τους υπερβολικά υψηλούς ρυθμούς εκπομπών και τις κωδικοποιήσεις με υψηλούς ρυθμούς κωδικοποίησης (coding rates), συμβάλλοντας έτσι ουσιαστικά στην εξοικονόμηση ενέργειας στον κόμβο.

Λέξεις-κλειδιά: Συνελικτικοί Κώδικες, Κώδικες Ανίχνευσης Σφαλμάτων, Επιδημικοί Αλγόριθμοι, Επιδημικά Μοντέλα SI, SIR, SIS, Peer to Peer, Διαφορισμός.

Περιεχόμενο: Η μορφή του περιεχομένου της Διπλωματικής εργασίας αποτελείται από κείμενο, πίνακες, σχήματα από τη διεθνή και Ελληνική βιβλιογραφία, πρόγραμμα προσομοίωσης μέσω j-sim και από μια εφαρμογή που κάνει χρήση του επιδημικού αλγορίθμου με δυναμικό β σε γλώσσα Java.

Epidemic data dissemination and error control coding

Zaimidis Eyripidis

Supervisor	**Member 1**	**Member 2**
Efstathios Hadjiefthimiades	**Georgios Glentis**	**Georgios Kormentzas**

ABSTRACT:

In this study, the optimization of data transmission in ad hoc wireless networks using an epidemic algorithm model was investigated.

Techniques of adjusting the transmission probability β and error control coding were applied; a convolutional error control coding scheme which introduces the respective overhead was utilized. Based on the number of errors and duplicates stemming from the radio channel quality and the multipath propagation effect, the probability of transmission, β, from each node and the coding rate are dynamically adjusted. Additionally, node mobility is an important issue, also is considered in this study.

Through the utilization of β and code rate dynamic adjustment mechanisms, high spectral efficiency is finally achieved. Moreover, extremely high transmission and coding rates are avoided, contributing to the conservation of energy in the node.

Key-words: convolutional codes, error detection codes, epidemic algorithms, epidemic model SI, SIS, SIR, Peer to Peer, diversity.

Content: The thesis consists of text, tables, figures from international and Greek bibliography, a simulation program in j-sim and software in java code.

ΕΥΧΑΡΙΣΤΙΕΣ

Πρώτα από όλα θα ήθελα να ευχαριστήσω το Ελληνικό Ανοικτό Πανεπιστήμιο που μου έδωσε τη δυνατότητα να ολοκληρώσω τις Μεταπτυχιακές μου Σπουδές στο αντικείμενο που από την αρχή ενδιαφερόμουν. Στη συνέχεια θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή της εργασίας μου, κ. Ευστάθιο Χατζηευθυμιάδη, ο οποίος μου εμπιστεύτηκε την εκπόνηση της διπλωματικής εργασίας και του συγκεκριμένου θέματος. Τον ευχαριστώ παράλληλα και για την πολύ σημαντική και υποδειγματική καθοδήγησή του, τις εύστοχες παρατηρήσεις του και το χρόνο που αφιέρωσε. Η συμβολή του υπήρξε καθοριστική και χωρίς τη βοήθειά του, θα ήταν αδύνατη η ολοκλήρωση της εργασίας. Παράλειψή μου θα ήταν αν δεν ευχαριστούσα τους κυρίους Χρήστο Αναγνωστόπουλο, Μεταδιδακτορικό Ερευνητή και Φάνη Κοντό, Υποψήφιο Διδάκτορα του Τμήματος Πληροφορικής και Τηλ/νιών του Πανεπιστημίου Αθηνών, για την ποικιλότροπη βοήθεια που μου παρείχαν, ώστε να φέρω σε πέρας τη διπλωματική αυτή εργασία.

Επίσης θα ήθελα να ευχαριστήσω τη σύζυγο μου Αθανασία για τη στήριξη που μου προσέφερε στα τέσσερα χρόνια των μεταπτυχιακών μου σπουδών καθώς και τα τρία μου παιδιά, Αγγελική, Χριστίνα και Αλέξανδρο, για την κατανόηση που έδειξαν ως προς τον περιορισμένο χρόνο που αφιέρωσα σε αυτά, λόγω της πίεσης των σπουδών μου.

Αφιερώνεται στην Οικογένεια μου καθώς και στα αγαπημένα μου πρόσωπα και φίλους που έφυγαν νωρίς από την ζωή.

Ζαϊμίδης Ευριπίδης

Περιεχόμενα

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

Εισαγωγή

Είναι αλήθεια ότι τα τελευταία χρόνια, το ενδιαφέρον όλου του τηλεπικοινωνιακού κόσμου είναι εστιασμένο γύρω από εξελιγμένους αλγορίθμους και τεχνικές ώστε να επιτευχθούν υψηλοί ρυθμοί μετάδοσης πληροφοριών σε ασύρματα δίκτυα. Στην παρούσα Διπλωματική στο 1ο Κεφάλαιο γίνεται μια εισαγωγή στους Κώδικες, συγκεκριμένα εστιάζουμε το ενδιαφέρον μας στους Κώδικες Ανίχνευσης και Διόρθωσης Λαθών. Στο 2ο Κεφάλαιο μελετάται μια κατηγορία εξ αυτών, οι Συνελικτικοί Κώδικες οι οποίοι έχουν μεγάλη εφαρμογή στην ασύρματη μετάδοση. Στο 3ο Κεφάλαιο γίνεται αναφορά σε διάφορα Επιδημικά Μοντέλα και τους αντίστοιχους Επιδημικούς Αλγορίθμους που απορρέουν από αυτά, παράλληλα περιγράφεται ο επιδημικός τρόπος μετάδοσης, καθότι τα διάφορα μηνύματα θα μπορούν να βρουν την πορεία προς τον στόχο από διαφορετικές διαδρομές, μιας και εφαρμόζεται η λογική της επιδημικής διάδοσης. Στο 4ο Κεφάλαιο περιγράφεται και αναλύεται το Μοντέλο προσομοίωσης που υλοποιήθηκε προκειμένου να βελτιστοποιήσει την εξάπλωση της πληροφορίας σε ένα μοντέλο επιδημικής διάδοσης σε ασύρματο ad hoc, παράλληλα περιγράφονται και οι διάφοροι μηχανισμοί που υλοποιήθηκαν προκειμένου να επιτευχθεί αυτό. Στο 5ο Κεφάλαιο παρουσιάζονται τα διάφορα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν στα πλαίσια της Διπλωματικής, προκειμένου να υλοποιηθεί η προσομοίωση. Ο κύριος στόχος της εργασίας επιτυγχάνεται στο 6ο Κεφάλαιο, όπου παρουσιάζονται τα αποτελέσματα καθώς και τα συμπεράσματα της παρούσας διπλωματικής εργασίας.

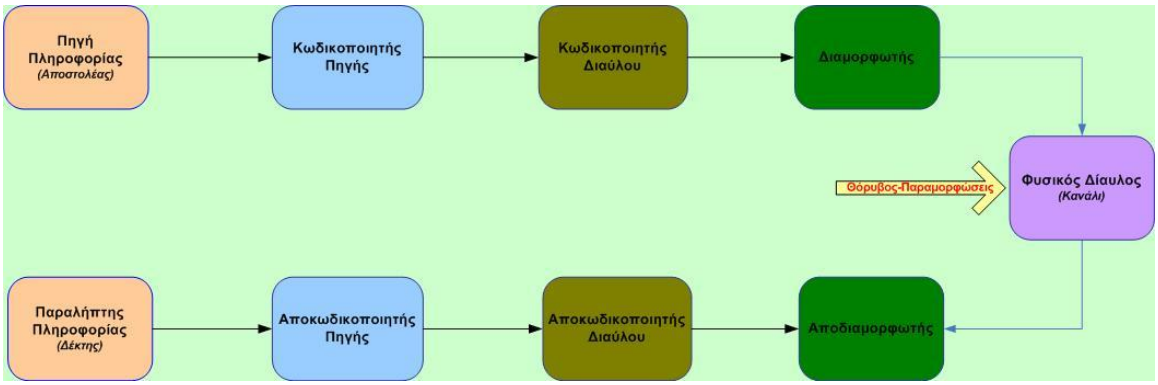
Στα Παραρτήματα γίνονται αναφορές απαραίτητες για την εργασία. Στο Παράρτημα Α αναλύονται τα Μοντέλα Κινητικότητας και περιγράφεται αυτό που θα χρησιμοποιηθεί, επίσης ενσωματώθηκε μέρος του script αρχείου n100.tcl. Τέλος στο Παράρτημα Β παρέχεται μέρος του κώδικα που χρησιμοποιήθηκε για την προσομοίωση.

1. Εισαγωγή στους κώδικες

Το κεφάλαιο αυτό παρέχει τις απαραίτητες έννοιες για την κατανόηση των επόμενων κεφαλαίων. Παραθέτουμε τις βασικές αρχές που διέπουν ένα τηλεπικοινωνιακό σύστημα ενώ παράλληλα γίνεται μια Ιστορική Αναδρομή στους Κώδικες και περιγράφουμε τα μοντέλα που έχουν σχέση με τον Έλεγχο και τη Διόρθωση Λαθών.

1.1 Αναλυτικό μοντέλο Τηλεπικοινωνιακού Συστήματος

Ο πρωταρχικός στόχος ενός Τηλεπικοινωνιακού Συστήματος είναι η αξιόπιστη μετάδοση μηνυμάτων από την πηγή στον προορισμό, ώστε να ανταπεξέρχεται στις αλλοιώσεις που τυχόν εισάγει ο θόρυβος κατά τη διάρκεια της επικοινωνίας. Το Σχήμα 1-1 περιγράφει τις θεμελιώδεις βασικές δομές, που συγκροτούν ένα ψηφιακό τηλεπικοινωνιακό σύστημα [6].



Σχήμα 1.1 Αναλυτικό Μοντέλο Τηλεπικοινωνιακό Τηλεπικοινωνιακού Συστήματος

Το παραπάνω μαθηματικό μοντέλο διακριτού-χρόνου περιλαμβάνει τα ακόλουθα υποσυστήματα τα οποία και θα αναλύσουμε παρακάτω:

1) Η ***Πηγή Πληροφορίας*** (Information source). Η πηγή πληροφορίας παράγει μηνύματα τα οποία φέρουν την πληροφορία που πρόκειται να μεταδοθεί. Τα

εξερχόμενα μηνύματα μπορεί να είναι είτε αναλογικά σήματα (π.χ. στην περίπτωση της μετάδοσης αναλογικού σήματος εικόνας), είτε διακριτού-χρόνου ακολουθίες ψηφιακών συμβόλων από κάποιο αλφάβητο. Πιο συχνά η πηγή πληροφορίας παρέχει δυαδικά σύμβολα μετάδοσης.

2) Ο ***Κωδικοποιητής Πηγής*** (source encoder). Ο κωδικοποιητής πηγής μετατρέπει όλα τα μηνύματα σε διακριτού-χρόνου ακολουθίες ψηφιακών συμβόλων, των οποίων ο μέσος όρος του μήκους των ακολουθιών ανά πρότυπο της πηγής, δεν μπορεί να είναι μικρότερος από την εντροπία της πηγής. Στην πραγματικότητα, όλες οι πηγές παράγουν μηνύματα που περιέχουν κάποιου είδους πλεονασμό. Ο στόχος του κωδικοποιητή πηγής είναι να αναπαραστήσει τα μηνύματα της πηγής, από ένα ελάχιστο αριθμό ψηφιακών συμβόλων, δηλαδή ελάχιστο πλεονασμό, με στόχο να αφαιρέσει τελείως τον πλεονασμό και να οδηγήσει σε μια αποτελεσματική περιγραφή της πηγής. Η διαδικασία αυτή αναφέρεται και ως συμπίεση δεδομένων (data compression). Η ακολουθία που προκύπτει μετά τη διαδικασία της κωδικοποίησης της πηγής, είναι στατιστικώς ανεξάρτητη και αποτελείται από ισοδύναμα σύμβολα, τα οποία αν και δεν περιέχουν κάποιου είδους πλεονασμό, επιτρέπουν την αναδημιουργία της πηγής. Αν ο κωδικοποιητής πηγής παράγει $\mathbf{r_b}$ δυαδικά ψηφία ανά δευτερόλεπτο (bps), το μέγεθος $\mathbf{r_b}$ καλείται ρυθμός δεδομένων (Data Rate).

3) Ο ***Κωδικοποιητής Διαύλου*** (Channel Encoder). Η μη ύπαρξη ιδανικού καναλιού προκαλεί αλλοίωση των μηνυμάτων κατά τη μετάδοση, με αποτέλεσμα ο δέκτης να λαμβάνει τα μηνύματα με μικρή ή μεγάλη απόκλιση από τα αρχικά. Για να μειωθούν όσο γίνεται περισσότερο τα λάθη που οφείλονται στη μετάδοση, μέσα από ένα μη ιδανικό κανάλι, γίνεται προσθήκη πλεονάζουσας πληροφορίας σε κάθε μήνυμα. Η ολοκληρωτική απαλοιφή του πλεονασμού καθιστά τα bit, που έχουν προκύψει μετά την παραπάνω διαδικασία, εξαιρετικά ευαίσθητα σε λάθη (π.χ. σε μια εικόνα που έχει υποστεί υψηλή συμπίεση ένα σφάλμα σε ένα bit μπορεί να οδηγήσει στον πλήρη συσκοτισμό της, εφόσον το λανθασμένο bit χρησιμοποιείται για την ανασυγκρότηση της πηγής). Επομένως ο πρωταρχικός ρόλος του κωδικοποιητή διαύλου είναι η μεγιστοποίηση της αξιοπιστίας της μετάδοσης λαμβάνοντας υπόψη τους περιορισμούς της ισχύος του σήματος μετάδοσης, το εύρος ζώνης του συστήματος και την πολυπλοκότητα του

κυκλώματος. Ο κωδικοποιητής διαύλου επανεισάγει κάποιον πλεονασμό στην ακολουθία των δεδομένων, που παρέχει ο κωδικοποιητής πηγής. Ο πλεονασμός αυτός περιέχει μια δομή η οποία επιτρέπει και διευκολύνει την απευθείας ανίχνευση και πιθανόν τη διόρθωση σφαλμάτων, εφόσον η παραμόρφωση δεν είναι τόσο μεγάλη. Αυτό οδηγεί σε χαμηλότερο ρυθμό μετάδοσης δεδομένων και στη χρήση μεγαλύτερου εύρους ζώνης, σε σχέση με τη μετάδοση χωρίς κωδικοποίηση. Ο κωδικοποιητής της πηγής είναι μια διάταξη διακριτής εισόδου / διακριτής εξόδου, που ορίζει μια:μία-προς-μία απεικόνιση των ακολουθιών της πηγής {ui} σε κωδικά σύμβολα {cl}. Συγκεκριμένα, αναθέτει σε κάθε πλαίσιο (frame) k ψηφίων ένα πλαίσιο μεγαλυτέρου μήκους αποτελούμενο από n ψηφία και καλείται κωδική λέξη (codeword). Ένας καλός κώδικας ελέγχου λαθών παράγει διαφορετικές μεταξύ τους κωδικές λέξεις. Αυτό καθιστά την επικοινωνία πιο ανεκτική σε λάθη που εισέρχονται από το κανάλι. Ο κάθε κώδικας χαρακτηρίζεται από το λόγο R=k/n<1, ο οποίος καλείται ρυθμός κώδικα (code rate). Ο ρυθμός εξόδου του κωδικοποιητή διαύλου είναι $\boldsymbol{r_c}=\boldsymbol{r_b}/R$ *bps*.

4) Ο ***Διαμορφωτή***ς (modulator). Σε αντίθεση με τη διακριτή φύση της εξόδου του κωδικοποιητή διαύλου, οι δίαυλοι μετάδοσης είναι αναλογικής φύσεως και αποδέχονται μόνο αναλογικές κυματομορφές στην είσοδό τους. Επομένως, ο διαμορφωτής προσαρμόζει τα ψηφιακά δεδομένα, στα ιδιαίτερα χαρακτηριστικά του φυσικού διαύλου, αναπαριστώντας τα σύμβολα cl με κατάλληλες συναρτήσεις συνεχούς χρόνου, καλούμενες κυματομορφές διαύλου. Η αναλογική κυματομορφή s(t) μεταδίδεται πάνω από το φυσικό δίαυλο επικοινωνίας, π.χ. ένα κανάλι ραδιοσυχνοτήτων, όπου βρίσκεται εκτεθειμένη σε διάφορες παραμορφώσεις τυχαίας φύσεως, όπως είναι η ηλεκτρομαγνητική παρεμβολή. Ένας Μιαδικός διαμορφωτής αντιστοιχίζει μια δομή l δυαδικών δεδομένων σε μία από τις M πιθανές κυματομορφές, όπου M=2l. Η χρονική διάρκεια της κυματομορφής στην έξοδο του αποδιαμορφωτή είναι T δευτερόλεπτα, και αναφέρεται ως διάστημα σήμανσης (signaling interval), ενώ ο ρυθμός συμβόλου (symbol rate) είναι ίσος με rs=1/T. Το ελάχιστο εύρος ζώνης που απαιτείται για τη μετάδοση του σήματος είναι ίσο με rs Hz, όπου το rs εκφράζεται ως rs= rb/Rl. Η διαμόρφωση μπορεί να πραγματοποιηθεί μεταβάλλοντας το πλάτος, τη φάση ή τη συχνότητα μιας ημιτονοειδούς κυματομορφής, που καλείται φορέας (carrier).

5) Ο ***δίαυλος επικοινωνίας*** (channel). Στα πλαίσια αυτής της διπλωματικής εργασίας επικεντρωνόμαστε στο θεμελιώδες μοντέλο του διαύλου Προσθετικού Λευκού Γκαουσσιανού Θορύβου (Additive White Gaussian Noise, AWGN). Η μεταδιδόμενη πληροφορία μεταβάλλεται με την πρόσθεση μιας τυχαίας μεταβλητής θορύβου n μηδενικού μέσου και Γκαουσσιανής κατανομής. Παραδείγματα διαύλων επικοινωνίας αποτελούν οι ενσύρματες γραμμές, οι μικροκυματικές ραδιοζεύξεις ελευθέρου χώρου, οι δορυφορικές ζεύξεις, οι οπτικές ίνες και τα μέσα μαγνητικής εγγραφής. Πολύ συχνά με τον όρο δίαυλος αναφέρεται το εύρος της συχνότητας, που κατανέμεται σε μια συγκεκριμένη υπηρεσία, όπως η τηλεόραση ή τηλεφωνικός δίαυλος. Δύο βασικοί παράγοντες που περιορίζουν την απόδοση των πραγματικών διαύλων είναι ο θερμικός θόρυβος και το πεπερασμένο εύρος ζώνης. Επιπρόσθετα, οι δίαυλοι κινητής ασύρματης εκπομπής υποφέρουν από πολυοδική μετάδοση, οι οπτικές ίνες από διασπορά του σήματος και τα μέσα μαγνητικής εγγραφής εκτίθνέται σε σκόνη και φυσική καταστροφή.

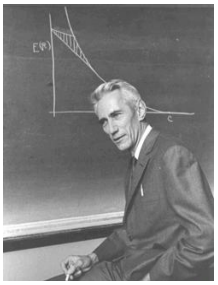
6) Ο ***Αποδιαμορφωτής*** (demodulator). Η παραμορφωμένη από το θόρυβο κυματομορφή r(t) εισάγεται στον αποδιαμορφωτή στην πλευρά του λήπτη. Ο αποδιαμορφωτής μετατρέπει τη ληφθείσα αναλογική κυματομορφή, σε μια ακολουθία από εκτιμήσεις {yi}42 των μεταδιδομένων κωδικών συμβόλων {cl}. Οι εκτιμήσεις αυτές είναι ακόμα αναλογικής φύσεως.

7) Ο ***Αποκωδικοποιητής Διαύλου*** (channel decoder). Στόχος του αποτελεί η ελαχιστοποίηση των παραμορφώσεων που εισάγει ο θόρυβος του διαύλου. Ο αποκωδικοποιητής διαύλου παράγει εκτιμήσεις των αποσταλμένων δεδομένων ui, βασιζόμενος στον πλεονασμό και στη δομή των bit ισοτιμίας, που παρέχει ο κωδικοποιητής διαύλου, καθώς επίσης λαμβάνει υπόψη του τα χαρακτηριστικά του διαύλου.

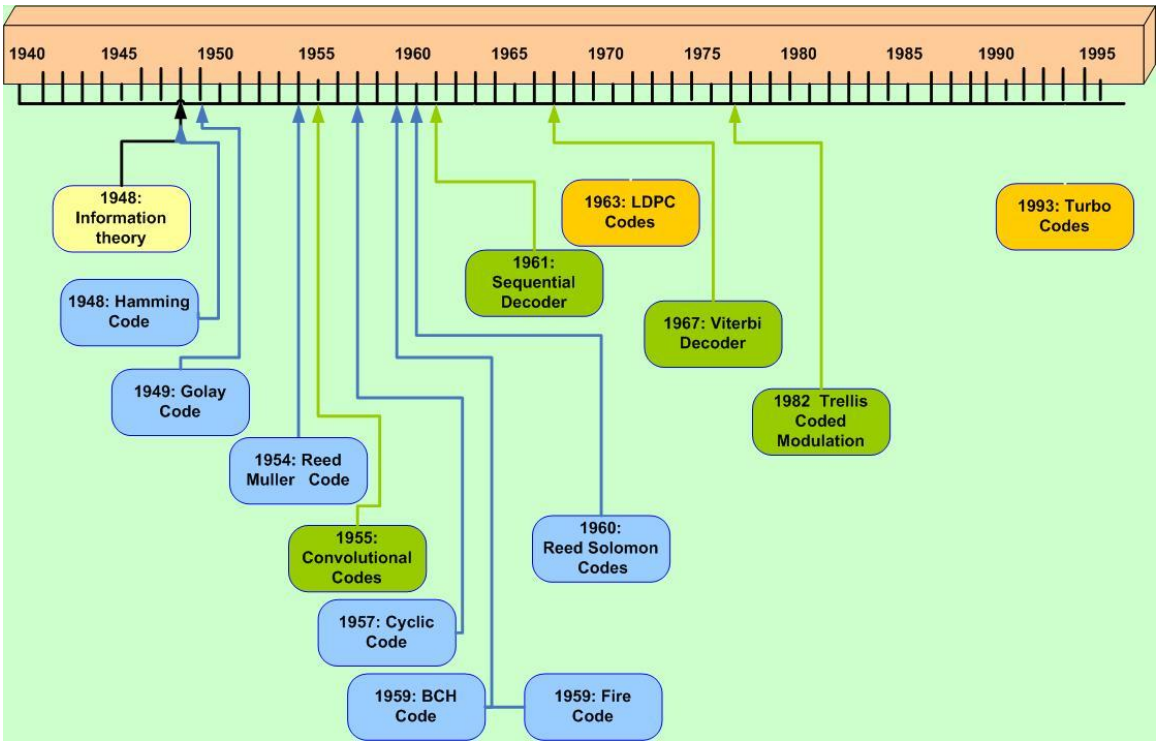
8) Ο ***Αποκωδικοποιητής Πηγής*** (source decoder). Ο αποκωδικοποιητής πηγής, βασιζόμενος στον κανόνα της κωδικοποίησης που τελεί ο κωδικοποιητής πηγής, παρέχει μια εκτίμηση της εξόδου της πηγής και παραδίδει τα δεδομένα στο δέκτη.

9) Ο ***Παραλήπτης Πληροφορίας*** (information sink). Αποτελεί τον τελικό δέκτη του απεσταλμένου μηνύματος.

1.2 Ιστορική Αναδρομή στους Κώδικες



Η δημοσίευση της εργασίας του C. E. Shannon το 1948 ο οποίος θεωρείται «πατέρας» της Θεωρίας της Πληροφορίας, στην οποία ανέπτυξε τη μαθηματική θεμελίωση των επικοινωνιών, αποτέλεσε την έναρξη μιας μεγάλης προσπάθειας εκατοντάδων επιστημόνων ανά τον κόσμο για την επίτευξη ρυθμών αξιόπιστης μετάδοσης, κοντά στη χωρητικότητα του καναλιού. Παραθέτουμε κατά χρονολογική σειρά τα επιτεύγματα από θεωρητικής πλευράς, στο πεδίο της θεωρίας πληροφοριών στο Σχήμα 1.2.



Σχήμα 1.2 Ιστορική Αναδρομή στους Κώδικες

Ακολούθησε ο Ελβετός μαθηματικός Marcel J. E. Golay, με την εφεύρεση του ***Κώδικα Golay*** το 1949, καθώς και του εκτεταμένου κώδικα Golay, οι οποίοι κώδικες έχουν γίνει αντικείμενο μελέτης για πολλούς θεωρητικούς της κωδικοποίησης και μαθηματικούς. Στο βιβλίο των Mac Williams & Sloane, υπάρχει μια εκτενής κάλυψη τόσο του κώδικα Golay όσο και του εκτεταμένου κώδικα Golay, όπου ένα ολόκληρο κεφάλαιο αφιερώνεται στην αλγεβρική τους δομή. Πέραν όμως από την αξιοσημείωτη αλγεβρική

τους δομή, οι κώδικες αυτοί έχουν χρησιμοποιηθεί σε πολλά επικοινωνιακά συστήματα. Χαρακτηριστικό παράδειγμα αποτελεί η χρήση του εκτεταμένου κώδικα Golay στην αποστολή Voyager. Το πρώτο διαστημικό εξερευνητικό όχημα της σειράς ήταν το Voyager 2, το οποίο εκτοξεύτηκε από το Διαστημικό Κέντρο την 20/7/1979, ενώ το Voyager 1, ακολουθώντας μια μικρότερη τροχιά, εκτοξεύτηκε την 5/9/1977. Τα δυο spaceprode σχεδιάστηκαν κυρίως για να πραγματοποιήσουν επιστημονικές παρατηρήσεις της ατμόσφαιρας, της μαγνητόσφαιρας, των δακτυλίων και των δορυφόρων των τεσσάρων «γιγάντων» του ηλιακού συστήματος, που είναι ο Δίας, ο Κρόνος, ο Ουρανός και ο Ποσειδώνας. Στα πλαίσια αυτής της αποστολής, ο εκτεταμένος κώδικας Golay χρησιμοποιήθηκε με μεγάλη επιτυχία, για την αποστολή φωτογραφιών του Δία και του Κρόνου, κατά τα έτη 1979, 1980 και 1981 κυρίως.

Στους θεμελιωτές των ψηφιακών συστημάτων συγκαταλέγεται και ο ***Hamming***, που το 1950 δημοσίευσε τις εργασίες του, πάνω στην ανίχνευση και διόρθωση σφαλμάτων. Σημαντική συνεισφορά υπήρξε και εκείνη του Kotelnikov το 1947, που ανέλυσε διάφορα ψηφιακά συστήματα επικοινωνιών, με βάση τη γεωμετρική προσέγγιση, καθώς και η επέκταση της εργασίας του από τους Wozencraft και Jacobs, το 1965. Ουσιαστικά ο Hamming ανακάλυψε την πρώτη τάξη γραμμικών μπλοκ κωδίκων για διόρθωση λαθών, ανακαλύφθηκαν το 1950, δύο χρόνια αφότου ο Shannon εξέδωσε τη δημοσίευση-ορόσημο, που επιβεβαίωνε ότι με κατάλληλη κωδικοποίηση της πληροφορίας, τα λάθη που προκαλούνται από ένα κανάλι με θόρυβο ή ένα μέσο αποθήκευσης, μπορούν να μειωθούν σε κάθε επιθυμητό επίπεδο, χωρίς να θυσιαστεί ο ρυθμός μετάδοσης ή αποθήκευσης της πληροφορίας. Οι κώδικες Hamming έχουν την ελάχιστη απόσταση των 3 bits και συνεπώς, μπορούν να διορθώνουν οποιοδήποτε λάθος κατά μήκος του μπλοκ κώδικα. Ο συντελεστής βάρους των κωδίκων Hamming είναι γνωστός. Είναι τέλειοι κώδικες και μπορούν να αποκωδικοποιηθούν εύκολα, απλά ελέγχοντας ένα ταμπλό αντιστοίχισης. Καλοί κώδικες με μια ελάχιστη απόσταση του 4 για διόρθωση ενός λάθους, μπορούν να κατασκευαστούν μειώνοντας κατάλληλα το μήκος των κωδίκων Hamming. Οι κώδικες Hamming και οι μειωμένου μήκους εκδοχές τους, χρησιμοποιήθηκαν ευρέως για έλεγχο λαθών στις ψηφιακές επικοινωνίες και στα συστήματα αποθήκευσης δεδομένων, μιας και διαθέτουν υψηλό ρυθμό μετάδοσης και απλή διαδικασία αποκωδικοποίησής τους.

Η δεύτερη τάξη γραμμικών μπλοκ κωδίκων, που κατασκευάστηκε στις αρχές της διόρθωσης και του εντοπισμού των λαθών, ήταν οι ***Reed-Muller κώδικες***. Οι Reed-Muller κώδικες, ήταν οι πρώτοι που ανακαλύφθηκαν από τον David E.Muller το 1954, για σχεδίαση διακοπτικών κυκλωμάτων και διόρθωση λαθών. Ο Irwin S.Reed επίσης, ήταν αυτός που στα 1954 αναμόρφωσε τους κώδικες αυτούς, για εντοπισμό και διόρθωση λαθών στις επικοινωνίες και τα συστήματα αποθήκευσης και επινόησε τον πρώτο αλγόριθμο αποκωδικοποίησης, για αυτούς τους κώδικες. Οι κώδικες Reed-Muller αποτελούν μια μεγάλη τάξη κωδίκων για διόρθωση πολλαπλών τυχαίων λαθών. Οι κώδικες αυτοί είναι απλοί στην κατασκευή και πλούσιοι σε δομικές ιδιότητες. Μπορούν να αποκωδικοποιηθούν με πολλούς τρόπους, χρησιμοποιώντας είτε soft-decision είτε hard-decision αλγορίθμους αποκωδικοποίησης. Ο αλγόριθμος αποκωδικοποίησης του Reed, για τους κώδικες Reed-Muller, παρατηρήθηκε ότι συνδυάζει χαμηλή πολυπλοκότητα με αποτελεσματική διόρθωση λαθών.

Στη συνέχεια εισάχθηκαν οι ***Συνελικτικοί κώδικες,*** από τον Elias το 1955, ως εναλλακτική επιλογή έναντι των μπλοκ κωδίκων. Λίγο αργότερα, οι Wozencraft και Reifen πρότειναν την ακολουθιακή αποκωδικοποίηση, ως αποδοτική μέθοδο αποκωδικοποίησης Συνελικτικών κωδίκων, με μεγάλα μήκη εξαναγκασμού και σύντομα έκαναν την εμφάνισή τους πειραματικές μελέτες. Το 1963 ο Massey πρότεινε μια λιγότερο αποδοτική, αλλά ευκολότερα υλοποιήσιμη μέθοδο αποκωδικοποίησης, την καλούμενη αποκωδικοποίηση κατωφλίου (threshold decoding). Αυτή η εξέλιξη αποτέλεσε αφορμή για μια σειρά από πρακτικές εφαρμογές των Συνελικτικών κωδίκων, στην ψηφιακή μετάδοση σε τηλεφωνικά, δορυφορικά και ασύρματα κανάλια. Αργότερα, το 1967, ο Viterbi πρότεινε έναν αλγόριθμο αποκωδικοποίησης μέγιστης πιθανοφάνειας (maximum likelihood, ML), σχετικά εύκολα υλοποιήσιμο, για αποκωδικοποιητή soft-απόφασης Συνελικτικών κωδίκων, με μικρά μήκη εξαναγκασμού. Ο αλγόριθμος Viterbi, παράλληλα με εκδοχές soft-απόφασης της ακολουθιακής αποκωδικοποίησης, οδήγησαν στην εφαρμογή των Συνελικτικών κωδίκων σε διαστημικά και δορυφορικά συστήματα επικοινωνιών, τη δεκαετία του ’70. Το 1974 οι Bahl, Cocke, Jelinek και Raviv (BCJR) εισήγαγαν έναν αλγόριθμο αποκωδικοποίησης, μέγιστης εκ των υστέρων πιθανότητας (maximum a posteriori probability, MAP), για συνελικτικούς κώδικες με άνισες εκ των προτέρων (a priori) πιθανότητες για τα bits πληροφορίας. Ο αλγόριθμος BCJR, έχει βρει εφαρμογή πρόσφατα σε σχήματα αποκωδικοποίησης Soft-απόφασης, όπου οι εκ των

προτέρων πιθανότητες των bits πληροφορίας μεταβάλλονται από υλοποίηση σε υλοποίηση.

Οι ***κώδικες BCH,*** έχουν ονομαστεί από τα αρχικά των Bose, Chadhuri, Hocquenghem, οι οποίοι τους εφηύραν. Πρόκειται για κυκλικούς κώδικες ναι παρέχουν στο σχεδιαστή ευρεία γκάμα επιλογών για το μέγεθος της λέξης κωδικοποίησης, τον ρυθμό του κώδικα και τη διορθωτική ικανότητα. Οι BCH κώδικες έχουν την καλύτερη δυνατή επίδοση, από όλες τις υπόλοιπες κλάσεις συμπαγών κωδίκων του ίδιου μεγέθους λέξης και ρυθμού κώδικα, όταν το μήκος λέξης τους είναι της τάξης των 100 ψηφίων.

Οι ***Κώδικες Fire,*** έχουν πάρει την ονομασία τους από τον εμπνευστή τους P. Fire, εμφανίστηκαν το 1959 και ανήκουν στην κατηγορία των δυαδικών κυκλικών κωδίκων, που παρουσιάζουν εκπληκτική ικανότητα ως προς τη διόρθωση μπλοκ σφαλμάτων.

Οι ***κώδικες Reed – Solomon,*** αποτελούν ειδική υποκατηγορία των κωδίκων BCH, που εξετάστηκαν στην προηγούμενη παράγραφο. Ωστόσο, είναι μη δυαδικοί, κυκλικοί κώδικες, δηλαδή το αλφάβητό τους αποτελείται από σύμβολα μήκους μεγαλύτερου του ενός ψηφίου. Χαρακτηριστικό παράδειγμα και ορόσημο, στην πορεία της κωδικοποίησης καναλιού, αποτελεί η δημιουργία μιας οικογένειας κωδίκων, από τον τότε μεταπτυχιακό φοιτητή στο MIT P. G. Gallager, ο οποίος στις αρχές της δεκαετίας του 60 πρότεινε τους ***κώδικες Low Density Parity Check*** (***LDPC***) και μια μορφή επαναληπτικής αποκωδικοποίησης. Επειδή όμως τότε δεν ήταν δυνατόν να υλοποιηθούν σε υλικό οι αλγόριθμοι αυτοί, εξαιτίας της αυξημένης τους πολυπλοκότητας, ξεχάστηκαν και έμειναν στο περιθώριο για παραπάνω από 35 χρόνια.

Ο ***αλγόριθμος Viterbi,*** αναπτύχθηκε από τον Andrew J. Viterbi, σε μια πρωτοποριακή για την εποχή εργασία που δημοσιεύτηκε τον Απρίλιο του 1967 [8]. Από τότε η εργασία του έχει συνεχιστεί και έχει επεκταθεί, από άλλους ερευνητές. Ο αλγόριθμος Viterbi, παρόλο που κάνει αποκωδικοποίηση Μέγιστης Πιθανοφάνειας, μειώνει το υπολογιστικό φορτίο, κάνοντας χρήση της ειδικής δομής του διαγράμματος Trellis. Το πλεονέκτημα της αποκωδικοποίησης με τη χρήση του αλγορίθμου Viterbi, συγκρινόμενη με την αποκωδικοποίηση στην οποία ελέγχουμε όλες τις περιπτώσεις, είναι ότι η πολυπλοκότητα του αποκωδικοποιητή δεν είναι συνάρτηση του αριθμού των συμβόλων της μεταδιδόμενης ακολουθίας.

Το 1993 οι ερευνητές Berrou, Glavieux και Thitimajshima πρότειναν μία νέα δομή κωδίκων, τους οποίους ονόμασαν ***Turbo Codes*** [9]. Οι κώδικες αυτοί εμπεριέχουν

εύκολη άλγεβρα, χρησιμοποιούν επαναληπτικούς και κατανεμημένους αλγόριθμους, εισάγουν την έννοια της τυχαιότητας στη διαδικασία της αποκωδικοποίησης και παρουσιάζουν εξαίρετες επιδόσεις που φτάνουν πολύ κοντά στο όριο του Shannon. Προσπαθώντας να εξηγήσουν την υπερβολικά καλή απόδοση των Turbo Codes, οι ερευνητές παρατήρησαν ότι υπήρχαν κοινά χαρακτηριστικά με τους LDPC κώδικες. Η ομοιότητα αυτή επανέφερε το ενδιαφέρον των ερευνητών για τους LDPC κώδικες και μάλιστα παρατηρήθηκε μια θεαματική αύξηση τόσο σε θεωρητικό επίπεδο, με την ανάπτυξη νέων αλγορίθμων ή τροποποιήσεων παλαιοτέρων, όσο και σε επίπεδο υλοποίησης, με τη ραγδαία ανάπτυξη της τεχνολογίας ολοκληρωμένων κυκλωμάτων πολύ μεγάλης κλίμακας (VLSI). Μάλιστα, τα 24 τελευταία χρόνια παρατηρούμε ότι οι LDPC κώδικες έχουν ενσωματωθεί σε πολλά τηλεπικοινωνιακά standards, όπως DVB-S2 [10], WiMax(802.16e) [11], 1000GBase-T Ethernet [12] κ.ά. κάτι το οποίο υποδηλώνει την πολύ καλή τους απόδοση, καθώς και την καταλληλότητά τους, όσον αφορά την υλοποίησή τους σε υλικό.

Θα πρέπει να αναφερθεί εδώ ότι, η απόδοση ενός κώδικα είναι συνάρτηση της πολυπλοκότητας αυτού και μπορεί να εξαρτάται από πολλές παραμέτρους. Από το πλήθος των κωδίκων που συναντά κάποιος στη βιβλιογραφία γίνεται, σαφές ότι δεν υπάρχει ένας κώδικας ο οποίος να είναι κατάλληλος για όλες τις εφαρμογές, να παρουσιάζει τη βέλτιστη απόδοση, να συγκλίνει γρήγορα και να έχει τη μικρότερη πολυπλοκότητα, όσον αφορά στην υλοποίηση αυτού. Κώδικες οι οποίοι έχουν χρησιμοποιηθεί κατά κόρον στο παρελθόν είναι αυτοί που απεικονίζονται στο Σχήμα 1.2.

1.3 Κώδικες Ανίχνευσης Σφαλμάτων

1.3.1 Κωδικοποίηση Ελέγχου Bit Ισοτιμίας (parity check)

Η πιο απλή μέθοδος ανίχνευσης σφαλμάτων, είναι η προσθήκη ενός bit ισοτιμίας (***parity bit***), στο τέλος της κάθε λέξης πληροφορίας. Ο δέκτης ομαδοποιεί σε ομάδες συγκεκριμένου μεγέθους τα bit πληροφορίας και προσθέτει στο τέλος κάθε ομάδας, ένα "1" ή ένα "0", ώστε το συνολικό πλήθος των "1" στην κωδικοποιημένη λέξη να είναι

άρτιο (***άρτια ισοτιμία***) ή περιττό (***περιττή ισοτιμία***). Συνήθως η άρτια ισοτιμία, χρησιμοποιείται για σύγχρονη εκπομπή και η περιττή ισοτιμία, για ασύγχρονη εκπομπή. Το χαρακτηριστικό της παραπάνω μεθόδου είναι η ιδιαίτερη απλότητά της. Σειρά όμως μειονεκτημάτων περιορίζουν την εφαρμογή της, σε συγκεκριμένες μόνο περιπτώσεις, όπως στην εγγραφή δεδομένων σε αποθηκευτικά μέσα από Η/Υ. Η συγκεκριμένη κωδικοποίηση προσφέρει την πληροφορία ύπαρξης λάθους, όχι όμως και της θέσης του. Αυτό πρακτικά σημαίνει ότι δεν είναι δυνατή η διόρθωση του σφάλματος, συνιστά δηλαδή κωδικοποίηση ανίχνευσης και όχι διόρθωσης λάθους. Άρτιο πλήθος λαθών είναι μη ανιχνεύσιμο. Αν n το πλήθος των bit που απαρτίζουν το block της αρχικής μη κωδικοποιημένης λέξης, η εφαρμογή της συγκεκριμένης κωδικοποίησης έχει ως αποτέλεσμα την πτώση του ρυθμού καθαρής πληροφορίας, από R σε $R' = \frac{nR}{n+1}$.

Παράλληλα, υπάρχουν όμως και μια σειρά από μειονεκτήματα, που αφορούν τη μέθοδο αυτή, όπως:

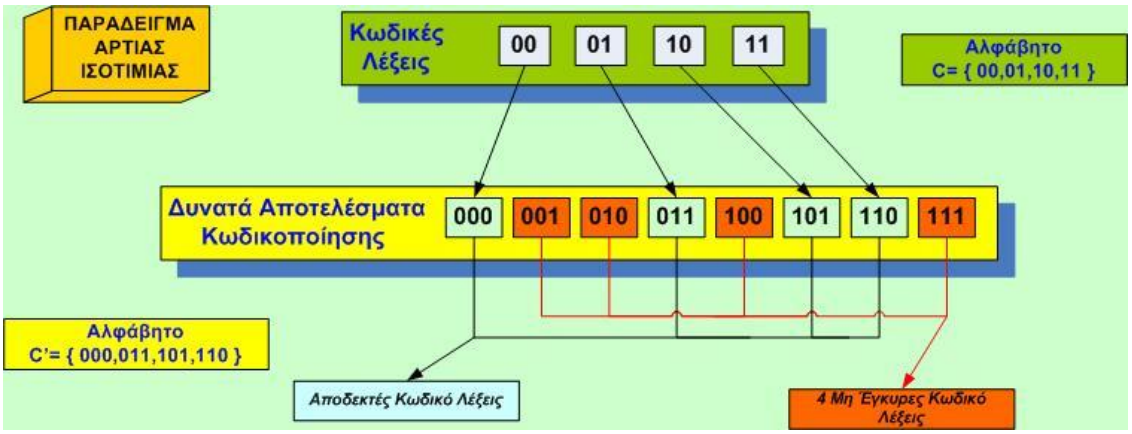
i. Στην περίπτωση που περιττό πλήθος λαθών μεγαλύτερου του ενός συμβαίνει, ο δέκτης δεν το αντιλαμβάνεται, απλά αναγνωρίζει την ύπαρξη σφάλματος.

ii. Ένα ιδιαίτερα σημαντικό μειονέκτημα της μεθόδου αφορά στην περίπτωση που το λάθος συμβεί στο bit ισοτιμίας. Σε αυτή την περίπτωση, αν και η λαμβανόμενη πληροφορία στο δέκτη είναι σωστή, αυτός πιστεύει ότι είναι λάθος.

iii. Θα πρέπει να τονιστεί ότι όσο μεγαλύτερο το μέγεθος του block, τόσο μεγαλύτερη η πιθανότητα ανίχνευσης σφάλματος, κάτι που προφανώς υποβαθμίζει την απόδοση του συστήματος. Από την άλλη, όταν το μέγεθος του block είναι μικρό, μικραίνει ο καθαρός εκπεμπόμενος ρυθμός πληροφορίας. Επιπλέον κάτι τέτοιο αυξάνει την πιθανότητα τα τυχόν σφάλματα που ανιχνεύονται, να μην οφείλονται σε λανθασμένη λήψη του σήματος πληροφορίας, αλλά σε σφάλμα στο bit ισοτιμίας.

iv. Θα πρέπει να αναφέρουμε επίσης ότι, αν δύο (ή οποιοσδήποτε άρτιος αριθμός) bit αντιστραφεί λόγω σφάλματος, δημιουργείται ένα μη ανιχνεύσιμο σφάλμα.

v. Η χρήση του bit ισοτιμίας δεν είναι απόλυτα ασφαλής, δεδομένου ότι τα σήματα θορύβου είναι συχνά αρκετά μεγάλα σε διάρκεια, για να καταστρέψουν περισσότερα του ενός bit, ιδιαίτερα σε υψηλούς ρυθμούς μετάδοσης.

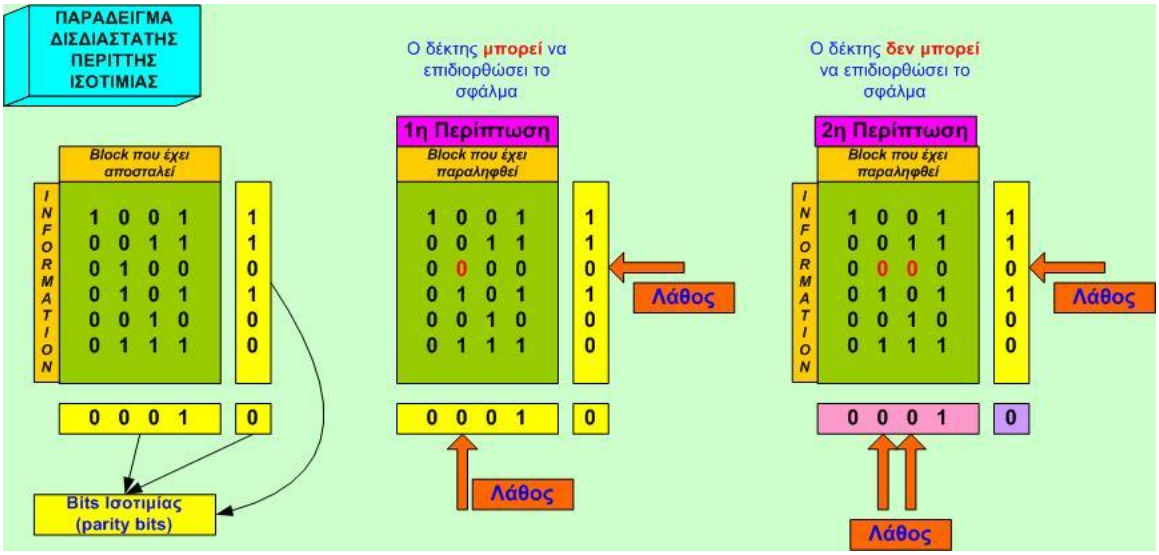


Σχήμα 1.3 Παράδειγμα Κωδικοποίησης απλής Άρτιας Ισοτιμίας

Στο παραπάνω παράδειγμα [5], έχουμε το αλφάβητο C = {00, 01, 10, 11}, από το οποίο στέλνουμε τα μηνύματά μας μέσω ενός θορυβώδους καναλιού. Όπως φαίνεται, κάθε κωδική λέξη έχει μήκος 2 και ο κώδικας περιλαμβάνει όλους τους δυνατούς συνδυασμούς μήκους 2, των ψηφίων '0' και '1'. Συνεπώς, κάθε λέξη των δύο ψηφίων που καταλήγει στο δέκτη, αποτελεί κωδική λέξη και για το λόγο αυτό, δεν υπάρχει η δυνατότητα ανίχνευσης κάποιου λάθους. Αν τώρα σε κάθε απεσταλμένο μήνυμα προσθέταμε, με βάση την παραπάνω μέθοδο που περιγράψαμε, ένα επιπλέον bit, ώστε να έχουμε ***άρτια ισοτιμία***, τότε ο δέκτης θα μπορεί να διακρίνει αν έχουν συμβεί σφάλματα στην αρχική λέξη. Άρα, εφαρμόζοντας αυτόν τον κανόνα στο προηγούμενο αλφάβητό μας, το μετασχηματίζουμε στο εξής επόμενο: C' = {000, 011, 101, 110}. Λαμβάνοντας τώρα ο δέκτης μια λέξη, η οποία δεν ανήκει σε αυτό το υποσύνολο των ακολουθιών ψηφίων μήκους 3 (πχ. κάποια από τις λέξεις που βρίσκονται στα πορτοκαλί πλαίσια του Σχήματος 1.3), συμπεραίνει ότι έχει αλλοιωθεί κάποιο από τα ψηφία αυτής και η ληφθείσα λέξη δεν είναι έγκυρη. Συνεπώς, προσθέτοντας ένα επιπλέον ψηφίο, ο δέκτης είναι σε θέση να διαπιστώσει αν έχει γίνει λάθος, σε ένα από τα ψηφία της λέξης.

