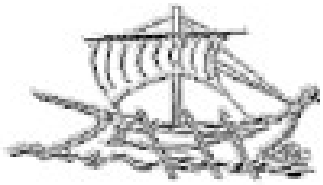


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

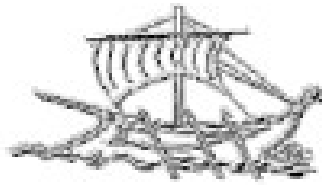
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υπολογιστικά συστήματα ανοχής σφαλμάτων

Καρφάκης Θ. Παναγιώτης
Αριθμός Μητρώου: 39576

Εισηγητής: Δρ. Έλληνας Ιωάννης

ΑΘΗΝΑ
ΦΕΒΡΟΥΑΡΙΟΣ 2018



PIRAEUS UNIVERSITY OF APPLIED SCIENCES

DEPARTMENT OF ELECTRONIC COMPUTER SYSTEMS ENGINEERING

THESIS

Fault Tolerant Computer Systems

Panagiotis Karfakis
Registry Number: 39576

Coordinator: Dr. Ellinas Ioannis

ATHENS
FEBRUARY 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υπολογιστικά Συστήματα Ανοχής Σφαλμάτων

Παναγιώτης Θ. Καρφάκης
Α.Μ. 39576

Εισηγητής:

Δρ. Έλληνας Ιωάννης, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης : .. /.. /

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος ΚΑΡΦΑΚΗΣ ΠΑΝΑΓΙΩΤΗΣ, του ΘΕΟΔΩΡΟΥ, με αριθμό μητρώου 39576 , φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό των αξιόπιστων υπολογιστικών συστημάτων και της ανοχής σφαλμάτων. Θα ήθελα να ευχαριστήσω την οικογένειά μου για την υπερπροσπάθεια που κατέβαλαν για να έχω την δυνατότητα να σπουδάσω σε αυτούς τους δύσκολους καιρούς. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, ο κ. Έλληνας Ιωάννης, τον οποίο και θα ήθελα να ευχαριστήσω για την συνεργασία μας.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με τεχνικές που χρησιμοποιούνται για την επίτευξη αξιόπιστων υπολογιστικών συστημάτων. Σε αυτό το πεδίο έχουν αναπτυχθεί από την αρχή της ιστορίας τους, πλήθος αξιόπιστων συστημάτων για κρίσιμες αποστολές σε τομείς όπως η αεροδιαστημική, η ιατρική, το εμπόριο και οι στρατιωτικές εφαρμογές. Τα συστήματα θα μελετηθούν από την σκοπιά της αξιοπιστίας τους και όχι τόσο από την πλευρά των επιδόσεων. Οι έννοιες της εξαρτησιμότητας, της διαθεσιμότητας, της συντηρησιμότητας και της ασφάλειας (Reliability Availability Maintainability Safety - RAMS) είναι μερικά χαρακτηριστικά που πρέπει να αποδοθούν σε ένα αξιόπιστο σύστημα. Κατά τον σχεδιασμό αξιόπιστων συστημάτων, η ανοχή σφαλμάτων αποτελεί την βασικότερη τεχνική αντιμετώπισης βλαβών και αποτυχιών. Οι πιθανότητες αποτυχίας ενός συστήματος μελετώνται και μοντελοποιούνται σύμφωνα με τις προδιαγραφές που μας δίνονται κάθε φορά. Γίνεται η περιγραφή των δυο διαθέσιμων τρόπων μοντελοποίησης αξιόπιστων υπολογιστικών συστημάτων: Reliability Block Diagrammes και αλυσίδες Markov. Για να αντιμετωπιστούν τα σφάλματα σε ένα σύστημα χρησιμοποιείται με μεγάλη επιτυχία η εφεδρεία ή αλλιώς πλεονασμός (Redundancy). Τα τέσσερα πεδία όπου εφαρμόζεται είναι ο πλεονασμός υλικού, λογισμικού, πληροφορίας και χρόνου. Αυτά εμπεριέχουν αξιόπιστες τρόπους για την πρόληψη πιθανών βλαβών και της αποτυχίας ενός συστήματος. Έτσι λοιπόν εντός της εργασίας αναφέρονται οι τεχνικές αυτές μαζί με παραδείγματα. Τέλος θα γίνει ο σχεδιασμός ενός υπολογιστικού συστήματος με την χρήση των τεχνικών αυτών για να εξασφαλίσουμε αξιόπιστους υπολογισμούς.

ΕΠΙΣΤΗΜΟΝΙΚΑ ΠΕΔΙΑ: Αξιοπιστία Υπολογιστικών Συστημάτων, Μοντελοποίηση Συστημάτων, Ανοχή σφαλμάτων, Υλικό υπολογιστών, Λογισμικό, Θεωρία Κωδίκων, Πιθανότητες, Στατιστική , Λογική.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: σφάλμα, λάθος, αποτυχία, αξιοπιστία, διαθεσιμότητα, κώδικες, πλεονασμός, ανοχή σφαλμάτων, επιδιόρθωση.

ABSTRACT

The present thesis concerns the study and development of reliable computer systems. From the origins of computer science there has been a continuous development of reliable computer systems for many mission critical appliances such as aerospace, medical, commercial and military. In this work, systems will be examined from the scope of reliability and not exclusively from the performance perspective. Attributes that accompany these systems are Reliability, Availability, Maintainability and Safety (RAMS), to mention a few. During the designation of such systems, *fault tolerance* is the only means of protection against faults and system failures. The possibilities of system failure are studied and modeled based on the initial specifications of a design. The models and techniques which are examined are Reliability Block Diagrammes and Markov chains. Furthermore, in order to make a system fault tolerant, redundancy has been successfully used as a countermeasure to faults. Redundancy varies and depends on the field it will be used. We will examine Information, Hardware, Software and Time Redundancy schemes and present some typical examples. In the end, we will develop a reliable computer system using these techniques to ensure reliable computations.

FIELDS: Systems Reliability, System Modelling, Fault Tolerance, Computer Hardware, Software, Coding Theory, Probabilities, Statistics, Logic.

KEYWORDS: Error, fault, failure, reliability, availability, redundancy, spares, fault tolerance, maintenance.

Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ESS Electonic Switching System
TMR Triple modular Redundancy
NMR N-Modular Redundancy
EMC Electromagnetic compatibility
ESD Electrostatic discharge
MTBF Mean Time Before Failure
MTTF Mean Time To Failure
MTTR Mean Time To Repair
EDC Error Detecting Code
ECC Error Correcting Code
MSB Most Significant Bit
LSB Less Significant Bit
CRC Circular Redundancy Check
RS Reed Solomon
LDPC Low Parity Parity Check
ARQ Automated Request Query
PNS Pair And Spare
NVP N-Version Programming
RBD Reliability Block Diagramme
COTS Custom Off The Shelf
ALU Arithmetic Logic Unit
RESO Recompute with Shifted Operands
RESWO Recompute with Swapped Operands
IoT Internet of Things
GF(q) Galois field of variable q

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας και γίνεται μια ιστορική αναδρομή γύρω από τις μεθόδους που έχουν παρουσιαστεί σε αυτήν την επιστημονική περιοχή.

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η ανοχή σφαλμάτων είναι η ικανότητα ενός συστήματος να συνεχίζει τις λειτουργίες του ακόμα και υπό την παρουσία σφαλμάτων. Γενικά, η ανοχή σφαλμάτων είναι συνδεμένη με την έννοια της αξιοπιστίας, δηλαδή την συνεχή αδιάλειπτη λειτουργία ενός συστήματος, χωρίς αυτό να αποτύγχει. Ένα σύστημα ανοχής σφαλμάτων θα πρέπει να είναι σε θέση να διαχειρίζεται τόσο τα σφάλματα υλικού όσο και αυτά του λογισμικού, τις διακοπές τροφοδοσίας ρεύματος, καθώς και άλλα είδη προβλημάτων, που δεν είναι προβλέψιμα, ώστε να συνεχίζει να πληρεί τις προδιαγραφές του. Τα σφάλματα μπορούν να κατηγοριοποιηθούν σε κατηγορίες, όπως: τα μεταβατικά, τα τυχαία ή μόνιμα, της σχεδίασης υλικού και λογισμικού, τα ανθρώπινα, καθώς και αυτά που οφείλονται σε εξωτερικές διαταραχές.

Η αναγκαιότητα της ανοχής σφαλμάτων σε οποιοδήποτε σύστημα γίνεται κατανοητή με το εξής οξύμωρο σχήμα, για να υπάρξει ιδανικό σύστημα θα πρέπει να μην υπάρχουν σφάλματα. Τα υπολογιστικά συστήματα, με την πάροδο του χρόνου, εξελίσσονται συνεχώς και γίνονται περίπλοκα ως προς τον σχεδιασμό αλλά και την αρχιτεκτονική τους. Δεν είναι ασυνήθιστο φαινόμενο πλέον να έχει κανείς έναν υπολογιστή αποτελούμενο από εκατοντάδες ως χιλιάδες εξαρτήματα τα οποία αλληλεπιδρούν με την βοήθεια λογισμικού για την υποστήριξη μιας συγκεκριμένης υπηρεσίας. Το πρόβλημα εμφανίζεται όμως καθώς η πολυπλοκότητα των συστημάτων αυξάνεται. Η αξιοπιστία είναι αντιστρόφως ανάλογη ως προς την πολυπλοκότητα τους, διότι αυξάνοντας τα χρησιμοποιούμενα εξαρτήματα αυξάνονται και οι πιθανότητες σφάλματος της κάθε μονάδας ξεχωριστά, συνεισφέροντας στο συνολικό κίνδυνο αποτυχίας του τελικού συστήματος.

Ακόμη ένα πρόβλημα εντοπίζεται στο γεγονός, ότι όσο και αν οι σχεδιαστές συστημάτων προσπαθούν να εξαλείψουν τις ανομοιομορφίες του υλικού και τα σφάλματα του λογισμικού, πάντα θα εμφανίζονται σφάλματα. Η ιστορία μας δείχνει ότι ο στόχος αυτός δεν είναι εφικτός. Είναι αναπόφευκτο είτε επειδή δεν λαμβάνονται υπόψην οι απρόσμενοι εξωτερικοί περιβαλλοντολογικοί παράγοντες, είτε δεν έχουν προβλεφθεί πιθανά λάθη του χειριστή του συστήματος, που είναι συνήθως άνθρωπος. Έστω η περίπτωση που σχεδιαστεί ένα σύστημα σύμφωνα με τις προδιαγραφές, σφάλματα θα προκληθούν από καταστάσεις εκτός του πεδίου έλεγχου των σχεδιαστών, για παράδειγμα κατά την διάρκεια του χρόνου ζωής του συστήματος.

Κατά τα πρώτα στάδια ανάπτυξης ενός υπολογιστικού συστήματος έως το τελικό προϊόν της διαδικασίας, υπάρχει η έννοια του ρίσκου αποτυχίας ή σφάλματος. Το πρώτο βήμα είναι ο ορισμός των προδιαγραφών του. Έπειτα προχωράμε στον σχεδιασμό και στην δημιουργία ενός πρωτοτύπου για δοκιμές. Έπειτα στη μαζική υλοποίηση του. Το τελευταίο βήμα αποτελεί η εγκατάσταση και η συντήρησή του. Όλα τα στάδια αυτά εμπεριέχουν αυτόν τον κίνδυνο και κρίνεται αναγκαία η επικύρωση της αξιοπιστίας σε κάθε στάδιο της διαδικασίας, ως μέτρο ελέγχου και πρόληψης.

Ως σφάλμα ορίζεται η φυσική ανομοιομορφία ή ελάττωμα που λαμβάνει χώρα σε κάποιο μέρος ενός συστήματος. Ένα σφάλμα σε πρώιμο στάδιο ανάπτυξης είναι δεδομένο ότι θα εμφανιστεί σε μετέπειτα στο στάδιο παραγωγής ή λειτουργίας. Αυτά τα ελαττώματα θα γίνουν γνωστά με την μορφή λαθών. Όταν ένα λάθος παρατηρηθεί κατά την λειτουργία του συστήματος, θα οδηγήσει σε αποτυχία συστήματος στην περίπτωση που δεν υπάρχει η απαραίτητη τεχνική επισκευής του. Ένα σύστημα χαρακτηρίζεται ως επιβλαβές όταν δεν είναι σε θέση να αποδώσει την αναμενόμενη λειτουργία του. Με τον όρο σύστημα θεωρούμε τις ανεξάρτητες μονάδες, οι οποίες αλληλεπιδρούν μεταξύ τους και σχηματίζουν μια ενιαία οντότητα, με σκοπό την παροχή μιας συγκεκριμένης υπηρεσίας. Ως αποτυχία συστήματος μπορεί ακόμη να θεωρηθεί και η παύση λειτουργίας. Η απόδοση ενός συστήματος μπορεί να μετρηθεί άλλοτε ποιοτικά και άλλοτε ποσοτικά, ανάλογα με την επιδείνωση ή αστάθεια της προσφερόμενης λειτουργίας.

Ο στόχος του σχεδιασμού για ανοχή σφαλμάτων είναι η ελάττωση της πιθανότητας αποτυχίας του συστήματος. Οι αποτυχίες συνήθως έχουν δυσάρεστα αποτελέσματα, μερικά ενδεικτικά αποτελούν οι απλές ενοχλήσεις των χρηστών, οι οικονομικές ζημίες για μια επιχείρηση, ο τραυματισμός και ο θάνατος ανθρώπων ακόμη και οι περιβαλλοντολογικές καταστροφές.

Τα συστήματα ανοχής σφαλμάτων και τα αξιόπιστα υπολογιστικά συστήματα περιλαμβάνουν ένα ευρύ φάσμα από εφαρμογές, όπως ενσωματωμένα συστήματα πραγματικού χρόνου, εμπορικά συστήματα ανταλλαγής πληροφοριών, μεταφορικά συστήματα, αεροδιαστημικά και στρατιωτικά συστήματα. Βασικοί τομείς έρευνας αποτελούν οι αρχιτεκτονικές των συστημάτων και οι τεχνικές σχεδίασης τους, οι θεωρίες κωδίκων, οι μέθοδοι ελέγχου και οι επικυρώσεις ορθότητας. Επιπλέον τομείς αποτελούν η μοντελοποίηση συστημάτων, η αξιοπιστία λογισμικού, τα λειτουργικά συστήματα καθώς και η παράλληλη και πραγματικού χρόνου επεξεργασία. Οι παραπάνω τομείς εμπλέκουν τεχνικό προσωπικό από διάφορα γνωστικά πεδία, όπως είναι τα Μαθηματικά και η Φυσική, η Θεωρία Ελέγχου, η Ιατρική την Μηχανική.

Τελειώνοντας, χωρίς τα συστήματα αυτά δεν θα μπορούσε να δομηθεί η κοινωνία μας στην μορφή που βρίσκεται σήμερα. Η συνεισφορά τους στην ανάπτυξη του ανθρώπινου πολιτισμού ήταν κρίσιμη όπως αναφέρεται στις επόμενες σελίδες. Ακολουθούν παραδείγματα τέτοιων συστημάτων και οι επιπτώσεις που μπορούν να προκληθούν σε περίπτωση σφάλματος, όπως έχουν καταγραφεί ιστορικά.

1.2 Εφαρμογές και Ιστορία

Αρχικά οι τεχνικές ανοχής σφαλμάτων χρησιμοποιήθηκαν ώστε να αντιμετωπιστούν φυσικές παραμορφώσεις σε εξαρτήματα υλικού. Οι πρώτοι σχεδιαστές υπολογιστικών συστημάτων χρησιμοποιούσαν τον τριπλασιασμό βασικών εξαρτημάτων ως μέθοδο εφεδρείας και ψηφοφορία πλειοψηφίας για την απόκρυψη σφαλμάτων, εντός του συστήματος. Πρωτοπόρος της ενσωμάτωσης πλεοναζόντων στοιχείων στα συστήματα ήταν ο John von Neumann τη δεκαετία του 1950 [1].

Μετά το πέρας του δευτέρου παγκοσμίου πόλεμου, ο τομέας της ανοχής σφαλμάτων αναπτύχτηκε σε μεγάλο βαθμό. Αυτό οφείλονταν στις συχνές βλάβες στον ηλεκτρονικό εξοπλισμό των στρατιωτικών εφαρμογων, καθιστώντας τον αναξιόπιστο για χρήση κατά την περίοδο του πόλεμου. Οπότε η έρευνα για αξιόπιστα

οπλικά συστήματα άρχισε να δέχεται και την ανάλογη προσοχή, χρηματοδοτώντας την ανοχή σφαλμάτων ως νέο πεδίο ανάπτυξης.

Ο πρώτος υπολογιστής που είχε την δυνατότητα να ανέχεται σφάλματα αναπτύχθηκε από ομάδα μηχανικών, υπό την εποπτεία του A. Svoboda το 1950-1954, στην Πράγα της Τσεχοσλοβακίας[2]. Ονομάστηκε SAPO και αποτελούνταν από 7,000 ρελέ και μαγνητική μνήμη τύπου drum των 1024 32-bit λέξεων. Ο επεξεργαστής του χρησιμοποιούσε τριπλασιασμό και ψηφοφορία πλειοψηφίας. Επιπλέον η ενσωματωμένη μνήμη αξιοποιούσε τεχνικές εντοπισμού σφαλμάτων με αυτοματοποιημένη επαναπροσπάθεια. Οι δυνατότητες αυτές απέδειξαν ότι η ανοχή σφαλμάτων κρίνεται απαραίτητη ιδιότητα σε μεγάλα συστήματα της εποχής εκείνης, που υπέφεραν συχνά από βλάβες και ανεπάρκεια χρηματοδότησης για έρευνα. Λειτούργησε από το 1957 ως το 1960 μέχρι που μια σπίθα από τα ρελέ πυροδότησε φωτιά, καταστρέφοντας το ολοσχερώς. Αργότερα, ένας δεύτερος υπολογιστής αναπτύχθηκε από την ίδια ομάδα με την ονομασία EPOS. Περιείχε ακόμη πιο ολοκληρωμένες δυνατότητες ανοχής σφαλμάτων. Οι δυνατότητες των μηχανών αυτών αναπτύχθηκαν εξαιτίας της έλλειψης αξιόπιστων εξαρτημάτων και την μεγάλη πιθανότητα αντιποίνων από τις κυβερνητικές αρχές, σε περίπτωση αποτυχίας.

Τα πιο ευρέως χρησιμοποιούμενα συστήματα υπολογιστών με ανοχή βλαβών αναπτύχθηκαν κατά τη διάρκεια της δεκαετίας του 1960[2]. Ήταν τα ηλεκτρονικά συστήματα μεταγωγής (ESS), που χρησιμοποιούνται σε γραφεία τηλεφωνικών κέντρων. Η πρώτη από αυτές ήταν της AT&T. Το No 1 ESS, όπως λέγονταν, είχε ως στόχο λιγότερο από δύο ώρες αποτυχία σε 40 χρόνια. Οι υπολογιστές από το σημείο αυτό και μετά διπλασιάζονται, επιπλέον σχεδιάζονται με ειδικό υλικό και εκτεταμένο λογισμικό, με δυνατότητες εντοπισμού σφαλμάτων και της άμεσης αντικατάστασης του προβληματικού εξαρτήματος. Τα μηχανήματα αυτά έχουν εξελιχθεί από τότε σε επόμενες γενιές στο No 5 ESS, το οποίο χρησιμοποιεί ένα κατανεμημένο σύστημα, που ελέγχεται από τον 3B20D υπολογιστή για την αντιμετώπιση βλαβών.

Η NASA (National Aeronautics and Space Administration) ήταν από τους πρώτους χορηγούς των FTCS (Fault Tolerant Computer Systems)[2].Ο πρώτος υπολογιστής ανοχής σφαλμάτων αναπτύχθηκε και εκτοξεύτηκε για το OAO (Orbiting Astronomical Observatory) και χρησιμοποιούσε επικαλυψη (fault masking) σε επίπεδο τρανζίστορ. Ο υπολογιστής του JPL (Jet Propulsion Laboratory), ο Self-Testing-And-Repairing ή STAR, ήταν ο επόμενος που κατασκευάστηκε για αυτήν, στα τέλη της δεκαετίας του 60, για μια δεκαετή διαστημική αποστολή προς τους εξωτερικούς πλανήτες της Γης. Αρχηγός της ομάδας σχεδιασμού ήταν ο A.Avizienis και ο STAR ηταν η πρώτη μηχανή που υλοποιούσε δυναμική επιδιόρθωση σε όλες τις πτυχές του συστήματος. Πολλές μονάδες του υπολογιστή είχαν την ικανότητα να εντοπίζουν εσωτερικά σφάλματα άλλα και να σημάνουν τα συμβάντα αυτά σε βοηθητικό επεξεργαστή, που όριζε την αναδιάταξη και επιδιόρθωση των προβληματικών μονάδων. Πιθανότατα η πιο πετυχημένη αποστολή στο διάστημα αποτελεί αυτή του JPL-Voyager του οποίου οι υπολογιστές λειτουργούν ήδη 38 χρόνια και συνεχίζουν να στέλνουν σήματα. Χρησιμοποιεί δυναμικό πλεονασμό αποτελούμενο από ζευγάρια εφεδρικών υπολογιστών, που ελέγχουν ο ένας τον άλλον με ανταλλαγή μηνυμάτων. Σε περίπτωση αποτυχίας ενός εκ των δύο, ο βοηθός του αναλαμβάνει τους υπολογισμούς. Το παραπάνω σχέδιο έχει αξιοποιηθεί σε αρκετά μετέπειτα διαστημόπλοια και αποστολές.

Ένα από τα πρώτα λειτουργικά μηχανήματα παρόμοιου τύπου ήταν ο υπολογιστής καθοδήγησης Saturn V, που αναπτύχθηκε στη δεκαετία του 1960[2]. Περιείχε

έναν TMR επεξεργαστή και διπλές μνήμες, που κάθε μια ενσωμάτωνε τεχνικές ανίχνευσης σφάλματος. Σφάλματα του επεξεργαστή καλύφθηκαν από την ψηφοφορία πλειοψηφίας και τα σφάλματα μνήμης παρακάμφθηκαν διαβάζοντας από εφεδρική. Η επόμενη μηχανή αυτού του τύπου ήταν ο υπολογιστής του Διαστημικού Λεωφορείου. Ήταν ένα ad-hoc σχέδιο που χρησιμοποίησε τέσσερις υπολογιστές που εκτελούσαν τα ίδια προγράμματα και διεξάγονταν ψηφοφορία για την επαλήθευση των αποτελεσμάτων. Ένας πέμπτος, μη-εφεδρικός υπολογιστής συμπεριλήφθηκε με διορθωτικά προγράμματα σε περίπτωση που ένα σφάλμα λογισμικού προέκυπτε και επηρέαζε την λειτουργία ορισμένων από αυτούς.

Κατά τη διάρκεια της δεκαετίας του 1970, δύο ακόμη μηχανήματα με ανοχή σφαλμάτων αναπτύχθηκαν από τη NASA για την αυτονομία καύσιμων των αεροσκαφών, καθώς απαιτούν συνεχή έλεγχο κατά την πτήση. Είχαν σχεδιαστεί για να ανταποκρίνονται στις πιο αυστηρές απαιτήσεις αξιοπιστίας, που υπολογιστής μπορούσε να προσφέρει ως εκείνη τη στιγμή. Και τα δύο μηχανήματα χρησιμοποιούσαν υβριδικές αρχιτεκτονικές εφεδρικών εξαρτημάτων.

Η πρώτη μηχανή, η Software Implemented Fault Tolerance (SIFT), αναπτύχθηκε από την SRI International[2,16]. Χρησιμοποιούσε off-the-shelf υπολογιστές, υλοποιούσε την ψηφοφορία και την αναδιαμόρφωση μέσω του λογισμικού. Μια εμπορική εταιρεία, η August Systems ήταν ένα υποπροϊόν του προγράμματος SIFT. Ανέπτυξε ένα σύστημα TMR που προορίζονταν για εφαρμογές ελέγχου βιομηχανικής διαδικασίας.

Η δεύτερη μηχανή, η Fault-Tolerant Multiprocessor (FTMP), που αναπτύχθηκε από το Εργαστήριο C.S.Draper, χρησιμοποιεί εξειδικευμένο υλικό για την πραγματοποίηση της επιδιόρθωσης σφαλμάτων[2,16]. Η FTMP έχει εξελιχθεί σε Fault-Tolerant επεξεργαστή (FTP), που χρησιμοποιείται από την Draper σε διάφορες εφαρμογές αλλά και σε Fault-Tolerant Παράλληλο επεξεργαστή (FTPP), που επιτρέπει διεργασίες να τρέχουν σε ένα μοναδικό μηχάνημα σε duplex, triplex ή τετραπλές ομάδες επεξεργαστών. Αυτό το ιδιαίτερα καινοτόμο σχέδιο είναι ανθεκτικό και επιτρέπει πολλαπλές ομάδες πλεοναζόντων επεξεργαστών να διασυνδεθούν για να σχηματίσουν κλιμακωτά συστήματα γνωστά ως cluster.

Καταλυτικό ρόλο στην ανάπτυξη των τεχνικών ανοχής σφαλμάτων έπαιξε και ο γνωστός αγώνας για την κατάκτηση του διαστήματος. Οι Ηνωμένες Πολιτείες της Αμερικής συναγωνιστήκαν την Σοβιετική Ένωση για την κατασκευή του πρώτου δορυφόρου, που τέθηκε σε τροχιά γύρω από την γη. Η εκτόξευση του Sputnik έγινε με επιτυχία το 1975. Η απότομη αυτή στροφή προς τις διαστημικές εφαρμογές οδήγησε στην δημιουργία ερευνητικών και στρατιωτικών προγραμμάτων, από τα υπουργεία αμύνης και διαστήματος αρκετών κρατών με στόχο την μελέτη και ανάπτυξη αξιόπιστων συστημάτων για κρίσιμες αποστολές, όπως είναι αυτή του διαστήματος.

Με την παράλληλη πρόοδο της τεχνολογίας των ημιαγωγών [2,12], τα ηλεκτρονικά εξαρτήματα έγιναν περισσότερο αξιόπιστα με αποτέλεσμα να εξαλειφτεί η ανάγκη για ανοχή σφαλμάτων όσον αφορά τις εφαρμογές γενικού σκοπού. Παρόλα αυτά, η ανοχή σε σφάλματα παρέμεινε σημαντική ιδιότητα για εμπορικά συστήματα, κρίσιμων απόστολων και ασφάλειας.

Σε εφαρμογές οπού απαιτείται κρίσιμη ασφάλεια, η ανοχή σφαλμάτων είναι μονόδρομος. Σημαντικές εφαρμογές ύψιστης ασφάλειας αποτελούν τα συστήματα υποστήριξης ζωής και περιβαλλοντολογικών σκοπών, ώστε να αποφευχθούν δυσάρεστες συνέπειες. Ειδικότερα, τα συστήματα πυρηνικών εργοστασίων, τα

μηχανήματα έκθεσης ακτινοβολιών, οι καρδιακοί και εγκεφαλικοί βηματοδότες αποτελούν παραδείγματα που χρίζουν αναγκαία την ενσωμάτωση μηχανισμών αντιμετώπισης σφαλμάτων.

Ένα δυσάρεστο παράδειγμα αποτέλεσε η μηχανή ακτινοβολιών Therac-25, στα μέσα της δεκαετίας του 1980 [3]. Η υψηλής ισχύος δέσμη ηλεκτρονίων χτύπαγε τους ασθενείς με περίπου 100 φορές την προβλεπόμενη δόση ακτινοβολίας, παρέχοντας μια δυνητικά θανατηφόρα δόση ακτινοβολίας βήτα. Δημιουργήθηκε από σφάλμα λογισμικού, που οδηγούσε μηχανικά την μονάδα φιλτραρίσματος. Η αίσθηση αυτή περιγράφεται από τον ασθενή Ray Cox ως "ένα έντονο ηλεκτρικό σοκ», αναγκάζοντας τον να κραυγάζει και να τρέχει έξω από το δωμάτιο θεραπείας. Αρκετές ημέρες αργότερα, τα εγκαύματα από ακτινοβολία εκδηλώθηκαν και οι ασθενείς σε τρεις περιπτώσεις εμφάνισαν τα συμπτώματα της δηλητηρίασης από ακτινοβολία. Δυστυχώς οι ασθενείς αργότερα έχασαν τη ζωή τους. Αργότερα αποδείχθηκε ότι τα λάθη δεν οφείλονταν μόνο σε ένα λόγο.

Ακόμη ένα τέτοιο γεγονός που έχει μείνει χαραγμένο στην ανθρώπινη ιστορία αποτελεί η έκρηξη του πυρηνικού αντιδραστήρα στο Chernobyl της τότε Σοβιετικής ένωσης, το 1986 [4]. Η δοκιμή σχεδιάστηκε για να αποδείξει ότι ο αντιδραστήρας θα τροφοδοτείται με επαρκή ποσότητα νερού ψύξεως, ακόμη και αν μια πλήρης απώλεια του ρεύματος συνέβαινε στο μεγάλο ηλεκτρικό συγκρότημα παραγωγής, ενώ το σύστημα ψύξης έκτακτης ανάγκης δεν ήταν διαθέσιμο για χρήση. Οι μηχανικοί που σχεδίασαν το τεστ ήταν ειδικοί σε ηλεκτρικές γεννήτριες, όχι σε πυρηνικούς αντιδραστήρες. Τα ιστορικά αρχεία δείχνουν ότι υπήρξε ελάχιστη διαβούλευση με ειδικούς του πυρηνικού αντιδραστήρα κατά τη διαδικασία προετοιμασίας. Έτσι λοιπόν είχαμε τεράστια περιβαλλοντολογική καταστροφή και θάνατο ανθρώπων από ραδιενέργεια, που συνεχίζεται ακόμα και σήμερα, γενιές αργότερα.

Όσον αφορά το εμπόριο, οι επιχειρήσεις πρέπει να λειτουργούν με μια συνεχεία και συνέπεια, ώστε να αποφεύγονται οικονομικές ζημίες άλλα και να γίνονται οι αγοραπωλησίες σε ταχυτάτους ρυθμούς. Τα σύγχρονα χρηματιστήρια υποστηρίζονται από τεράστια υπολογιστικά συστήματα πραγματικού χρόνου, που λειτουργούν με ταχυτάτους ρυθμούς. Η περίπτωση σφάλματος σε ένα τέτοιο σύστημα μπορεί να προκαλέσει οικονομική ζημιά τόσο στις επιχειρήσεις όσο και στο κράτος μέσα στο όποιο λειτουργεί, μεταβάλλοντας την ισοτιμία του νομίσματος και επηρεάζοντας αισθητά το εμπόριο. Χαρακτηριστικά παραμένουν τα γεγονότα της Μαύρης Δευτέρας το 1987[14]. Όπου ο δείκτης του Dow Jones έπεσε κατά 22.6%.

Η μεγαλύτερη εμπορική επιτυχία σε σφάλματα υπολογιστών έγινε στον τομέα της επεξεργασίας συναλλαγών σε τράπεζες και στις ηλεκτρονικές κρατήσεις εισιτηρίων, δωματίων και κάθε μορφής εμπορίου. Η Tandem Computers ήταν η πρώτη μεγάλη πάροχος και είναι ο σημερινός ηγέτης στην αγορά αυτή [16]. Η προσέγγιση που ακολουθήθηκε είναι ένα κατανεμημένο σύστημα που χρησιμοποιεί μια εξελιγμένη μορφή επικάλυψης σφαλμάτων. Για κάθε διεργασία που τρέχει, υπάρχει μια διεργασία που δημιουργεί αντίγραφα ασφαλείας που εκτελείται παράλληλα σε έναν άλλο υπολογιστή είτε τοπικά, είτε απομακρυσμένα. Η βασική διεργασία είναι υπεύθυνη για την δημιουργία αρχείου κατάστασης σε δίσκους. Αν αποτύχει, η δημιουργία αντιγράφων ασφαλείας σε κάποιο απρόσμενο σημείο της διαδικασίας, τότε μπορεί να γίνει επανεκκίνηση από το τελευταίο σημείο ελέγχου.

Η Stratus Computers έχει γίνει ένας άλλος σημαντικός κατασκευαστής μηχανών για εφαρμογές υψηλής διαθεσιμότητας [16]. Η προσέγγισή τους χρησιμοποιεί

υπολογιστές αμφίδρομου ελέγχου τοποθετημένα σε ζευγάρια, όπου κάθε υπολογιστής του ζεύγους διπλασιάζεται και συγκρίνεται για την παροχή υψηλής κάλυψης και σύγχρονης ανίχνευσης σφαλμάτων. Το ζεύγος αμφίδρομου ελέγχου στους υπολογιστές λειτουργεί συγχρονισμένα, έτσι ώστε αν κάποιος αποτύχει, ο άλλος να μπορεί να συνεχίσει τους υπολογισμούς χωρίς καθυστέρηση.

Άξιο αναφοράς αποτελεί η σειρά κεντρικών υπολογιστών της IBM, όπως ο S360. Τα συστήματα αυτά εξελίχθηκαν και χρησιμοποιούσαν εκτεταμένες τεχνικές ανοχής βλαβών, αυτόματη επαναπροσπάθεια εντολών και εναλλαγή των πλεοναζόντων μονάδων, για την παροχή υψηλού βαθμού διαθεσιμότητας της προσφερόμενης υπηρεσίας [16].

Η αγορά των server αποτέλεσε μια νέα και ταχέως αναπτυσσόμενη αγορά για τις μηχανές με ανοχή σφαλμάτων καθώς ήταν ιδανικοί για την ανάπτυξη του Διαδικτύου, των τοπικών δικτύων και την κάλυψη των ανάγκων για την συνεχή παροχή υπηρεσίας. Πολλοί κατασκευαστές διακομιστών προσφέρουν συστήματα που περιέχουν επιπλέον επεξεργαστές, δίσκους και τροφοδοτικά, που αυτόματα μεταπηδούν σε αντίγραφα σε περίπτωση προβλήματος. Παραδείγματα αποτελούν ο SUN ft-SPARC και ο HP / Stratus Continuum 400 [12]. Άλλες εταιρείες εργάζονται για ανοχή σφαλμάτων με τεχνολογία cluster, όπου άλλα μηχανήματα τοποθετημένα σε ένα δίκτυο μπορούν να αναλάβουν τα καθήκοντα ενός αποτυχημένου μηχανήματος. Ένα παράδειγμα είναι η MSCS τεχνολογία της Microsoft.

Για κρίσιμες αποστολές, η επιτυχής ολοκλήρωση τους καθορίζεται ως ο απόλυτος στόχος. Τέτοιες είναι οι αποστολές δορυφόρων, διαστημόπλοιων και οι στρατιωτικές. Οι στρατιωτικές εφαρμογές ανήκουν σε αυτή την κατηγορία επειδή η αποτελεσματικότητα και η ακρίβεια είναι αναγκαίες κατά την περίοδο ενός πολέμου. Ωστόσο, ένα παράδειγμα που έχει καταγραφεί συνέβη το 1983 και ήταν το άμεσο αποτέλεσμα ενός σφάλματος λογισμικού στο σύστημα έγκαιρης προειδοποίησης των Σοβιετικών [11]. Το αμυντικό σύστημα τους ενημέρωσε ότι οι ΗΠΑ είχαν ξεκινήσει επίθεση εναντίον τους με πέντε βαλλιστικούς πυραύλους. Η υποτιθέμενη εισερχόμενη καταστροφή οφείλονταν σε σφάλμα στο λογισμικό, που είχε ως ευθύνη να φιλτράρει τις ψεύτικες ανιχνεύσεις πυραύλων, που προκαλούνται από δορυφόρους με αντανακλάσεις ηλιακού φωτός στα σύννεφα.

Στα μέσα της δεκαετίας του 1990 [12] η επιστήμη της ανοχής σφαλμάτων συνέχισε την θεαματική της ανάπτυξη. Από τη μια μεριά είχαμε την ελάττωση του περιθωρίου θορύβου στους ημιαγωγούς σε κρίσιμα επίπεδα και από την άλλη την σμίκρυνση των παραγομένων στοιχείων, τη μείωση της τάσεως τροφοδοσίας και την αύξηση της επεξεργαστικής ισχύος. Το γεγονός αυτό έφερε την ακμή των ολοκληρωμένων κυκλωμάτων, ενισχύοντας την βιομηχανία. Η νεότερη έκδοση των CMOS-VLSI, η G4, χρησιμοποιεί κωδικοποίηση για τα μητρώα και επικάλυψη για την ανίχνευση σφαλμάτων. Ακόμη περιέχει επιπλέον επεξεργαστές, μνήμες, μονάδες εισόδου/εξόδου και τροφοδοτικά για να επανέρχεται σε κατάσταση ορθής λειτουργίας από ελαττώματα υλικού. Έτσι γίνεται εφικτή η παροχή υψηλών επίπεδων αξιοπιστίας σε προσιτές τιμές. Από την άλλη μεριά, τα ολοκληρωμένα είναι ευαίσθητα εξαρτήματα υλικού. Επηρεάζονται από περιβαλλοντολογικά φαινόμενα και παρεμβολές ακτινοβολιών, με συνέπεια να υποφέρουν από στιγμιαία σφάλματα. Έτσι ο σχεδιασμός ολοκληρωμένων με αντοχή σε αυτά οριοθέτησε την αξιοπιστία τους σε γενικά αποδεκτά επίπεδα.

Παράλληλα όμως με την ανάπτυξη των ολοκληρωμένων κυκλωμάτων, αναπτύχθηκαν εφαρμογές πραγματικού χρόνου. Τότε κυρίως υπήρχε η ζήτηση για

έξυπνα συστήματα λογισμικού, που βοήθησαν στην ερεύνα μεθόδων για την καταπολέμηση σφαλμάτων λογισμικού. Τα συστήματα λογισμικού προσφέρουν πρακτικό σχεδιασμό σε ανταγωνιστικό κόστος. Αντί να κατασκευαστεί ηλεκτρονικό κύκλωμα που υλοποιεί μια συγκεκριμένη λειτουργία, δημιουργείται κώδικας προγράμματος, που έπειτα τροφοδοτείται σε έναν επεξεργαστή για εκτέλεση. Με τον τρόπο αυτό αν αλλάξει η λειτουργικότητα του συστήματος στο μέλλον, με μερικές μόνο αλλαγές στο πρόγραμμα γίνεται προσαρμογή, που στην περίπτωση του υλικού θα χρειάζονταν ο επανασχεδιασμός νέου κυκλώματος.

Το λογισμικό λοιπόν έχει τη δυνατότητα να εξαλείφει τα μειονεκτήματα του υλικού, όπως είναι τα ελαττώματα της σχεδίασης σε φυσικό επίπεδο και ο πεπερασμένος χρόνο ζωής. Παρόλα αυτά, ένα αναπόφευκτο πρόβλημα σχετικά με αυτό είναι ότι ο σχεδιασμός συστημάτων λογισμικού γίνεται από άτομα που δεν έχουν εξειδίκευση στο σύστημα αυτό. Για παράδειγμα, ο σχεδιαστής συστημάτων φρενών για αυτοκίνητα αποφασίζει τον τρόπο ανάπτυξης των φρενών και έπειτα πληροφορεί τον μηχανικό λογισμικού, ο οποίος δημιουργεί πρόγραμμα με την ζητούμενη λειτουργικότητα. Αυτό το χάσμα πληροφορίας μεταξύ των δυο δημιουργεί προβλήματα στα έργα λογισμικού ακόμα και στις μέρες μας, γενιές αργότερα

Το 1996 δημιουργήθηκε το πρώτο μη επανδρωμένο δορυφορικό σύστημα, ήταν ο πύραυλος Ariane 5 [5,11], ο οποίος όμως “σκόπιμα” ανατινάχθηκε λίγα δευτερόλεπτα μετά την απογείωσή του στην παρθενική πτήση του, από το Κουρού της Γαλλικής Γουιάνας. Ο Ευρωπαϊκός Οργανισμός Διαστήματος εκτίμησε ότι η συνολική ανάπτυξη του Ariane 5 στοίχισε περισσότερο από 8 δις $. Εντός του Ariane 5 υπήρχαν τέσσερις επιστημονικούς δορυφόρους, συνολικού κόστους 500 εκ. $ έκαστος, που θα αξιοποιούνταν για να μελετήσουν με ποιό τρόπο το μαγνητικό πεδίο της Γης αλληλεπιδρά με τους ηλιακούς ανέμους. Αυτός ο απότομος τερματισμός της λειτουργίας του πυραύλου αποδείχθηκε αργότερα ότι οφείλονταν σε διαδοχική σειρά από σφάλματα. Η αυτοκαταστροφή πυροδοτήθηκε από τον υπολογιστή αυτόματης πλοήγησης κατά την εισαγωγή ενός αριθμού των 64 bit σε 16 bit καταχωρητή.

Δύο διαστημικά οχήματα, το Mars Climate Orbiter [6] και το Mars Polar Lander ήταν μέρος ενός διαστημικού προγράμματος το οποίο, το 1998, έπρεπε να μελετήσει το Αρειανό καιρό, κλίμα, την περιεκτικότητα σε νερό και το διοξείδιο του άνθρακα της ατμόσφαιρας. Αλλά το πρόβλημα εμφανίστηκε όταν ένα σφάλμα πλοήγησης προκάλεσε την άκατο να πετάξει πολύ χαμηλά στην ατμόσφαιρα ώσπου τελικά καταστράφηκε. Μια υπο-ανάδοχος στο πρόγραμμα της NASA, είχε χρησιμοποιήσει βρετανικές μονάδες, όπως χρησιμοποιούνται στις ΗΠΑ, παρά αυτές που προσδιορίζει η NASA, δηλαδή μονάδες του μετρικού συστήματος, όπως χρησιμοποιείται στην Ευρώπη.