Βελτίωση της παραπάνω τεχνικής αποτελεί η Δ***ισδιάστατη Ισοτιμία*** [14]. Σε αυτή την περίπτωση, οι χαρακτήρες στέλνονται κατά ομάδες, αντίστοιχα με προηγουμένως, ενώ μαζί με τους χαρακτήρες δεδομένων αποστέλλεται και μία επιπλέον ομάδα ελέγχου ισοτιμίας (parity byte). Ο υπολογισμός των ψηφίων ισοτιμίας γίνεται τόσο κατά την οριζόντια όσο και κατά την κατακόρυφη διεύθυνση, γεγονός που επιτρέπει στον παραλήπτη όχι μόνο να διαπιστώσει την ύπαρξη σφάλματος, κατά τη διάρκεια της

μετάδοσης, αλλά επιπλέον να εντοπίσει και τη θέση του εσφαλμένου ψηφίου. Η διαδικασία εντοπισμού σφαλμάτων, περιλαμβάνει την αναπαραγωγή του χαρακτήρα ισοτιμίας από τον παραλήπτη, και τη σύγκρισή του με εκείνον που παραλήφθηκε μαζί με το μήνυμα. Εάν τα δύο αποτελέσματα ταυτίζονται, τότε το πακέτο δεδομένων έχει παραληφθεί σωστά, διαφορετικά η διαδικασία μετάδοσης των δεδομένων, χαρακτηρίζεται ως εσφαλμένη. Πρακτικά τα δεδομένα τοποθετούνται σε ένα πίνακα και το άθροισμα όλων των στηλών και γραμμών, πρέπει να είναι άρτιο (***άρτια ισοτιμία***) ή περιττό (***περιττή ισοτιμία***). Η συγκεκριμένη μέθοδος προσφέρει δυνατότητα ανίχνευσης και διόρθωσης, πέραν του ενός λάθους σε μία λέξη. Συγκριτικά με την προηγούμενη τεχνική, είναι περισσότερο αποδοτική και εμφανίζει καλύτερη πιθανότητα ανίχνευσης λαθών [13].



Σχήμα 1.4 Παραδείγματα Κωδικοποίησης Δισδιάστατης Περιττής Ισοτιμίας

Στο Σχήμα 1.4 αποτυπώνεται η περίπτωση μετάδοσης συγκεκριμένου σήματος και εφαρμογής δισδιάστατης ισοτιμίας, με δύο περιπτώσεις σφαλμάτων εκ των οποίων στην $1^{η}$ περίπτωση έχει συμβεί ένα σφάλμα. Με χρησιμοποίηση των bit ισοτιμίας και στις δύο στήλες, ο δέκτης μαθαίνει τη θέση του σφάλματος και μπορεί να το διορθώσει. Στη $2^{η}$ περίπτωση έχουν συμβεί δύο σφάλματα, τα οποία και ανιχνεύονται από τα bit ισοτιμίας της οριζόντιας στήλης. Επειδή όμως αντιστοιχούν στην ίδια οριζόντια στήλη, δεν ανιχνεύονται από τα bit ισοτιμίας της κατακόρυφης και έτσι δε γίνεται να διορθωθούν. Αυτό θα ήταν εφικτό αν αντιστοιχούσαν σε διαφορετικές οριζόντιες στήλες.

1.3.2 Μέθοδος Κυκλικού Πλεονασμού *(Cyclic Redundancy Check, CRC)*

Ένας από τους πιο κοινούς, και ένας από τους πιο ισχυρούς κώδικες ανίχνευσης σφαλμάτων, είναι ο ***Κυκλικός Έλεγχος Πλεονασμού (Cyclic Redundancy Check ,CRC).*** Για ένα block ή μήνυμα, που αποτελείται από k bit, ο πομπός δημιουργεί μια ακολουθία bit μήκους (n-k), η οποία λέγεται ακολουθία πλαισίου (Frame check sequence, FCS), έτσι ώστε το πλαίσιο που προκύπτει, το οποίο αποτελείται από n bit, να διαιρείται ακριβώς με κάποιο προκαθορισμένο αριθμό, ο οποίος είναι γνωστός στον αποστολέα καθώς και στο δέκτη. Ο δέκτης διαιρεί το εισερχόμενο πλαίσιο με τον αριθμό αυτό και αν δεν υπάρξει υπόλοιπο, θεωρεί ότι δεν υπάρχει κανένα σφάλμα [3].

Για να διευκρινίσουμε όλα αυτά θα παρουσιάσουμε τη διαδικασία αναλυτικά, με τρείς τρόπους:

1. ***Την Αριθμητική modulo 2***
2. ***Τα Πολυώνυμα***
3. ***Την Ψηφιακή Λογική.***

1. Περιγραφή της Μεθόδου Αριθμητική modulo 2

Η αριθμητική modulo 2, χρησιμοποιεί δυαδική πρόσθεση χωρίς κρατούμενα, η οποία είναι απλά η πράξη ***XOR***. Η δυαδική αφαίρεση χωρίς κρατούμενα, ερμηνεύεται και αυτή ως πράξη ***XOR*** [3].

Ορίζουμε τα παρακάτω

T = *πλαίσιο n-bit που πρόκειται να μεταδοθεί*
D = *μπλοκ δεδομένων k-bit , ή μήνυμα, τα πρώτα k-bit του* T
F = *η* FCS *(n-k)-bit , τα τελευταία (n-k)-bit του* T
P = *ο προκαθορισμένος διαιρέτης (n-k+1) bit*

Σχήμα 1.5 Απαραίτητοι ορισμοί για την Περιγραφή της Μεθόδου Αριθμητική modulo 2

Το ζητούμενο είναι η διαίρεση του $\frac{T}{P}$ να μην έχει υπόλοιπο. Είναι προφανές ότι το T μπορεί να πάρει τη μορφή $T = 2^{n-k}D + F$. Πολλαπλασιάζοντας δηλαδή το D με το 2^{n-k} , ουσιαστικά μετατοπίζουμε προς τα αριστερά κατά (n-k) bit, συμπληρώνοντας τις κενές θέσεις με μηδενικά. Η προσθήκη του F δίνει τη σύνδεση σε σειρά των D και F, η οποία

είναι το T. Θέλουμε το T να διαιρείται ακριβώς με τον P. Υποθέτουμε ότι το $2^{n-k}D$ διαιρούμενο με τον P και κάνοντας χρήση της ευκλείδειας διαίρεσης, προκύπτει $\frac{2^{n-k}D}{P} = Q + \frac{R}{P}$ (1.1), υπάρχει ένα πηλίκο και ένα υπόλοιπο. Επειδή η διαίρεση είναι modulo 2, το υπόλοιπο είναι πάντοτε τουλάχιστον κατά ένα bit λιγότερο από το διαιρέτη. Θα χρησιμοποιήσουμε αυτό το υπόλοιπο για το FCS, άρα θα προκύψει η παρακάτω σχέση $T = 2^{n-k}D + R$ (1.2) . Το ερώτημα είναι αν αυτό το R ικανοποιεί τη συνθήκη μας, ότι η διαίρεση του $\frac{T}{P}$ δεν έχει υπόλοιπο. Προκειμένου να ελέγξουμε αν την ικανοποιεί θεωρούμε τα παρακάτω:

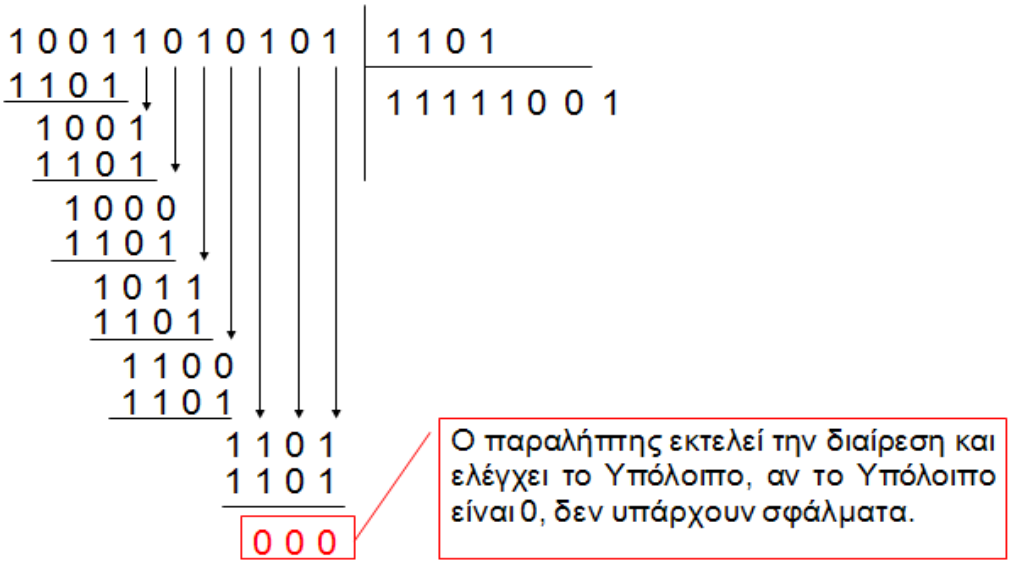
$\frac{T}{P} = \frac{2^{n-k}D+R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P} = Q + \frac{R}{P} + \frac{R}{P} = Q$ Το τελευταίο προκύπτει από την εξής πράξη σε modulo 2: οποιοσδήποτε δυαδικός αριθμός προστεθεί στον εαυτό του, δίνει μηδέν (R+R=0). Παρατηρούμε ότι δεν υπάρχει υπόλοιπο και συνεπώς το T διαιρείται ακριβώς με τον P. Επομένως, η FCS δημιουργείται εύκολα : Απλά διαιρούμε τον $2^{n-k}D$ με τον P και χρησιμοποιούμε το υπόλοιπο (n-k) bit για FCS. Στη λήψη, ο δέκτης θα διαιρέσει το T με το P και αν δεν υπάρχουν σφάλματα, δε θα προκύψει υπόλοιπο.

Παράδειγμα

Μήνυμα D = *10011010 (8 bit)*
Διαιρέτης P = *1101 (4 bit)*
FCS R = *(3 bit)*
n=11 bit k=8 bit και n-k=3 bit

Το μήνυμα πολλαπλασιάζεται με το 2^3 και μας δίνει 10011010 το γινόμενο που έχει προκύψει Διαιρείται με τον P δηλ. 1101 . Στα παρακάτω δύο σχήματα βλέπουμε τις ενέργειες που κάνει ο αποστολέας και ο παραλήπτης αντίστοιχα.

Σχήμα 1.6 Παράδειγμα για την Περιγραφή της Μεθόδου Αριθμητική modulo 2



Σχήμα 1.7 Υλοποίηση του παραδείγματος της Μεθόδου Αριθμητική modulo 2

Ο διαιρέτης P επιλέγεται να είναι κατά ένα bit μακρύτερος από την επιθυμητή FCS και η ακριβής σειρά των bit του διαιρέτη που επιλέγεται, εξαρτάται από τον τύπο των σφαλμάτων που αναμένονται. Κατ' ελάχιστο, τα δυο ακραία bit (δηλ. το υψηλότερης και χαμηλότερης τάξης bit) του P, πρέπει να είναι 1 .Υπάρχει μια σύντομη μέθοδος για τον καθορισμό της εμφάνισης ενός ή περισσοτέρων σφαλμάτων. Ένα σφάλμα έχει ως αποτέλεσμα την αναστροφή ενός bit. Αυτό ισοδυναμεί με το XOR, μεταξύ του bit και του 1 (δηλ. πρόσθεση modulo 2 του 1 στο bit). Κατά συνέπεια, τα σφάλματα σε ένα πλαίσιο n-bit, μπορούν να αναπαρασταθούν από ένα πεδίο n-bit με 1, σε κάθε θέση σφάλματος. Το πλαίσιο $T_r = T \oplus E$ όπου

T → μεταδιδόμενο πλαίσιο

E → διάταξη σφαλμάτων με 1 στις θέσεις που υπάρχουν σφάλματα

T_r → λαμβανόμενο πλαίσιο

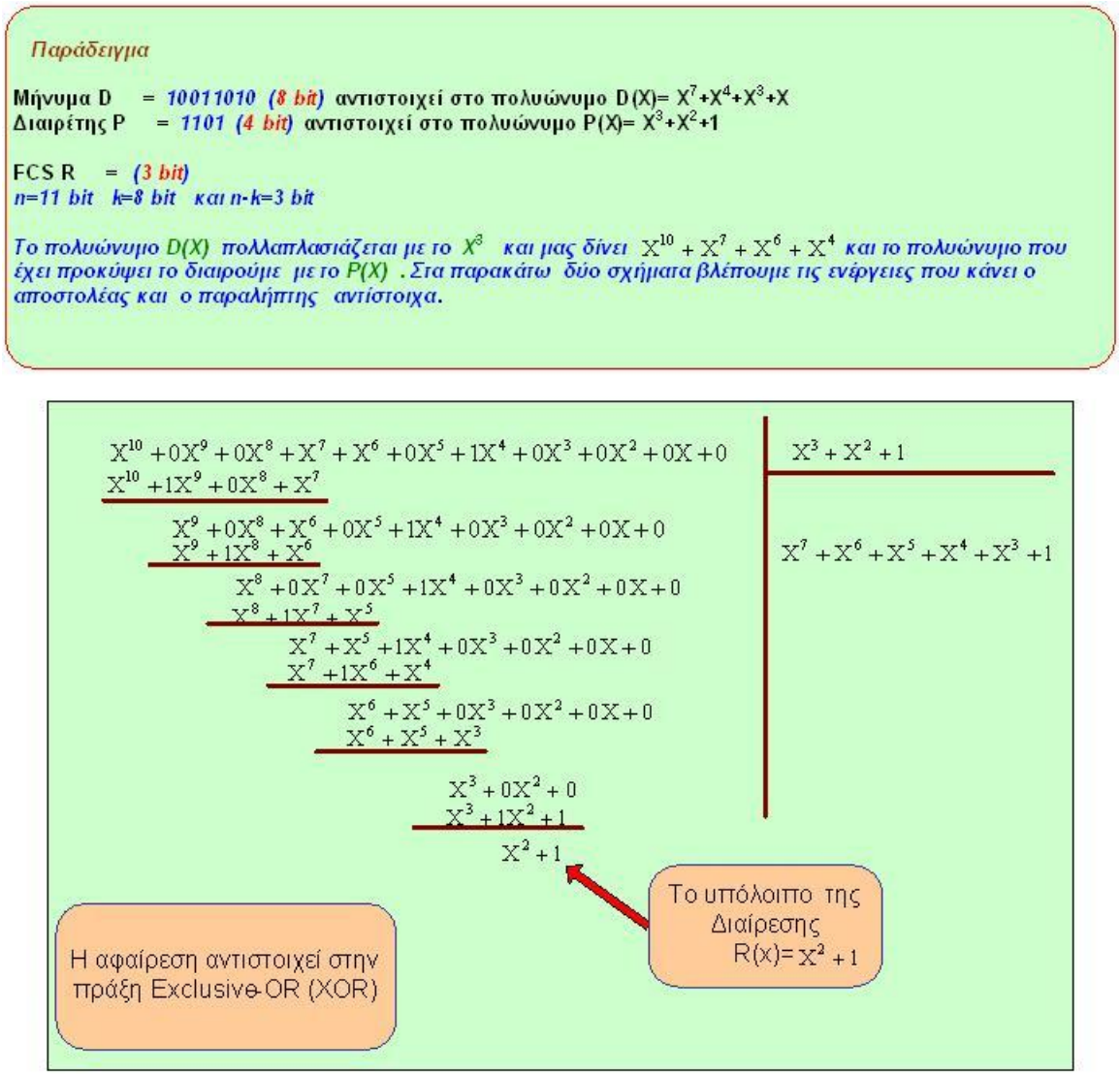
Αν υπάρχει ένα σφάλμα (E≠0), ο δέκτης θα αποτύχει να ανιχνεύσει ένα σφάλμα, αν και μόνο αν το T_r διαιρείται με τον P, γεγονός που ισοδυναμεί με το αν το E διαιρείται με το P. Αυτό όμως είναι απίθανο να συμβεί.

2. Πολυώνυμα

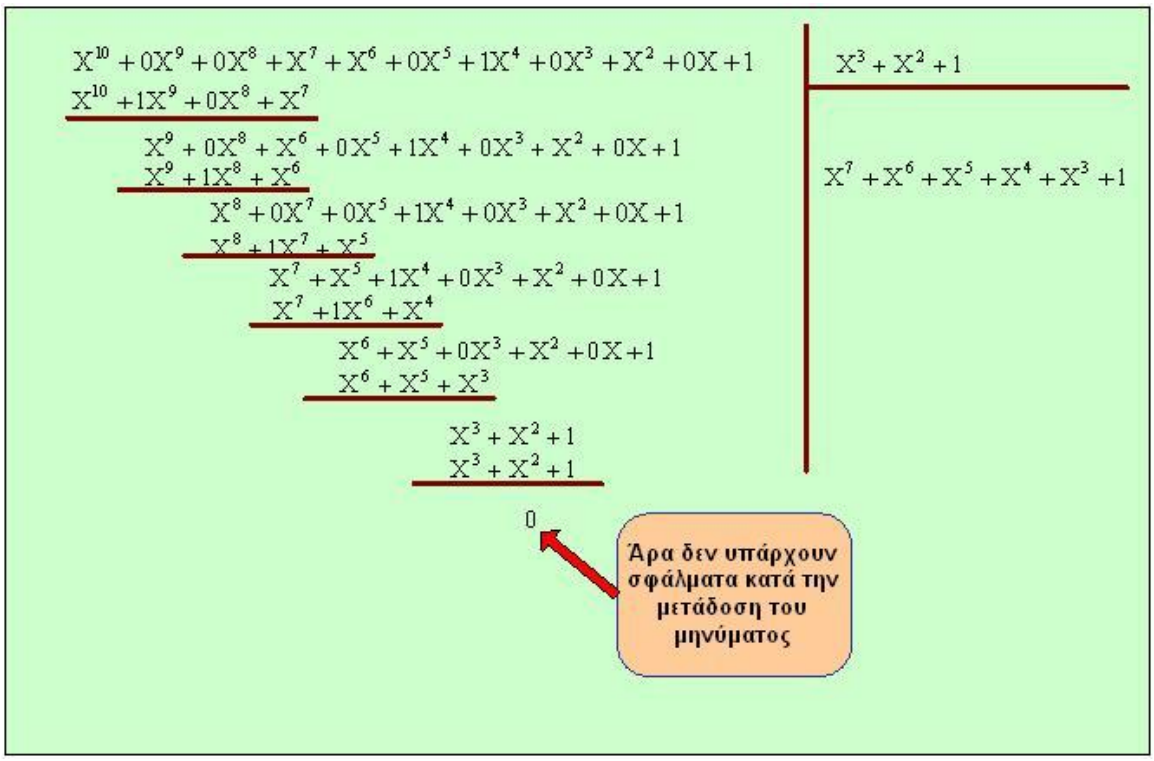
Ο δεύτερος τρόπος για να δούμε τη διαδικασία ***CRC,*** είναι να εκφράσουμε όλες τις τιμές υπό τη μορφή πολυωνύμων, σε μια πλασματική μεταβλητή X, με δυαδικούς

συντελεστές. Οι συντελεστές αντιστοιχούν στα bit του δυαδικού αριθμού. Οι αριθμητικές πράξεις είναι πάλι modulo 2. Η διαδικασία CRC μπορεί τώρα να περιγραφεί ως εξής [3]:

$$\frac{X^{n-k}D(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)} \quad (2.1) \qquad \text{και} \qquad T(X) = X^{n-k}D(X) + R(X) \quad (2.2)$$



Σχήμα 1.8 Παράδειγμα για την Περιγραφή της Μεθόδου με Πολυώνυμα



Σχήμα 1.9 Υλοποίηση του Παραδείγματος βάσει της Μεθόδου με Πολυώνυμα

3. Ψηφιακή Λογική

Η διαδικασία CRC μπορεί να αναπαρασταθεί και να υλοποιηθεί, ως ένα κύκλωμα διαίρεσης αποτελούμενο μόνο από πύλες XOR και από έναν καταχωρητή ολίσθησης . Ο καταχωρητής ολίσθησης αποτελείται από μια σειρά στοιχείων αποθήκευσης τους ενός bit. Κάθε στοιχείο έχει μια γραμμή εξόδου, η οποία δείχνει την τρέχουσα αποθηκευμένη τιμή, και μια γραμμή εισόδου. Σε διακριτές χρονικές στιγμές, που λέγονται χρόνοι ρολογιού, η τιμή στο στοιχείο αποθήκευσης αντικαθιστάται από την τιμή που υποδεικνύεται από την γραμμή εισόδου της. Τα στοιχεία αποθήκευσης του καταχωρητή χρονίζονται ταυτόχρονα, προκαλώντας ολίσθηση ενός bit, κατά μήκος ολόκληρου του καταχωρητή [3].

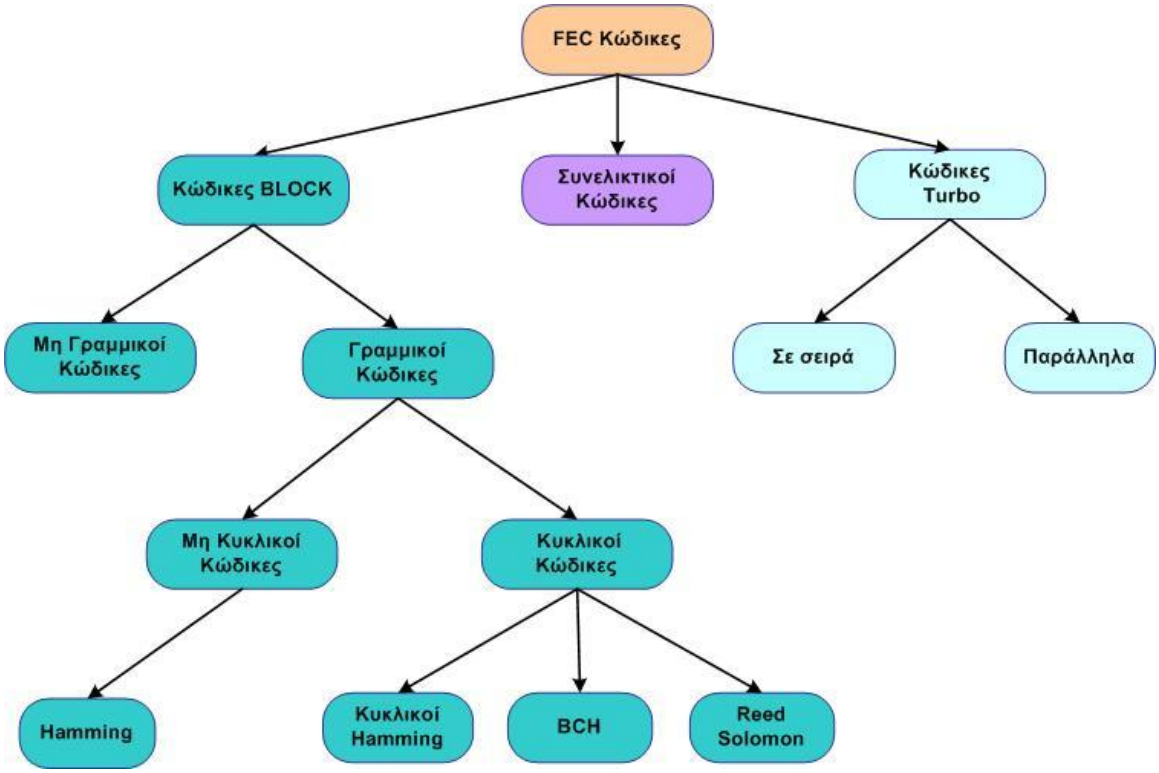
1.4 Τεχνικές Ελέγχου Σφαλμάτων

Σε περίπτωση εμφάνισης σφάλματος στο δέκτη, δύο διαφορετικές τεχνικές μπορούν να εφαρμοστούν, για την αντιμετώπισή τους, καθώς και μια τρίτη τεχνική, που αποτελεί το συνδυασμό τους.

Η πρώτη τεχνική καλείται ***Αυτόματη Αίτηση Επανεκπομπής (Automatic Repeat Request, ARQ)***. Σε ένα τέτοιο σύστημα, ο δέκτης εκτελεί ανίχνευση των σφαλμάτων και απλά ζητά από τον πομπό, ***επανεκπομπή των εσφαλμένων πακέτων δεδομένων***. Η συγκεκριμένη τεχνική συμβάλλει στην αξιοπιστία της λαμβανόμενης πληροφορίας, αν και έχει αυξημένη πολυπλοκότητα, αφού απαιτεί την ύπαρξη ενός καναλιού ανάδρασης, το οποίο όμως δεν είναι πάντα διαθέσιμο, καθιστώντας την έτσι μη πρακτική για αρκετές εφαρμογές. Στην περίπτωση της διόρθωσης των σφαλμάτων με επαναμετάδοση, ο παραλήπτης αποστέλλει στον αποστολέα, ένα ειδικό πλαίσιο ελέγχου, το οποίο περιέχει μια θετική επιβεβαίωση (ACK), εφόσον η μετάδοση υπήρξε επιτυχής, ή μια αρνητική επιβεβαίωση (NAK), εφόσον εντοπίστηκαν σφάλματα στα διακινούμενα δεδομένα. Η μέθοδος αυτή εμφανίζεται σε τρεις παραλλαγές οι οποίες χρησιμοποιούνται ανάλογα με τις περιστάσεις. Στην πρώτη περίπτωση, ***ARQ Παύσης και Αναμονής*** (ARQ Stop and Wait ή τεχνική άμεσης αναγνώρισης), για κάθε μεταδιδόμενο πακέτο, αποστέλλεται ένα πλαίσιο επιβεβαίωσης στον αποστολέα. Εάν η επιβεβαίωση αυτή είναι θετική, ο αποστολέας στέλνει το επόμενο πακέτο, ενώ στην αντίθετη περίπτωση, επαναμεταδίδει αυτό που έφτασε εσφαλμένο. Στη δεύτερη περίπτωση ***ARQ Οπισθοδρόμησης Κατά N*** (ARQ Go-back-N ή τεχνική έμμεσης αναγνώρισης), ο αποστολέας δεν περιμένει την επιβεβαίωση για το κάθε πακέτο, αλλά αποστέλλει όλα τα πακέτα της πληροφορίας, το ένα μετά το άλλο. Εάν όμως λάβει αρνητική επιβεβαίωση για κάποιο από τα προηγούμενα πακέτα, τότε επαναμεταδίδει αυτό το πακέτο, και μαζί με αυτό, όλα τα πακέτα από εκεί και κάτω, ακόμα και αν αυτά έχουν μεταδοθεί σωστά. Παραλλαγή αυτής της περίπτωσης είναι η τρίτη μέθοδος, ***ARQ Επιλεκτική Απόρριψης*** (ARQ Selective – reject ή τεχνική έμμεσης αναγνώρισης με επιλεκτική επαναμετάδοση) όπου ο αποστολέας ξαναστέλνει μόνο το εσφαλμένο πακέτο και όχι όλα τα υπόλοιπα.

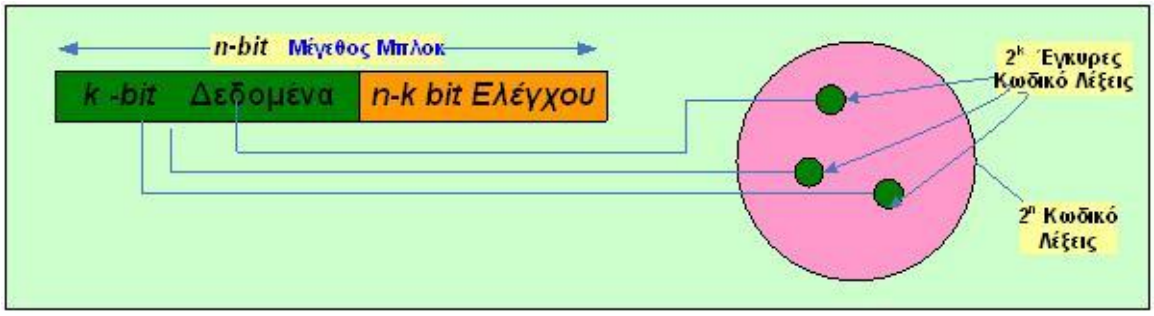
Σύμφωνα με τη δεύτερη τεχνική, που ονομάζεται ***Προωθημένη διόρθωση λαθών ή Πρόσθια διόρθωση λαθών (Forward Error Correction, FEC)***, ο δέκτης σε περίπτωση ανίχνευσης σφάλματος προβαίνει και στη διόρθωσή του σύμφωνα με τους κανόνες κωδικοποίησης. Η κωδικοποίηση FEC χρησιμοποιείται στις τηλεπικοινωνίες και στην θεωρία πληροφορίας, ως ένα σύστημα ελέγχου λαθών (error correction code), ο αποστολέας αποστέλλει πλεονάζουσα πληροφορία στο μήνυμά του, που επιτρέπει στο δέκτη να ανιχνεύσει και να διορθώσει λάθη, χωρίς να ζητήσει αναμετάδοση πακέτων. Η

τεχνική αυτή, αν και δυσκολότερη στην εφαρμογή από την ARQ, το πλεονέκτημα της έγκειται στο ότι δεν απαιτείται αμφίδρομο κανάλι για την επικοινωνία προορισμού – πηγής και συνεπώς αποφεύγεται η αναμετάδοση πακέτων και η αντίστοιχη σπατάλη εύρους ζώνης. Βασική διαφορά των δύο τεχνικών αποτελεί η διόρθωση σφαλμάτων, διαδικασία που συμβαίνει μόνο στη FEC. Η τεχνική FEC περιλαμβάνει τις παρακάτω μεγάλες κατηγορίες κωδίκων που αποτυπώνονται στο Σχήμα 1.10.



Σχήμα 1.10 FEC Κώδικες

Στους κώδικες δομής, κάθε διαδικασία κωδικοποίησης εξαρτάται μόνο από την τρέχουσα πληροφορία εισόδου και όχι από προηγούμενα ψηφία εισόδου, πράγμα που σημαίνει ότι ο κωδικοποιητής δεν έχει μνήμη. Σύμφωνα με τους κανόνες, n-k, πλεονάζοντα δυαδικά ψηφία προστίθεται σε k δυαδικά ψηφία πληροφορίας, για να σχηματίσουν τα n κωδικοποιημένα δυαδικά ψηφία [5].



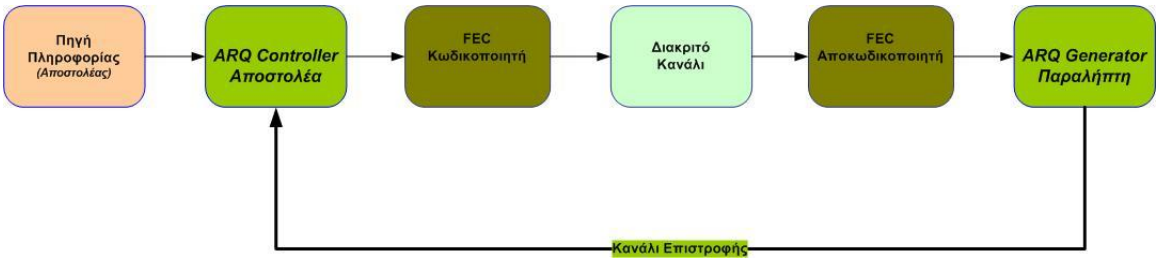
Σχήμα 1.11 Παρουσίαση έγκυρων κωδικό λέξεων που παράγονται από ένα Μπλοκ μεγέθους n-bit

Αντίθετα, στους συνελικτικούς κώδικες η έξοδος του κωδικοποιητή κάθε στιγμή, εξαρτάται όχι μόνο από την τρέχουσα πληροφορία εισόδου, αλλά και από Block δυαδικών ψηφίων που προηγήθηκαν. Μετατρέπουν δηλαδή, μια ολόκληρη ροή δεδομένων σε μία και μοναδική κωδική λέξη.

Τα μειονεκτήματα της παραπάνω μεθόδου είναι τα εξής :

- Στην περίπτωση που οι προϋποθέσεις μετάδοσης στο κανάλι είναι ευνοϊκές (δεν υπάρχουν θόρυβοι,…), επειδή προστίθεται πλεονάζουσα πληροφορία, έχει ως αποτέλεσμα τη μείωση απόδοσης .
- Σε αντίθετη περίπτωση, που οι προϋποθέσεις μετάδοσης δεν μπορούν να χαρακτηριστούν ιδανικές, δεν επαρκεί η ικανότητα διόρθωσης λαθών του FEC κώδικα, με αποτέλεσμα να εμφανίζονται σφάλματα τα οποία είναι μη διορθώσιμα.

Η κάθε μια από τις παραπάνω αναφερόμενες μεθόδους ***FEC*** και ***ARQ,*** κατέχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Ιδανικό σενάριο θα ήταν τα πλεονεκτήματα των παραπάνω Στρατηγικών, επικερδώς να τα ενσωματώσουμε σε μια νέα μέθοδο. Έτσι ώστε όταν έχουμε να κάνουμε με μεσαίες προς καλές συνθήκες καναλιού, να φροντίζει ο FEC κώδικας, για τη χωρίς σφάλματα μετάδοση, ξεπερνώντας το μειονέκτημα που παρουσιάζει το ARQ. Στην περίπτωση που οι συνθήκες που επικρατούν στο κανάλι δεν είναι ιδανικές (όπως τις προηγούμενες που περιγράψαμε), οπότε θα υπάρχουν σφάλματα κατά την μετάδοση, θα φροντίζει η ARQ μέθοδος για τη συγκεκριμένη επαναμετάδοση. Οι παραπάνω δυο Στρατηγικές, ενσωματώνονται σε μια νέα μέθοδο, που την καλούμε ***Υβριδική FEC/ ARQ Μέθοδος***, η μορφή της οποίας παρουσιάζεται στο Σχήμα 1.12 [1].



Σχήμα 1.12 Απεικόνιση ενός Υβριδικού FEC-ARQ Συστήματος

Μια μικρή περιγραφή του παραπάνω Υβριδικού FEC-ARQ Συστήματος, είναι η παρακάτω: Οι επανεκπομπές δεν είναι ενεργοποιημένες πάντα, παρά μόνο όταν ο FEC Αποκωδικοποιητής έχει διαπιστώσει λάθος στην λήψη των πακέτων και δεν μπορεί να αντιμετωπίσει δηλ. να διορθώσει τα λάθη, τότε μέσω της Γεννήτριας ARQ ενεργοποιεί την επανεκπομπή, μόνο για το πακέτο αυτό που τα λάθη του δεν αντιμετωπίστηκαν. Παρατηρούμε ότι οι επανεκπομπές ενεργοποιούνται μόνο, όταν είναι απαραίτητο και η ARQ συμβάλει αποτελεσματικά στην βελτίωση του συστήματος μετάδοσης, σχετικά με την χρήση μόνο ευθείας κωδικοποίησης σφάλματος FEC .

Στο επόμενο κεφάλαιο, θα ασχοληθούμε με τους συνελικτικούς κώδικες, που αποτελούν μια από τις βασικές κατηγορίες από τους FEC κώδικες, οι οποίοι βρίσκουν ευρεία εφαρμογή, τόσο σε εμπορικά όσο και σε στρατιωτικά συστήματα δορυφορικών επικοινωνιών, διότι υπάρχουν ιδιαίτερα αποδοτικοί αλγόριθμοι κωδικοποίησης για διόρθωση λαθών, που οδηγούν σε μεγάλα κέρδη κωδικοποίησης .

1.5 Τεχνικές Διαφορικής Λήψης

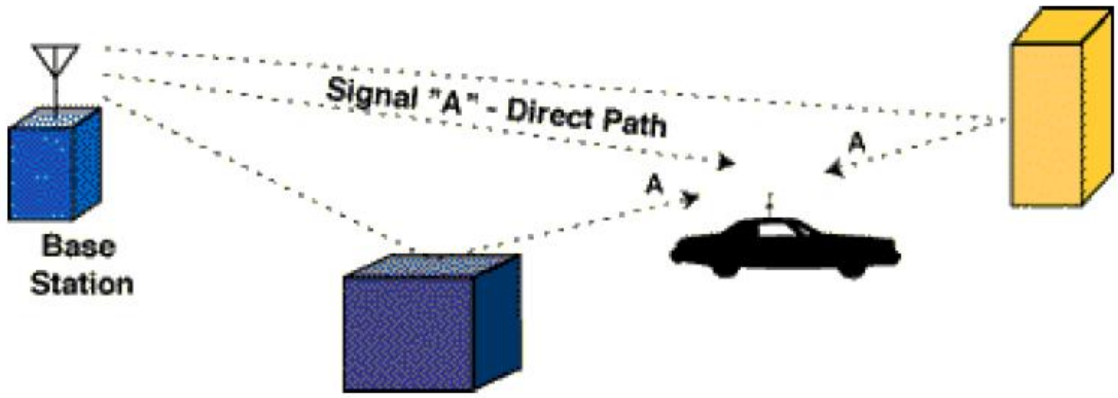
Στις τηλεπικοινωνίες, ο όρος διαφορική λήψη ή διαφορισμός (diversity) αναφέρεται σε μια μέθοδο, για την βελτίωση της λήψης ενός ραδιοσήματος σε περιβάλλον διαλείψεων. Σύμφωνα με την μέθοδο αυτή, δυο ή περισσότερα αντίγραφα του αρχικού σήματος λαμβάνονται σε δέκτη, του οποίου οι κεραίες είναι τοποθετημένες σε θέσεις, που απέχουν κάποια συγκεκριμένη απόσταση (η απόσταση αυτή μπορεί να είναι και λιγότερη από μέτρο) [15]. Ένα ηλεκτρονικό κύκλωμα, ή ένα κατάλληλο λογισμικό, συνδυάζει ή επιλέγει από την κεραία λήψης, έτσι ώστε να λάβει ένα σήμα βελτιωμένης ποιότητας.

Υπάρχουν πολλοί παράγοντες τους οποίους πρέπει να λάβουμε υπόψη, κατά τη διάδοση του ηλεκτρομαγνητικού κύματος, μέσω ενός ασύρματου καναλιού. Αυτοί οι παράγοντες αποτελούν και χαρακτηριστικά συγχρόνως, τα οποία βοηθούν ώστε να σχεδιαστεί όσο το δυνατό αρτιότερα, το όλο ασύρματο τηλεπικοινωνιακό κανάλι. Τέτοιοι παράγοντες είναι τα κύρια χαρακτηριστικά του περιβάλλοντος, μεταξύ του πομπού και του δέκτη, ο καιρός που επικρατεί, τυχόν εμπόδια που μπορεί να παρεμβάλλονται μεταξύ των δύο κεραιών και γενικά οτιδήποτε μπορεί να θεωρηθεί ότι επηρεάζει το όλο σύστημα, από τη στιγμή που το κύμα θα εκπεμφθεί από την κεραία εκπομπής.

Ανάλογα τώρα με το περιβάλλον που λειτουργεί, ένα ασύρματο τηλεπικοινωνιακό σύστημα, όπως είναι φυσιολογικό, το ηλεκτρομαγνητικό κύμα, το σήμα δηλαδή, φτάνει στο δέκτη διαμέσου πολλών διαδρομών. Αυτές οι πολλαπλές διαδρομές που είναι δυνατόν να ακολουθήσει το σήμα μας, μέχρι να φτάσει από τον πομπό στο δέκτη, δημιουργούν μικρής κλίμακας διακυμάνσεις στη στάθμη ισχύος του σήματος, ένα φαινόμενο που περιγράφεται με τον όρο διαλείψεις (fading). Το αποτέλεσμα αυτού του φαινομένου είναι η ποιότητα της επικοινωνίας να υποβαθμίζεται ανάλογα με το πόσο έντονα λαμβάνει χώρα στο κανάλι μας το φαινόμενο. Για την αντιμετώπιση του φαινομένου των διαλείψεων σε ένα ασύρματο τηλεπικοινωνιακό κανάλι εφαρμόζεται η τεχνική του διαφορισμού (diversity), η οποία αποτελεί την δημοφιλέστερη μέθοδο καταπολέμησης των διαλείψεων. Έτσι με την κατάλληλη αξιοποίηση των πολλαπλών αντιγράφων του σήματος εκπομπής στο δέκτη έχουμε πολύ καλές επιδόσεις ακόμη κι όταν το σήμα μας έχει υποστεί μεγάλη εξασθένιση. Η τεχνική του διαφορισμού βασίζεται στο γεγονός πως η πιθανότητα όλα τα λαμβανόμενα αντίγραφα να έχουν χαμηλή στάθμη ισχύος είναι πολύ μικρή. Εκμεταλλευόμενοι λοιπόν το γεγονός ότι ένα τουλάχιστον αντίγραφο θα είναι ισχυρό, υιοθετούμε κανόνες επιλογής με βάση τους οποίους επιλέγουμε το βέλτιστο αντίγραφο του λαμβανόμενου σήματος

Πολλά σύγχρονα αλλά και ανερχόμενα ασύρματα συστήματα επικοινωνίας κάνουν χρήση μέσω κάποιας μορφής των τεχνικών διαφόρισης (diversity) μιας τεχνικής που αποσκοπεί στην καταπολέμηση του φαινομένου των διαλείψεων λόγω των πολλαπλών διαδρομών (multipath) που μπορεί να ακολουθήσει το σήμα.

Επομένως το ηλεκτρομαγνητικό κύμα ακολουθεί διαφορετικές διαδρομές διαφορετικού μήκους στη μετάδοσή του από τον πομπό στο δέκτη. Αυτό το τελευταία καλείται φαινόμενο των πολλαπλών διοδεύσεων (multipath).



Σχήμα 1.13 Απεικόνιση Φαινομένου Πολυδιόδευσης

Υπάρχουν τεχνικές διαφορικής εκπομπής μέσω των οποίων επιτυγχάνεται ο σωστός συνδυασμός των σημάτων εκπομπής ώστε στο δέκτη να έχουμε μια αξιόπιστη λήψη για μεγάλο σχετικά χρονικό διάστημα. Υπάρχουν και τεχνικές διαφορικής λήψης που έχουν εφάμιλλο με τις τεχνικές διαφορικής εκπομπής στόχο, τον περιορισμό των διαλείψεων πολυδιόδευσης (multipath fading).

Γνωστές τεχνικές διαφορισμού [15]:

- Διαφορισμός Χώρου (*space diversity*)
- Διαφορισμός Πόλωσης (polarization diversity)
- Διαφορισμός Συχνότητας (frequency diversity)
- Διαφορισμός Χρόνου (time diversity)
- Διαφορισμός Κατεύθυνσης (direction diversity)
- Διαφορισμός Διαδρομής (path diversity)

1.5.1 Διαφορισμός Μεγάλης Κλίμακας (Macrodiversity)

Στον τομέα της ασύρματης επικοινωνίας, η τεχνική του μακροδιαφορισμού

(macrodiversity) αποτελεί ένα είδος σχεδίου διαφορισμού όπου διάφορες κεραίες δεκτών ή/και οι κεραίες πομπών χρησιμοποιούνται για τη μεταφορά του ίδιου σήματος. Η απόσταση μεταξύ των πομπών είναι πολύ μεγαλύτερη από το μήκος κύματος, σε αντιδιαστολή με τις τεχνικές μικροδιαφορισμού (microdiversity) όπου η απόσταση είναι της ίδιας τάξεως ή μικρότερη από το μήκος κυμάτων [16].

Σε ένα κυψελοειδές δίκτυο ή σε ένα ασύρματο τοπικό δίκτυο, η τεχνική μακροδιαφορισμού προϋποθέτει ότι οι κεραίες είναι τοποθετημένες σε διαφορετικές περιοχές ή σημεία πρόσβασης. Βάση της τεχνική αυτής προϋποθέτει στον δέκτη είναι μια μορφή συνδυασμού κεραιών, και απαιτεί μια υποδομή η οποία μετατρέπει τα σήματα από τις τοπικές κεραίες ή δέκτες σε έναν κεντρικό δέκτη. Ο μακροδιαφορισμός των πομπών μπορεί να είναι μια μορφή ταυτόχρονης εκπομπής, όπου το ίδιο σήμα στέλνεται από διάφορους κόμβους. Εάν τα σήματα στέλνονται από το ίδιο φυσικό κανάλι (π.χ. η συχνότητα καναλιών και η ακολουθία διάδοσης),τότε λέγεται ότι οι πομποί διαμορφώσουν ένα ενιαίο δίκτυο κοινής συχνότητας - ένας όρος που χρησιμοποιείται ειδικά στον κόσμο της ραδιοφωνικής αναμετάδοσης.

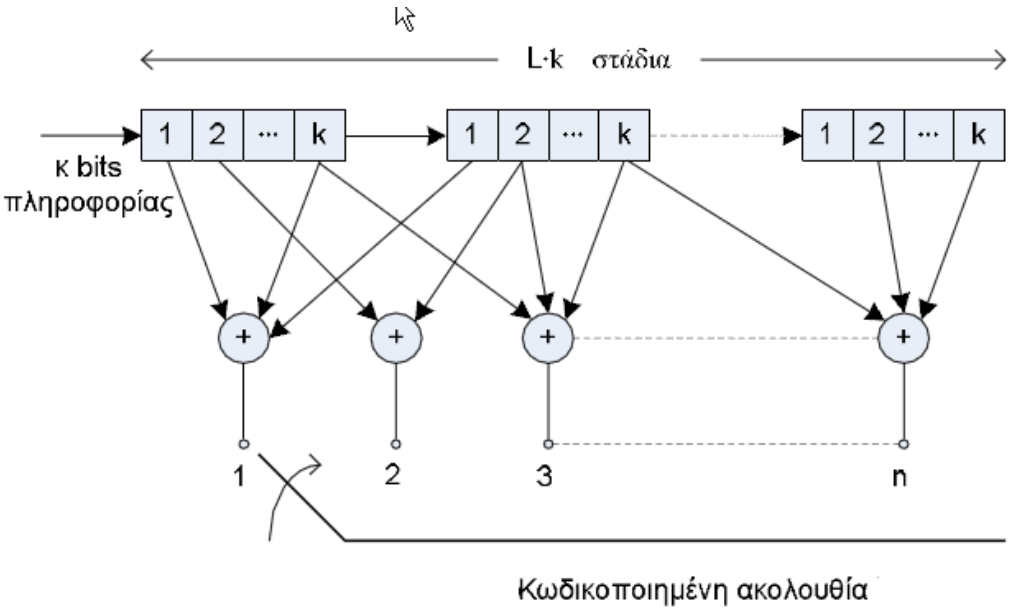
Ο απώτερος στόχος είναι να καταπολεμηθεί η εξασθένιση και να αυξηθεί η δύναμη των σημάτων λήψης και η ποιότητά τους σε θέσεις οι οποίες είναι εκτεθειμένες μεταξύ των σταθμών βάσεων ή των σημείων πρόσβασης. Η τεχνική αυτή μπορεί επίσης να διευκολύνει και να παρέχει πιο αποδοτικές υπηρεσίες ραδιοφωνικής αναμετάδοσης και ταυτόχρονης εκπομπής, όπου το ίδιο κανάλι συχνότητας μπορεί να χρησιμοποιηθεί για όλους τους πομπούς που στέλνουν τις ίδιες πληροφορίες. Το σχέδιο διαφορισμού μπορεί να βασιστεί στην macrodiversity τεχνική συσκευών αποστολής σημάτων (downlink) ή/και την macrodiversity των δεκτών. (uplink).

2.Συνελικτικοί Κώδικες (Convolutional Codes)

Στο προηγούμενο Κεφάλαιο πραγματοποιήθηκε εισαγωγή στους κώδικες και στην συνέχεια θα εστιάσουμε το ενδιαφέρον μας στους συνελικτικούς κώδικες γιατί αυτό πηγάζει κυρίως από τη μεγάλη εφαρμογή τους στην ασύρματη μετάδοση. Στο κεφάλαιο αυτό περιγράφουμε τους Συνελικτικούς Κώδικες ως προς τη διαδικασία της Κωδικοποίησης, καθώς και της Αποκωδικοποίησής τους [3].

2.1 Εισαγωγή στους Συνελικτικούς Κώδικες

Οι Συνελικτικοί κώδικες αποτελούν μια κατηγορία κωδίκων, οι οποίοι επιτυγχάνουν μεγάλα κέρδη κωδικοποίησης και χρησιμοποιούνται ευρέως στα σύγχρονα τηλεπικοινωνιακά συστήματα. Το όνομά τους αυτό προέρχεται από τον τρόπο με τον οποίο επιτυγχάνεται η κωδικοποίηση του σήματος πληροφορίας, αφού ο σχηματισμός τους μπορεί να θεωρηθεί ως η συνέλιξη των ψηφίων πληροφορίας, με την κρουστική απόκριση ενός καταχωρητή ολίσθησης. Οι Συνελικτικοί Κώδικες μεταχειρίζονται τα δεδομένα εισόδου, σαν μια συνεχή ακολουθία δεδομένων, σε αντίθεση με τους Κώδικες block που συγκεντρώνουν τα εισερχόμενα bits σε ομάδες (blocks), και παράγουν μεγαλύτερες ομάδες στην έξοδό τους.



Σχήμα 2.1 Σχηματικό Διάγραμμα ενός Συνελικτικού Κωδικοποιητή

Το κύριο χαρακτηριστικό τους είναι ότι διαθέτουν μνήμη, κατά τη διαδικασία Κωδικοποίησης. Με τη χρήση Συνελικτικών Κωδίκων, ένα μπλοκ k δυαδικών ψηφίων, αντιστοιχίζεται σε ένα μπλοκ n δυαδικών ψηφίων. Τα n δυαδικά ψηφία είναι αυτά τα οποία μεταδίδονται μέσω ενός τηλεπικοινωνιακού καναλιού. Το βασικό χαρακτηριστικό των Συνελικτικών κωδίκων είναι ότι, το μπλοκ των καινούργιων n δυαδικών ψηφίων δεν εξαρτάται μόνο από το τρέχον μπλοκ των k δυαδικών ψηφίων, αλλά και από προηγούμενα δυαδικά ψηφία. Η εξάρτηση από τα προηγούμενα δυαδικά ψηφία μετατρέπει τον Κωδικοποιητή σε μια Μηχανή Πεπερασμένων Καταστάσεων. Στη γενική περίπτωση, ένας Συνελικτικός κωδικοποιητής αποτελείται από ένα καταχωρητή ολίσθησης k·L σταδίων και n modulo-2 αθροιστές και απεικονίζεται στο Σχήμα 2.1 [2].
Το L είναι το Μήκος Περιορισμού του κώδικα. Σε κάθε χρονική στιγμή, k δυαδικά ψηφία πληροφορίας εισέρχονται στον καταχωρητή ολίσθησης και τα περιεχόμενα του τελευταίου σταδίου του καταχωρητή ολίσθησης, k δυαδικά ψηφία, εξάγονται. Το Μήκος Περιορισμού είναι ο αριθμός των ολισθήσεων των k δυαδικών ψηφίων της εισόδου, που μπορούν να επηρεάσουν την έξοδο του κωδικοποιητή. Από τη στιγμή που τα k δυαδικά ψηφία εισέρχονται στον καταχωρητή ολίσθησης, n γραμμικοί συνδυασμοί των περιεχομένων του καταχωρητή ολίσθησης υπολογίζονται και χρησιμοποιούνται για την παραγωγή της κωδικοποιημένης εξόδου. Από τα παραπάνω γίνεται φανερό ότι η έξοδος των n δυαδικών ψηφίων, δεν εξαρτάται μόνο από τα k τελευταία δυαδικά ψηφία που μπήκαν στον κωδικοποιητή, αλλά και από τα (L-1)·k περιεχόμενα του καταχωρητή ολίσθησης, που υπάρχουν μετά την εισαγωγή των k δυαδικών ψηφίων. Ο καταχωρητής ολίσθησης είναι μια μηχανή πεπερασμένων καταστάσεων με $2^{(L-1)\cdot k}$ στάδια. Επειδή, στον συγκεκριμένο κωδικοποιητή, για κάθε k δυαδικά ψηφία εισόδου, παράγονται n δυαδικά ψηφία εξόδου, ο ρυθμός του κώδικα είναι $R_c = \frac{k}{n}$.

Ένας Συνελικτικός Κώδικας χαρακτηρίζεται από τις τρεις παραμέτρους **(n,k,m)** :

- **n** είναι ο αριθμός των δυαδικών ψηφίων εξόδου
- **k** είναι ο αριθμός των δυαδικών ψηφίων εισόδου
- **m** είναι ο αριθμός των καταχωρητών (flip – flop)

Συχνά όμως, ένας Συνελικτικός κώδικας χαρακτηρίζεται και από τις εξής τρεις παραμέτρους (n,k,L), όπου L είναι το Μήκος Περιορισμού του κώδικα και ορίζεται ως: Μήκος Περιορισμού: L= k (m-1) .

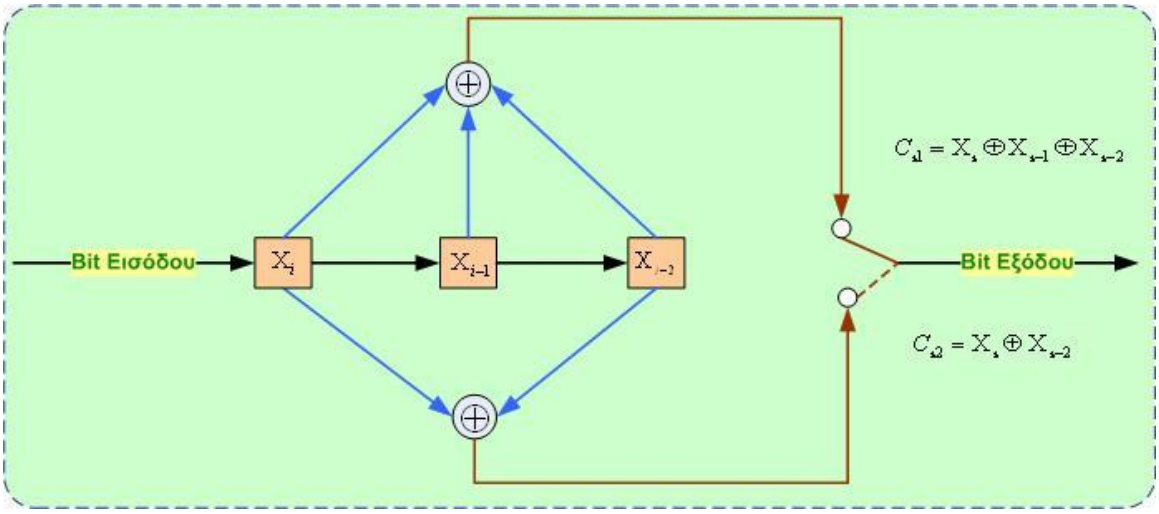
Γενικά τους Συνελικτικούς κώδικες τους διακρίνουμε σε δυο κατηγορίες: τους Συστηματικούς και τους Μη Συστηματικούς, οι οποίοι με την σειρά τους διακρίνονται στους Αναδρομικούς και στους Μη Αναδρομικούς. Το παράδειγμα που θα ακολουθήσει έχει να κάνει με Συστηματικό Αναδρομικό Συνελικτικό Κωδικοποιητή. Ένας Συνελικτικός Κώδικας καλείται Συστηματικός όταν, το μπλοκ της εισόδου των κ δυαδικών ψηφίων παρουσιάζεται αυτούσιο στην έξοδο, ως ένα μέρος από τα n δυαδικά ψηφία της εξόδου. Ένας Μη Συστηματικός Συνελικτικός Κώδικας δημιουργείται όταν η έξοδος δεν περιέχει αυτούσια τα δυαδικά ψηφία της εισόδου. Οι Μη Συστηματικοί Συνελικτικοί Κώδικες παρουσιάζουν καλύτερη απόδοση, δηλ. μεγαλύτερη απόσταση μεταξύ των κωδικών λέξεων, για το ίδιο Μήκος Περιορισμού και τον ίδιο ρυθμό του κώδικα. Γενικά οι Συστηματικοί Συνελικτικοί κώδικες προτιμούνται περισσότερο από τους Μη Συστηματικούς Συνελικτικούς κώδικες και αυτό κυρίως γιατί μπορούν εύκολα να ελεγχθούν, ως προς την ορθότητά τους, απαιτούν λιγότερο υλικό για την κατασκευή του κωδικοποιητή και επιπλέον δεν είναι Καταστροφικοί (ένας Κώδικας καλείται καταστροφικός όταν μεταδίδει ένας λάθος που έχει συμβεί σε μια χρονική στιγμή και σε όλες τις επόμενες μέχρι το τέλος του μηνύματος) [2] .

2.2 Εναλλακτικοί τρόποι Αναπαράστασης Συνελικτικού Κωδικοποιητή

Η διαδικασία κωδικοποίησης για έναν Συνελικτικό κώδικα είναι πολύ απλή. Ο κωδικοποιητής ενός συνελικτικού κώδικα αποτελείται από απλά επί μέρους στοιχεία: Καταχωρητές ολίσθησης (Shift Registers) και αθροιστές modulo-2. Αρχικά ο κωδικοποιητής περιέχει μόνο μηδενικά, πριν εισέλθει το πρώτο bit πληροφορίας. Τα bits πληροφορίας εισέρχονται στον κωδικοποιητή ανά k bits τη φορά και τα αντίστοιχα n bits της εξόδου του στέλνονται για μετάδοση. Η διαδικασία αυτή συνεχίζεται μέχρι την καταχώρηση και της τελευταίας ομάδας των k bits στον κωδικοποιητή και την αποστολή της τελευταίας n-άδας των bits εξόδου μέσω του καναλιού. Ο Συνελικτικός κώδικας είναι γραμμικός, πράγμα που σημαίνει ότι αν οι κωδικές λέξεις c1 και c2 αντιστοιχούν στα

μηνύματα m1 και m2 αντίστοιχα, τότε το άθροισμα των κωδικών λέξεων αντιστοιχεί στο άθροισμα των μηνυμάτων.

Οι Συνελικτικοί Κωδικοποιητές μπορούν να κατανοηθούν πιο εύκολα εξετάζοντας ένα συγκεκριμένο παράδειγμα το οποίο φαίνεται στο παρακάτω Σχήμα 2.2.



Σχήμα 2.2 Συστηματικός Συνελικτικός Κωδικοποιητής ρυθμού $\frac{1}{2}$ (n,k,m)=(2,1,3)

Για το παράδειγμά μας, ο κωδικοποιητής μετατρέπει το ένα bit εισόδου Xi σε δυο bit εξόδου C_{i1} και C_{i2}, χρησιμοποιώντας κάθε φορά τα τρία πιο πρόσφατα bit. Το πρώτο bit εξόδου, δημιουργείται από το πάνω λογικό κύκλωμα $C_{i1} = X_i \oplus X_{i-1} \oplus X_{i-2}$ και το δεύτερο bit εξόδου, από το κάτω λογικό κύκλωμα $C_{i2} = X_i \oplus X_{i-1} \oplus X_{i-2}$. Στο παράδειγμά μας θα θεωρήσουμε ως ακολουθία εισόδου τη ***X=(1011)*** **.**

Χρόνος	Είσοδος	Καταχωρητές Ολίσθησης			Έξοδος	
T	x	Register 1	Register 2	Register 3	C_1	C_2
0	1	1	0	0	1	1
1	0	0	1	0	1	0
2	1	1	0	1	0	0
3	1	1	1	0	0	1
4	0	0	1	1	0	1
5	0	0	0	1	1	1

Πίνακας 2-1 Προσδιορισμός της ακολουθίας εξόδου με ακολουθία εισόδου την 1011

Παρατηρούμε ότι προκύπτει η ακόλουθη κωδικοποιημένη ακολουθία ***111000010111.*** Στη συνέχεια θα προσπαθήσουμε με πέντε διαφορετικούς τρόπους να παραστήσουμε τους Συνελικτικούς Κώδικες.

2.2.1 Διακριτή συνέλιξη της ακολουθίας εισόδου με βάση την Κρουστική Απόκριση του Κωδικοποιητή

Η κρουστική απόκριση ενός συνελικτικού κωδικοποιητή ορίζεται ως η απόκρισή του σε ένα και μόνο ψηφίο 1, το οποίο διέρχεται από αυτόν. Αν ένας κωδικοποιητής έχει K στάδια, τότε η μονάδα θα βρίσκεται επί K χρονικά βήματα στον κωδικοποιητή, οπότε η κρουστική απόκριση του κωδικοποιητή θα αποτελείται από K n-άδες ψηφίων. Γνωρίζοντας την κρουστική απόκριση του κωδικοποιητή, είναι δυνατό να υπολογιστεί η έξοδός του σε μία συγκεκριμένη ακολουθία ψηφίων εισόδου, αν υπολογιστεί η συνέλιξη των ψηφίων εισόδου με την απόκριση του κωδικοποιητή. Ισοδύναμα, υπολογίζεται το modulo – 2 άθροισμα των αποκρίσεων του κωδικοποιητή σε κάθε ψηφίο εισόδου, αφού πρώτα οι αποκρίσεις αυτές έχουν μετατοπιστεί κατάλληλα στο χρόνο, ανάλογα με τη χρονική στιγμή που εισήχθη το αντίστοιχο ψηφίο εισόδου στον κωδικοποιητή. Σε κάθε είσοδο 1, η έξοδος του αποκωδικοποιητή παίρνεται ίση με την κρουστική απόκρισή του, ενώ σε κάθε είσοδο 0, η απόκρισή του θεωρείται μηδενική. Από τα παραπάνω, γίνεται φανερό ότι οι Συνελικτικοί κώδικες είναι γραμμικοί.

Θα προσπαθήσουμε να προσεγγίσουμε τη λειτουργία του κωδικοποιητή μέσα από την έννοια της κρουστικής απόκρισης, της απόκρισης δηλαδή του κωδικοποιητή σε ένα και μοναδικό bit, που μετακινείται στις βαθμίδες. Στον πίνακα που ακολουθεί περιγράφονται οι έξοδοι του κωδικοποιητή του παραδείγματός μας, καθώς μετακινείται μέσα του μια μονάδα .

Χρόνος	**Καταχωρητές Ολίσθησης**			**Έξοδος**	
t	**Register 1**	**Register 2**	**Register 3**	**C_1**	**C_2**
t_0	1	0	0	**1**	**1**
t_1	0	1	0	**1**	**0**
t_2	0	0	1	**1**	**1**

Πίνακας 2-2 Απόκριση του Κωδικοποιητή με ένα και μοναδικό bit να μετακινείται στις βαθμίδες του

Η Κρουστική Απόκριση του κωδικοποιητή είναι η ***111011 .***

Γνωρίζοντας την Κρουστική Απόκριση ενός κωδικοποιητή, μπορεί κανείς να βρει την ακολουθία εξόδου για μια συγκεκριμένη ακολουθία μέσου, του γραμμικού αθροίσματος μετατοπισμένων κρουστικών αποκρίσεων. Στο παράδειγμά μας για την ακολουθία εισόδου 1011 προκύπτουν τα παρακάτω [7]:

*	11	10	11			
1	11	10	11			
0		00	00	00		
1			11	10	11	
1				11	10	11
Άθροισμα Modulo-2	**11**	**10**	**00**	**01**	**01**	**11**

Πίνακας 2-3 Εναλλακτικός τρόπος παραγωγής της ακολουθίας εξόδου με ακολουθία εισόδου 1011

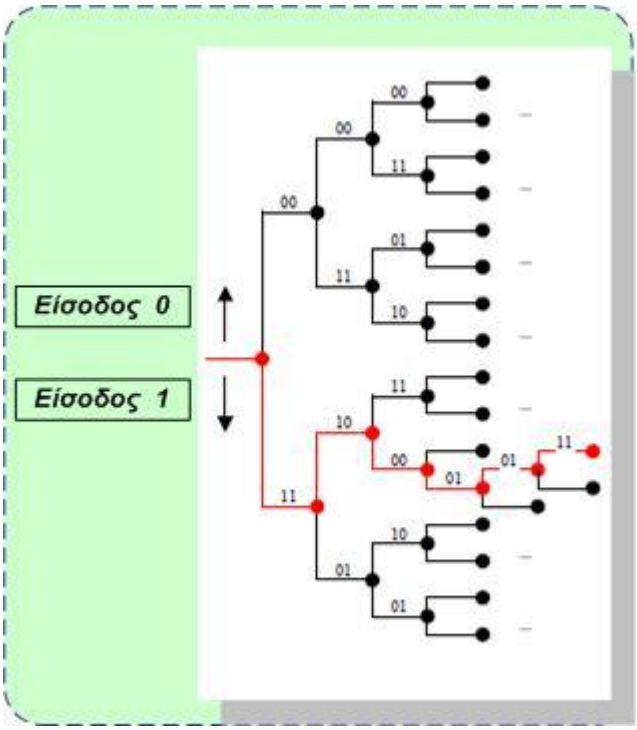
Παρατηρούμε ότι λάβαμε την ίδια ακολουθία εξόδου όπως και πριν, κάτι που δείχνει ότι Συνελικτικοί κώδικες είναι γραμμικοί.

2.2.2 Δεντρικό Διάγραμμα (Tree Diagram)

Το Δεντρικό Διάγραμμα ή και Δένδρο Κωδικοποίησης, αποτελεί το δεύτερο τρόπο παράστασης της διαδικασίας Συνελικτικής Κωδικοποίησης, ο οποίος δίνει πληροφορία και για την εξέλιξη της κωδικοποίησης στο χρόνο. Σε κάθε χρονική στιγμή, που καθορίζεται με την έλευση ενός δυαδικού ψηφίου, διασχίζουμε το διάγραμμα από αριστερά προς τα δεξιά. Κάθε κλαδί του δέντρου δείχνει την έξοδο του συστήματος, ενώ κάθε κόμβος του δέντρου δείχνει την κατάσταση που βρίσκεται το σύστημα. Ο κανόνας για την εύρεση της ακολουθίας εξόδου είναι ο εξής: Αν το δυαδικό ψηφίο εισόδου είναι μηδέν, τότε γίνεται μετακίνηση στο δέντρο, προς το κλαδί που βρίσκεται όσο γίνεται πιο δεξιά και προς τα πάνω. Αν το δυαδικό ψηφίο εισόδου είναι ένα, τότε γίνεται μετακίνηση στο δέντρο, προς το κλαδί που βρίσκεται όσο γίνεται πιο δεξιά και προς τα κάτω.

Αν θεωρηθεί ότι η αρχική κατάσταση του κωδικοποιητή είναι η μηδενική, τότε το δεντρικό διάγραμμα δείχνει ότι αν το δυαδικό ψηφίο εισόδου είναι μηδέν, τότε η έξοδος είναι η 00 και παραμένει στην κατάσταση 00, ενώ αν το δυαδικό ψηφίου εισόδου είναι ένα, η έξοδος είναι 11 και το σύστημα μεταβαίνει στην κατάσταση 10.Το δεντρικό διάγραμμα μπορεί να εισάγει τη διάσταση του χρόνου, αλλά δημιουργεί και ένα σοβαρό μειονέκτημα. Μεγαλώνει τον αριθμό των κλαδιών με συνάρτηση 2L, όπου L είναι ο αριθμός των διαφορετικών εξόδων του συστήματος. Αυξάνονται έτσι οι απαιτήσεις, τόσο σε αποθηκευτικό χώρο, όσο και σε υπολογιστική ισχύ, για τον υπολογισμό μεγάλου

αριθμού κλαδιών. Μετά την πάροδο σχετικά λίγων χρονικών στιγμών, η χρήση του Δεντρικού Διαγράμματος γίνεται απαγορευτική [7].



Σχήμα 2.3 Δένδρο κωδικοποίησης Συνελικτικού Κώδικα ρυθμού $\frac{1}{2}$ (n,k,m)=(2,1,3)

Ωστόσο, το δένδρο κωδικοποίησης, σε αντίθεση με το διάγραμμα καταστάσεων που είναι στατικό στο χρόνο, είναι δυναμικό και μετά από ελάχιστες χρονικές στιγμές το μέγεθος του αυξάνει κατά πολύ. Για δυαδικούς συνελικτικούς κώδικες, με $k=1$, σε κάθε χρονική στιγμή, από κάθε κόμβο, έχουμε δύο διακλαδώσεις, οπότε την i-οστή χρονική στιγμή, έχουμε 2^{i-1} κόμβους. Η ιδιότητα αυτή καθιστά την αναπαράσταση με το δένδρο κωδικοποίησης, σχετικά δύσχρηστη.

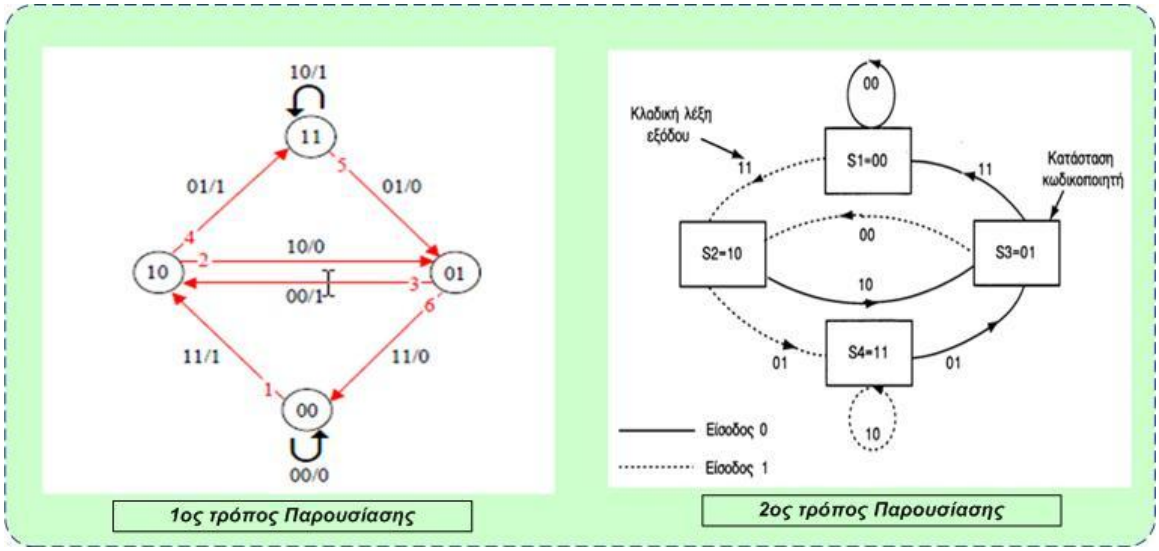
2.2.3 Διάγραμμα Καταστάσεων (State Diagram)

Ένας Συνελικτικός κωδικοποιητής ανήκει στην κατηγορία των Μηχανών Πεπερασμένων Καταστάσεων, άρα μπορεί και να παρασταθεί με τη βοήθεια ενός διαγράμματος μηχανής πεπερασμένων καταστάσεων. Έστω ένας $n,1,K$ Συνελικτικός κώδικας. Ως κατάσταση του κωδικοποιητή, σε κάθε δεδομένη χρονική στιγμή, ορίζεται το περιεχόμενο των K-1 δεξιότερων σταδίων του καταχωρητή ολίσθησης. Συνεπώς, για

ένα δυαδικό Συνελικτικό κώδικα υπάρχουν 2^{K-1} καταστάσεις. Η γνώση της παρούσας κατάστασης του κωδικοποιητή και του ψηφίου εισόδου, αρκεί για να καθοριστεί η έξοδος του συνελικτικού κωδικοποιητή, καθώς και η επόμενη κατάστασή του.

Στο παρακάτω Σχήμα 2.4 παρουσιάζεται το διάγραμμα καταστάσεων, με δύο διαφορετικούς τρόπους :

Στον πρώτο τρόπο παρουσίασης του αριστερού διαγράμματος καταστάσεων κάθε κύκλος συμβολίζει και μια κατάσταση. Σε οποιαδήποτε τυχαία χρονική στιγμή, ο κωδικοποιητής βρίσκεται αναγκαστικά σε μία από αυτές τις καταστάσεις. Σε κάθε μετάβαση, από μία κατάσταση στην επόμενη (βέλος), εικονίζονται δύο νούμερα. Το δεύτερο συμβολίζει το δυαδικό ψηφίο εισόδου, που χρειάζεται ο κωδικοποιητής για να μεταβεί σε επόμενη κατάσταση και το πρώτο συμβολίζει την έξοδο που θα παραχθεί [7] .



Σχήμα 2.4 Τυπικό διάγραμμα καταστάσεων κωδικοποίησης Συνελικτικού Κώδικα ρυθμού $\frac{1}{2}$ (n,k,m)=(2,1,3)

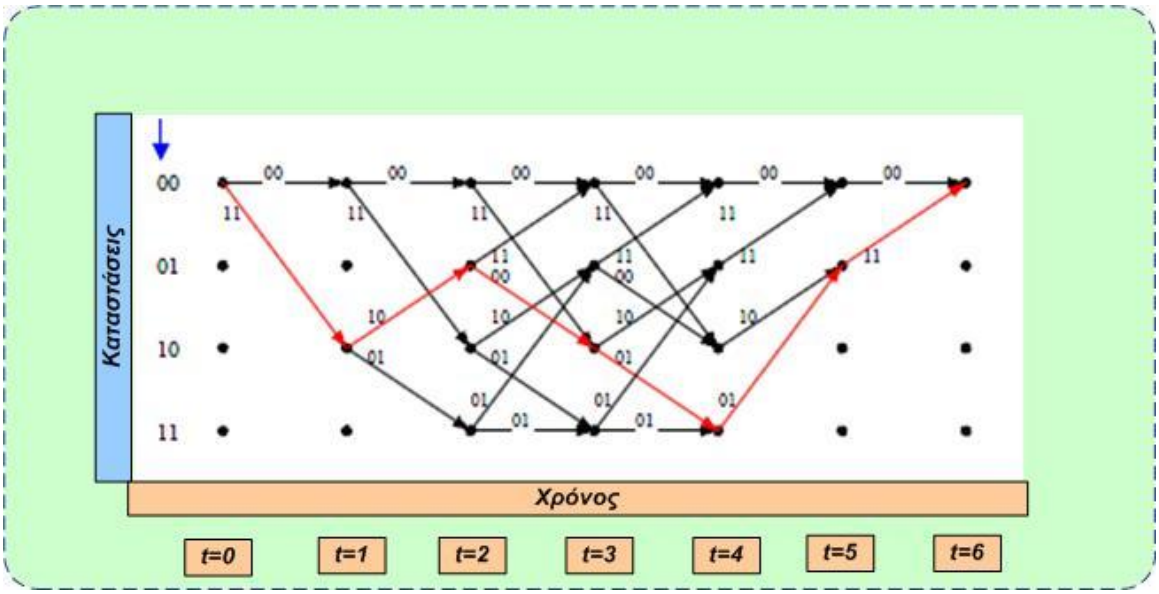
Στο δεύτερο τρόπο παρουσίασης, οι μεταβάσεις μεταξύ των καταστάσεων παρουσιάζονται στο δεξιό διάγραμμα καταστάσεων, με βέλη από την τωρινή στην επόμενη κατάσταση, δίπλα στα οποία αναγράφεται και η έξοδος (n ψηφίων), που δίνει ο κωδικοποιητής κατά τη μετάβαση αυτή. Συνήθης πρακτική είναι η μετάβαση από μία κατάσταση στην επόμενη, όταν το ψηφίο εισόδου είναι 1 να παριστάνεται με διακεκομμένη γραμμή, ενώ στην περίπτωση μηδενικής εισόδου, η γραμμή είναι συνεχής.

Στο ανωτέρω σχήμα παρουσιάζεται ένας $2,1,3$ Συνελικτικός κωδικοποιητής, ο οποίος σύμφωνα και με τα προηγούμενα έχει $2^{3-1}=4$ καταστάσεις, δύο πιθανές μεταβάσεις από κάθε κατάσταση, ανάλογα με το πλήθος των δυνατών εισόδων ($k=1\Rightarrow 2^k=2$) και έξοδο μήκους $n=2$ δυαδικών ψηφίων.

Τέλος θα πρέπει να αναφέρουμε ότι, ένα βασικό μειονέκτημα του Διαγράμματος Καταστάσεων είναι ότι δεν απεικονίζει το χρόνο. Δεν υπάρχει δυνατότητα αποθήκευσης χρονικών στιγμών, έτσι ώστε να γίνεται γνωστό π.χ. τη χρονική στιγμή 5 σε ποια κατάσταση βρισκόταν ο κωδικοποιητής. Συγκρίνοντας το Διάγραμμα Καταστάσεων, με τον πίνακα αναφοράς του κωδικοποιητή, μπορεί να παρατηρηθεί ότι περιέχουν τις ίδιες ακριβώς πληροφορίες, με τη διαφορά ότι το Διάγραμμα Καταστάσεων τις αναπαριστά γραφικά.

2.2.4 Διάγραμμα Trellis (Trellis Diagram)

Άλλος τρόπος παρουσίασης είναι το Διάγραμμα Trellis, του οποίου το μεγάλο πλεονέκτημα είναι η προσθήκη της διάστασης του χρόνου, αλλά χωρίς τα προβλήματα που δημιουργεί το Δεντρικό Διάγραμμα. Τοποθετώντας τις δυνατές καταστάσεις του συστήματος, τη μία κάτω από την άλλη, στον άξονα y, και τις διακριτές χρονικές στιγμές ξεκινώντας από το μηδέν, στον άξονα x, δημιουργείται το Διάγραμμα Trellis, που φαίνεται στο Σχήμα 2.5. Με το πέρασμα του χρόνου, γίνεται μετακίνηση προς τα δεξιά στο διάγραμμα. Κάθε μετάβαση σημαίνει την έλευση και ενός καινούριου δυαδικού ψηφίου.



Σχήμα 2.5 Διάγραμμα Trellis

2.2.5 Γεννήτορας Πίνακας Συνελικτικού Κώδικα

Ένας άλλος εναλλακτικός τρόπος είναι η περιγραφή της διαδικασίας με πολυώνυμα γεννήτορες, όπως ακριβώς και στους Κώδικες Block. Για το παράδειγμά μας υπάρχουν δυο τέτοια πολυώνυμα που προκύπτουν από τις XOR εξόδους του κωδικοποιητή, τα $g_1(x) = 1 + x + x^2$ και $g_2(x) = 1 + x^2$. Αν παριστάνουμε και την ακολουθία εισόδου 1011 υπό την μορφή πολυωνύμου $I(x) = 1 + x^2 + x^3$ και στην συνέχεια με την πράξη του πολλαπλασιασμού προκύπτουν τα $c_1(x) = I(x) * g_1(x)$ και $c_2(x) = I(x) * g_2(x)$. Για το παράδειγμά μας προκύπτουν τα παρακάτω:

$$c_1(x) = I(x) * g_1(x) = (1 + x^2 + x^3) * (1 + x + x^2)$$
$$= 1 + x + x^2 + x^3 + x^4 + x^3 + x^4 + x^5 =$$

$=1 + x + 2x^2 + 2x^3 + 2x^4 + x^5$ → 11 00 01

$$c_2(x) = I(x) * g_2(x) = (1 + x^2 + x^3) * (1 + x^2) = 1 + x^2 + x^2 + x^4 + x^3 + x^5 =$$

$=1 + 2x^2 + x^3 + x^4 + x^5$ → 10 01 11

→ Έξοδος του Κωδικοποιητή ***11 10 00 01 01 11***

Διαφορετικός τρόπος παρουσίασης θα ήταν, αν αντικαταστήσουμε τα πολυώνυμα με το Γεννήτορα πίνακα. Γράφουμε όλα τα πολυώνυμα στη δυαδική τους μορφή και για την κατασκευή του πίνακα χρησιμοποιούμε την Κρουστική Απόκριση του κωδικοποιητή που είναι η ***11 10 11.*** Ο Γεννήτορας πίνακας προκύπτει, αφού γράψουμε στην πρώτη γραμμή την Κρουστική Απόκριση και στη συνέχεια οι επόμενες προκύπτουν με δεξιά ολίσθηση, η μια κάτω από την άλλη, κι έτσι προκύπτει η παρακάτω μορφή του γεννήτορα πίνακα [7] :

$$G = \begin{pmatrix} 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 \end{pmatrix}$$

και στη συνέχεια από τον τύπο $C = I * G$

$$C = (1011) * \begin{pmatrix} 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 \end{pmatrix} = (11\ 10\ 00\ 01\ 01\ 11)$$

2.3 Αλγόριθμοι Αποκωδικοποίησης Συνελικτικών Κωδίκων

Για την αποκωδικοποίηση των Συνελικτικών κωδίκων υπάρχουν διάφορες τεχνικές, που ομαδοποιούνται σε δύο βασικές κατηγορίες:

- Ακολουθιακή Αποκωδικοποίηση
    - ο Αλγόριθμος του Fano
- Αποκωδικοποίηση Μέγιστης Πιθανοφάνειας
    - ο Αλγόριθμος Viterbi

Και οι δύο κατηγορίες αποκωδικοποίησης εκπροσωπούν δύο διαφορετικές προσεγγίσεις της ίδιας, όμως, ιδέας. Για την καλύτερη κατανόηση της αποκωδικοποίησης, θα χρησιμοποιηθεί και πάλι ο κώδικας (2,1,3) του Σχήματος 2.2. Ας θεωρηθεί ότι πρέπει να γίνει αποστολή τεσσάρων δυαδικών ψηφίων, με τη χρήση του παραπάνω κώδικα. Η κωδικοποίηση θα μετατρέψει τα τέσσερα δυαδικά ψηφία σε οχτώ, τα οποία και θα σταλούν στο δέκτη μέσω του καναλιού. Η μετατροπή των τεσσάρων δυαδικών ψηφίων

σε οκτώ είναι μοναδική και έτσι κάθε ακολουθία εισόδου 4 δυαδικών ψηφίων, θα έχει μοναδική κωδικοποιημένη ακολουθία οκτώ δυαδικών ψηφίων. Ωστόσο, λόγω της μη ύπαρξης ιδανικού καναλιού, μπορεί να προκύψουν λάθη κατά τη μετάδοση και έτσι ο δέκτης να λάβει οποιονδήποτε δυνατό συνδυασμό των οκτώ δυαδικών ψηφίων. Όλοι οι δυνατοί συνδυασμοί των οκτώ δυαδικών ψηφίων μας δίνουν $2^8 = 256$ δυνατές ακολουθίες. Τα τέσσερα δυαδικά ψηφία πριν την κωδικοποίηση, έχουν ως αποτέλεσμα την ύπαρξη δέκα έξι διαφορετικών ακολουθιών εισόδου, οι οποίες αντιστοιχούν και σε δεκαέξι κωδικοποιημένες ακολουθίες, των οκτώ πλέον δυαδικών ψηφίων. Στόχος του αποκωδικοποιητή είναι να αποφανθεί, ποια από τις δεκαέξι ακολουθίες εισόδου έχει σταλεί μέσα από ένα σύνολο 256 δυνατών ακολουθιών. Έστω ότι ο δέκτης λαμβάνει μια ακολουθία, η οποία δεν είναι μια από τις έγκυρες ακολουθίες δηλ. είναι εσφαλμένη. Άρα, πρωταρχικός στόχος των μεθόδων αποκωδικοποίησης είναι η ανίχνευση των λαθών, και αν είναι εφικτό, η διόρθωση όσο το δυνατόν περισσοτέρων από αυτά.

Για την αποκωδικοποίηση μπορούμε να χρησιμοποιήσουμε τον παρακάτω τρόπο:

Μπορεί να γίνει σύγκριση της ληφθείσης ακολουθίας, με όλες τις επιτρεπτές δυνατές ακολουθίες και να επιλεγεί αυτή, με τη μικρότερη απόσταση Hamming (ή λιγότερη ασυμφωνία δυαδικών ψηφίων). Σε περίπτωση που ακολουθεί κάποιος την παραπάνω μέθοδο κωδικοποίησης πολλές φορές μπορεί να παρατηρηθεί ότι δυο ή περισσότερες ισοπίθανες δυνατές επιλογές, οι οποίες θα διαφέρουν μόνο κατά ένα ψηφίο από μια έγκυρη ακολουθία. Μεγάλο μειονέκτημα της μεθόδου αυτής είναι ότι, δεν υπάρχει τρόπος να καταλάβει κάποιος ποια από τις παραπάνω ακολουθίες μεταδόθηκε. Παράλληλα η αύξηση των δυαδικών ψηφίων προς κωδικοποίηση και αποκωδικοποίηση, προκαλεί εκθετική αύξηση της υπολογιστικής ισχύος. Συμπεραίνουμε λοιπόν από τα παραπάνω ότι η μέθοδος αυτή που περιγράψαμε, είναι πρακτικά μη υλοποιήσιμη.

Θα πρέπει να χρησιμοποιηθεί μια μέθοδος, που να μην εξετάζει όλες τις δυνατές περιπτώσεις και μάλιστα, να μπορεί να αντιμετωπίζει και τις περιπτώσεις των ισοβαθμιών. Ένας από τους σημαντικότερους αλγορίθμους που ανήκει στην κατηγορία Αποκωδικοποίηση Μέγιστης Πιθανοφάνειας, είναι ο Αλγόριθμος Viterbi και βελτιώσεις αυτού, όπως ο Αλγόριθμος SOVA.

Η πολυπλοκότητα του Αλγορίθμου Viterbi είναι ανάλογη του πλήθους καταστάσεων του διαγράμματος Trellis. Αυτό σημαίνει ότι η πολυπλοκότητα αυξάνει εκθετικά, με το μήκος –εξαναγκασμού των Συνελικτικών Κωδίκων. Οπότε ο αλγόριθμος Viterbi μας

βολεύει σε κώδικες με μικρό μήκος εξαναγκασμού. Αξίζει να αναφέρουμε ότι για κώδικες με μεγαλύτερα μήκη εξαναγκασμού, είχαν προταθεί στο παρελθόν άλλοι σχεδόν –βέλτιστοι τύποι αποκωδικοποίησης όπως :

- Ακολουθιακός Αποκωδικοποιητής Wozencraft 1957
- Αλγόριθμος Fano 1963
- Αλγόριθμος «Στοίβας (stack)» Zigangirov 1966 και Jelinec 1969
- Αλγόριθμος Αποκωδικοποίησης – Ανάδρασης Heller 1975
- Αλγόριθμος Πλειοψηφικής Λογικής Massey 1963

2.3.1 Αλγόριθμος Viterbi

Ο αλγόριθμος Viterbi, αναπτύχθηκε από τον Andrew Viterbi το 1967, σε μια πρωτοποριακή για την εποχή του εργασία, που δημοσιεύτηκε τον Απρίλιο του 1967. Προτάθηκε σαν ένα σύστημα διόρθωσης λαθών (error-correction coding), στις ψηφιακές επικοινωνιακές ζεύξεις με θόρυβο. Έχει χρησιμοποιηθεί κυρίως στην αποκωδικοποίηση των Συνελικτικών κωδίκων σε συστήματα τηλεπικοινωνιών, όπως CDMA και GSM digital cellular, dial-up modems, σε δορυφορικές επικοινωνίες deep-space communications, και 802.11 wireless LANs. Η βασική αρχή αυτού του αλγορίθμου προκύπτει από την παρατήρηση ότι, εάν η βέλτιστη διαδρομή, μεταξύ του σημείου Α και του σημείου Γ περνά από ένα ενδιάμεσο σημείο Β, τότε το τμήμα αυτής της διαδρομής μεταξύ του σημείου Α και του σημείου Β είναι η βέλτιστη διαδρομή μεταξύ αυτών των δύο σημείων. Γενικά όταν ο αλγόριθμος Viterbi υπολογίζει το βέλτιστο μονοπάτι σε ένα trellis, χωρίς να είναι γνωστό εκ των προτέρων ποιοί κόμβοι θα αποτελούν το τελικό βέλτιστο μονοπάτι. Για το λόγο αυτό υπολογίζεται σε κάθε στιγμή το optimum path για κάθε κόμβο. Έτσι σε κάθε χρονική και για κάθε κόμβο διατηρείται ένα survivor path. Θα πρέπει να σημειωθεί ότι ο Viterbi συνεισφέρει στη μείωση του υπολογιστικού φόρτου, γεγονός που απορρέει από την χρήση των trellis.

2.3.1.1 Βήματα Αλγορίθμου Viterbi

Ο Αλγόριθμος μπορεί να περιγραφεί με τα παρακάτω βήματα :

Βήμα 1. Ο αλγόριθμος υπολογίζει το path metric για κάθε path σε όλους τους κόμβους τη χρονική στιγμή k, προσθέτοντας το metric του survivor path προς τους κόμβους αυτούς τη χρονική στιγμή k-1 στο branch weight που έχει ο κάθε κόμβος τη χρονική στιγμή k.

Βήμα 2. Έπειτα για κάθε path προς τους κόμβους τη χρονική στιγμή k, κρατάει το path με το καλύτερο metric και το ανάγει στο survivor path.

Βήμα 3. Στη συνέχεια αποθηκεύει το survivor path και το metric του για κάθε κόμβο στη χρονική στιγμή k.

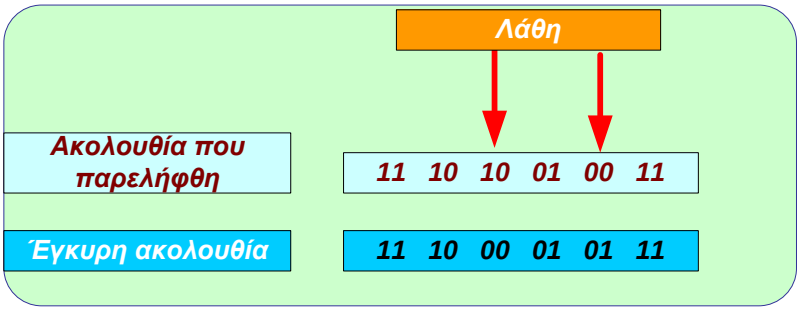
Βήμα 4. Τέλος αυξάνει το k και επαναλαμβάνει το βήμα 1.

Σε πολλές περιπτώσεις εύρεσης του shortest path στα trellises, ο αριθμός των branches που χρειάζονται μπορεί να είναι απεριόριστα μεγάλος. Στην εφαρμογή του αλγορίθμου, καταφεύγουμε σε κάποιες τεχνικές, ώστε να παραχθούν τα outputs πριν από το τέλος της λαμβανόμενης ακολουθίας, δεδομένου ότι η ακολουθία που λαμβάνεται μπορεί να συνεχιστεί κατά τρόπο αόριστο. Μία συνηθισμένη προσέγγιση στο πρόβλημα αυτό είναι να διατηρούνται όλα τα paths του trellis σε κάποιο παράθυρο με περιορισμένο μήκος. Όταν το μήκος του path γίνει ίσο με το μήκος του παραθύρου, επιλέγεται το state με το μικρότερο κόστος. Το path που οδηγεί σε αυτό το state ανιχνεύεται και το πρώτο branch στο παράθυρο επιλέγεται ως έξοδος του αλγόριθμου. Στην επόμενη χρονική στιγμή το παράθυρο ολισθαίνει κατά ένα και τα paths στο τέλος του παραθύρου, επεκτείνονται και επιλέγεται ξανά το path με το μικρότερο κόστος, για να παραχθεί πάλι η επόμενη έξοδος. Η ίδια διαδικασία συνεχίζεται μέχρι να παραχθεί όλη η ακολουθία των outputs. Αν το παράθυρο έχει επιλεχθεί να είναι αρκετά μεγάλο, αυτή η προσέγγιση σχεδόν πάντα παράγει το optimum path. Ωστόσο, εφόσον το optimum path καθορίζεται από την αρχή

μέχρι το τέλος της ακολουθίας, είναι πιθανόν η μέθοδος του Viterbi με τη χρήση παραθύρων να οδηγήσει σε λάθος αποτέλεσμα. Αυτό όμως συμβαίνει περιστασιακά [8].

2.3.1.2 Σχηματική Αναπαράσταση του Αλγορίθμου Viterbi

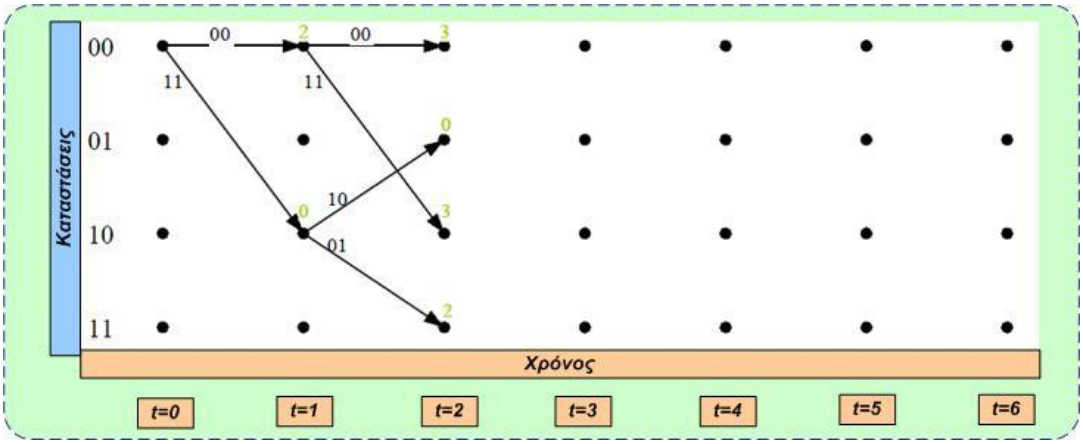
Η κατανόηση της διεργασίας της αποκωδικοποίησης μπορεί να γίνει ευκολότερα, αν επεκτείνουμε τα διάγραμμα καταστάσεων του Σχήματος 2.4, στο Διάγραμμα Trellis. Για καλύτερη κατανόηση του αλγορίθμου θα τον παρουσιάσουμε γραφικά με τα Διαγράμματα Trellis. Έστω ότι για το παράδειγμά μας ισχύουν τα παρακάτω και πιο συγκεκριμένα, έχουν παραληφθεί δυο λάθη που δείχνουν τα κόκκινα βέλη [7].