Το έτος 2000, υπήρξε το πρόβλημα του λογισμικού που αφορά την πρακτική της αποθήκευσης ημερομηνιών του έτους σε διψήφια μορφή, έτσι ώστε το έτος 1997 να αποθηκεύεται ως 97. Στο τέλος του 20ου αιώνα[9,11], πολλές εφαρμογές λογισμικού σταμάτησαν να λειτουργούν ή δημιούργησαν εσφαλμένα αποτελέσματα κατά τη μετάβαση στο έτος 2000, τα μεσάνυχτα της 31ης Δεκεμβρίου 1999. Αυτό οφείλονταν στο γεγονός ότι πολλές εφαρμογές χρησιμοποιούν ημερομηνίες για τον υπολογισμό ευαίσθητων χρονικά λειτουργιών και η σειρά του 99 ακολουθούμενο από 00 μπορεί να προκαλέσει συγκρούσεις λογισμικού για πολλούς σημαντικούς υπολογισμούς, όπως τα επιτόκια και στεγαστικών δανείων πληρωμές. Ήταν ένα προγραμματιστικό λάθος. Η ζημιά ήταν τεράστια και όμως τα υπολογιστικά συστήματα αναφέρεται ότι θα συναντήσουν το ίδιο πρόβλημα το 2038.

Το θέμα της Airbus το 2006 επισήμανε τα προβλήματα πολλών εταιρειών με το λογισμικό[7,8,11]. Τι συμβαίνει όταν ένα πρόγραμμα δεν επικοινωνεί με το άλλο; Στην περίπτωση αυτή, το πρόβλημα προκλήθηκε από δύο κομμάτια του ίδιου προγράμματος, του λογισμικού CATIA, που χρησιμοποιείται για το σχεδιασμό και τη συναρμολόγηση ενός εκ των μεγαλύτερων αεροσκάφων του κόσμου, το A380 Airbus. Τμήματά του κατασκευάζονται σε δύο χώρες, την Γαλλία και την Γερμανία Με απλά λόγια, το γερμανικό σύστημα χρησιμοποίησε μια προηγούμενη έκδοση του CATIA και το γαλλικό σύστημα την τελευταία έκδοση. Έτσι, όταν η Airbus είχε φέρει μαζί τα δύο μισά του αεροσκάφους, το διαφορετικό λογισμικό είχε ως συνέπεια τη διαφορετική καλωδίωση του ενός από το άλλο. Τελικά το πρόβλημα αυτό λύθηκε, κοστίζοντας στην εταιρεία καθυστερήσεις παραπάνω του ενός έτους στην κατασκευή και σπατάλη τεράστιων οικονομικών πόρων για την επιδιόρθωση.

Περίπου 17.000 αεροπλάνα ήταν καθηλωμένα στο Διεθνές Αεροδρόμιο του Λος Άντζελες στις αρχές του έτους 2007, λόγω ενός προβλήματος του λογισμικού[11]. Η εν λόγω συσκευή ήταν μια κάρτα δικτύου, που αντί να τερματιστεί η λειτουργία του, επέμενε στην αποστολή λανθασμένων δεδομένων σε ολόκληρο το δίκτυο. Τα δεδομένα στη συνέχεια, αποστέλλονταν κλιμακωτά προς τα έξω μέχρι που χτύπησε το σύνολο του δικτύου στο USCBP (United States Customs and Border Protection agency) και το έφερε σε στασιμότητα. Κανείς δεν μπορούσε να εγκαταλείψει μήτε να εισέλθει στις ΗΠΑ μέσω του αεροδρομίου για οκτώ ώρες. Με αποτέλεσμα ο κόσμος να παραμείνει εγκλωβισμένος εντός του αεροδρομίου από τις αστυνομικές αρχές.

Η πυρηνική καταστροφή της Fukushima Daiichi [13] ήταν ένα ενεργειακό ατύχημα στο πυρηνικό εργοστάσιο της Fukushima που ξεκίνησε κυρίως από το τσουνάμι που ακολούθησε τον σεισμό Tohoku στις 11 Μαρτίου 2011. Αμέσως μετά τον σεισμό, οι ενεργείς αντιδραστήρες έκλεισαν αυτόματα τις συνεχείς αντιδράσεις σχάσης τους. Εντούτοις, το τσουνάμι απενεργοποίησε τις γεννήτριες έκτακτης ανάγκης που θα είχαν παράξει ενέργεια για τον έλεγχο και τη λειτουργία των αντλιών που είναι απαραίτητες για την ψύξη των αντιδραστήρων. Η ανεπαρκής ψύξη οδήγησε σε τρεις πυρηνικές τήξεις, σε χημικές εκρήξεις υδρογόνου-αέρα και στην απελευθέρωση ραδιενεργού υλικού στις Μονάδες 1, 2 και 3. Η απώλεια ψύξης προκάλεσε επίσης υπερθέρμανση στην πισίνα αποθήκευσης καυσίμων του αντιδραστήρα 4, λόγω της αποσύνθεσης των ράβδων καυσίμων. Τα αίτια ήταν από φυσικές καταστροφές αλλά ο ανθρώπινος παράγοντας έπαιξε τον ρόλο του στην επέκταση της μόλυνσης.

Μόλις πρόσφατα η Ινδία κατάφερε να στείλει τον πρώτο της δορυφόρο με μεγάλη επιτυχία στον πλανήτη Άρη[14]. Η επιτυχία οφείλεται στο γεγονός ότι ήταν η παρθενική της αποστολή δορυφόρου, κατασκευασμένου από μη αξιόπιστα εξαρτήματα. Κατασκευάστηκε στο ένα τέταρτο του κόστους σε σχέση με προηγούμενες ανάλογες αποστολές. Έτσι η Ινδία εισχώρησε στις ως τώρα δυνάμεις του διαστήματος μαζί με τις ΗΠΑ, Ρωσία, και την Ευρωπαϊκή Διαστημική Εταιρεία.

Η κατανεμημένη επεξεργασία πραγματικού χρόνου προσφέρει την παράλληλη εκτέλεση λειτουργιών και την προσομοίωση εικονικών μηχανών. Οι Cloud αρχιτεκτονικές θα φέρουν και νέες προκλήσεις για αντιμετώπιση. Τα αξιοπιστα συστήματα με βάση τον σκοπό που εξυπηρετούν χωρίζονται σε τρεις γενικές κατηγορίες: Μεγάλης διάρκειας, μη συντηρήσιμοι υπολογιστές, υπεραξιόπιστοι, υπολογιστές πραγματικού χρόνου και υψηλής διαθεσιμότητας υπολογιστές (Long-

life, unmaintainable computers, ultradependable, real-time computers, and high-availability computers).

1.3 Long-life, Unmaintainable computers

Εφαρμογές όπως οι αεροδιαστημικές προϋποθέτουν τους υπολογιστές να λειτουργούν για μεγάλες χρονικές περιόδους χωρίς εξωτερική επιδιόρθωση. Τυπικές προδιαγραφές είναι να έχουμε 95% πιθανότητα ορθής λειτουργιάς για μια διάρκεια 5-10 ετών. Μηχανές αυτού του τύπου πρέπει να αξιοποιούν το υλικό του συστήματος με έξυπνο τρόπο, επιπλέον να περιορίζονται σε χαμηλή κατανάλωση, βάρος και μέγεθος. Δεν είναι περίεργο που η NASA ήταν από τους πρώτους χορηγούς των Fault Tolerant Computer Systems για διαστημικές εφαρμογές μακράς διάρκειας.

1.4 Ultradependable, real-time computers

Πρόκειται για υπολογιστές για τους οποίους ένα λάθος ή μια καθυστέρηση μπορεί να αποδειχθεί καταστροφική. Έχουν σχεδιαστεί για εφαρμογές ελέγχου των αεροσκαφών, συστημάτων μαζικής μεταφοράς και ελέγχου πυρηνικών σταθμών. Οι εφαρμογές αυτές δικαιολογούν τις μαζικές επενδύσεις σε πλεονάζον υλικό, λογισμικό και δοκιμές. Η νέα γενιά fly-by-wire αεροσκαφών παρουσιάζει πολύ υψηλό βαθμό ανοχής σε βλάβες ελέγχου πτήσεως σε πραγματικό χρόνο. Για παράδειγμα, τα Airbus Airliners χρησιμοποιούν εφεδρικά κανάλια με διαφορετικούς επεξεργαστές και ποικίλο λογισμικό για προστασία από σφάλματα σχεδιασμού, καθώς και σφάλματα λογισμικού. Άλλοι τομείς περιλαμβάνουν τον έλεγχο των συστημάτων μαζικής μεταφοράς και τα κατανεμημένα συστήματα υπολογιστών, όπου σήμερα ενσωματώνονται και σε αυτοκίνητα.

1.5 High-Availability Computers

Πολλές εφαρμογές απαιτούν πολύ υψηλή διαθεσιμότητα, αλλά μπορούν να ανεχθούν ένα περιστασιακό σφάλμα ή πολύ μικρές καθυστερήσεις (της τάξης των λίγων δευτερολέπτων), ενώ παράλληλα εκτελείται η ανάκαμψη από τη βλάβη. Σχεδία υλικού για τα συστήματα αυτά είναι συχνά λιγότερο δαπανηρά από εκείνα που χρησιμοποιούνται στους υπεραξιόπιστους υπολογιστές πραγματικού χρόνου. Υπολογιστές αυτού του τύπου χρησιμοποιούν συχνά duplex αρχιτεκτονικές για την επικοινωνία τους. Παραδείγματα τέτοιων εφαρμογών είναι η τηλεφωνική μεταγωγή και επεξεργασία ηλεκτρονικών συναλλαγών και το χρηματιστήριο.

Στην εποχή που ζούμε κάθε χρήστης αλληλεπιδρά με μια συσκευή για να εκτελέσει μια συγκεκριμένη λειτουργία, είτε γίνεται χρήση μικρών και φτηνών επεξεργαστικών στοιχείων επικοινωνίας, τα όποια αλληλεπιδρούν σύγχρονα και κατανέμονται σε κάθε πτυχή της καθημερινότητας μας. Με ελάχιστη παρατήρηση γύρω μας καταλαβαίνουμε ότι τα έξυπνα κινητά και οι ταμπλέτες έχουν εισέλθει στις ζωές εκατομμυρίων ανά τον κόσμο προσφέροντας τις ευκολίες της τεχνολογίας στις άκρες των δαχτύλων τους. Όλες αυτές οι σχέσεις ανανέωσαν το ενδιαφέρον για ανοχή σφαλμάτων καθώς τα συστήματα έγιναν πιο πολύπλοκα και η διαθεσιμότητα τους αναγκαία.

Συνοψίζοντας, τα περασμένα εβδομήντα χρόνια πλήθος από μηχανές ανοχής σφαλμάτων υλοποιήθηκαν και εφαρμόστηκαν άλλοτε με επιτυχία και άλλες φορές με αποτυχία. Αξίζει όμως να σημειωθεί στο σημείο αυτό, ότι η προσφορά τους στον ανθρώπο είναι και θα παραμείνει αναγκαία. Το επόμενο ορόσημο των συστημάτων ανοχής σφαλμάτων αποτελεί η αποστολή επανδρωμένου πληρώματος στον Άρη ως το 2020 για την ανακάλυψη της βιωσιμότητας του πλανήτη, το Internet of Things (IoT) και το Industry 4.0.

ΚΕΦΑΛΑΙΟ 2 ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΑΞΙΟΠΙΣΤΙΑ

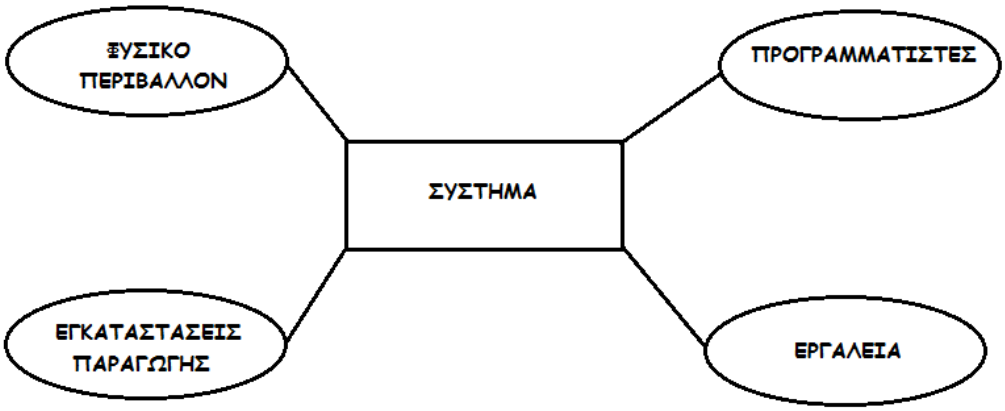
2.1 Κύκλος ζωης συστήματος

Ο κύκλος ζωής ενός συστήματος αποτελείται από την φάση της *ανάπτυξης* και τη *περίοδο λειτουργίας*. Η φάση της ανάπτυξης περιλαμβάνει όλες τις δραστηριότητες από την παρουσίαση της αρχικής ιδέας και τελειώνει με τις πετυχημένες δοκιμές επικύρωσης για το σύστημα, ώστε να είναι σε θέση για την παροχή πλήθους υπηρεσιών στο περιβάλλον δράσης του.

Ο κύκλος ζωής ενός συστήματος ξεκινά με τον ορισμό των προδιαγραφών, που περιέχει τα χαρακτηριστικά της αξιοπιστίας και της απόδοσης του. Προχωράμε στο σχεδιασμό, η οποία εμπεριέχει την ενσωμάτωση των μηχανισμών ανοχής σφαλμάτων και όταν το υπολογιστικό σύστημα περάσει όλες τις δοκιμές επικύρωσης με επιτυχία, είναι έτοιμο για χρήση. Το μοντέλο ανάπτυξης καταρράκτη αποτελεί τον γενικότερο τρόπο υλοποίησης ενός συστήματος. Βέβαια υπάρχουν και άλλες εξειδικευμένες, όπως το σπειροειδές μοντέλο, το μοντέλο V και το SDLC. Οι μεθοδολογίες αυτές χρησιμοποιούνται στην ανάπτυξη συστημάτων λογισμικού και υλικού, όμως εξαρτώνται από την εφαρμογή και τους διαχειριστές του έργου. Για την σχεδίαση συστημάτων με δυνατότητες ανοχής σφαλμάτων χρησιμοποιούνται τρείς βασικές μεθοδολογίες: τα RBD, οι αλυσίδες Markov και η συνδιαστική μοντελοποίηση όπως θα περιγραφεί σε επόμενη ενότητα.

Κατά τη διάρκεια της ανάπτυξης, το σύστημα αλληλεπιδρά με το περιβάλλον μέσα στο οποίο αναπτύσσεται και είναι πιθανό να εισαχθούν αναπτυξιακά σφάλματα σε αυτό. Το περιβάλλον αυτό αποτελείται από τα ακόλουθα τέσσερα στοιχεία :

1. **τον φυσικό κόσμο:** με τα φυσικά φαινόμενα του**.**
2. **προγραμματιστές και σχεδιαστές:** κάποιοι ενδεχομένως αναρμόδιοι ή με κακόβουλους στόχους.
3. **εργαλεία ανάπτυξης**: το λογισμικό και το υλικό που χρησιμοποιείται.
4. **εγκαταστάσεις παραγωγής και δοκιμής**: ο χώρος ανάπτυξης.

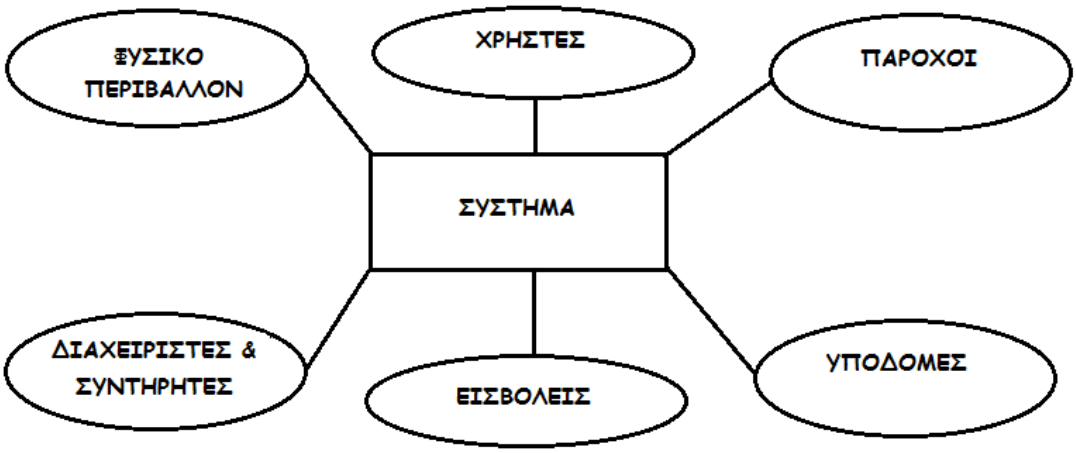


Εικόνα 2.1: Περιβάλλον ενός συστήματος κατά την φάση ανάπτυξης.

Η φάση της χρήσης ενός συστήματος αρχίζει όταν το σύστημα εγκρίνεται για αξιοποίηση και ξεκινά την παροχή των υπηρεσιών του προς τους χρήστες. Η λειτουργία του αποτελείται από εναλλασσόμενες περιόδους μεταξύ της *σωστής παροχής υπηρεσιών*, των *υπηρεσιών διακοπής* και *τερματισμού των υπηρεσιών*. Μια διακοπή λειτουργίας προκαλείται από μια αποτυχία των υπηρεσιών. Είναι η περίοδος που η λανθασμένη λειτουργία επηρεάζει την υπηρεσία στην διεπαφή της, δηλαδή γίνεται αντιληπτό από τους χρήστες του. Μια διακοπή της λειτουργίας έκτος από την επίδραση σφαλμάτων, μπορεί να οφείλεται από εξουσιοδοτημένο φορέα, κατά την εκτέλεση λειτουργιών συντήρησης, με σκοπό την προστασία του συστήματος. Από τα παραπάνω καταλαβαίνουμε ότι τα λάθη θα είναι αναπόφευκτο κομμάτι για ένα σύστημα.

Το περιβάλλον λειτουργίας αποτελείται από τα ακόλουθα στοιχεία:

1. **Ο φυσικός κόσμος:** με τα φαινόμενα του
2. **Οι διαχειριστές και συντηρητές:** οντότητες, άνθρωποι ή άλλα συστήματα που έχουν την εξουσία να διαχειρίζονται, να τροποποιούν, να επισκευάζουν και να το χρησιμοποιούν.
3. **Οι χρήστες:** οντότητες που λαμβάνουν την υπηρεσία από το σύστημα στις διεπαφές τους π.χ. τερματικά.
4. **Οι πάροχοι:** οντότητες που παρέχουν υπηρεσίες στο σύστημα ως διεπαφές χρήστη.
5. **Οι υποδομές:** οι φορείς που παρέχουν εξειδικευμένες υπηρεσίες στο σύστημα, όπως πηγές πληροφοριών (π.χ., χρόνος, GPS, κλπ), τις επικοινωνίες, πηγές ενέργειας, ροή αέρα ψύξεως, κλπ
6. **Οι εισβολείς:** κακόβουλες οντότητες που επιχειρούν να υπερβούν με κάθε τρόπο την ασφάλεια του, για να τροποποιήσουν την λειτουργικότητα ή την απόδοση του συστήματος. Ακόμη επιθυμούν να έχουν πρόσβαση σε εμπιστευτικές πληροφορίες. Παραδείγματα αποτελούν οι χάκερ και το διεφθαρμένο προσωπικό, που έχουν δικαιωματικά την ευθύνη εμπιστευτικών πληροφοριών και μπορεί να είναι από εχθρικές κυβερνήσεις, οργανισμούς ή ανταγωνίστριες εταιρείες και κάνουν χρήση κακόβουλου λογισμικού για να επιτύχουν τους σκοπούς τους.



Εικόνα 2.2 Περιβάλλον ενός συστήματος κατά την φάση χρήσης.

Έτσι λοιπόν οι αλληλεπιδράσεις με ένα σύστημα από τις αρχές του μέχρι το τελικό στάδιο της χρήσης του είναι πάρα πολλές και ορισμένες από αυτές απρόβλεπτες. Οι οντότητες που υπάρχουν στο περιβάλλον του εισάγουν αναπτυξιακά σφάλματα, τα οποία εκδηλώνονται στο αμέσως επόμενο στάδιο της παραγωγής ή σε μετέπειτα στάδιο κατά το χρόνο ζωής του συστήματος. Στην περίοδο λειτουργίας του, η συντήρηση είναι η τεχνική που χρησιμοποιείται για την επέκταση της ζωής του και αποτελεί τον μοναδικό μηχανισμό ανοχής σφαλμάτων μέχρι και το τέλος της ζωής του.

2.2 Βασικές Αρχές Αξιοπιστίας

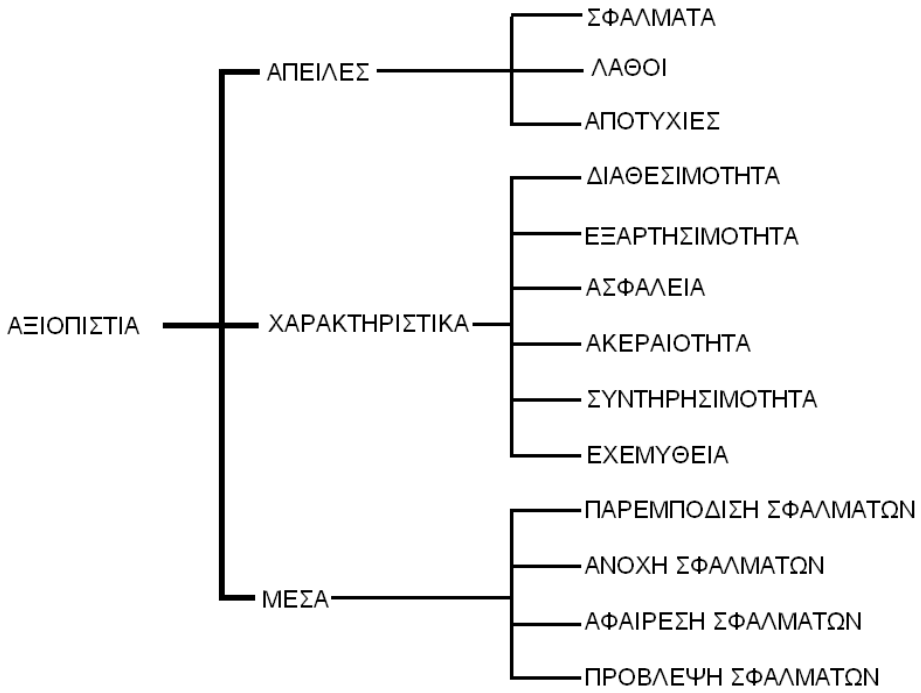
Τα υπολογιστικά συστήματα χαρακτηρίζονται από πέντε στοιχειώδη χαρακτηριστικά[1]: την *λειτουργικότητα*, την *χρησιμότητα*, την *απόδοση*, το *κόστος* και την *αξιοπιστία*. Η αξιοπιστία ενός υπολογιστικού συστήματος είναι η ικανότητα προσφοράς μιας συγκεκριμένης υπηρεσίας, την οποία μπορούν να εμπιστευτούν χωρίς αμφισβήτηση οι χρήστες του. Ειδικότερα, *Αξιοπιστία είναι η ιδιότητα ενός συστήματος να ενσωματώνει χαρακτηριστικά όπως εξαρτησιμότητα, διαθεσιμότητα, ασφάλεια, επιβιωσιμότητα και συντηρησιμότητα.*

Με τον όρο *υπηρεσία* περιγράφουμε τη λειτουργία του συστήματος όπως το αντιλαμβάνονται οι χρήστες του. Από την άλλη πλευρά ο *χρήστης* αποτελεί ένα ακόμη σύστημα, φυσικό ον, το οποίο αλληλεπιδρά με στη *διεπαφή της υπηρεσίας* που προσφέρει το υπολογιστικό σύστημα.

Η λειτουργία ενός συστήματος εκφράζεται με τον σκοπό που υπηρετεί και περιγράφεται από τις *λειτουργικές προδιαγραφές*. Η *ορθή λειτουργία* παρέχεται όταν πραγματοποιεί την αναμενόμενη υπηρεσία του. Η *αποτυχία του συστήματος* είναι ένα γεγονός που συμβαίνει όταν η παρεχόμενη υπηρεσία έχει αποκλίνει από την ορθή του λειτουργία. Με την σειρά της, η προσφορά εσφαλμένης υπηρεσίας αναφέρεται ως *διακοπή λειτουργίας*. Με βάση τα παραπάνω μπορούμε να καταλήξουμε σε έναν πιο συγκεκριμένο ορισμό της *αξιοπιστίας,* που συμπληρώνει τον αρχικό ως κριτήριο απόδοσης ενός βαθμού εμπιστοσύνης σε ένα σύστημα.

Έτσι λοιπόν, *Αξιοπιστία είναι η ικανότητα ενός συστήματος να αποφεύγει τις αποτυχίες, που συμβαίνουν πιο συχνά ή είναι πιο σοβαρές καθώς και τις διακοπές λειτουργίας, με μεγαλύτερη διάρκεια, οι οποίες δεν είναι αποδεκτές από τους χρήστες του.* Στην αντίθετη περίπτωση το σύστημα θεωρείται ως *αναξιόπιστο*, δηλαδή δεν πείθει τους χρήστες του ώστε να το αξιοποιήσουν.

Η δομημένη ανάλυση της συνοψίζεται σε τρεις συντελεστές, τις *απειλές* , τα *χαρακτηριστικά* και τα *μέσα* με τα οποία την επιτυγχάνουμε. Οι απειλές είναι οι αρνητικοί παράγοντες που βλάπτουν την αξιοπιστία ενός υπολογιστικού συστήματος. Τα χαρακτηριστικά αποτελούν τις ιδιότητες που επιθυμούμε να αποδώσουμε σε αυτό και τα μέσα τις τεχνικές με τις οποίες την διασφαλίζουμε.



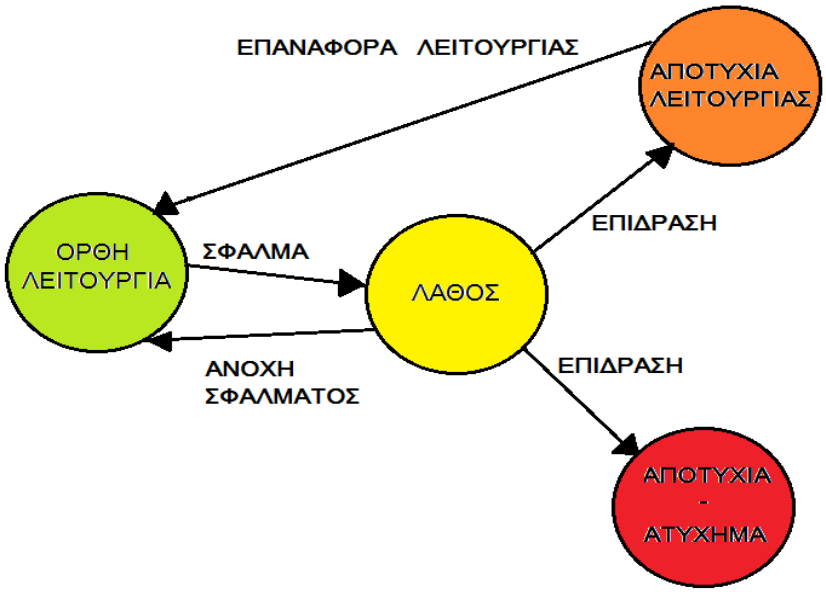
Εικόνα 2.3 Βασική ανάλυση της αξιοπιστίας συστημάτων.

2.3 Οι Απειλές της Αξιοπιστίας

Ένα σύστημα μπορεί να αποτύχει είτε επειδή δεν συμβαδίζει με τις προδιαγραφές για το οποίο αρχικά είχε σχεδιαστεί, είτε οι προδιαγραφές ήταν ανεπαρκείς για να περιγράψουν την λειτουργία του. Ένα *λάθος* είναι ένα γεγονός το οποίο επηρεάζει την κατάσταση του συστήματος, με επερχόμενη συνέπεια την διακοπή της λειτουργίας του. Ειδικότερα η διακοπή αυτή θα συμβεί όταν ένα η περισσότερα λάθη φτάσουν στην *διεπαφή της υπηρεσίας* και μεταβάλουν την ίδια την υπηρεσία. Είναι πλέον εύκολο να διατυπωθεί ότι ένα *σφάλμα είναι το κατακριτέο ή υποθετικό αίτιο ενός λάθους*. Έτσι λοιπόν μπορούμε να χαρακτηρίσουμε ένα σφάλμα *ενεργό* όταν παράγει ένα λάθος, στην αντίθετη περίπτωση αναφέρεται ως *αδρανειακό* ή *ανενεργό*.

Ένα σύστημα αποτελείται από ένα ή περισσότερα εξαρτήματα τα οποία αλληλεπιδρούν ως μία ενιαία οντότητα. Άρα οι καταστάσεις ενός συστήματος καθορίζονται από τις επιμέρους καταστάσεις του κάθε εξαρτήματος. Ένα σφάλμα προκαλεί ένα λάθος σε ένα ή περισσότερα εξαρτήματα, όμως η αποτυχία ενός συστήματος δεν επέρχεται μόνο από ένα σφάλμα αλλά από συνδυασμό τους.

Τα σφάλματα κατηγοριοποιούνται με τη σειρά τους με βάσει τα χαρακτηριστικά της αποτυχίας που προκαλούν[1,3], τα οποία μπορεί να είναι λάθη χρονισμού ή/και υπολογισμού τιμών. Ακόμη μπορεί να είναι *ελέγξιμα ή ανεξέλεγκτα*, *ομοιόμορφα ή ανομοιόμορφα*, αν δηλαδή το λάθος επηρεάζει δύο εξαρτήματα ή χρήστες με τον ίδιο τρόπο και με την ίδια κρισιμότητα. Ένα σφάλμα είναι εντοπίσημο με την μορφή λάθους αν η παρουσία του υποδεικνύεται με ένα ανάλογο *μήνυμα* ή *σήμα σφάλματος,* από κάποιο κύκλωμα ελέγχου. Ένα λάθος το οποίο υπάρχει αλλά δεν έχει εντοπιστεί ονομάζεται *λανθάνων.*



Εικόνα 2.4: Καταστάσεις λειτουργίας ενός συστήματος

Οι πηγές των σφαλμάτων είναι ποικιλόμορφες. Τα αίτια λαθών μπορούν να προκληθούν από φυσικά και ανθρώπινα αίτια. Ορισμένα από αυτά δεν έχουν εξηγήσεις. Ερωτήματα της μορφής 'γιατί οι προγραμματιστές κάνουν λάθη;' και γιατί τα ολοκληρωμένα κυκλώματα αποτυγχάνουν; δημιουργούν με την σειρά τους μια αναδρομική ροή ερωτημάτων τα οποία καταλήγουν στην αρχή του ζητήματος, γιατί να γίνονται λάθη;

Το σφάλμα ως ιδέα θα πρέπει να θέτει τέλος στην αναδρομή. Αυτός είναι και ο λόγος που περιγράφεται ως υποθετική ή/και δικαιολογημένη αιτία λάθους. Όμως η αιτία ενός συγκεκριμένου λάθους μπορεί να διαφέρει από την σκοπιά του μηχανικού συντήρησης, του προγραμματιστή, του σχεδιαστή ολοκληρωμένων, του project manager.

Στο διάγραμμα (Α.2.1) εμφανίζονται αναλυτικά οι τρόποι με τους οποίους μπορεί να προκύψει ένα λάθος σε ποιο στάδιο συνέβη και από ποιόν γίνεται. Οπότε είναι φανερό ότι οι υποκατηγορίες που δημιουργούνται είναι: *οι ατέλειες λογισμικού, η επιβλαβής λογική, τα λάθη υλικού, οι ατέλειες παραγωγής, η φυσική παρακμή, οι φυσικές παρεμβολές, οι επιθέσεις και τα λάθη εισόδου*. Από τις υποκατηγορίες αυτές προκύπτουν τρεις μεγάλες κατηγορίες σφαλμάτων, τα *σφάλματα σχεδιασμού και ανάπτυξης*, τα *φυσικά σφάλματα* και τα *σφάλματα αλληλεπιδράσεων.* Με το διάγραμμα αυτό (Α.2.1), γίνεται ευκολότερη η ανάλυση των σφαλμάτων σε προτάσεις. Έστω για παράδειγμα έχει προκύψει ένα σφάλμα ανάπτυξης που οφείλεται στο λογισμικό. Το αίτιο είναι ανθρώπινο και η πρόθεση του μπορεί να ήταν σκόπιμη ή τυχαία. Στην περίπτωση που ήταν σκόπιμο θεωρείται σφάλμα επιβλαβής λογικής με μόνιμες συνέπειες στο σύστημα.

Τα μη επιβλαβή σκόπιμα σφάλματα προκύπτουν είτε στη φάση της ανάπτυξης είτε κατά την διάρκεια λειτουργίας. Κατά την ανάπτυξη, προκύπτουν γενικά από συμβιβασμούς που εξυπηρετούν την διατήρηση ικανοποιητικής απόδοσης και διευκολύνουν την αξιοποίηση του συστήματος. Ακόμη υπάρχει περίπτωση να οφείλονται σε οικονομικούς λόγους. Τέτοια σφάλματα συνήθως είναι οι παραβιάσεις ασφαλείας με την μορφή συγκαλυμμένων καναλιών (covert channels) από διεφθαρμένο προσωπικό από ανταγωνιστές, είτε από την αμέλεια των διευθυντών

ενός project για να προλάβουν τις ημερομηνίες παράδοσης του έργου στο λιγότερο δυνατό κόστος.

Τα μη επιβλαβή σκόπιμα σφάλματα αλληλεπιδράσεων μπορεί να προκύψουν από ενέργειες του διαχειριστή ενός συστήματος. Έχουν ως σκοπό την αντιμετώπιση μιας απρόσμενης κατάστασης που έχει συμβεί, με σκόπιμη μεταβολή μιας διαδικασίας χωρίς να λάβει υπόψη την πιθανές βλαβερές συνέπειες των πράξεών του. Η κατηγορία αυτή έχει ως χαρακτηριστικό την αναγνώριση του λάθους έπειτα από μη αποδεκτή συμπεριφορά του συστήματος. Άρα η επακόλουθη αποτυχία είναι προϊόν πολλών αίτιων π.χ. του αρμόδιου τεχνικού, που δεν αποσαφήνισε τις απαραίτητες προϋποθέσεις της υπηρεσίας, του σχεδιαστή που δεν περιέγραψε την περίπτωση αυτή στο documentation, του διαχειριστή που εκτέλεσε εις γνώση του την αλλαγή χωρίς να συνειδητοποιήσει ότι οι συνέπειες των πράξεων του ήταν λάθος εκείνη την χρονική στιγμή, που μπορεί και να μην ήταν αλλά να είχαν αντίκτυπο σε λειτουργεί πέρα του τομέα ευθύνης του. Οπότε ο ανθρώπινος παράγοντας είναι κρίσιμο στοιχείο για την αξιοπιστία όταν αλληλεπιδρά με το σύστημα καθώς υποπίπτει σε λάθη πολλές φορές ακόμα και άθελά του.

Τα επιβλαβή σφάλματα χωρίζονται σε δύο κατηγορίες, την επιβλαβή λογική η οποία περιέχει σφάλματα ανάπτυξης όπως *Δούρειους Ίππους*, *λογικές βόμβες* ή *βόμβες χρονισμού* και τις πίσω πόρτες (trapdoors). Οι τελευταίες είναι τρύπες στην ασφάλεια ενός συστήματος οι οποίες δημιουργήθηκαν εσκεμμένα από τους σχεδιαστές ή συντηρητές για μη εξουσιοδοτημένη πρόσβαση. Επιπλέον μπορεί να περιέχουν λειτουργικά σφάλματα όπως είναι οι *ιοί* και τα *σκουλήκια* (viruses and worms) που επηρεάζουν τα μητρώα του λειτουργικού συστήματος.

Η δεύτερη κατηγορία είναι οι *εισχωρήσεις* σε ένα σύστημα. Παρουσιάζουν ενδιαφέρον καθώς υπάρχουν αρκετές ομοιότητες ανάμεσα σε μια εισχώρηση η οποία εκμεταλλεύεται ένα εσωτερικό σφάλμα με την μορφή φυσικού εξωτερικού σφάλματος και στην έλλειψη αμυντικών μηχανισμών. Ο εξωτερικός χαρακτήρας των εισχωρήσεων δεν αποκλείει την πιθανότητα οι διαχειριστές να υπερβαίνουν τα δικαιώματά τους, παρά το γεγονός ότι μπορεί να αξιοποιούν φυσιολογικά μέσα να επιτύχουν τους σκοπούς τους. Τα παραπάνω είδη σφαλμάτων εντάσσονται στην υποκατηγορία των *επιθέσεων*.

Μερικές φορές τα σφάλματα σχεδιασμού που επηρεάζουν το λογισμικό προκαλούν το φαινόμενο της *γήρανσης λογισμικού* (software aging). Τα χαρακτηριστικά του είναι η σταδιακή εξέλιξη λαθών, με αποτέλεσμα την υποβάθμιση της λειτουργίας του συστήματος. Μερικά παραδείγματα αποτελούν η επέκταση λαθών, η διαρροή πληροφορίας στις μνήμες, ο κατακερματισμός χώρου αποθήκευσης δίσκων και βάσεων δεδομένων (information bloating and leaking).

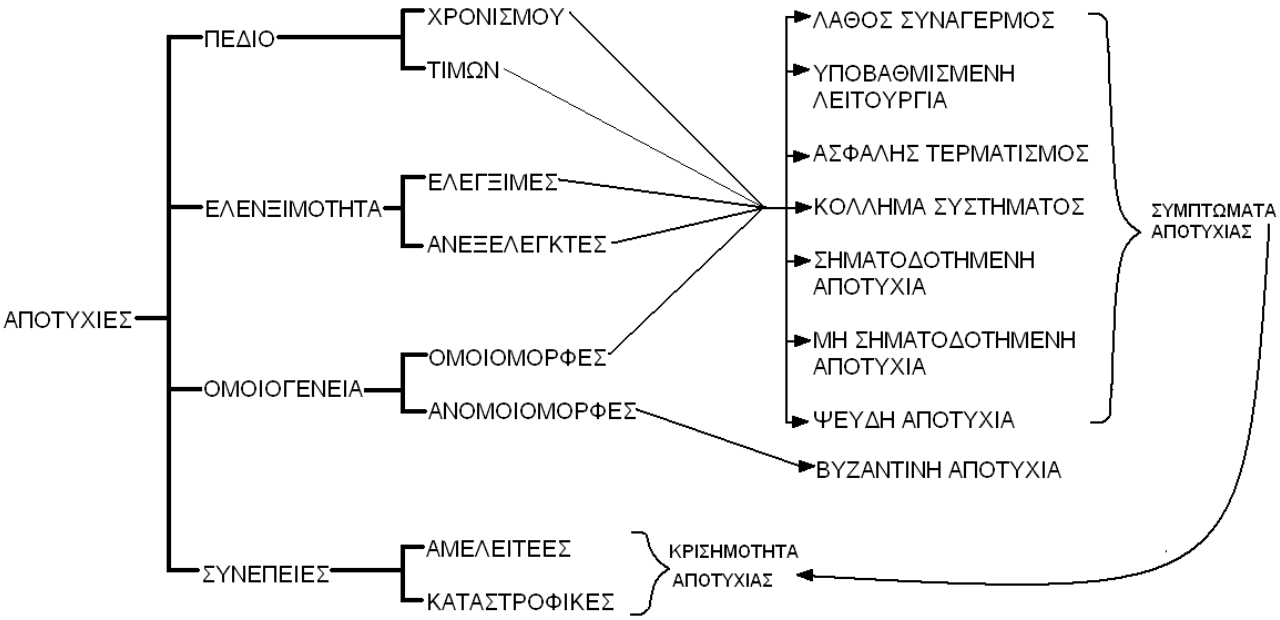
Στο σημείο αυτό πρέπει να αναφερθεί ότι κάθε σφάλμα δεν επηρεάζει με τον ίδιο τρόπο ένα σύστημα. Οι συνθήκες κάτω από τις οποίες ένα σύστημα αποτυγχάνει εκφράζεται από τις *καταστάσεις αποτυχίας. Α*υτές μελετούνται με βάση την συμπεριφορά, και τα παραγόμενα αποτελέσματα, από τα δοκιμασμένα πιθανά αίτια μιας αποτυχίας του συστήματος. Κατατάσσονται ανάλογα με την *κρισιμότητα της αποτυχίας*. Ουσιαστικά έχει να κάνει με τον βαθμό που η εσφαλμένη υπηρεσία εμποδίζει την συνολική λειτουργία του συστήματος. Ο τρόπος με τον οποίο αποτυγχάνει ένα σύστημα μελετάται από τέσσερις σκοπιές, το *πεδίο* που βρίσκεται το λάθος, την *ελεγξιμότητα του,* την *ομοιογένεια* του και τέλος τις *συνέπειες του,* τόσο για το ίδιο το σύστημα αλλά και το περιβάλλον στο οποίο αλληλεπιδρά.

Το *πεδίο* μιας αποτυχίας περιγράφει τον τύπο του σφάλματος που έχει συμβεί. Συγκεκριμένα μπορεί να είναι μια *αποτυχία χρονισμού* ή *αποτυχία τιμής*, χωρίς να αποκλείεται η περίπτωση του ταυτόχρονου συμβάντος και των δύο. Οι χρονικές αποτυχίες, οφείλονται σε πρόωρο ή καθυστερημένο χρονισμό της παράδοσής πληροφοριών στην διεπαφή της υπηρεσίας. Από την άλλη πλευρά, οι αποτυχίες τιμών έχουν να κάνουν με την παράδοση λανθασμένων αποτελεσμάτων στην διεπαφή της υπηρεσίας. Έτσι το σύστημα ως συνέπεια μπορεί να κολλήσει, δηλαδή να μην μεταβάλλει την κατάσταση του έπειτα από ενέργειες του χρήστη αλλά να συνεχίζει να λειτουργεί με υποβαθμισμένη λειτουργία.