Σχήμα 2.6 Απεικόνιση Λαθών του παραδείγματός μας

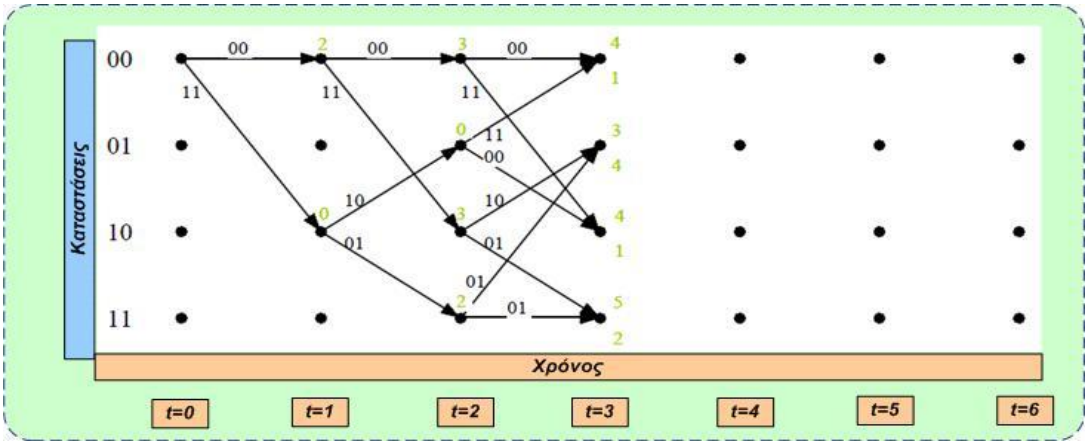
Ο Αλγόριθμος Viterbi ξεκινάει την αποκωδικοποίηση, για να βρει τη maximum likelihood Sequence. Για να υπολογίσει το branch weight χρησιμοποιεί την απόσταση Hamming των συμβόλων που έλαβε, από αυτά που θα περίμενε. Θεωρούμε ότι το κύκλωμα ξεκινά από την Κατάσταση 00. Το πρώτο λαμβανόμενο σύμβολο είναι το 11 και ο αλγόριθμος υπολογίζει αποστάσεις Hamming από αυτό το σύμβολο και τα αναμενόμενα από την κατάσταση 00 σύμβολα και αυτό με βάση τις καταστάσεις του Σχήματος 2.6, από το οποίο προκύπτει ότι από την κατάσταση 00 μπορούμε να μεταβούμε στο 00 με εκπεμπόμενο σύμβολο το 00 και στην κατάσταση 10 με εκπεμπόμενο σύμβολο το 11. Οι αποστάσεις Hamming φαίνονται στο Σχήμα 2.7 και είναι 2 και 0 αντίστοιχα και αναγράφονται. Στη συνέχεια εισάγεται το σύμβολο 10 και υπολογίζεται η απόσταση Hamming, για όλες τις μεταβάσεις κάθε εισερχόμενου κόμβου στο Trellis. Προς το παρόν βλέπουμε ότι το transition από την κατάσταση 00 στο 10 και

από εκεί στη 01 έχουν παραχθεί λάθη σε bits με μηδενική απόσταση και φαίνεται να είναι το καλύτερο path προσωρινά .

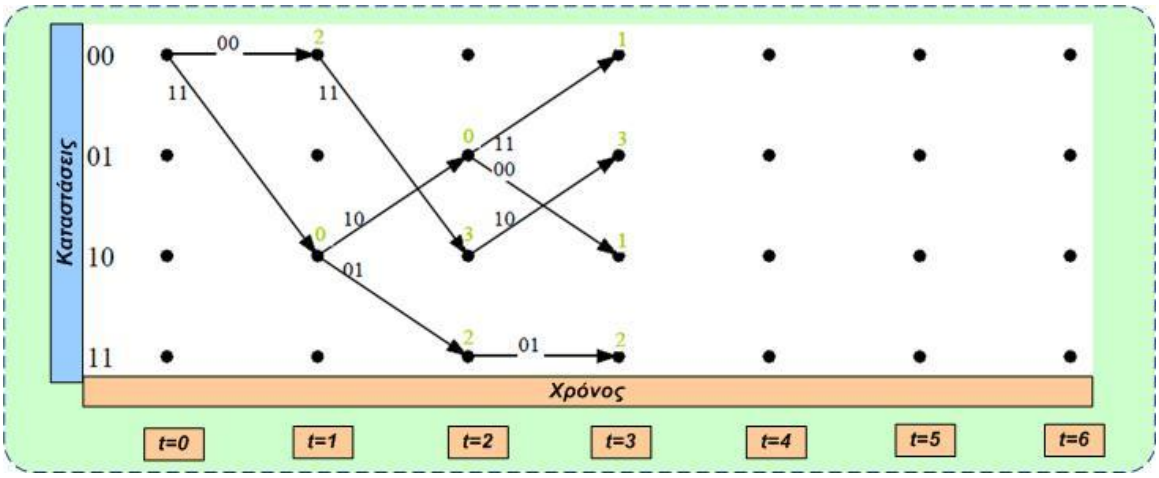


Σχήμα 2.7 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 1 από 8

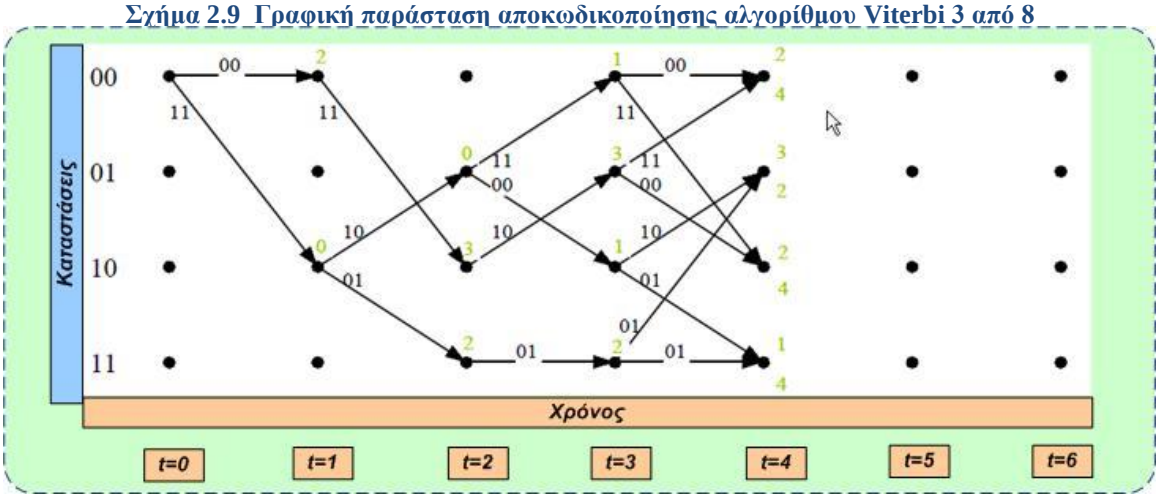
Για κάθε σύμβολο που εισάγεται, υπολογίζονται τα paths σε κάθε κατάσταση και η απόσταση Hamming με ίδιο τρόπο όπως προηγουμένως. Αν σε κάποια κατάσταση φθάνουν δυο path, την ίδια στιγμή επιλέγεται αυτό με τη μικρότερη Hamming Distance. Στα επόμενα σχήματα παρουσιάζεται η εξέλιξη του Trellis, με σκοπό να βρεθεί το optimum path που θα οδηγήσει στη maximum likelihood Sequence.



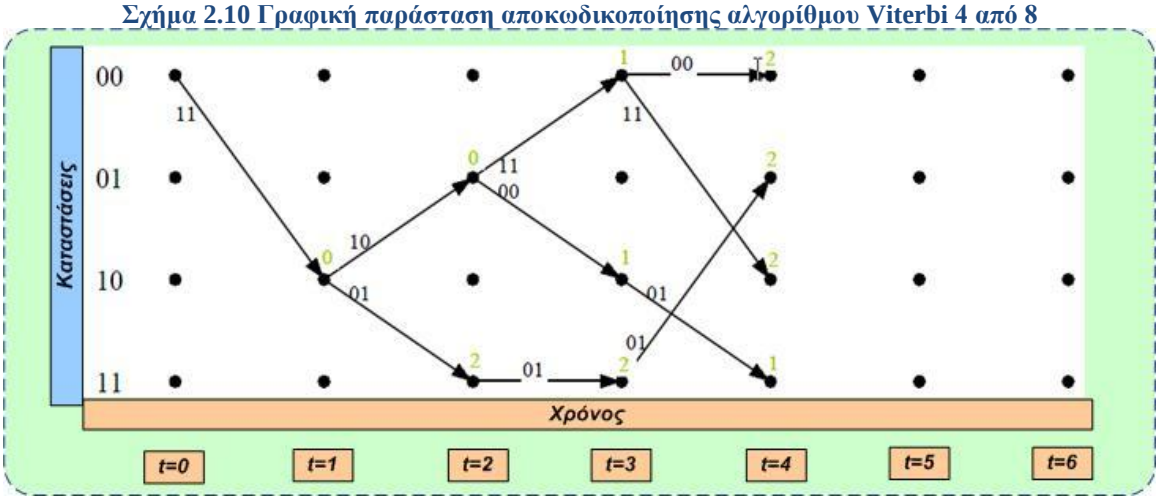
Σχήμα 2.8 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 2 από 8



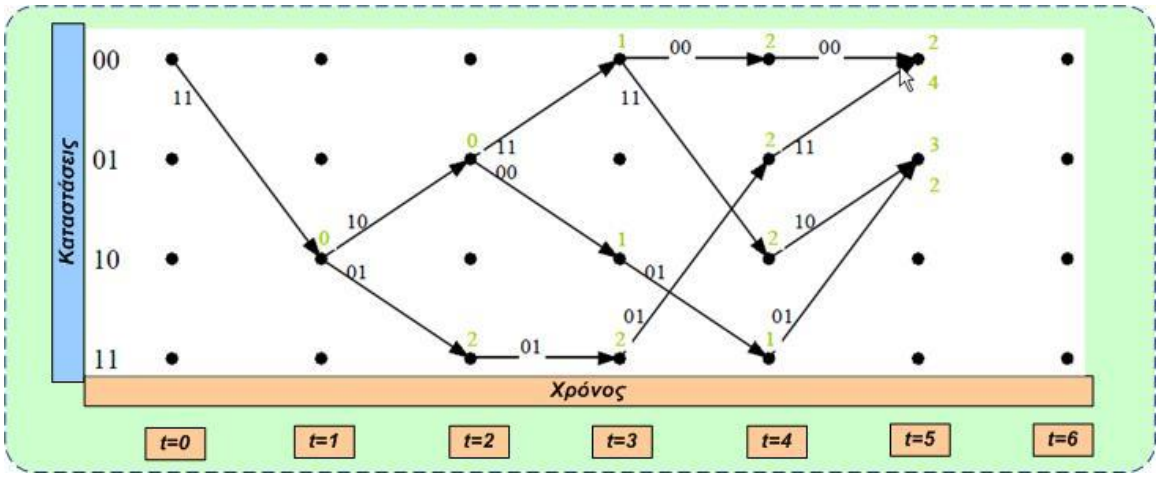
Σχήμα 2.9 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 3 από 8



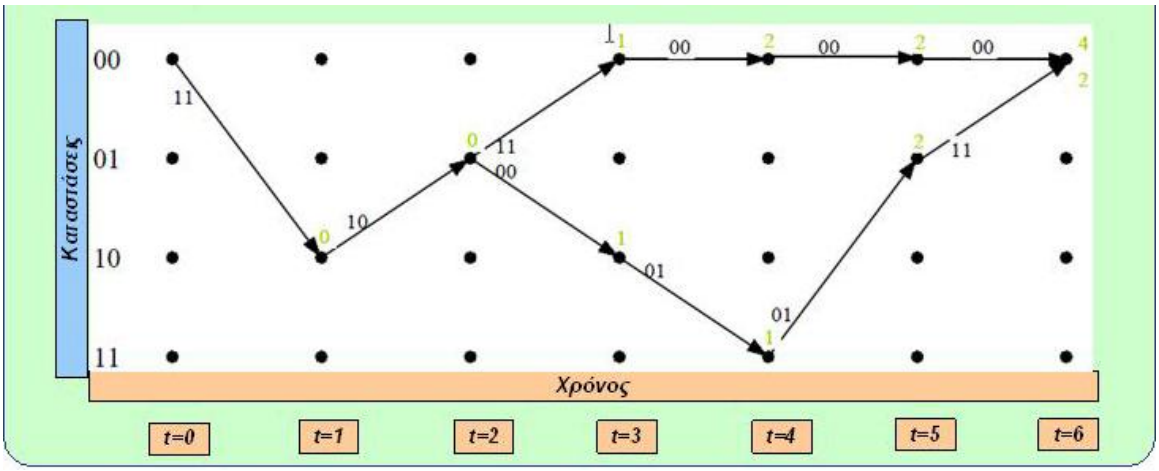
Σχήμα 2.10 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 4 από 8



Σχήμα 2.11 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 5 από 8

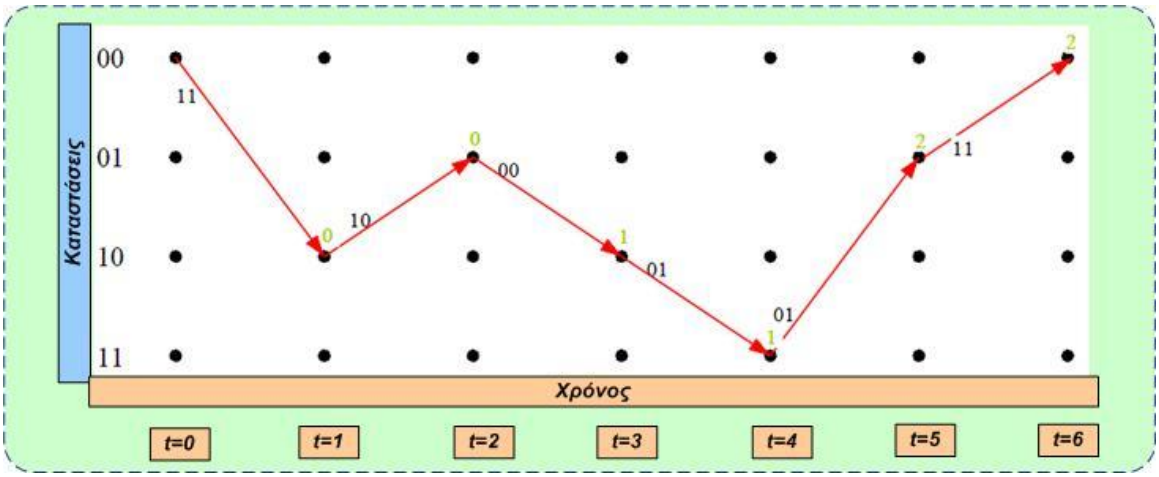


Σχήμα 2.12 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 6 από 8



Σχήμα 2.13 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 7 από 8

Στο τέλος του Trellis επιλέγεται το shortest path, αυτό δηλαδή που έχει τη μικρότερη Hamming Distance, στο παράδειγμά μας την απόσταση 2. Αυτό το optimum path στη συνέχεια γίνεται trace back, για να ανακτηθεί η αρχική ακολουθία συμβόλων που εκπέμφθηκε.



Σχήμα 2.14 Γραφική παράσταση αποκωδικοποίησης αλγορίθμου Viterbi 8 από 8

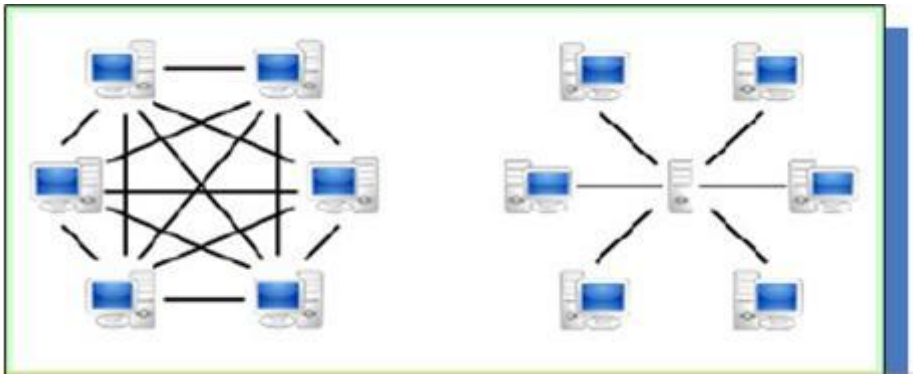
Το optimum path καταδεικνύεται από την κόκκινη γραμμή του Σχήματος 2.14. Και η απόσταση Hamming του shortest path είναι 2, γεγονός που σημαίνει ότι διορθώθηκαν δύο λάθη. Από την ακολουθία των συμβόλων που μεταδόθηκαν, ο αποκωδικοποιητής είναι δυνατόν να λάβει το αρχικό μήνυμα συγκρίνοντας τα λαμβανόμενα σύμβολα που απορρέουν από το Trellis, ως το maximum likelihood Sequence, με τα transition state του Συνελικτικού Κωδικοποιητή. Στο συγκεκριμένο παράδειγμα που μελετήσαμε πιο πάνω προκύπτει η ακολουθία του optimum path είναι η ***11 10 00 01 01 11***.

3. Επιδημικοί Αλγόριθμοι

Στις μέρες μας η τεχνολογία των κατανεμημένων συστημάτων, όσον αφορά τα φυσικά μέσα, τις τεχνικές μετάδοσης και τα πρωτόκολλα επικοινωνίας σε μεταβαλλόμενης τοπολογίας δίκτυα, παρουσιάζει ραγδαία εξέλιξη. Η κινητή ασύρματη επικοινωνία δεδομένων, η οποία εξελίσσεται και τεχνολογικά αλλά και στη χρήση/διάδοσή της, είναι μια κινητήρια δύναμη χάρη στο Internet και την επιτυχία της τρίτης γενιάς των κινητών δικτύων με κυψέλες. Στο άμεσο μέλλον, ο ρόλος και οι δυνατότητες της ανταλλαγής δεδομένων σε κοντινή απόσταση αναμένεται να μεγαλώσουν, και να εξυπηρετούν σαν συμπλήρωμα στην παραδοσιακή μεγάλης κλίμακας επικοινωνία. Οι περισσότερες επικοινωνίες, μεταξύ ανθρώπου και μηχανής καθώς κι η προφορική επικοινωνία μεταξύ ανθρώπων, συμβαίνουν σε αποστάσεις μικρότερες των 10 μέτρων. Η πιο διαδεδομένη αντίληψη για ένα κινητό **ad-hoc** δίκτυο είναι ένα δίκτυο σχηματισμένο χωρίς καμία κεντρική διαχείριση που αποτελείται από κινούμενους κόμβους που χρησιμοποιούν ένα ασύρματο μέσο για να στέλνουν πακέτα δεδομένων. Αυτοί οι κόμβοι σε ένα δίκτυο τέτοιου είδους μπορούν να λειτουργήσουν και σαν δρομολογητές (routers) αλλά και σαν εξυπηρετητές (servers), μπορούν να προωθούν πακέτα εκ μέρους άλλων κόμβων και να τρέχουν εφαρμογές χρηστών. Τα δίκτυα Mobile Ad-Hoc Networks (**MANET**) αποτελούν μια επέκταση των δικτύων Ad-Hoc, αφού αποτελούνται από κόμβους που κινούνται δυναμικά και αυθαίρετα, δημιουργώντας έτσι δυναμικές τοπολογίες δικτύου. Στο Κεφάλαιο αυτό θα κάνουμε μια εισαγωγή στα peer to peer συστήματα, μιας και είναι αυτά τα συστήματα που επικρατούν όλο και περισσότερο σε σχέση με τα συστήματα Client-Server που σιγά-σιγά εκλείπουν. Στην περαιτέρω πορεία αυτής της διπλωματικής θα παρουσιάσουμε τους Επιδημικούς Αλγορίθμους και θα εστιάσουμε το ενδιαφέρον μας σε ορισμένους από αυτούς, οι οποίοι πρόσφατα έχουν αποκτήσει μεγάλη δημοτικότητα, μιας και ανήκουν σε κατηγορία εξελιγμένων και αποτελεσματικών αλγορίθμων που αποτελούν έναν εξελικτικό και αξιόπιστο τρόπο για τη μεταφορά πληροφοριών σε έναν παραλήπτη ή ομάδα παραληπτών, σε περιβάλλοντα δικτύων υπολογιστών ή κινητών τερματικών κόμβων .

3.1 Peer –to-Peer (p2p) Συστήματα

Ένα Κατανεμημένο Σύστημα αποτελείται από γεωγραφικά ανεξάρτητες, αυτόνομες υπολογιστικές συσκευές, που επικοινωνούν μεταξύ τους και λειτουργούν συντονισμένα για την επίτευξη ενός κοινού στόχου. Σε αντίθεση με τα κεντρικοποιημένα συστήματα, ο σχεδιασμός των κατανεμημένων συστημάτων, η κατανόηση της λειτουργίας τους υπό ιδιαίτερες συνθήκες, καθώς και η ανάλυση της συμπεριφοράς τους, απαιτεί ειδικές γνώσεις και ικανότητες.



Σχήμα 3.1 Παρουσίαση μοντέλων Peer-to-Peer και Server-Client

Τη βασική κατηγορία των κατανεμημένων συστημάτων αποτελούν τα συστήματα ομοτίμων κόμβων (Peer-to-Peer ή P2P συστήματα). Ως βασικό χαρακτηριστικό των P2P συστημάτων είναι η ισοδυναμία των κόμβων ως προς τις υποχρεώσεις τους και τις δυνατότητές τους, δηλ. οι κόμβοι συμμετέχουν ισότιμα (όσο γίνεται) στο δίκτυο, εν αντιθέσει με τα κεντρικοποιημένα συστήματα, στα οποία είναι απαραίτητη η παρουσία ενός κεντρικού κόμβου (του Server) για να επιτευχθεί η επικοινωνία μεταξύ των κόμβων του δικτύου. Τα συστήματα P2P έχουν γίνει θέμα μεγάλης ερευνητικής δραστηριότητας τα τελευταία χρόνια με σκοπό κυρίως την αποδοτικότητα, την επεκτασιμότητα και την ανοχή σφαλμάτων που μπορούν να ανεχθούν. Επίσης, παρουσιάζονται όλο και πιο πολύπλοκα συστήματα, αλλά με καλύτερες ιδιότητες από τα προηγούμενα.

3.1.1 Εισαγωγή στα Peer - to - Peer (p2p) Συστήματα

Τα δίκτυα ομότιμων κόμβων, αντίθετα με τις παραδοσιακές εφαρμογές πελάτη-εξυπηρετητή (Client-Server), είναι κατανεμημένα συστήματα ομότιμων κόμβων τα οποία χρησιμοποιούν την υποδομή και τους πόρους του διαδικτύου για την επικοινωνία μεταξύ

των κόμβων. Η επικοινωνία μεταξύ τους γίνεται πάνω από το φυσικό TCP/IP δίκτυο, χωρίς να απαιτείται κάποιος κεντρικός έλεγχος. Τα συστήματα αυτά χρησιμοποιούνται κυρίως για διαμοιρασμό διαφόρων αρχείων και δεδομένων μεταξύ των χρηστών. Η άποψη ότι τέτοια συστήματα, πέρα από το διαμοιρασμό αρχείων δεν έχουν να προσφέρουν κάτι παραπάνω, είναι μάλλον λανθασμένη. Η ιδέα ενός κατανεμημένου συστήματος είναι αρκετά ελκυστική σε πολλούς τομείς της πληροφορικής, όπως οι βάσεις δεδομένων (κατανεμημένες Β.Δ.). Η μελέτη λοιπόν των συστημάτων ομότιμων κόμβων θα δώσει τη δυνατότητα επέκτασης και εξέλιξης των συστημάτων, έτσι θα υπάρχει η δυνατότητα με ένα σύστημα ομότιμων κόμβων να καταφέρει κανείς και άλλα πράγματα εκτός από το διαμοιρασμό αρχείων. Το κύριο χαρακτηριστικό των κόμβων αυτών είναι το γεγονός ότι είναι ομότιμοι, δηλαδή κάθε κόμβος συμμετέχει ισάξια στο σύστημα, έχει τα ίδια δικαιώματα και τις ίδιες υποχρεώσεις με τους υπόλοιπους.

Στις αρχές του 1999 ο Σον Φάνινγκ (Shawn Fanning) ξεκίνησε την υλοποίηση μιας ιδέας, η οποία θα του έδινε τη δυνατότητα αυτός και οι φίλοι του να αναζητήσουν στο Διαδίκτυο μουσικά κομμάτια MP3 της προτίμησής τους. Μερικούς μήνες αργότερα, η Napster μετρούσε πάνω από 21 εκατομμύρια χρήστες. Σε καμία περίπτωση, όμως, ο 18χρονος τότε μαθητής δεν μπορούσε να φανταστεί ότι το δημιούργημά του θα άλλαζε τον τρόπο με τον οποίο απολαμβάνουμε πολυμεσικές εφαρμογές και γενικά να επικοινωνούμε και λόγω της μεγάλης απήχησής του ακολούθησαν και άλλες ευρέως διαδεδομένες εφαρμογές όπως είναι το Gnutella, KaZaA, BitTorrent κ.ά.. **[1]** Τα προαναφερόμενα δίκτυα αποτελούν ένα μεγάλο μέρος της κίνησης του διαδικτύου, για αυτό και κέντρισαν το ενδιαφέρον της ερευνητικής κοινότητας εδώ και κάποια χρόνια.

Γενικά υπάρχουν διάφορες αρχιτεκτονικές για δίκτυα ομότιμων κόμβων, τα κεντρικοποιημένα και τα μη κεντρικοποιημένα, τα οποία χωρίζονται σε δομημένα και αδόμητα.

3.1.2 Γενικά χαρακτηριστικά των συστημάτων ομότιμων κόμβων

Θα δούμε κάποιες ιδιότητες οι οποίες είναι αντιπροσωπευτικές και δίνουν μια πιο πλήρη εικόνα για τα συστήματα αυτά και το πώς λειτουργούν.

I. **Δεν υπάρχει κεντρικός συντονισμός:** Κάθε κόμβος λειτουργεί σε σχέση με την επικοινωνία που αναπτύσσει με τους γείτονές του, καθώς μόνο αυτούς γνωρίζει.

Δεν υπάρχει τρόπος να μιλήσει κάποιος άμεσα με όλο το δίκτυο και δε γνωρίζει τι γίνεται σε μια περιοχή του δικτύου μακριά του. Κάθε χρήστης λειτουργεί ως πελάτης (client) αλλά και ως εξυπηρετητής (server).

II. **Όλη η πληροφορία στο δίκτυο είναι προσβάσιμη από κάθε κόμβο:** Ο τρόπος που επικοινωνεί κάθε κόμβος, αν και περιορισμένος, είναι ικανός να προσφέρει πρόσβαση σε όλα τα δεδομένα του δικτύου.

III. **Κάθε κόμβος είναι αυτόνομος:** Κάθε κόμβος λειτουργεί ανεξάρτητα από τους υπόλοιπους. Μπορεί να μπει στο δίκτυο και να αποχωρήσει από το δίκτυο όποια στιγμή θελήσει, χωρίς να πρέπει να περιμένει κάποιον άλλον κόμβο (π.χ. ολοκλήρωση μεταφοράς αρχείου).

IV. **Κάθε κόμβος θεωρείται αναξιόπιστος:** Κάθε κόμβος μπορεί να μπει και να αφήσει το δίκτυο οποιαδήποτε στιγμή, κι αυτό τον καθιστά αναξιόπιστο. Εφόσον όλοι οι κόμβοι έχουν την ίδια συμπεριφορά, αυτή η ιδιότητα είναι αρκετά σημαντική για τον τρόπο με τον οποίο σχεδιάζεται ένα τέτοιο σύστημα.

Υπάρχουν τρεις γενικές αρχιτεκτονικές, οι οποίες χρησιμοποιήθηκαν από κάποια συστήματα.

I. **Κεντρικοποιημένη αρχιτεκτονική**: Αυτή την αρχιτεκτονική είχε το Napster. Οι χρήστες του συνδεόντουσαν σε ένα κεντρικό υπολογιστή, και όλες οι αναζητήσεις αρχείων γίνονταν από εκεί. Ο υπολογιστής αυτός γνώριζε όσους ήταν συνδεδεμένοι, καθώς και τα αρχεία που μοιραζόντουσαν όλοι οι χρήστες.

II. **Κατανεμημένη αρχιτεκτονική**: Αυτήν την αρχιτεκτονική υιοθέτησε το σύστημα Gnutella και το Freenet. Εδώ δεν υπάρχει κανένας κεντρικός συντονισμός των χρηστών. Όλη η κίνηση στο δίκτυο γινόταν με την επικοινωνία μεταξύ των κόμβων.

III. **Ιεραρχική αρχιτεκτονική**: Η τρίτη αρχιτεκτονική είναι ένας συνδυασμός των δύο παραπάνω. Υπάρχουν κάποιοι κόμβοι που είναι περισσότερο σημαντικοί από τους υπόλοιπους (υπερκόμβοι). Το Kazaa χρησιμοποιεί υπερκόμβους στο δίκτυο του.

Πλεονεκτήματα P2P Συστημάτων

1. Οι απαιτήσεις για να ξεκινήσει και να αναπτυχθεί ένα τέτοιο σύστημα είναι χαμηλές καθώς δεν απαιτείται καμία διοικητική ή οικονομική συμφωνία μεταξύ

παροχέα και χρήστη (Server και Client), σε αντίθεση με τα κεντρικοποιημένα συστήματα.

2. Προσφέρουν έναν τρόπο αξιοποίησης της υπολογιστικής και αποθηκευτικής δυνατότητας των υπολογιστών, που είναι συνδεδεμένοι στο Internet.
3. Η αποκεντρωμένη και κατανεμημένη φύση των P2P συστημάτων τα καθιστά περισσότερο ανθεκτικά σε σφάλματα και επιθέσεις. Έτσι μπορούν να θεωρηθούν ιδανικά για μακροπρόθεσμη αποθήκευση δεδομένων ή μακροσκελείς υπολογισμούς.

Μειονεκτήματα P2P Συστημάτων

1. Η αποκέντρωση και η κατανομή της ευθύνης, όπως αναφερθήκαμε στην παραπάνω παράγραφο, είναι ένα από τα πλεονεκτήματα των P2P συστημάτων. Ωστόσο η έλλειψη ενός κεντρικού σημείου ελέγχου εκτός από τα θετικά στοιχεία, μπορεί να θεωρηθεί ότι μπορεί να προκαλέσει και αρνητικά αποτελέσματα. Ως παράδειγμα μπορεί να έχουμε P2P δίκτυο στο οποίο να κινείται ανεξέλεγκτα πληροφορία, το περιεχόμενο της οποίας να είναι συχνά παράνομο.
2. Σε ένα P2P δίκτυο δεν είναι δυνατόν να δοθούν εγγυήσεις για την ποιότητα υπηρεσιών που παρέχονται.

.

3.1.3 Αδόμητα και Δομημένα Συστήματα και παραδείγματα αυτών

Τα αδόμητα συστήματα ομότιμων κόμβων είναι αυτά που το δίκτυο που σχηματίζουν δεν έχει κάποια συγκεκριμένη τοπολογία. Η έλλειψη τοπολογίας στο δίκτυό τους, κάνει δύσκολη την αναζήτηση, καθώς μόνο γενικοί αλγόριθμοι αναζήτησης μπορούν να εφαρμοστούν. Σε αυτήν την κατηγορία ανήκουν τα πρώτα συστήματα που δημιουργήθηκαν. Βέβαια, κάποια από αυτά δεν είναι πλήρως κεντρικοποιημένα.

Napster

Το πρώτο σύστημα ομότιμων κόμβων ήταν το Napster, το οποίο όμως, δεν ήταν καθαρά κατανεμημένο, αφού είχε μια Κεντρικοποιημένη δομή. Υπήρχε ένας κεντρικός εξυπηρετητής, ο οποίος γνώριζε τα αρχεία που μοιράζονταν οι συνδεδεμένοι στο Napster χρήστες. Όταν ένας νέος χρήστης έμπαινε στο δίκτυο, επικοινωνούσε με τον κεντρικό εξυπηρετητή, και τον ενημέρωνε για τα διαθέσιμα αρχεία του. Όταν ένας χρήστης έκανε

μια αναζήτηση, ρωτούσε μόνο τον εξυπηρετητή. Στη συνέχεια οι χρήστες δημιουργούσαν μία απευθείας σύνδεση μεταξύ τους για να πραγματοποιηθεί η μεταφορά του αρχείου. Το πρόβλημα με το Napster ήταν ότι σε περίπτωση αποτυχίας του κεντρικού εξυπηρετητή, όλο το δίκτυο ήταν ανενεργό.

Kazaa

Το Kazaa, είναι ένα σύστημα που ακολουθεί το ιεραρχικό μοντέλο. Το σύστημα αυτό έχει κάποιους υπερκόμβους, που ονομάζονται Ultrapeers ή Super Nodes (SNs). Κάθε τέτοιος κόμβος είναι συνδεδεμένος με αρκετούς κόμβους φύλλα, τους Ordinary Nodes (ONs). Κάθε ON κόμβος είναι συνδεδεμένος μόνο με έναν SN κόμβο. Οι SN κόμβοι έχουν συνδέσεις μεταξύ τους, και δημιουργούν μεταξύ τους ένα δίκτυο, μέσα από το οποίο επικοινωνούν μεταξύ τους. Αυτή η δομή περιορίζει το μεγάλο πλήθος μηνυμάτων στη διάρκεια μιας αναζήτησης.

Gnutella

Περισσότερο ενδιαφέρον παρουσιάζει το Gnutella, ένα δίκτυο στο οποίο όλοι οι κόμβοι είναι ισότιμοι και οι διαδικασίες εισαγωγής και αναζήτησης γίνονται χωρίς τη βοήθεια ειδικών κόμβων.

Όταν ένας νέος χρήστης, έστω ο Α, θέλει να συνδεθεί στο δίκτυο της Gnutella, απλά επικοινωνεί με έναν συνδεδεμένο στο δίκτυο χρήστη, π.χ. τον Β. Αυτό επιτυγχάνεται από τον Α στέλνοντας ένα broadcast μήνυμα. Το μήνυμα αυτό θα το λάβει ο Β, ο οποίος απαντάει, στη συνέχεια ο Α στέλνει ένα αναγνωριστικό μήνυμα μέσα στο δίκτυο της Gnutella, με σκοπό να γίνει γνωστός και στους υπόλοιπους, καθώς επίσης και ο Α να μάθει και κάποιους άλλους κόμβους εκτός του Β. Το μήνυμα αυτό δεν ταξιδεύει σε όλο το δίκτυο, αλλά έχει ένα συγκεκριμένο χρόνο ζωής TTL (Time To Live). Έτσι ο Α γνωρίζει τους γείτονές του. Για να γίνει μία αναζήτηση μέσα στο δίκτυο της Gnutella, χρησιμοποιείται η τεχνική της πλημμύρας. Η τεχνική της πλημμύρας περιγράφεται στην αμέσως επόμενη ενότητα.

Ένα πρόβλημα που εμφανίστηκε, το οποίο είναι αρκετά σημαντικό και συμβάλλει στην αδυναμία κλιμάκωσης του συστήματος, είναι οι "ελεύθεροι χρήστες" (free riders). Οι χρήστες αυτοί δεν μοιράζονται αρχεία με αποτέλεσμα πολλές αναζητήσεις (οι οποίες έχουν ένα χρόνο ζωής TTL) να τερματίζονται χωρίς αρκετά αποτελέσματα. Ένα άλλο μειονέκτημα είναι ότι η ανάπτυξη λογισμικού για τη χρήση του δικτύου γίνεται από

ανεξάρτητους φορείς και πολλές βελτιώσεις αλλά και αλλαγές στο πρωτόκολλο, για την επίλυση τυχόν προβλημάτων, διαφέρουν σε κάθε υλοποίηση, με αποτέλεσμα να μην υπάρχει ουσιαστική επίλυση του προβλήματος.

Γενικά όμως το Gnutella έχει αρκετά πλεονεκτήματα, καθώς οι αναζητήσεις επιστρέφουν συχνά αποτελέσματα σε ερωτήσεις των χρηστών. Υπάρχει μεγάλη ανοχή σε σφάλματα, και το σύστημα είναι ευέλικτο σε αλλαγές στην τοπολογία του συστήματος.

Η άλλη κατηγορία για την οποία θα αναφερθούμε είναι τα **δομημένα συστήματα**, των οποίων τα δίκτυα έχουν συγκεκριμένη μορφή. Τα συστήματα αυτά έχουν γρήγορους αλγόριθμους αναζήτησης, αλλά ως κόστος πληρώνουμε τη διατήρηση του δικτύου, παραδείγματα τέτοιων δικτύων είναι τα δίκτυα Can, Chord και Pastry.

Η ιδέα πίσω από συστήματα αυτής της κατηγορίας είναι η αντιστοίχηση των δεδομένων όλου του δικτύου με έναν μοναδικό αριθμό, και κάθε κόμβος να είναι διατεθειμένος να αποθηκεύσει δεδομένα άλλου κόμβου. Αυτό μπορεί να γίνει είτε με αντιγραφή των αρχείων, είτε με δημιουργία δεικτών. Γίνεται χρήση μιας συνάρτησης κατακερματισμού για να επιτευχθεί η αντιστοίχηση των δεδομένων με τον αριθμό αυτό. Στη συνέχεια, κάθε κόμβος είναι υπεύθυνος για κάποια αρχεία με συγκεκριμένο αριθμό. Για κάθε αρχείο, αρκεί το όνομά του να αντιστοιχιστεί με έναν αριθμό key, με τη συνάρτηση lookup(key). Στη συνέχεια το αρχείο αυτό θα σταλεί στον κόμβο που είναι υπεύθυνος για τα αρχεία με αριθμό το lookup(key). Για την αναζήτηση, η ίδια συνάρτηση μας λέει ποιος κόμβος μπορεί να έχει το αρχείο.

Ένα τέτοιο σύστημα για να δουλέψει σωστά και αποδοτικά, πρέπει να φροντίσει για κάποια θέματα που προκύπτουν. Για αρχή, πρέπει η συνάρτηση κατακερματισμού να μοιράζει τα κλειδιά στα αρχεία με έναν ομοιόμορφο τρόπο, έτσι όλοι οι κόμβοι θα έχουν ίδιο φόρτο. Κάθε κόμβος να μπορεί να προωθεί μια αναζήτηση σε έναν κόμβο, πιο κοντά στον τελικό κόμβο. Αυτός ο απλός κανόνας είναι αρκετός για την ορθότητα της αναζήτησης. Είναι ευνόητο ότι κάθε κόμβος θα διατηρεί και έναν πίνακα δρομολόγησης.

3.2 Εισαγωγή στους Αλγορίθμους Μετάδοσης Πληροφορίας βάσει Επιδημικών Μοντέλων

Η μετάδοση πληροφοριών μέσα από συστήματα υπολογιστών ή μέσω του διαδικτύου καθώς και μέσω δικτύων κινητών κόμβων αλλά και συνδυασμένων συστημάτων επικοινωνίας (διάχυτα περιβάλλοντα), αποτελεί ένα σημαντικό πεδίο

επενδύσεων από κυβερνητικούς οργανισμούς (π.χ. ασφάλειας, στρατιωτικούς) και εταιρείες εφαρμοσμένης έρευνας, από εκπαιδευτικούς οργανισμούς και ιδιωτικές εταιρείες (π.χ. πολυτεχνικές σχολές, εταιρείες επικοινωνιών) και θεωρητικής μελέτης. Πολλά ερωτήματα παραμένουν ανοιχτά ενώ και νέα προκύπτουν και παραδίδονται στη βάσανο της μελέτης και της έρευνας. Σ' αυτό το πλαίσιο έχουν αναπτυχθεί και μελετηθεί διάφοροι τύποι αλγορίθμων, όπως οι ντετερμινιστικοί αλγόριθμοι πλημμύρας (flooding algorithms) και οι επιδημικοί αλγόριθμοι (epidemic algorithms).

Οι επιδημικοί αλγόριθμοι έχουν μεγάλη αποδοχή, ως ένας σημαντικός και εξελικτικός τρόπος μετάδοσης πληροφοριών σε κατανεμημένα συστήματα και ειδικά σε συστήματα μεγάλης κλίμακας. Οι πληροφορίες διαδίδονται σε ένα κατανεμημένο σύστημα με τον ίδιο τρόπο, που μια επιδημία θα διαδιδόταν σε μια ομάδα ατόμων: Κάθε διαδικασία που επιθυμεί να διαδώσει ένα νέο κομμάτι πληροφορίας στο σύστημα, δεν το στέλνει σε έναν κεντρικό υπολογιστή ή σε μια συστάδα κεντρικών υπολογιστών, με σκοπό τη διάδοσή του. Προτιμά την αποστολή του σε ένα σύνολο παρόμοιων διαδικασιών, που επιλέγονται τυχαία. Στη συνέχεια, κάθε μία από τις διαδικασίες κάνει το ίδιο πράγμα, και ούτω καθ' εξής.

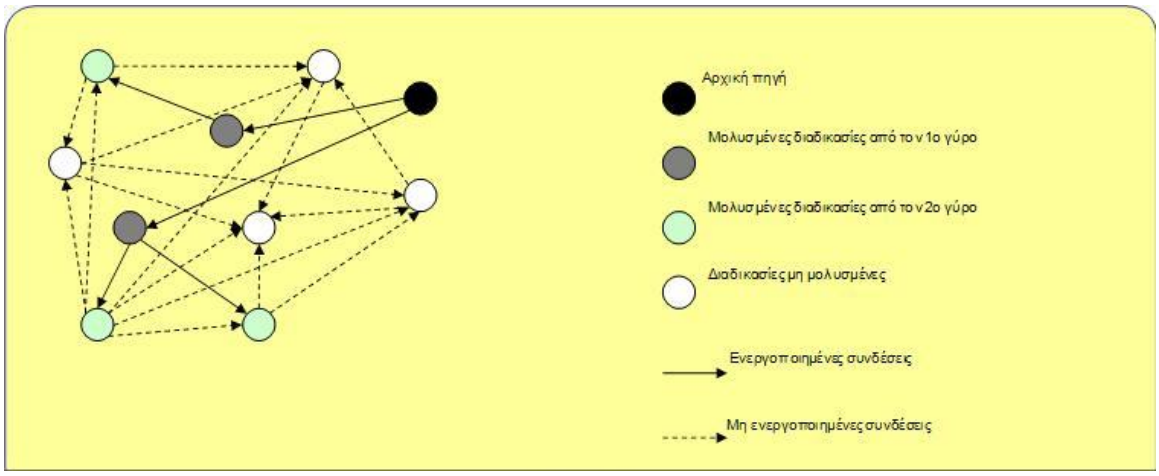
Οι επιδημικοί αλγόριθμοι έχουν μελετηθεί θεωρητικά και η ανάλυσή τους στηρίζεται στις καλές μαθηματικές υποδομές .

Η αρχή που κρύβεται πίσω από αυτού του τύπου τη διάδοση πληροφοριών, μιμείται την τεχνική διάδοσης μιας επιδημίας, έτσι αναφερόμαστε σε επιδημική διάδοση πληροφορίας. Κατ' αναλογία βρίσκουμε τη διάδοση μιας φήμης μεταξύ ανθρώπων που κουτσομπολεύουν: τότε αναφερόμαστε σε διάδοση κουτσομπολιού, από εδώ προκύπτει και η άλλη ονομασία που του έχει αποδοθεί ως **Gossip** Αλγόριθμοι.

Μόλις αρχίσουν οι επιδημίες, είναι δύσκολο να εκλείψουν. Μερικοί άνθρωποι που μολύνονται από μια μεταδοτική ασθένεια (επιδημία), είναι σε θέση να τη διαδώσουν, άμεσα ή έμμεσα, σε ένα μεγάλο πληθυσμό. Οι επιδημίες είναι ελαστικές στις αποτυχίες, κατά τη διαδικασία της μόλυνσης. Δηλαδή, ακόμα κι αν πολλοί μολυσμένοι άνθρωποι έχουν πεθάνει πριν μεταδώσουν την ασθένεια, ή είναι ανοσοποιημένοι, η επιδημία είναι ακόμα ικανή να μεταδίδεται στους πληθυσμούς.

Οι επιδημικοί αλγόριθμοι σχετίζονται με την πιθανοθεωρητική διάχυση της πληροφορίας, σ' ένα ανάλογο δίκτυο. Η έννοια της πιθανοθεωρητικής διάχυσης πληροφορίας δηλώνει ότι, ένας κόμβος i επιλέγει κάποιο γειτονικό του j, με κάποια

κατανομή πιθανότητας (μπορεί και τυχαία) και μεταδίδει (ή ανακτά) την πληροφορία σ' αυτόν (ή από αυτόν), (δηλ. ο κόμβος i μολύνει τον κόμβο j ή το αντίθετο). Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου όλοι οι κόμβοι (ή ένα ικανοποιητικό ποσοστό κόμβων) του δικτύου να έχουν μολυνθεί (δηλ. να έχουν λάβει την πληροφορία).



Σχήμα 3.2 Παρουσίαση αποτελεσμάτων του Επιδημικού τρόπου μετάδοσης μετά από 2 γύρους

Η συμπεριφορά των επιδημικών αλγορίθμων παρουσιάζει ιδιαίτερο ενδιαφέρον. Συγκεκριμένα, η τοπολογία του δικτύου, η κινητικότητα των κόμβων μέσα σε καθορισμένο χώρο, καθώς και το μοντέλο κινητικότητας, το πλήθος των κόμβων η ακτίνα μετάδοσης, ο χρόνος ανακάλυψης (ο απαιτούμενος χρόνος για να εγκαθιδρυθεί επικοινωνία με έναν κόμβο που βρίσκεται μέσα στην ακτίνα επικοινωνίας),ο ρυθμός μόλυνσης β (ο ρυθμός μόλυνσης υγειών γειτονικών κόμβων από έναν μολυσμένο κόμβο), ο ρυθμός ίασης δ (ο ρυθμός ίασης ενός μολυσμένου κόμβου), η φύση της επιδημίας (στατικό πλαίσιο πληροφορίας ή δυναμικά και ισχυρά μεταβαλλόμενο πλαίσιο), η φύση της πληροφορίας που πρόκειται να προωθηθεί και ο τύπος του επιδημικού αλγορίθμου, επηρεάζουν το συνολικό ποσοστό μόλυνσης των κόμβων του δικτύου, τον ολικό χρόνο μόλυνσης των κόμβων, το χρόνο παραμονής των κόμβων σε κατάσταση μόλυνσης.

Σε έναν επιδημικό αλγόριθμο, κάθε διαδικασία του συστήματος ενδεχομένως να εμπλέκεται στη διάδοση. Βασικά, κάθε διαδικασία αποθηκεύει κάθε μήνυμα (μονάδα πληροφορίας) που λαμβάνει, μέχρι μιας ορισμένης χωρητικότητας b (buffer capacity), και προωθεί το μήνυμα κατά ένα περιορισμένο αριθμό φορών t. Η διαδικασία διαβιβάζει

το μήνυμα κάθε φορά σε ένα τυχαία επιλεγμένο σύνολο διαδικασιών περιορισμένου μεγέθους f, που καλείται άπλωμα (fanout) της διάδοσης.

Πολλές παραλλαγές των επιδημικών αλγορίθμων διάδοσης υπάρχουν και διακρίνονται από τις τιμές των παραπάνω παραμέτρων: b, t και f. Αυτές οι παράμετροι διάδοσης μπορούν να είναι, είτε ανεξάρτητες από τον αριθμό n των διαδικασιών στο σύστημα, είτε μπορούν να μεταβάλλονται με το μέγεθος N του συστήματος. Η αξιοπιστία της μετάδοσης των πληροφοριών εξαρτάται από τις τιμές των παραμέτρων, καθώς επίσης και από το μέγεθος n του συστήματος.

Εγγενώς, η αξιοπιστία των επιδημικών αλγορίθμων βασίζεται σ' έναν προ-νοητικό μηχανισμό, στον οποίον ο πλεονασμός και η τυχαιότητα λειτουργούν έτσι ώστε να παρακάμπτονται πιθανές αποτυχίες διαδικασίας, καθώς και αποτυχίες συνδέσεων δικτύου. Αυτό συμβαίνει επειδή, εξ ορισμού, κάθε διαδικασία επιλέγει τυχαία ένα υποσύνολο του μεγέθους f άλλων διαδικασιών, στις οποίες οι πληροφορίες διαβιβάζονται. Μια διαδικασία που λαμβάνει τις πληροφορίες, επιλέγει ένα άλλο υποσύνολο και ούτω καθ' εξής. Καμία πρόνοια και κανένας μηχανισμός δεν απαιτείται, για την ανίχνευση και την αναμόρφωση των αποτυχιών. Αντίθετα από τους αναδραστικούς αλγορίθμους, στους οποίους οι διαδικασίες αναδρούν στις αποτυχίες, με την αναμετάδοση των ελλειπουσών πληροφοριών. Κατά κάποιο τρόπο, οι επιδημικοί αλγόριθμοι εμφανίζουν μια διττή συμπεριφορά: υπάρχει ένα κρίσιμο κατώτατο όριο στις τιμές των παραμέτρων διάδοσης, για τις οποίες μια αξιόπιστη μετάδοση έχει πολύ υψηλή πιθανότητα. Η πιθανοτική εγγύηση αξιόπιστης μετάδοσης του μηνύματος σχετίζεται, άμεσα με τις τιμές των παραμέτρων διάδοσης. Αυτές οι παράμετροι μπορούν να ρυθμιστούν έτσι ώστε με μια αυθαίρετα υψηλή πιθανότητα, ο επιδημικός αλγόριθμος να ικανοποιεί τις εγγυήσεις αξιοπιστίας, που παρέχουν και οι ντετερμινιστικοί αλγόριθμοι.

Ορισμός 1	Ένας πληθυσμός είναι ένα σύνολο κόμβων. Ο πληθυσμός μπορεί να οριστεί χωρικά, πχ, όλοι οι κόμβοι που βρίσκονται σε κάποια συγκεκριμένη γεωγραφική περιοχή. Οι κόμβοι ενός πληθυσμού είναι ικανοί να διαχέουν πληροφορία πλαισίου μεταξύ τους, με στόχο να ενημερώνουν και να συμπεραίνουν το πλαίσιο, συνεργαζόμενοι μεταξύ τους. Έστω N το μέγεθος του πληθυσμού των κόμβων και ο δείκτης i, ο κάθε κόμβος, i του N.	
Ορισμός 2	Οι γείτονες ενός κόμβου i δημιουργούν τη γειτονιά του i, (Γ_i υποσύνολο N). Γ_i είναι το σύνολο των κόμβων (συμπεριλαμβανομένου και του i), που βρίσκονται σε ακτίνα επικοινωνίας με τον i. Ο επιδημικός αλγόριθμος επιλέγει τυχαία ένα υποσύνολο (μεγέθους f) του Γ_i , με σκοπό να διαχύσει ο i κόμβος μια ενημερωμένη έκδοση ενός κομματιού πληροφορίας. Τα κριτήρια επιλογής αναφέρονται παρακάτω.	
Ορισμός 3	Ένας κόμβος i ονομάζεται υγιής (Susceptible), αν δε μεταφέρει ενημερωμένο πλαίσιο πληροφορίας και είναι ευαίσθητος (εύτρωτος) προς μόλυνση (μπορεί να δεχτεί ενημερωμένο πλαίσιο πληροφορίας), από ένα μολυσματικό κόμβο j που ανήκει στη γειτονιά Γ_i ή αν έχει ιαθεί (π.χ. μεταφέρει ένα πολυδιατηρημένο κομμάτι πλαισίου).	
Ορισμός 4	Ένας κόμβος i ονομάζεται μολυσμένος, όταν για ένα καθορισμένο χρονικό διάστημα μεταφέρει ενημερωμένο πλαίσιο πληροφορίας και αφού μολυνθεί παραμένει μολυσματικός, μέχρι αυτή η πληροφορία να γίνει άχρηστη.	
Ορισμός 5	Ένας κόμβος i ονομάζεται ανοσοποιημένος (Recovered), όταν για ένα καθορισμένο χρονικό διάστημα μολύνθηκε, στη συνέχεια ιάθηκε αλλά δε μπορεί να ξανά μολυνθεί (έχει πλέον αντισώματα).	
Ορισμός 6	Επιδημική μεταστοιχείωση σημαίνει, πως η επιδημία αλλάζει, αλλά συνεχίζει να παραμένει μολυσματική (αναφερόμενη σε δυνατότερο ιό). Στο πλαίσιό μας, η μεταστοιχειωμένη επιδημία αναφέρεται σε μια πιο εξειδικευμένη συνεπαγόμενη κατάσταση (μόλυνση ανώτερου επιπέδου), ώστε ο πληθυσμός να μπορεί να ξανά μολυνθεί.	
Ορισμός Παραμέτρων	**N**	Το πλήθος των κόμβων του δικτύου.
	b	Η χωρητικότητα της μνήμης των κόμβων (buffer capacity)
	t	Ο αριθμός των φορών που ο κάθε κόμβος προωθεί το ενημερωμένο πλαίσιο.
	f	Άπλωμα (fanout) της διάδοσης είναι το μέγεθος του υποσυνόλου της Γειτονιάς, στο οποίο κάθε φορά διαβιβάζεται το μήνυμα.
	m	Συμπεριφορά κινητικότητας (η τιμή στο σύνολο [0, 1], καθορίζει την κίνηση ή όχι των κόμβων)
	r	Ακτίνα επικοινωνίας (καθορίζει το χώρο εντός του οποίου θεωρούμε ότι οι κόμβοι είναι σε επαφή – δυνατότητα ανταλλαγής πληροφοριών)
	β	Ο ρυθμός μόλυνσης (εκφράζει την πιθανότητα ένας κόμβος να μολυνθεί)
	δ	Ο ρυθμός ίασης (εκφράζει την πιθανότητα ένας μολυσμένος κόμβος να ιαθεί)
	λ→β/δ	Επιδημικό όριο (Η τιμή του καθορίζει αν έχουμε ενδημία, ή πανδημία)

Πίνακας 3-1 Απαραίτητοι Ορισμοί στους Επιδημικούς Αλγορίθμους [24]

3.2.1 Τύποι διάδοσης πληροφορίας

Στους επιδημικούς αλγορίθμους χρησιμοποιούνται δύο τύποι διάδοσης πληροφορίας. Ο πρώτος αναφέρεται ως «Διάδοση πληροφορίας πλαισίου» («contextual information dissemination») και ο δεύτερος ως «Διάδοση πληροφορίας κατάστασης» («situational context dissemination»). Ως πλαίσιο (context) ορίζεται οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει μια συγκεκριμένη ολότητα, (entity). Με τον όρο ολότητα μπορεί να περιγραφεί ένας άνθρωπος, είτε μία θέση, είτε ένα αντικείμενο, που θεωρείται σχετικό για την ολοκλήρωση μεταξύ του χρήστη και της εφαρμογής συμπεριλαμβανομένων και των ιδίων. Στον πρώτο τύπο μετάδοσης πληροφορίας, η επιδημία αναπαρίσταται από τη μεταφορά δεδομένων πλαισίου (contextual data), όπως μετρήσεις από αισθητήρες (φως, ταχύτητα, θερμοκρασία), οι οποίες είναι έγκυρες για περιορισμένο χρονικό διάστημα και γεωγραφική περιοχή. Μια ομάδα κόμβων μπορεί να μοιραστεί αυτές τις πληροφορίες με συνεργατικό τρόπο. Αυτό σημαίνει πως κόμβοι που δεν έχουν αισθητήρες, άρα δεν είναι ικανοί να πάρουν μόνοι τους μετρήσεις, μπορούν να αποκτήσουν τα δεδομένα αυτά, μετά την εφαρμογή της επιδημικής διάχυσης της πληροφορίας. Στο δεύτερο τρόπο διάδοσης η συνεπαγόμενη θέση ή κατάσταση ενός κόμβου, μπορεί να μεταδοθεί μέσα από ένα σύνολο κόμβων. Πρέπει να τονιστεί πως, η μετάδοση πληροφορίας κατάστασης εξαρτάται άμεσα και ισχυρά από τη μετάδοση πληροφορίας πλαισίου, επειδή η κατάσταση ορίζεται (ή συνεπάγεται) ως συνδυασμός επιμέρους συστατικών πληροφορίας πλαισίου. Κατ’ αναλογία με το βιολογικό μοντέλο, όπως τα βακτήρια μπορούν να δημιουργούν διαφορετικούς τύπους ασθενειών, έτσι από διάφορα κομμάτια πληροφορίας πλαισίου μπορούν να συνεπάγονται διαφορετικές καταστάσεις. Αντιλαμβανόμαστε πως η διάδοση πλαισίου αντιστοιχεί στη διάδοση των βακτηρίων και η διάδοση κατάστασης αντιστοιχεί στην επιδημία [25].

Για καλύτερη κατανόηση των εννοιών, παρουσιάζουμε στο παρακάτω σχήμα αναλυτικά την αντιστοίχηση της επιδημιολογικής έννοιας, με την έννοια της διάχυσης της πληροφορίας [38].

Παράμετρος	Επιδημιολογία	Διάχυση πληροφορίας σε Δικτυακά Μοντέλα
N	Αριθμός ατόμων	Αριθμός κόμβων στο σύστημα
S(t)	Αριθμός των υγιών ατόμων	Αριθμός των συσκευών που ενδιαφέρονται για τα πακέτα πληροφορίας
I(t)	Αριθμός των μολυσμένων ατόμων	Αριθμός των συσκευών που κατέχουν τα πακέτα πληροφορίας για τα οποία ενδιαφέρονται οι άλλοι κόμβοι
x	Επαφές των ατόμων ανά μονάδα χρόνου και ανά ποσότητα ατόμων	Επαφές των κόμβων ανά μονάδα χρόνου και ανά ποσότητα κόμβων
β	Η πιθανότητα μετάδοσης ασθένειας μεταξύ ενός μολυσμένου ατόμου και ενός υγιούς ατόμου	Η πιθανότητα μετάδοσης ασθένειας μεταξύ μιας συσκευής ή κόμβου που κατέχει το πακέτο και ενός κόμβου που θέλει να το παραλάβει
$a = \frac{\beta \cdot x}{N}$	Ρυθμός μετάδοσης της ασθένειας	Ρυθμός μετάδοσης των κόμβων που κατέχουν τα πακέτα προς τους κόμβους που ενδιαφέρονται να τα αποκτήσουν

Πίνακας 3-2 Πίνακας αντιστοίχησης της Επιδημιολογικής έννοιας με την έννοια διάχυσης της Πληροφορίας

3.3 Μαθηματική Μοντελοποίηση των επιδημιών

Η επιστήμη ασχολείται με τις «επιδημίες» από την εποχή του Ιπποκράτη, τον 4 αιώνα π.χ., ο οποίος είχε αναφερθεί για πρώτη φορά στις επιδημίες. Στη συνέχεια αναφέρθηκαν μετά από έρευνες στον τομέα της Ιατρικής ο John Graunt (1662), Louis Pasteur, και Robert Koch τον 19 αιώνα. Στον τομέα των Μαθηματικών ασχολήθηκε ο Daniel Bernoulli (1760) με την μοντελοποίηση των επιδημιών. Ένα απλό Επιδημικό μοντέλο παρουσιάστηκε και αναπτύχθηκε το 1911 από τον Ross και το 1927 από τους Kermack και McKendrick's παρουσιάστηκε για πρώτη φορά το Γενικό μοντέλο επιδημιών.

Είναι εύκολο να αντιληφθούμε την αναλογία της μετάδοσης πληροφορίας μεταξύ κινητών συσκευών που χρησιμοποιούνται από χρήστες και της διάδοσης μιας μολυσματικής επιδημίας μεταξύ μιας ομάδας ανθρώπων. Και οι δύο μπορούν να χαρακτηριστούν σαν διαδικασίες κατά τις οποίες πρέπει να συμβεί κάποιου είδους «επαφή» για την εξάπλωσή τους, για παράδειγμα ένα «μολυσμένο» αντικείμενο να πλησιάσει σε κάποια απόσταση άλλα μη μολυσμένα αντικείμενα. Έτσι, το είδος των αλγόριθμων που περιγράφουν με αυτόν τον τρόπο τη διάχυση της πληροφορίας, ονομάζονται επιδημικοί αλγόριθμοι.

Οι επιδημικοί αλγόριθμοι μπορούν να κατηγοριοποιηθούν σε τρεις μεγάλες κατηγορίες, ανάλογα με τον τρόπο που επικοινωνούν οι γειτονικοί κόμβοι μεταξύ τους **[23],** οι οποίοι παρουσιάζονται και περιγράφονται στο **Σχήμα 3.3**.

Σχήμα 3.3 Περιγραφή και Παρουσίαση Αλγορίθμων Push-based, Pull-based και Push & Pull-based

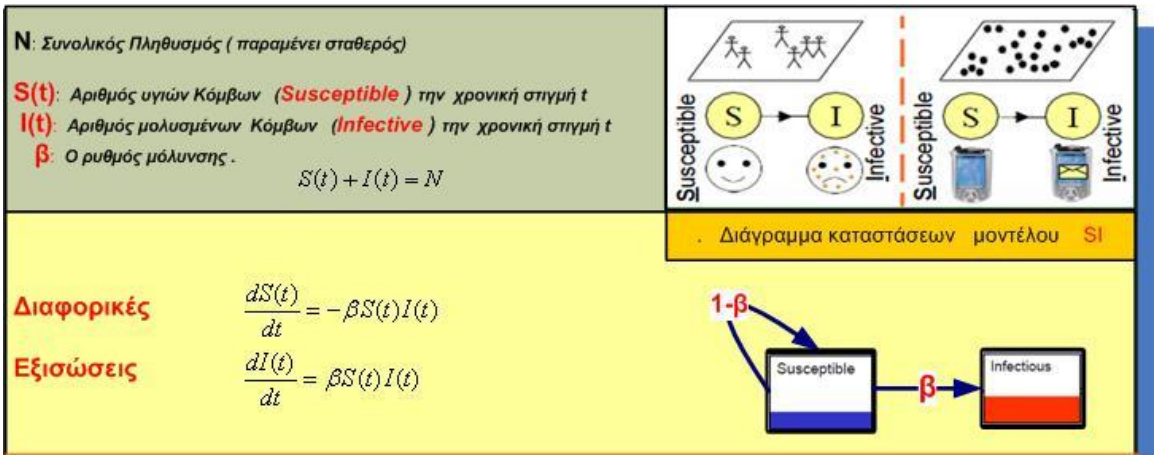
3.4 Μοντέλα Επιδημικών Αλγορίθμων

Συνδυάζοντας μόλυνση και ίαση (και γενικά τις διάφορες καταστάσεις στις οποίες μπορεί να βρεθεί ένας κόμβος) προκύπτουν διάφορα Επιδημικά Μοντέλα, όπως π.χ. τα παρακάτω: ***SI, SEI, SEIS, SIR, SIRS, SEIR, SEIRS, MSEIR*** *και* ***MSEIRS***, στα οποία μοντέλα οι συμβολισμοί παριστάνουν τις διάφορες καταστάσεις που μπορούν να εμφανιστούν: Susceptible **(S)**, Infectious (**I)**, Removed (**R**), Exposed (**E**) και iMmume (**M**). Τα κυριότερα από τα παραπάνω Επιδημικά Μοντέλα τα περιγράφουμε στις παρακάτω υποενότητες **[38]:**

3.4.1 Μοντέλο SI

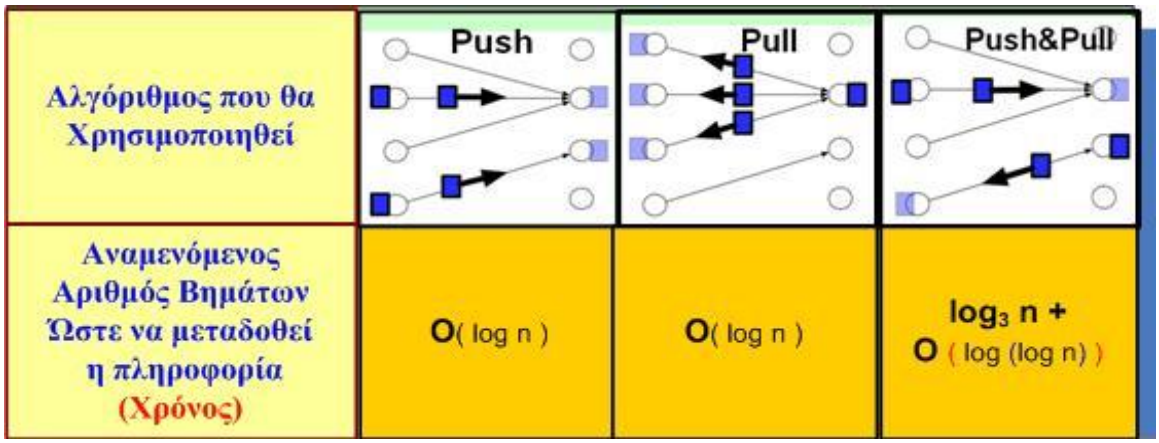
Στο Susceptible **(S)** - Infected (**I)** (SI), το οποίο περιγράφεται και σαν κλασικό επιδημικό μοντέλο, κάθε κόμβος θεωρούμαι ότι βρίσκεται σε μια από τις δύο διακριτές καταστάσεις: υγιής (ευάλωτος) ή μολυσμένος. Σ' αυτό το μοντέλο αν ένας κόμβος μολυνθεί, δε μπορεί να ιαθεί (παραμένει για πάντα μολυσμένος – διατηρεί την πληροφορία πλαισίου που έχει λάβει και η οποία θεωρείται έγκυρη για πάντα). Στο μοντέλο αυτό δ=0

Σχηματικά η κατάσταση αυτή περιγράφεται στο **Σχήμα 3.4** που ακολουθεί.



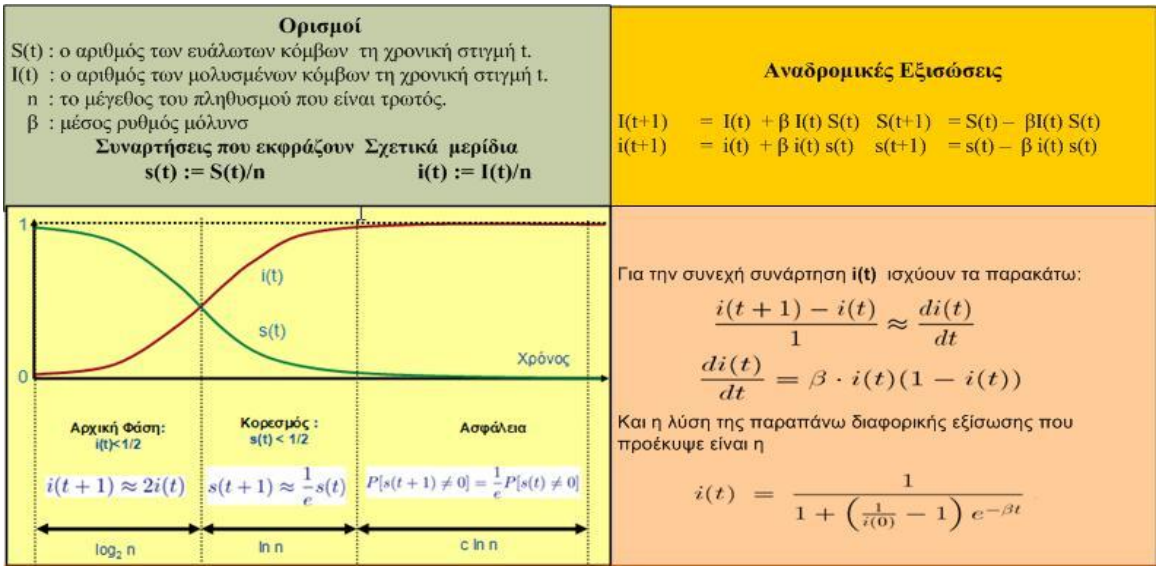
Σχήμα 3.4 Περιγραφή του Κλασικού Επιδημικού Μοντέλου SI

Στο **Σχήμα 3.5** αναγράφονται οι Πολυπλοκότητες του Απλού Μολυσματικού Μοντέλου SI ως προς τις διαφορετικές τεχνικές διάχυσης της πληροφορίας **[30].**



Σχήμα 3.5 Πολυπλοκότητα του Επιδημικού Μοντέλου SI

Στη συνέχεια θα αναλύσουμε το Μοντέλο SI κάνοντας χρήση της τεχνικής μετάδοσης Push και βάσει των ορισμών που είναι ενσωματωμένοι στο **Σχήμα 3.6** έχει προκύψει η γραφική παράσταση. Είναι εύκολο να δούμε ότι η αύξηση των susceptible υγειών είναι αντιστρόφως ανάλογη της αύξησης της μόλυνσης. Το μοντέλο υποθέτει ότι ο αρχικός αριθμός των κόμβων που είναι susceptible, είναι αρκετά μεγαλύτερος του αριθμού αυτών που είναι μολυσμένοι. Σαν αποτέλεσμα, ο αρχικός ρυθμός μόλυνσης είναι εκθετικός. Καθώς ο αριθμός των susceptible και infectious κόμβων έρχεται σε μία ισορροπία, ο ρυθμός μόλυνσης αρχίζει να μειώνεται, αλλά δε σταματά μέχρις ότου όλοι οι κόμβοι που είναι ευάλωτοι να γίνουν μολυσμένοι. Μετά το σημείο ισορροπίας αρχίζει ο αριθμός των υγειών κόμβων να ελαττώνεται με εκθετικό ρυθμό **[30].**



Σχήμα 3.6 Γραφική παράσταση των συναρτήσεων i(t) και s(t)

3.4.2 Μοντέλο SIS : Susceptible-Infected-Susceptible

Το επιδημικό μοντέλο ονομάζεται Susceptible-Infected-Susceptible (SIS), δηλαδή Υγιής-Μολυσμένος-Υγιής. Αυτό είναι ένα πολύ διαδεδομένο μοντέλο που χρησιμοποιείται στην επιδημιολογική έρευνα.

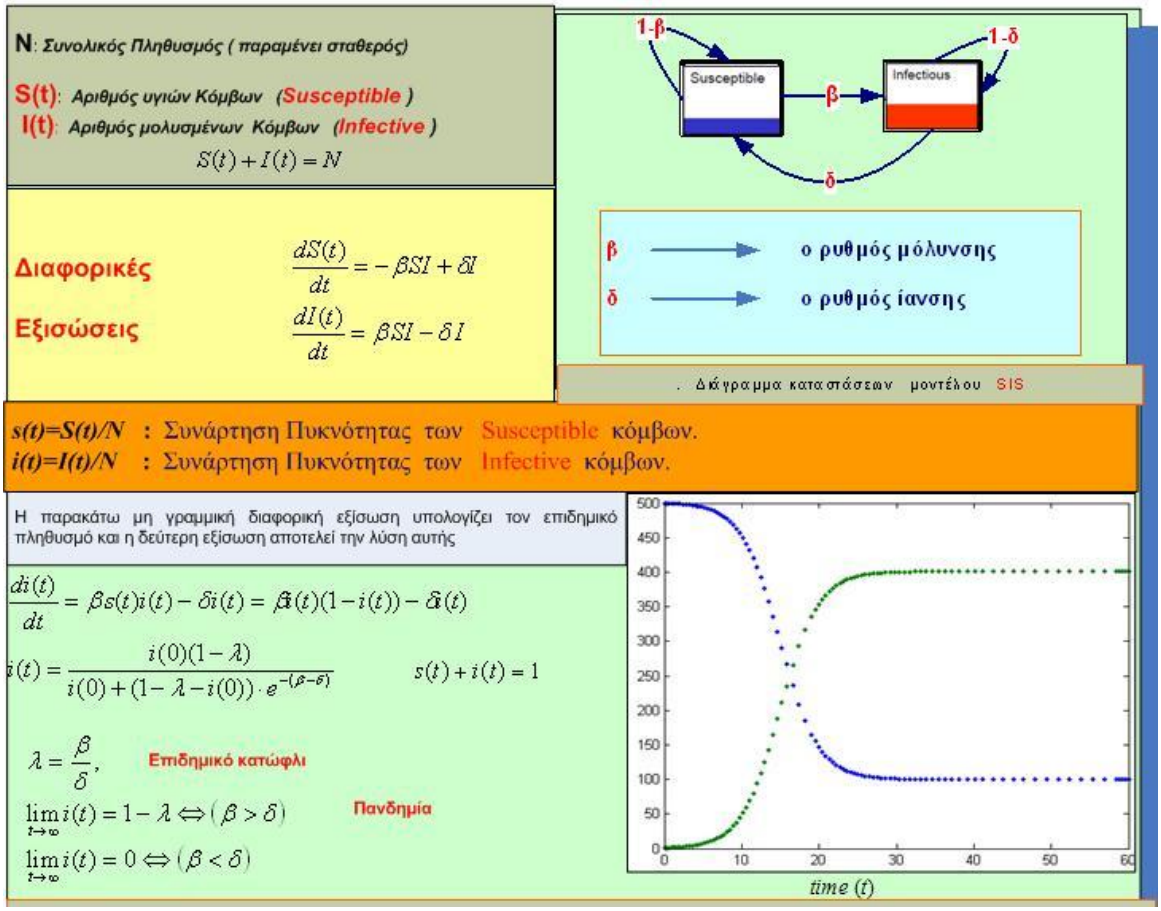
Το μοντέλο SIS:

i. αγνοεί τις λεπτομέρειες μόλυνσης μέσα στον κόμβο,

ii. απλοποιεί την επιδημική μετάδοση, ως πιθανότητες ανά χρονική μονάδα των ρυθμών β και δ και

iii. θεωρεί έναν κόμβο σε μία από τις δύο διακριτές καταστάσεις: μολυσμένος ή υγιής.

Το μοντέλο SIS θεωρεί ότι ένας μολυσμένος κόμβος δεν μπορεί να ξαναμολυνθεί. Στην περίπτωση που αναφερόμαστε στη διάχυση πληροφορίας πλαισίου, ο περιορισμός της μη επαναμόλυνσης ενός ήδη μολυσμένου κόμβου είναι αποδεκτός. Από τη άλλη όταν αναφερόμαστε σε διάχυση κατάστασης, στην οποία η επιδημία μπορεί να μεταλλαχθεί (περιστασιακό πλαίσιο situational context), το μοντέλο SIS επεκτείνεται ώστε ένας μολυσμένος κόμβος να μπορεί να ξαναμολυνθεί από ισχυρότερη μόλυνση (μόλυνση ανώτερου επιπέδου). Επιπλέον το μοντέλο υποθέτει ότι ένας κόμβος δεν μπορεί να γίνει λιγότερο επιρρεπής, μετά από οποιαδήποτε μόλυνση. Έτσι υποθέτουμε ότι τα β και δ δεν αλλάζουν με τον χρόνο **[21].**



Σχήμα 3.7 Περιγραφή του μοντέλου SIS

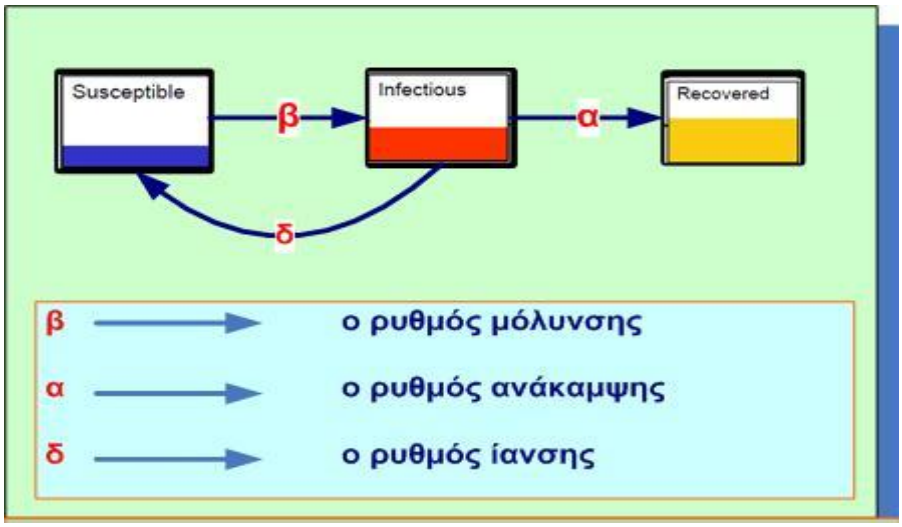
3.4.3 Μοντέλο SIR: S-susceptible, I - infected, R - recovered/removed

Σύμφωνα με το μοντέλο των Kermack και McKendrick, ένας κόμβος μπορεί να βρίσκεται σε τρείς καταστάσεις : Μολυσμένος (Infected), Υγιής ή Επιρρεπής (Susceptible), δηλ. είναι επιρρεπής στη μόλυνση από την επιδημία και τέλος στην κατάσταση που έχει αποκτήσει ανοσία (Recovered). Το επιδημικό μοντέλο ονομάζεται Susceptible-Infected- Recovered (SIR) δηλαδή Υγιής-Μολυσμένος-Ανοσοποιημένος.

Περίπτωση 1η

Στο μοντέλο SIR, ένας κόμβος που είναι Υγιής, μπορεί με κάποια πιθανότητα να μολυνθεί και να γίνει Μολυσμένος. Από την κατάσταση της μόλυνσης μπορεί να ιαθεί και είτε να ξανακαταστεί ευαίσθητος (Susceptible) για μόλυνση, είτε να αποκτήσει αντισώματα και να μη μπορεί πλέον να ξαναμολυνθεί (Recovered).

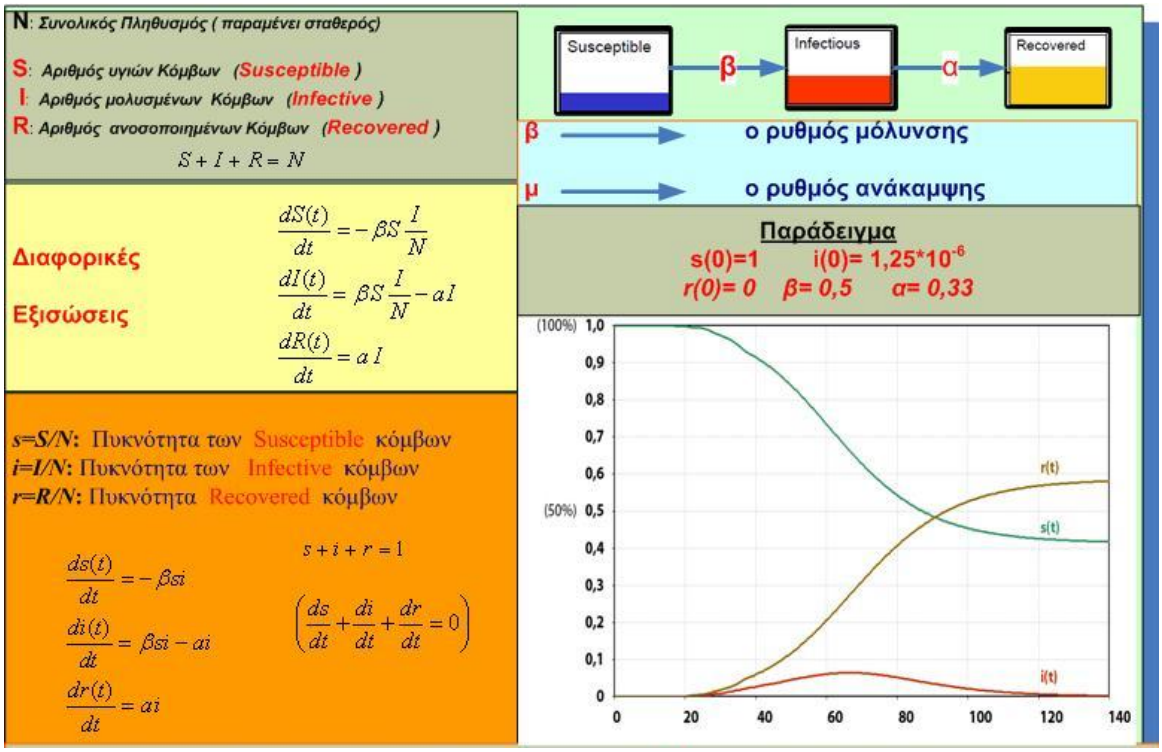
Στο μοντέλο SIR εκτός από τους ρυθμούς β και δ ορίζεται και ο ρυθμός α ως ο ρυθμός ανοσοποίησης (εκφράζει την πιθανότητα ένας μολυσμένος κόμβος να μεταβεί από την κατάσταση Infected στην κατάσταση Recovered).



Σχήμα 3.8 Περιγραφή του μοντέλου SIR - Περίπτωση 1η

Περίπτωση 2η

Μια άλλη εκδοχή του Μοντέλου SIR πιο απλή σε διάγραμμα μετάβασης καταστάσεων και πιο συχνά εμφανιζόμενη είναι η παρακάτω:



Σχήμα 3.9 Περιγραφή του μοντέλου SIR - Περίπτωση 2η

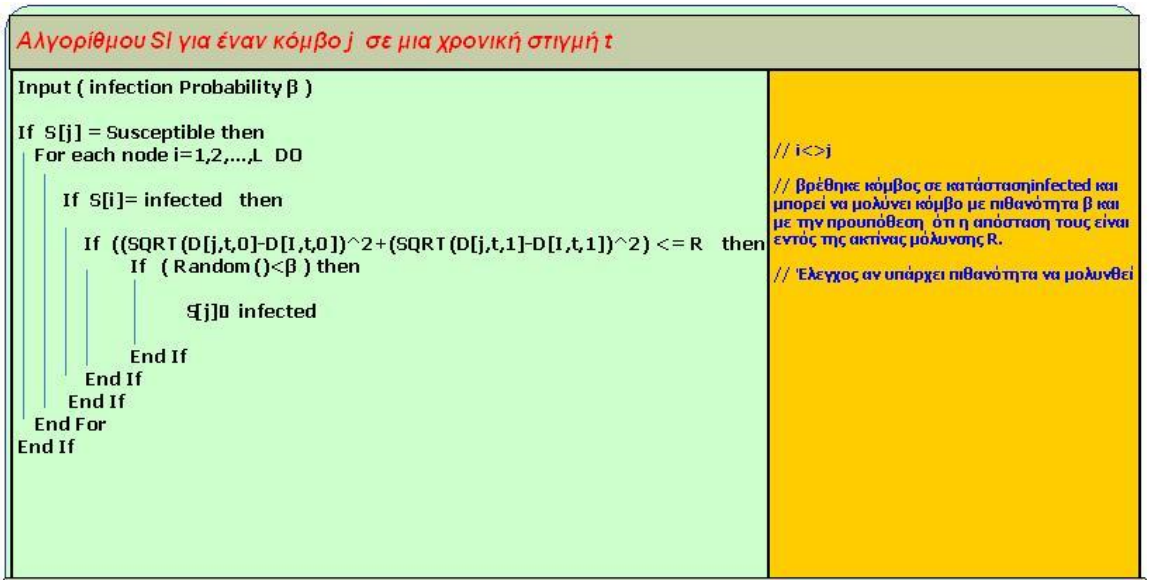
Η γραφική παράσταση που απεικονίζεται στο **Σχήμα 3.9** έχει προκύψει από την εφαρμογή του παραπάνω μοντέλου, για τις τιμές των παραμέτρων του παραδείγματος.

3.5 Περιγραφή των αλγόριθμων διάδοσης Πληροφορίας με βάση τα Επιδημικά μοντέλα που περιγράψαμε

3.5.1 Ο SI Αλγόριθμος S-susceptible, I – infected

Στον αλγόριθμο SI οι καταστάσεις ενός κόμβου μπορούν να είναι δύο: υγιής και μολυσμένος. Οι μεταβάσεις μεταξύ αυτών είναι μόνο από S→I, που περιγράφεται ως μόλυνση κόμβου.

Ελέγχεται, αν ένας κόμβος j που βρίσκεται σε υγιή κατάσταση, μπορεί με κάποια τυχαία πιθανότητα να μολυνθεί από κάποιον από όλους τους μολυσμένους γειτονικούς του κόμβους. Ο αλγόριθμος παρουσιάζεται παρακάτω:



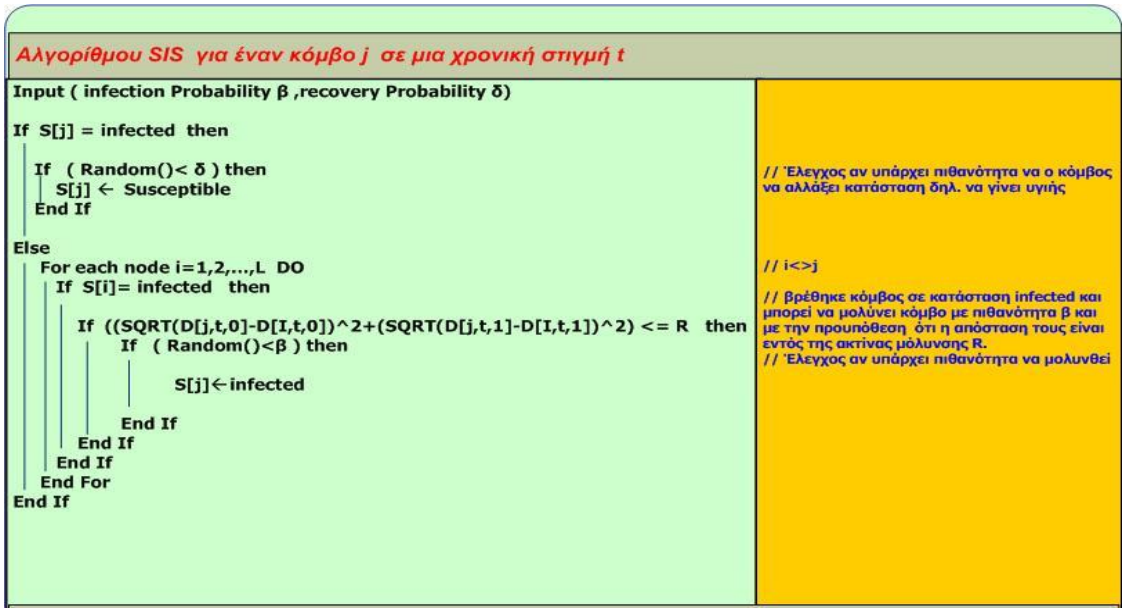
Σχήμα 3.10 Παρουσίαση Επιδημικού Αλγορίθμου SI

3.5.2 Ο SIS Αλγόριθμος S-susceptible, I - infected, S-susceptible

Στον αλγόριθμο SIS οι καταστάσεις ενός κόμβου μπορούν να είναι δύο: υγιής και μολυσμένος. Έτσι, οι μεταβάσεις μεταξύ αυτών είναι από 1 σε 0, που περιγράφεται σαν ίαση κόμβου και από 0 σε 1, που αντιστοιχεί στη μόλυνση κόμβου. Σχηματικά η κατάσταση αυτή περιγράφεται στο **Σχήμα 3.7**. Αρχικά ελέγχεται αν ένας κόμβος j, που

βρίσκεται σε μολυσμένη κατάσταση, με κάποια τυχαία πιθανότητα μπορεί να μεταβεί σε υγιή (ίαση), διαφορετικά για όλους τους μολυσμένους γειτονικούς του κόμβους, ελέγχεται αν μπορεί κάποιος από αυτούς να τον μολύνει.

Ο αλγόριθμος παρουσιάζεται παρακάτω:



Σχήμα 3.11 Παρουσίαση Επιδημικού Αλγορίθμου SIS

3.5.3 *SIR Αλγόριθμος S-susceptible, I - infected, R - recovered/removed*

Στον αλγόριθμο SIR οι καταστάσεις ενός κόμβου μπορούν να είναι τρεις: υγιής, μολυσμένος και ανοσοποιημένος. Έτσι, οι μεταβάσεις μεταξύ αυτών παρουσιάζονται στο **Σχήμα 3.8.**

Ο αντίστοιχος αλγόριθμος που μπορεί να προκύψει από το διάγραμμα μετάβασης καταστάσεων, περιγράφεται στο **Σχήμα 3.12**. Αρχικά ελέγχεται αν ένας κόμβος j, που βρίσκεται σε μολυσμένη κατάσταση, με κάποια τυχαία πιθανότητα μπορεί να μεταβεί σε υγιή (ίαση) ή σε ανοσοποιημένη κατάσταση, διαφορετικά, κι εφόσον δεν είναι Recovered, για όλους τους μολυσμένους γειτονικούς του κόμβους ελέγχεται αν μπορεί κάποιος από αυτούς να τον μολύνει.

Αλγορίθμος SIR

```
Input ( infection Probability β ,recovery Probability δ, cure Probability α)

If S[j] = infected then
|  If  ( Random()< δ ) then                                   // Έλεγχος αν υπάρχει πιθανότητα ο κόμβος να αλλάξει κατάσταση δηλ. να γίνει Recovered
|  |    S[j] ← Recovered
|  |    Else If ( δ < Random () < δ+α )                       // Έλεγχος αν υπάρχει πιθανότητα ο κόμβος να αλλάξει κατάσταση δηλ. να γίνει Susceptible
|  |       S[j] ← Susceptible
|  |    End If
|  End If
Else
|  For each node i=1,2,...,L  DO                               // i<>j
|  |  If  S[i]= infected  then
|  |  |
|  |  |  If ((SQRT(D[j,t,0]-D[I,t,0])^2+(SQRT(D[j,t,1]-D[I,t,1])^2) <= R  then   // βρέθηκε κόμβος σε κατάσταση infected και μπορεί να μολύνει κόμβο με πιθανότητα β και με την προυπόθεση ότι η απόσταση τους είναι εντός της ακτίνας μόλυνσης R.
|  |  |     If  ( Random()<β ) then                           // Έλεγχος αν υπάρχει πιθανότητα να μολυνθεί
|  |  |     |
|  |  |     |     S[j]←infected
|  |  |     |
|  |  |     End If
|  |  |  End If
|  |  End If
|  End For
End If
```

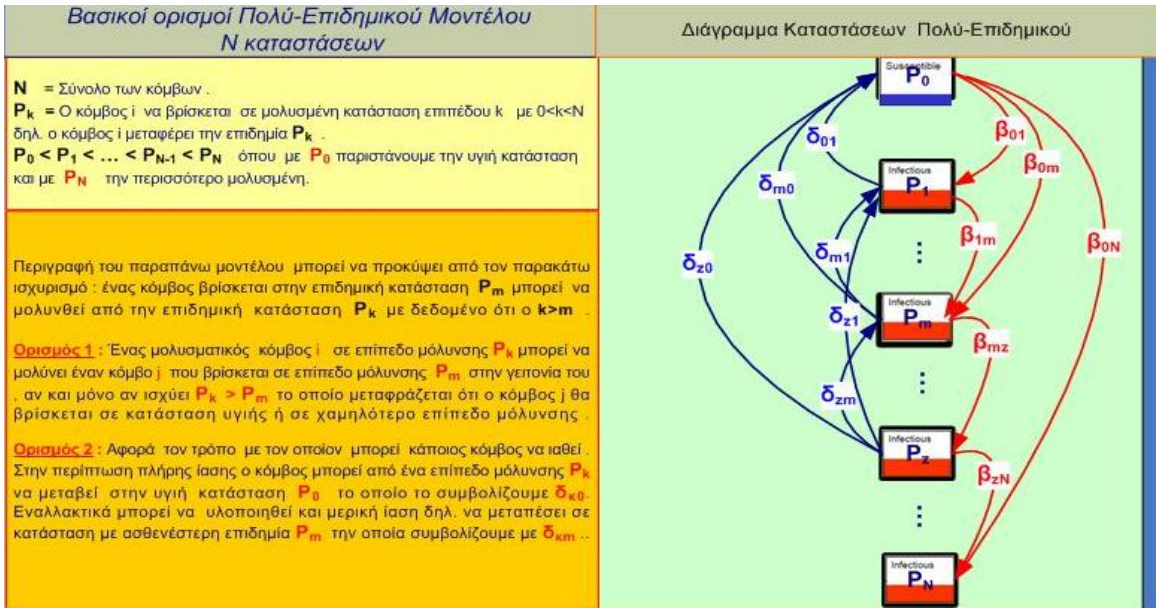
Σχήμα 3.12 Παρουσίαση Αλγορίθμου SIR για έναν κόμβο j σε μια χρονική στιγμή t

3.6 Πολύ-Επιδημική Διάδοση Πληροφορίας

Η πολύ-επιδημική διάδοση σχετίζεται με πολλές καταστάσεις μόλυνσης, για τους κόμβους που αποτελούν το δίκτυο. Στην επόμενη ενότητα που ακολουθεί, θα περιγράψουμε ένα Πολύ-Επιδημικό Μοντέλο.