Η *ελεγξιμότητα* αφορά τον τρόπο συμβάντος μιας αποτυχίας. Μια αποτυχία μπορεί να είναι ελέγξιμη αν έχει ενεργοποιηθεί κατάλληλο προειδοποιητικό σήμα, που θα ενημερώνει τον διαχειριστή για το τι ακριβώς έχει συμβεί αλλά και για τη θέση του στο σύστημα. Έτσι εκτελείται ο ασφαλής τερματισμός του. Έχουμε *ανεξέλεγκτη αποτυχία* όταν δεν πυροδοτείται κάποιος μηχανισμός προειδοποίησης και το σύστημα σταματά να προσφέρει την υπηρεσία του. Οι μηχανισμοί προειδοποίησης παράγουν κατάλληλο μήνυμα για την σηματοδοτημένη αποτυχία, αλλά πρέπει να αναφερθεί και η περίπτωση της ψευδής αποτυχίας (false alarm). Σύμφωνα με αυτή ο μηχανισμός μπορεί να πυροδοτηθεί από κάτι απρόσμενο ή να έχει συμβεί σφάλμα σε αυτόν, χωρίς να υπάρχει πρόβλημα με την υπηρεσία.

Όταν η εσφαλμένη υπηρεσία προσφέρεται σε όλους τους χρήστες του και γίνεται αντιληπτό από όλους με κοινές συνέπειες τότε η αποτυχία αυτή έχει ομοιογένεια, π.χ. ένας ανενεργός υπολογιστής server εμφανίζει ανάλογο μήνυμα στους περιηγητές. Αν αυτή προσφέρεται σε μερικούς χρήστες μόνο, οι οποίοι έχουν διαφορετικές μορφές της εσφαλμένης υπηρεσίας και παρόλα αυτά υπάρχουν ορισμένοι που συνεχίζουν να έχουν σωστή υπηρεσία, τότε έχουμε ανομοιογένεια στην αποτυχία. Τέτοιας μορφής αποτυχία συχνά αναφέρεται και ως Βυζαντινή αποτυχία, από το πρόβλημα των στρατηγών του Βυζαντινού στρατού και αποτελεί από τα δυσκολότερα προβλήματα επίλυσης στην σύγχρονα συστήματα ανοχής σφαλμάτων[]. Η δυσκολία βρίσκεται στην ταυτοποίηση, για παράδειγμα, των δεδομένων ως σωστή, είτε εσφαλμένη υπηρεσία, καθώς η κατάστασή της συνεχώς μεταβάλλεται.

Τέλος, όλες οι αποτυχίες έχουν τις συνέπειες τους. Χαρακτηρίζονται *αμελητέες* όταν τα οφέλη από την υπηρεσία που παρέχεται, υπερκαλύπτει το κόστος της αποτυχίας, όχι αναγκαστικά οικονομικό αλλά ποιοτικό, χρονικό. Κ*αταστροφικές* είναι όταν οι επιπτώσεις της αποτυχίας είναι τάξεις μεγέθους πάνω από τις αναμενόμενες προσδοκίες και εξαλείφουν τα πλεονεκτήματα χρήσης μιας υπηρεσίας. Οι τελευταίες έχουν επιπτώσεις τόσο στους χρήστες αλλά και στο περιβάλλον μέσα στο οποίο δρουν αλλά μπορεί να έχει και επιπτώσεις στον φυσικό κόσμο.



Εικόνα 2.5: Ανάλυση των αποτυχιών σε κατηγορίες με βάση το πεδίο, την ελενξιμότητα, την ομοιογένεια και τις συνέπειες τους.

2.4 Τα Χαρακτηριστικά της Αξιοπιστίας

Ένα σύστημα προορίζεται για την προσφορά συγκεκριμένων υπηρεσιών στους χρήστες του. Ανάλογα όμως με την εφαρμογή, πρέπει να υπολογιστούν τα απαραίτητα χαρακτηριστικά, που διέπουν την λειτουργία του. Ως *προδιαγραφές αξιοπιστίας εννοούμε την περιγραφή των στόχων, σύμφωνα με τα αποδεκτά επίπεδα σφάλματος, την κρισιμότητα των καταστάσεων αποτυχίας για ορισμένο πλήθος σφαλμάτων και για συγκεκριμένο περιβάλλον δράσης.* Η αξιοπιστία όμως είναι μια έννοια η όποια ενσωματώνει τα ακόλουθα βασικά χαρακτηριστικά:

- ***Διαθεσιμότητα (Availability) :*** *Η ετοιμότητα για προσφορά ορθής υπηρεσίας.*
- ***Εξαρτησιμότητα (Reliability) :*** *Η συνεχής προσφορά της υπηρεσίας.*
- ***Ασφάλεια (Safety):*** *Η έλλειψη καταστροφικών επιπτώσεων στον χρήστη και το περιβάλλον δράσης του.*
- ***Εχεμύθεια (Confidentiality):*** *Η απουσία απόκρυψης πληροφοριών, από εξουσιοδοτημένα άτομα (και μη) ή συστήματα.*
- ***Ακεραιότητα (Integrity):*** *Η έλλειψη ακατάλληλων μεταβολών των καταστάσεων του συστήματος.*
- ***Συντηρησιμότητα (Maintainability) :*** *Η δυνατότητα επιδιόρθωσης και μεταβολών ενός συστήματος.*

Η Διαθεσιμότητα και η Εξαρτησιμότητα είναι τα άκρως απαραίτητα χαρακτηριστικά για οποιοδήποτε συστήματα και κυρίως τα υπολογιστικά. Κάθε σύστημα πρέπει να είναι σε θέση ετοιμότητας, δηλαδή να προσφέρει τα αναμενόμενα ανά πάσα στιγμή και να εκτελεί τις ανάλογες ενέργειες.

Η Ασφάλεια είναι σύνθετη έννοια, η οποία αποτελείται από την Εχεμύθεια την Ακεραιότητα και την Διαθεσιμότητα. Ουσιαστικά εκφράζει την παρεμπόδιση τρίτων

από την διαχείριση και διαγραφή πληροφοριών του συστήματος. Ένας συνοπτικός ορισμός θα ήταν, *η απουσία αναρμόδιας εισόδου και διαχείρισης της κατάστασης ενός συστήματος*. Επιπλέον μπορεί να εκφραστεί με την έννοια της προστασίας από απρόσμενες συνθήκες και την παρεμπόδιση των συνεπειών.

Η Ακεραιότητα είναι υποχρεωτικό χαρακτηριστικό για την Διαθεσιμότητα, την Εξαρτησιμότητα και την Ασφάλεια. Όταν ένα υπολογιστικό σύστημα υλοποιεί υπηρεσίες επικύρωσης για τις υπομονάδες του χαρακτηρίζεται ως ακέραιο. Όταν όμως ακατάλληλες μεταβολές παρεμποδίζουν τις καταστάσεις των συστημάτων, προκαλούν βλάβες. Επομένως λέμε ότι καταργείται το κριτήριο αυτό.

Η Συντηρησιμότητα μπορεί και αυτή να επεκταθεί από την απλή επιτυχή επιδιόρθωση, σε μια μορφή προσαρμογής και βελτιστοποίησης του συστήματος. Είναι το κριτήριο με το οποίο χαρακτηρίζουμε τα συστήματα ως επιδιορθώσιμα. Αν υλοποιεί μηχανισμούς εντοπισμού και διόρθωσης λέμε ότι είναι και συντηρήσιμο, καθώς φροντίζει για την ομαλή επιστροφή στην προσφερόμενη υπηρεσία. Η συντήρηση αρκετές φορές πραγματοποιείται και από ανθρώπινο εξειδικευμένο προσωπικό.

Εκτός από τα βασικά χαρακτηριστικά που αναφέρθηκαν προηγουμένως μπορούν να οριστούν αρκετά δευτερεύοντα που αποτελούν συνδυασμό τους ή εξειδικεύσεις των έξι παραπάνω. Για παράδειγμα η Προστασία εμπεριέχει ταυτόχρονα τις έννοιες της Διαθεσιμότητας για εγκεκριμένους χρήστες, αν μιλάμε για ένα σύστημα λογισμικού, καθώς και την Εχεμύθεια και την Ακεραιότητα. Ένα ακόμη είναι η Ευρωστία, δηλαδή η αξιοπιστία με γνώμονα τα εξωτερικά σφάλματα. Περιγράφει την αντίδραση του συστήματος σε πλήθος σφαλμάτων που προέρχονται από το περιβάλλον δράσης ρου και όχι από το ίδιο. Άλλο εξίσου σημαντικό χαρακτηριστικό είναι η Υπευθυνότητα, δηλαδή η διαθεσιμότητα και η ακεραιότητα των πράξεων του ατόμου που εκτελεί μια εργασία. Η Αυθεντικότητα έχει να κάνει με την ακεραιότητα του περιεχομένου και της προέλευσης ενός μηνύματος π.χ. σε ένα δίκτυο υπολογιστών.

Ανάλογα με τα χαρακτηριστικά που επιθυμούμε να αποδώσουμε σε ένα σύστημα επηρεάζεται η ισορροπία των τεχνικών που θα επιστρατευτούν για να προσδώσουμε τα αναμενόμενα επίπεδα εμπιστοσύνης σε ένα υπολογιστικό σύστημα. Αυτό πολλές φορές δημιουργεί πρόβλημα. Πολλά από τα χαρακτηριστικά αντικρούονται μεταξύ τους ως προς τις προδιαγραφές αξιοπιστίας. Ο γενικός κανόνας είναι η τήρηση των απαραίτητων συμβιβασμών μεταξύ των κριτηρίων, αλλιώς χάνεται η ισορροπία, πράγμα ανεπιθύμητο .

2.5 Τα Μέσα Εξασφάλισης της Αξιοπιστίας

2.5.1 Ανοχή σφαλμάτων

Η *Ανοχή Σφαλμάτων (fault tolerance)* έχει ως σκοπό την ανάπτυξη συστημάτων που λειτουργούν σωστά ακόμα και στην παρουσία βλαβών[2]. Η ανοχή σφαλμάτων επιτυγχάνεται με τη χρήση κάποιας μορφής πλεονασμού. Ο πλεονασμός είναι η μέθοδος με την οποία ένα σφάλμα μπορεί να *αποκρυφτεί,* να *ανιχνευτεί* είτε να τεθεί σε *περιορισμό* ώστε το σύστημα να ξεκινήσει την *ανάκτηση.* Είναι απαραίτητος, όμως για να γίνει κατανοητός, θα πρέπει να αναφέρουμε ένα παράδειγμα. Έστω δύο πανομοιότυπα εξαρτήματα συνδεδεμένα παράλληλα. Το σύστημα δεν έχει δυνατότητες ανοχής σφαλμάτων, εκτός εάν υπάρχει κάποια μορφή

ελέγχου, είτε σε υλικό με κύκλωμα, είτε σε λογισμικό με πρόγραμμα, η οποία αναλύει τα αποτελέσματα του και επιλέγει κάθε φορά το σωστό.

Η *απόκρυψη σφάλματος (fault masking)* είναι η διαδικασία με την οποία εξασφαλίζονται μόνο σωστές τιμές στην έξοδο του συστήματος. Αυτό γίνεται εφαρμόζοντας προληπτικά μέτρα στο σύστημα, ώστε να μην επηρεαστεί από λάθη που ενδέχεται να συμβούν κάποια στιγμή στο μέλλον. Δεδομένου ότι το σύστημα δεν εμφανίζει το συμβάν του σφάλματος, περνά απαρατήρητο από τον χρήστη. Για παράδειγμα, μία μνήμη με κώδικα διόρθωσης σφαλμάτων διορθώνει τα ελαττωματικά bits πριν το σύστημα χρησιμοποιήσει τα δεδομένα. Άλλο ένα παράδειγμα απόκρυψης σφάλματος είναι τα συστήματα πλεονασμού με ψηφοφορία πλειοψηφίας στην έξοδο, που κρύβουν τις ελαττωματικές μονάδες για να συνεχιστεί η ομαλή λειτουργία.

Για να προσδιοριστεί ένα λάθος μέσα σε ένα σύστημα χρησιμοποιείται η *ανίχνευση σφάλματος*. Τεχνικές για την ανίχνευση σφαλμάτων είναι οι δοκιμές επικύρωσης και σύγκρισης. Οι δοκιμές επικύρωσης χρησιμοποιούνται για συστήματα που αξιοποιούν όμοια εξαρτήματα ως μέθοδο εφεδρείας. Από την άλλη πλευρά οι δοκιμές αποδοχής συναντώνται σε συστήματα λογισμικού, όπου το αποτέλεσμα ενός προγράμματος υποβάλλεται σε δοκιμή. Εάν επιτύχει τότε το πρόγραμμα συνεχίζει την εκτέλεση του, ειδάλλως έχει συμβεί ένα σφάλμα. Η σύγκριση είναι μια εναλλακτική λύση για την ανίχνευση σφαλμάτων. Για να επιτευχθεί θα πρέπει να αποτελείται από δύο πανομοιότυπα εξαρτήματα που εκτελούν την ίδια λειτουργία. Έτσι λοιπόν τα αποτελέσματα των εξόδων τους συγκρίνονται από ένα συγκριτή και μια διαφωνία σηματοδοτεί ένα σφάλμα σε μια μονάδα.

Ο *εντοπισμός σφάλματος (fault detection)* είναι η τεχνική με την οποία γίνεται ο προσδιορισμός του σημείου στο σύστημα που ένα σφάλμα έχει συμβεί. Μια αποτυχημένη δοκιμή επικύρωσης δεν μπορεί να χρησιμοποιηθεί για να εντοπίσει ένα σφάλμα. Η μόνη πληροφορία που μπορεί κάποιος να αποκομίσει είναι ότι απλώς υπάρχει λάθος σε κάποιο μέρος του συστήματος και όχι για την περιοχή του συγκεκριμένα. Ομοίως, όταν μια διαφωνία προκύψει κατά τη σύγκριση δύο εξαρτημάτων, δεν είναι σε θέση να εξακριβωθεί ποια από τα δύο απέτυχε. Έτσι λοιπόν με αυτή την τεχνική βρίσκουμε και την προβληματική μονάδα ώστε να εκτελεστεί η επιδιόρθωση.

Ο *περιορισμός σφάλματος (fault containment)* απομονώνει ένα σφάλμα και προλαμβάνει την επίδραση του σε όλο το σύστημα. Αυτό επιτυγχάνεται συνήθως με συχνούς ελέγχους ανίχνευσης, που περιλαμβάνουν πρωτόκολλα ARQ και επιβεβαίωσης των συνδέσεων μεταξύ των εξαρτημάτων. Το πρώτο αξιοποιείται στα συστήματα λογισμικού κατά την μετάδοση πληροφορίας μέσω δικτύου υπολογιστών ενώ το δεύτερο έχει καθολική εμβέλεια.

Αν εντοπιστεί ένα ελαττωματικό εξάρτημα, ένα σύστημα ανακτά την λειτουργία του, με αναδιαμόρφωση των εξαρτημάτων του, έτσι ώστε να το απομονώσει από το υπόλοιπο σύστημα και να επανέλθει σε ορθή κατάσταση λειτουργίας. Οι τρόποι που εκτελείται η διαδικασία αυτή είναι με αντικατάσταση του προβληματικού εξαρτήματος με εφεδρικό. Μια άλλη περίπτωση η είναι υποβαθμισμένη λειτουργία, οπού απομακρύνεται το προβληματικό εξάρτημα και το σύστημα συνεχίζει την λειτουργία του υποβαθμισμένα, προσφέροντας ακόμη μέρος από την υπηρεσία του.

2.5.2 Πρόληψη σφαλμάτων

Η *Πρόληψη Σφαλμάτων (fault prevention)* είναι ένα σύνολο τεχνικών που προσπαθούν να αποτρέψουν την είσοδο λαθών στο σύστημα. Ουσιαστικά αποτελεί το πρώτο στάδιο αντιμετώπισης των σφαλμάτων. Με την πρόληψη είμαστε σε θέση να αξιοποιούμε τεχνικές ποιοτικού ελέγχου κατά την διαδικασία σχεδιασμού. Έτσι λοιπόν ένα σχέδιο το οποίο είναι καθορισμένο να υλοποιηθεί θα αξιολογηθεί, θα εξεταστούν τα μέρη από τα οποία αποτελείται ξεχωριστά, ως προς τις προδιαγραφές και θα τελειώσει με τις δοκιμές. Για ένα σύστημα λογισμικού, αυτό περιλαμβάνει δομημένο τρόπο προγραμματισμού, τον διαχωρισμό του σε υποενότητες και τις δοκιμές ελέγχου.

Πολλά από τα σφάλματα που αφορούν τις προδιαγραφές μπορούν εύκολα να εξαλειφτούν με αυστηρή παρατήρηση του σχεδίου. Ένα σχέδιο που είναι δοκιμασμένο, προσφέρει την δυνατότητα πρόληψης, δηλαδή να εντοπίζονται λάθη και να αντιμετωπίζονται καταστάσεις πριν τη δημιουργία τους. Ακόμη οι βλάβες που οφείλονται σε εξωτερικές διαταραχές, εμποδίζονται με θωράκιση και σφάλματα λειτουργίας αποφεύγονται με συχνές διαδικασίες συντήρησης. Τέλος οι σκόπιμες κακόβουλες επιθέσεις που προκαλούνται από ιούς ή χάκερ και αντιμετωπίζονται εκ των προτέρων, για παράδειγμα με ένα firewall ή παρόμοια μέσα προσδίδοντας ένα μέτρο ασφάλειας από το αρχικό στάδιο της ανάπτυξης.

2.5.3 Αφαίρεση σφαλμάτων

Η διαδικασία αυτή στοχεύει στη ελάττωση του αριθμού των σφαλμάτων που είναι παρόντα στο σύστημα και γίνεται κατά τη διάρκεια της ανάπτυξης και λειτουργίας του συστήματος[2]. Το αναπτυξιακό στάδιο ενός σχεδίου περιλαμβάνει τρία στάδια: *την επαλήθευση*, τη *διάγνωση* και τη *διόρθωση*. Από την άλλη πλευρά, η διάρκεια ζωής ενός συστήματος αποτελείται από δύο μέρη την *διορθωτική* και την *προληπτική συντήρηση (corrective and preventive maintainance)*. Η φάση ελέγχου του συστήματος για να διευκρινιστούν αν λειτουργεί με επιτυχία κάτω από συγκεκριμένες συνθήκες ονομάζεται επαλήθευση. Στην περίπτωση που ικανοποιούνται οι συνθήκες, όλα λειτουργούν ομαλά. Στις περιπτώσεις που δεν ικανοποιούνται γίνεται διάγνωση για τους λόγους που δεν μπορεί να προσφέρει την αναμενόμενη λειτουργία του και στη συνέχεια ακολουθεί η επιδιόρθωση του.

Με την *προληπτική συντήρηση* έχουμε την αντικατάσταση προβληματικών εξαρτημάτων και μερικές φορές γίνονται τροποποιήσεις στο σύστημα. Με τον τρόπο αυτό επεκτείνεται αισθητά η αξιοπιστία καθώς αντιμετωπίζεται το φαινόμενο της φυσιολογικής φθοράς των ηλεκτρονικών εξαρτημάτων. Τέλος η *διορθωτική συντήρηση* εκτελείται μετά την επίδραση ενός σφάλματος ή την πιθανή αποτυχία μέρους και ολόκληρου του συστήματος επειδή έχει ως στόχο την επαναφορά του σε συνθήκες ορθής λειτουργίας.

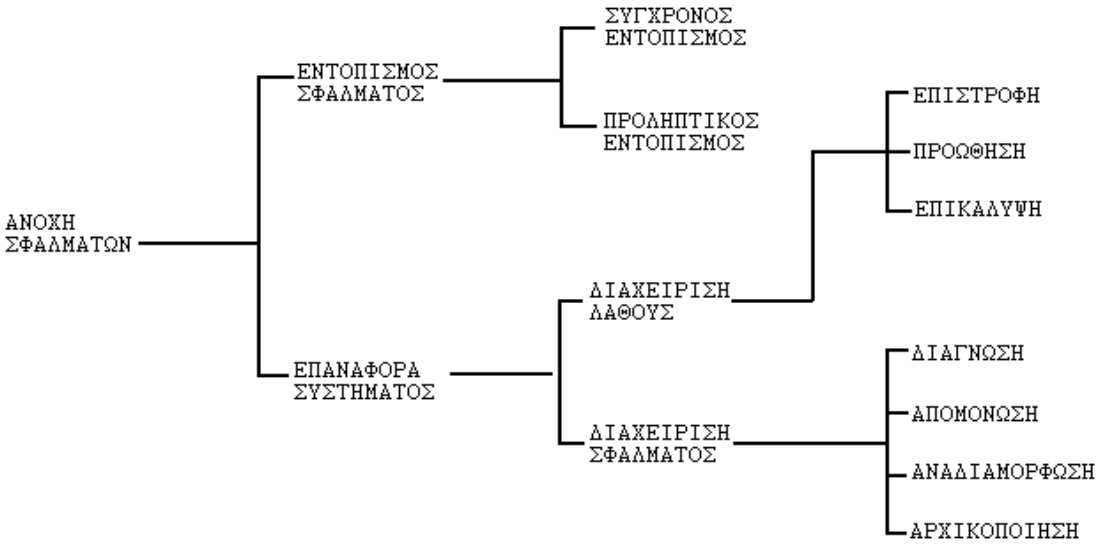
2.5.4 Πρόβλεψη σφαλμάτων

Όταν υπάρχουν σφάλματα στο σύστημα, πρέπει να είμαστε σε θέση να μπορούμε να εκτιμήσουμε την κατάσταση. Ο τρόπος με τον οποίο επιδρά ένα σφάλμα στο σύστημα μπορεί να δημιουργήσει νέα σφάλματα και να επέλθει σταδιακά μια αποτυχία λειτουργίας. Οπότε θα πρέπει να υπάρχει ένας μηχανισμός

πρόβλεψης της επίδρασης κάθε πιθανού σφάλματος και να προβλέπει την εξέλιξη του, ώστε να αποφευχθούν οι συνέπειες του. Η *Πρόβλεψη Σφάλματος* γίνεται αξιολογώντας την συμπεριφορά του συστήματος στις κατηγορίες λαθών που μπορεί να προκληθούν ανάλογα πάντα με την εφαρμογή που σχεδιάζεται. Η αξιολόγηση αναλύεται σε ποιοτική και ποσοτική.

Ο υπολογισμός των πιθανοτήτων για κάθε κατάσταση αποτυχίας και το συνδυασμό τους, έχει να κάνει με την ποιοτική αξιολόγηση ενός συστήματος. Η ποσοτική αξιολόγηση έχει να κάνει με τον υπολογισμό του ποσοστού, σε πιθανότητες, που το σύστημα ικανοποιεί τα χαρακτηριστικά της αξιοπιστίας. Μερικοί τρόποι εκτέλεσης των παραπάνω ενεργειών είναι η μέτρηση της εφεδρείας του συστήματος, ώστε να μετρηθούν τα πιθανά επιτυχή μονοπάτια λειτουργίας, τα οποία ξεκινούν από την είσοδο και καταλήγουν στην έξοδο. Άλλος τρόπος είναι η κατανόηση του τρόπου με τον οποίο ένα σφάλμα μπορεί μεταβάλλει την λειτουργία του συστήματος και την ανοχή του, ώστε να βρεθούν νέοι τρόποι αντιμετώπισης.

Για την κατανόηση των παραπάνω τρόπων, θα αναφερθούν χαρακτηριστικές τεχνικές στα επόμενα κεφάλαια που εφαρμόζονται στα υπολογιστικά συστήματα με επιτυχία και ανήκουν στην ευρύτερη κατηγορία της ανοχής σφαλμάτων. Το παρακάτω διάγραμμα απεικονίζει στα αριστερά τις κατηγορίες της ανοχής σφαλμάτων, κατά τύπο και που εντάσσονται οι τεχνικές της[1].



Εικόνα 2.6: Η ανοχή σφαλμάτων και οι τεχνικές της.

Με τον εντοπισμό των σφαλμάτων βρίσκουμε το λάθος. Αυτό είτε γίνεται σύγχρονα είτε προληπτικά. Ο σύγχρονος εντοπισμός ενός λάθους γίνεται κατά την διάρκεια λειτουργίας του συστήματος. Ο προληπτικός εντοπισμός πραγματοποιείται κατά την περίοδο αναβολής της λειτουργίας ενός συστήματος, όπως είναι η περίοδος συντήρησης και ανίχνευσης λανθανόντων σφαλμάτων.

Για την επαναφορά του συστήματος σε κατάσταση ορθής λειτουργίας γίνεται διαχείριση από δυο σκοπιές.

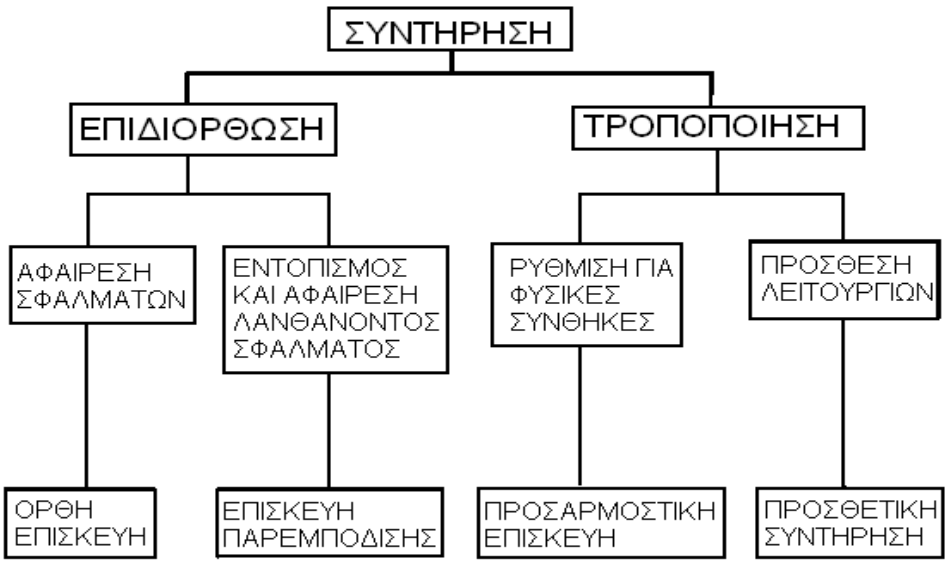
Η μια είναι από την πλευρά των λαθών, που έχουμε τρεις τεχνικές: *την επιστροφή, την προώθηση και την επικάλυψη*. Με την επιστροφή έχουμε επαναφορά του συστήματος σε προηγούμενη κατάσταση, όπου δεν υπήρχε το λάθος που

έχουμε στην τωρινή κατάσταση. Έτσι το σύστημα αποφεύγει το λάθος και το σύστημα συνεχίζει κανονικά την λειτουργία του. Η προώθηση (forwarding) είναι η αντίστροφη διαδικασία καθώς προχωράμε το σύστημα σε μεταγενέστερη κατάσταση ώστε να αποφύγουμε το λάθος. Η επικάλυψη χρησιμοποιείται για την κάλυψη λαθών από το σύστημα, για να διασφαλιστεί η λειτουργία του. Ενδεικτικές μέθοδοι είναι οι ψηφοφορίες πλειοψηφίας και μέσου όρου (majority and average voting). Σε αυτήν γίνεται χρήση πλεονασμού στα απαραίτητα επίπεδα για να μην επηρεάζεται η λειτουργία του συστήματος.

Η άλλη είναι από την σκοπιά των αιτιών, όπου εκτελούνται διαδικασίες της διάγνωσης, της απομόνωσης, της αναδιαμόρφωσης και της αρχικοποίησης. Η διάγνωση ασχολείται με την αναγνώριση και καταγραφή των αιτιών των λαθών τοπολογικά και βάσει τύπου (βλ. ΠαραρτημαΑ2.1). Η απομόνωση σφάλματος χρησιμοποιεί τον φυσικό και λογικό αποκλεισμό προβληματικών εξαρτημάτων από το σύστημα και την μετατροπή λαθών σε προγενέστερη κατάσταση, την λανθάνουσα. Η αναδιαμόρφωση εκτελείται είτε ενεργοποιώντας εφεδρικές μονάδες για την διατήρηση της λειτουργίας, είτε αναθέτει την υπηρεσία σε μη προβληματικές μονάδες. Δύο γνωστοί τρόποι αποτελούν ο *πλεονασμός* και ο *σχεδιασμός με διαφορετικότητα.* Ο σχεδιασμός αυτός χρησιμοποιεί διάφορες υλοποιήσεις σε υλικό και λογισμικό για την προσφορά της ίδιας υπηρεσίας. Έτσι σε περίπτωση αποτυχίας μιας από αυτές, γίνεται ανακατεύθυνση σε νέα υλοποίηση, χωρίς το σύστημα να σταματήσει την προσφορά της υπηρεσίας.

Με την αρχικοποίηση ελέγχεται η νέα αναδιαμόρφωση, αποθηκεύονται οι καταστάσεις των εξαρτημάτων και καταγράφονται σε πίνακες οι μεταβλητές του συστήματος. Έτσι λοιπόν αναβαθμίζεται εκ νέου η διάρκεια ζωής του συστήματος. Αξιοσημείωτο είναι το γεγονός ότι η επισκευή και η συντήρηση των συστημάτων και η ανοχή σφαλμάτων είναι σχετικές έννοιες. Η διαφορά τους βρίσκεται στο γεγονός ότι για την συντήρηση συνεπάγεται η συμμετοχή ενός εξωτερικού παράγοντα, πχ. ένας επισκευαστής, δοκιμαστικός εξοπλισμός ακόμα και με τη χρήση απομακρυσμένης επιδιόρθωσης. Επιπλέον, ο επισκευαστής είναι μέρος της αφαίρεσης σφαλμάτων και της πρόβλεψης, διότι γνωρίζει τις καταστάσεις επισκευής του κάθε συστήματος. Η ανοχή σφαλμάτων δεν προϋποθέτει την εξωτερική επισκευή, την περιέχει σαν έννοια αλλά έχει την δυνατότητα να δρα αυτοματοποιημένα πολλές φορές από το ίδιο το σύστημα.

Τέλος η συντήρηση χωρίζεται στην επιδιόρθωση και στην τροποποίηση[1]. Η επιδιόρθωση ενσωματώνει την διαδικασία του εντοπισμού και της αφαίρεσης σφάλματος. Η τροποποίηση αφορά την παραμετροποίηση μονάδων του συστήματος ανάλογα με τις φυσικές συνθήκες που επικρατούν και την πρόσθεση νέων λειτουργιών, για την βελτίωση της προσφερόμενης υπηρεσίας. Ένα παράδειγμα επιδιορθωτικής συντήρησης αποτελεί η αναβάθμιση εκδόσεων λογισμικού για την έκλειψη σφαλμάτων (software bugs) προηγούμενων εκδόσεων. Σε αυτό έχουμε επισκευή καθώς παρεμποδίστηκαν τα αίτια προβλημάτων και αφαίρεση των σφαλμάτων. Ένα παράδειγμα συντήρησης με τροποποίηση είναι η πρόσθεση νέων λειτουργιών σε ένα σύστημα λογισμικού και η ενίσχυση της συσκευασίας ενός συστήματος υλικού για την αντιμετώπιση της υγρασίας.



Εικόνα 2.7: Η συντήρηση συστημάτων και οι διαφορές μεταξύ των μεθόδων εκτέλεσής της.

2.6 Μοντελοποίηση και Αξιολόγηση της Αξιοπιστίας

Ένα σύστημα για να είναι αξιόπιστο πρέπει να έχει σχεδιαστεί κάτω από αυστηρά καθορισμένους κανόνες. Το σύνολο των κανόνων αυτών σχηματίζουν το μοντέλο αξιοπιστίας του συστήματος. Είναι στενά συνδεμένα και με τα μοντέλα σφάλματος, τα οποία εφαρμόζονται για την ανοχή σφαλμάτων. Αυτά ανάλογα με το κυκλωματικό επίπεδο που θα χρησιμοποιηθούν, μεταβάλλουν το κόστος και την ακρίβεια της αντιμετώπισης των σφαλμάτων, όπως υποδεικνύει και το παρακάτω σχήμα.

Εικόνα 2.8: Μοντέλα σφάλματος ανάλογα με το επίπεδο ανάπτυξης

Η αξιοπιστία εισάγεται κατά το στάδιο της ανάπτυξης ενός σχεδίου και επομένως χρειάζεται αναλυτική περιγραφή των προδιαγραφών του. Ακόμη είναι αναγκαίο να αναπτυχθούν τα απαραίτητα μοντέλα για κάθε εξάρτημα, των

συνδέσεων τους και ολόκληρου του υπολογιστικού συστήματος. Άρα πρέπει να διατυπωθούν οι τρόποι αυτοί ώστε ένας σχεδιαστής να μπορεί να την υπολογίζει σε οποιοδήποτε στάδιο της παραγωγής, είτε αυτό είναι ένα η πολλά εξαρτήματα ή το σύστημα στο σύνολο του. Τα μοντέλα αξιοπιστίας χωρίζονται σε υποενότητες που περιέχουν μηχανισμούς για την μελέτη και μοντελοποίηση αξιόπιστων συστημάτων και μπορούν να εφαρμοστούν σε κάθε επίπεδο ανάπτυξης ξεχωριστά. Ουσιαστικά βελτιώνοντας την αξιοπιστία ενός συγκεκριμένου εξαρτήματος στο επίπεδο πυλών θα βελτιώσει την αξιοπιστία του μπλοκ που θα το περιέχει. Έτσι υπολογίζεται η αξιοπιστία σε κάθε επίπεδο και εμφανίζονται τρόποι βελτίωσης της στην ολότητα του συστήματος. Η μέθοδος του διαίρει και βασίλευε είναι η επικρατούσα λογική.

Για την μέτρηση της πιθανότητας αποτυχίας πρέπει να υπολογιστεί η αξιοπιστία του με ανάλογο τρόπο. Ο τρόπος αυτός αναλύεται με τις εξισώσεις της αξιοπιστίας, διαθεσιμότητας και συντηρησιμότητας. Επίσης οι τρόποι πρόβλεψης της πιθανότητας σφάλματος χρησιμοποιεί βασικά στοιχεία στατιστικής και πιθανοτήτων και βοηθούν στην ανάπτυξη αξιόπιστων συστημάτων.

Όπως αναφέραμε όλα έχουν να κάνουν με πιθανότητες, οπότε θα πρέπει να ορίσουμε την πιθανότητα ενός σφάλματος που είναι γνωστό ως *ποσοστό αποτυχίας*. Συνήθως υπολογίζεται για μια περίοδο 10^6 ωρών ή περισσότερων ωρών για κρίσιμες αποστολές. Πρέπει να σημειωθεί ότι η παράμετρος αυτή καθορίζεται με βάση μια συνθήκη, καθώς μια αποτυχία μπορεί να συμβεί μια τυχαία χρονική στιγμή όπως είναι γνωστό. Οπότε πρέπει να λάβουμε υπόψη επιπλέον παραμέτρους που μελετούν αυτήν την πιθανότητα σφάλματος και να συνδυαστεί μαζί με την αρχή τυχαιότητας των συμβάντων. Πράγμα πάρα πολύ περίπλοκο.

Η π*υκνότητα αποτυχίας f(t)* ορίζεται ως η πιθανότητα που το σύστημα παρουσιάζει την *πρώτη* αποτυχία του μια χρονική στιγμή t, με δεδομένο ότι το σύστημα λειτουργούσε κανονικά από την χρονική στιγμή μηδέν.

Ο ρ*υθμός αποτυχίας r(t)* ορίζεται ως η πιθανότητα ανά μονάδα χρόνου, που το εξάρτημα ή σύστημα βιώνει μια αποτυχία στο χρόνο t, με δεδομένο ότι λειτουργούσε κατά την χρονική στιγμή μηδεν και έχει συντηρηθεί μέχρι τη χρονική στιγμή t.

Το *ποσοστό αποτυχίας λ(t)* του συστήματος, ορίζεται ως η πιθανότητα ανά μονάδα χρόνου όταν το σύστημα βιώνει μια αποτυχία στο χρόνο t, δεδομένου ότι λειτουργούσε ή επισκευάστηκε να είναι τόσο καλό όσο ένα καινούργιο, την χρονική στιγμή μηδέν και λειτουργεί κατά τον χρόνο t.

Η *συχνότητα αποτυχίας ω(t)* ορίζεται ως η πιθανότητα ανά μονάδα χρόνου που το σύστημα βιώνει μια αποτυχία στο χρόνο t, δεδομένου ότι το εξάρτημα ή σύστημα λειτουργεί στο χρόνο μηδέν. Η παράμετρος αυτή είναι χωρίς συνθήκη.

Η διαφορά ανάμεσα στο ρυθμό αποτυχίας ενός συστήματος και του ποσοστού του βρίσκεται στο γεγονός ότι ο r(t) έχει να κάνει με το συμβάν της πρώτης αποτυχίας και όχι οποιασδήποτε αποτυχίας του εξαρτήματος ή συστήματος. Στις ειδικές περιπτώσεις που ο ρυθμός σφαλμάτων παραμένει σταθερός με την πάροδο του χρόνου ή το εξάρτημα είναι μη επιδιορθώσιμο, οι δύο αυτοί παράμετροι είναι ίδιοι. Οπότε ισχύει,

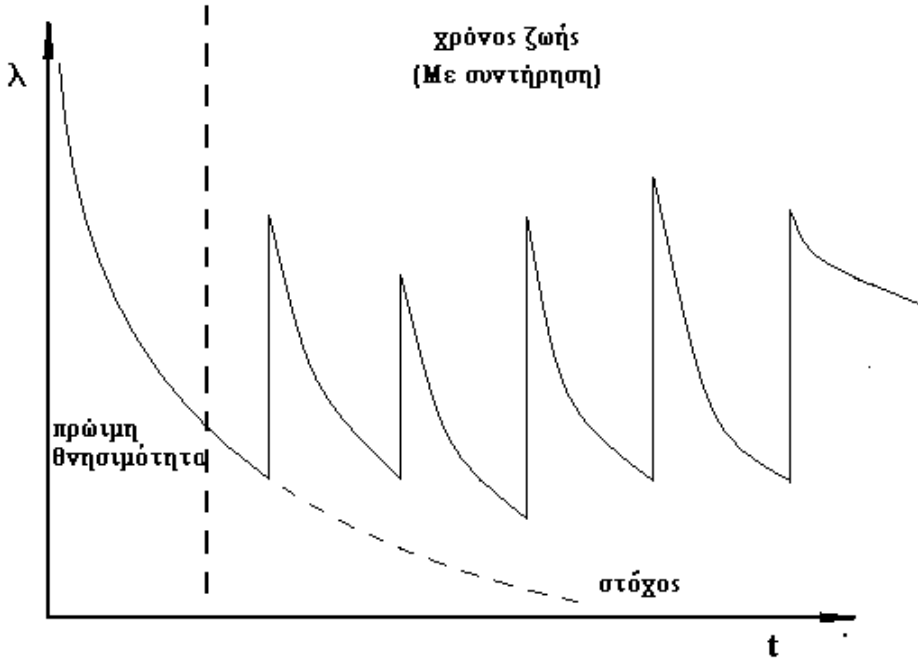
$$r(t) = \lambda(t) \quad : \mu\eta \ \ \varepsilon\pi\iota\delta\iota o\rho\theta\acute{\omega}\sigma\iota\mu\alpha \ \ \sigma\upsilon\sigma\tau\acute{\eta}\mu\alpha\tau\alpha \ \ \acute{\eta} \ \ \sigma\tau\alpha\theta\varepsilon\rho\acute{\eta} \ \ \alpha\pi o\tau\upsilon\chi\acute{\iota}\alpha$$
$$r(t) \neq \lambda(t) \quad : \gamma\varepsilon\nu\iota\kappa\acute{\eta} \ \ \pi\varepsilon\rho\acute{\iota}\pi\tau\omega\sigma\eta$$

Οι βαθμοί αποτυχίας βασίζονται σε πολύπλοκα μοντέλα που περιλαμβάνουν συγκεκριμένες προδιαγραφές των εξαρτημάτων όπως η θερμοκρασία λειτουργίας, το περιβάλλον και την φυσιολογική παρακμή των ολοκληρωμένων. Στα μοντέλα

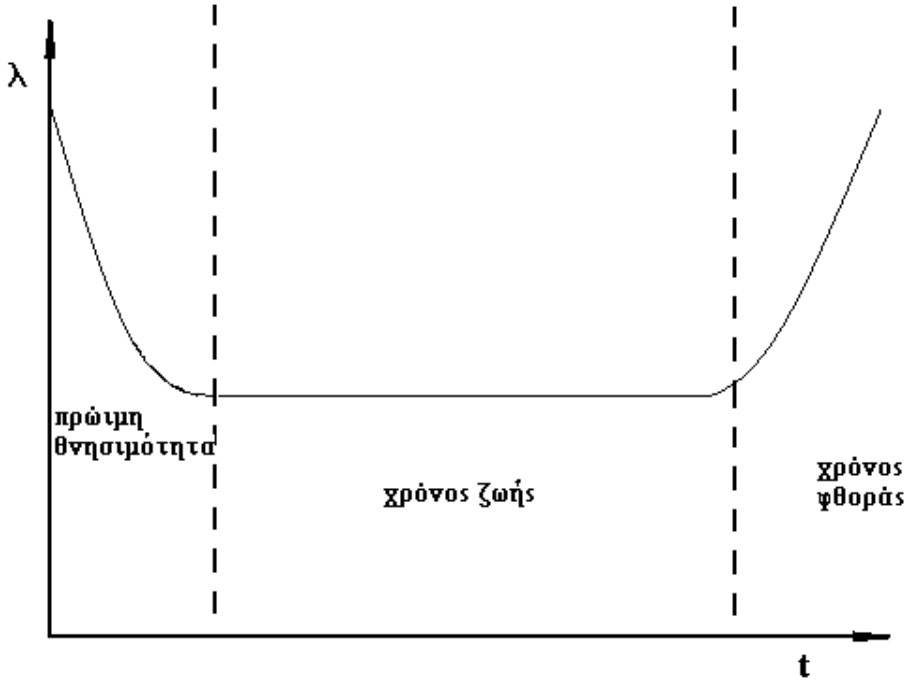
πρόβλεψης τα εξαρτήματα συνδέονται μεταξύ τους σειριακά. Οπότε ο βαθμός αποτυχίας αυτών θα είναι η πρόσθεση του ρυθμού αποτυχίας κάθε εξαρτήματος, αντιπροσωπεύοντας το σύστημα. Μερικά χειροπιαστά παραδείγματα μοντέλων με σκοπό την βελτίωση της αξιοπιστίας ενός υπολογιστικού συστήματος αποτελούν τα [4]:

- Μοντέλα μείωσης θερμικών, ηλεκτρικών και μηχανικών κινδύνων
- Μοντέλα διασύνδεσης και αλληλεπίδρασης.
- Μοντέλα απλοποίησης σχεδιασμού και ανάπτυξης.
- Η χρήση ποιοτικά καλύτερων εξαρτημάτων στο σχέδιο υλικού.
- Μοντέλα προστασίας από ηλεκτρομαγνητισμό, EMC και ESD
- Μοντέλα προστασίας κρίσιμων εξαρτημάτων
- Η χρήση πλεονασμού

Κατά την ανάλυση των συστημάτων πρέπει να γίνουν υποθέσεις σχετικά με την επίδραση των σφαλμάτων, τον χρόνο που θα συμβούν και για την περίοδο λειτουργίας τους. Ένα σύστημα υλικού δεν έχει την ίδια βιωσιμότητα με ένα σύστημα λογισμικού και αυτό οφείλεται στην διαφορετική τους φύση. Τα παρακάτω διαγράμματα περιγράφουν την βιωσιμότητα τους από την πρώτη στιγμή λειτουργίας τους, την περίοδο κανονικής λειτουργίας και το τέλος τους. Η διαφορά αυτή είναι ορατή στα παρακάτω διαγράμματα. Το πρώτο αφορά τα συστήματα λογισμικού και το δεύτερο τα συστήματα υλικού. Όμως υπάρχουν και άλλες διαφορές στο παράρτημα [Α.2.2].