3.6.1 Παρουσίαση του Πολύ-Επιδημικού Μοντέλου Ν καταστάσεων

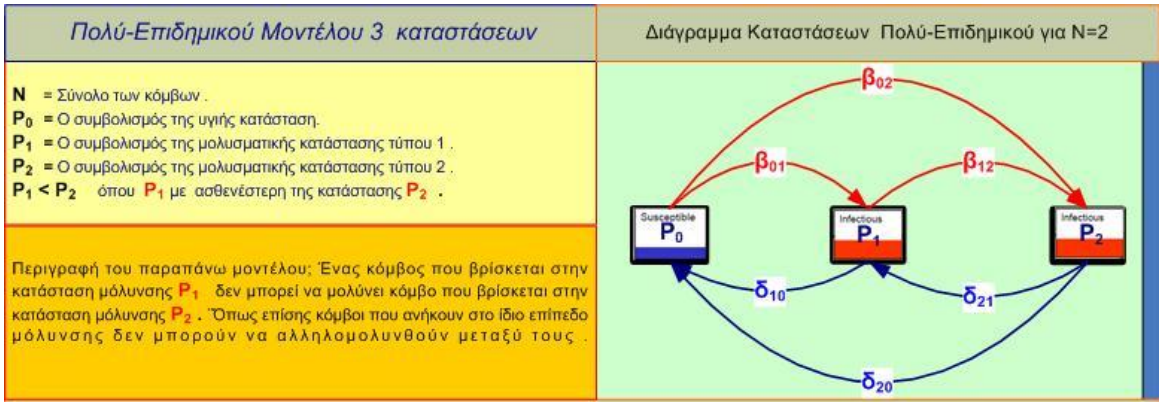
Η πολύ-επιδημική σχετίζεται με πολλές καταστάσεις μόλυνσης, για τους κόμβους που αποτελούν το δίκτυο. Στο παρακάτω σχήμα που ακολουθεί, θα περιγράψουμε ένα Πολύ-Επιδημικό Μοντέλο**[37]**.



Σχήμα 3.13 Περιγραφή του Πολύ-Επιδημικού Μοντέλου Διάδοσης Πληροφορίας

3.6.2 Αλγόριθμοι και περιγραφή Πολύ-Επιδημικού Μοντέλου για N=2

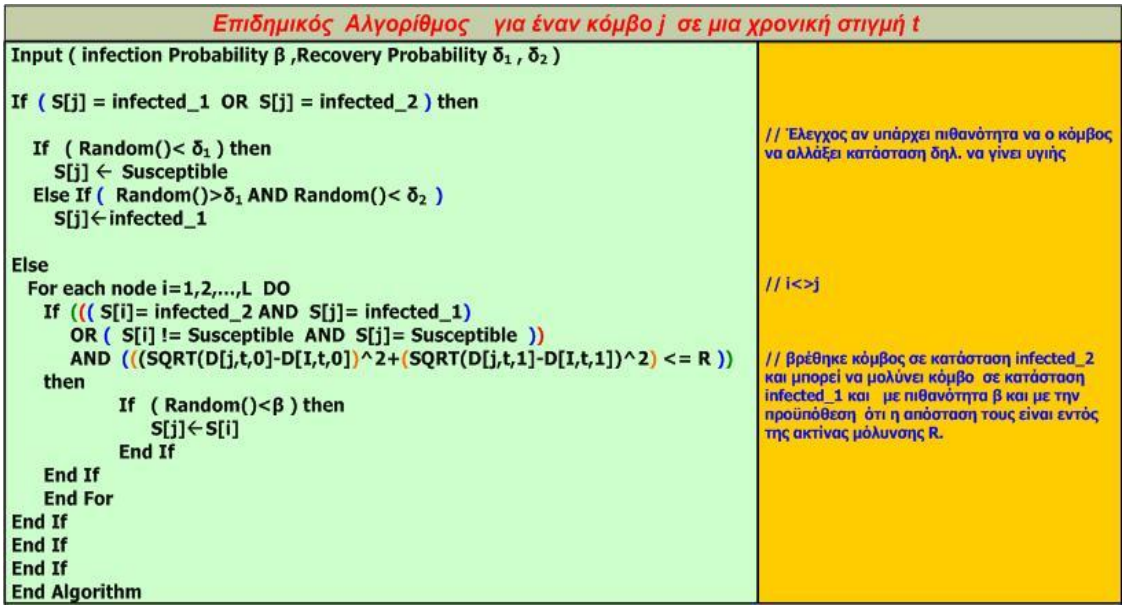
Παρακάτω και στο **Σχήμα 3.14,** περιγράφεται ένα Πολύ-επιδημικό Μοντέλο 3 καταστάσεων για έναν κόμβο: **[26]**



Σχήμα 3.14 Περιγραφή του Πολύ-Επιδημικού Μοντέλου Διάδοσης Πληροφορίας για N=2

Στη συνέχεια και στο **Σχήμα 3.15** θα παρουσιάσουμε τον Αλγόριθμου που υλοποιεί το παραπάνω διάγραμμα καταστάσεων του Πολύ Επιδημικού Μοντέλου για N=2. Ο προτεινόμενος αλγόριθμος εκτελεί τα εξής βήματα:

Για ένα δεδομένο κόμβο j, αρχικά γίνεται έλεγχος αν με τυχαία διαφορετική κάθε φορά πιθανότητα μπορεί να ιαθεί από κατάσταση P_2 σε P_1 (μόλυνση χαμηλότερου επιπέδου ή μερική ίαση), ή από P_2 σε P_0 (ίαση) ή από P_1 σε P_0 (ίαση). Στη συνέχεια, για όλους τους γειτονικούς του κόμβους (κόμβοι που βρίσκονται εντός της ακτίνας «δράσης» του), ελέγχεται αν οι γείτονές του σε κατάσταση μόλυνσης P_2 μπορούν να τον μολύνουν (ο έλεγχος γίνεται για αυτούς τους κόμβους j που δεν είναι στην κατάσταση μόλυνσης P_2). Αν συμβεί μόλυνση επιπέδου P_2, τότε ελέγχεται από την αρχή ο επόμενος κόμβος j +1, διαφορετικά ελέγχεται αν κάποιοι από τους υπόλοιπους γείτονές του j, που βρίσκονται σε κατάσταση μόλυνσης P_1, μπορούν να τον μολύνουν σε επίπεδο μόλυνσης P_1. Αν μολυνθεί, τότε ελέγχεται ξανά αν μπορεί να μολυνθεί από τους γείτονές του, που βρίσκονται σε κατάσταση μόλυνσης P_2. Δηλαδή, ο δεδομένος κόμβος j, μπορεί να αλλάξει διαδοχικά καταστάσεις περισσότερες από μία φορά.

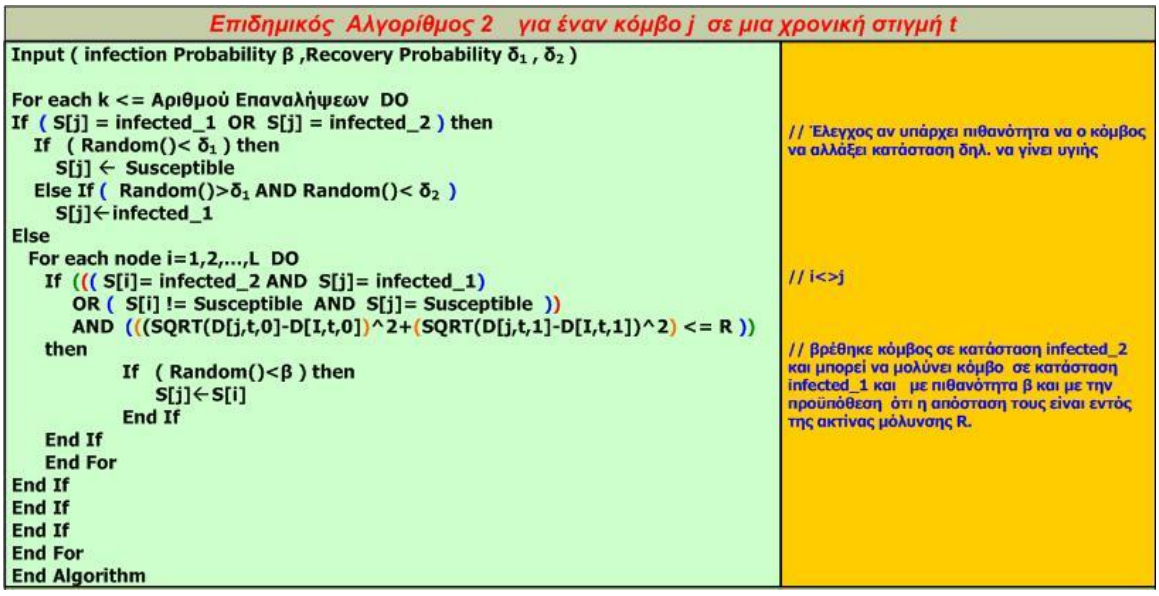


Επιδημικός Αλγορίθμος για έναν κόμβο j σε μια χρονική στιγμή t

```
Input ( infection Probability β ,Recovery Probability δ1 , δ2 )

If ( S[j] = infected_1 OR S[j] = infected_2 ) then

   If  ( Random()< δ1 ) then                                  // Έλεγχος αν υπάρχει πιθανότητα να ο κόμβος
      S[j] ← Susceptible                                      να αλλάξει κατάσταση δηλ. να γίνει υγιής
   Else If ( Random()>δ1 AND Random()< δ2 )
      S[j]←infected_1

Else
  For each node i=1,2,...,L DO                                // i<>j
    If ((( S[i]= infected_2 AND S[j]= infected_1)
       OR ( S[i] != Susceptible AND S[j]= Susceptible ))      // βρέθηκε κόμβος σε κατάσταση infected_2
       AND (((SQRT(D[j,t,0]-D[I,t,0])^2+(SQRT(D[j,t,1]-D[I,t,1])^2) <= R ))   και μπορεί να μολύνει κόμβο σε κατάσταση
    then                                                      infected_1 και με πιθανότητα β και με την
            If  ( Random()<β ) then                           προϋπόθεση ότι η απόσταση τους είναι εντός
               S[j]←S[i]                                      της ακτίνας μόλυνσης R.
            End If
    End If
    End For
End If
End If
End If
End Algorithm
```

Σχήμα 3.15 Παρουσίαση Επιδημικού Αλγορίθμου έκδοση 1η

Μια άλλη προσέγγιση του προτεινόμενου πολύ-επιδημικού αλγορίθμου είναι η μελέτη του, στην περίπτωση που οι κόμβοι του συστήματος είναι ακίνητοι. Στην περίπτωση αυτή κάθε κόμβος, που παραμένει ακίνητος, «προσπαθεί» να μολύνει τους γειτονικούς του, εφαρμόζοντας τον επιδημικό αλγόριθμο για μια σειρά επαναλήψεων (1000 βήματα). Ο αλγόριθμος για τις δύο καταστάσεις μόλυνσης 1 και 2, λαμβάνοντας υπόψη την ακινησία των κόμβων του συστήματος, παρουσιάζεται παρακάτω, στο **Σχήμα 3.16**:



Επιδημικός Αλγορίθμος 2 για έναν κόμβο j σε μια χρονική στιγμή t

```
Input ( infection Probability β ,Recovery Probability δ1 , δ2 )

For each k <= Αριθμού Επαναλήψεων DO
If ( S[j] = infected_1 OR S[j] = infected_2 ) then
   If  ( Random()< δ1 ) then                                  // Έλεγχος αν υπάρχει πιθανότητα να ο κόμβος
      S[j] ← Susceptible                                      να αλλάξει κατάσταση δηλ. να γίνει υγιής
   Else If ( Random()>δ1 AND Random()< δ2 )
      S[j]←infected_1
Else
  For each node i=1,2,...,L DO                                // i<>j
    If ((( S[i]= infected_2 AND S[j]= infected_1)
       OR ( S[i] != Susceptible AND S[j]= Susceptible ))      // βρέθηκε κόμβος σε κατάσταση infected_2
       AND (((SQRT(D[j,t,0]-D[I,t,0])^2+(SQRT(D[j,t,1]-D[I,t,1])^2) <= R ))   και μπορεί να μολύνει κόμβο σε κατάσταση
    then                                                      infected_1 και με πιθανότητα β και με την
            If  ( Random()<β ) then                           προϋπόθεση ότι η απόσταση τους είναι εντός
               S[j]←S[i]                                      της ακτίνας μόλυνσης R.
            End If
    End If
    End For
End If
End If
End If
End For
End Algorithm
```

Σχήμα 3.16 Παρουσίαση Επιδημικού Αλγορίθμου – έκδοση 2η για ακίνητους κόμβους

4. Περιγραφή του επιδημικού μοντέλου προσομοίωσης

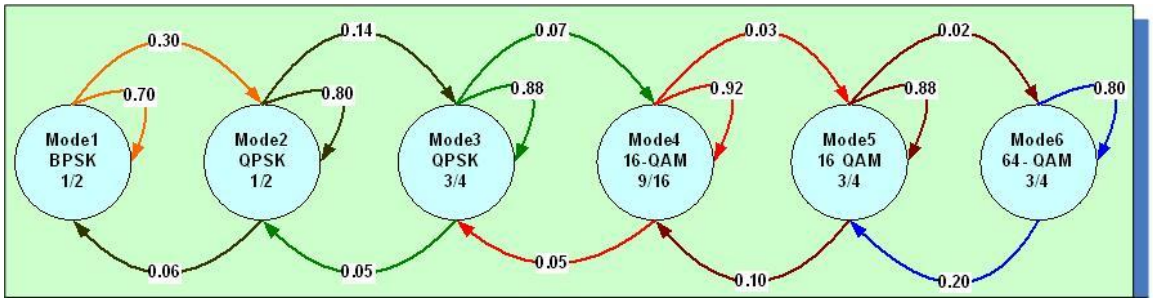
Στο μοντέλο προσομοίωσης που θα περιγράψουμε παρακάτω, εξετάζουμε την επιδημική διάδοση σε ένα δίκτυο 100 κόμβων. Θεωρούμε ότι ξεκινάμε με έναν «μολυσμένο» κόμβο δηλ. ο κόμβος που κατέχει μια πληροφορία, η οποία πληροφορία αντιστοιχεί σε ένα πλήθος πακέτων που κατοικούν σε μια ουρά προκειμένου να εκπεμφθούν. Όταν υπάρχουν και άλλοι κόμβοι σε απόσταση μικρότερη από την εμβέλεια του ασύρματου πομποδέκτη του κόμβου, και ο κόμβος έχει πακέτα στην ουρά (δηλ. είναι «μολυσμένος», τότε αυτός αποπειράται να εκπέμψει τα διάφορα πακέτα βάση του επιδημικού τρόπου μετάδοσης, με πιθανότητα επιτυχίας β (το β του επιδημικού αλγορίθμου). Με την απόπειρα εκπομπής τα περιεχόμενα της ουράς σβήνονται. Το καινοτόμο της παρούσας υλοποίησης βρίσκεται στο ότι το **β του επιδημικού αλγορίθμου είναι δυναμικά αναπροσαρμοζόμενο**, δηλ. ξεκινώντας από μια δεδομένη αρχική τιμή αναπροσαρμόζεται ανάλογα με το πλήθος σφαλμάτων λόγω του θορύβου στο ασύρματο κανάλι και του πλήθους των διπλότυπων πακέτων που προκαλούνται από την πολυδιαδρομικότητα. Κάθε κόμβος είναι εξοπλισμένος με ειδικό μηχανισμό ο οποίος αποτελείται από δυο επίπεδα ελέγχου λειτουργίας. Το 1ο επίπεδο καλείται Επίπεδο Κωδικοποίησης και το 2ο Επίπεδο καλείται Επίπεδο Ελέγχου της επιδημικής διάδοσης. Στο 1ο επίπεδο στο οποίο στο εξής θα αναφερόμαστε ως επίπεδο κωδικοποίησης, ο κάθε κόμβος αποφασίζει σύμφωνα με τα σφάλματα που λαμβάνει ή γενικότερα βάσει των σφαλμάτων που προκύπτουν από τις μεταδόσεις, και μπορεί να αλλάζει κωδικοποιητή.

Συνελικτικοί Κωδικοποιητές και η αντίστοιχοι ρυθμοί ανά διαμόρφωση και κωδικοποίηση

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
Modulation	BPSK	QPSK	QPSK	16-QAM	16-QAM	64-QAM
Coding rate R_c	1/2	1/2	3/4	9/16	3/4	3/4
Rate (bits/sym.)	0.50	1.00	1.50	2.25	3.00	4.50
a_n	274.7229	90.2514	67.6181	50.1222	53.3987	35.3508
g_n	7.9932	3.4998	1.6883	0.6644	0.3756	0.0900
γ_{pn}(dB)	-1.5331	1.0942	3.9722	7.7021	10.2488	15.9784

Πίνακας 4-1 Συνελικτικοί Κωδικοποιητές Προσομοίωσης

Συγκεκριμένα για τον έλεγχο του 1ου Επιπέδου, υλοποιήθηκε Μηχανή Πεπερασμένων Καταστάσεων (Finite State Machines - FSM) που παρουσιάζεται στο Σχήμα 4.1, για τους κωδικοποιητές του Πίνακα 4-1, έναν από τους οποίους μπορεί να έχει ενεργοποιημένον ο κάθε κόμβος, μια δεδομένη χρονική στιγμή.



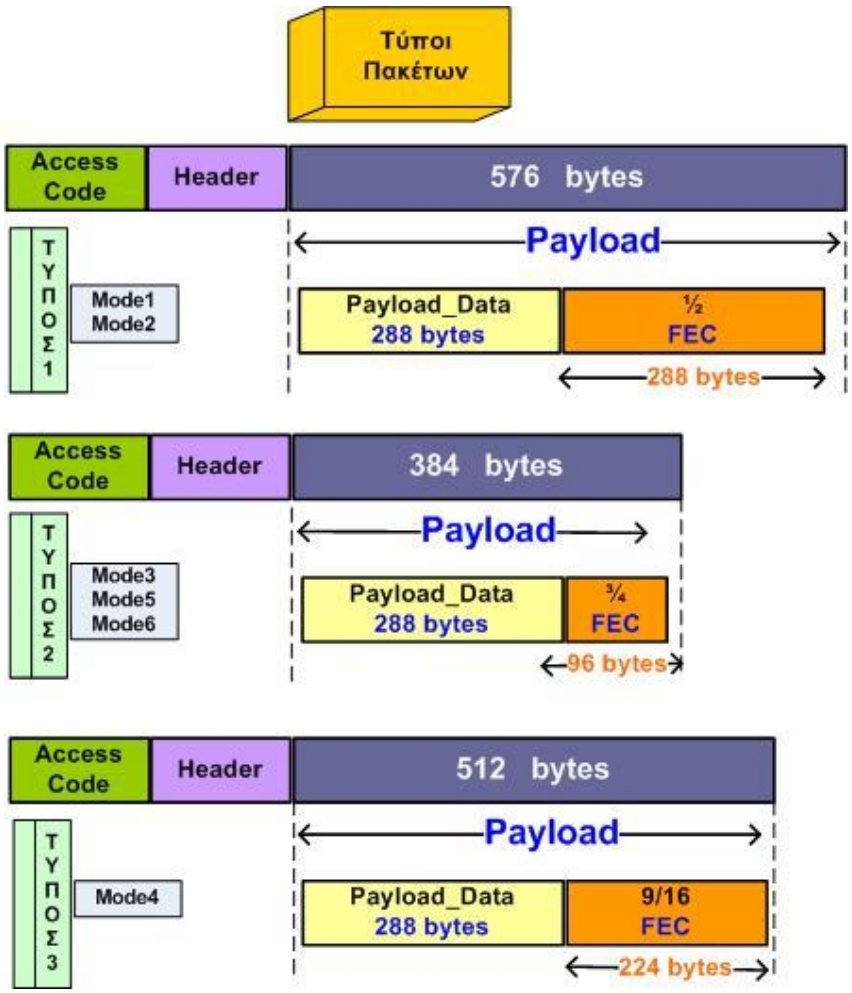
Σχήμα 4.1 Παρουσίαση Μηχανής Πεπερασμένων Καταστάσεων για τους 6 διαφορετικούς κωδικοποιητές

Για την κατασκευή της παραπάνω Μηχανής Πεπερασμένων Καταστάσεων υιοθετήσαμε τους ρυθμούς των κωδικοποιητών που αναφέρονται στον Πίνακα 4-1 και στο Paper [1]. Ο κάθε κωδικοποιητής περιγράφεται από την πιθανότητα παραμονής (εμμονής) στη χρήση του, καθώς και την πιθανότητα μετάβασης από αυτήν. Έχουμε μεταβάσεις μόνο μεταξύ κωδικοποιητών γειτονικών ρυθμών. Η πιθανότητα παραμονής στον κωδικοποιητή **Mode4** είναι αρκετά μεγάλη (της τάξης του 92% ενώ η πιθανότητα εξόδου, μικρή), ουσιαστικά αποτελεί μια καλή κατάσταση του καναλιού, από την οποία το σύστημα φεύγει δυσκολότερα από άλλες καταστάσεις. Οι μεταδόσεις που πραγματοποιούνται, καθώς ο κόμβος βρίσκεται σε συγκεκριμένη κατάσταση (κωδικοποιητής), χαρακτηρίζονται και από τον αντίστοιχο πλεονασμό.

Η λογική που θα χρησιμοποιηθεί είναι η εξής:

Ξεκινάμε αρχικά με ένα «μολυσμένο» κόμβο όπως προαναφέραμε, ουσιαστικά ο 1ος από τους 100 κόμβους αρχικοποιείται με τα διαθέσιμα πακέτα προς μετάδοση. Οι τύποι των πακέτων που συμμετέχουν στην προσομοίωση περιγράφονται στο Σχήμα 4-2 και σε όλα τα πακέτα, τα 288 bytes αποτελούν το ωφέλιμο και λόγω της αντίστοιχης κωδικοποίησης ½ FEC, ¾ FEC και 9/16 FEC το αντίστοιχο Payload των πακέτων διαμορφώνεται σε 576, 384 και 512 bytes αντίστοιχα. Στην συνέχεια ακολουθεί ο επιδημικός τρόπος μετάδοσης, σύμφωνα με τον οποίον ελέγχουμε επαναληπτικά τους

γείτονες του κόμβου αυτού, και σε αυτούς που βρίσκονται εντός της ακτίνας δράσης του, μεταδίδουμε τα πακέτα με κάποια πιθανότητα β, χωριστά για κάθε κόμβο.



Σχήμα 4.2 Τύποι πακέτων που συμμετέχουν στην προσομοίωση

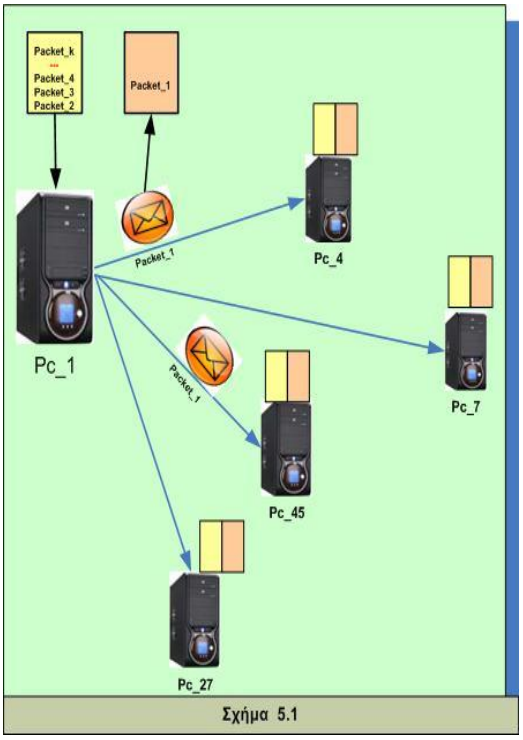
Ο κάθε κόμβος διαθέτει κατάλληλο μηχανισμό βάσει του οποίου ελέγχει αν το πακέτο που παρέλαβε είναι εσφαλμένο, και στην περίπτωση που είναι εσφαλμένο, καταγράφεται στο αρχείο καταγραφής εσφαλμένων πακέτων Errorfile.csv, σε διαφορετική περίπτωση ελέγχει αν το πακέτο αυτό το είχε παραλάβει στο παρελθόν, στην περίπτωση του που δεν το είχε παραλάβει το καταγράφει στα αρχεία infection.csv, infectionF2.csv, διαφορετικά καταγράφει το πακέτο στο αρχείο καταγραφής κίνησης των μολύνσεων infectionF2.csv (αποτελεί το αρχείο από το οποίο θα προκύψουν τα διπλότυπα). Για την περίπτωση που το πακέτο παρελήφθη για πρώτη φορά ο κόμβος έχει την υποχρέωση, να εκπέμψει το πακέτο με βάσει τον επιδημικό τρόπο μετάδοσης, αφού πρώτα κάνει χρήση του μηχανισμού 1ου επιπέδου που διαθέτει, και πάντα σε σχέση με

τις πληροφορίες που αντιλαμβάνεται ως προς το κανάλι μετάδοσης, αλλάζει ή διατηρεί το mode κωδικοποίησης, υπολογίζει το overhead και ελαττώνει κατά ένα τον αριθμό TTL που συνόδευε το πακέτο. Στην περίπτωση που ο αριθμός TTL είναι μηδέν το πακέτο δεν πρέπει να ξανά προωθηθεί και χάνεται, διαφορετικά ο κόμβος πρέπει να προωθήσει τα πακέτα που έχει στην ουρά του, βάσει του επιδημικού τρόπου μετάδοσης. Στην συνέχεια ενεργοποιείται ο μηχανισμός του 2ου επιπέδου που διαθέτει ο κόμβος και υπολογίζει το δυναμικό β βάση της παρακάτω σχέσης:

$$\beta=\beta_{\text{εκκίνησης}} - p_{\text{Διπλοτύπων}} * \beta_{\text{εκκίνησης}} + p_{\text{Σφαλμάτων}} * \beta_{\text{εκκίνησης}}$$

όπου το $p_{\text{Διπλοτύπων}}$ εκφράζει το ποσοστό των διπλοτύπων που έχει καταγράψει μέχρι στιγμής ο κόμβος και $p_{\text{Σφαλμάτων}}$ εκφράζει το ποσοστό των σφαλμάτων που έχει καταγράψει ο κόμβος. Το πακέτο προωθείται προς τους γειτονικούς κόμβους που βρίσκονται εντός της ακτίνας δράσης του, με πιθανότητα β δηλ. επαναλαμβάνεται η ίδια λογική του επιδημικού τρόπου μετάδοσης. Για να μη βρεθεί το β εκτός ορίων του διαστήματος [0,1], τίθεται ο ανελαστικός περιορισμός $0.15 \leq \beta \leq 0.95$.

Μια γενική εικόνα για τον αριθμό των διπλοτύπων που παραλαμβάνει ο εκάστοτε κόμβος καθώς και την τελική τιμή του δυναμικού β παρουσιάζεται στο αρχείο infectAll.csv.



Γενικά, στο μοντέλο που θα υλοποιήσουμε, κάθε κόμβος διαθέτει μηχανισμό ο οποίος αρχικά καταγράφει τα πακέτα που εισέρχονται στην ουρά του (κίτρινος φάκελος), και είναι διαθέσιμα προς αποστολή . Αρχικά ελέγχει αν η πιθανότητα λάθους είναι μικρότερη από το PER, τότε το πακέτο καταγράφεται στο αρχείο των εσφαλμένων πακέτων. Διαφορετικά ελέγχεται αν ο παραλήπτης έχει στο παρελθόν λάβει το ίδιο πακέτο κι αν όχι το προσθέτει στην ουρά των πακέτων του και το καταγράφει στο αρχείο των πακέτων που μεταδόθηκαν με επιτυχία (πορτοκαλί φάκελος), διαφορετικά το

Σχήμα 4.3 Παρουσίαση του επιδημικού τρόπου μετάδοσης και έναρξη με τον αρχικό κόμβο που διαθέτει τα πακέτα προς μετάδοση

πακέτο δεν προωθείται και καταγράφεται στο αρχείο διπλοτύπων.

Στα βήματα του παρακάτω αλγόριθμου αναλύεται η φιλοσοφία με την οποία κατασκευάστηκε το μοντέλο προσομοίωσης.

Στο κεφάλαιο 5 και στο Παράρτημα Β, παρουσιάζεται ο κώδικας και η προγραμματιστική φιλοσοφία που υιοθετήθηκε.

Στην έκδοση που επισυνάπτεται, το β υπολογίζεται δυναμικά για κάθε κόμβο και η υλοποίηση γίνεται με δυναμικό πίνακα. Οι συντελεστές βάρους που χρησιμοποιούνται, είναι το ποσοστό των διπλοτύπων και το ποσοστό των εσφαλμένων πακέτων, (πάντα σε σχέση με τα συνολικά πακέτα που παρέλαβε και που αντιλαμβάνεται ο εκάστοτε κόμβος). Με βάση αυτούς και το αρχικό β, που δίδεται ως παράμετρος κατά την εκτέλεση της προσομοίωσης, υπολογίζεται το δυναμικό β.

Βήμα 1 **myepidemicnomob.java**	Διαβάζει από το αρχείο των θέσεων των κόμβων τις θέσεις 100 κόμβων σε ένα πίνακα. Για την περίπτωση των σταθερών κόμβων διατηρεί τις 100 πρώτες θέσεις του αρχείου, αρχίζοντας να διαβάζει μετά από το σημείο εκκίνησης. Για την περίπτωση των κινητών κόμβων διαβάζει από το αρχείο των συντεταγμένων κάθε φορά τις επόμενες 100 γραμμές.
Βήμα 2 **myepidemicnomob.java**	Εκτελεί τον επιδημικό αλγόριθμο των βημάτων 3 - 7.
Βήμα 3 **myepidemicnomob.java**	Για κάθε κόμβο j=1,…, N βρίσκει τους γειτονικούς του (ευκλείδεια απόσταση), δηλαδή αυτούς που απέχουν λιγότερο από την ακτίνα r.
Βήμα 4 **myepidemicnomob.java**	Για κάθε γειτονικό κόμβο i=1,…, N (i not j) υπολογίζει μια πιθανότητα μόλυνσης-μετάδοσης η οποία αν είναι μεγαλύτερη από το b, και ο κόμβος j έχει πακέτα στην ουρά του, τότε ο κόμβος j μεταδίδει τα πακέτα του στον i (βήματα 5-7). Διαφορετικά πηγαίνει στον επόμενο κόμβο j.
Βήμα 5 **Enode.java**	Για κάθε πακέτο που περιέχει ο κόμβος j, επιλέγεται μια κωδικοποίηση και σύμφωνα με αυτήν υπολογίζεται η πιθανότητα σφάλματος μετάδοσης PER, $\mathrm{PER}_n(\gamma) \approx \begin{cases} 1, & \text{if } 0 < \gamma < \gamma_{pn}, \\ a_n \exp(-g_n \gamma), & \text{if } \gamma \geq \gamma_{pn} \end{cases}$ (5) σύμφωνα με τον τύπο 5 που υπάρχει στο paper [1] και το επιπλέον φορτίο του πακέτου (overhead) καθώς και μια τυχαία τιμή πιθανότητας λάθους.
Βήμα 6 **Enode.java**	Σε περίπτωση που η πιθανότητα λάθους είναι μικρότερη από το PER τότε το πακέτο καταγράφεται στο αρχείο των εσφαλμένων πακέτων. Διαφορετικά ελέγχεται αν ο παραλήπτης έχει στο παρελθόν λάβει το ίδιο πακέτο κι αν όχι το προσθέτει στην ουρά των πακέτων του και το καταγράφει στο αρχείο πακέτων

	που μεταδόθηκαν με επιτυχία, διαφορετικά το πακέτο απορρίπτεται και καταγράφεται στο αρχείο διπλοτύπων.
Βήμα 7 **Enode.java**	Όταν τελειώσει ο έλεγχος όλων των γειτόνων του κόμβου j, τα πακέτα καθαρίζονται από την ουρά πακέτων του κόμβου j.
Βήμα 8 **myepidemicnomob.java**	Η κλήση του επιδημικού αλγόριθμου επαναλαμβάνεται (βήματα 3 – 7) μέχρι σε μια κλήση να μη συμβεί ούτε μία μεταφορά πακέτου.

Πίνακας 4-1 Παρουσίαση βημάτων του αλγορίθμου του επιδημικού μοντέλου της προσομοίωσης

5. Υλοποίηση προσομοίωσης

Στην ενότητα αυτή περιγράφονται τα χαρακτηριστικά της συγκεκριμένης υλοποίησης, όπως η πλατφόρμα ανάπτυξης και τα προγραμματιστικά εργαλεία που επιλέχθηκαν προκειμένου να εκπονηθεί η παρούσα Διπλωματική Εργασία.

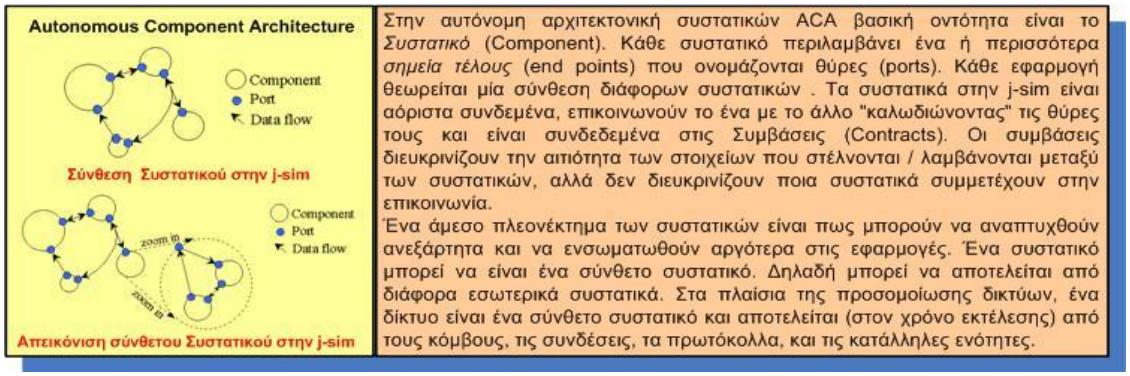
Σχήμα 5.1 Προγραμματιστικά Εργαλεία που χρησιμοποιήθηκαν

5.1 Πλατφόρμες και Προγραμματιστικά Εργαλεία

5.1.1 J-SIM

Η πλατφόρμα προσομοίωσης **J-SIM** χρησιμοποιήθηκε για τη μοντελοποίηση της κίνησης των 100 κόμβων του ασύρματου δικτύου, το περιβάλλον προσομοίωσης της βασίζεται στα συστατικά (components) και παρέχεται από το Distributed Computing Laboratory του Ohio State University. Η j-sim είναι μια εφαρμογή της Autonomous Component Architecture (ACA) της Java, που την καθιστά διαθέσιμη για σχεδόν οποιαδήποτε πλατφόρμα σήμερα. Η αρχική της έκδοση (2001) στόχευε στην έρευνα για ενσύρματα δίκτυα, αλλά το 2004 συμπεριέλαβε την επέκταση για τα ασύρματα / κινητά δίκτυα. Έτσι, γίνεται ένα ιδανικό εργαλείο προγραμματισμού για την ασύρματη δικτυακή μοντελοποίηση και την προσομοίωση δικτύων **[32]** .

Μια μικρή περιγραφή των βασικότερων συστατικών τους περιβάλλοντος προσομοίωσης J-Sim παρουσιάζεται στο **Σχήμα 5.2**.



Scripts	Η j-sim είναι ένα διπλό-γλωσσικό περιβάλλον: Η Java χρησιμοποιείται για να υλοποιήσει όλες τις κλάσεις και η Tcl χρησιμοποιείται ως συνδετική γλώσσα script για τη δημιουργία, τη διαμόρφωση, ή/και την προσομοίωση δικτύων ελέγχου στο χρόνο εκτέλεσης. Εκτός μερικών από τις τυποποιημένες εντολές της Tcl, η j-sim χρησιμοποιεί επίσης δύο σύνολα εκτεταμένων εντολών Tcl: Tcl/Java : καθορίζει την επέκταση της Java σε Tcl και είναι η "γέφυρα" μεταξύ τους. Οι εντολές Tcl/Java μπορούν να δημιουργήσουν και να έχουν πρόσβαση σε οποιαδήποτε αντικείμενα της Java μέσα από το περιβάλλον της Tcl. Εντολές συστημάτων RUV : Η έννοια των σύνθετων συστατικών σε j-sim καθιστά πιθανό να οργανώσει τα συστατικά σε μια συστατική ιεραρχία. Η ιεραρχία είναι παρόμοια με ένα σύστημα αρχείων σε ένα σύγχρονο λειτουργικό σύστημα υπολογιστή. Για να διευκολυνθεί η διαμόρφωση της προσομοίωσης δικτύων, έχει αναπτυχθεί το σύνολο εντολών RUV, δηλαδή, ένα σύνολο εικονικών εντολών συστημάτων χρόνου εκτέλεσης (RUntime Virtual system commands) για να χειριστεί ο προγραμματιστής τη συστατική ιεραρχία με τον ίδιο τρόπο όπως διάφορες εντολές του λειτουργικού συστήματος Unix κάνουν στο σύστημα αρχείων.
Συστατικό Mobility Model	Το συστατικό *MobilityModel* μιμείται τη κίνηση του ασύρματου κόμβου σ' ένα adhoc δίκτυο. Δύο πρότυπα κινητικότητας μπορούν να εφαρμοστούν: Το *Random Way Point* (RWP) πρότυπο κινητικότητας και το πρότυπο που βασίζεται σε καθορισμένη τροχιά. Εάν μια τροχιά έχει εγκατασταθεί, το συστατικό *MobilityModel* αυτόματα προσομοιώνει την κίνηση των κόμβων σύμφωνα μ' αυτήν, σε διαφορετική περίπτωση χρησιμοποιείται το *Random Way Point* πρότυπο κινητικότητας.
Συστατικό NodePosition Tracker	ΑυτόΑυτό το συστατικό διαιρεί το επίπεδο της προσομοιούμενης περιοχής σε πολλαπλάσιες υποπεριοχές.
Συστατικό FileComponent	Αυτό το συστατικό γράφει σε αρχείο τα εισερχόμενα δεδομένα από τους κόμβους. Δηλαδή στην περίπτωση της διπλωματικής τις συντεταγμένες των θέσεων των κόμβων.
Συστατικό Plotter	Με το συστατικό αυτό γίνεται γραφική έξοδος της κίνησης των κόμβων στην οθόνη προσομοίωσης tcl.
Random Waypoint μοντέλο κινητικότητας	Το Random Way Point λειτουργεί με την εξής διαδικασία: μετά από την εκκίνηση του συστατικού MobilityModel και εφόσον σ' αυτό δεν έχει καθοριστεί συγκεκριμένη τροχιά, αρχίζει να επιλέγει τυχαία έναν προορισμό στον οποίο θα μετακινηθεί ο κόμβος. Οι συντεταγμένες του επιλεγμένου προορισμού πρέπει να είναι μέσα στην περιοχή που έχει οριστεί για την κίνησή του και έχει προηγουμένως διευκρινιστεί στη λειτουργία setTopologyParameters. .Η ταχύτητα κίνησης speed είναι σταθερή και δεν μπορεί να είναι μεγαλύτερη από την ανώτατη ταχύτητα που είναι ήδη καθορισμένη στη λειτουργία setPosition. Κατά συνέπεια ο χρόνος που χρειάζεται ο κινητός κόμβος για να κινηθεί από την τρέχουσα θέση στον προορισμό του υπολογίζεται από τον τύπο: $\text{changeDestinationPeriod} = \frac{dist}{speed}$

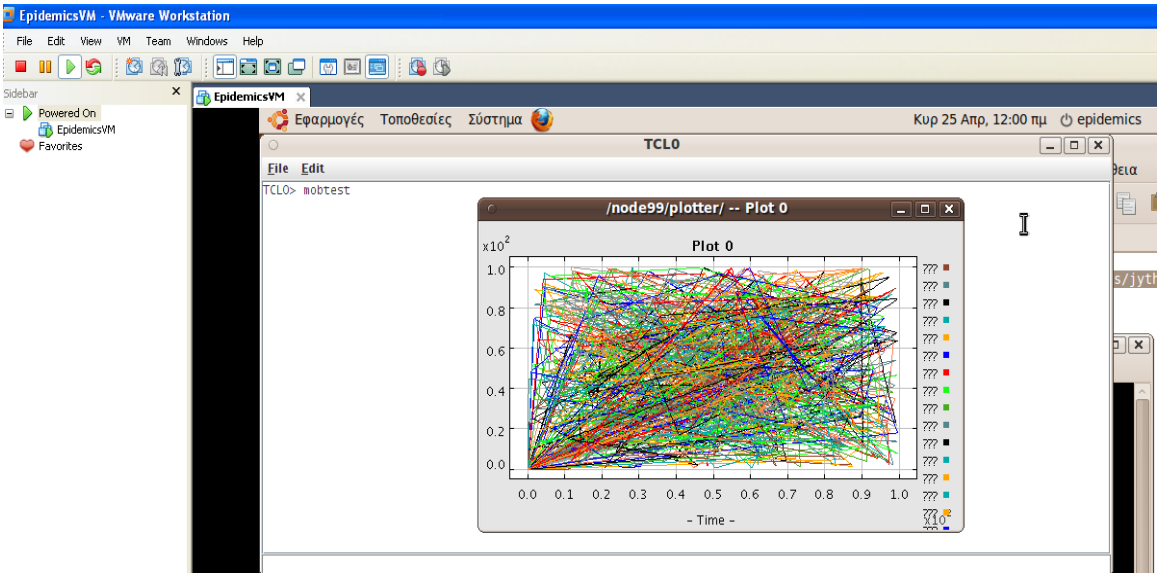
Σχήμα 5.2 Παρουσίαση των συστατικών στην j-sim

Στα πλαίσια της διπλωματικής αυτής εργασίας η j-sim χρησιμοποιήθηκε αποκλειστικά για τη δημιουργία των θέσεων των εκατό κόμβων για 1000 βήματα εκτέλεσης.

Αρχικά εγκαταστάθηκε μια εικονική μηχανή, η VMware Workstation και στη συνέχεια το λειτουργικό σύστημα Ubuntu. Στο προηγούμενο λειτουργικό σύστημα εγκαταστάθηκε το περιβάλλον του J-Sim και κατασκευάστηκε αρχείο tcl, που ονομάστηκε **n100.tcl** **[25]** και εφαρμόστηκε σ' αυτό το μοντέλο κινητικότητας Random Way Point [**A_1]** .

Στη συνέχεια ανοίγουμε το τερματικό από την επιφάνεια εργασίας και γράφουμε ***" java drcl.ruv.system n100.tcl"***. Στο καινούργιο πρόγραμμα που θα ανοίξει (άσπρο παράθυρο) αν εμφανιστεί κάποιο error ότι κάποιος φάκελος υπάρχει ήδη, το κλείνουμε και

βρίσκουμε από το ανοικτό παράθυρο του φακέλου jsim-1.3 το φάκελο που έγραφε και τον διαγράφουμε. Μετά ξανατρέχουμε την εντολή ***"java drcl.ruv.system n100.plot"*** στο Τερματικό. Αν το script μας δεν παρουσιάζει λάθη, το πρόγραμμα θα πρέπει να ανοίξει δύο παράθυρα, το ένα κεντρικό και το άλλο μικρότερο, όπου θα αρχίσει να σχηματίζεται το γράφημα όπως ακριβώς στο **Σχήμα 5.3**. Αφού περάσει λίγη ώρα και σιγουρευτούμε ότι το γράφημα έχει σίγουρα σταματήσει, κλείνουμε τα δύο παράθυρα και το αρχείο συντεταγμένων που παράγεται από την προσομοίωση ονομάστηκε **n100.plot** και είναι ένα αρχείο κειμένου το οποίο το μετονομάζουμε σε **n100.txt** .



Σχήμα 5.3 Στιγμιότυπο από την εκτέλεση του n100.tcl αρχείου στην πλατφόρμα J-Sim

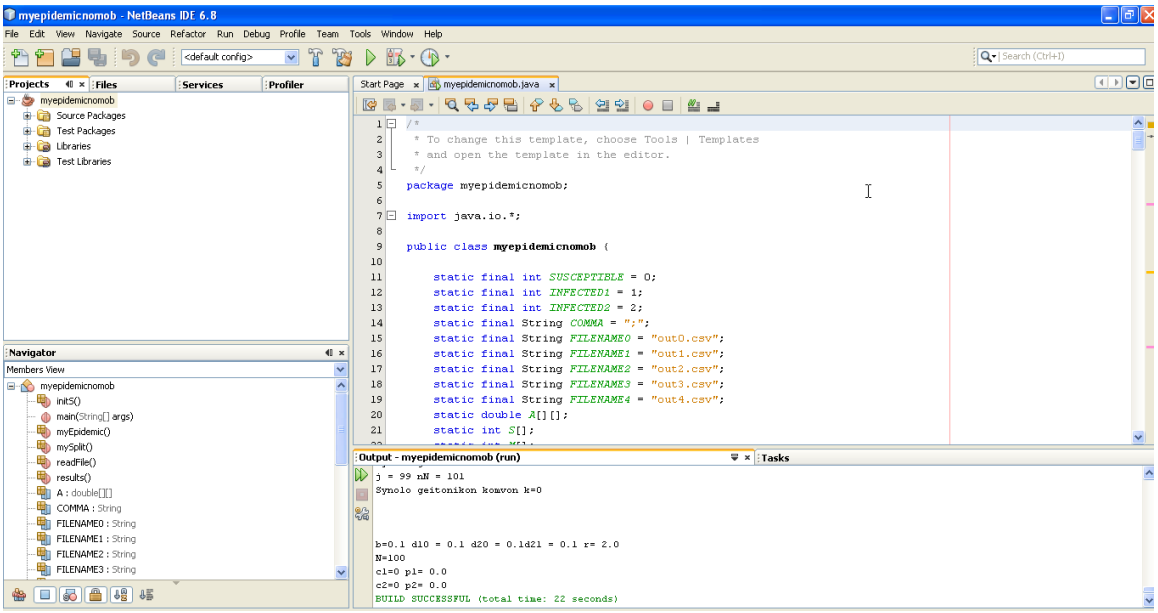
Στο παράρτημα [A_2] παρουσιάζεται το αρχείο **n100.tcl** και μικρό απόσπασμα του αρχείου συντεταγμένων **n100.txt [17].**

5.1.2 NetBeans

Το εργαλείο NetBeans αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης, με τακτικές ενημερώσεις από τη Sun, παράλληλα αποτελεί μια πλατφόρμα για την ανάπτυξη διαφόρων εφαρμογών που θα χρησιμοποιηθούν στα πλαίσια κάποιου δικτύου είτε θα χρησιμοποιηθούν ως ανεξάρτητες (Stand-alone) εφαρμογές. Το εργαλείο NetBeans υποστηρίζει διάφορες γλώσσες προγραμματισμού (Java, c, c++, PHP, Python, JavaScript). Παρέχει ένα πλήρες περιβάλλον ανάπτυξης εφαρμογών (IDE: Integrated

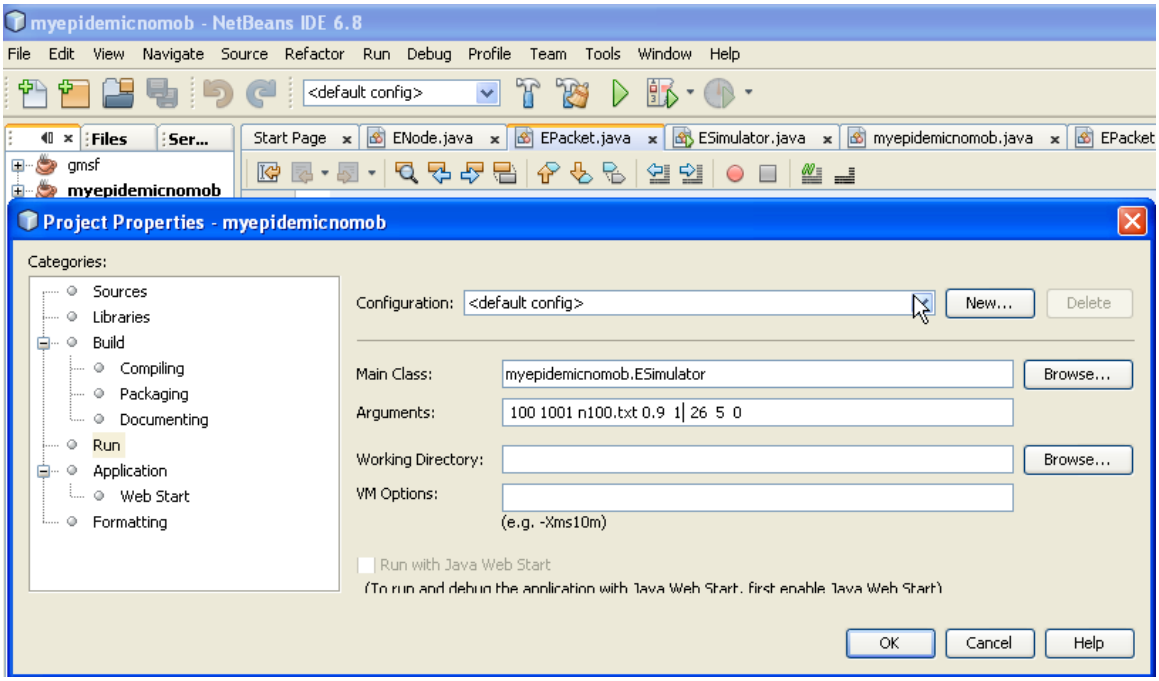
Development Environment), το οποίο αποτελεί ένα από τα δημοφιλέστερα IDE σε παγκόσμια κλίμακα για τη δημιουργία Προγραμμάτων / Συστημάτων σε Java. Γενικά παρέχει βοηθήματα στον προγραμματιστή τα οποία κάνουν ευκολότερη και αποδοτικότερη την ανάπτυξη των εφαρμογών. Τα βασικότερα από τα βοηθήματα αυτά αφορούν κυρίως τον Source Code Editor (Επισήμανση συντακτικού στον κώδικα, live επισήμανση λαθών και live parsing, pop up παράθυρα βοήθειας). Επίσης παρέχονται βοηθήματα για την κατασκευή GUI, αλλά και βοηθήματα διαχείρισης βάσεων δεδομένων. Η έκδοση που χρησιμοποιήθηκε για την υλοποίηση της προσομοίωσης είναι η ***NetBeans 6.8*** **[34]**.

Για να μπορέσουμε να γράψουμε προγράμματα σε Java, χρειάζεται να διαθέτουμε το ειδικό πρόγραμμα ανάπτυξης εφαρμογών Java SE Development Kit (JDK) 5.0 Update 19 (Version 1.5.0_19) ή JDK 6 Update 14 ή μεταγενέστερη έκδοση. Σε περίπτωση που δεν το διαθέτουμε μπορούμε να το βρούμε στον παρακάτω σύνδεσμο [19]. Στη συνέχεια κατεβάζουμε το NetBeans 6.8 από το σύνδεσμο [20] και το εγκαθιστάμε, το περιβάλλον του NetBeans φαίνεται στο παρακάτω **Σχήμα 5.4**.



Σχήμα 5.4 Περιβάλλον NetBeans μέσω του οποίου υλοποιείται η προσομοίωση

Στο περιβάλλον του NetBeans από το μενού **File** την επιλογή **Project Properties** και στην καρτέλα **Run** στην θέση **Arguments** τοποθετούμε τα ορίσματα που δέχεται η εφαρμογή προσομοίωσης. Στην παρακάτω εικόνα βλέπουμε τα ορίσματα που δέχεται η τελική έκδοση της εφαρμογής προσομοίωσης, το πλήθος αυτών είναι 8.



Σχήμα 5.5 Απεικόνιση ορισμάτων που δέχεται η εφαρμογή

Αναλυτικά τα ορίσματα είναι τα παρακάτω:

1. Το πλήθος των κόμβων.
2. Το set των συντεταγμένων με το οποίο θέλουμε να ξεκινήσει η υλοποίηση του αλγορίθμου.
3. Το αρχείο που έχει τις συντεταγμένες των κόμβων.
4. Το β, ο ρυθμός μόλυνσης με τον οποίο ξεκινά να μολύνει ο κόμβος
5. Η παράμετρος της κινητικότητας, με την τιμή **0** για σταθερούς κόμβους και για την τιμή **1** κινητοί κόμβοι.
6. Ο αριθμός TTL που είναι ενσωματωμένος στο κάθε πακέτο.
7. Η ακτίνα δράσης ή επικοινωνίας καθορίζει το μέγεθος της γειτονιάς του κόμβου.
8. Η παράμετρος καθορίζει αν το β θα είναι δυναμικό ή σταθερό, με την τιμή **0** αλλάζει το β δυναμικά βάση αυτών που αντιλαμβάνεται ο κόμβος και για την τιμή **1** διατηρείται σταθερό.

Τα αρχεία εξόδου που παράγει το πρόγραμμα, τα οποία καταγράφουν τη συνολική κίνηση του δικτύου και όλα τα σχετικά περί κωδικοποίησης είναι τα παρακάτω:

1. Errorfile: Καταγράφει όλα τα εσφαλμένα πακέτα που παρατηρήθηκαν βάση του Per και του mode της κωδικοποίησης που χρησιμοποιήθηκε.
2. Infection: Καταγράφει το σύνολο των πακέτων που παρελήφθησαν σωστά, το mode της κωδικοποίησης που χρησιμοποιήθηκε, το υπολογιζόμενο overhead, κλπ.
3. infectionF2: Καταγράφει όλη την κίνηση για όλους τους κόμβους, το overhead, το mode της κωδικοποίησης που χρησιμοποιήθηκε, από το αρχείο αυτό προκύπτουν τα διπλότυπα.
4. infectionAll: Καταγράφει για κάθε πακέτο ανά κόμβο το συνολικό αριθμό πακέτων που παρελήφθησαν σωστά πλην των εσφαλμένων, επίσης καταγράφει την τελευταία τιμή του β.

Στο **Παράρτημα Β** παρουσιάζονται τα Διαγράμματα των κλάσεων και ο κώδικας σε java που συμμετέχει στη διαδικασία της προσομοίωσης [18].

6. Συμπεράσματα προσομοίωσης και Μελλοντικές Επεκτάσεις

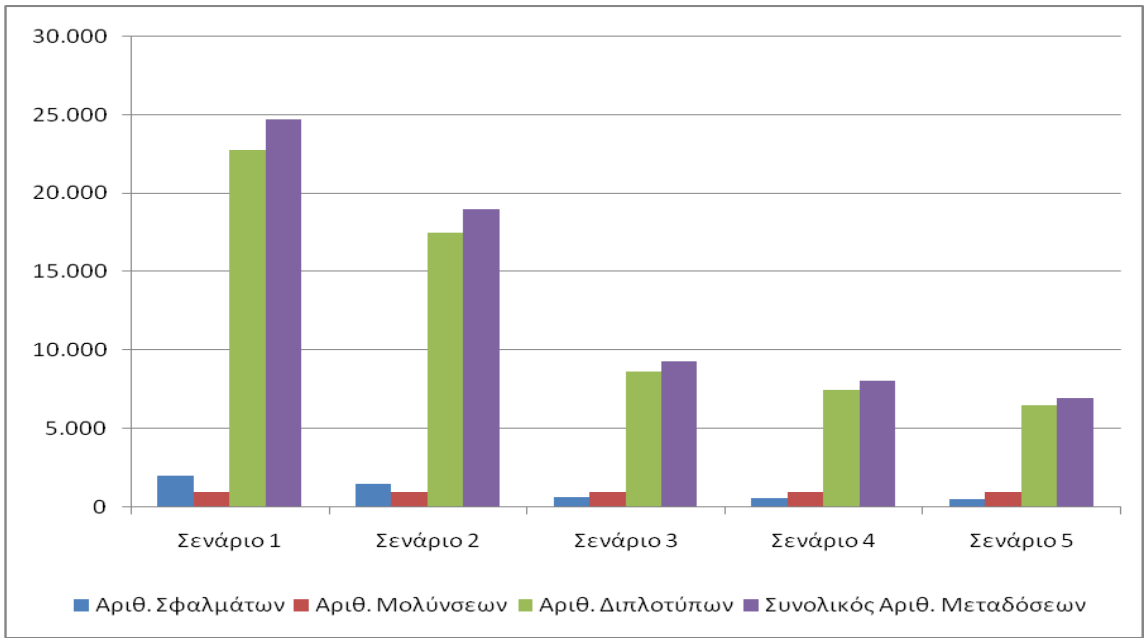
6.1 Συμπεράσματα

Στα κεφάλαια 4 και 5 περιγράφηκαν το Μοντέλο Προσομοίωσης, καθώς και τα Προγραμματιστικά Εργαλεία που χρησιμοποιήθηκαν. Στη συνέχεια θα αναλύσουμε τα διάφορα Σενάρια, καθώς και τα αποτελέσματα που προέκυψαν από την υλοποίηση της προσομοίωσης. Για όλες τις περιπτώσεις ο αριθμός των κόμβων είναι σταθερός και ανέρχεται σε 100. Τα αποτελέσματα χωρίζονται σε δυο μεγάλες κατηγορίες ως προς την κινητικότητα τους, σε αυτά **Με Κινητικότητα Κόμβων (ΜΚΚ)** και σε αυτά **Χωρίς Κινητικότητα Κόμβων (ΧΚΚ)**.

Τα παρακάτω αποτελέσματα προέκυψαν, από την υλοποίηση προσομοίωσης με κινητικότητα κόμβων και αφορούν 10 πακέτα, τα οποία διαθέτει ένας μόνο κόμβος και ξεκινά την προώθηση αυτών, βάσει του επιδημικού τρόπου μετάδοσης, για τα διάφορα Σενάρια που ακολουθούν παρακάτω. Αξίζει να αναφερθεί ότι ο αριθμός πακέτων μπορεί να παραχθεί από την εφαρμογή δυναμικά, και τα αποτελέσματα που προέκυψαν για τις διάφορες τιμές του συνόλου των πακέτων, ήταν ανάλογα με αυτά που θα παρουσιάσουμε. Οι αριθμοί στους παρακάτω πίνακες αντιστοιχούν σε αριθμούς πακέτων.

Αριθμός Πακέτων προς Αποστολή από αρχικό κόμβο : 10								
ΜΚΚ		b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων
Σταθερό β	Σενάριο 1	0,9	5	26	1.991	990	22.768	24.759
Σταθερό β	Σενάριο 2	0,7	5	26	1.485	990	17.524	19.009
Δυναμικό β	Σενάριο 3	0,9	5	26	673	990	8.637	9.310
Δυναμικό β	Σενάριο 4	0,7	5	26	580	990	7.485	8.065
Δυναμικό β	Σενάριο 5	0,9	5	20	507	990	6.476	6.983

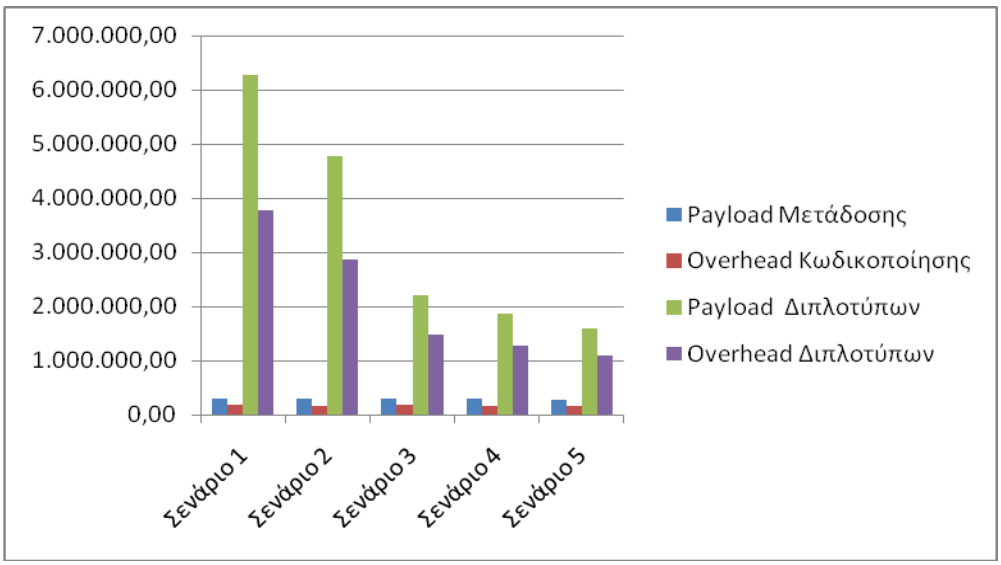
Πίνακας 6-1 Πίνακας καταγραφής αποτελεσμάτων κατηγορίας ΜΚΚ για τα διάφορα Σενάρια



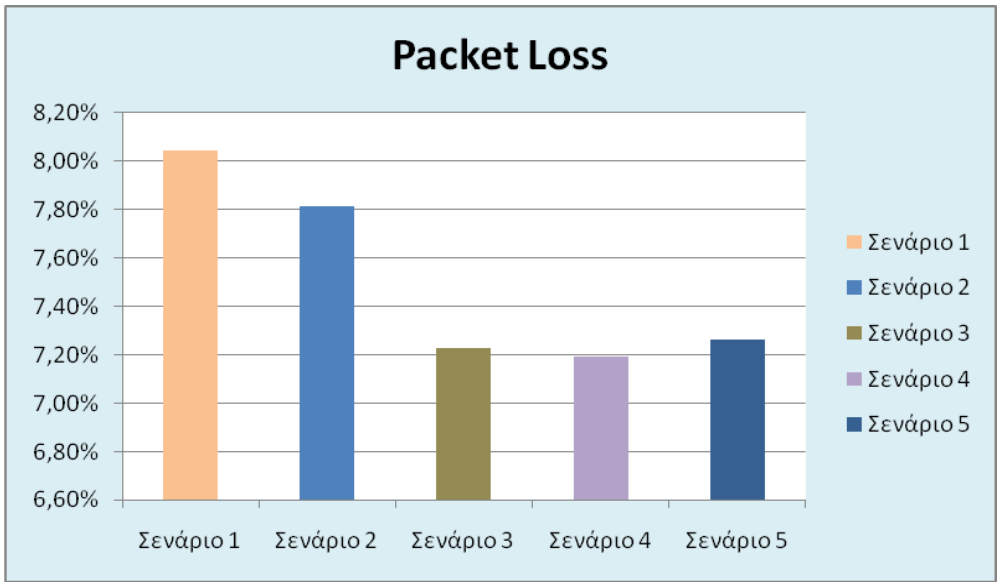
Σχήμα 6.1 Γράφημα που απεικονίζει τα αποτελέσματα των πέντε Σεναρίων (MKK)

MKK		b	R	TTL	Payload Μετάδοσης	Overhead Κωδικοποίησης	Σύνολο Μετάδοσης	Payload Διπλοτύπων	Overhead Διπλοτύπων	Συνολο Διπλοτύπων
Σταθερό β	Σενάριο 1	0,9	5	26	285.120,00	172.288,00	457.408,00	6.272.064,00	3.778.592,00	10.050.656,00
Σταθερό β	Σενάριο 2	0,7	5	26	285.120,00	168.704,00	453.824,00	4.761.792,00	2.866.944,00	7.628.736,00
Δυναμικό β	Σενάριο 3	0,9	5	26	285.120,00	174.528,00	459.648,00	2.202.336,00	1.477.984,00	3.680.320,00
Δυναμικό β	Σενάριο 4	0,7	5	26	285.120,00	170.048,00	455.168,00	1.870.560,00	1.280.224,00	3.150.784,00
Δυναμικό β	Σενάριο 5	0,9	5	20	285.120,00	165.504,00	445.440,00	1.585.152,00	1.094.080,00	2.679.232,00

Πίνακας 6-2 Πίνακας με τα διάφορα Overhead για κάθε Σενάριο (MKK)



Σχήμα 6.2 Γράφημα που απεικονίζει τα διάφορα Overhead (MKK)



Σχήμα 6.3 Γράφημα που απεικονίζει τις Απώλειες πακέτων για τα διάφορα Σενάρια (MKK)

Θα αναλύσουμε τις παρακάτω χρήσιμες Μετρικές στο Επιδημικό Μοντέλο:

1. Μετρική No 1 → Μέσο παραγόμενο overhead
2. Μετρική No 2→ (Αριθμός Διπλοτύπων)/(Αριθμός γνήσιων πακέτων)
3. Μετρική No 3→ (Αριθμός Μολύνσεων)/(Μέσο overhead)
4. Μετρική No 4→ (Αριθμός Μολύνσεων)/(Αριθμός Διπλοτύπων)

Στους παρακάτω πίνακες αναλύονται οι παραπάνω μετρικές για το μοντέλο MKK.

				Σε bytes				
MKK	b	R	TTL	Overhead Κωδικοποίησης	Overhead Διπλοτύπων	Overhead Εσφαλμένων	Συνολικό Overhead	Average overhead
Σενάριο 1	0,9	5	26	172.288	3.778.592	326.624	4.277.504	172,76562
Σενάριο 2	0,7	5	26	168.704	2.866.944	244.832	3.280.480	172,5751
Σενάριο 3	0,9	5	26	174.528	1.477.984	105.184	1.757.696	188,79656
Σενάριο 4	0,7	5	26	170.048	1.280.224	93.824	1.544.096	191,45642
Σενάριο 5	0,9	5	20	165.504	1.094.080	82.015	1.341.599	192,12359

Πίνακας 6-3 Πίνακας καταγραφής Overhead του μοντέλου MKK

ΜΚΚ	b	R	TTL	Μετρική Νρ. 2	Μετρική Νρ. 3	Μετρική Νρ. 4
Σενάριο 1	0,9	5	26	23,00	5,73	0,04
Σενάριο 2	0,7	5	26	17,70	5,74	0,06
Σενάριο 3	0,9	5	26	8,72	5,24	0,11
Σενάριο 4	0,7	5	26	7,56	5,17	0,13
Σενάριο 5	0,9	5	20	6,54	5,15	0,15

Πίνακας 6-4 Πίνακας καταγραφής ορισμένων μετρικών

Στο Σενάριο 5, που παρουσιάστηκε παραπάνω, μπορούμε εύκολα να διαπιστώσουμε ότι αποτελεί την καλύτερη περίπτωση και αυτό γιατί προκαλεί λιγότερη κίνηση στο δίκτυο, με λιγότερο overhead και με μεγαλύτερη μόλυνση και μικρότερο ποσοστό σφαλμάτων.

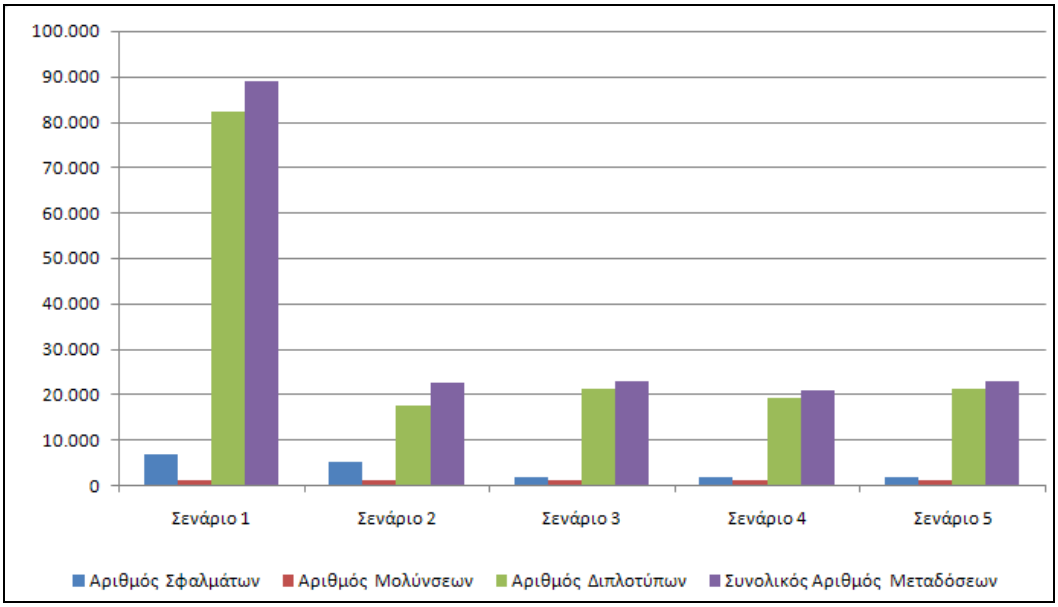
Ο τρόπος με τον οποίο παράγεται το συγκεκριμένο σενάριο, κάνοντας χρήση του δυναμικού β, καθώς και μικρού αριθμού TTL, αποτελεί και την καινοτομία της συγκεκριμένης διπλωματικής, για την περίπτωση του ΜΚΚ καθώς και για την ΧΚΚ όπως θα διαπιστωθεί παρακάτω.

Ανάλογα συμπεράσματα προκύπτουν και για το μοντέλο κινητικότητας ΧΚΚ. Για να γίνουν καλύτερα κατανοητά, τα συμπεράσματα του, θα παρουσιάσουμε τα 5 Σενάρια που αφορούν και σε σταθερούς κόμβους, δηλ. κόμβους χωρίς κινητικότητα.

Στα πλαίσια της προσομοίωσης για τους σταθερούς κόμβους, η εφαρμογή προσομοίωσης εκτελέστηκε τρεις φορές, κάθε φορά με διαφορετικό σημείο εκκίνησης που δίδεται μέσω του $2^{ου}$ ορίσματος που δέχεται η εφαρμογή. Πιο συγκεκριμένα, ως τιμές εκκίνησης, οι οποίες παράλληλα αποτελούν και τιμές της $2^{ης}$ παραμέτρου που δέχεται η εφαρμογή και είναι σχετικές με την καταγραφή των θέσεων για τους εκατό κόμβους, χρησιμοποιήθηκαν οι 101, 1001 και 2001 αντίστοιχα και προέκυψαν οι θέσεις για τους 100 κόμβους με μέσες αποστάσεις τις 1,6 m , 8,6 m και 16 m αντίστοιχα.

Αριθμός Πακέτων προς Αποστολή από αρχικό κόμβο : 10									
ΧΚΚ	Τιμή Εκκίνησης	Σενάριο	b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων
Σταθερό β	101	Σενάριο 1	0,9	5	26	6.632	990	82.299	88.931
Σταθερό β	101	Σενάριο 2	0,7	5	26	5.085	990	17.524	22.609
Δυναμικό β	101	Σενάριο 3	0,9	5	26	1.722	990	21.205	22.927
Δυναμικό β	101	Σενάριο 4	0,7	5	26	1.777	990	19.013	20.790
Δυναμικό β	101	Σενάριο 5	0,9	5	20	1.647	990	21.007	22.654

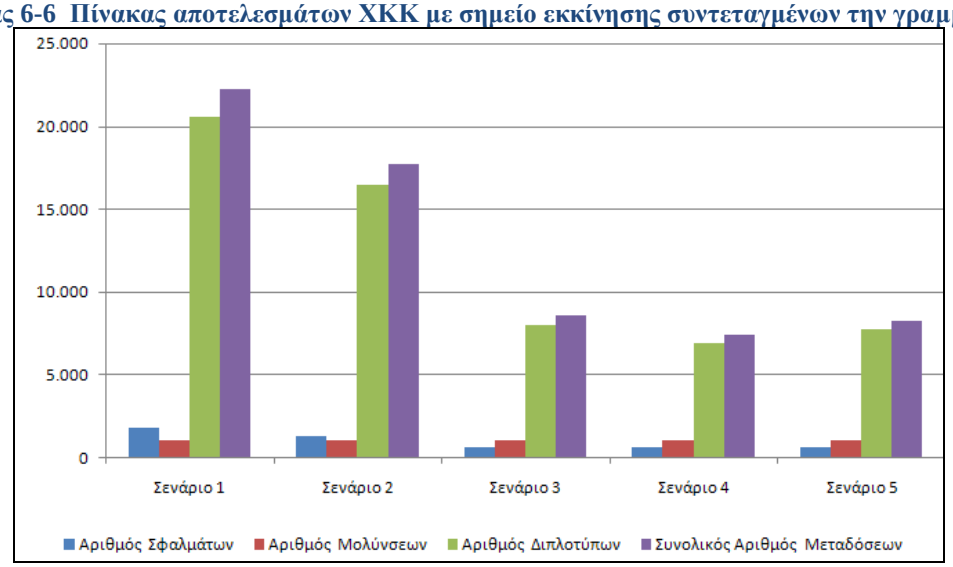
Πίνακας 6-5 Πίνακας αποτελεσμάτων ΧΚΚ με σημείο εκκίνησης συντεταγμένων την γραμμή 101



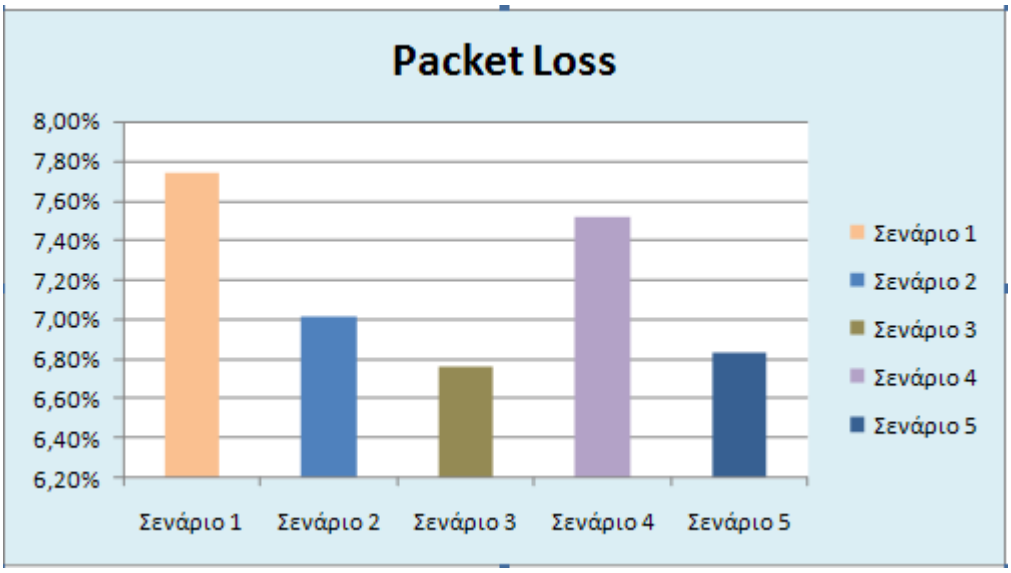
Σχήμα 6.4 Γράφημα παρουσίασης Σεναρίων για XKK με τιμή εκκίνησης συντεταγμένων την γραμμή 101

Αριθμός Πακέτων προς Αποστολή από αρχικό κόμβο : 10									
XKK	Τιμή Εκκίνησης	Σενάριο	b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων
Σταθερό β	1001	Σενάριο 1	0,9	5	26	1.720	990	20.519	22.239
Σταθερό β	1001	Σενάριο 2	0,7	5	26	1.239	990	16.424	17.663
Δυναμικό β	1001	Σενάριο 3	0,9	5	26	578	990	7.975	8.553
Δυναμικό β	1001	Σενάριο 4	0,7	5	26	557	990	6.856	7.413
Δυναμικό β	1001	Σενάριο 5	0,9	5	20	564	975	7.687	8.251

Πίνακας 6-6 Πίνακας αποτελεσμάτων XKK με σημείο εκκίνησης συντεταγμένων την γραμμή 1001



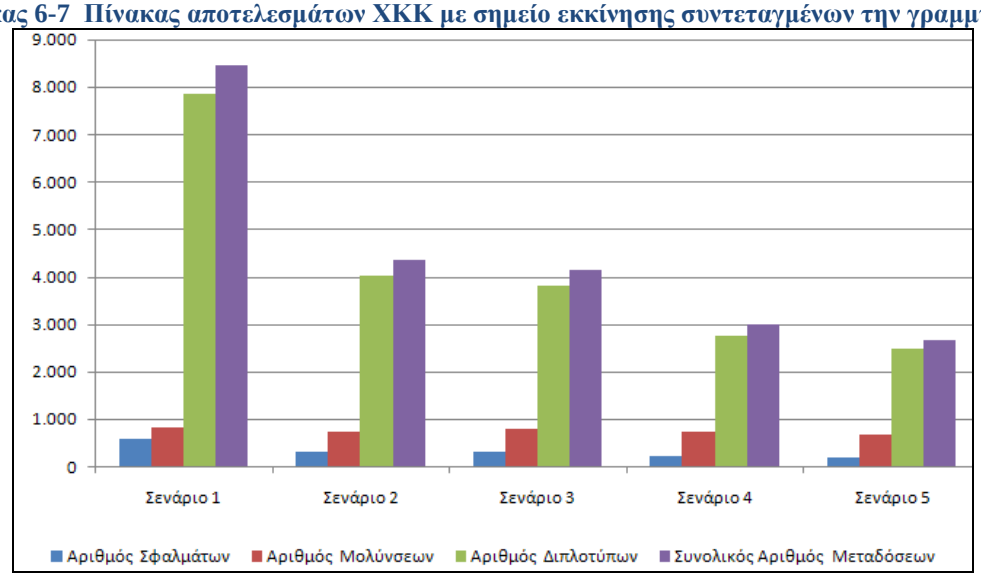
Σχήμα 6.5 Γράφημα παρουσίασης αποτελεσμάτων Σεναρίων για XKK με τιμή εκκίνησης συντεταγμένων την γραμμή 1001



Σχήμα 6.6 Γράφημα που απεικονίζει τις Απώλειες πακέτων για τα διάφορα Σενάρια (ΧΚΚ)

Αριθμός Πακέτων προς Αποστολή από αρχικό κόμβο : 10									
ΧΚΚ	Τιμή Εκκίνησης	Σενάριο	b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων
Σταθερό β	2001	Σενάριο 1	0,9	5	26	587	829	7.857	8.444
Σταθερό β	2001	Σενάριο 2	0,7	5	26	317	737	4.020	4.337
Δυναμικό β	2001	Σενάριο 3	0,9	5	26	315	794	3.809	4.124
Δυναμικό β	2001	Σενάριο 4	0,7	5	26	216	722	2.759	2.975
Δυναμικό β	2001	Σενάριο 5	0,9	5	20	168	667	2.491	2.659

Πίνακας 6-7 Πίνακας αποτελεσμάτων ΧΚΚ με σημείο εκκίνησης συντεταγμένων την γραμμή 2001



Σχήμα 6.7 Γράφημα παρουσίασης Σεναρίων για ΧΚΚ με τιμή εκκίνησης συντεταγμένων την γραμμή 2001

Για την περίπτωση των σταθερών κόμβων και με σημείο εκκίνησης για τις συντεταγμένες την 101 γραμμή, υπολογίστηκαν τα διάφορα overhead που προέκυψαν και παρουσιάζονται στον Πίνακα 6.8. Στα Σενάρια 3,4 και 5, που αφορούν το δυναμικά αναπροσαρμοζόμενο β, επιτυγχάνουμε πολύ μικρότερο overhead. Ανάλογα συμπεράσματα μπορούν να προκύψουν και στις άλλες περιπτώσεις, που αφορούν στους κόμβους χωρίς κινητικότητα.

ΧΚΚ		Σημείο Εκκίνησης	b	r	ttl	Overhead Κωδικοποίησης	Overhead Διπλοτύπων	Overhead Εσφαλμένων
Σταθερό β	Σενάριο 1	101	0,9	5	26	159.936	13.555.168	1.098.336
Σταθερό β	Σενάριο 2	101	0,7	5	26	169.664	10.570.848	820.256
Δυναμικό β	Σενάριο 3	101	0,9	5	26	162.816	3.550.368	283.968
Δυναμικό β	Σενάριο 4	101	0,7	5	26	165.056	3.101.728	302.624
Δυναμικό β	Σενάριο 5	101	0,9	5	20	170.560	3.526.944	271.328

Πίνακας 6-8 Πίνακας καταγραφής Overhead (ΧΚΚ) με σημείο εκκίνησης συντεταγμένων την γραμμή 101

Τα σημαντικότερα συμπεράσματα που αποκομίσαμε από την εργασία αυτή είναι τα παρακάτω :

- **Με τη χρήση δυναμικά αναπροσαρμοζόμενου β**, επιτυγχάνουμε μεγαλύτερο ποσοστό μόλυνσης, με λιγότερες εκπομπές και διπλότυπα. Αυτό φαίνεται αν συγκρίνουμε τα σενάρια 1,3,5 και 2 με 4, για κάθε περίπτωση που παρουσιάστηκε.
- Σε ένα τέτοιο μηχανισμό, σημαντικό ρόλο παίζουν παράμετροι όπως **αρχικό β**, η **μέση απόσταση μεταξύ κόμβων** σε σχέση με την εμβέλεια της ραδιοεπικοινωνίας, η **κινητικότητα** και το **TTL**.
- Συγκεκριμένα, η μόλυνση διευκολύνεται (γίνεται πιο αποτελεσματικά), όταν **υπάρχει κινητικότητα**, όταν **η απόσταση μεταξύ των κόμβων δεν είναι μεγάλη σε σχέση με την εμβέλεια**, όταν το **TTL είναι μεγάλο**.

Η απόδοση (efficiency) ενός σχήματος επιδημικής διάδοσης, μπορεί να μετρηθεί ενδεικτικά με το λόγο: → A = Πλήθος μολύνσεων/πλήθος εκπομπών.

Ορίζουμε τα παρακάτω:

$$E_0 = L_0 * T_0 * M_0 = \left(L_{data} + L_{overhead,0}\right) * T_0 * M_0$$

$$E_{dyn} = \langle L \rangle * T_0 * \langle M \rangle = \left(L_{data} + \langle L_{overhead} \rangle\right) * T_0 * \langle M \rangle$$

E_0 = Ενέργεια που δαπανάται για σταθερό β
L_0 = Μήκος Πακέτου για σταθερό β
T_0 =Ενέργεια που δαπανάται για την αποστολή 1 byte
M_0 = Συνολικός αριθμός Μεταδόσεων για σταθερό β
L_{data} = Μήκος δεδομένων ενός πακέτου →288 bytes
$L_{overhead,0}$ =Μήκος του αντίστοιχου overhead που εισάγεται από την Κωδικοποίηση..

E_{dyn} = Ενέργεια που δαπανάται για δυναμικό β
L= Μήκος Πακέτου για δυναμικό β
T_0 =Ενέργεια που δαπανάται για την αποστολή 1 byte
M = Συνολικός αριθμός Μεταδόσεων για δυναμικό β
L_{data} = Μήκος δεδομένων ενός πακέτου →288 bytes
$L_{overhead}$ =Μήκος του αντίστοιχου overhead που εισάγεται από την Κωδικοποίηση..

Εξοικονόμηση Ενέργειας (Energy Saved) → $E\_S = (E_0 - E_{dyn})/E_0$

- Παρατηρούμε, λοιπόν, ότι **επιτυγχάνεται καλύτερη απόδοση καθώς και εξοικονόμηση ενέργειας**, διότι χρειαζόμαστε λιγότερες εκπομπές για να πετύχουμε μόλυνση ενός αριθμού κόμβων.

Μετά από υπολογισμούς παραπάνω συνοψίζονται στον ακόλουθο πίνακα:

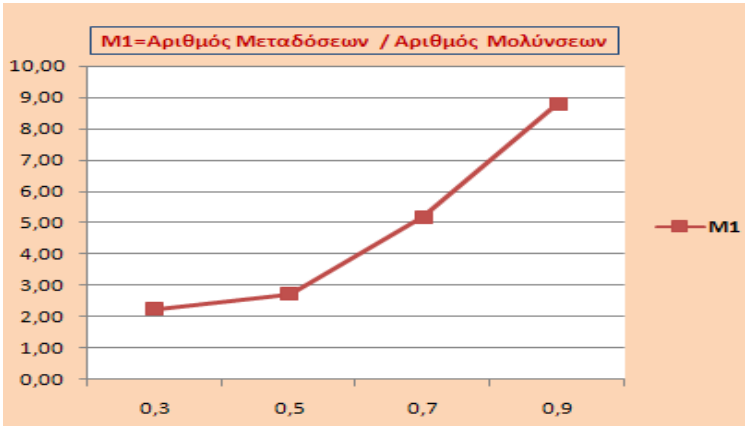
ΜΚΚ	Τιμή Εκκίνησης	Σενάριο	b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων	A	Μέσο Overhead	E_S
Σταθερό β	1001	Σενάριο 1	0,9	5	26	1.991	990	22.768	24.759	0,04	172,77	
Σταθερό β	1001	Σενάριο 2	0,7	5	26	1.485	990	17.524	19.009	0,05	172,58	
Δυναμικό β	1001	Σενάριο 3	0,9	5	26	673	990	8.637	9.310	0,11	188,80	0,61
Δυναμικό β	1001	Σενάριο 4	0,7	5	26	580	990	7.485	8.065	0,12	191,46	0,56
Δυναμικό β	1001	Σενάριο 5	0,9	5	20	507	990	6.476	6.983	0,14	192,12	0,71

Πίνακας 6-9 Πίνακας αποτελεσμάτων (ΜΚΚ) με ενσωματωμένες ειδικές μετρικές

Τα συμπεράσματα που αναφέραμε παραπάνω επιβεβαιώνονται και με την δημιουργία **Εναλλακτικών Σεναρίων του Σεναρίου 5** που ακολουθούν μέσο της **μετρικής M1**.

ΜΚΚ	Τιμή Εκκίνησης	Σενάριο	b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων	M1
Δυναμικό β	1001	Σενάριο 5_1	0,3	5	20	120	778	1.626	1.746	2,24
Δυναμικό β	1001	Σενάριο 5_2	0,5	5	20	124	834	2.138	2.262	2,71
Δυναμικό β	1001	Σενάριο 5_3	0,7	5	20	311	923	4.468	4.779	5,18
Δυναμικό β	1001	Σενάριο 5	0,9	5	20	630	980	7.986	8.616	8,79

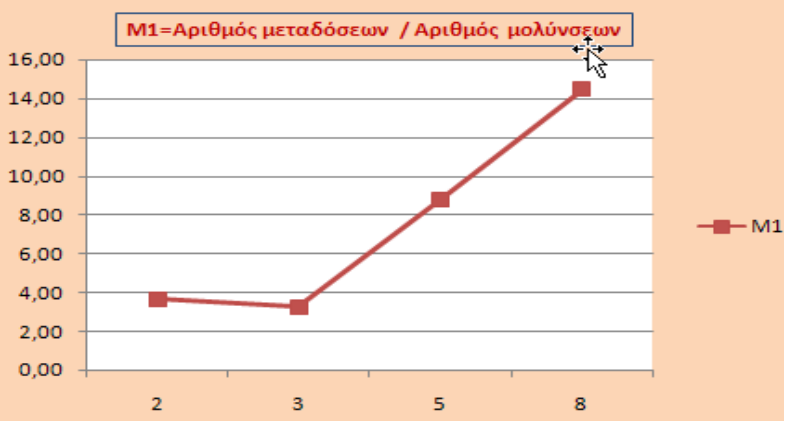
Πίνακας 6-10 Πίνακας αποτελεσμάτων για σενάρια με μεταβλητό δυναμικό β και σταθερή ακτίνα και σταθερό TTL



Σχήμα 6.8 Γράφημα παρουσίασης της μετρικής M1 για Σενάρια με σταθερή ακτίνα και διαφορετικές τιμές του δυναμικά μεταβαλλόμενου β

ΜΚΚ	Τιμή Εκκίνησης	Σενάριο	b	R	TTL	Αριθμός Σφαλμάτων	Αριθμός Μολύνσεων	Αριθμός Διπλοτύπων	Συνολικός Αριθμός Μεταδόσεων	M1
Δυναμικό β	1001	Σενάριο 5_4	0,9	2	20	106	467	1.592	1.698	3,64
Δυναμικό β	1001	Σενάριο 5_5	0,9	3	20	157	804	2.473	2.630	3,27
Δυναμικό β	1001	Σενάριο 5	0,9	5	20	630	980	7.986	8.616	8,79
Δυναμικό β	1001	Σενάριο 5_6	0,9	8	20	1.057	990	13.242	14.299	14,44

Πίνακας 6-11 Πίνακας αποτελεσμάτων για σενάρια με σταθερό δυναμικό β για διάφορες τιμές της ακτίνας δράσης των κόμβων και με μέση απόσταση των κόμβων 8.6



Σχήμα 6.9 Γράφημα παρουσίασης της μετρικής M1 για Σενάρια με διαφορετική ακτίνα

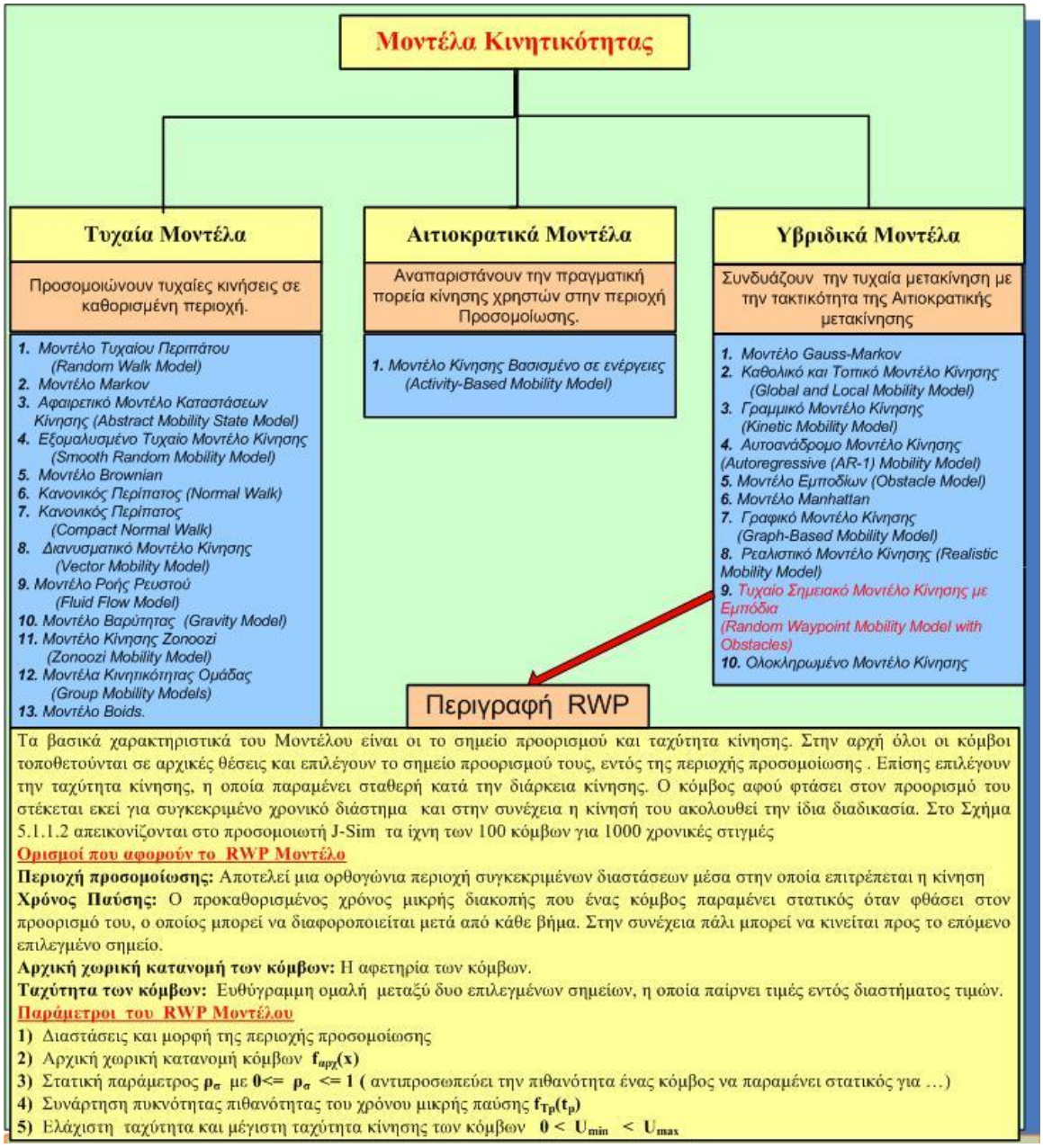
6.2 Μελλοντικές Επεκτάσεις

- Η εκτέλεση των αντιστοίχων Σεναρίων κάτω από διαφορετικές συνθήκες δικτύου (διαφορετικό μοντέλο κινητικότητας, άλλη πυκνότητα, …).
- Επέκταση σε διαφορετικά επιδημικά μοντέλα (SIS, SIR, …), με περισσότερες καταστάσεις για τους κόμβους όπως μερική μόλυνση, ολική μόλυνση .
- Διαφορετικοί ρόλοι για τους κόμβους όπως απενεργοποίηση του αρχικού κόμβου μετά από κάποια χρονική στιγμή.
- Διαφορετικοί τρόποι υπολογισμού για το αναπροσαρμοζόμενο β ώστε να μπορεί να επιτευχθούν υψηλότερα κέρδη ενέργειας όπως:
  1. Ο υπολογισμός του δυναμικού β να προκύπτει από πίνακα δύο διαστάσεων.
  2. Ο υπολογισμός να προκύπτει από Σιγμοειδή συνάρτηση.
  3. Ο υπολογισμός να προκύπτει από συνάρτηση με συντελεστές βαρύτητες τους $\beta_{\text{εκκίνησης}}$, αριθμό σφαλμάτων και τον αριθμό διπλοτύπων.

1. Παράρτημα Α: Μοντέλα κινητικότητας και script αρχεία

A_1: Μοντέλα κινητικότητας

Στο Σχήμα Α.1 παρουσιάζονται οι βασικότεροι τύποι Μοντέλων Κινητικότητας και ειδικότερα του Μοντέλου που επιλέχθηκε να χρησιμοποιήσει ο προσομοιωτής κίνησης των κόμβων.