Εικόνα 2.9: Διάγραμμα αξιοπιστίας συστήματος λογισμικού χρησιμοποιώντας συντήρηση.



Εικόνα 2.10: Διάγραμμα αξιοπιστίας συστήματος υλικού .

Όπως είναι φανερό τα διαγράμματα αυτά περιγράφονται από συναρτήσεις, οπότε η συμπεριφορά ενός εξαρτήματος μπορεί εύκολα να μοντελοποιηθεί με δεδομένο ότι θα γίνει σωστή υπόθεση και μελέτη αυτής κατά την ανάπτυξή του. Η πρόβλεψη αυτή σχετίζεται με το ποσοστό αποτυχίας και αξιοπιστίας ενός συστήματος καθώς χρησιμοποιούνται θεμελιώδη θεωρήματα της στατιστικής και των πιθανοτήτων για να καθορίσουν χειροπιαστά αποτελέσματα. Με τον τρόπο αυτό οι βασικές συναρτήσεις πυκνότητας , από την στατιστική, μπορούν να διαμορφώνουν συγκεκριμένα μοντέλα για την συμπεριφορά ενός εξαρτήματος και οi αθροιστικές κατανομές αυτών να περιγράψουν την αξιοπιστία του με τη μορφή διαγράμματος. Μερικές από αυτές θα αναφερθούν παρακάτω για κατανόηση του ρόλου τους στην ανάπτυξη και λειτουργία των συστημάτων ανοχής σφαλμάτων. Αναλυτική περιγραφή υπάρχει στο παράρτημα [Α.2.3].

Η διωνυμική συνάρτηση που είναι γνωστή από τον μαθηματικό Bernoulli, χρησιμοποιείται ευρέως επειδή είναι κατανομή διακριτής και τυχαίας μεταβλητής. Αξιοποιείται στην διασφάλιση της αξιοπιστίας και στον ποιοτικό έλεγχο. Μια εφαρμογή του είναι για τον καθορισμό της επιτυχίας ή αποτυχίας μιας δοκιμής ή αποστολής. Επιπλέον μας δίνει την δυνατότητα να υπολογίσουμε την αξιοπιστία σε συστήματα τύπου Μ-από-Ν, που είναι απαραίτητη η λειτουργία συγκεκριμένου πλήθους από εξαρτήματα κάθε φορά.

Η κατανομή Poisson χρησιμοποιείται για την πρόβλεψη γεγονότων που θα συμβούν χωρίς να γνωρίζουμε το πλήθος τους. Είναι σημαντική καθώς μας δίνει την δυνατότητα να προβλέψουμε ακριβώς τις πιθανές αποτυχίες ενός εξαρτήματος ή συστήματος, σε δεδομένο χρόνο. Μια εφαρμογή της κατανομής αποτελεί ο υπολογισμός k ή λιγότερων αποτυχιών και συμβάλλει στον καθορισμό του πλήθους των εφεδρικών μονάδων σε ένα σύστημα, που χρησιμοποιεί τεχνικές πλεονασμού αναμονής.

Η εκθετική συνάρτηση αποτελεί μια από τις βασικότερες συναρτήσεις στη μηχανική αξιοπιστίας επειδή έχει σταθερό ποσοστό αποτυχίας λ(t). Αυτή προσομοιώνει την ζωή ηλεκτρονικών και ηλεκτρολογικών εξαρτημάτων και είναι χρήσιμη για την εξής υπόθεση: ότι ένα χρησιμοποιημένο εξάρτημα είναι εξίσου κάλο και έτοιμο να αποδώσει σαν ένα καινούργιο. Η υπόθεση αυτή αιτιολογείται επειδή είναι η μόνη συνεχής κατανομή που μπορεί να περιγράψει πιστά τον χρόνο ζωής ενός εξαρτήματος υλικού, όπως φαίνεται στο διάγραμμα ζωής υλικού προηγουμένως.

Ο Weibull όρισε την ομώνυμη κατανομή του, η οποία αποτελεί από τις σημαντικότερες για την αξιοπιστία, επειδή έχει την δυνατότητα να περιγράφει την συμπεριφορά ενός συστήματος υπό την επίδραση διακυμάνσεων, που οφείλονται στα ποσοστά αποτυχίας λ(t). Ρυθμιστικοί παράμετροι αποτελούν τα θ, β, γ τα οποία είναι οι συντελεστές κλίμακας, σχήματος και τοποθέτησης. Είναι πάντα θετικές ποσότητες και με κατάλληλη ρύθμιση των παραμέτρων μπορούν να ακολουθήσουν άλλες κατανομές όπως η εκθετική και η κανονική. Είναι κατάλληλη για την περιγραφή των σταδίων της πρώιμης θνησιμότητας και φθοράς των εξαρτημάτων.

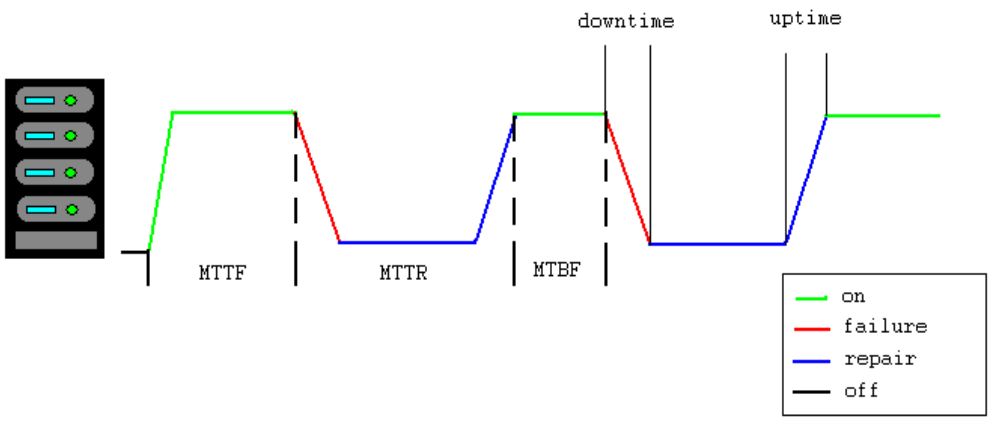
Η συνάρτηση που χρησιμοποιείται για την περιγραφή καταστάσεων υποβάθμισης της λειτουργίας είναι αυτή του Rayleigh. Παρέχει ευελιξία στην περιγραφή της περιόδου λειτουργίας ενός εξαρτήματος, λόγω των χαρακτηριστικών της ως κατανομή.

Η κατανομή γάμμα μπορεί να χρησιμοποιηθεί ως συνάρτηση πιθανότητας για την μελέτη της αποτυχίας εξαρτημάτων με παραμορφωμένο τρόπο λειτουργίας. Σε αυτή οι συντελεστές α και β προσδιορίζουν το σχήμα και την κλίμακα αντίστοιχα. Επιπλέον αν αξιοποιηθεί συνδυαστικά με την εκθετική κατανομή μπορεί να υπολογίσει τον χρόνο της n- στής αποτυχίας ενός συστήματος.

Για την μελέτη της ευαισθησίας ενός συστήματος από εξωτερικές πιέσεις σε φυσικό επίπεδο αξιοποιείται η κανονική συνάρτηση. Αυτή η κατανομή αναλύει το στάδιο της φυσικής παρακμής πολλών μηχανολογικών συστημάτων. Οι δύο παράμετροι μ και σ είναι η μέση τιμή και η τυπική απόκλιση της κατανομής. Ακόμη προϊόν αυτής είναι η κανονική λογαριθμική κατανομή, που είναι η καταλληλότερη για την περιγραφή ηλεκτρονικών εξαρτημάτων όπως είναι οι ημιαγωγοί.

Επομένως ανάλογα με την εφαρμογή και τα εξαρτήματα που θα χρησιμοποιήσει ο σχεδιαστής να κατασκευάσει ένα σύστημα, θα πρέπει να γίνει σωστή υπόθεση για τον τρόπο λειτουργίας του κάθε εξαρτήματος, χρησιμοποιώντας παντα τις κατανομές ως κριτήριο αναφοράς και παραμέτρους που περιγράφουν την επίδραση των αποτυχιών σε ένα σύστημα (βλ. Παράρτημα Α.2.3). Υπάρχουν τρείς παράμετροι των βαθμών αποτυχίας για ένα σύστημα και αυτά είναι[1,2,3]:

- Ο μέσος χρόνος ανάμεσα σε αποτυχίες – **Mean Time Between Failures (MTBF)**
- Ο μέσος χρόνος μέχρι μια αποτυχία – **Mean Time To Failure (MTTF)**
- Ο μέσος χρόνος μέχρι την επισκευή /επιδιόρθωση – **Mean Time To Repair (MTTR)**



Εικόνα 2.11: Οι έννοιες MTTF MTTF MTTR, downtime και uptime σε ένα σύστημα.

Ο MTBF χρησιμοποιείται για την μέτρηση της αξιοπιστίας συντηρήσιμων εξαρτημάτων. Περιγράφεται ως η πάροδος του χρόνου για ένα εξαρτήμα ή συστήμα μέχρι μία αποτυχία να συμβεί, με την προϋπόθεση ότι λ(t) σταθερό. Ο ορισμός αυτός μπορεί να επεκταθεί ως η πάροδος χρόνου μεταξύ δύο διαδοχικών αποτυχιών ενός συστήματος. Ο όρος αυτός σχεδιάστηκε για συντηρήσιμα συστήματα αν και χρησιμοποιείται και σε μη συντηρήσιμα. Σε αυτά είναι ο χρόνος μέχρι να συμβεί η πρώτη και μοναδική αποτυχία. Ορίζεται ως:

$$MTBF = 1/\lambda \quad ή \quad 1/(2\ αποτυχίες\ x10^6) = 500.000\ ώρες/αποτυχία$$

Ο MTTF είναι το μέτρο της αξιοπιστίας για τα μη συντηρήσιμα συστήματα. Είναι ο ενδιάμεσος χρόνος μέχρι να συμβεί η πρώτη αποτυχία ενός εξαρτήματος ή συστήματος. Χρησιμοποιείται για μεγάλη διάρκεια χρόνου και σε συστήματα με μεγάλο πλήθος εξαρτημάτων. Για συστήματα με σταθερό λ(t) ο MTTF είναι το αντίστροφο του βαθμού αποτυχίας. Αν όμως το λ(t) είναι εκφρασμένο σε αποτυχίες ανά ώρα τότε το MTTF= 1.000.000/ λ(t) είναι για εξαρτήματα με εκθετική κατανομή.

$$MTBF = 10^6/\lambda$$

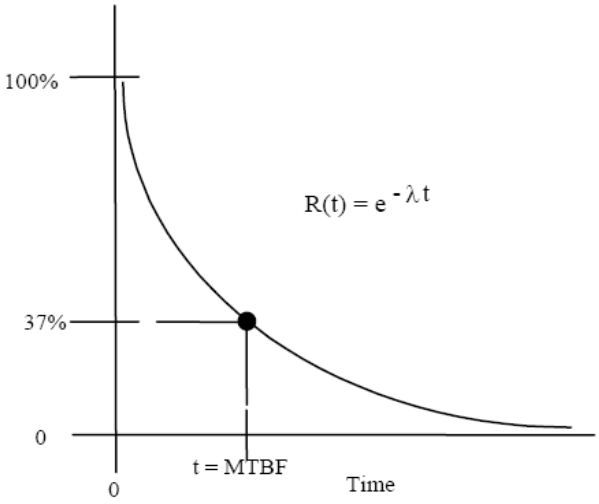
Ο MTTR περιγράφεται ως ο χρόνος που αναλώνεται για την ορθή επισκευή ή την επισκευή παρεμπόδισης ανά αριθμό επισκευών που έγιναν στο σύστημα. Ουσιαστικά είναι ο χρόνος από την αποτυχία ή τον ασφαλή τερματισμό μέχρι την ολοκλήρωση της συντήρησης. Αποτελείται από τον συντελεστή επιδιόρθωσης μ(t) που μετριέται σε ώρες ανά επισκευή και ορίζεται ως εξής:

$$MTTR = 1/\mu$$

2.6.1 Εξαρτησιμότητα

Η *εξαρτησιμότητα* R(t) του συστήματος τη χρονική στιγμή t είναι η πιθανότητα που λειτουργεί το σύστημα χωρίς βλάβη στο διάστημα [0, t], δεδομένου ότι το σύστημα εκτελεί σωστά σε χρόνο μηδέν. Η συνάρτηση f(t) είναι η συνάρτηση αποτυχίας ενός εξαρτήματος ή συστήματος. Αυτή επιλέγεται από τις προηγούμενες στατιστικές κατανομές ανάλογα με το εξάρτημα που μελετάμε. Είναι το μέτρο της συνεχούς παροχής ορθής υπηρεσίας. Μαθηματικά αυτό εκφράζεται ως εξής:

R(t)=P(T > t) <=> $R(t) = \int_t^{\infty} f(t)dt \quad => \quad R(t) = e^{-\lambda t}$ για t >0



Εικόνα 2.12: Διάγραμμα εξαρτησιμότητας-χρόνου

Εναλλακτικά, μπορούμε να ορίσουμε αναξιοπιστία F(t) του συστήματος σε χρόνο t, τη πιθανότητα ότι το σύστημα αποτυγχάνει στο διάστημα [0, t] για μια η περισσότερες φορές, δεδομένου ότι λειτουργούσε σωστά την χρονική στιγμή 0. Η *Αναξιοπιστία* εκφράζει την πιθανότητα της αποτυχίας του συστήματος και εκφράζεται με τον παρακάτω τύπο.

$$F(t) = 1 - R(t)$$

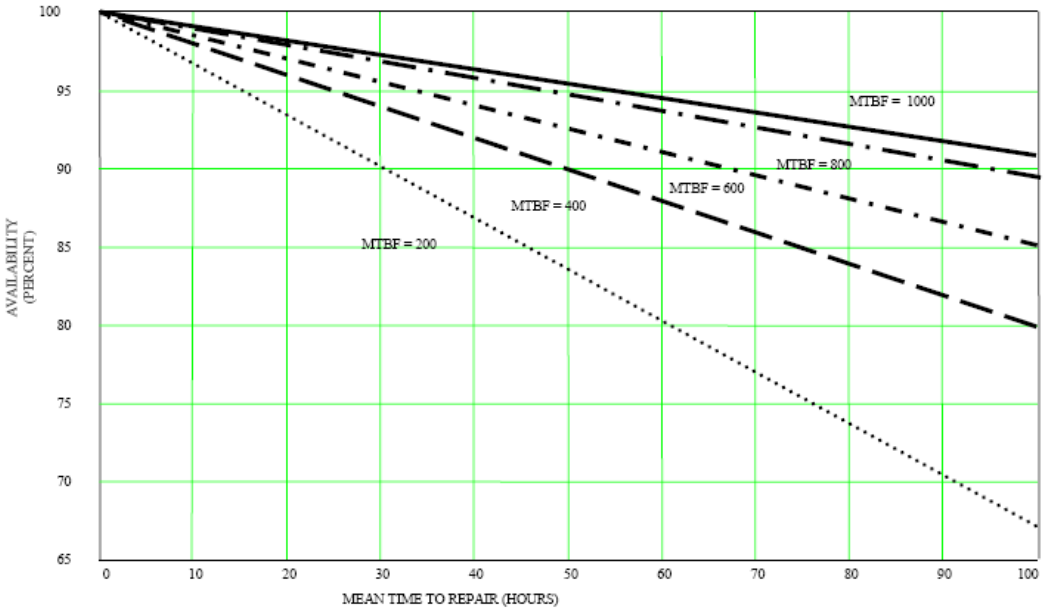
2.6.2 Διαθεσιμότητα

Η διαθεσιμότητα ορίζεται ως η περίοδος λειτουργίας του συστήματος από την χρονική στιγμή [0, t] με δεδομένο ότι λειτουργούσε ήδη στο 0. Είναι το μέτρο προσφοράς της υπηρεσίας σε ένα επιδιορθώσιμο σύστημα. Είναι ο χρόνος λειτουργίας του συστήματος, ανά τον χρόνο ώσπου συμβαίνει η αποτυχία , εκτελείται η επιδιόρθωση και επανέρχεται στην κατάσταση ορθής λειτουργίας.

$$A(t) = MTBF / (MTBF + MTTR)$$

ή

$$A(t) = \frac{Uptime}{Downtime + Uptime}$$



Εικόνα 2.13: Διάγραμμα Διαθεσιμότητας-χρόνου[3]

Η μη διαθεσιμότητα ενός συστήματος είναι η πιθανότητα του συστήματος ότι από την χρονική στιγμή t δεν λειτουργεί αν και δούλευε την χρονική στιγμή 0 κανονικά. Συμβολίζεται με Q(t) και η εξίσωση είναι :

$$Q(t) = 1 - A(t)$$

Και οι τέσσερις όροι που έχουμε αναφέρει μέχρι στιγμής χρησιμοποιούνται για την μελέτη του κόστους και της ασφάλειας ενός συστήματος. Για συστήματα τα οποία έχουν δυνατότητα συντήρησης ισχύει,

$$Q(t) \leq F(t)$$

ενώ για μη συντηρήσιμα εξαρτήματα ή συστήματα ισχύει,

$$Q(t) = F(t)$$

μόνο όταν όλα τα εξαρτήματα δεν έχουν επιδιορθωθεί ως την χρονική στιγμή μιας αποτυχίας.

2.6.3 Συντηρησιμότητα

Είναι η ιδιότητα ενός συστήματος να επανέρχεται σε συνθήκες ορθής λειτουργίας μετά από μια αποτυχία, σε πεπερασμένο χρόνο το οποίο καθορίζεται από τις τεχνικές και το είδος της συντήρησης που θα πραγματοποιηθεί. Η διαδικασία εμπεριέχει την αντικατάσταση, αφαίρεση, αποσυναρμολόγηση και επανασυναρμολόγησή του συστήματος. Η διαδικασία αυτή για ένα σύστημα περιγράφεται με τον παρακάτω τύπο, όπου μ είναι ο συντελεστής επιδιόρθωσης:

$$M(t) = 1 - e^{-\mu t}$$

2.6.4 Ασφάλεια

Ασφάλεια S (t) του συστήματος σε χρόνο t είναι η πιθανότητα ότι το σύστημα είτε εκτελεί λειτουργία του σωστά ή διακόπτει τη λειτουργία του κατά τρόπο ασφαλή από αστοχία στο διάστημα [0, t], δεδομένου ότι το σύστημα λειτουργεί σωστά σε χρόνο μηδέν. Η ασφάλεια γενικά περιγράφεται με:

Margin Of Safety= (Failure Load / Design Load x Design Safety Factor) – 1

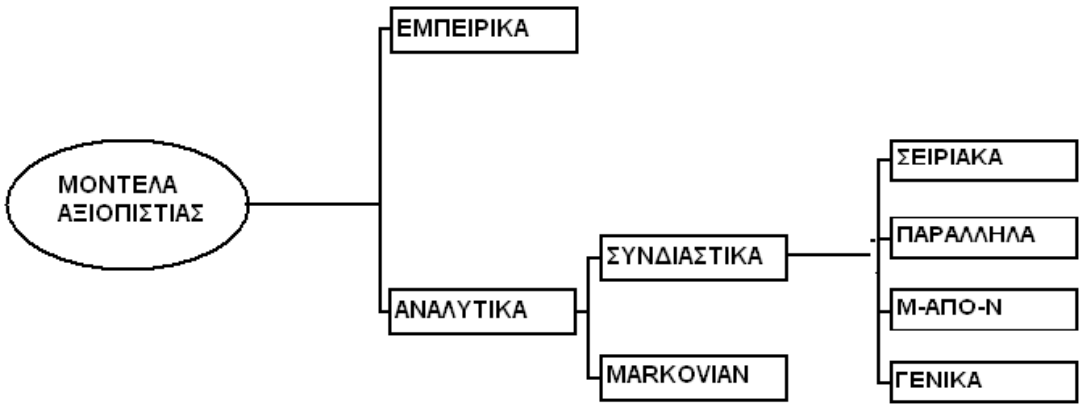
Η παραπάνω σχέση δίνει το περιθώριο ασφάλειας σχετικά με το ποσοστό αποτυχίας, το ποσοστό της σχεδίασης και τις προδιαγραφές ασφάλειας προς σχεδιασμό.

Έχοντας αναφέρει όλα τα παραπάνω είμαστε σε θέση να μελετήσουμε τους δύο βασικότερους τρόπους μοντελοποίησης ενός αξιόπιστου συστήματος. Χρησιμοποιούνται δύο τεχνικές για τον σκοπό αυτό, οι *εμπειρικές* και οι *αναλυτικές*.

2.7 Το εμπειρικό μοντέλο

Σύμφωνα με αυτό μια σειρά από διαδοχικά Ν συστήματα δοκιμάζονται σε λειτουργία για μια μεγάλη περίοδο χρόνου. Ο αριθμός των συστημάτων που θα αποτύχουν στην χρονική διάρκεια αυτή καταγράφεται. Μετά από αυτόν τον πειραματικό τρόπο υπολογίζεται το ποσοστό της αξιοπιστίας. Ο λόγος αυτός εκφράζεται με το πλήθος των συστημάτων που απέτυχαν, προς τον αριθμό των επιτυχόντων. Για παράδειγμα αν κάποιο σύστημα λειτούργησε για 100 φορές και από αυτές οι αποτυχίες ήταν 10, τότε είναι εύκολο να υπολογιστεί με σχετική ακρίβεια ότι το ποσοστό είναι 90%.

Όμως η μέθοδος αυτή έχει δύο αρνητικές πτυχές[2,3]. Η μια αφορά το πλήθος των συστημάτων, το οποίο πρέπει να είναι πολύ μεγάλο για να επιτύχουμε υψηλή αξιοπιστία. Έτσι αυξάνεται κατακόρυφα το κόστος. Το δεύτερο αφορά την χρονική διάρκεια του πειράματος και μπορεί να είναι πολύ μεγάλη. Για παράδειγμα ένα σύστημα το οποίο αποτελείται από 1000 επεξεργαστικά στοιχεία για ένα διαστημικό λεωφορείο θα πάρει πολύ χρόνο και θα πρέπει να διατεθούν τεράστια ποσά για να αναπτυχθεί πειραματικά



Εικόνα 2.14: Κατανομή των μοντέλων αξιοπιστίας.

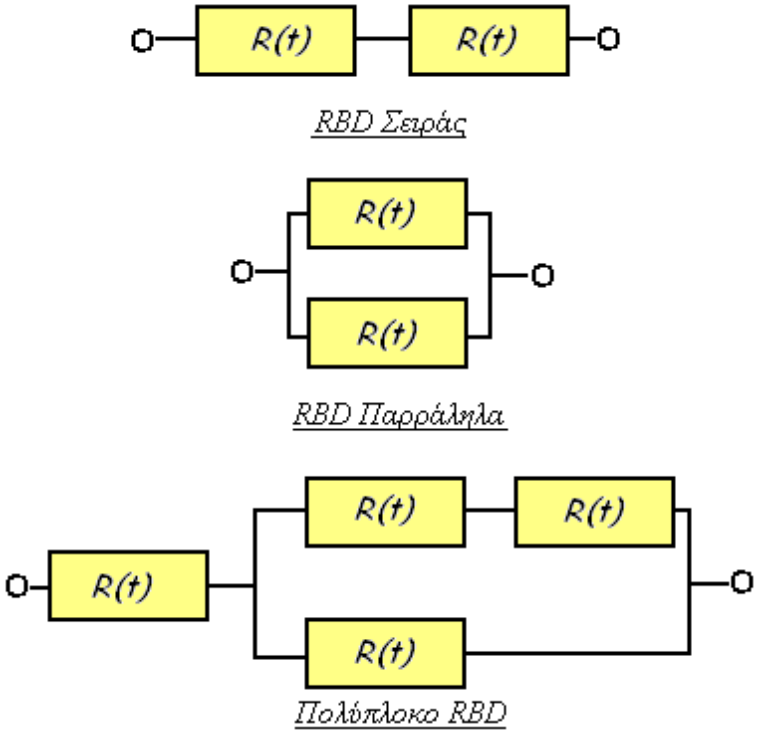
2.8 Το Αναλυτικό Μοντέλο

Η τεχνική αυτή χρησιμοποιεί τις πιθανότητες κάθε ξεχωριστού μοντέλου ώστε να μετρηθεί το συνολικό ποσοστό αποτυχίας ενός συστήματος. Πρέπει να σημειωθεί ότι το ποσοστό αυτό εξαρτάται τόσο από την αποτυχία κάθε εξαρτήματος ξεχωριστά

αλλά και από τον τρόπο που συνδέονται αυτά μεταξύ τους ώστε ενοποιώντας τα να σχηματίσουν το επιθυμητό επίπεδο αξιοπιστίας στο σύστημα. Υπάρχουν δύο είδη περιγραφής του αναλυτικού μοντέλου, το *συνδυαστικό*, το οποίο ονομάζεται συνεχές και το *Markovian*, που ονομάζεται διακριτό.

2.8.1 Συνδυαστικό μοντέλο

Στο συνδυαστικό μοντέλο οι τρόποι που ένα σύστημα συνεχίζει αδιάλειπτα την λειτουργία του, λαμβάνοντας υπόψη την πιθανότητα αποτυχίας των ξεχωριστών μονάδων από το οποίο αποτελείται, είναι πεπερασμένοι. Υπάρχει λοιπόν συγκεκριμένος αριθμός από τρόπους με τους οποίους μπορούν να συνδεθούν μεταξύ τους τα εξαρτήματα για την επίτευξη μιας λειτουργίας. Οπότε τα επίπεδα αξιοπιστίας είναι εύκολα υπολογίσιμα καθώς αφορά την διάταξη των εξαρτημάτων μέσα σε ένα σύστημα. Στον πίνακα 2.16 συνοψίζονται οι πιθανοί τρόποι καθώς και οι απαραίτητες εξισώσεις που τους περιγράφουν.



Εικόνα 2.15: Διαγράμματα RBD.

Συμφώνα με το συνδυαστικό μοντέλο, για να περιγράψουμε την αξιοπιστία του συστήματος δημιουργούμε τα μπλοκ διάγραμματα Αξιοπιστίας (RBD – Reliability Block Diagramme). Με αυτό δημιουργούμε την σύνδεση των εξαρτημάτων από την σκοπιά της αξιοπιστίας, η οποία είναι συμπληρωματική με την σκοπιά της απόδοσης. Το μοντέλο αυτό είναι εξαιρετικά χρήσιμο στην μοντελοποίηση πολύπλοκων συστημάτων σε απλούστερα μέρη. Σύμφωνα με το θεώρημα του Bayes, έστω ένα σύστημα ότι αποτελείται από δύο η περισσότερα εξαρτήματα, επιλέγουμε ένα από αυτά και το ονομάζουμε Α. Με βάση αυτό, δύο γεγονότα μπορούν να συμβούν είτε το σύστημα να λειτουργεί με το Α κανονικά είτε να λειτουργεί χωρίς αυτό. Οι περιπτώσεις αυτές εκφράζονται με την παράσταση:

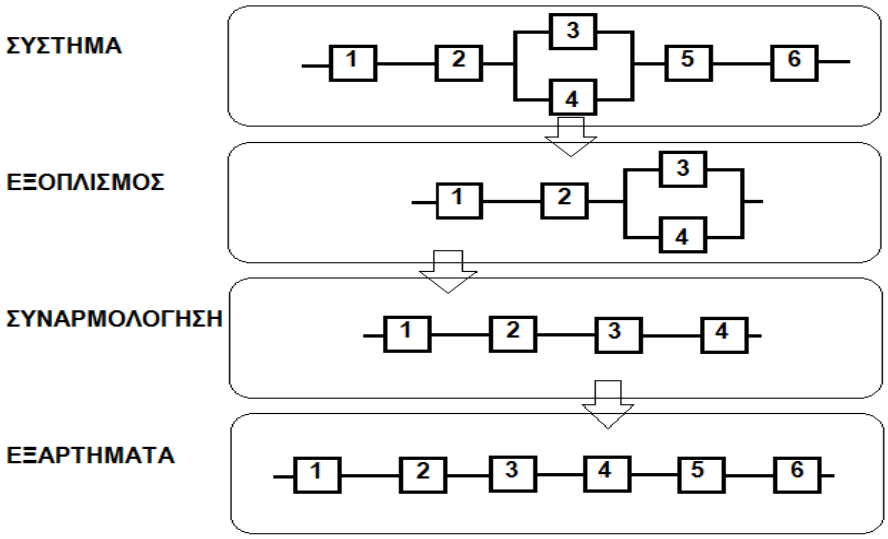
$$Psystem = (Psystem \mid A)PA + (Psystem \mid \bar{A})(1-PA)$$

όπου PA είναι η πιθανότητα λειτουργίας του A και η πιθανότητα (1-PA) ότι το A δεν λειτουργεί. Το κατά προσέγγιση μοντέλο προσφέρει το πάνω όριο της αξιοπιστίας ενός συστήματος με βάσει τα εξαρτήματα από τα οποία αποτελείται. Το μοντέλο αυτό αποτελείται από παράλληλα μονοπάτια από την είσοδο ως την έξοδο για τον υπολογισμό της αξιοπιστίας του. Ωστόσο παρότι είναι απλό στην χρήση το συνδυαστικό μοντέλο έχει και τα αρνητικά του. Οριοθετεί την αξιοπιστία με πολύπλοκες εκφράσεις που σημαίνει δυσκολότερη ανάλυση και υλοποίηση. Τέλος, δεν επιτρέπει την μοντελοποίηση της Επιδιόρθωσης και της Διαθεσιμότητας ενός συστήματος. Στον παρακάτω πίνακα συνοψίζονται οι πιθανοί τρόποι καθώς και οι απαραίτητες εξισώσεις διασύνδεσης.

ΤΕΧΝΙΚΕΣ	ΕΞΙΣΩΣΕΙΣ ΑΞΙΟΠΙΣΤΙΑΣ
ΣΕΙΡΑΣ	$R_{series}(t) = \prod_{i=1}^{N} R_i$
ΠΑΡΑΛΛΗΛΑ	$R_{parallel}(t) = 1 - \prod_{i=1}^{N}(1-R_i)$
ΣΕΙΡΑΣ / ΠΑΡΑΛΛΗΛΑ	$R_{sp}(t) = \prod_{i=1}^{N} R_{parallel_i}$, $R_{parallel_i}$, i -στης ομάδας
ΠΑΡΑΛΛΗΛΑ / ΣΕΙΡΑΣ	$R_{ps}(t) = 1 - \prod_{i=1}^{N}(1-R_{series_i})$ R_{series_i} , i -στης ομάδας
M - ΑΠΟ - N	$R_{MoN} = \sum_{i=0}^{N-M} \binom{N}{i} (1-R)^i * R^{N-i}$
ΚΑΤΑ ΠΡΟΣΕΓΓΙΣΗ	$R_{system} \leq 1 - \prod_{i=1}^{n}(1-R_{path})$, n μονοπάτια από την είσοδο στην έξοδο
ΕΠΑΚΡΙΒΩΣ	$P_{system} = (P_{system}\mid A)P_A + (P_{system}\mid \bar{A})(1-P_A)$

Εικόνα 2.16: Πίνακας εξισώσεων αξιοπιστίας για κάθε συνδεσμολογία.

Έτσι λοιπόν μπορούμε να αναλύσουμε ένα σύστημα στα συστατικά του μέρη και επιπλέον να κατανοήσουμε την δομή του κάθε εξαρτήματος που θα συνεισφέρει ένα μερίδιο στην αξιοπιστία του. Η ανάλυση αυτή είναι εύκολα παρατηρήσιμη από το παρακάτω σχήμα. Η μέθοδος αυτή στοχεύει στην τοπολογική διασύνδεση των εξαρτημάτων ενώ η επόμενη μελετά τις πιθανές καταστάσεις που μπορεί να συμβούν σε ένα σύστημα.



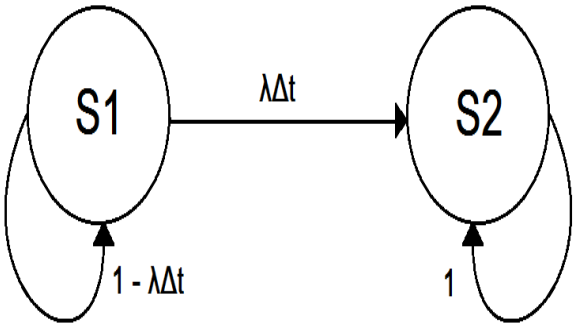
Εικόνα 2.17: Διάγραμμα RBD ενός συστήματος και των συστατικών του.

Το παραπάνω διάγραμμα απεικονίζει την ανάλυση του εξαρτήματος 4 στα επιμέρους κομμάτια του. Έπειτα της μονάδας 1 κ.ο.κ.

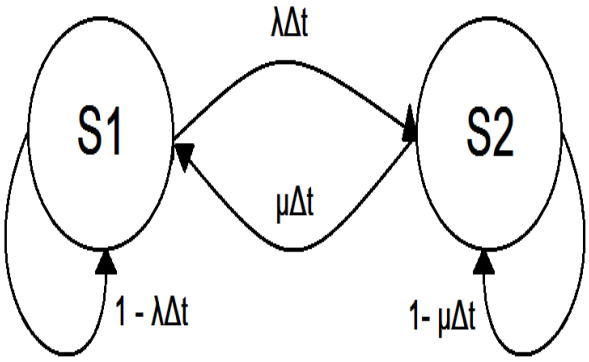
2.8.2 Markovian Model

Το μοντέλο αυτό [2,4] είναι διακριτό και όχι αναλογικό. Χρησιμοποιεί αλυσίδες Markov για την απεικόνιση των καταστάσεων ενός συστήματος. Οι μεταβολές από τις διάφορες καταστάσεις εκφράζονται στον λεγόμενο *χώρο καταστάσεων.* Είναι τυχαίες μεταβολές και εξαρτώνται από την παρούσα κατάσταση και μόνο, δεν υπάρχει μνήμη των προηγούμενων καταστάσεων που έχει εισέλθει το σύστημα. Ο τρόπος αυτός μοντελοποίησης είναι βολικός για την προσομοίωση διεργασιών πραγματικού χρόνου και επομένως των συστημάτων ανοχής σφαλμάτων.

Οι καταστάσεις μεταβάλλονται σε διακριτά τμήματα χρόνου Δt και αποτελούνται από συνθήκες ορθής λειτουργίας και εσφαλμένης λειτουργίας. Η μετάβαση από μια συνθήκη στην επόμενη γίνεται ακολουθιακά με την πάροδο του χρόνου. Έστω λοιπόν το διάγραμμα καταστάσεων ενός εξαρτήματος βλ. 2.18. Υπάρχουν δύο καταστάσεις, η ορθή και η εσφαλμένη. Η κατάσταση S1 αντιπροσωπεύει την κατάσταση ορθής λειτουργίας, ενώ η S2 την εσφαλμένη. Για να μεταβεί από την S1 στην S2 εισάγουμε την παράμετρο λ που είναι ο *ρυθμός αποτυχίας*. Η ευελιξία ενός τέτοιου μοντέλου εμφανίζεται όταν θέλουμε να εισάγουμε και τις καταστάσεις επιδιόρθωσης του συστήματος. Ο *ρυθμός επιδιόρθωσης* συμβολίζεται με την παράμετρο μ. Έτσι λοιπόν στο ίδιο μοντέλο μπορούμε να απεικονίσουμε ολοκληρωμένα τις καταστάσεις ορθής, εσφαλμένης και επιδιόρθωσης βλ. 2.19.



Εικόνα 2.18: Καταστάσεις ενός εξαρτήματος χωρίς επιδιόρθωση



Εικόνα 2.19: Καταστάσεις ενός εξαρτήματος με επιδιόρθωση

Από τα παραπάνω σχήματα μπορούμε να γράψουμε τις εξισώσεις πιθανότητας σχετίζοντας την κατάσταση του συστήματος μια χρονική στιγμή *t* και έπειτα από πάροδο χρόνου *t+Δt*.

$$P_1(t+\Delta t) = (1-\lambda\Delta t)\cdot P_1(t) + \mu\Delta t \cdot P_2(t)$$
$$P_2(t+\Delta t) = \lambda\Delta t \cdot P_1(t) + (1-\mu\Delta t)\cdot P_2(t)$$

Οι παραπάνω εκφράσεις αναπαριστώνται με την μορφή πίνακα ως εξής:

$$\begin{bmatrix} P_1 & (t+\Delta t) \\ P_2 & (t+\Delta t) \end{bmatrix} = \begin{bmatrix} 1-\lambda\Delta t & \mu\Delta t \\ \lambda\Delta t & 1-\mu\Delta t \end{bmatrix} \cdot \begin{bmatrix} P_1 & (t) \\ P_2 & (t) \end{bmatrix}$$

ή απλούστερα με την μορφή $P(t+\Delta t) = A \cdot P(t)$, όπου ο πίνακας Α αποτελεί τον *πίνακα μετάβασης του συστήματος*. Αυτός είναι χρήσιμος για τον υπολογισμό της κατάστασης ενός συστήματος κάθε χρονική στιγμή. Έτσι λοιπόν καταλήγουμε στον γενικό τύπο υπολογισμού $P(n\Delta t) = A^n \cdot P(0)$.

Η διαθεσιμότητα και η ασφάλεια του συστήματος υπολογίζεται από την πιθανότητα της κατάστασης S1=P1(t) για την περίπτωση χωρίς επιδιόρθωση. Η περίπτωση όπου κυριαρχεί ανασφάλεια είναι η S2=P2(t). Τέλος για την περίπτωση με επιδιόρθωση, η ασφάλεια και η διαθεσιμότητα είναι η πιθανότητα του συστήματος να βρίσκεται στην S1=P1(t)+P2(t), ενώ είναι ανασφαλές για την S2=P2(t).

Πολλές φορές για την επίλυση πολύπλοκων συστημάτων γίνεται χρήση του μετασχηματισμού Laplace για απλοποίηση διαφορικών εξισώσεων σε αλγεβρικές. Οι εξισώσεις αξιοπιστίας και διαθεσιμότητας με τον τρόπο αυτό μελετώνται ευκολότερα.

Και το σύστημα μοντελοποιείται ευκολότερα ώστε να βρεθούν τρόποι αποτελεσματικότερης συντήρησης.

Συνοψίζοντας το κεφάλαιο αυτό, παρατηρήσαμε ότι κάθε εξάρτημα περιγράφεται από μια στατιστική κατανομή, που διέπει την λειτουργία του. Όσο πιο πιστά την προσεγγίσουμε τόσο πιο αξιόπιστο γίνεται το σύστημα μέσα στο οποίο θα χρησιμοποιηθεί. Παρουσιάστηκαν οι βασικές μεθοδολογίες για την μοντελοποίηση και ανάλυση συστημάτων από την σκοπιά της αξιοπιστίας, με τα μοντέλα RBD και Markov. Οι τεχνικές αυτές χρησιμοποιούνται για την περιγραφή των συστημάτων στον χώρο καταστάσεων και για την κατανόηση της συμπεριφοράς τους σε συνθήκες σφαλμάτων, πράγμα το οποίο δεν θα ήταν δυνατό αν το μελετούσαμε μόνο από την σκοπιά της απόδοσης τους.

ΚΕΦΑΛΑΙΟ 3 Information Redundancy

Τα ψηφιακά συστήματα χρησιμοποιούν δεδομένα για τις εσωτερικές αλληλεπιδράσεις τους, τα οποία έχουν την μορφή ομάδας από δυαδικά ψηφία. Επομένως υπάρχει η πιθανότητα κατά την επεξεργασία και την αποθήκευση δεδομένων, αυτά να μεταβληθούν λόγω φυσικών ατελειών του υλικού. Θα πρέπει να υπάρχει μια πρόληψη σε περίπτωση λανθασμένων bit, εντός μιας δυαδικής λέξης. Επιπλέον είναι αναγκαίο να διορθώνονται λάθη των δεδομένων, ώστε το σύστημα να μπορεί να επανέρχεται σε συνθήκες κανονικής λειτουργίας. Η ιδιότητα αυτή προϋποθέτει την χρήση εφεδρικών bit για τον εντοπισμό και την επιδιόρθωση. Έτσι, το μήκος, δηλαδή ο αριθμός των bit που αποτελούν μια λέξη, πρέπει να είναι μεγαλύτερο από τα αρχικά δεδομένα.