Σχήμα Α.1 Παρουσίαση Μοντέλων Κινητικότητας

A_2: Παρουσίαση μέρους του script αρχείου n100.tcl

H:\j_sim_diplom\jsim-1.3\n100.tcl - Notepad++

Αρχείο Επεξεργασία Αναζήτηση Επισκόπηση Μορφή Γλώσσα Ρυθμίσεις Μακροεντολή Εκτέλεση TextFX Πρόσθετα Παράθυρο

n100.tcl

```
1   puts mobtest
2   set minx 0.0
3   set maxx 100.0
4   set miny 0.0
5   set maxy 100.0
6   set maxz 0.0
7   set minz 0.0
8   set dx 1.0
9   set dy 1.0
10  set dz 0.0
11  set maxSpeed 2.0
12  cd [ exec mkdir mobtest]
13  for {set i 0} {$i<=99} {incr i} {
14  mkdir drcl.inet.Node node$i
15  cd node$i
16  set mob$i [mkdir drcl.inet.mac.MobilityModel mobility]
17  set convert$i [mkdir PositionReportConvert converter]
18  set plot [mkdir drcl.comp.tool.Plotter plotter]
19  set file [mkdir drcl.comp.io.FileComponent .file]
20  $file open "n100.plot";
21  connect -c $plot/.output@ -to $file/in@
22  cd ..
23  }
24
25  $mob0 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
26  $mob0 setPosition $maxSpeed 0 0 0
27  $mob1 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
28  $mob1 setPosition $maxSpeed 0 0 0
29  $mob2 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
30  $mob2 setPosition $maxSpeed 0 0 0
31  $mob3 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
32  $mob3 setPosition $maxSpeed 0 0 0
```

• • •

```
217 $mob96 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
218 $mob96 setPosition $maxSpeed 0 0 0
219 $mob97 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
220 $mob97 setPosition $maxSpeed 0 0 0
221 $mob98 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
222 $mob98 setPosition $maxSpeed 0 0 0
223 $mob99 setTopologyParameters $maxx $maxy $maxz $minx $miny $minz $dx $dy $dz
224 $mob99 setPosition $maxSpeed 0 0 0
225 $plot setXRange 0 $minx $maxx
226 $plot setYRange 0 $miny $maxy
```

```
227  connect $mob0/.query@ -and $convert0/query@
228  connect -c $convert0/out@ -to $plot/0@0
229  connect $mob1/.query@ -and $convert1/query@
230  connect -c $convert1/out@ -to $plot/1@0
231  connect $mob2/.query@ -and $convert2/query@
232  connect -c $convert2/out@ -to $plot/2@0
233  connect $mob3/.query@ -and $convert3/query@
234  connect -c $convert3/out@ -to $plot/3@0
235  connect $mob4/.query@ -and $convert4/query@
236  connect -c $convert4/out@ -to $plot/4@0
```

• • •

```
419  connect $mob96/.query@ -and $convert96/query@
420  connect -c $convert96/out@ -to $plot/96@0
421  connect $mob97/.query@ -and $convert97/query@
422  connect -c $convert97/out@ -to $plot/97@0
423  connect $mob98/.query@ -and $convert98/query@
424  connect -c $convert98/out@ -to $plot/98@0
425  connect $mob99/.query@ -and $convert99/query@
426  connect -c $convert99/out@ -to $plot/99@0
427  set sim [attach_simulator .]
428  run .
429  rt . stopAt 1000
```

Παρουσίαση μέρους του αρχείου *n100.txt*

C:\Documents and Settings\user\Τα έγγραφά μου\2009\Διπλωματική\stathis\n100.txt - Notepad++

Αρχείο Επεξεργασία Αναζήτηση Επισκόπηση Μορφή Γλώσσα Ρυθμίσεις Μακροεντολή Εκτέλεση TextFX Πρόσθετα Παράθυρο

n100.txt

```
1   ###   NEW PLOT--0--<No Title>--'- Time -'
2   #####  NEW SET--0--39--???
3   m--0--39--0.2975557890737861--9.766519948288604E-4--???
4   #####  NEW SET--0--38--???
5   m--0--38--1.137685190686975--0.358297666244716--???
6   #####  NEW SET--0--37--???
7   m--0--37--1.4743983169853416--1.0034844351867074--???
8   #####  NEW SET--0--36--???
9   m--0--36--0.5087524426528307--0.7644423208475746--???
10  #####  NEW SET--0--35--???
```

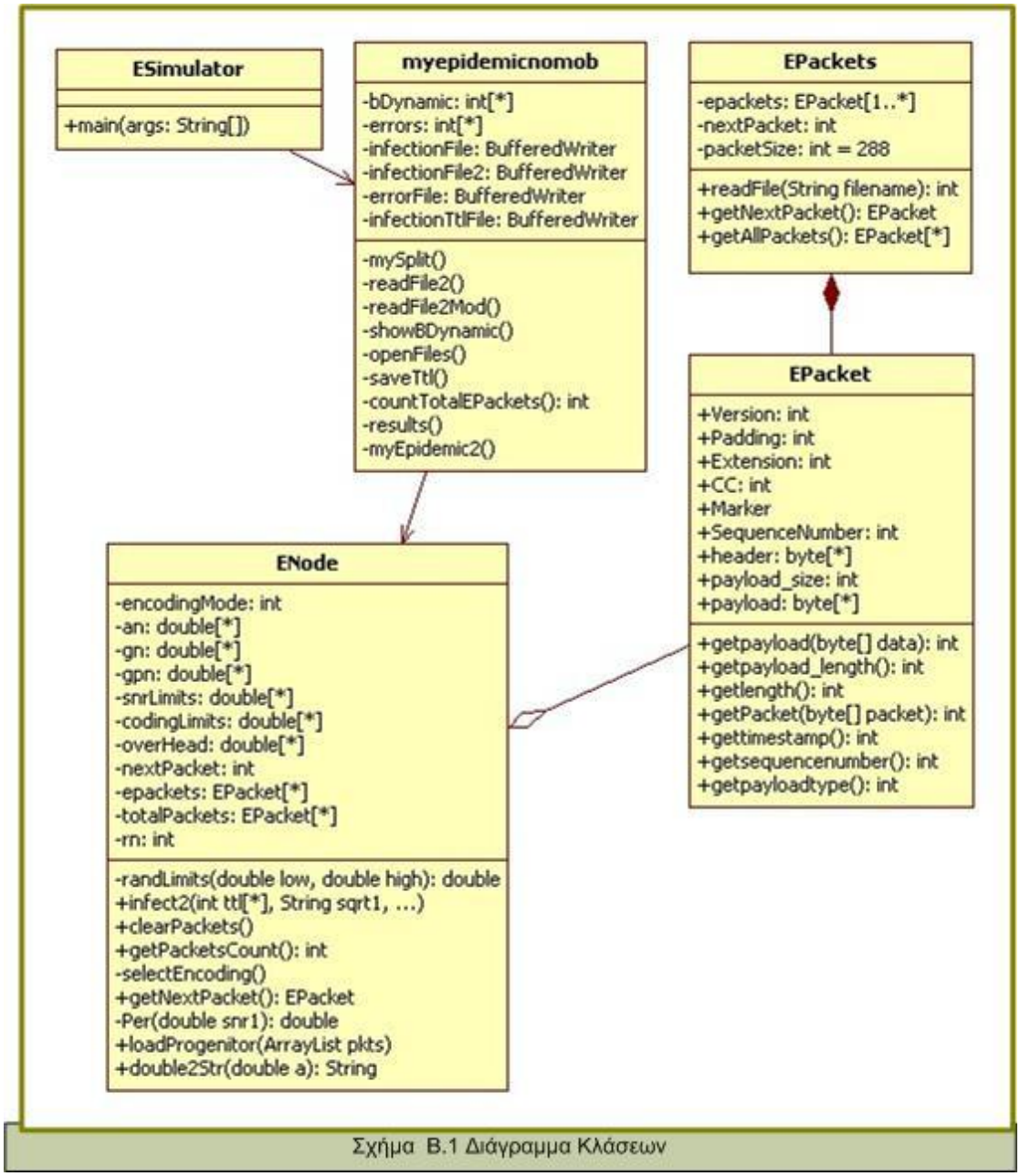
• • •

```
99927  p--0--45--32.25094025058311--81.94497514616516--???
99928  p--0--44--73.45684416582412--56.19645117845016--???
99929  p--0--43--72.06755063948187--24.04800788506901--???
99930  p--0--42--18.153914435412986--28.297579791887056--???
99931  p--0--41--84.26024845530847--40.903994607788746--???
99932  p--0--40--93.7039726825192--57.188576914602784--???
99933
```

Normal text file | 5133761 chars 5233693 bytes 99933 lines | Ln : 1 Col : 1 Sel : 0 (0 bytes) in 0 ranges

Παράρτημα Β: Απεικόνιση μέρους κώδικα του προγράμματος σε Java

Παρουσίαση του Διαγράμματος Κλάσεων



Σχήμα Β.1 Διάγραμμα Κλάσεων

ESimulator.java

ESimulator.java

```
1  package myepidemicnomob;
2  /**
3   *
4   * @author Admin
5   */
6  public class ESimulator {
7
8      public static void main(String[] args) {
9          System.out.println(args.length);
10         if (args.length < 8) {
11             System.err.println(" Usage:java myepidemicnomob.ESimulator:
12             #ofNodes startcount inputFilename b kin radius dyn_beta ");
13             return;
14         }
15         myepidemicnomob myepid = new myepidemicnomob(args);
16
17         return;
18     }
19
20 }
21
```

EPackets.java

EPackets.java

```
1  package myepidemicnomob;
2  import java.util.ArrayList;
3  import java.io.*;
4  import java.util.*;
5  /**
6   *
7   * @author Admin
8   */
9  public class EPackets {
10     private ArrayList epackets;
11     private int nextPacket;
12     private int pktSize = 288;   // Ουσιαστικά αποτελεί το μέγεθος του data payload
13     public EPackets() {
14         nextPacket = 0;
15     }
16     /**
17      * Διαβάζει το αρχείο που δίνεται ως παράμετρος ως τμήματα bytes
18      * και δημιουργεί τα πακέτα αυτού του μεγέθους
19      * @param filename Το αρχείο από όπου θα διαβαστούν τα πακέτα
20      */
21     public int readFile(String filename, int ttl2) {
22         int offset = 0;
23         int cread = 0;
24         int packetCounter = 0;
25         epackets = new ArrayList();
26         byte[] cbuf = new byte[pktSize];
27         try {
28             InputStream is = new FileInputStream(filename);
29             cread = is.read(cbuf, offset, pktSize);
30             String s = "";
31             EPacket ep;
```

```
32              while (cread > 0) {
33                  // Δημιουργεί το νέο πακέτο δίνοντας :
34                  // τύπο πακέτου (όλα 1)
35                  // αύξοντα αριθμό πακέτου
36                  // χρόνο (όλα 0)
37                  // δεδομένα
38                  // μέγεθος δεδομένων (όσα διαβάστηκαν - το πολύ pktSize)
39                  ep = new EPacket(1, packetCounter, 0, cbuf, cread, ttl2);
40                  epackets.add(ep);
41                  cread = is.read(cbuf, offset, pktSize);
42                  packetCounter++;
43                  //System.out.println("============"+ep.SequenceNumber);
44              }
45          } catch (IOException e) {
46              System.err.println("Exception caught: " + e.getMessage());
47          }
48          return packetCounter;
49      }
50      /**
51       * Επιστρέφει το επόμενο πακέτο που διαβάστηκε από το αρχείο
52       * Αν δεν υπάρχει άλλο πακέτο επιστρέφει null
53       * @return Ένα πακέτο
54       */
55      public EPacket getNextPacket() {
56          if (nextPacket < epackets.size()) {
57              System.out.println("---------------" + nextPacket + "----------");
58              return (EPacket) epackets.remove(nextPacket);
59          } else {
60              return null;
61          }
62      }
63      /*
64       * Επιστρέφει λίστα όλων των πακέτων. Χρησιμοποιείται μόνο για να φορτωθεί
65       * το ιστορικό του πρώτου κόμβου (υποτίθεται δηλαδή ότι από τον πρώτο κόμβο
66       * πέρασαν όλα τα πακέτα)
67       */
68      public ArrayList getAllPackets() {
69          return epackets;
70      }
71  }
72
```

EPacket.java

EPacket.java

```
1   package myepidemicnomob;
2   /**
3    * Κλάση που παριστάνει ένα τυπικό RTP πακέτο.
4    * Στην προσομοίωσή μας δεν χρησιμοποιούνται όλα τα πεδία, τα οποία
5    * διατηρήθηκαν στις δηλώσεις για χάρη της πληρότητας.
6    * @author Admin
7    */
8   public class EPacket {
9       //size of the RTP header:
10      static int HEADER_SIZE = 12;
11      //Fields that compose the RTP header
12      public int Version;
13      public int Padding;
14      public int Extension;
15      public int CC;
16      public int Marker;
17      public int PayloadType;
```

```
18      public int SequenceNumber;
19      public int TimeStamp;
20      public int Ssrc;
21      //Bitstream of the RTP header
22      public byte[] header;
23      //size of the RTP payload
24      public int payload_size;
25      //Bitstream of the RTP payload
26      public byte[] payload;
27      //Constructor of an RTPpacket object
28      //from header fields and payload bitstream
29
30      public EPacket(int PType,int Framenb,int Time,byte[] data,int data_length){
31          //fill by default header fields:
32          Version = 2;
33          Padding = 0;
34          Extension = 0;
35          CC = 0;
36          Marker = 0;
37          Ssrc = 0;
38          //fill changing header fields:
39          SequenceNumber = Framenb;
40          TimeStamp = Time;
41          PayloadType = PType;
42          //build the header bistream:
43          //--------------------------
44          header = new byte[HEADER_SIZE];
45
46          //.............
47          //TO COMPLETE
48          //.............
49          //fill the header array of byte with RTP header fields
50
```

```
51          //header[0] = ...
52          //fill the payload bitstream:
53          //--------------------------
54          payload_size = data_length;
55          payload = new byte[data_length];
56          //fill payload array of byte from data (given in parameter of
57          // the constructor)
58          // ! Do not forget to uncomment method printheader() below !
59          }
60      //Constructor of an RTPpacket object from the packet bistream
61      public EPacket(byte[] packet, int packet_size) {
62          //fill default fields:
63          Version = 2;
64          Padding = 0;
65          Extension = 0;
66          CC = 0;
67          Marker = 0;
68          Ssrc = 0;
69
70          //check if total packet size is lower than the header size
71          if (packet_size >= HEADER_SIZE) {
72              //get the header bitsream:
73              header = new byte[HEADER_SIZE];
74              for (int i = 0; i < HEADER_SIZE; i++) {
75                  header[i] = packet[i];
76              }
77
```

```
78              //get the payload bitstream:
79              payload_size = packet_size - HEADER_SIZE;
80              payload = new byte[payload_size];
81              for (int i = HEADER_SIZE; i < packet_size; i++) {
82                  payload[i - HEADER_SIZE] = packet[i];
83              }
84
85              //interpret the changing fields of the header:
86              PayloadType = header[1] & 127;
87              SequenceNumber = unsigned_int(header[3]) +256 * unsigned_int(header[2]);
88              TimeStamp = unsigned_int(header[7]) + 256 * unsigned_int(header[6]) +
89              65536 * unsigned_int(header[5]) + 16777216 * unsigned_int(header[4]);
90          }
91      }
92
93      //--------------------------
94      //getpayload: return the payload bistream of the RTPpacket and its size
95      //--------------------------
96      public int getpayload(byte[] data) {
97
98          for (int i = 0; i < payload_size; i++) {
99              data[i] = payload[i];
100         }
101
102         return (payload_size);
103     }
104
105     //--------------------------
106     //getpayload_length: return the length of the payload
107     //--------------------------
108     public int getpayload_length() {
109         return (payload_size);
110     }
```

```
111     //getlength: return the total length of the RTP packet
112     public int getlength() {
113         return (payload_size + HEADER_SIZE);
114     }
115     //getpacket: returns the packet bitstream and its length
116     public int getpacket(byte[] packet) {
117         //construct the packet = header + payload
118         for (int i = 0; i < HEADER_SIZE; i++) {
119             packet[i] = header[i];
120         }
121         for (int i = 0; i < payload_size; i++) {
122             packet[i + HEADER_SIZE] = payload[i];
123         }
124
125         //return total size of the packet
126         return (payload_size + HEADER_SIZE);
127     }
128     //gettimestamp
129     public int gettimestamp() {
130         return (TimeStamp);
131     }
132     //getsequencenumber
133     public int getsequencenumber() {
134         return (SequenceNumber);
135     }
136     //getpayloadtype
137     public int getpayloadtype() {
138         return (PayloadType);
139     }
```

```
140     //print headers without the SSRC
141         public void printheader() {
142         //TO DO: uncomment
143     /*
144         for (int i=0; i < (HEADER_SIZE-4); i++)
145         {
146         for (int j = 7; j>=0 ; j--)
147         if (((1<<j) & header[i] ) != 0)
148         System.out.print("1");
149         else
150         System.out.print("0");
151         System.out.print(" ");
152         }
153
154         System.out.println();
155     */
156     }
157     //return the unsigned value of 8-bit integer nb
158     static int unsigned_int(int nb) {
159         if (nb >= 0) {
160             return (nb);
161         } else {
162             return (288 + nb);
163         }
164     }
```

ENode.java

ENode.java

```
1   /*
2    * To change this template, choose Tools | Templates
3    * and open the template in the editor.
4    */
5   package myepidemicnomob;
6   import java.io.*;
7   import java.util.ArrayList;
8   import java.util.*;
9   /**
10   * Κλάση που παριστάνει ένα κόμβο.
11   * Περιέχει μεθόδους αποστολής πακέτων, υπολογισμού λαθών, κλπ.
12   * @author Admin
13   */
14  public class ENode {
15      private int encodingMode;
16      private double[] an = {274.7229, 90.2514, 67.6181, 50.1222, 53.3987, 35.3508};
17      private double[] gn = {7.9932, 3.4998, 1.6883, 0.6644, 0.3756, 0.0900};
18      private double[] gpn = {-1.5331, 1.0942, 3.9722, 7.7021, 10.2488, 15.9784};
19      private double rn;
20      private double[][] snrLimits = {{1, 3}, {3, 6}, {6, 8.5}, {10, 11.5}, {13, 15}, {18, 21}};
21      private double[][] codingLimits = {{0, 0.70, 0.30}, {0.06, 0, 80, 0.14},
22          {0.05, 0.88, 0.07}, {0.05, 0.92, 0.03},
23          {0.10, 0.88, 0.02}, {0.20, 0.80, 0}};
24      private double[][] overhead = {{1, 2}, {1, 2}, {3, 4}, {9, 16}, {3, 4}, {3, 4}};
25      private int nextPacket;
26      private ArrayList epackets;
27      private ArrayList totalPackets;
28      private ArrayList successTransferred;
```

```
29      public ENode() {
30          nextPacket = 0;
31          encodingMode = 3;
32          epackets = new ArrayList();
33          totalPackets = new ArrayList();
34          successTransferred = new ArrayList();
35      }
36      /**
37       * Υπολογίζει μια τυχαία τιμή ανάμεσα σε δύο όρια.
38       * @param low   Το κατώτερο όριο
39       * @param high  Το ανώτερο όριο
40       * @return  Η τυχαία τιμή
41       */
42      private double randLimits(double low, double high) {
43          return (low + Math.random() * (high - low));
44      }
45      /**
46       * Η συνάρτηση εκτελείται όταν ένας κόμβος στέλνει ένα πακέτο.
47       * Υπολογίζει την πιθανότητα λάθους και το overhead ανάλογα με την
48       * κωδικοποίηση και την πιθανότητα αλλαγής κωδικοποίησης.
49       * @param all   Ο πίνακας διπλότυπων (γενικά όλων των πακέτων των κόμβων)
50       * @param sqrt1 Η απόσταση μεταξύ των δύο κόμβων
51       * @param source Ο κόμβος που στέλνει το πακέτο
52       * @param target Ο κόμβος που λαμβάνει το πακέτο (ως αριθμός)
53       * @param outfile  Το αρχείο καταγραφής μεταφορών
54       * @param outfile2 Το αρχείο καταγραφής όλων των πακέτων (και διπλότυπων)
55       * @param errorfile Το αρχείο καταγραφής των εσφαλμένων πακέτων
56       * @param targetNode Ο κόμβος παραλήπτης
57       * @param b Ο πίνακας πιθανοτήτων b
58       * @param starB  Συμμετέχει στον υπολογισμό του δυναμικού  b
59       * @param tPackets  Σύνολο Πακέτων
60       * @param dyn_beta Καθορίζει αν το β θα είναι δυναμικό ή σταθερό
61       */
62      public void infect2(int all[][], String sqrt1, int source, int target,
63              BufferedWriter outfile, BufferedWriter outfile2,
64              BufferedWriter errorfile, ENode targetNode, double b[],
65              int errors[], double startB, int tPackets, int dyn_beta) {
66          double p1, p2;
67          int c1 = 0, c2 = 0;
68          try {
69              if (epackets.size() != 0) {
70                  Iterator iter = epackets.iterator();
71                  while (iter.hasNext()) {
72                      EPacket pkt = (EPacket) iter.next();
73                  if (pkt.ttl != 0) {
74                    // επιλέγω κωδικοποίηση μεταφοράς
75                    selectEncoding();
76                    rn = Math.random();
77                    double snr = randLimits(snrLimits[encodingMode][0], snrLimits[encodingMode][1]);
78                    double per = Per(snr);
79                    String perStr = Double.toString(per);
80                    String rnStr = Double.toString(rn);
81                    perStr = perStr.replace('.', ',');
82                    rnStr = rnStr.replace('.', ',');
83                    // αν υπάρχει σφάλμα μετάδοσης
84                    if (per > rn) {
85                        errorfile.write(source + ";" + target + ";" + perStr + ";" + rnStr
86                        + ";" + pkt.SequenceNumber + ";" + encodingMode);
87                        errorfile.newLine();
88                        errors[target]++;
```

```
89                  } else {
90                              // υπολογισμός επιβάρυνσης overhead
91                              double ov = (pkt.payload_size / overhead[encodingMode][0])
92                                      * (overhead[encodingMode][1] - overhead[encodingMode][0]);
93                              String ovStr = Double.toString(ov);
94                              ovStr = ovStr.replace('.', ',');
95
96                              if (!targetNode.totalPackets.contains(pkt)) {
97                                  targetNode.epackets.add(pkt);
98                                  targetNode.totalPackets.add(pkt);
99
100                                 // γράφω στο αρχείο το πακέτο που μεταφέρθηκε σωστά
101                                 outfile.write(source + ";" + target + ";" + perStr + ";" + rnStr
102                                         + ";" + pkt.SequenceNumber + ";" + encodingMode + ";"
103                                         + ovStr + ";" + pkt.ttl + ";" + sqrt1);
104                                 outfile.newLine();
105                                 //pkt.ttl--;
106                                 // καταγράφω ότι τα πακέτο μεταφέρθηκε σωστά
107                                 successTransferred.add(pkt);
108                             }
109                             c1 = 0;
110                             c2 = 0;
111                             all[target][pkt.SequenceNumber]++;
112
113                             for (int tk1 = 0; tk1 < tPackets; tk1++) {
114                                 c2 += all[target][tk1];
115                                 if (all[target][tk1] != 0) {
116                                     c1++;
117                                 }
118                             }
119                             //Υπολογισμος ποσοστών για συμμετοχή ως βαρύτητες
120                             p1 = 0.0;
121                             p2 = 0.0;
122                             if ((c2) > 0) {
123                                 p1 = (double) (c2 - c1) / (c2 + errors[target]);
124                                 p2 = (double) errors[target] / (c2 + errors[target]);
125                             }
126                         // Το 8 όρισμα καθορίζει αν το β είναι δυναμικό ή σταθερό
127                             if (dyn_beta==0) {
128
129                                     b[target] = startB - p1 * startB + p2 * startB;
130                               }
131                             else {
132                                     b[target] = startB ;
133                                 }
134 /* Εναλακτικοί τρόποι υπολογισμού δυναμικού β,με συντελεστή βαρύτητας για το β το ίδιο το β
135 b[target] = (startB- p1*startB + p2*startB)*(1-startB) + startB*startB;
136 b[target] = startB -0.05*ttl[target][pkt.SequenceNumber] + 0.1*errors[target];*/
137                             //  Για να μην ξεφύγουν οι τιμές
138                             if (b[target] > 0.90) {
139                                 b[target] = startB;
140                             } else {
141                                 if (b[target] < 0.10) {
142                                     b[target] = 0.15;
143                                 }
144                             }
145                             // γράφω στο αρχείο το πακέτο που μεταφέρθηκε σωστά
146                             outfile2.write(source + ";" + target + ";" + perStr + ";" + rnStr
147                                     + ";" + pkt.SequenceNumber + ";" + encodingMode + ";"
148                                     + ovStr+ ";" + pkt.ttl+ ";" + sqrt1);
149                             outfile2.newLine();
150                         }
151                     }
152                 }
153             }
154         } catch (Exception ex) {
155             ex.printStackTrace();
156         }
157     }
```

```
158     /**
159      * Μειώνει το ttl των πακέτων του κόμβου που μεταφέρθηκαν επιτυχώς κατά 1
160      */
161     public void updateNodeTtl() {
162         Iterator iter = epackets.iterator();
163         while (iter.hasNext()) {
164             EPacket pkt = (EPacket) iter.next();
165             if (successTransferred.contains(pkt)) {
166                 pkt.ttl--;
167             }
168         }
169     }
170     /**
171      * Αδειάζει τα πακέτα που μπορεί να υπάρχουν στην ουρά ενός κόμβου
172      */
173     public void clearPackets() {
174         epackets.removeAll(epackets);
175     }
176     /**
177      * Επιστρέφει τον αριθμό των πακέτων που υπάρχουν στην ουρά ενός κόμβου
178      * @return συνολικός αριθμός πακέτων ουράς αναμονής του κόμβου
179      */
180     public int getPacketsCount() {
181         return epackets.size();
182     }
183     /**
184      * Υπολογίζει το encodingMode σύμφωνα με τα όρια των κωδικοποιήσεων
185      */
186     private void selectEncoding() {
187         double r = Math.random();
188         if (r < codingLimits[encodingMode][0]) {
189             encodingMode--;
190         } else if (r > codingLimits[encodingMode][0] + codingLimits[encodingMode][1]) {
191             encodingMode++;
192         }
193     }
194      /**
195       * επιστρέφει το επόμενο πακέτο του κόμβου (αυτό που φεύγει παρακάτω)
196       */
197      public EPacket getNextPacket() {
198          if (nextPacket < epackets.size()) {
199              System.out.println("---------------" + nextPacket + "----------");
200              return (EPacket) epackets.remove(nextPacket);
201          } else {
202              return null;
203          }
204      }
205      /**
206       * Υπολογίζει την πιθανότητα Per.
207       * @param snr1
208       * @return
209       */
210      private double Per(double snr1) {
211          double per = 0;
212          if (snr1 >= gpn[encodingMode]) {
213              per = an[encodingMode] * Math.exp(-gn[encodingMode] * snr1);
214          } else if (snr1 > 0 && snr1 < gpn[encodingMode]) {
215              per = 1;
216          } else {
217              System.out.println("PER DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD");
218          }
219          return per;
220      }
```

```
221     /**
222      * Φορτώνει την ουρά ενός κόμβου με όλα τα πακέτα που δίνονται.
223      * @param pkts  Λίστα με τα αρχικά πακέτα του συστήματος
224      */
225     public void loadProgenitor(ArrayList pkts) {
226         epackets.addAll(pkts);
227         totalPackets.addAll(pkts);
228     }
229     /**
230      * Μετατρέπει έναν double σε string κατάλληλο για εγγραφή σε αρχείο excel
231      * @param a Ο αριθμός που θα μετατρέψει
232      * @return  Το string που μπορεί να γραφεί στο αρχείο
233      */
234     public static String double2Str(double a) {
235         String aStr = Double.toString(a);
236         aStr = aStr.replace('.', ',');
237         return aStr;
238     }
239 }
240
```

myepidemicnomob.java

myepidemicnomob.java

```
1  package myepidemicnomob;
2  import java.io.*;
3  import java.util.logging.Level;
4  import java.util.logging.Logger;
5  public class myepidemicnomob {
6      public final int SUSCEPTIBLE = 0;
7      public final int INFECTED1 = 1;
8      public final int INFECTED2 = 2;
9      public final String COMMA = ";";
10     public final String FILENAME0 = "out0.csv";
11     public double A[][];
12     public int S[];
13     public int M[];
14     public int N = 0;
15     public int N1 = 0;
16     public int N2 = 0;
17     public int startcount = 0;
18     public int nN;
19     public String filename = "";
20     public String line = "";
21     public double b = 0.0;
22     public double kin = 0.0; // χρησιμοποιείται ως παράμετρος κίνησης των κόμβων με τιμές 0 ή 1
23     public double r = 0.0;
24     private ENode enodes[];
25     private EPackets eps;
26     private int totalPackets;
27     private BufferedWriter infectionFile, infectionFile2, errorFile,
28             distancesFile, infectionAllFile;
29     private int progenitor, infCount, infAttempts, countTotalTransfers;
30     private boolean hasLoadPackets;
31     private int[][] infAll;
32     private double[] bDynamic;
33     private int[] errors;
34     private int ttl;
35     public int dyn_beta=0 ;
```

```
36      public myepidemicnomob(String[] args) {
37          System.out.println(args.length);
38          if (args.length < 8) {
39              System.err.println(" Usage: java myepidemicnomob.ESimulator:
40              #ofNodes startcount inputFilename b kin ttl radius dyn_beta  ");
41              return;
42          }
43          N = Integer.parseInt(args[0]);
44          startcount = Integer.parseInt(args[1]);
45          filename = args[2];
46          b = Double.parseDouble(args[3]);
47          kin = Double.parseDouble(args[4]);
48          ttl = Integer.parseInt(args[5]);
49          r = Double.parseDouble(args[6]);
50          dyn_beta = Integer.parseInt(args[7]);
51          A = new double[N][2];
52          M = new int[3];
53          progenitor = -1;        // δηλώνει ότι ακόμη κανείς δεν είναι "άρρωστος"
54          hasLoadPackets = false;
55          infCount = 0;
56          infAttempts = 0;
57          countTotalTransfers = 0;
58          enodes = new ENode[N];          // πίνακας με τις δομές των κόμβων
59          for (int i = 0; i < N; i++) {
60              enodes[i] = new ENode();
61          }
62          eps = new EPackets();
63          totalPackets = eps.readFile("readme.txt", ttl);
64          infAll = new int[N][totalPackets];
65          errors = new int[N];
66          bDynamic = new double[N];
67          for (int i = 0; i < N; i++) {
68              bDynamic[i] = b;
69          }
70          initS();
71          if (kin == 0) {
72              readFile2();
73          } else if (kin == 1) {
74              readFile2Mob();
75          }
76          // results();
```

```
77      }
78  // Αρχικοποίηση του πίνακα Καταστάσεων των Κόμβων
79      private void initS() {
80          int i;
81          // N1 = (int) (N / 5);
82          // N2 = (int) (N / 10);
83          S = new int[N];
84          /* for (i = 0; i < N1; i++) {
85          S[i] = INFECTED1;
86          }
87          for (i = N1; i < N1 + N2; i++) {
88          S[i] = INFECTED2;
89          }*/
90          for (i = 0; i < N; i++) {
91              S[i] = SUSCEPTIBLE;
92          }
93      }
94      /**
95       * Χωρίζει μια γραμμή από το αρχείο θέσεων των κόμβων για να πάρει τη
96       * θέση του κάθε κόμβου
97       */
```

```
98      private void mySplit() {
99          int node, stop, start = 0;
100         double x, y;
101         String s;
102         stop = line.indexOf("--", start);
103         s = line.substring(start, stop);
104         start = stop + 2;
105         stop = line.indexOf("--", start);
106         s = line.substring(start, stop);
107         start = stop + 2;
108         stop = line.indexOf("--", start);
109         s = line.substring(start, stop);
110         node = Integer.parseInt(s);
111         start = stop + 2;
112         stop = line.indexOf("--", start);
113         s = line.substring(start, stop);
114         x = Double.parseDouble(s);
115         start = stop + 2;
116         stop = line.indexOf("--", start);
117         s = line.substring(start, stop);
118         y = Double.parseDouble(s);
119         if (kin==0) {
120             if (A[node][0]==0) {
121                 A[node][0] = x;
122             }
123             if (A[node][1]==0) {
124                 A[node][1] = y;
125             }
126         }
127         else {
128             A[node][0] = x;
129             A[node][1] = y;
130         }
131
132     }
```

```
133     /**
134      * Διαβάζει τις γραμμές του αρχείου με τις θέσεις των κόμβων και καλεί τη
135      * myepidemic2 για να υπολογίσει τις μεταφορές πακέτων.
136      * Λειτουργει για σταθερές θέσεις κόμβων.
137      */
138     private void readFile2() {
139         //Διαβάζει το αρχείο συντεταγμένων για κάθε χρονική στιγμή-βήμα
140         try {
141             FileReader f = new FileReader(filename);
142             BufferedReader b = new BufferedReader(f);
143             BufferedWriter outfile[] = new BufferedWriter[1];
144             outfile[0] = new BufferedWriter(new FileWriter(FILENAME0));
145             //Δημιουργεί το αρχείο εξόδου 0
146             openFiles();
147             line = b.readLine();
148             int lineCounter = 0;
149             int x = 0;
150             String s = "";
151             while (line != null) {
152                 // οι γραμμές που αρχίζουν με # είναι σχόλια
153                 if (line.charAt(0) != '#') {
154                     if (lineCounter>startcount) {
155                         // χωρίζει τη γραμμή και παίρνει τα Node, x, y
156                         mySplit();
157                     }
158                     lineCounter++;
```

```
159                 }
160                 line = b.readLine();
161             }
162             myEpidemic2();
163           while (countTotalTransfers != 0) {
164                 myEpidemic2();
165                 System.out.println(countTotalTransfers);
166                 this.countTotalEPackets();
167             }
168             b.close();
169             outfile[0].flush();
170             outfile[0].close();
171         } catch (IOException e) {
172             System.err.println("Exception caught: " + e.getMessage());
173         }
174         saveAll();
175         showbDynamic();
176         for (int i = 0; i < N; i++) {
177             try {
178                 distancesFile.write(ENode.double2Str(A[i][0]) + ";"
179                         + ENode.double2Str(A[i][1]));
180                 distancesFile.newLine();
181             } catch (IOException ex) {
182                 Logger.getLogger(myepidemicnomob.class.getName()).log(Level.SEVERE, null, ex);
183             }
184         }
185         // κλείνω το αρχείο καταγραφής μολύνσεων
186         try {
187             infectionFile.close();
188             infectionFile2.close();
189             errorFile.close();
190             distancesFile.close();
191             infectionAllFile.close();
192         } catch (IOException e) {
193             System.err.println("Exception caught: " + e.getMessage());
194         }
195     }
```

```
196     /**
197      * Διαβάζει τις γραμμές του αρχείου με τις θέσεις των κόμβων και καλεί τη
198      * myepidemic2 για να υπολογίσει τις μεταφορές πακέτων.
199      * Λειτουργει για μεταβλητές θέσεις κόμβων.
200      */
201     private void readFile2Mob() {
202         //Διαβάζει το αρχείο συντεταγμένων για κάθε χρονική στιγμή-βήμα
203         try {
204             FileReader f = new FileReader(filename);
205             BufferedReader b = new BufferedReader(f);
206             BufferedWriter outfile[] = new BufferedWriter[1];
207             outfile[0] = new BufferedWriter(new FileWriter(FILENAME0));
208             openFiles();
209             line = b.readLine();
210             int lineCounter = 0;
211             int x = 0;
212             String s = "";
213             while (line != null) {
214                 // οι γραμμές που αρχίζουν με # είναι σχόλια
215                 if (line.charAt(0) != '#') {
216                     // χωρίζει τη γραμμή και παίρνει τα Node, x, y
217                     mySplit();
218                     lineCounter++;
219                     // όταν φτάσουμε σε γραμμή ίση (ή πολλαπλάσια) με το συνολικό αριθμό κόμβων
220                     if (lineCounter % N == 0) {
221                         if (lineCounter > startcount) {
222                             myEpidemic2();
```

```
                                System.out.println("TotalPackets: " + this.countTotalEPackets());
                                /* while (countTotalTransfers!=0) {
                                myEpidemic2();
                                System.out.println(countTotalTransfers);
                                }*/
                            }
                        }
                    }
                    line = b.readLine();
                }
                b.close();
                outfile[0].flush();
                outfile[0].close();
            } catch (IOException e) {
                System.err.println("Exception caught: " + e.getMessage());
            }
            saveAll();
            showbDynamic();
            for (int i = 0; i < N; i++) {
                try {
                    distancesFile.write(ENode.double2Str(A[i][0]) + ";"
                            + ENode.double2Str(A[i][1]));
                    distancesFile.newLine();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
            // κλείνω το αρχείο καταγραφής μολύνσεων
            try {
                infectionFile.close();
                infectionFile2.close();
                errorFile.close();
                distancesFile.close();
                infectionAllFile.close();
            } catch (IOException e) {
                System.err.println("Exception caught: " + e.getMessage());
            }
        }
```

```
        private void showbDynamic() {
            for (int i = 0; i < N; i++) {
                System.out.println(bDynamic[i] + "\t");
            }
        }
        private void openFiles() {
            try {
                // ανοίγω το αρχείο καταγραφής μολύνσεων
                infectionFile = new BufferedWriter(new FileWriter("infection.csv"));
                infectionFile.write("Source;Target;Per;Random;Packet;Enconding;Overhead;TTL;Distance");
                infectionFile.newLine();
                infectionFile2 = new BufferedWriter(new FileWriter("infectionF2.csv"));
                infectionFile2.write("Source;Target;Per;Random;Packet;Enconding;Overhead;TTL");
                infectionFile2.newLine();
                errorFile = new BufferedWriter(new FileWriter("errorfile.csv"));
                errorFile.write("Source;Target;Per;Random;Packet;Enconding");
                errorFile.newLine();
                distancesFile = new BufferedWriter(new FileWriter("distancesfile.csv"));
                distancesFile.write("Source;Target");
                distancesFile.newLine();
                infectionAllFile = new BufferedWriter(new FileWriter("infectionAll.csv"));
                for (int i = 0; i < totalPackets; i++) {
                    infectionAllFile.write("Packet" + i + ";");
                }
                infectionAllFile.write("b" + ";");
                infectionAllFile.newLine();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
```

```
291     /**
292      * Αποθηκεύει τον πίνακα με τον αριθμό πακέτων που λαμβάνει κάθε κόμβος
293      * σε αρχείο. Αποθηκεύει και το τελικό b του κάθε κόμβου
294      */
295     private void saveAll() {
296         try {
297             for (int i = 0; i < N; i++) {
298                 for (int j = 0; j < totalPackets; j++) {
299                     infectionAllFile.write(infAll[i][j] + ";");
300                 }
301                 infectionAllFile.write(ENode.double2Str(bDynamic[i]) + ";");
302                 infectionAllFile.newLine();
303             }
304         } catch (IOException ex) {
305             ex.printStackTrace();
306         }
307     }
308     /**
309      * Καταμετρά συνολικά πόσα πακέτα έχει ένας κόμβος
310      * @return
311      */
312     private int countTotalEPackets() {
313         int s = 0;
314         for (int i = 0; i < N; i++) {
315             s = s + enodes[i].getPacketsCount();
316         }
317         System.out.println(s);
318         return s;
319     }
320     private void myEpidemic2() {
321         int i, j;
322         countTotalTransfers = 0;
323 
324         double sqrt1 = 0;
325         System.out.println("EPIDEMIC RUNS");
326         for (j = 0; j < N; j++) {
327             for (i = 0; i < N; i++) {
328                 if (j == i) {
329                     continue;
330                 }
331                 sqrt1 = Math.sqrt(Math.pow(A[j][0] - A[i][0], 2) + Math.pow(A[j][1] - A[i][1], 2));
332                 String sqrt1Str = Double.toString(sqrt1);
333                 sqrt1Str = sqrt1Str.replace('.', ',');
334                 /*infectionFile2.write(sqrt1Str);
335                 infectionFile2.newLine();*/
336                 if (sqrt1 < r) {
337                     // αν είναι κοντά
338                     //                    if (S[i] == 1) { // κι αν είναι μολυσμένος
339                     if (Math.random() < bDynamic[i]) {
340                         // υπάρχει πιθανότητα να μας μολύνει
341                         S[j] = 1;
342                         if (progenitor == -1) {
343                             if (!hasLoadPackets) {
344                                 // αν αυτός είναι ο πρώτος που στέλνει
345                                 progenitor = j; // ο πρώτος κόμβος που στέλνει-μολύνει
346                                 enodes[j].loadProgenitor(eps.getAllPackets());
347                                 hasLoadPackets = true;
348                             }
349                         }
350                         //   System.out.print("Source: " + j + " target: " + i + " _  ");
351                         infAttempts++;
352                         // εδώ λειτουργεί ανάποδα: ενώ ο i μολύνει τον j,
353                         // τα πακέτα στέλνονται από τον j στον i
354                         if (enodes[j].getPacketsCount() > 0) {
355                             enodes[j].infect2(infAll, sqrt1Str, j, i,
356                                     infectionFile, infectionFile2,
```

```
357                                     errorFile, enodes[i], bDynamic, errors, b,
358                                     totalPackets,dyn_beta);
359                                 infCount++;
360                                 countTotalTransfers++;
361                             }
362                         }
363                     }
364                 }
365                 // εδώ τελειώνουν οι υποτιθέμενοι γείτονες του j, οπότε αδειάζω την ουρά πακέτων του
366                 enodes[j].updateNodeTtl();
367                 enodes[j].clearPackets();
368             }
369         }
370     }
371
```

Βιβλιογραφία και Πηγές

[1]: Q. Liu, S. Zhou *and* G. Giannakis, *"Cross-Layer Combining of Adaptive Modulation and Coding With Truncated ARQ Over Wireless Links"*, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 3, NO. 5, SEPTEMBER 2004

[2]: John G. Proakis, Masoud Salehi , «*Συστήματα Τηλεπικοινωνιών*», Έκδοση 2002 Εθνικού και Καποδιστριακό Πανεπιστήμιο Αθηνών, Μετάφραση Επιμέλεια Καρούμπαλος Κ., Ζέρβας Ε., Καραμπογιάς Σ. και Σαγκριώτης Ε.

[3]: William Stallings, « *Ασύρματες Επικοινωνίες και Δίκτυα* », Εκδόσεις Τζιόλα

[4]: William Stallings, « *Επικοινωνίες Υπολογιστών και Δεδομένων* », έκτη έκδοση Εκδόσεις Τζιόλα

[5]: http://www.medientheorien.hu-berlin.de/vlz/Zufall.pdf

[6]: Lin S. and Costello D. J. Jr., *"Error Control Coding: Fundamentals and Applications"*, Prentice – Hall.

[7]: Harry Briem, Stefan Gelbke , Σεμινάριο «*Συνελικτικοί κώδικες*» , Technische Universität Chemnitz
http://www.tu-chemnitz.de/informatik/ThIS/downloads/courses/ss03/kv/Faltungscodes.pdf

[8]: Γεώργιος Ευθύμογλου , «*Convolutional encoder / Viterbi decoder*», Σημειώσεις μαθήματος Ψηφιακές Επικοινωνίες του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.
http://dtps.unipi.gr/files/notes/2006-2007/eksamino_5/pshfiakes_epikoinwnies/mathhma_13o_2006.pdf

[9]: Claude Berrou, Alain Glavieux und Punya Thitimajshima: "*Near Shannon Limit error-correcting coding and decoding: Turbo-codes "*, Proceedings of IEEE International Communications Conference 1993
http://de.wikipedia.org/wiki/Turbo-Code

[10]: Walter Fischer,"*Digitale Fernseh- und Hörfunktechnik in Theorie und Praxis*", Neue Generation von DVB-Standards, Springer – Verlag Berlin Heidelberg 2010

[11]: http://el.wikipedia.org/wiki/WiMAX

[12]: http://wapedia.mobi/de/V.34

[13]: Γαβαλάς Δαμιανός, Διάλεξη #3# Μαθήματος «*Δίκτυα Υπολογιστών*», του Πανεπιστημίου Αιγαίου

[14]: Μετάδοση Δεδομένων « Κώδικες ανίχνευσης και διόρθωσης σφαλμάτων », http://www.e-yliko.gr/htmls/diktya/senario4/theory/files/DiktyMetds1Kef2.pdf

[15]: http://en.wikipedia.org/wiki/Diversity_scheme

[16]: http://en.wikipedia.org/wiki/Macrodiversity

[17]: Thomas Feuerstack / Thomas Gravel, *"Was ist das, Tcl/Tk?"*, ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/unplugged/upl016.pdf

[18]: Εργαστήριο Πολυμέσων, «*ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA*», Εθνικό Μετσόβειο Πολυτεχνείο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ.

[19]: http://java.sun.com/javase/downloads

[20]: http://netbeans.org/community/releases/68/

[21]: Christos Anagnostopoulos, «Epidemical Models», University of Athens, Pervasive Computing Research Group, epidemics.ppt

[22]: James F. Kurose , Keith W. Ross , «Δικτύωση Υπολογιστών Προσέγγιση από Πάνω προς τα Κάτω», 4η Έκδοση, Εκδόσεις : Μ. Γκιούρδας .

[23]: Christian Schindelhauer, Peter Mahlmann , «Algorithmen für Peer-to-Peer-Netzwerke», Έκδοση 4.4.1 / 2004 ,Institut für Informatik Universität Paderborn

[24]: C. Anagnostopoulos, S. Hadjiefthymiades, «A Biological Model for Collaborative Context Aware Systems», Transactions on Mobile Computing, TMC-0061-0207

[25]: «Μελέτη Επιδημικών Αλγορίθμων σε Περιβάλλοντα Διάχυτου Υπολογισμού» Δ.Ε. του ΕΑΠ του κ. Παπαδόπουλου Νίκου με επιβλέποντα τον κ.Χατζηευθυμιάδη Στάθη.

[26]: International Mobile Multimedia Communications Conference 2007 « An Epidemiological Model for Semantics Dissemination », Christos Anagnostopoulos, Evangelos Zervas, Stathes Hadjiefthymiades

[27]: C. Anagnostopoulos, S. Hadjiefthimiades, E. Zervas, Y. Ntarladimas, « 'Epidemics-based Collaborative Context Awareness» , Mobiquitous Dublin Ireland, 2008

[28]: «A Generalized Broadcasting Technique for Mobile Ad Hoc Networks» Institut fÄur Parallele und Verteilte Systeme (IPVS) der UniversitÄat Stuttgart 2007 Abdelmajid Khelil.

[29]: **http://el.wikipedia.org/wiki/Peer-to-peer**

[30]: http://netbeans.org/index_el.html

[31]: NetBeans Platform 6, Rich-Client Entwicklung mit Java, Galileo Computing

[32]: **http://sites.google.com/site/jsimofficial/start-with-j-sim**

[33]: Μιλτιάδης Κυριακάκου, « *Προσομοίωση Κινητικότητας και Πρόβλεψη Κίνησης σε Ασύρματα Δίκτυα Κινητών επικοινωνιών*», Διδακτορική Διατριβή του Εθνικού και Καποδιστριακό Πανεπιστημίου Αθηνών της Σχολή Θετικών Επιστημών, Τμήμα Πληροφορικής και Τηλεπικοινωνιών.

[34]: http://netbeans.org/community/releases/68/install_de.html