Η διαδικασία ενσωμάτωσης επιπλέον ψηφίων ελέγχου στα ψηφία της πληροφορίας ονομάζεται *κωδικοποίηση*. Η αντίστροφη , δηλαδή ο διαχωρισμός της πληροφορίας από ένα codeword, ονομάζεται *αποκωδικοποίηση*. Ο αριθμός σε bit της πληροφορίας προς τον αριθμό των bit μετά την κωδικοποίηση λέγεται *ρυθμός κώδικα.* Επιπροσθέτως, ένας κώδικας των n-bit ο οποίος παράχθηκε από κωδικοποίηση k-bit δεδομένων, έχει 2k έγκυρες λέξεις, 2^n-2^k μη έγκυρες και ρυθμό k / n. Το σύνολο από 2^k έγκυρων κωδικών ονομάζεται *μπλοκ κώδικας*.
Οι βασικές προδιαγραφές τέτοιων κωδικών είναι οι ακόλουθες:

- Εντοπίζει όλα τα πιθανά σφάλματα.
- Ικανοποιεί σε μεγάλο βαθμό τον εντοπισμό σφαλμάτων με όσο το δυνατό λιγότερο πλεονασμό.
- Η διαδικασία της κωδικοποίησης και αποκωδικοποίησης να είναι απλή και γρήγορη.

Αναφέρονται στην ενσωμάτωση πρόσθετων πληροφοριών στα δεδομένα με σκοπό τον εντοπισμό και την επιδιόρθωση σφαλμάτων. Η μορφή αυτή πλεονασμού υλοποιείται με την χρήση κωδικών εντοπισμού σφαλμάτων (EDC) και κωδικών επιδιόρθωσης σφαλμάτων (ECC) που έχουν μεγάλη εφαρμογή στα λογικά κυκλώματα,όπως οι μνήμες, αλλά και στα δεδομένα που αποστέλλονται σε δίκτυα υπολογιστών, την σειριακή και παράλληλη επικοινωνία.

- *Error Detecting Codes EDC:* Χρησιμοποιούνται για την ανίχνευση λαθών σε οποιοδήποτε δεδομένα. Ο εντοπισμός σφαλμάτων γίνεται στην φάση της αποκωδικοποίησης, φανερώνοντας την ασυμβατότητα των απεσταλμένων δεδομένων. Εφαρμόζονται κατά την διαδικασία αντικατάστασης ενός προβληματικού εξαρτήματος.

- *Error Correcting Codes ECC:* Είναι οι κώδικες που έχουν την δυνατότητα να αναγνωρίζουν τα σφάλματα δεδομένων και να καθορίζουν ποια θα έπρεπε να είναι η σωστή πληροφορία κατά τη λήψη τους. Χρησιμοποιούνται για την απόκρυψη σφαλμάτων σε ένα σύστημα, ώστε αυτό να συνεχίζει την λειτουργία του, χωρίς να διακοπεί η υπηρεσία του.

Οι κώδικες μπορούν να χαρακτηρισθούν από μια ιδιότητα η οποία είναι η διαχωρισιμότητα. Αν οι πληροφορίες των κωδικών είναι ξεχωριστά από τα δεδομένα τότε είναι διαχωρίσιμος, οπότε δεν είναι αναγκαίο το στάδιο της αποκωδικοποίησης για να ξεχωρίσουμε τα δεδομένα. Στην περίπτωση που εφαρμόζεται στα δεδομένα είναι μη διαχωρίσιμος και η κωδικοποίηση πρέπει να γίνει από έναν αποκωδικοποιητή. Εξαιτίας αυτού οι μη διαχωρίσιμοι κώδικες δεν έχουν διαδεδομένη εφαρμογή παρά μόνο σε συγκεκριμένες εφαρμογές, όπως στην κωδικοποίηση εισόδων/εξόδων και στην ανάθεση καταστάσεων σε ακολουθιακά κυκλώματα, όπου αξιοποιούνται αποδοτικά. Ένας διαχωρίσιμος κώδικας αποτελούμενος από k bit δεδομένων είναι συστηματικός, όταν όλοι οι 2^k συνδυασμοί δεδομένων προκύπτουν στις αντίστοιχες κωδικοποιημένες. Ένας κώδικας χαρακτηρίζεται συστηματικός όταν η ακολουθία της πληροφορίας είναι μέρος της κωδικοποιημένης ακολουθίας. Γενικά, οι γραμμικοί μπλοκ κώδικες ενσωματώνουν την πληροφορία είτε στην αρχή είτε στο τέλος του codeword.

Θεωρητικές έννοιες

Οι κώδικες που μπορούν να αποστείλουν μια συμβολοσειρά δεδομένων σε συνθήκες ηλεκτρομαγνητικού θορύβου ή σε αναξιόπιστα κανάλια επικοινωνίας ονομάζονται μπλοκ κώδικες και περιγράφονται από την σχέση $C : \Sigma^k \to \Sigma^n$, k είναι το μήκος του μηνύματος ενώ n το μήκος του μπλοκ κώδικα. Το Σ είναι ένα πεπερασμένο σύνολο ενώ το k και n είναι ακέραιοι αριθμοί

Αλφάβητο: Τα δεδομένα για να κωδικοποιηθούν πρέπει να ικανοποιούν τους κανόνες του κώδικα. Οι κανόνες είναι το αλφάβητο ενός κώδικα και συμβολίζεται με q. Για q=2 έχουμε δυαδικό κώδικα. Σε πολλές εφαρμογές θεωρείται ως δύναμη για τον ορισμό του συνόλου Σ σε πεπερασμένο GF(q) πεδίο.

Ρυθμός κώδικα: συμβολίζεται με R και ισούται με την ποσότητα k / n. Ένα μεγάλο ποσοστό σημαίνει ότι το ποσό του μηνύματος που μεταδίδεται ανά μπλοκ είναι υψηλό. Υπό την έννοια αυτή , το ποσοστό αυτό μετρά την ταχύτητα μετάδοσης και η ποσότητα 1 -R μετρά την επιβάρυνση που προκύπτει λόγω της κωδικοποίησης με τον μπλοκ κωδικό. Γενικά το ποσοστό αυτό δεν μπορεί να υπερβαίνει το 1 καθώς τα δεδομένα δεν μπορούν να συμπιεστούν χωρίς απώλειες.

Code Word (CW): είναι μια συλλογή από σύμβολα, που αντιπροσωπεύουν χρήσιμα δεδομένα και ομαδοποιούνται σύμφωνα με προκαθορισμένους κανόνες κωδικοποίησης. Για παράδειγμα, μια λέξη των 4-bit κωδικοποιημένη στο δυαδικό είναι η 1010.

Hamming weight: Το βάρος Hamming μιας συμβολοσειράς είναι ο αριθμός των συμβόλων που είναι διαφορετικό από το μηδέν βάσει του αλφαβήτου που χρησιμοποιείται. Είναι έτσι ισοδύναμη με την απόσταση Hamming από τα μηδενικά στοιχεία του ίδιου μήκους. Για την πιο τυπική περίπτωση, της δυαδικής αλφαβήτου αυτό είναι ο αριθμός 1.

Hamming Distance: είναι το πλήθος των ψηφίων που δύο δυαδικές λέξεις διαφέρουν.
π.χ. HD (0101, 0010) = 3 , που σημαίνει ότι ο δεύτερη λέξη πρέπει να μεταβληθεί τρείς φορές για να προκύψει ο πρώτος.
Code Distance: Εντός ενός συγκεκριμένου συνόλου λέξεων, δηλώνει το ελάχιστο HD μεταξύ των λέξεων.

Για παράδειγμα έστω ένας κώδικας που αποτελείται από τους όρους 10101, 01001, 01110. Έπειτα από σύγκριση των όρων προκύπτουν τα εξής Hamming Distances:

$$Binarycode = (10101, 01001, 01110)$$

$$HD(10101, 01001) = 3$$
$$HD(10101, 01110) = 4$$
$$HD(01001, 01001) = 3$$

Επομένως είναι προφανές ότι CD=3. Σε επικοινωνία βασισμένη σε δυαδικές λέξεις, αν ληφθεί η λέξη 01100 θα δημιουργηθούν σφάλματα, καθώς δεν αποτελεί έγκυρη λέξη στον παραπάνω κώδικα.

Θεωρήματα

Ισχύουν οι παρακάτω σχέσεις με βάση τους κώδικες:
Για εντοπισμό σφαλμάτων έχουμε,

$$CD \geq d + 1 \quad (1)$$

όπου d ο αριθμός σφαλμάτων. Για παράδειγμα CD=3, τότε έχουμε $d \leq 2$, που σημαίνει ότι μπορούν να εντοπιστούν ένα ή/και δύο σφάλματα.
Για επιδιόρθωση σφαλμάτων έχουμε,

$$CD \geq 2t + 1 \quad (2)$$

όπου t ο αριθμός σφαλμάτων που χρειάζονται επιδιόρθωση.
Για παράδειγμα CD=3, τότε έχουμε $t \leq 1$,που σημαίνει ότι μπορεί να επιδιορθωθεί μόνο ένα από τα δύο σφάλματα που εντοπίστηκαν προηγουμένως.
Για εντοπισμό και επιδιόρθωση σφαλμάτων έχουμε,

$$CD \geq d + t + 1 \quad , \quad t \leq d \quad (3)$$

όπου d ο αριθμός σφαλμάτων προς εντοπισμό και t ο αριθμός σφαλμάτων προς επιδιόρθωση. Ως CD στη σχέση αυτή, λαμβάνουμε την περίπτωση για $t = 0$, δηλαδή την μεταβολή από την αρχική λέξη , στην τελική του κώδικα.
Ας συνεχίσουμε το παράδειγμα με τον δυαδικό μας κώδικα, εφαρμόζοντας τα παραπάνω θεωρήματα. Ο κώδικας βρήκαμε ότι έχει CD=3 οπότε βάσει της σχέσης (1) έχουμε $d \leq 2$. Δηλαδή μπορούν να εντοπιστούν έως και δυο σφάλματα. Από την σχέση (2) προκύπτει $t \leq 1$, που σημαίνει ότι ένα από τα δυο σφάλματα θα έχει την δυνατότητα να επιδιορθωθεί. Η σχέση (3) με αντικατάσταση των t, d μας δίνει D=4 ο οποίος είναι ο μέγιστος αριθμός διαφορών σε ψηφία που υπάρχουν στον κώδικα αυτό.
Στο σημείο αυτό μπορούμε να εφαρμόσουμε έναν απλό τρόπο για την επιδιόρθωση. Σε επικοινωνία βασισμένη σε δυαδικές λέξεις, αν ληφθεί η λέξη 01100 θα δημιουργηθούν σφάλματα, καθώς δεν αποτελεί έγκυρη λέξη στον παραπάνω

κώδικα. Το πρόβλημα αυτό λύνεται με την πραγματοποίηση της πράξης XOR μεταξύ της εσφαλμένης και κάθε λέξη του κώδικα μας.
Έτσι λοιπόν έχουμε,

$$\begin{aligned} 01100 \oplus 10101 &= 11001 \quad ,HD=3 \\ 01100 \oplus 01001 &= 00101 \quad ,HD=2 \\ 01100 \oplus 01110 &= 00010 \quad ,HD=1 \end{aligned}$$

Παρατηρώντας τις πράξεις και γνωρίζοντας ότι $t \le 1$ συμπεραίνουμε ότι η λέξη πρέπει να ήταν η 01110.

Παράδειγμα
Έστω ότι έχουμε τον δυαδικό κώδικα [00000,11111].Βρίσκουμε ότι
$CD=5, \quad 5-1 \ge d \rightarrow d \le 4$

Από τα τέσσερα σφάλματα που μπορούν να συμβούν βρίσκουμε ότι
$CD=5, \quad 5-1 \ge 2t \rightarrow t \le 2$
που σημαίνει ότι μπορούν να διορθωθούν 2 , 1 ή κανένα λάθος.
Για επαλήθευση εφαρμόζουμε την σχέση (3) για t=0
$CD \ge 4+0+1 \rightarrow CD=5$
Τέλος, ακολουθεί ενδεικτικός συνοπτικός πίνακας για την κατανόηση των θεωρημάτων:

Code	CD	d	t
(00,11)	2	1	0
(000,111)	3	2 ή λιγότερα	1 ή 0
(0000,1111)	4	3 ή λιγότερα	1 ή 0
(00000,11111)	5	4 ή λιγότερα	2 ή 1 ή 0

Πίνακας 3.1 Σύνοψη των θεωρημάτων για κώδικες.

3.1 Repetition Codes

Ένας κώδικας επανάληψης αποτελεί ένα σύστημα κωδικοποίησης, το οποίο επαναλαμβάνει τα bits σε ένα κανάλι για να επιτευχθεί επικοινωνία χωρίς λάθη[1]. Δίνεται μια ροή δεδομένων προς μετάδοση και τα δεδομένα χωρίζονται σε μπλοκ από bits. Κάθε μπλοκ μεταδίδεται κάποιο προκαθορισμένο αριθμό φορών N. Για παράδειγμα, για να στείλετε το μοτίβο bit "1011", το μπλοκ τεσσάρων-bit μπορεί να επαναληφθεί τρεις φορές, δημιουργώντας έτσι "1011 1011 1011". Ωστόσο, εάν η λέξη των δώδεκα-bit αποκωδικοποιήθηκε ως "1010 1011 1011" - όπου η πρώτη ομάδα διαφέρει με τις άλλες δύο - μπορεί να καθοριστεί ότι έχει προκύψει κάποιο σφάλμα.

Οι κώδικες επανάληψης είναι πολύ αναποτελεσματικοί και μπορεί είναι επιρρεπείς σε προβλήματα αν το σφάλμα εμφανίζεται ακριβώς στην ίδια θέση για κάθε ομάδα (π.χ. το "1010 1010 1010" στο προηγούμενο παράδειγμα θα ανιχνεύεται ως σωστή). Το πλεονέκτημα των κωδίκων επανάληψης είναι ότι είναι εξαιρετικά απλοί και ότι η πιθανότητα των σφαλμάτων μειώνεται για μεγαλύτερο αριθμό επαναλήψεων μεγάλου όγκου δεδομένων. Η λογική τους είναι από τις πρώτες μορφές ανοχής σφαλμάτων και έχουν έμμεση συσχέτιση με την μέθοδο πλεονασμού NMR (N-Modular Redundancy) του υλικού.

3.2 Κώδικες Borden

Είναι μια μεθοδολογία που έχει διαδεδομένη χρήση σε αρκετούς κώδικες εντοπισμού σφαλμάτων λόγω της προσαρμοστικότητάς του. Οι κώδικες Borden είναι μη διαχωρίσιμοι κώδικες. Ακολουθεί ο ορισμός:

Αν $C(n/t)$ το σύνολο των κωδικών μήκους n bits, για το οποίο ακριβώς m bits είναι '1', τότε η συνένωση όλων αυτών των κωδικών σύμφωνα με τον τύπο, είναι γνωστός ως κώδικας Borden.

$$w = (n/2) \, mod \, (t+1)$$

$$C(n,t) = \sum \binom{n}{w}$$

έστω ότι επιθυμούμε να κατασκευάσουμε τον κώδικα Borden[5/2]. Με εφαρμογή του τύπου προκύπτει ότι w=(5/2)mod (3)=2 . Αυτό σημαίνει ότι το m ανήκει στο σύνολο {0, 2, 4}. Επιπλέον κάθε codeword μήκους 5 bit που αποτελείται από κανένα , δύο ή τέσσερις άσσους ανήκει στον κώδικα αυτό. Ακολουθούν όλοι οι συνδιασμοί του κώδικα αυτού: 0000, 00011, 00110, 01100, 11000, 00101, 01010, 10100, 01001, 10010, 10001, 01111, 11110, 10111, 11011, 11101.

Ο κώδικας C(n, t) έχει την ικανότητα να εντοπίσει t μεταβολές από ΄1΄ σε ΄0΄ και αντίστροφα αλλά όχι και τα δύο ταυτόχρονα. Είναι ο βέλτιστος κώδικας για τον εντοπισμό σφαλμάτων τέτοιου τύπου.

Τέλος, ενδιαφέρον παρουσιάζουν δύο ειδικές περιπτώσεις του κώδικα Borden,

- Αν t=1 , τότε ο κώδικας Borden μετατρέπεται σε ισοτιμίας
- Αν n=2t , τότε προκύπτει ένας ειδικός κώδικας γνωστός ως k/2k ή k από 2k.

3.3 Berger code

Είναι ένας κώδικας όπως ο Borden με την διαφορά ότι είναι διαχωρίσιμος. Λειτουργεί με την προσκόλληση ψηφίων ελέγχου στα δεδομένα για τον εντοπισμό σφαλμάτων κατά την λήψη πληροφοριών. Τα check bits αναπαριστούν τον αριθμό των ΄0΄ της δυαδικής λέξης. Για παράδειγμα για μια λέξη των 3 bit χρειαζόμαστε δύο ψηφία ελέγχου. Οι κώδικες Berger χρησιμοποιούνται σε *κυκλώματα καθυστέρησης*. Είναι πιο απλός από τους Borden, αν και ο Bose που ακολουθεί είναι περισσότερο αποτελεσματικός.

Δεδομένα	Ψηφία ελέγχου	Codeword
000	11	00011
001	10	00110
010	10	01010
011	01	01101
100	10	10010
101	01	10101
110	01	11001
111	00	11100

$$k = \log_2(n+1)$$
$$n = 2^k - 1$$
$$cw = k + n$$

Πίνακας 3.2 Κώδικας Berger τριών ψηφίων.

Η λειτουργία του περιγράφεται με τους παραπάνω τύπους, όπου k είναι ο αριθμός των ψηφίων ελέγχου και n το μήκος της λέξης προς κωδικοποίηση.

3.4 Bose codes

Οι κώδικες αυτοί προσφέρουν την ίδια ικανότητα εντοπισμού με τους Berger, όμως με το πλεονέκτημα της χρήσης λιγότερων ψηφίων ελέγχου. Χρησιμοποιεί ακριβώς n check bits για 2^n λέξεις δεδομένων. Οι κώδικες Bose κατασκευάζονται με τους εξής κανόνες:

- Αν το πλήθος των μηδενικών μιας λέξης είναι μηδενικό ή 2^n, τότε συμπληρώνουμε τα πρώτα $2^{(n-1)}$ ψηφία και θέτουμε το $2^{(n-1)} - 1$ στις θέσεις των check bits.
- Αν το πλήθος των μηδενικών είναι ανάμεσα σε $2^{(n-1)}$ και $2^n - 1$, τότε αντικαθιστούμε στο τέλος την δυαδική αναπαράσταση του αριθμού αυτού, όπως γίνεται στους κώδικες Berger.
- Αν βρίσκονται ανάμεσα στο 1 και $2^{(n-1)} - 1$ τότε πρόσθεσε στο τέλος το αποτέλεσμα της αφαίρεσης του αριθμού με το ένα.

Ακολουθεί ο πίνακας Bose για πληροφορίες μήκους τεσσάρων ψηφίων,

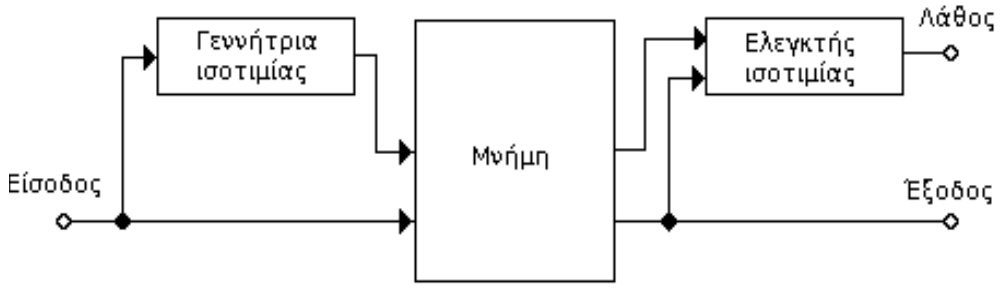
Data	Check bits	Codeword
0000	01	110001
0001	11	000111
0010	11	001011
0011	10	001110
0100	11	010011
0101	10	010110
0110	10	011010
0111	00	011100
1000	11	100011
1001	10	100110
1010	10	101010
1011	00	101100
1100	10	110010
1101	00	110100
1110	00	111000
1111	01	001101

Πίνακας 3.3 Κώδικας Bose τεσσάρων ψηφίων.

Έστω για το πρώτο στοιχείο του πίνακα (0000) , ισχύει ο πρώτος κανόνας. Παρατηρούμε ότι το πλήθος των μηδενικών είναι τέσσερα άρα $2^n = 4 \Leftrightarrow n = 2$ που υποδεικνύει ότι θα χρησιμοποιήσουμε δύο ψηφία ελέγχου για την κωδικοποίηση. Έπειτα ακολουθεί η συμπλήρωση των δύο MSB των δεδομένων. Ως αποτέλεσμα προκύπτει το 1100. Τέλος υπολογίζουμε τον αριθμό $2^{(n-1)} - 1 = 1_{10} \equiv 01_2$ και τον προσθέτουμε στο τέλος της λέξης. Έτσι προκύπτει η κωδικοποιημένη λέξη 110001.

3.5 Κώδικες Ισοτιμίας

Οι κώδικες αυτοί έχουν εφαρμογή στις μνήμες υπολογιστών[1]. Τα δεδομένα κατά την ανάγνωση και την εγγραφή, πρέπει να διατηρούν την ακεραιότητα τους. Κατά την εγγραφή μιας λέξης στη μνήμη γίνεται ο υπολογισμός του ψηφίου ισοτιμίας από μια γεννήτρια, που είναι απαραίτητο για την διαδικασία αυτή. Στην ανάκτηση από την μνήμη γίνεται έλεγχος βάσει της ισοτιμίας και της λέξης. Το υπολογισθέν ψηφίο συγκρίνεται, από τον ελεγκτή ισοτιμίας, με τα δεδομένα που είναι αποθηκευμένα στη μνήμη για την διάγνωση πιθανού σφάλματος.



Εικόνα 3.1: Κύκλωμα κωδικοποιητή ισοτιμίας.

Πρέπει να αναφερθεί ότι για μια λέξη των n-bit δεδομένων υπάρχουν 2^n CW. Η πρόσθεση ενός bit ισοτιμίας θα έχει ως αποτέλεσμα $2^{(n+1)}$ CW. Από αυτές τις λέξεις οι $2^{(n+1)}/2$ θα αποτελούνται από περιττό αριθμό άσσων ενώ οι υπόλοιπες ζυγό αριθμό.

Για περιττή/ ζυγή ισοτιμία μόνο οι λέξεις με περιττό/ζυγό αριθμό άσσων αποτελούν έγκυρες λέξεις. Στην περίπτωση λάθους, η περιττή/ζυγή λέξη θα μεταβληθεί σε ζυγή/περιττή και έτσι θα εντοπιστεί το σφάλμα. Πρέπει να σημειωθεί ότι στην περίπτωση δύο λαθών, ο εντοπισμός δεν θα είναι εφικτός.

Γενικά οι κώδικες ισοτιμίας έχουν την ιδιότητα να εντοπίζουν περιττό αριθμό σφαλμάτων. Για τον εντοπισμό ζυγού αριθμού σφαλμάτων χρησιμοποιούνται περίπλοκοι κώδικες ισοτιμίας.

3.6 Hamming code

Οι κώδικες Hamming είναι από τους πρώτους κώδικες ανίχνευσης και επιδιόρθωσης σφαλμάτων. Έχουν την δυνατότητα να εντοπίζουν δύο λάθη και να επιδιορθώνουν ένα ψηφίο, κατά την λήψη της πληροφορίας. Χρησιμοποιεί κωδικοποίηση ισοτιμίας στην μετάδοση των δεδομένων με επιπλέον ψηφία, που είναι διαχωρίσιμα από τα δεδομένα. Επισημαίνεται ότι εντοπισμός δύο λαθών δεν μπορεί να γίνει ταυτόχρονα με την επιδιόρθωση επειδή τα σφάλματα των δύο ψηφίων θα μεταβληθούν σύμφωνα με την ισοτιμία, οπότε υπάρχει κίνδυνος αλλαγής ορθού ψηφίου των δεδομένων.

Στο ένα άκρο της επικοινωνίας ο αποστολέας κωδικοποιεί την πληροφορία με τον κώδικα και την αποστέλλει. Στο άλλο άκρο παραλήπτης αποκωδικοποιεί την πληροφορία με παρόμοια μέθοδο, διαχωρίζοντας τα ψηφία ισοτιμίας από την λέξη. Στην περίπτωση που τα ληφθέντα δεδομένα συμφωνούν με την ισοτιμία τότε έχουμε

επιτυχή αποστολή πληροφοριών. Σε αντίθετη περίπτωση έχουμε διαφορές ως προς τις τιμές της ισοτιμίας, που σημαίνει ότι υπάρχουν σφάλματα στην πληροφορία. Για να διορθωθεί αυτό το πρόβλημα προσθέτουμε τα ψηφία που έχουν την διαφορά ισοτιμίας και εντοπίζουμε την θέση με το λάθος. Με τον τρόπο αυτό έχουμε την δυνατότητα να διορθώσουμε το λανθασμένο ψηφίο αλλά και να βρούμε ποια ήταν τα έγκυρα δεδομένα χωρίς να είναι υποχρεωτικό ο αποστολέας να αποστείλει ξανά τις ίδιες πληροφορίες. Οι τύποι που περιγράφουν ένα τέτοιο κώδικα είναι:

$$H\left[n,k,r\right] \quad \rightarrow \quad r \geq 2 \quad , \quad n = 2^r - 1 \quad , \quad k = 2^r - r - 1$$

όπου n είναι ο αριθμός των ψηφίων που θέλουμε να αποστείλουμε, k είναι ο αριθμός των ψηφίων για τα δεδομένα και r ο αριθμός των ψηφίων ισοτιμίας που θα χρησιμοποιήσουμε για την κωδικοποίηση. Αναφέρονται ενδεικτικά μερικές περιπτώσεις κωδικών για διάφορες τιμές του r.

R	n	k	Όνομα	Ρυθμός R= k/n
2	3	1	H[3,1,2]	≈ 0.333
3	7	4	H[7,4,3]	≈ 0.571
4	15	11	H[15,11,4]	≈ 0.733
5	31	26	H[31,26,5]	≈ 0.839
6	63	57	H[63,57,6]	≈ 0.904
…				

Πίνακας 3.4 Ιδιότητες κωδικών Hamming.

Αναλυτικότερα, η γεννήτρια ισοτιμίας δημιουργεί έναν πίνακα H ο οποίος περιλαμβάνει έναν μοναδιαίο πίνακα και τους υπόλοιπους εναπομείναντες συνδυασμούς του στο δυαδικό σύστημα. Ο πίνακας αυτός ονομάζεται πίνακας ισοτιμίας. Επιπλέον για την κωδικοποίηση χρησιμοποιείται ένας ακόμη πίνακας G που περιέχει και αυτός έναν μοναδιαίο πίνακα και τους συνδυασμούς του. Ο πίνακας αυτός είναι γνωστός ως πίνακας γεννήτρια.

Ένα μήνυμα για να αποσταλεί είναι απαραίτητο να κωδικοποιηθεί. Για να γίνει αυτό, το πολλαπλασιάζουμε με τον πινάκα γεννήτρια G. Το γινόμενο αυτό είναι το διάνυσμα κωδικοποίησης c ή αλλιώς codeword. Ο παραλήπτης για να ελέγξει την εγκυρότητα του μηνύματος υπολογίζει το γινόμενο s μεταξύ του ληφθέντος codeword και του ανάστροφου του πίνακα ισοτιμίας. Αν το διάνυσμα του γινομένου αυτού είναι μηδενικό τότε έχουμε έγκυρη λήψη πληροφορίας.

Παρόλο που σφάλματα είναι πιθανό να συμβούν, λόγω ηλεκτρομαγνητικού θορύβου στο κανάλι επικοινωνίας ή εσφαλμένης ανάγνωσης εγγραφής σε μια μνήμη RAM. Θα προκύψει μη μηδενικό γινόμενο αποκωδικοποίησης. Για να εντοπιστεί ποιο από τα ψηφία της πληροφορίας έχει μεταβληθεί από 1 σε 0 και αντίστροφα, ελέγχουμε με βάσει το s τις στήλες του πίνακα H. Με την διαδικασία αυτή παράγεται το διάνυσμα σφάλματος e.

Έτσι μπορούμε πλέον να υπολογίσουμε την σωστή πληροφορία ,απλά διορθώνοντας το ψηφίο, με πρόσθεση του σφάλματος και του codeword. Ακολουθεί συνοπτικός πίνακας με τις βασικές σχέσεις των κωδικών Hamming.

Τύπος	**Περιγραφή**
$H = \lfloor I_{(n-k)} \quad , \quad A \rfloor$	Πίνακας Ισοτιμίας
$G = \lfloor A^T , \quad I_k \rfloor$	Πίνακας Γεννήτρια
$c = m \times G$	Διάνυσμα κωδικοποίησης
$s = c \times H^T$	Γινόμενο αποκωδικοποίησης
$\vec{e}_i = H^T \ni \vec{s}$	Διάνυσμα σφάλματος
$r = c + e_i$	Διάνυσμα επιδιόρθωσης

Πίνακας 3.5 Μέθοδος κωδικοποίησης κωδικών Hamming.

Έστω για παράδειγμα ότι επιθυμούμε να δημιουργήσουμε τον κώδικα Hamming (7,4,3). Αρχικά θα πρέπει να κατασκευάσουμε τους πίνακες H και G. Εφαρμόζοντας τους τύπους βρίσκουμε ότι μέσα σε αυτούς θα δημιουργήσουμε έναν μοναδιαίο πίνακα 3x3 ,που περιέχει τους αριθμούς 1,2,4 στις στήλες τα ψηφία ισοτιμίας και τον πίνακα A με τους υπόλοιπους συνδυασμούς. Επομένως έχουμε τους πίνακες,

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Με τον ίδιο τρόπο κατασκευάζουμε τον πίνακα γεννήτριας G, που θα αποτελείται από έναν μοναδιαίο 4x4 και τον ανάστροφο του πίνακα A. Έτσι λοιπόν έχουμε,

$$A^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Έστω ότι επιθυμούμε να κωδικοποιήσουμε το μήνυμα $\vec{m} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$.

Στην διαδικασία αυτή είναι απαραίτητο να γίνει πολλαπλασιασμός modulo 2,που ισοδυναμεί με την λογική πράξη AND, του μηνύματος με τον πίνακα γεννήτρια. Αποτέλεσμα αυτού είναι το διάνυσμα κωδικοποίησης. Η πληροφορία αυτή περιλαμβάνει το μήνυμα καθώς και τα απαραίτητα ψηφία ισοτιμίας για την επαλήθευση της εγκυρότητας από τον ελεγκτή ισοτιμίας. Στο στάδιο αυτό η πληροφορία είναι έτοιμη να αποσταλεί σε ένα κανάλι επικοινωνίας προς τον παραλήπτη

$$\vec{c} = \vec{m} \times G = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Από την πλευρά του ενεργείται η διαδικασία της αποκωδικοποίησης. Για να γίνει αυτό λαμβάνουμε το γινόμενο του διανύσματος κωδικοποίησης με τον ανάστροφο του πίνακα ισοτιμίας.

$$\vec{s} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Το αποτέλεσμα θα είναι το διάνυσμα αποκωδικοποίησης. Το μέγεθος του προφανώς θα είναι από τις ιδιότητες πινάκων 1x3. Στην περίπτωση που η τιμή του είναι μηδενική, η πληροφορία που έλαβε ο παραλήπτης είναι ορθή, όπως φαίνεται και στο παραπάνω παράδειγμα.

Όμως θα πρέπει να αναρωτηθεί κανείς τι γίνεται αν η πληροφορία που έλαβε ο παραλήπτης δεν είναι έγκυρη. Ο R. Hamming φρόντισε να αντιμετωπιστεί ως ενός βαθμού την εμφάνιση τέτοιων σφαλμάτων. Η διαδικασία της επιδιόρθωσης είναι από τις σημαντικές ιδιότητες των κωδικών αυτών. Οπότε ας υποθέσουμε ότι για κάποιο λόγο η κωδικοποιημένη πληροφορία είναι η 101001.Ακολουθείται η ίδια ακριβώς διαδικασία αποκωδικοποίησης με προηγουμένως για να βρούμε ότι το διάνυσμα αποκωδικοποίησης είναι μη μηδενικό.

$$\vec{s} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Το διάνυσμα αυτό θα χρησιμοποιηθεί για την παραγωγή ενός άλλου διανύσματος, που είναι γνωστό ως διάνυσμα σφάλματος. Οι διαστάσεις του έχουν τις διαστάσεις του codeword. Πρακτικά γίνεται έλεγχος του με τις γραμμές του ανάστροφου πίνακα ισοτιμίας H. Αν υπάρχει σε κάποια γραμμή, τότε θα δημιουργηθεί το διάνυσμα σφάλματος με άσσο στην συγκεκριμένη θέση του διανύσματος και μηδενικά οπουδήποτε αλλού. Έτσι παρατηρώντας ότι το αποτέλεσμα είναι στοιχείο της τρίτης γραμμής του ανάστροφου πίνακα H, συμπεραίνουμε ότι η τρίτη θέση του διανύσματος έχει υποστεί σφάλμα.

$$\vec{s} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, \quad \vec{e}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Όμως κατά το σχεδιασμό ενός τέτοιου κώδικα έχει δημιουργηθεί ένας πίνακας σφαλμάτων από πριν, ως μέτρο πρόληψης. Ο πίνακας αυτός χρησιμοποιείται για την αναζήτηση και τον εντοπισμό της θέσης του λάθους. Για τον συγκεκριμένο κώδικα έχει την παρακάτω μορφή,

Διάνυσμα σφάλματος	Διάνυσμα αποκωδικοποίησης
1000000	100
0100000	010
0010000	001
0001000	110
0000100	101
0000010	011
0000001	111

Πίνακας 3.6 Μέθοδος άποκωδικοποίησης κωδικών Hamming.

Το τελευταίο βήμα που απομένει είναι η επιδιόρθωση της πληροφορίας. Η διαδικασία της πρόσθεσης του διανύσματος κωδικοποίησης και του διανύσματος σφάλματος έχει ως αποτέλεσμα την έγκυρη πληροφορία. Πρέπει να σημειωθεί ότι η πρόσθεση είναι και αυτή δυαδική και ισοδυναμεί με την λογική πράξη XOR μεταξύ των ψηφίων. Έτσι λοιπόν έχουμε την επιδιόρθωση της πληροφορίας κάθε φορά που έχουμε μεταβολή ενός ψηφίου δεδομένων ή ψηφίων ισοτιμίας.

$$\vec{r} = \vec{c} + \vec{e}_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Συνοψίζοντας, ο κώδικας Hamming είναι από τους πιο αποτελεσματικούς κώδικες επιδιόρθωσης καθώς προσφέρει ποιοτική σχέση μεταξύ εντοπισμού και επιδιόρθωσης στον ίδιο χώρο εφαρμογής. Ως χώρο λαμβάνουμε την ποσότητα σε ψηφία που προστίθενται στην πληροφορία ώστε να ενεργήσει ο εντοπισμός και η επιδιόρθωση. Αναφέραμε όμως για τον εντοπισμό και την επιδιόρθωση μόνο ενός ψηφίου για τους κώδικες αυτούς. Ο εντοπισμός δύο λαθών αναλύεται στην επόμενη παράγραφο με την οικογένεια κωδικών SECDED, οι οποίοι είναι επέκταση των Hamming.

3.7 SECDED

Όπως αναφέρθηκε προηγουμένως οι κώδικες Hamming έχουν την δυνατότητα να εντοπίζουν ως και δύο λάθη. Για να γίνει αυτό πραγματοποιήσιμο πρέπει να τροποποιηθούν. Την επέκταση τους αποτελούν οι κωδικοί SECDED (Single Error Correction and Double Error Detection). Είναι χρήσιμοι στις περιπτώσεις που ένα λάθος είναι πιθανό να συμβεί, αλλά όταν συμβαίνουν παραπάνω λάθη να μην επέρχεται άμεσα η αποτυχία λειτουργίας του συστήματος.

Η επέκταση που επιδέχονται βρίσκεται στην ενσωμάτωση ενός επιπλέον ψηφίου ισοτιμίας κατά την κωδικοποίηση. Αυτό σημαίνει ότι ο προηγούμενος κώδικας θα γίνει Hamming[8, 4, 4]. Το ψηφίο αυτό λαμβάνει τιμές ανάλογα με την ισοτιμία ολόκληρης της πληροφορίας συμπεριλαμβανομένων και των υπόλοιπων ψηφίων ισοτιμίας. Είναι προφανές λοιπόν ότι θα μεταβληθούν οι πίνακες ισοτιμίας Η και γεννήτριας G. Λαμβάνει την μορφή μιας επιπλέον γραμμής στο κάτω μέρος του πίνακα Η και μιας επιπλέον στήλης στον G. Όποτε έχουν παρακάτω τελική μορφή:

$$H=\begin{bmatrix}1&0&0&0&1&1&0&1\\0&1&0&0&1&0&1&1\\0&0&1&0&0&1&1&1\\0&0&0&1&0&0&0&0\end{bmatrix},\quad G=\begin{bmatrix}1&1&0&0&1&0&0&0\\1&0&1&0&0&1&0&0\\0&1&1&0&0&0&1&0\\1&1&1&0&0&0&0&1\end{bmatrix}$$

Ας συνεχίσουμε το προηγούμενο παράδειγμα κωδικοποιώντας το μήνυμα 0011. Το παραγόμενο διάνυσμα κωδικοποίησης θα έχει διαστάσεις 1x8 και υπολογίζεται ως εξής,

$$\vec{c}=\vec{m}\times G=\begin{bmatrix}0&0&1&1\end{bmatrix}\times G=\begin{bmatrix}1&1&0&0&1&0&0&0\\1&0&1&0&0&1&0&0\\0&1&1&0&0&0&1&0\\1&1&1&0&0&0&0&1\end{bmatrix}=\begin{bmatrix}1&0&0&0&0&0&1&1\end{bmatrix}$$

Κατά την λήψη της πληροφορίας ή την ανάγνωση από την μνήμη ορισμένες φορές μπορούν να πραγματοποιηθούν μεταβολές σε ορισμένα ψηφία της λέξης, συμπεριλαμβανομένων και αυτών της ισοτιμίας Είναι σημαντικό να αναφερθεί ότι αυτή η ιδιότητα μπορεί να εντοπίσει παραπάνω από δύο σφάλματα σε πολλές περιπτώσεις, αλλά δεν εγγυάται τον εντοπισμό όλων των πιθανών συνδυασμών διπλών σφαλμάτων κάθε φορά. Ας παρατηρήσουμε την περίπτωση ενός διπλού λάθους που ακολουθεί:

$$\vec{s}=\begin{bmatrix}1&0&1&1&0&1&1\end{bmatrix}\times\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\1&1&0&0\\1&0&1&0\\0&1&1&0\\1&1&1&0\end{bmatrix}=\begin{bmatrix}0&0&1&1\end{bmatrix},\quad \begin{matrix}C=p1+p2+p4\\P=p8\end{matrix}$$

Συμπερασματικά, πρέπει να διευκρινιστεί ότι τα πρώτα τρία ψηφία του διανύσματος αποκωδικοποίησης είναι υπεύθυνα για κάθε ψηφίο ενώ το τέταρτο για

ολόκληρη την πληροφορία. Βρίσκουμε λοιπόν ότι η πληροφορία έχει διπλό λάθος στο p4 και άλλο ένα στο p8. Όμως μόνο το π4 είναι διορθώσιμο. Έστω ότι η πληροφορία έπειτα από παρεμβολή θορύβου στο κανάλι επικοινωνίας έχει υποστεί αλλοίωση, υπάρχουν τέσσερις περιπτώσεις οι οποίες μπορεί να συμβούν:

- Αν C=0 και P=0 , τότε η πληροφορία είναι έγκυρη.
- Αν C≠0 και P=0 , τότε έχουμε σφάλμα σε ένα ψηφίο, που μπορεί να διορθωθεί.
- Αν C≠0 και P=1 , τότε έχει συμβεί ένα διπλό σφάλμα, το οποίο είναι εντοπίσημο.
- Αν C=0 και P=1 , τότε η το ψηφίο ισοτιμίας της λέξης έχει σφάλμα.

Η επιδιόρθωση γίνεται με τον ακριβώς ίδιο τρόπο, δηλαδή με την αναζήτηση σε πίνακα σφαλμάτων και την δημιουργία του διανύσματος σφάλματος. Από τον πίνακα είναι φανερό ότι η πληροφορία θα προστεθεί με το διάνυσμα σφάλματος 00100000 και θα προκύψει η 1001011. Που υποδεικνύει ότι υπάρχει ακόμη ένα λάθος προς επιδιόρθωση το οποίο είναι αόρατο, λόγω της ισοτιμίας που θα είναι ζυγού αριθμού.

Τα τελευταίας γενιάς τσιπ μνήμης υψηλής πυκνότητας δημιουργούν νέα προβλήματα αξιοπιστίας. Χαρακτηριστικά παραδείγματα είναι τα σφάλματα λογισμικού που προκαλούνται από σωματίδια άλφα και νετρόνια, που προκαλούνται από κοσμικές ακτίνες των υψηλής πυκνότητας RAM chips. Αυτά τα λάθη μπορεί να είναι ισοσκελιστούν με υπάρχοντα λάθη υλικού, προκαλώντας πολλαπλά σφάλματα που δεν μπορούν να διορθωθούν με απλούς κωδικούς SECDED.

Διάνυσμα σφάλματος	**Διάνυσμα αποκωδικοποίησης**
10000000	1000
01000000	0100
00100000	0010
00010000	0001
00001000	1100
00000100	1010
00000010	0110
00000001	1110

Πίνακας 3.7 Μέθοδος αποκωδικοποίησης κωδικών SECDED.

Για να λυθούν αυτά τα προβλήματα, έχουν αναπτυχθεί με το πέρασμα των ετών αρκετοί τρόποι αντιμετώπισης. Μερικοί από αυτούς είναι η τεχνική της *ανάγνωσης με επανάκτηση,* στην οποία τα λάθη λογισμικού εξαφανίζονται κατά τη

διάρκεια του επόμενου κύκλου ανάγνωσης, *η τεχνική εφεδρείας* , η οποία αντικαθιστά ένα ελαττωματικό εξάρτημα με ένα εφεδρικό χωρίς απαιτείται ανθρώπινη παρέμβαση και την τεχνική διόρθωσης επικάλυψης σφαλμάτων , η οποία απαιτεί κάποιες πρόσθετες λειτουργίες για τον εντοπισμό και τη διόρθωση των λαθών υλικού. Η μέθοδος της επαναπροσπάθειας απεικονίζεται στο ακόλουθο παράδειγμα :

1 0 0 0 0 0 1 1	Έγκυρη πληροφορία
1 0 h a 0 0 1 1	Με σφάλμα λογισμικού a σφάλμα υλικού h
1 0 1 1 0 0 1 1	Ανάγνωση από μνήμη
0 1 0 0 1 1 0 0	Συμπλήρωμα και εγγραφή στη μνήμη
0 1 0 1 1 1 0 0	Διόρθωση σφάλματος a
1 0 1 0 0 0 1 1	Ανάγνωση
0 1 0 1 1 1 0 0	Συμπλήρωμα και εγγραφή στη μνήμη
0 1 1 1 1 1 0 0	Ανάγνωση με επαναπροσπάθεια
1 0 0 0 0 0 1 1	Επιδιορθωμένη πληροφορία

Πίνακας 3.8 Παράδειγμα αποκωδικοποίησης κωδικών SECDED.

Παρατηρούμε λοιπόν ότι μία από τις παραπάνω ιδιότητες συνδυαστικά με τους κώδικες SECDED, έχουν την δυνατότητα να επιδιορθώσουν παραπάνω του ενός σφάλματα με τον συνδυασμό περιττής είτε άρτιας ισοτιμίας για την κατασκευή πιο περίπλοκων κωδικών.

Ανακεφαλαιώνοντας, Η μεθοδολογία των κωδικών SECDED βρίσκει εφαρμογή σε συστήματα πραγματικού χρόνου καθώς και στην ανάγνωση/εγγραφή στις μνήμες RAM και στην διαδικτυακή επικοινωνία υπολογιστών. Αυτός ο παραλλαγμένος κώδικας χρησιμοποιεί ένα διαφορετικό σύστημα ελέγχου ισοτιμίας bit, που εξισορροπεί τον αριθμό εισόδων με κάθε bit ελέγχου, ως εκ τούτου τον αριθμό των εισόδων σε κάθε κύκλωμα που κάνει τον έλεγχο. Η εξισορρόπηση επιπλέον ελαχιστοποιεί την καθυστέρηση εξαιτίας του λάθους στα κυκλώματα διόρθωσης και ανίχνευσης, με αποτέλεσμα την ταχύτερη απόκριση του συστήματος υπό την παρουσία σφαλμάτων.

3.8 Κυκλικοί κώδικες

Είναι γραμμικοί κώδικες με μια επιπλέον ιδιότητα, η περιστροφή οποιασδήποτε έγκυρης λέξης του κώδικα παράγει έγκυρη λέξη που και αυτή ανήκει στον κώδικα. Η περιστροφή μπορεί να είναι είτε δεξιόστροφη είτε αριστερόστροφη ανάλογα με την εφαρμογή .Οι κώδικες αυτοί κατασκευάζονται με την ίδια λογική όπως οι Hamming[7]. Για παράδειγμα ο κώδικας (00000, 1101000, 0110100,

0010110, 0001011) είναι έγκυρος κυκλικός κώδικας καθώς το κριτήριο περιστροφής συνεχίζει και διατηρεί την γραμμικότητά του.

Περιλαμβάνουν τον πίνακα γεννήτριας G με την μόνη διαφορά ότι κάθε στήλη εκφράζει ένα διάνυσμα. Επιπλέον αν οι στήλες του πίνακα G είναι περιστροφές ενός συγκεκριμένου διανύσματος τότε οποιοσδήποτε συνδιαδμός των στηλών αυτών μπορεί να εκφραστεί ως συνδιασμός περιστροφών στηλών.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g(x)} \\ \mathbf{xg(x)} \\ \mathbf{x^2 g(x)} \\ \mathbf{x^3 g(x)} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}.$$

Η βασική ιδέα των κωδικών στηρίζεται στην αναπαράσταση των δυαδικών διανυσμάτων σε πολυωνυμική μορφή. Η αναπαράσταση είναι εφικτή για κάθε διάνυσμα όταν ο βαθμός του πολυωνύμου είναι μικρότερος από τον αριθμό ψηφίων του διανύσματος κατά ένα Το MSB λαμβάνει τον υψηλότερο βαθμό ενώ το LSB το μηδέν. Για να γίνει κατανοητός ο τρόπος αυτός δίνονται οι λέξεις A=11001 και B=00111. Ισχύουν τα εξής,

$$A \cdot B = \left(x^4 + x^3 + 1\right) \cdot \left(x^2 + x + 1\right) = x^6 + x^3 + x^2 + x + 1$$

$$A \div B = \left(x^4 + x^3 + 1\right) \div \left(x^2 + x + 1\right) \Rightarrow A = \left(x^2 + 1\right) B + x$$

Έτσι λοιπόν μπορούμε να προσθέσουμε πλεονασμό στο αρχικό διάνυσμα της πληροφορίας του οποίου οι ρίζες είναι οι g0,g1,g2....g(n-k).Το G αυτό ονομάζεται πολυώνυμο γεννήτορας και έχει βαθμό **r=k-n** Το γινόμενο αντιπροσωπεύει την πληροφορία και έχει μήκος ακριβώς n bits και το πολυώνυμο του έχει βαθμό r= n-1. Επομένως η διαδικασία της κωδικοποίησης ενός μηνύματος m(x) σε codeword c(x) γίνεται με πολλαπλασιασμό του μηνύματος με το πολυώνυμο γεννήτρια g(x),

$$c(x) = m(x) \cdot g(x)$$

Για την αποκωδικοποίηση πραγματοποιείται η αντίστροφη διαδικασία, η διαίρεση με το πολυώνυμο g(x),

$$m'(x) = c(x) \div g(x)$$

Για να αποφευχθούν τα σφάλματα στην διαδικασία της κωδικοποίησης είναι αναγκαία μια μέθοδος επαλήθευσης. Ο τρόπος αυτός είναι με τον πολλαπλασιασμό modulo και έχει ως αποτέλεσμα έγκυρη λέξη.

$$c(x) \cdot h(x) \bmod \left(x^n + 1\right) = m(x) \cdot g(x) \cdot h(x) \bmod \left(x^n + 1\right)$$
$$\rightarrow m(x) \cdot \left(x^n + 1\right) \bmod \left(x^n + 1\right) = 0$$

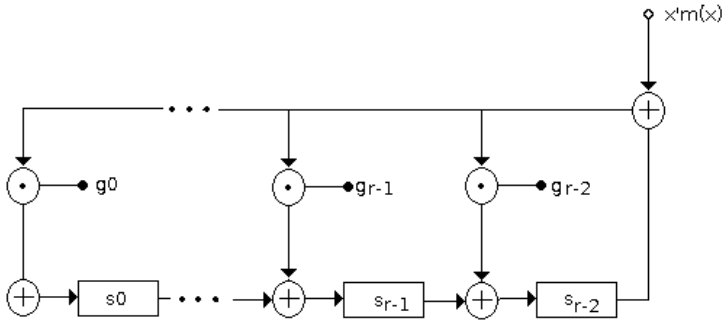
Ωστόσο μερικές φορές είναι πιθανό να συμβούν σφάλματα τότε το εκφράζουμε το κωδικοποιημένο μήνυμα ως c'(x)=c(x)+e(x), πιο αναλυτικά έχουμε:

$$c'(x) \cdot h(x) \bmod \left(x^n + 1\right) = c(x) \cdot h(x) \bmod \left(x^n + 1\right) + e(x) \cdot h(x) \bmod \left(x^n + 1\right)$$
$$\rightarrow e(x) \cdot h(x) \bmod \left(x^n + 1\right)$$

Η σημασία του παραπάνω αποτελέσματος είναι η εξάρτηση του codeword μόνο από το σφάλμα. Άρα μπορεί να χρησιμοποιηθεί ως γινόμενο για επιδιόρθωση λαθών. Το γινόμενο αξιοποιείται με την σχετική αναζήτηση του στον πίνακα σφαλμάτων για την παραγωγή του πολυώνυμου σφάλματος. Η παραπάνω μέθοδος είναι γενική και μπορεί να χρησιμοποιηθεί σε οποιοδήποτε κυκλικό κώδικα.

Μια χρήσιμη κατηγορία αποτελούν οι διαχωρίσιμοι συστημικοί κυκλικοί κώδικες. Η λειτουργία τους περιγράφεται με την προσθήκη επιπλέον ψηφίων ισοτιμίας βάσει κάποιου κανόνα, ώστε να διατηρείται το χαρακτηριστικό της περιστροφής. Ένας τέτοιος κώδικας κατασκευάζεται με την προσθήκη των ψηφίων ισοτιμίας στο τέλος του μηνύματος σε αντίθεση με προηγουμένως και περιγράφεται από την εξίσωση $c(x) = x' m(x) \cdot x' m(x) mod g(x)$

Ο πρώτος όρος αποτελεί τον αριθμό των ολισθήσεων του μηνύματος ώστε να συμβαδίζει με τα ψηφία της ισοτιμίας. Ο δεύτερος όρος περιγράφει τα ψηφία ισοτιμίας και συμβολίζεται με x'm(x)mod*g(x)*.Πρέπει να σημειωθεί ότι η μέθοδος κωδικοποίησης διαφέρει από τις άλλες μεθόδους κατά τον πολλαπλασιασμό με το γινόμενο.



Εικόνα 3.2 Κύκλωμα κυκλικής κωδικοποίησης.

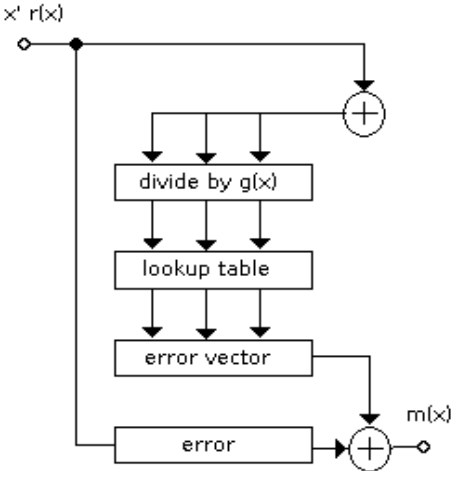
Στο κύκλωμα αυτό το ολισθημένο μήνυμα εισάγεται σταδιακά. Αν εισέλθει πρώτα το MSB, τότε μετά από k βήματα θα η υπόλοιπη πληροφορία θα παρθεί από τους ενδιάμεσους μανταλωτές. Το κωδικοποιημένο μήνυμα εξάγεται με την τοποθέτηση των ψηφίων ισοτιμίας στο τέλος του αρχικού μηνύματος, έτσι λοιπόν το γινόμενο αποκωδικοποίησης υπολογίζεται με την διαίρεση του ολισθημένου μηνύματος με το πολυώνυμο g(x).Μαθηματικά αυτό γράφεται ως εξής:

$$x^r c(x) mod g(x) = \left[x^r mod g(x)\right] \cdot \left[x^r m(x) + x^r m(x) mod g(x)\right] = 0$$

Για λανθασμένη λέξη θα έχουμε:

$$s(x) = x^r c'(x) mod g(x) = x^r (c(x) + e(x)) mod g(x) = x^r e(x) mod g(x)$$

Παρατηρούμε ότι το γινόμενο s(x) είναι εξαρτώμενο μόνο από το e(x), όπως στην περίπτωση των μη διαχωρίσιμων κωδικών. Ουσιαστικά κάθε σφάλμα θα έχει ένα γινόμενο που θα συσχετίζεται μοναδικά με αυτό. Μια πιθανή υλοποίηση του κυκλώματος αποκωδικοποίησης με πίνακα αναζήτησης σφάλματος αποτελεί το παρακάτω κύκλωμα που αποτελεί την βασική ιδέα και για πολλούς άλλους κώδικες.



Εικόνα 3.3: Κύκλωμα κυκλικής αποκωδικοποίησης.

3.9 CRC codes

Οι κώδικες αυτοί χρησιμοποιούνται για τον εντοπισμό σφαλμάτων και όχι για την επιδιόρθωση. Είναι κυκλικοί κώδικες και χρησιμοποιούνται συμπληρωματικά με κυκλώματα για να επιτύχουν ανοχή σε σφάλματα. Αξιοποιούνται για την διόρθωση ατελειών του αποκωδικοποιητή των κυκλικών κωδικών. Είναι ευρέως διαδεδομένα στα μικροϋπολογιστικά συστήματα, αρχικά για την εύρεση λαθών σε δίσκους floppy, μετέπειτα στα CD και στους σκληρούς δίσκους. Η επιλογή ενός συγκεκριμένου κώδικα CRC για μια εφαρμογή πρέπει να συμβαδίζει με τον κυκλικό κώδικα που θα χρησιμοποιηθεί σε αυτή[4,5]. Αυτό σημαίνει ότι το κριτήριο είναι το πολυώνυμο γεννήτριας g(x). Εδώ το ανάλογο πολυώνυμο ανήκει σε μια κατηγορία, που ονομάζονται πρωτεύοντα πολυώνυμα.

Μια CRC συσκευή υπολογίζει μια σύντομη, σταθερού μήκους δυαδική ακολουθία, που είναι γνωστή ως τιμή ελέγχου ή CRC, για κάθε μπλοκ δεδομένων που πρέπει να αποσταλούν ή να αποθηκευθούν και προσαρτάτε στα δεδομένα κατά την κωδικοποίηση. Όταν μια κωδικολέξη ληφθεί, η συσκευή είτε συγκρίνει την τιμή ελέγχου με μια πιο πρόσφατη που υπολογίστηκε από το μπλοκ δεδομένων, ή ισοδύναμα, εκτελεί μια CRC σε ολόκληρη την κωδικολέξη και συγκρίνει την προκύπτουσα τιμή ελέγχου με μια αναμενόμενη σταθερά υπολοίπου, από την διαίρεση με το πολυωνύμου g(x). Αν οι τιμές ελέγχου δεν ταιριάζουν, τότε το μπλοκ περιέχει ένα σφάλμα δεδομένων. Η συσκευή μπορεί να λάβει διορθωτικά μέτρα, όπως η εκ νέου ανάγνωση του μπλοκ ή να ζητηθεί να σταλεί ξανά. Διαφορετικά, τα δεδομένα είναι χωρίς λάθη, αν και πάντα υπάρχει μια μικρή πιθανότητα να περιέχει απαρατήρητα σφάλματα.

Ας ορίσουμε αρχικά την έννοια του πρωτεύοντος πολυωνύμου. Πρωτεύον λέγεται το πολυώνυμο g(x) το οποίο έχει βαθμό r-1 και επιπλέον ικανοποιεί τις παρακάτω συνθήκες:

$$g(x)\cdot mod\ \left(x^{(2^r-1)}\right)\ \kappa\alpha\iota\ \ g(x)\cdot mod\ \left(x^m-1\right)\neq\ 0\ \ ,m<2^{(r-1)}-1$$

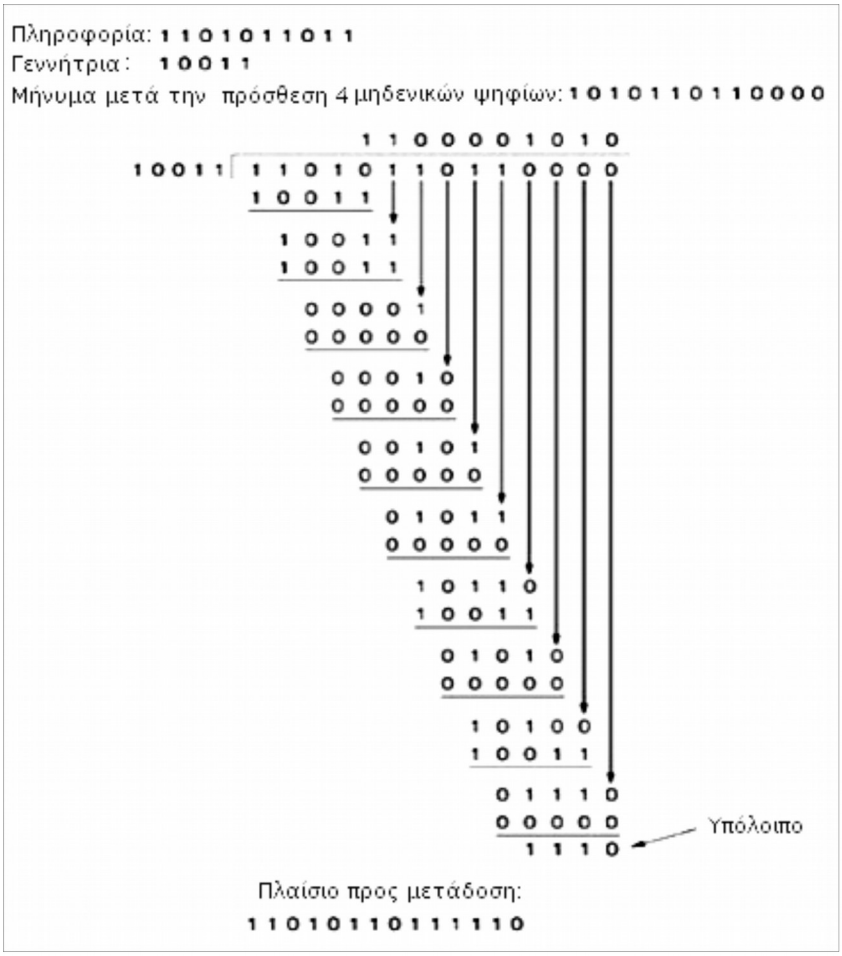
Το πλεονέκτημα της επιλογής ενός πρωτεύοντος πολυώνυμου ως γεννήτρια για έναν κώδικα CRC είναι ότι ο κώδικας που θα προκύψει έχει μέγιστο μήκος μπλοκ, με την έννοια ότι όλα τα σφάλματα 1-bit εντός του μήκους μπλοκ έχουν διαφορετικά υπόλοιπα, που ονομάζονται σύνδρομα s. Επομένως, δεδομένου ότι το υπόλοιπο είναι μια γραμμική συνάρτηση του μπλοκ, ο κώδικας μπορεί να ανιχνεύσει όλα τα σφάλματα 2-bit εντός του μήκους μπλοκ. Εφόσον το r είναι ο βαθμός του πρωτεύοντος πολυώνυμο γεννήτρια, τότε το μέγιστο συνολικό μήκος της λέξης είναι 2 ^r – 1, και ο σχετικός κώδικας είναι σε θέση να ανιχνεύσει σφάλματα ενός bit ή δύο-bit.Μπορούμε να βελτιώσουμε αυτή τη κατάσταση,άν χρησιμοποιούμε το πολυώνυμο γεννήτρια g (x) = ρ (x) (1 + x), όπου ρ (x) είναι ένα πρωτεύον πολυώνυμο βαθμού r μείον ένα. Τότε το μέγιστο συνολικό μήκος μπλοκ είναι 2 ^ {r – 1} – 1, και ο κωδικός είναι σε θέση να ανιχνεύσει μονά, διπλά, τριπλά και κάθε μονό αριθμό σφαλμάτων.

Έτσι λοιπόν παρατηρούμε ότι με την κατάλληλη επιλογή ενός πολυωνύμου μεταβάλουμε τις δυνατότητες εντοπισμού ενός κυκλικού κώδικα. Παρακάτω απεικονίζεται η διαδικασία της κωδικοποίησης, καθώς και μερικά χαρακτηριστικά πολυώνυμα.

Όνομα	Πολυώνυμο
CRC-8	x^8+x^2+x+1
CRC-10	$x^{10}+x^9+x^5+x^4+x^2+1$
CRC-16	$x^{16}+x^{12}+x^5+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Πίνακας 3.9 Πολυώνυμα ορισμένων κωδικών CRC.

Η μεθοδολογία αναλυτικά είναι η εξής:



Εικόνα 3.4 CRC Κωδικοποίηση του παραδείγματος.

Τέλος πρέπει να αναφέρουμε τις εφαρμογές τους. Αξιοποιούνται στα σύγχρονα συστήματα τηλεπικοινωνιών, όπως στην ενσύρματη και ασύρματη κινητή τηλεφωνία, στα δίκτυα υπολογιστών και στα βιομηχανικά συστήματα διεργασιών.

3.10 Convolution Codes

Στον τομέα των τηλεπικοινωνιών, ένας συνελικτικός κώδικας είναι ένας τύπος κώδικα διόρθωσης σφαλμάτων που δημιουργεί σύμβολα ισοτιμίας μέσω της ολισθαίνουσας εφαρμογής ενός πολυωνύμου, σε ένα ρεύμα δεδομένων. Οι συνελικτικοί κώδικες μοιάζουν με τους μπλοκ κώδικες στο ότι περιλαμβάνουν τη μετάδοση των bits ισοτιμίας, που υπολογίζονται από τα ψηφία του μηνύματος. Σε αντίθεση με τους κωδικούς μπλοκ σε συστηματική μορφή, ο αποστολέας δεν στέλνει το μήνυμα ακολουθόμενο από τα δυαδικά ψηφία ισοτιμίας, ο αποστολέας στέλνει μόνο τα bits ισοτιμίας.

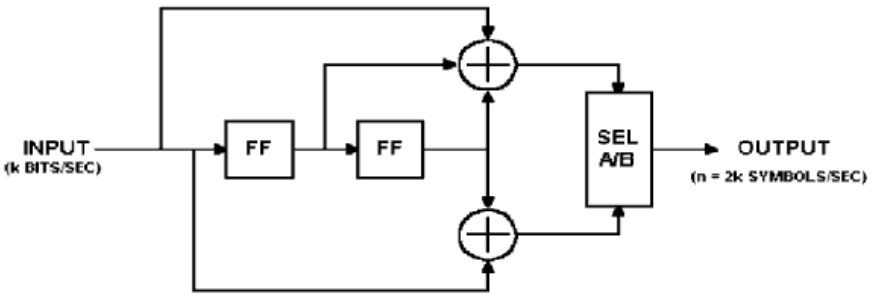
Η εφαρμογή της ολίσθησης αντιπροσωπεύει την «συνέλιξη» του κωδικοποιητή επί των δεδομένων. Η φύση των συνελικτικών κωδίκων διευκολύνει την βαθμιδωτή αποκωδικοποίηση με πλαίσιο σταθερού μεγέθους. Ο αποκωδικοποιητής με τη σειρά του, επιτρέπει τη μέγιστη πιθανότητα ελέγχου, με την βοήθεια λογισμικού του συνελικτικού κώδικα. Η ικανότητά του να εκτελεί οικονομικά την μέγιστη πιθανότητα αποκωδικοποίησης με την βοήθεια λογισμικού είναι ένα από τα σημαντικότερα οφέλη των συνελικτικών κωδικών. Αυτό έρχεται σε αντίθεση με τους κλασσικούς κώδικες μπλοκ οι οποίοι λαμβάνουν αποφάσεις με το υλικό κατά την αποκωδικοποίηση.

Οι συνελικτικοί κώδικες συχνά χαρακτηρίζονται από τον ρυθμό κώδικα και τη μνήμη του κωδικοποιητή. Ο ρυθμός κώδικα τυπικά δίνεται n / k, όπου n είναι ο ρυθμός δεδομένων εισόδου και k είναι ο ρυθμός συμβόλων εξόδου. Η μνήμη συχνά αποκαλείται και μήκος περιορισμού «K», όπου η έξοδος είναι συνάρτηση των προηγούμενων K-1 εισόδων. Η μνήμη δίνεται ως ο αριθμός των στοιχείων 'n' στο πολυώνυμο ή τον μέγιστο δυνατό αριθμό των καταστάσεων του κωδικοποιητή, συνήθως 2^n. Οι συνελικτικοί κώδικες συχνά περιγράφονται ως συνεχείς, έτσι μπορούμε να πούμε ότι έχουν αυθαίρετο μήκος μπλοκ, δεδομένου ότι οι συνελικτικός κωδικοποιήσεις εκτελούνται σε μπλοκ δεδομένων. Αναλυτικά με gi το K στοιχείο μνήμης που έχει πολυώνυμο γεννήτριας για κάθε bit ισοτιμίας pi . Μπορούμε στη συνέχεια γράψουμε το pi ως εξής :

$$p_i[n] = (\sum_{j=0}^{k-1} g_i[j]x[n-j]) \bmod 2.$$

Η μορφή της παραπάνω εξίσωσης είναι μια συνέλιξη του πολυωνύμου g και του μηνύματος x. Ο αριθμός των πολυωνύμων γεννήτριας είναι ίσος με τον αριθμό των παραγόμενων ψηφίων ισοτιμίας r, σε κάθε ολίσθηση. Έτσι λοιπόν προκύπτουν οι παρακάτω εξισώσεις για ρυθμό κώδικα 1/2 και K=3.

$$\begin{aligned} p_0[n] &= x[n] + x[n-1] + x[n-2] \\ p_1[n] &= x[n] + x[n-1] \end{aligned}$$



Εικόνα 3.5: Κύκλωμα συνελικτικής κωδικοποίησης.

Για να κωδικοποιούν συνελικτικά τα δεδομένα, ξεκινάμε με καταχωρητές μνήμης που συμβολίζονται με k, που το καθένα διαχειρίζεται 1 bit εισόδου. Εκτός αν ορίζεται διαφορετικά, όλοι οι καταχωρητές μνήμης ξεκινούν με αξία 0. Ο κωδικοποιητής έχει n αθροιστές modulo-2 . Ένας αθροιστής μπορεί να υλοποιηθεί με μία απλή πύλη XOR και n πολυώνυμα γεννήτριας, ένα για κάθε αθροιστή. Ένα bit εισόδου τροφοδοτείται στο αριστερότερο μητρώο. Χρησιμοποιώντας τα πολυώνυμα και τις υπάρχουσες τιμές στα υπόλοιπα μητρώα, ο κωδικοποιητής εξάγει n σύμβολα. Αυτά τα σύμβολα μεταδίδονται ή μεταβάλλονται, ανάλογα με τον επιθυμητό ρυθμό κώδικα. Έπειτα γίνεται μετατόπιση όλων των τιμών του μητρώου προς τα δεξιά και αναμένει για το επόμενο bit εισόδου. Εάν δεν υπάρχουν άλλα bits εισόδου, ο κωδικοποιητής συνεχίζει να μετατοπίζεται έως ότου όλες οι καταχωρήσεις έχουν επιστρέψει στην αρχική τους κατάσταση. Έπειτα η διαδικασία αυτή επαναλαμβάνεται για το επόμενο τμήμα του μηνύματος.

Η ικανότητα διόρθωσης t από ένα συνελικτικό κώδικα είναι ο αριθμός των λαθών που μπορεί να διορθωθούν από τον κώδικα . Μπορεί να υπολογιστεί ως t = (d - 1) / 2 . Δεδομένου ότι ένας τέτοιος κώδικας δεν χρησιμοποιεί μπλοκ , επεξεργάζεται μια συνεχής ροή από ψηφία , η τιμή του t εφαρμόζεται σε ένα τμήμα των σφαλμάτων που βρίσκονται σχετικά κοντά το ένα στο άλλο Δηλαδή, πολλές ομάδες από λάθη έχουν την δυνατότητα να καθοριστούν, όταν είναι σχετικά μεγάλη η απόσταση μεταξύ των συμβάντων τους.

Το γεγονός ότι τα σφάλματα εμφανίζονται με την μορφή ομάδων, που είναι γνωστά στην ξένη ορολογία ως burst errors, είναι μια ιδιότητα που θα πρέπει να λαμβάνεται υπόψη κατά το σχεδιασμό ενός συνεχούς κωδικοποιητή με εσωτερικό συνελικτικό κώδικα. Η πιο δημοφιλής λύση για αυτό το πρόβλημα σχεδιασμού είναι να παρεμβληθούν τα δεδομένα στην επεξεργασία ενός εξωτερικού μπλοκ κώδικα, συνήθως ενός Reed-Solomon που θα εξηγηθεί στην επόμενη ενότητα, πριν τη συνελικτική κωδικοποίηση , έτσι ώστε να διορθωθούν τα περισσότερα από τα σφάλματα.

3.11 Reed-Solomon code

Ορισμένες οικογένειες κωδικών, όπως οι Reed-Solomon, λειτουργούν με αλφάβητα μεγαλύτερα από το δυαδικό[1]. Αυτή η ιδιότητα δημιουργεί κωδικούς με ισχυρές δυνατότητες διόρθωσης σφαλμάτων. Σκεφτείτε έναν κωδικό που λειτουργεί στο πεδίο F2^m . Κάθε σύμβολο του αλφαβήτου μπορεί να αντιπροσωπεύεται από m

bits. Αν C είναι ένας [n, k] κώδικας Reed-Solomon πάνω στο F(2 ^ m), μπορούμε να σκεφτούμε C(nk, mk) κώδικα πάνω σε GF(2).

Οι κώδικες RS είναι ισχυροί για διόρθωση burst errors. Κάθε σύμβολο εκπροσωπείται από m bits, και σε γενικές γραμμές, δεν έχει σημασία πόσα από αυτά είναι λανθασμένα, είτε ένα ενιαίο κομμάτι, ή ένα σύνολο των m bits περιέχουν λάθη, από τη σκοπιά της αποκωδικοποίησης εξακολουθεί να είναι ένα μονό σφάλμα. Με άλλα λόγια, δεδομένου ότι τα σφάλματα ριπής τείνουν να εμφανίζονται σε ομάδες, υπάρχει μια ισχυρή πιθανότητα πολλών δυαδικών σφαλμάτων συμβάλλουν σε ένα μόνο σφάλμα.

Παρατηρήστε ότι μια ριπή από (m + 1) λάθη μπορεί να επηρεάσει σε τουλάχιστον 2 χαρακτήρες, και μια ριπή από 2m + 1, μπορεί να επηρεάσει τουλάχιστον 3 σύμβολα. Στη συνέχεια, μια ριπή από tm + 1 μπορεί να επηρεάσει σε t + 1 σύμβολα Αυτό σημαίνει ότι ένας κώδικας αποτελούμενος από t σύμβολα μπορεί να διορθώσει μια έκρηξη του μήκους το πολύ (t-1) m + 1.

Reed-Solomon code	
Μήκος κώδικα	n = q-1
Μήκος Μηνύματος	k
Απόσταση	d=n-k+1
Αλφάβητο	q=p (prime p)
Συμβολισμός	[n, k, n-k+1]

Πίνακας 3.10 Βασικές ιδιότητες κωδικών RS.

Σχετικά με την μεθοδολογία αυτού του κώδικα κάθε κωδικοποιημένη λέξη της πληροφορίας εκφράζεται ως υποσύνολο από αξίες με την μορφή ακολουθίας. Η ακολουθία αυτή προκύπτει από την ένα πολυώνυμο χαμηλότερου βαθμού. Ειδικότερα, προκειμένου να ληφθεί μία λέξη του κώδικα Reed-Solomon, το μήνυμα ερμηνεύεται ως η περιγραφή του ενός πολυωνύμου p βαθμού μικρότερου από k επί του πεπερασμένου πεδίου F με στοιχεία q. Με τη σειρά του, το πολυώνυμο ρ αξιολογείται σε n διακριτά σημεία , a1... an του πεδίου F, και η αλληλουχία των τιμών είναι η αντίστοιχη κωδική λέξη. Επισήμως, η σειρά C κωδικών λέξεων του κώδικα Reed-Solomon ορίζεται παρακάτω.

Το μήνυμα συμβολίζεται με x και πολλαπλασιάζεται με τον πίνακα γεννήτριας A για να προκύψει η κωδικοποιημένη ακολουθία.

$$p_x(a) = \sum_{i=1}^{k} x_i a^{(i-1)}$$

$$C = \left(p_x(a_1),\ p_x(a_2)\ .\ .\ .\ p_x(a_n) \right) \qquad A = \begin{bmatrix} 1 & \dots & 1 \\ a_1 & \dots & a_n \\ a_1^2 & \dots & a_n^2 \\ . & \dots & . \\ . & \dots & . \\ . & \dots & . \\ a_1^{(k-1)} & \dots & a_n^{(k-1)} \end{bmatrix}$$

Το σύνδρομο αποκωδικοποίησης υπολογίζεται με την διαίρεση του με ένα πολυώνυμο g(x). Επομένως ισχύουν τα ακόλουθα:

$$s(x) = \sum_{i=0}^{n-1} c_i x^i \qquad g(x) = \prod_{j=1}^{n-k} x - a^j$$

$$s(a^i) = 0 \quad , \quad i = 1,2,\ldots,n-k$$

Αν υπάρχει σφάλμα στην πληροφορία τότε θα πρέπει να χρησιμοποιηθούν οι αντίστοιχοι τύποι σφάλματος:

$$r(x) = s(x) + e(x)$$

$$e(x) = \sum_{i=0}^{n-1} e_i x^i \Leftrightarrow e(x) = \sum_{k=1}^{v} e_{(i_k)} x^{(i_k)}$$

Ο στόχος του αποκωδικοποιητή είναι να βρεθεί ο αριθμός των λαθών (ν), τις θέσεις των σφαλμάτων (ik), και οι τιμές σφάλματος σε αυτές τις θέσεις (eik). Από εκείνα, e (x) μπορεί να υπολογισθεί και αφαιρεθεί από το r (x) για να πάρει το αρχικό μήνυμα s(x).

Οι Reed και Solomon περιέγραψαν έναν θεωρητικό αποκωδικοποιητή, που διορθώνει τα λάθη από την εύρεση του πιο δημοφιλούς πολυωνύμου μηνύματος. Ο αποκωδικοποιητής για RS(n, k) κώδικα θα εξετάσει όλα τα πιθανά υποσύνολα k συμβόλων από το σύνολο των N συμβόλων που παραλήφθηκαν. Για τον κώδικα να είναι διορθώσιμος γενικά, πρέπει τουλάχιστον k σύμβολα να λαμβάνονται σωστά, ώστε να προσδιοριστεί το πολυώνυμο μηνύματος. Ο αποκωδικοποιητής θα εισάγει ένα πολυώνυμο μηνύματος για κάθε υποσύνολο, και θα καταγράφει ποια είναι τα υποψήφια ορθά πολυώνυμα . Το πιο δημοφιλές μήνυμα είναι το διορθωμένο αποτέλεσμα.

Όμως η μέθοδος αυτή είναι αναποτελεσματική καθώς θα πρέπει να ελεγχθούν όλες οι πιθανότητες των υποσυνόλων, που σημαίνει ότι απαιτείται πάρα πολύς χρόνος και ποσότητα πόρων για να επιτευχθεί η διαδικασία αυτή, για μεγάλους κώδικες. Έτσι έπρεπε να κατασκευαστεί ένας πιο πρακτικός αποκωδικοποιητής. Άξιοι αναφοράς είναι ο αποκωδικοποιητής του Peterson και των Berlekamp–Massey. Μια ακόμη υπολογιστική μέθοδο αποτελεί ο Ευκλείδειος αποκωδικοποιητής.

Τέλος, οι κώδικες Reed-Solomon έχουν εφαρμογές από την επικοινωνία στο βαθύ διάστημα μέχρι και σε ηλεκτρονικά είδη ευρείας κατανάλωσης. Χρησιμοποιούνται σε μεγάλη κλίμακα στα καταναλωτικά ηλεκτρονικά προϊόντα, όπως CD, DVD, δίσκους Blu-ray, στον τομέα των τεχνολογιών μετάδοσης δεδομένων, όπως DSL και WiMAX, τα συστήματα μετάδοσης, όπως DVB και ATSC αλλά και σε εφαρμογές πληροφορικής, όπως συστήματα RAID 6.

3.12 LDPC Codes

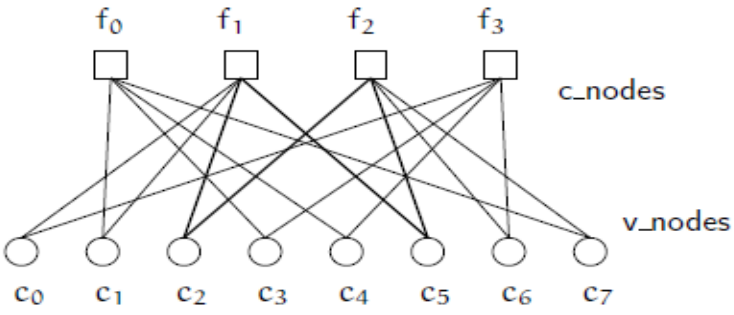
Οι χαμηλής πυκνότητας ελέγχου ισοτιμίας κώδικες (Low Density Parity Check) είναι μία κατηγορία των γραμμικών κωδίκων μπλοκ[1]. Το όνομα προέρχεται από το χαρακτηριστικό του πίνακα ελέγχου ισοτιμίας τους H, ο οποίος περιέχει μόνο μερικά 1 σε σύγκριση με την ποσότητα των μηδενικών του. Το κύριο πλεονέκτημά τους είναι ότι προσφέρουν σταθερές επιδόσεις πολύ κοντά στην χωρητικότητα πολλών διαφορετικών καναλιών και αξιοποιούν πολύπλοκους αλγόριθμους γραμμικού

χρόνου για την αποκωδικοποίηση. Επιπλέον, είναι οι κατάλληλοι για εφαρμογές που κάνουν βαριά χρήση παραλληλισμού.

Για την γραφική αναπαράστασή τους χρησιμοποιούν τα γραφήματα Tanner. Αυτά είναι διμερή γραφήματα. Αυτό σημαίνει ότι οι κόμβοι του γραφήματος χωρίζονται σε δύο σύνολα και οι ακμές συνδέουν μόνο κόμβους δύο διαφορετικών τύπων. Οι δύο τύποι κόμβων σε ένα διάγραμμα Tanner ονομάζονται κόμβοι μεταβλητών (v- nodes) και κόμβοι ελέγχου (c-nodes). Αυτοί χαρακτηρίζονται με βάση των αριθμό των άσσων μου περιέχει ο πίνακας ισοτιμίας H, σε κανονικής και ακανόνιστης μορφής. Αναλυτικότερα, ο πίνακας αυτός χαρακτηρίζεται από δύο μεγέθη, το Wr το οποίο είναι το πλήθος άσσων σε κάθε γραμμή και το Wc το οποίο είναι το πλήθος για κάθε στήλη.
Ένας πίνακας για να χαρακτηρισθεί ως χαμηλής πυκνότητας πρέπει να έχει Wr << n και Wc << m . Έτσι λοιπόν ο κώδικας είναι κανονικής μορφής αν η ποσότητα Wc είναι σταθερή για κάθε στήλη του H και Wr = Wc * (n/m) σταθερή για κάθε γραμμή. Παρακάτω δίνεται ο πίνακας ισοτιμίας κανονικής μορφής (Wr =2 ,Wc =4) καθώς και το γράφημα του.

Η κωδικοποίηση LDPC κωδίκων γίνεται με τον παρακάτω τρόπο. Πρώτα επιλέγουμε ορισμένους κόμβους μεταβλητών για να τοποθετήσετε το μήνυμα. Το δεύτερο βήμα αποτελεί ο υπολογισμός των τιμών των άλλων κόμβων. Μια προφανής λύση για αυτούς είναι οι εξισώσεις ελέγχου ισοτιμίας. Οι εργασίες αυτές αφορούν το σύνολο του. Η πολυπλοκότητα θα είναι και πάλι τετραγωνική στο μήκος μπλοκ. Στην πράξη, υπάρχουν πιο έξυπνο μέθοδοι για να εξασφαλίσουν κωδικοποίηση σε πολύ μικρότερο χρονικό διάστημα. Αυτές οι μέθοδοι μπορούν να χρησιμοποιήσουν την λιτότητα του πίνακα ελέγχου ισοτιμίας ή υπαγορεύουν μια ορισμένη δομή για το γράφημα Tanner.

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$



Εικόνα 3.6: Αναλυτικό διάγραμμα ενός LDPC κώδικα

Ο αλγόριθμος που χρησιμοποιείται για την αποκωδικοποίηση LDPC κωδίκων ανακαλύφθηκε ανεξάρτητα από ερευνητές αρκετές φορές και στην πραγματικότητα έχει και διαφορετικά ονόματα. Οι πιο κοινές από αυτές είναι ο αλγόριθμος belief propagation, message passing και ο sum-product. Για να γίνει εύκολα κατανοητός ο

τρόπος αυτός, θα αναλυθεί σε παρακάτω με τέσσερα βήματα, με μήνυμα το 10010101.

Στο πρώτο βήμα, όλοι οι ν-κόμβοι Ci στέλνουν ένα «μήνυμα» για τους C-κόμβους στους FJ, που περιέχει το κομμάτι του μηνύματος που πιστεύουν ότι να είναι η σωστή για αυτούς.

Στο δεύτερο στάδιο κάθε έλεγχος των κόμβων Fj θα παράξει μια έξοδο στο κάθε συνδεδεμένο κόμβο μεταβλητών. Το μήνυμα απάντησης περιέχει το ψηφίο που ο Fj πιστεύει ότι είναι η σωστή για αυτό το συγκεκριμένο v-node, υποθέτοντας ότι οι άλλοι v-κόμβοι που συνδέονται με το Fj είναι σωστές, όπως φαίνεται στο αμέσως επόμενο σχήμα.

c-node	received/sent			
v-node	y_i received	messages from check nodes		decision
c_0	1	$f_1 \rightarrow 0$	$f_3 \rightarrow 1$	1
c_1	1	$f_0 \rightarrow 0$	$f_1 \rightarrow 0$	0
c_2	0	$f_1 \rightarrow 1$	$f_2 \rightarrow 0$	0
c_3	1	$f_0 \rightarrow 0$	$f_3 \rightarrow 1$	1
c_4	0	$f_0 \rightarrow 1$	$f_3 \rightarrow 0$	0
c_5	1	$f_1 \rightarrow 0$	$f_2 \rightarrow 1$	1
c_6	0	$f_2 \rightarrow 0$	$f_3 \rightarrow 0$	0
c_7	1	$f_0 \rightarrow 1$	$f_2 \rightarrow 1$	1

Πίνακας 3.11 Παράδειγμα μετάδοσης της πληροφορίας μεταξύ κόμβων με κώδικα LDPC.

Η επόμενη φάση προϋποθέτει ότι όλοι οι ν-κόμβοι λαμβάνουν τα μηνύματα από τους κόμβους ελέγχου και χρησιμοποιούν αυτές τις πρόσθετες πληροφορίες για να αποφασίσουν εάν αρχικά τα ψηφία είναι ορθά. Ένας απλός τρόπος για να γίνει αυτό είναι η ψηφοφορία πλειοψηφίας.

Τέλος , εκτελείται η επαναληπτική τους φύση και επιστρέφουμε στο βήμα δύο για τον υπολογισμό του επόμενου μηνύματος. Οι κωδικοί χαμηλής πυκνότητας ελέγχου ισοτιμίας έχουν μελετηθεί πολύ τα τελευταία χρόνια και έχουν γίνει τεράστιες πρόοδοι στην κατανόηση και στην ικανότητα να σχεδιάζονται επαναληπτικά συστήματα κωδικοποίησης. Η επαναληπτική προσέγγιση αποκωδικοποίησης χρησιμοποιείται ήδη σε κώδικες turbo, οι οποίοι είναι προγενέστεροι, αλλά η δομή των κωδικών LDPC, μπορεί να δώσει ακόμα καλύτερα αποτελέσματα. Σε πολλές περιπτώσεις, επιτρέπουν υψηλότερο ποσοστό κώδικα και επίσης ένα χαμηλότερο κατώτατο όριο σφαλμάτων. Επιπλέον, καθιστούν δυνατή την εφαρμογή παραλληλοποιήσιμων αποκωδικοποιητών. Τα κύρια μειονεκτήματά τους είναι ότι οι κωδικοποιητές είναι κάπως πιο πολύπλοκοι συγκριτικά με άλλους και ότι το μήκος κώδικα πρέπει να είναι αρκετά μεγάλο για να αποδώσει ικανοποιητικά αποτελέσματα.

3.13 Automatic Request Query

Η Αυτόματη αίτηση επανάληψης (ARQ), επίσης γνωστή ως Αυτόματη Επανάληψη Ερωτήματος, είναι μια μέθοδος ελέγχου σφάλματος για τη μετάδοση των δεδομένων και χρησιμοποιεί αναγνωριστικά ψηφία, που είναι γνωστά ως

acknowledgement number ή ACK [6]. Τα ψηφία αυτά μας γνωστοποιούν ότι έχει λάβει σωστά ένα πλαίσιο δεδομένων με τη μορφή πακέτων, εντός των προκαθορισμένων χρονικών ορίων, δηλαδή τις συγκεκριμένες χρονικές περιόδους που μεσολαβούν πριν από την αναγνώριση λήψης, για να επιτευχθεί αξιόπιστη μετάδοση δεδομένων πάνω σε αναξιόπιστο κανάλι επικοινωνίας. Εάν ο αποστολέας δεν λάβει μια απόδειξη παραλαβής πριν από το χρονικό όριο, συνήθως μεταδίδει εκ νέου το πακέτο έως ότου λάβει μια επιβεβαίωση από τον παραλήπτη ή υπερβαίνει ένα προκαθορισμένο αριθμό από αναμεταδόσεις. Συνήθως, όταν ο πομπός δεν λάβει τη βεβαίωση μέσα σε εύλογο χρονικό διάστημα μετά την αποστολή του πλαισίου δεδομένων, αναμεταδίδει το πλαίσιο μέχρι να είναι σωστό αλλιώς το σφάλμα εξακολουθεί να υφίσταται πέρα από ένα προκαθορισμένο αριθμό των αναμεταδόσεων. Για την αντιμετώπιση τέτοιων φαινομένων δημιουργήθηκαν τρεις τύποι πρωτόκολλων ARQ: το Stop-and-wait, το Go-Back-N και το Selective Repeat.

3.13.1 Stop and wait

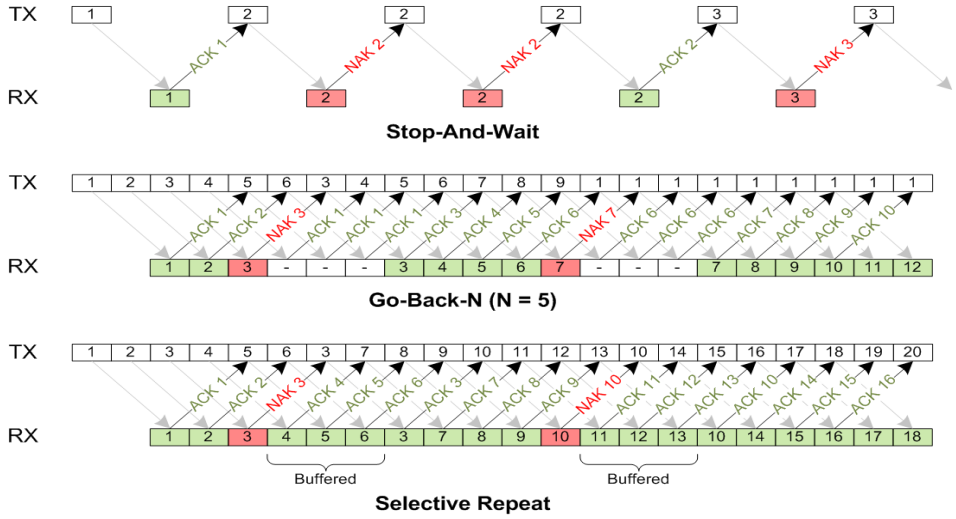
Για να αποφευχθούν αυτά τα προβλήματα, η πιο κοινή λύση είναι να καθοριστεί 1 bit σειράς της ακολουθίας στην κεφαλίδα του πλαισίου. Αυτός ο αριθμός ακολουθίας λαμβάνει τιμές συμπληρωματικά σε επόμενα πλαίσια. Όταν ο δέκτης στέλνει ένα ACK, που περιλαμβάνει τον αριθμό σειράς του επόμενου πακέτου που περιμένει. Με αυτό τον τρόπο, ο δέκτης μπορεί να ανιχνεύσει διπλά πλαίσια ελέγχοντας αν έχουν ίδιο αριθμό σειράς. Αν δύο επόμενα πλαίσια έχουν τον ίδιο αριθμό, είναι ίδια και το δεύτερο πλαίσιο απορρίπτεται. Ομοίως, αν τα δύο επόμενα ACKs αναφέρονται στον ίδιο αριθμό, αναγνωρίζεται το ίδιο πλαίσιο.
Το Stop-and-wait ARQ είναι αναποτελεσματικό σε σύγκριση με άλλες ARQs, επειδή ο χρόνος αποστολής των πακέτων, αν το ACK και τα δεδομένα έχουν ληφθεί επιτυχώς, είναι δύο φορές το χρόνο διέλευσης. Αυτό σημαίνει ότι ο χρόνος αποστολής είναι ίδιος με τον χρόνο λήψης του ACK.

3.13.2 Go-Back-N ARQ

Το Go-Back-N ARQ είναι ένα παράδειγμα αυτόματης αίτησης επανάληψης (ARQ) στο οποίο η διαδικασία μετάδοσης συνεχίζει με την αποστολή ενός αριθμού πλαισίων N, που καθορίζονται από το μέγεθος του παραθύρου, το οποίο είναι ένα όριο του καναλιού, μετρούμενο σε μονάδα μνήμης, ακόμη και χωρίς να λάβει επιβεβαίωση (ACK) του πακέτου από το δέκτη. Η διαδικασία είναι η εξής, ο δέκτης παρακολουθεί τον αριθμό ακολουθίας του επόμενου πλαισίου που περιμένει να λάβει, και στέλνει τον αριθμό αυτό με κάθε ACK. Ο δέκτης θα απορρίψει κάθε πλαίσιο που δεν έχει την ακριβή αλληλουχία αριθμού ακολουθίας, είτε είναι ένα αντίγραφο του πλαισίου που έχει ήδη αναγνωριστεί, ή ένα πλαίσιο εκτός σειράς που θα λάβει αργότερα και θα αποσταλεί εκ νέου ένα ACK για την τελευταίο επιτυχές πλαίσιο. Το Go-Back-N ARQ αποτελεί μια αποτελεσματικότερη λύση από το stop-and-wait ARQ, διότι περιμένει για μια αναγνώριση για κάθε πακέτο, ενώ η σύνδεση εξακολουθεί να χρησιμοποιείται ως αλληλοσυνδεόμενα πακέτα. Με άλλα λόγια, κατά τη διάρκεια του χρόνου που θα σπαταλούνταν σε αναμονή, θα μεταδίδονται περισσότερα πακέτα.

3.13.3 Selective Repeat

Όταν χρησιμοποιηθεί ως το πρωτόκολλο για την παράδοση των μηνυμάτων, η διαδικασία αποστολής εξακολουθεί να μεταδίδει ένα αριθμό των πλαισίων, ακόμη και μετά από μια απώλεια ενός πλαισίου. Σε αντίθεση με το Go-Back-N ARQ, κατά την διαδικασία λήψης ο παραλήπτης θα εξακολουθεί να δέχεται και να αναγνωρίζει τα πλαίσια που αποστέλλονται μετά από λάθος που έχει προκύψει.
Ο δέκτης παρακολουθεί τον αριθμό ακολουθίας του ως το πιο πρόσφατο πλαίσιο που δεν έχει λάβει, και στέλνει τον αριθμό αυτό με κάθε ACK. Εάν ένα πλαίσιο από τοναποστολέα δεν φτάσει στο δέκτη, ο αποστολέας συνεχίζει να στέλνει τα επόμενα πλαίσια έως ότου αδειάσει το παράθυρο του. Ο δέκτης συνεχίζει να γεμίσει το παράθυρο του με τα επόμενα πλαίσια, απαντώντας κάθε φορά με ένα ACK. Μόλις ο αποστολέας έχει στείλει όλα τα πλαίσια στο παράθυρο του, εκ νέου μεταδίδει τον αριθμό πλαισίου που επιλέγονται από τις επιβεβαιώσεις, και στη συνέχεια συνεχίζει από εκεί που σταμάτησε.



Εικόνα 3.7: Οι τεχνικές ARQ.

3.13.4 Hybrid ARQ

Η Υβριδική αυτόματη αίτηση επανάληψης, υβριδικό ARQ ή HARQ, είναι ένας συνδυασμός των υψηλού βαθμού κωδικών διόρθωσης σφαλμάτων με κωδικοποίηση προς τα εμπρός (FEC) και ARQ ελέγχου σφάλματος. Στο πρότυπο ARQ, τα πλεονάζοντα bits που προστίθενται στα δεδομένα πρέπει να διαβιβάζονται με τη χρήση ενός κώδικα ανίχνευσης σφάλματος EDC, όπως ένα κυκλικό έλεγχο πλεονασμού (CRC). Στην περίπτωση που ο παραλήπτης ανιχνεύσει ένα κατεστραμμένο μήνυμα θα ζητήσει ένα νέο μήνυμα από τον αποστολέα. Στο HARQ, τα αρχικά δεδομένα είναι κωδικοποιημένα με κώδικα διόρθωσης λάθους προς τα εμπρός (FEC), και τα bits ισοτιμίας είτε μεταδίδονται αμέσως μαζί με το μήνυμα ή διαβιβάζονται μόνο κατόπιν αιτήσεως, όταν ο παραλήπτης ανιχνεύει ένα εσφαλμένο

μήνυμα. Ο κωδικός EDC μπορεί να παραλειφθεί όταν ένας κώδικας έχει την δυνατότητα να εκτελέσει τόσο την προς τα εμπρός διόρθωση λάθους (FEC) και την ανίχνευση σφαλμάτων, τέτοιοι είναι οι κώδικες Reed-Solomon. Ένας FEC κώδικας επιλέγεται για να διορθώσει ένα μέρος του συνόλου των σφαλμάτων που μπορούν να προκύψουν, ενώ η μέθοδος ARQ χρησιμοποιείται ως εφεδρική για να διορθώσει τα σφάλματα που είναι δεν είναι διορθώσιμα, χρησιμοποιώντας μόνο τον πλεονασμό στην αρχική μετάδοση. Ως αποτέλεσμα αυτού, το υβριδικό ARQ αποδίδει καλύτερα από τα συνηθισμένα ARQ σε κακές συνθήκες σήματος, αλλά στην απλούστερη μορφή της, λειτουργεί αποδοτικότερα όταν υπάρχουν καλές συνθήκες σήματος. Υπάρχει συνήθως ένα σημείο όπου η ποιότητα σήματος έχει ένα όριο, κάτω από αυτό ένα απλό HARQ είναι το βέλτιστο για χρήση, ενώ πάνω από αυτό το βασικό ARQ είναι καλύτερο. Πρέπει να σημειωθεί ότι το HARQ διαχωρίζεται σε τρείς κατηγορίες, την τύπου I, τύπου II και τύπου III.

Η τύπου I αναμεταδίδει επανειλημμένα τα ίδια στοιχεία, και συνήθως απορρίπτει τα προηγουμένως ληφθέντα δεδομένα. Αυτό είναι αναποτελεσματικό, διότι ακόμη και αν υπάρχουν κάποια bits λάθος στα δεδομένα, η πληροφορία εξακολουθεί να περιέχει πολύτιμες πληροφορίες. Για το λόγο αυτό, ο τύπος II HARQ χρησιμοποιεί μια σειρά αναμετάδοσης, που περιέχει ως επί το πλείστον bits ισοτιμίας αντί της αρχικής ακολουθίας. Αυτά τα πλεονάζοντα bits σε συνδυασμό με τα προηγουμένως ληφθέντα δεδομένα θα δημιουργήσει ένα χαμηλότερου ρυθμού κώδικα, και όπως είδαμε πριν, όσο χαμηλότερος είναι ο ρυθμός κώδικα, τόσο υψηλότερη είναι η ανθεκτικότητα. Αυτή η διαδικασία ονομάζεται *αυξητικός πλεονασμός*. Ως εκ τούτου, αντί να έχει την ίδια πιθανότητα αποκωδικοποίησης για κάθε μετάδοση, όπως στον τύπο I, αυτή τη φορά κάθε αναμετάδοση αυξάνει την ποσότητα του πλεονασμού και επομένως, τη πιθανότητα επιτυχούς αποκωδικοποίησης. Έτσι, η HARQ II παρέχει ένα πολύ ευέλικτο επίπεδο προστασίας που προσαρμόζεται εύκολα σε χρονικά μεταβαλλόμενα κανάλια. Όμως αυτό το είδος είναι πιο περίπλοκο να υλοποιηθεί, γιατί απαιτεί μια στενή συνεργασία μεταξύ του FEC και ARQ και επιπλέον χρειάζεται πρόσθετους πόρους για να ελεγχθούν αποτελεσματικά οι ληφθέντες ακολουθίες. Ένα άλλο μειονέκτημα του τύπου II είναι ότι πολλές από τις αναμεταδόσεις περιέχουν μόνο ψηφία ισοτιμίας που δημιουργεί προβλήματα ως προς την ποιότητα του σήματος.

Η Τύπου III είναι παρόμοια με τον τύπο II με τη διαφορά ότι αυτή τη φορά κάθε πακέτο είναι αποκωδικοποιήσιμο ξεχωριστά από το επόμενο, πράγμα που σημαίνει ότι ακόμη και αν ένα πακέτο λείπει, ο παραλήπτης μπορεί να αποκωδικοποιήσει οποιαδήποτε συνδιασμό των πληροφοριών για να ανακτήσει τα μεταδιδόμενα δεδομένα. Αυτό είναι ένα τεράστιο πλεονέκτημα, αλλά και αυτό έρχεται με την τιμή της επιβάρυνσης των διαθέσιμων πόρων.

Υπάρχει μια ειδική περίπτωση όπου το ίδιο πλαίσιο με τις ίδιες πληροφορίες πλεονασμού μεταδίδεται για κάθε μετάδοση και συνδυάζεται με λογισμικό στο δέκτη για να εκμεταλλευτεί την ποικιλομορφία του χρόνου. Αυτό ονομάζεται chase combining. Θα μπορούσε να χαρακτηριστεί ως τύπου I HARQ με συνδυασμό λογισμικού ή τύπου III HARQ με μία εκδοχή πλεονασμού. Αυτή η μέθοδος είναι πολύ δημοφιλής, διότι παρέχει έναν καλό συμβιβασμό ανάμεσα στην απόδοση και τους πόρους που απαιτούνται για την υλοποίησή του. Ως εκ τούτου, είναι φτηνότερο για να εφαρμοστεί συγκριτικά με πιο πολύπλοκες μορφές.

Ανακεφαλαιώνοντας, το HARQ αυξάνει πραγματικά την αξιοπιστία και την αποτελεσματικότητα ενός καναλιού επικοινωνίας. Είναι συνήθως η πρώτη γραμμή άμυνας ενάντια στην απώλεια πακέτων, επιτρέποντας γρήγορη αναμετάδοση των λανθασμένων/χαμένων ακολουθιών. Αυτό δεν σημαίνει απαραίτητα ότι πρέπει σταδιακά να αντικαταστήσει την παραδοσιακή ARQ, η οποία ακόμα λειτουργεί σε υψηλότερο επίπεδο επικοινωνίας και με πιο αργό ρυθμό. Το 4G LTE (Long Term Evolution) είναι ένα καλό παράδειγμα ενός προτύπου που χρησιμοποιεί και τα δύο προηγούμενα πρωτόκολλα.

ΚΕΦΑΛΑΙΟ 4 Hardware Redundancy

Αρχικά οι τεχνικές πλεονασμού χρησιμοποιήθηκαν για την αντιστάθμιση της χαμηλής αξιοπιστίας των ηλεκτρονικών εξαρτημάτων. Οι σχεδιαστές υπολογιστικών συστημάτων τριπλασίαζαν χαμηλού επιπέδου εξαρτήματα όπως πύλες ή flip flop και χρησιμοποίησαν ψηφοφορία πλειοψηφίας για την επιδιόρθωση λαθών [1]. Με την βελτίωση της αξιοπιστίας τους, μεγαλύτερα εξαρτήματα είχαν την δυνατότητα να γίνουν εφεδρικά, όπως μνήμες και επεξεργαστές. Με τον τρόπο αυτό μειώθηκαν οι πιθανότητες αποτυχίας τους σε ικανοποιητικά επίπεδα.

Ο πλεονασμός υλικού ενδείκνυται στις περιπτώσεις που οι υπόλοιπες τεχνικές δεν είναι σε θέση να βελτιώσουν την αξιοπιστία του συστήματος. Επιτυγχάνεται με την προσθήκη δύο ή περισσότερων φυσικών αντιγράφων ενός εξαρτήματος υλικού. Ένα υπολογιστικό σύστημα μπορεί να περιέχει επιπλέον επεξεργαστές, μνήμες, διαύλους ή τροφοδοτικά για τον σκοπό αυτό. Είναι χρήσιμος σε περιπτώσεις, που ο εξοπλισμός δεν είναι σε θέση να συντηρηθεί, όπως δορυφόρους και διαστημικές αποστολές για την επέκταση του αδιάλειπτου χρόνου λειτουργίας.

Ωστόσο η αξιοποίηση του επιφέρει ένα πλήθος από ανταλλάγματα όπως την αύξηση του βάρους και όγκου του εξοπλισμού, την κατανάλωση ισχύος και τον επιπλέον χρόνο για τον σχεδιασμό, την δοκιμή και την παραγωγή των εφεδρικών μονάδων. Έτσι λοιπόν επηρεάζεται και το κόστος της παραγωγικής διαδικασίας. Παρ' όλα αυτά το βάρος και ο όγκος μπορούν να ελαττωθούν αν σμικρύνουμε τα εξαρτήματα του μεγαλυτέρου επιπέδου, ενώ το κόστος ελαττώνεται αν καλύπτει και ορισμένες λειτουργίες της συντήρησης.

Σύμφωνα με [1,2] υπάρχουν τρία είδη πλεονασμού υλικού. Ο *παθητικός , ο ενεργός* και ο *υβριδικός πλεονασμός*. Ο παθητικός πλεονασμός επιτυγχάνει την ανοχή σφαλμάτων με απόκρυψη σφαλμάτων από το σύστημα και χωρίς να είναι αναγκαία κάποια ενέργεια του διαχειριστή. Ο ενεργός πλεονασμός χρησιμοποιεί κάποια μέθοδο εντοπισμού πριν γίνει η ανοχή του σφάλματος. Μετά από τον εντοπισμό του ελαττωματικού εξαρτήματος οι ενέργειες προσδιορισμού της θέσης, του περιορισμού και της αποκατάστασης πραγματοποιούνται ώστε να αφαιρεθεί από το σύστημα. Ο υβριδικός πλεονασμός χρησιμοποιεί παθητικές και ενεργές προσεγγίσεις για την ανοχή σφαλμάτων. Η επικάλυψη σφαλμάτων χρησιμοποιείται για την παρεμπόδιση εσφαλμένων υπολογισμών. Ο εντοπισμός σφαλμάτων και οι ενέργειες εύρεσης της θέσης, του περιορισμού και της αποκατάστασης χρησιμοποιούνται για την αντικατάσταση εξαρτημάτων με εφεδρικό. Τέλος ο υβριδικός πλεονασμός επιτρέπει την αναδιάρθρωση του συστήματος με μηδενική επίδραση στην διαθεσιμότητα του συστήματος.

4.1 Δεσμεύσεις του Πλεονασμού

Η χρήση πλεονασμού σε ένα σύστημα δεν σημαίνει ότι θα έχουμε και άμεση βελτίωση της αξιοπιστίας του. Ο πλεονασμός αυξάνει την πολυπλοκότητα του συστήματος σε αισθητά επίπεδα, επομένως πρέπει να χρησιμοποιείται με αυστηρότητα και σε εξαρτήματα τα οποία το έχουν ανάγκη. Σε αντίθετη περίπτωση η συνολική αξιοπιστία κατακερματίζεται και γίνεται κακή διαχείριση των πόρων του συστήματος. Για να προκύψει ένα πιο αξιόπιστο σύστημα με χρήση πλεονασμού

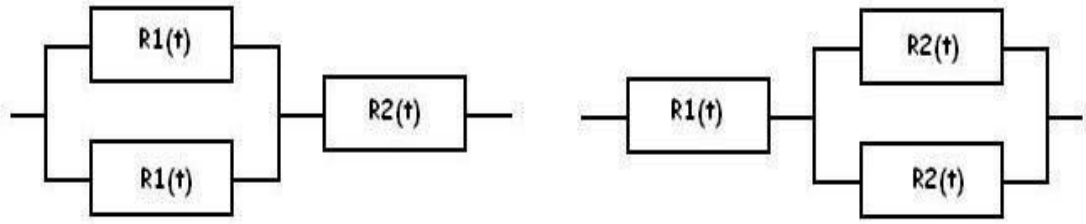
πρέπει να υπάρχει και μια ανάλυση που θα αποδεικνύει την ορθότητα του ισχυρισμού αυτού. Πρέπει να αναλυθούν πλήθος πιθανοτήτων ώστε να καθοριστεί σε ποιό βαθμό και ποιά εξαρτήματα θα πρέπει να γίνουν εφεδρικά.

Για να κατανοηθούν τα παραπάνω ακολουθεί ένα παράδειγμα ενός σειριακού συστήματος το οποίο αποτελείται από δύο είδη εξαρτημάτων. Το καθένα από αυτά έχει διαφορετικού βαθμού αξιοπιστία, R1(t) και R2(t). Αν η συνολική αξιοπιστία Rs(t) = R1(t)R2(t) , δεν συμφωνεί με τις προδιαγραφές που έχουν οριστεί για κάποιο από αυτά θα πρέπει να διπλασιαστούν μερικά από αυτά. Λαμβάνοντας υπόψη ότι οι αποτυχίες των δύο εξαρτημάτων είναι ανεξάρτητες μεταξύ τους, οι αντίστοιχες συνδεσμολογίες είναι οι παρακάτω :

$$R[I](t)=(2\mathrm{R1}(t)-R1^2(t))\,R2(t) \qquad (1)$$
$$R[II](t)=(2\mathrm{R2}(t)-R2^2(t))\,R1(t) \qquad (2)$$

$$R[I](t)-R[II](t)=R1(t)\,R2(t)(R2(t)-R1(t)) \qquad (3)$$

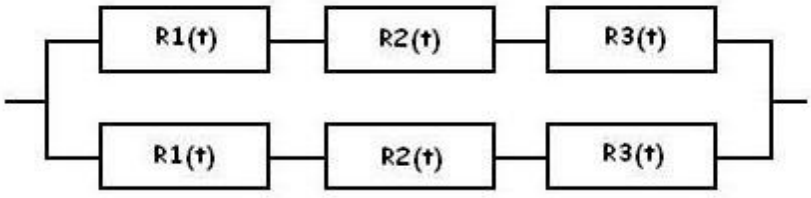
Από την (3) προκύπτει ότι έχουμε υψηλότερη αξιοπιστία αν διπλασιάσουμε το λιγότερο αξιόπιστο εξάρτημα. Αν R1(t) < R2(t) τότε το κύκλωμα (I) είναι προτιμότερο για χρήση, ενώ αν ισχύει το αντίθετο επιλέγουμε το δεύτερο τρόπο.



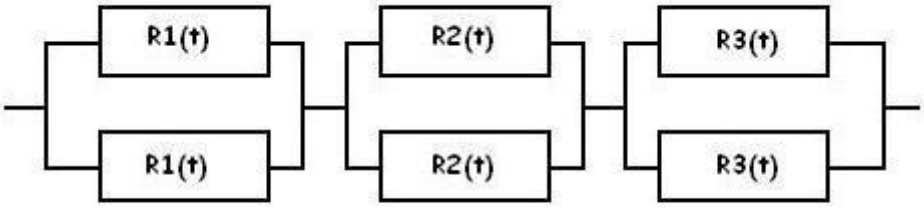
(I) (II)

Εκτός από το ποιο εξάρτημα επιλεγεί για πλεονασμό πρέπει να απαντηθεί και το ερώτημα, του επιπέδου που πρέπει να χρησιμοποιηθεί. Γενικά υπάρχουν δύο αντιμετωπίσεις, η μια είναι του υψηλού επιπέδου και η άλλη του χαμηλού επιπέδου.

Έστω ένα σύστημα που αποτελείται από τρία είδη εξαρτημάτων το καθένα με την δική του αξιοπιστία όπως και προηγουμένως. Ας συγκρίνουμε την αξιοπιστία των κυκλωμάτων (III) και (IV) για να βγάλουμε ορισμένα συμπεράσματα. Λαμβάνοντας πάλι υπόψην την ανεξαρτησία των αποτυχιών του κάθε εξαρτήματος έχουμε,



(III)



(IV)

$$R[III](t)=1-(1-R1(t)\,R2(t)\,R3(t))^2 \qquad (4)$$
$$R[IV](t)=1-(1-R1^2(t))\cdot(1-R2^2(t))\cdot(1-R3^2(t)) \qquad (5)$$

$$R[IV](t)-R[III](t)=6\mathrm{R}^3(t)(1-R^2(t)) \qquad (6)$$

Με παρατήρηση των σχέσεων (4) και (5) γίνεται αντιληπτό ότι τα κυκλώματα έχουν διαφορετική αξιοπιστία παρόλο που αποτελούνται από τα ίδια ακριβώς εξαρτήματα. Αν έχουν την ίδια αξιοπιστία, *R1(t)=R2(t)=R3(t)=R(t)*, η διαφορά που θα προκύψει είναι η σχέση (6). Από αυτή συμπεραίνουμε ότι R[IV](t)>R[III](t) και συνεπάγεται ότι *ο πλεονασμός χαμηλότερου επιπέδου προσδίδει μεγαλύτερη αξιοπιστία από αυτή των υψηλότερων επιπέδων.*

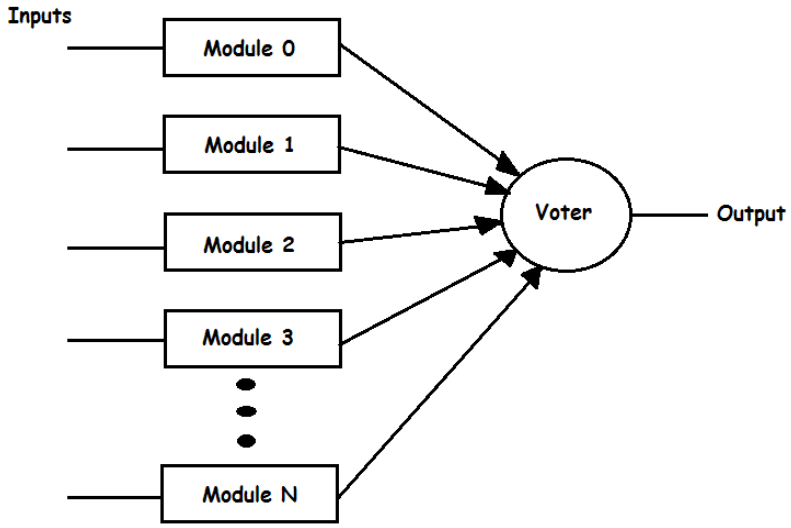
Τέλος κρίνεται απαραίτητο να τονιστεί ότι τα παραπάνω συμπεράσματα προέκυψαν για την περίπτωση ανεξαρτησίας των αποτυχιών σε κάθε εξάρτημα. Στην πραγματικότητα όμως τα κυκλώματα χαμηλού επιπέδου είναι περισσότερο ευάλωτα σε κοινές αιτίες σφαλμάτων [3], επειδή είναι λιγότερο απομονωμένα φυσικά μεταξύ τους, σε σχέση με αυτά των υψηλότερων επιπέδων, οπότε έτσι εξηγείται και το γεγονός ότι υποφέρουν από αποτυχίες κοινής αιτίας. Στο σημείο αυτό ακολουθεί αναλυτική περιγραφή των τεχνικών πλεονασμού υλικού.

4.2 Παθητικός Πλεονασμός

Χρησιμοποιείται για την απόκρυψη των σφαλμάτων σε ένα υπολογιστικό σύστημα και όχι για τον εντοπισμό τους[1]. Η απόκρυψη σφαλμάτων διασφαλίζει την ορθότητα των υπολογισμών κατά την μετάδοση της πληροφορίας στους αγωγούς του συστήματος, ακόμα και με την παρουσία σφάλματος. Οι τεχνικές της κατηγορίας αυτής χρησιμοποιούνται σε εφαρμογές υψηλής αξιοπιστίας, διότι έστω και μικρές διακοπές της ορθής λειτουργίας δεν είναι αποδεκτές επειδή δεν καθίσταται δυνατή η επισκευή και επιδιόρθωση του συστήματος. Μερικά παραδείγματα αποτελούν τα συστήματα ελέγχου πτήσης αεροσκαφών, τα ενσωματωμένα ιατρικά συστήματα και ηλεκτρονικά συστήματα διαστημικών αποστολών.

Χαρακτηριστικό παράδειγμα της κατηγορίας αυτής αποτελούν τα κυκλώματα N-Modular Redundancy ή NMR. Τα κυκλώματα αυτά αποτελούνται από N εξαρτήματα τα οποία τροφοδοτούνται από κοινή είσοδο δεδομένων. Η έξοδος κάθε εξαρτήματος οδηγείται σε έναν συμψηφιστή πλειοψηφίας, ο οποίος έχει το ρόλο

ελεγκτή των υπολογισμών σε κάθε εξάρτημα. Στην περίπτωση που υπάρχει ένα σφάλμα σε κάποιο εξάρτημα ο υπολογισμός διατηρεί την εγκυρότητά του με την βοήθεια του συμψηφιστή. Η απαραίτητη συνθήκη για την πλειοψηφία είναι το περιττό πλήθος των εξαρτημάτων όπως είναι λογικό. Ένα τέτοιο σύστημα απεικονίζεται παρακάτω.

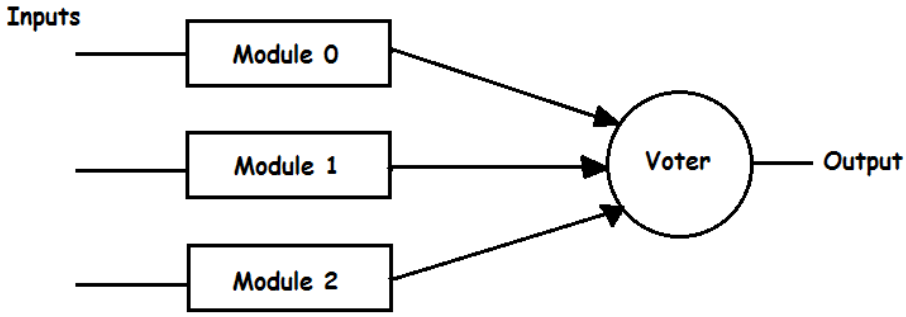


Εικόνα 4.1: Ένα σύστημα NMR με συμψηφιστή πλειοψηφίας.

Η πιο διαδεδομένη μορφή του κυκλώματος αυτού αποτελεί το TMR, από το Triple Modular Redundancy, όπου κάθε εξάρτημα τριπλασιάζεται και ο ίδιος υπολογισμός εκτελείται παράλληλα. Χρησιμοποιείται ο συμψηφιστής για τον καθορισμό του σωστού αποτελέσματος. Αν ένα από τα εξαρτήματα αποτύχει εξαιτίας κάποιου σφάλματος, αυτός είναι υπεύθυνος για την απόκρυψη του σφάλματος, λαμβάνοντας υπόψη τους υπολογισμούς των εναπομεινάντων δύο μονάδων. Ανάλογα με την εφαρμογή οι μονάδες μπορεί να είναι επεξεργαστές, μνήμες, δίσκοι, δίαυλοι κ.ο.κ. Έτσι λοιπόν ένα TMR είναι σε θέση να αποκρύβει μέχρι μία ελαττωματική μονάδα. Σε αντίθετη περίπτωση θα δημιουργούσε πρόβλημα κατά την σύγκριση στον συμψηφιστή, ο οποίος θα αποτύγχανε και αυτός με την σειρά του εκδίδοντας λάθος αποτέλεσμα προκαλώντας την αποτυχία του συστήματος ή την μειωμένη αξιοπιστία του. Γενικά χρησιμοποιείται σε εφαρμογές που είναι αναγκαία η μέγιστη αξιοπιστία για μικρό χρονικό διάστημα. Ένα τυπικό παράδειγμα [5] αποτελεί η χρήση TMR στον Saturn V ο οποίος αύξησε την αξιοπιστία της λογικής του μονάδας κατά 20 φορές περισσότερο σε σχέση με την απλή του μορφή, που δεν διέθετε πλεονασμό.

Όμως επειδή το κύκλωμα αυτό έχει την δυνατότητα απόκρυψης ενός ελαττωματικού εξαρτήματος από το σύστημα δεν σημαίνει άμεσα ότι είναι και πιο αξιόπιστο από ένα σύστημα που δεν χρησιμοποιεί πλεονασμό. Για να αποδειχτεί αυτό πρέπει να ληφθούν υπόψην οι αξιοπιστίες των εξαρτημάτων που θα χρησιμοποιηθεί και ο χρόνος αποστολής του συστήματος. Με την έννοια του *χρόνου αποστολής* δηλώνεται η διάρκεια μέσα στην οποία το σύστημα θα λειτουργήσει με την μεγαλύτερη αξιοπιστία, χωρίς να θέσει σε κίνδυνο την ευρύτερη υπηρεσία που θα προσδώσει.

Ας προχωρήσουμε σε μια γενικότερη ανάλυση αξιοπιστίας του συστήματος αυτού, χωρίς να αναφέρουμε συγκεκριμένα εξαρτήματα, αλλά για να μελετηθεί εκτενέστερα η λειτουργία του.



Εικόνα 4.2: Σύστημα TMR με συμψηφιστή πλειοψηφίας

4.2.1 Λειτουργία

Ένα TMR σύστημα αποτελείται από τρείς όμοιες μονάδες συνδεμένες με τον συμψηφιστή πλειοψηφίας παράλληλα. Πραγματοποιούν τον ίδιο υπολογισμό και παράγουν τα ίδια αποτελέσματα. Όπως αναφέραμε έχει την δυνατότητα να κρύβει μία ελαττωματική μονάδα και να συνεχίζει την ορθή λειτουργία του με τις υπόλοιπες δυο. Οπότε ο πίνακας αληθείας του συστήματος είναι:

Module 0	Module 1	Module 2	Voter
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Πίνακας 4.1 Πίνακας αληθείας ενός απλοποιημένου TMR.

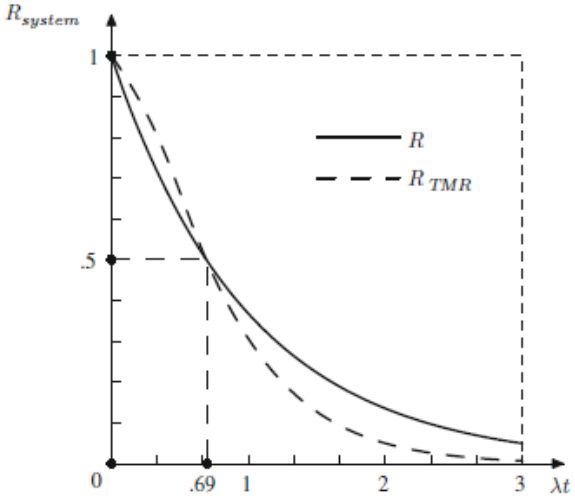
Παρατηρούμε ότι η έξοδος του συμψηφιστή έχει λογικό ένα για τις περιπτώσεις που λειτουργούν και οι τρείς μονάδες καθώς και για τις υπόλοιπες πού τουλάχιστον δύο είναι ενεργές. Από μια απλή ανάλυση αξιοπιστίας των περιπτώσεων προκύπτει :

$$R_{TMR} = R_1R_2R_3 + (1 - R_1)R_2R_3 + R_1(1 - R_2)R_3 + R_1y_2(1 - R_3)$$

Όπου R1, R2, R3, οι αξιοπιστία των εξαρτημάτων που είναι ενεργές και με (1- R1..3) όταν αυτές δεν είναι ενεργές. Ουσιαστικά αποτελούν τις πιθανότητες κάθε εξαρτήματος για ορθή και μη λειτουργία, και λαμβάνουν τιμές από 0 ως 1. Η μηδενική τιμή δηλώνει ότι το εξάρτημα δεν είναι αξιόπιστο, ενώ η τιμή ένα ότι είναι στην μέγιστη δυνατή αξιοπιστία. Για λόγους απλοποίησης, εφόσον κάθε εξάρτημα έχει την ίδια αξιοπιστία ισχύει,

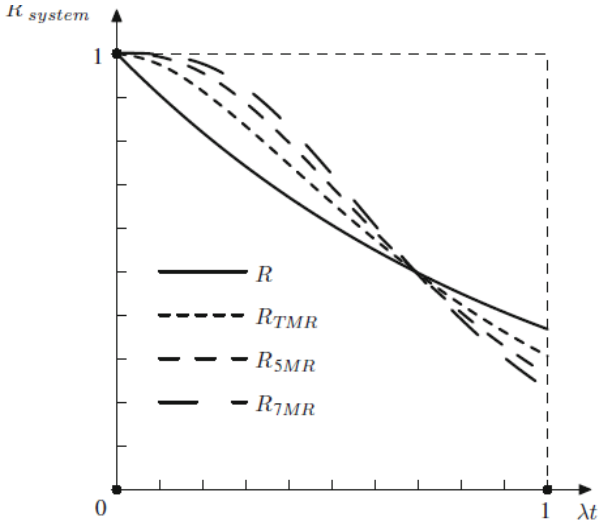
$$R_1 = R_2 = R_3 = R$$
$$R_{TMR} = 3R^2 - 2R^3 \quad \textbf{(7)}$$

Εισάγοντας την εκθετική κατανομή που περιγράφει την μεταβολή της αξιοπιστίας του κατά τον χρόνο ζωής των ηλεκτρονικών εξαρτημάτων, οπότε με αντικατάσταση προκύπτει η σχέση,



Εικόνα 4.3 Συγκριτικό διάγραμμα αξιοπιστίας TMR και μη πλεονάζοντος[2]

Από το διάγραμμα παραπάνω γίνεται προφανές ότι η αξιοπιστία ενός TMR είναι μεγαλύτερη στο διάστημα 0 – 0.69λt, που σημαίνει ότι είναι κατάλληλο για εφαρμογές που έχουν μικρό χρόνο ολοκλήρωσης. Ενώ από το διάστημα αυτό και μετά η αξιοπιστία του είναι μικρότερη από ένα συστήματος που δεν εφαρμόζει πλεονασμό. Στο παρακάτω διάγραμμα γίνεται η σύγκριση μεταξύ μερικών συστημάτων NMR από την σκοπιά της αξιοπιστίας που αποδίδουν. Γενικά ένα NMR είναι σε θέση να αποκρύψει N/2 σφάλματα σε ένα σύστημα.



Εικόνα 4.4: Συγκριτικό διάγραμμα αξιοπιστίας ορισμένων NMR συστημάτων[2]

Από το διάγραμμα αυτό μελετάται η αξιοπιστία των εξαρτημάτων για την ιδανική περίπτωση συμψηφιστή. Είμαστε υποχρεωμένοι όμως να μελετήσουμε και την περίπτωση αποτυχίας του, διότι αποτελεί σημείο μοναδικής αποτυχίας στο κύκλωμα.

Έτσι λοιπόν επειδή ο συμψηφιστής είναι σειριακά συνδεμένος στο κύκλωμα με τα εξαρτήματα, λαμβάνοντας υπόψην τον τύπο (7) προκύπτει,

$$R_{TMR} = (3\mathrm{R}^2 - 2\mathrm{R}^3) R_V$$

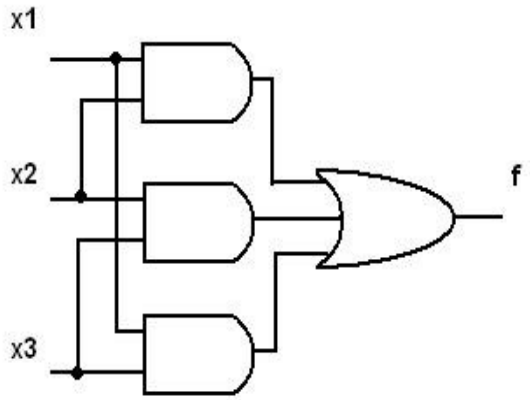
Για να υπολογιστεί η αξιοπιστία του πρέπει να βρεθεί η ελάχιστη περίπτωση που οριοθετεί και την λειτουργία του κυκλώματος. Δίνεται από τον τύπο:

$$R_V > 1/(3\mathrm{R} - 2\mathrm{R}^2)$$

και για την τιμή R=0.75 μεγιστοποιείται ο παρανομαστής, επομένως λαμβάνει την ελάχιστη τιμή Rv=0.899. Η τιμή αυτή είναι αρκετά υψηλότερη από τις τιμές των εξαρτημάτων και αυτό είναι λογικό καθώς είναι το μόνο εξάρτημα που εξάγει το αποτέλεσμα.

Ένα κύκλωμα ψηφοφορίας είναι απλό στην υλοποίηση του[6]. Αποτελείται από απλά εξαρτήματα σε επίπεδο πυλών. Απαρτίζεται από τρείς πύλες AND και μια OR όπως φαίνεται και παρακάτω από τον πίνακα αληθείας και το κύκλωμα.
Η λειτουργία του είναι πανομοιότυπη με αυτή των TMR, ωστόσο είναι σε θέση να δέχεται πολλές εισόδους.

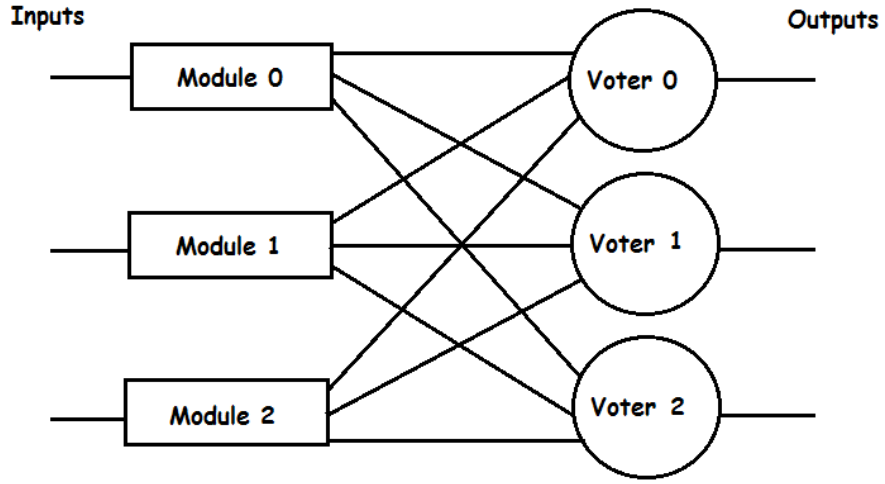
x1	**x2**	**x3**	**F**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Εικόνα 4.5: Πίνακας αληθείας και δομή κυκλώματος ψηφοφορίας 1-bit

Αν και η δομή του είναι απλή ακόμα υπάρχει η περίπτωση αποτυχίας. Στις περιπτώσεις αυτές χρησιμοποιούνται εφεδρικά κυκλώματα ψηφοφορίας. Μια από αυτές είναι ο τριπλασιασμός του, έτσι ώστε να αποφεύγονται τα μοναδικά σημεία αποτυχίας έπειτα από συνεχή ανταλλαγή μηνυμάτων μεταξύ τριών συμψηφιστών. Μια άλλη λύση είναι η χρήση TMR με εφεδρικούς συμψηφιστές σε βαθμίδες[7]. Δηλαδή να χρησιμοποιηθούν σε σειρά δύο η περισσότερα κυκλώματα TMR όπως το παρακάτω [4.6].

Η ψηφοφορία βασίζεται στον ακριβές χρονισμό των εισόδων [6], που σημαίνει ότι αν οι τιμές φτάσουν σε διαφορετικές χρονικές στιγμές έστω και ελάχιστες τότε το κύκλωμα αποτυγχάνει. Έτσι λοιπόν απαιτείται ο συγχρονισμός των μονάδων αυτών χρησιμοποιώντας επιπλέον χρονιστές ή ασύγχρονα πρωτόκολλα επικοινωνίας των μονάδων που βασίζονται στην πρόοδο του υπολογισμού με το πέρασμα συγκεκριμένης χρονικής διάρκειας.



Εικόνα 4.6: Διάγραμμα TMR με τρία κυκλώματα ψηφοφορίας.

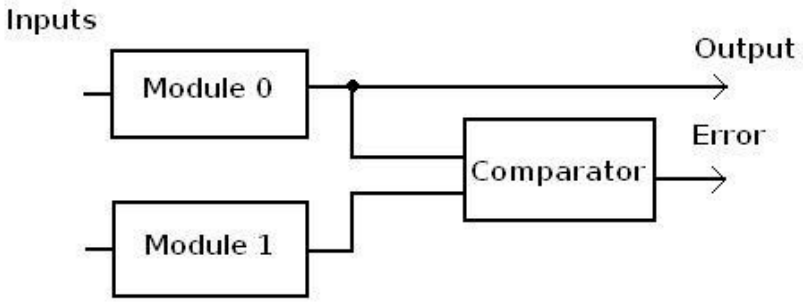
Άλλο πρόβλημα που μπορεί να συμβεί έχει να κάνει με τις τιμές που εισάγονται στην ψηφοφορία. Για παράδειγμα ,σε ένα A/D ελεγκτή οι τιμές μπορεί να διαφέρουν ελάχιστα κατά τον ίδιο υπολογισμό. Το πρόβλημα αυτό λύνεται λαμβάνοντας την μέση τιμή. Τέλος, για ειδικότερες μεθόδους και πρωτόκολλα ψηφοφορίας μπορούν να βρεθούν στο [6].

4.3 Ενεργητικός πλεονασμός

Η ανοχή σφαλμάτων επιτυγχάνεται αρχικά με τον εντοπισμό σφαλμάτων και στην συνέχεια ενεργοποιούνται οι κατάλληλες διαδικασίες επιδιόρθωσης για την επαναφορά του συστήματος σε κατάσταση ορθής λειτουργίας. Αυτή η μορφή πλεονασμού εφαρμόζεται σε περιπτώσεις που είναι απαραίτητη η υψηλή διαθεσιμότητα του συστήματος. Μερικές από αυτές είναι τα υπολογιστικά συστήματα διαμοιρασμού χρόνου και τα συστήματα συναλλαγών. Σπάνια σφάλματα είναι ανεκτά με την συνθήκη ότι θα επαναφερθεί στην αρχική του κατάσταση σε συγκεκριμένη χρονική διάρκεια. Υπάρχουν τρεις δημοφιλείς διαδικασίες πλεονασμού με βάση τα εξαρτήματα: ο διπλασιασμός και σύγκριση, η ετοιμότητα και ο διπλασιασμός μαζί με εφεδρεία.

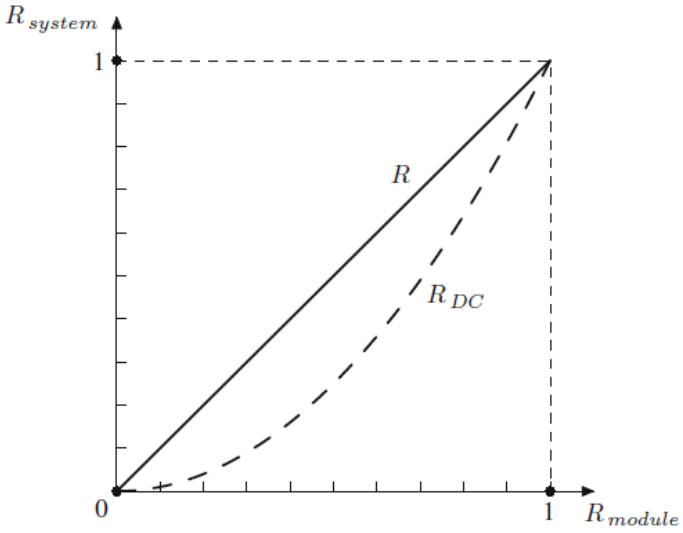
4.3.1 Διπλασιασμός και Σύγκριση

Αποτελεί βασική μέθοδο ενεργού πλεονασμού, χρησιμοποιεί δύο ίδιες μονάδες και εκτελείται η πράξη της σύγκρισης στην έξοδό τους για την διασφάλιση της εγκυρότητας των υπολογισμών. Οι μονάδες μπορεί να είναι επεξεργαστές, αισθητήρες, μνήμες κ.ο.κ και η σύγκριση γίνεται από κύκλωμα συγκριτή. Η μέθοδος αυτή προϋποθέτει όμως ότι οι δύο μονάδες λειτουργούν ορθά. Με αυτή μπορούν να εντοπιστεί ένα σφάλμα σε κάποια μονάδα. Όταν εντοπιστεί η ελαττωματική μονάδα το σύστημα δεν μπορεί να επανέλθει στην κανονική του λειτουργία χωρίς την επιδιόρθωση του, αυτόματα η με την βοήθεια τεχνικού προσωπικού.



Εικόνα 4.7: Διάγραμμα απλού συστήματος σύγκρισης δυο όμοιων μονάδων.

Όταν συμβεί το πρώτο σφάλμα, ο συγκριτής αντιλαμβάνεται την διαφορά στους υπολογισμούς και παράγει ένα μήνυμα λάθους. Δεν είναι σε θέση να αντιληφθεί ποιά από τις δύο είναι η προβληματική. Έτσι η μέθοδος αυτή είναι αποτελεσματική αν τα εξαρτήματα που θα χρησιμοποιηθούν έχουν μεγάλες τιμές αξιοπιστίας. Αυτό αιτιολογείται και από το παρακάτω διάγραμμα σύγκρισης ενός συστήματος χωρίς πλεονασμό και ένα με διπλασιασμό και σύγκριση. Τα εξαρτήματα με χαμηλές τιμές αξιοπιστίας δεν είναι κατάλληλα για την μέθοδο αυτή, όπως παρατηρείτε και από το παρακάτω διάγραμμα.

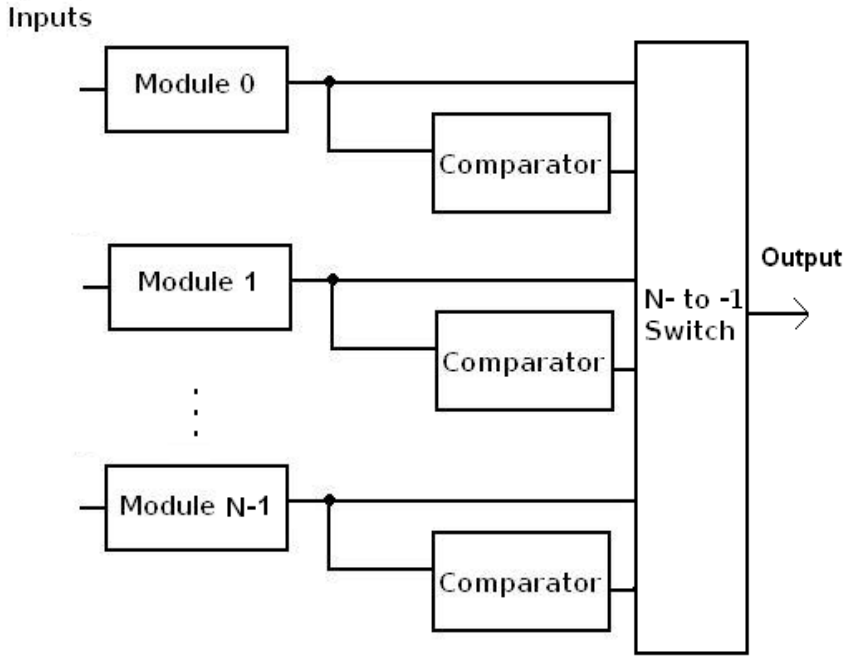


Εικόνα 4.8: Συγκριτικό διάγραμμα αξιοπιστίας συστημάτων εντοπισμού και σύγκρισης με μη πλεονάζων. [2]

4.3.2 Πλεονασμός ετοιμότητας

Η ιδέα πίσω από την υλοποίηση αυτή είναι απλή. Υπάρχει μια αρχική μονάδα η οποία είναι ενεργή και άλλες Ν ίδιες σε ετοιμότητα, με την μορφή εφεδρείας. Κάθε μονάδα έχει την δική της μονάδα εντοπισμού σφαλμάτων για την διασφάλιση των υπολογισμών. Όλα όμως τα εξαρτήματα είναι συνδεμένα παράλληλα με έναν διακόπτη. Ο διακόπτης αυτός είναι υπεύθυνος για την εναλλαγή των εξαρτημάτων σε περίπτωση αποτυχίας. Η εναλλαγή αυτή γίνεται με δύο τρόπους, την ζεστή και την κρύα (hot and cold swap) ή αλλιώς ενεργή και ανενεργή εναλλαγή.

Κατά την ζεστή εναλλαγή, όλες οι μονάδες βρίσκονται σε ισχύ ώστε να υπάρχει η ελάχιστη χρονική διάρκεια κατά την εναλλαγή με την εφεδρική και να υπάρξει ανοχή σφαλμάτων. Το μειονέκτημα άδω είναι η κατανάλωση ενέργειας. Στην κρύα εναλλαγή οι μονάδες εκτός από την αρχική βρίσκονται σε ανενεργή κατάσταση. Όταν προκύψει πρόβλημα με την αρχική μονάδα τότε ενεργοποιείται μια εφεδρική, εκτελείται πάλι ο υπολογισμός και το σύστημα επαναφέρεται στην κανονική του κατάσταση. Παρ' όλα αυτά και στις δύο περιπτώσεις υπάρχει το εξής πρόβλημα. Οι εναλλαγές θα γίνονται ώσπου να τελειώσουν τα εφεδρικά εξαρτήματα. Άρα η μέθοδος αυτή είναι χρήσιμη για την ανοχή N-1 σφαλμάτων σε ένα σύστημα, καθώς με την N-οστή αποτυχία θα επέλθει και αποτυχία του συστήματος.



Εικόνα 4.9: Διάγραμμα συστήματος ετοιμότητας N μονάδων.

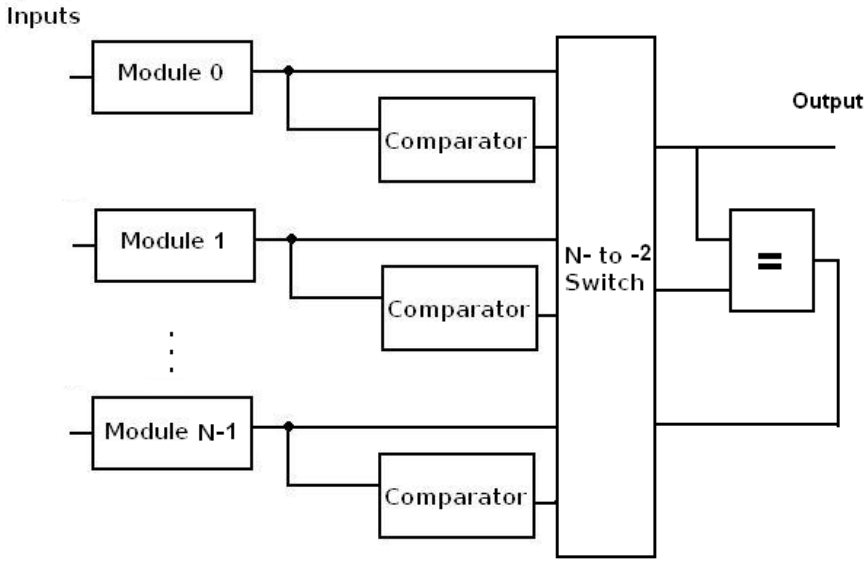
Επομένως η αξιοπιστία σε αυτή την μέθοδο εξαρτάται από από την αλληλεξάρτηση των εξαρτημάτων ή καλύτερα τις μεταβάσεις από την μια κατάσταση στην επόμενη. Είναι προφανές ότι η βέλτιστη μέθοδος ανάλυσης είναι οι αλυσίδες Markov. Πρέπει να τονιστεί εδώ σύμφωνα με [2] ότι σύγκριση των υπολογισμών δεν έχει σχέση με το αν υπάρχει η δυνατότητα κάλυψης των σφαλμάτων στις μονάδες. Για παράδειγμα, ένας EDC κώδικας είναι γνωστό ότι δεν μπορεί να εντοπίσει όλα τα σφάλματα που μπορούν να συμβούν σε μία μονάδα. Το γεγονός αυτό όμως δεν σημαίνει ότι η μονάδα σύγκρισης έχει αποτύχει στον σκοπό της, αλλά ότι δεν υπάρχει η μέγιστη κάλυψη από αυτήν. Όμως αν μια μονάδα σύγκρισης αποτύχει τότε μπορεί να επέλθει αποτυχία του συστήματος.

4.3.3 Ζεύγος και εφεδρεία (Pair-n-Spare)

Η τεχνική αυτή συνδυάζει την μέθοδο του διπλασιασμού και της ετοιμότητας. Λειτουργεί με δύο τρόπους. Η πρώτη μέθοδος αποτελείται από δύο ενεργές μονάδες που είναι τοποθετημένες παράλληλα. Τα αποτελέσματα της κάθε μονάδας εισάγονται

προς επεξεργασία από τους συγκριτές για να εντοπιστεί πιθανή ασάφεια. Αν παραχθεί λάθος, τότε ένα σήμα ελέγχου αποστέλλεται στον διακόπτη, γίνεται σύγκριση μεταξύ των μονάδων σύγκρισης της κάθε μονάδας ώστε να διαπιστωθεί ποιά είναι η προβληματική. Όταν εντοπιστεί, αντικαθίσταται από μια εφεδρική η οποία μπορεί να είναι σε ζεστή είτε κρύα ετοιμότητα. Ένα τέτοιο σύστημα μπορεί να ανεχτεί N -2 αποτυχίες μονάδων. Όταν όμως συμβεί το N-1 σφάλμα, θα εντοπιστεί και θα παραχθεί το σωστό αποτέλεσμα. Αλλά δεν θα υπάρχουν διαθέσιμες άλλες εφεδρικές μονάδες για αντικατάσταση, οπότε δεν θα συνεχιστεί η ορθή λειτουργία του συστήματος, καθώς θα υπάρχει αναξιοπιστία στις δύο μονάδες.

Παρ' όλα αυτά μπορούμε να εξελίξουμε την μέθοδο αυτή ώστε μία μονάδα να είναι σε λειτουργία μετά από την N-1 αποτυχία. Όταν ο διακόπτης λάβει το σήμα λάθους ο συγκριτής στην έξοδο μπορεί να αποσυνδεθεί. Με την διαδικασία αυτή ο διακόπτης μπορεί να αλλάξει την λειτουργία του σε N-σε-1 διακόπτη, που δέχεται σήμα σφάλματος από τους συγκριτές των μονάδων όπως στην μέθοδο της ετοιμότητας. Με τον τρόπο αυτό το σύστημα αυτό είναι σε θέση να εντοπίζει N σφάλματα και να αντιμετωπίζει N-1 σφάλματα υλικού. Η μέθοδος αυτή ονομάζεται *reconfigurable runtime fault tolerance.*



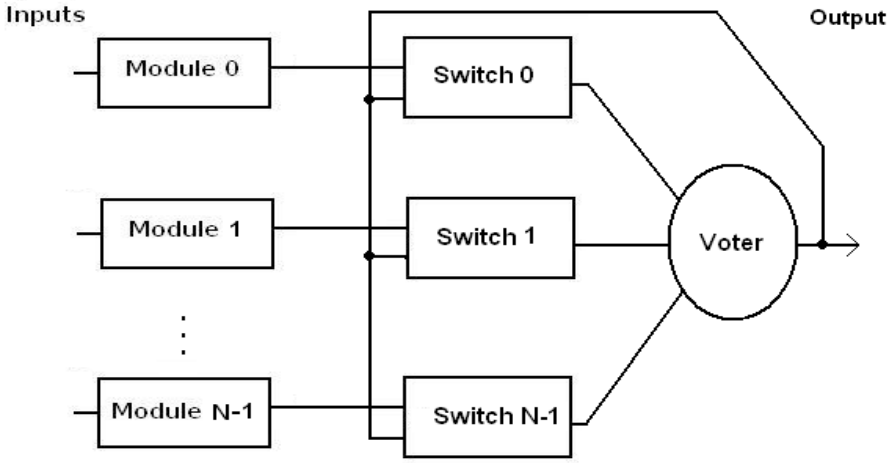
Εικόνα 4.10 Διάγραμμα PNS συστήματος.

4.4 Υβριδικός Πλεονασμός

Ο υβριδικός πλεονασμός αξιοποιεί τα πλεονεκτήματα του παθητικού και του ενεργού πλεονασμού, δημιουργώντας πιο συνθέτες λύσεις στα προβλήματα αξιοπιστίας των συστημάτων[1,2]. Ο παθητικός χρησιμοποιείται για την απόκρυψη στιγμιαίων υπολογισμών. Ο ενεργός χρησιμοποιείται για τον εντοπισμό και την επιδιόρθωση καθώς και την αφαίρεση από το σύστημα της προβληματικής μονάδας και αντικατάσταση της από νέα όμοια μονάδα.

4.4.1 Αυτό-αφαιρούμενος πλεονασμός

Αποτελείται από N ίδιες μονάδες οι οποίες πράττουν τον ίδιο υπολογισμό παράλληλα και συμμετέχουν σε διαδικασία ψηφοφορίας για την εκλογή ορθού αποτελέσματος. Η έξοδος του συμψηφιστή συγκρίνεται με τις επιμέρους μονάδες ελέγχου για τον εντοπισμό λαθών. Αν συμβεί ένα σφάλμα σε κάποια από τις μονάδες, ο διακόπτης το αφαιρεί από το κύκλωμα. Ο συμψηφιστής σχεδιάζεται ως πύλη κατωφλίου, με την δυνατότητα να μεταβάλλει την λειτουργία του ανάλογα με τις διαθέσιμες μονάδες του συστήματος, έτσι προσαρμόζεται στις μεταβολές που μπορούν να συμβούν.



Εικόνα 4.11 Διάγραμμα SPRN συστημάτων

Μια πύλη κατωφλίου περιγράφεται από τις εξής παραμέτρους, τις εισόδους *Xi*, το βάρος *Wi* , την τιμή κατωφλίου T και την έξοδο. Κάθε είσοδος έχει και ένα ειδικό βάρος Wi. Η έξοδος καθορίζεται από την σύγκριση του πλήθους των σταθμισμένων εισόδων με την τιμή κατωφλίου *T* . Επομένως οι σχέσεις που περιγράφουν ένα τέτοιο κύκλωμα είναι οι παρακάτω και υλοποιούν απλές πράξεις πρόσθεσης και πολλαπλασιασμού:

$$f=1 \quad , \quad \alpha\nu \quad \sum_{i=1}^{n} XiWi \quad > T$$

$$f=0 \quad , \quad \sigma\varepsilon \quad \acute{\alpha}\lambda\lambda\eta \quad \pi\varepsilon\rho\acute{\iota}\pi\tau\omega\sigma\eta$$

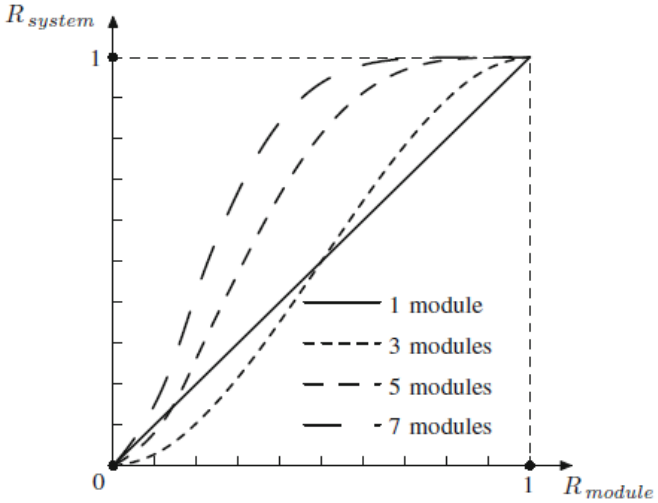
Επομένως ένας συμψηφιστής κατωφλίου αφαιρεί ένα εξάρτημα από το κύκλωμα με την μείωση του βάρους της προβληματικής μονάδας προς το μηδέν. Με τον τρόπο αυτό οι ελαττωματικές μονάδες δεν συνεισφέρουν στο συνολικό άθροισμα.

Ένα αυτό-αφαιρούμενο σύστημα με N λειτουργικές μονάδες είναι σε θέση να αποκρύψει N-2 σφάλματα. Με την αφαίρεση N-2 μονάδων, θα απομείνουν δυο ενεργές μονάδες και το σύστημα θα είναι σε θέση να εντοπίσει το επόμενο N-1 σφάλμα, όμως ο συμψηφιστής δεν θα μπορέσει να αξιολογήσει ποιο αποτέλεσμα είναι σωστό.

Από την σκοπιά της αξιοπιστίας, οι ελάχιστες ενεργές μονάδες που πρέπει να υπάρχουν στο σύστημα είναι δύο. Κάθε μονάδα εφόσον λειτουργεί παράλληλα και είναι ίδιες θα έχουν και κοινή αξιοπιστία R. Για να θεωρηθεί αναξιόπιστο είτε πρέπει όλες οι μονάδες να έχουν αποτύχει, είτε όλες εκτός από μια. Επομένως, επειδή αυτή θα είναι μια από τις N λειτουργικές προκύπτει ο παρακάτω τύπος:

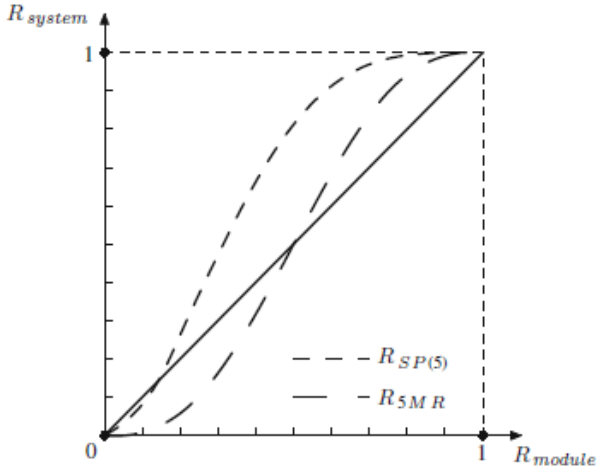
$$R_{SPR} = 1 - (1-R)^N - NR(1-R)^{(N-1)}$$

Ανάλογα με το πλήθος των εξαρτημάτων που θα χρησιμοποιηθούν, επηρεάζεται και ο βαθμός της αξιοπιστίας του συστήματος. Παρακάτω δίνεται συγκριτικό διάγραμμα μεταξύ αυτό αφαιρούμενων συστημάτων με 3,5,7 ενεργών μονάδων και η συνεισφορά τους στην συνολική αξιοπιστία ενός μεγαλύτερου συστήματος.



Εικόνα 4.12: Συγκριτικό διάγραμμα αξιοπιστίας SP3,5,7 συστημάτων.[2]

Τέλος παρουσιάζει ενδιαφέρον η σύγκριση των αυτό-αφαιρούμενων συστημάτων με τα παθητικά NMR, ίδιου πλήθους εξαρτημάτων. Όπως είναι φανερό από το διάγραμμα η αξιοπιστία των 5MR είναι κατά πολύ λιγότερη σε σχέση με ενός αυτό αφαιρούμενου συστήματος 5 μονάδων.



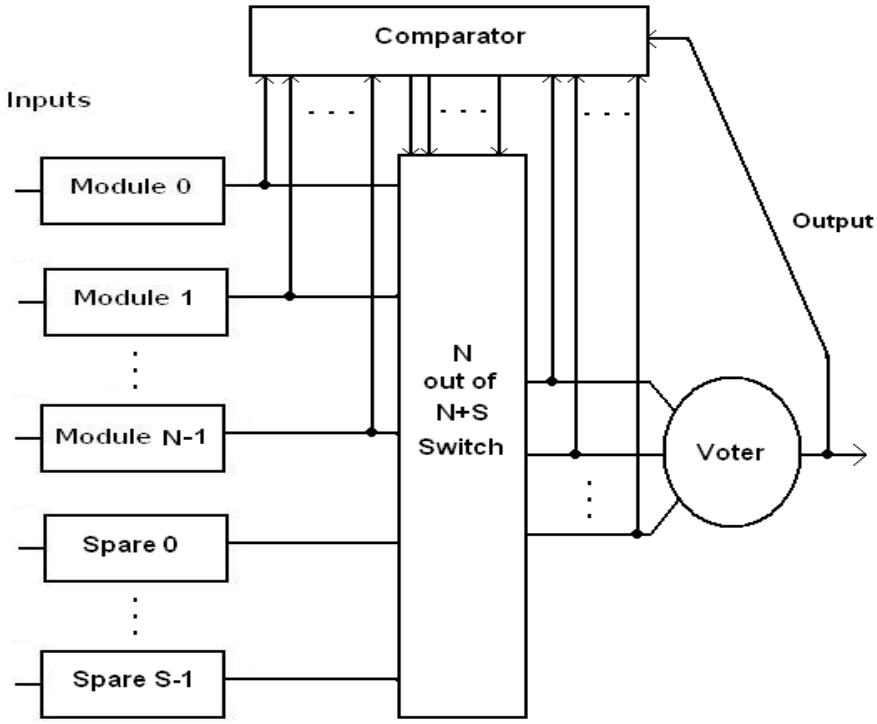
Εικόνα 4.13 Συγκριτικό διάγραμμα αξιοπιστίας 5MR και SPR5 συστημάτων.[2]

4.4.2 NMR πλεονασμός με εφεδρεία

Η μορφή αυτή αποτελεί συνδυασμό των NMR συστημάτων και του πλεονασμού εφεδρείας. Στην απλή του μορφή αποτελείται από ένα NMR με συγκεκριμένο πλήθος εφεδρικών μονάδων S. Οι μονάδες λειτουργούν παράλληλα και τροφοδοτούν την ψηφοφορία διαμέσου ενός διακόπτη. Σε περίπτωση που διαγνωστεί μια ελαττωματική μονάδα, ο διακόπτης την αφαιρεί και αντικαθιστά με μια εφεδρική. Υπάρχουν αρκετοί τρόποι για να διαγνωστούν προβληματικές μονάδες, μερικοί είναι οι κώδικες εντοπισμού EDC, εσωτερικά του διακόπτη και η ανατροφοδότηση της εξόδου σε κάθε μονάδα για την σύγκριση των αποτελεσμάτων.

Ενδεικτικά θα αναλύσουμε την μέθοδο της ανατροφοδότησης. Η έξοδος του συμψηφιστή και οι έξοδοι κάθε μονάδας εξετάζονται ξεχωριστά για πιθανές ασάφειες στα αποτελέσματα. Αν βρεθεί προβληματική μονάδα, που δεν συμφωνεί με την ψηφοφορία, αντικαθίσταται με εντολή του συγκριτή προς τον διακόπτη. Αυτό γίνεται για όλες τις περιπτώσεις μέχρι να μην υπάρχουν άλλες διαθέσιμες για αξιοποίηση.

Με την εξάντληση των εξαρτημάτων ο συγκριτής αυτόματα απενεργοποιείται και το σύστημα συνεχίζει την λειτουργία του ως ένα NMR σύστημα. Με τον τρόπο αυτό είναι σε θέση να αποκρύπτονται *(N/2)+S* σφάλματα εξαρτημάτων.



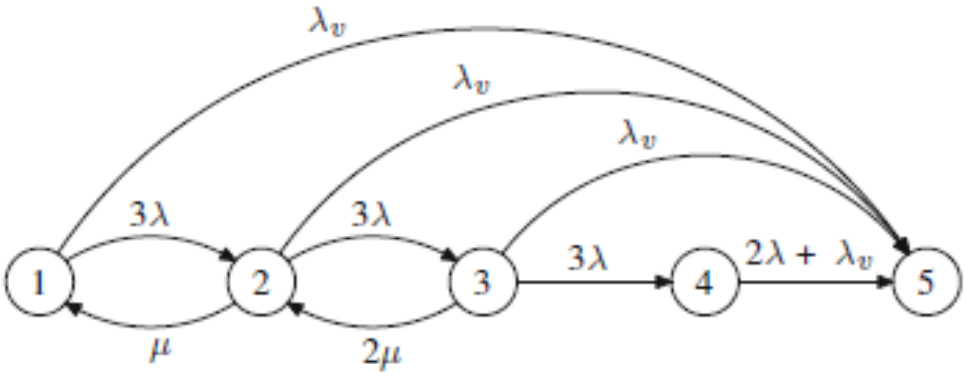
Εικόνα 4.14: NMR σύστημα με εφεδρεία και ανατροφοδότηση της εξόδου

Λειτουργία

Στο σημείο αυτό πρέπει να καθοριστούν οι καταστάσεις λειτουργίας των συστημάτων αυτών. Θα αναλύσουμε για λόγους κατανόησης την απλή περίπτωση

ενός TMR με δύο επιπλέον εφεδρικές μονάδες. Υποθέτουμε ότι ο ρυθμός αποτυχίας των μονάδων και των δύο εφεδρικών είναι *λ* και του συμψηφιστή είναι *λv.* Οι εφεδρικές δεν μπορούν να αποτύχουν όταν βρίσκονται σε ετοιμότητα. Θεωρούμε τον διακόπτη και τον συγκριτή ιδανικούς. Κάθε φορά που έχουμε επιδιόρθωση με την μέθοδο αντικατάστασης κάθε μονάδα έχει τον συντελεστή *μ*. Για να θεωρηθεί μια μονάδα ως ελαττωματική πρέπει να αναγνωριστεί από τον συγκριτή. Την στιγμή που το σύστημα θα λειτουργεί ως TMR και ο συγκριτής έχει απενεργοποιηθεί δεν επιδέχεται άλλη επιδιόρθωση. Οι παραπάνω υποθέσεις γίνονται για λογούς απλοποίησης. Επομένως μπορούμε να ορίσουμε τις εξής καταστάσεις:

- Οι τρεις μονάδες και οι δύο εφεδρικές βρίσκονται σε λειτουργία. (1)
- Μια μονάδα απέτυχε και αντικαταστάθηκε από εφεδρική. (2)
- Δεύτερη αποτυχία μονάδας και αντικατάσταση της. Τέλος εφεδρείας. (3)
- Αποτυχία σε μία από τις μονάδες του TMR (4)
- Αποτυχία συστήματος. (5)



Εικόνα 4.14:Διάγραμμα Markov του συστήματος TMR με δύο spares.

Όπως γίνεται προφανές με το παραπάνω διάγραμμα, υπάρχουν τέσσερις καταστάσεις λειτουργίας και μια αποτυχίας. Όταν αξιολογούμε την αξιοπιστία συστημάτων, κρίνεται σωστό να την περιγράφουμε με την αδιάλειπτη λειτουργία μέχρι το σύστημα να αποτύχει. Αυτό σημαίνει ότι η Διαθεσιμότητα δεν πρέπει να θίγεται για γεγονότα επιδιόρθωσης, πχ από εξωτερική ομάδα τεχνικών οι οποίοι θα πρέπει να σταματήσουν το TMR σύστημα, για να εντοπιστεί η προβληματική μονάδα, που είναι κρυμμένη από το ίδιο το σύστημα. Επιπλέον οι καταστάσεις αποτυχίας δεν είναι απαραίτητο να διαφοροποιηθούν, παρά την ορίζουμε σαν μία κατάσταση που οδηγεί στην αποτυχία όλου του συστήματος, όπως είναι και λογικό. Όλες οι πιθανές αιτείες οδηγούν στην πέμπτη κατάσταση, επομένως αποτυγχάνει.

Συνοψίζοντας, παρουσιάστηκαν οι πιο διαδεδομένες τεχνικές πλεονασμού με στόχο το υλικό των υπολογιστικών συστημάτων. Οι τρόποι αυτοί είναι οι βασικές τεχνικές και δεν κρίνεται αναγκαίο η περιγραφή των εξαρτημάτων, για παράδειγμα αν είναι επεξεργαστές , μνήμες δίσκοι κ.ο.κ παρά θεωρήσαμε ότι για μικρούς χρόνους αποστολής, η απόκρυψη σφαλμάτων είναι η βέλτιστη λύση. Όταν οι χρόνος αποστολής ενός συστήματος, πρέπει να έχει την ίδια διάρκεια με ένα που δεν διαθέτει πλεονασμό, ο αυτό-αφαιρούμενος πλεονασμός ενδείκνυται για χρήση. Για αποστολές μεγάλης χρονικής διάρκειας, τα συστήματα ετοιμότητας προσφέρουν την καλύτερη σχέση απόδοσης κόστους.

ΚΕΦΆΛΑΙΟ 5 Software Redundancy

Ο πλεονασμός εφαρμόζεται στα συστήματα λογισμικού από τις πρώτες μορφές του και έχει αναπτυχθεί σε μεγάλο βαθμό από τότε. Οι τεχνικές που χρησιμοποιούνται συνεχώς βελτιώνονται και αντιμετωπίζουν ένα μεγάλο φάσμα από σφάλματα, λάθη και τις πιθανές αιτίες τους. Στην ενότητα αυτή θα αναφερθούν βασικές τεχνικές καθώς και σύγχρονες εφαρμογές στα συστήματα λογισμικού.

Η ανοχή σφαλμάτων λογισμικού μπορεί να χωριστεί σε δύο ομάδες: στην ενιαία έκδοση και στις πολλαπλές εκδόσεις [1]. Οι τεχνικές απλής έκδοσης στοχεύουν στην ανοχή σφαλμάτων ενός στοιχείου λογισμικού με την προσθήκη των μηχανισμών για την ανίχνευση σφαλμάτων, τον περιορισμό, και την ανάκτηση. Οι τεχνικές πολλαπλών εκδόσεων χρησιμοποιούν εφεδρικά μέρη λογισμικού, τα οποία ακολουθούν εξελιγμένους κανόνες ποικιλομορφίας σχεδιασμού. Ως ιδέες όμως είναι αλληλένδετες καθώς χωρίς την ενίσχυση ενός στοιχείου λογισμικού με δυνατότητες αντιμετώπισης βλαβών δεν μπορεί να υπάρξει μια πολλαπλή έκδοση του λογισμικού αυτού με αντίστοιχες δυνατότητες. Οπότε από την απλή έκδοση καθορίζονται και οι πολλαπλές.

Κάθε εξάρτημα λογισμικού καθορίζεται από μια αρχιτεκτονική. Αυτή ορίζει τον τρόπο λειτουργίας ενός συστήματος λογισμικού και τον τρόπο σύνδεσης των συστατικών μερών του. Για την επίτευξη της ανοχής σφαλμάτων κάποιος θα πρέπει να ξεκινήσει από το επίπεδο αυτό. Ο απαραίτητος έλεγχος των υπολογισμών και των μηχανισμών επικύρωσης τους επηρεάζει τον τρόπο ανάπτυξης ενός συστήματος λογισμικού. Οι βασικές τεχνικές σχεδίασης εμφανίζονται με το παρακάτω σχήμα 5.1. Απεικονίζονται τρεις τεχνικές που χρησιμοποιούνται ευρύτατα για την ενσωμάτωση ανοχής σφαλμάτων στο λογισμικό[2]. Αυτές είναι η παράλληλη επικύρωση, η παράλληλη επιλογή και η σειριακές εναλλακτικές. Κάθε αρχιτεκτονική αποτελείται από δυο συντελεστές, την μονάδα που προσφέρει μια λειτουργία και τον επικριτή αυτής.

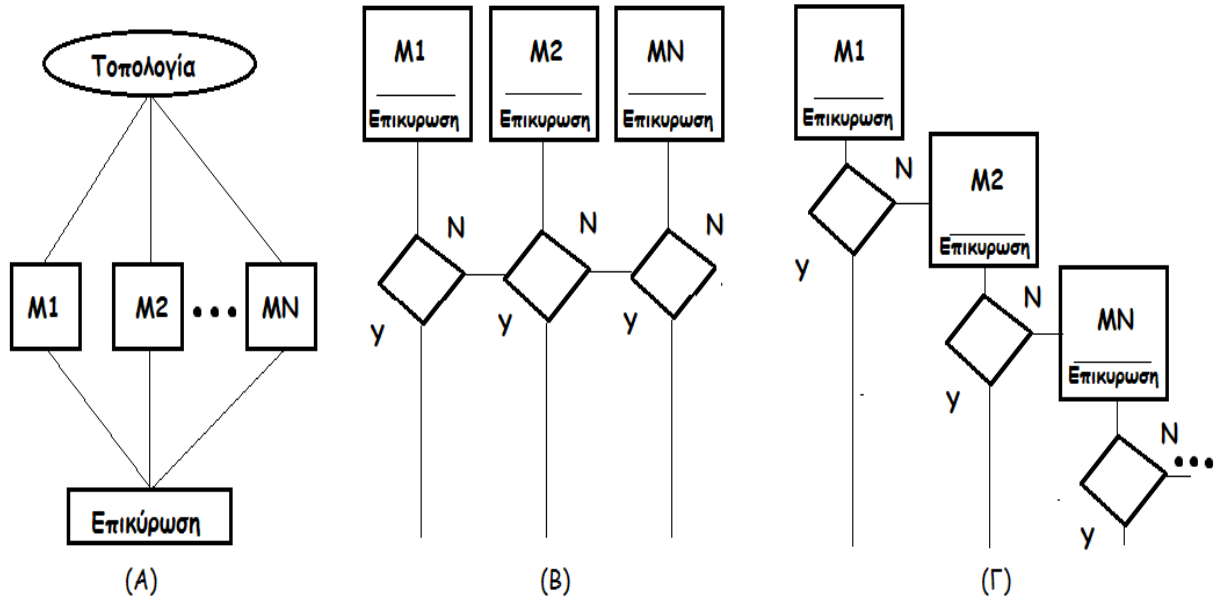
Η παράλληλη επικύρωση χρησιμοποιεί προγράμματα ενός έργου λογισμικού σε παράλληλη σύνδεση ώστε να λαμβάνονται οι απαραίτητες αποφάσεις από τον επικριτή. Κάθε μονάδα, η οποία μπορεί να είναι ένα πρόγραμμα είτε μια διεργασία, δέχεται ως είσοδο ορισμένα δεδομένα, εκτελούνται οι απαραίτητοι υπολογισμοί και επικυρώνονται ανάλογα τα αποτελέσματα.

Η παράλληλη επιλογή χρησιμοποιεί επικριτή μέσα σε κάθε πρόγραμμα και συγκριτή στην έξοδο κάθε προγράμματος, ώστε να επιβεβαιωθεί η ορθότητα των υπολογισμών. Σε περίπτωση λάθους ανατίθεται μια νέα παράλληλη μονάδα, αφού πρώτα έχει παρουσιαστεί ανάλογο μήνυμα αποτυχίας για την συγκεκριμένη μονάδα. Έτσι δεν σταματά η διαθεσιμότητα του συστήματος.

Παραπάνω μιλήσαμε για τις παράλληλες μορφές σύνδεσης που εφαρμόστηκαν σε μετέπειτα χρονολογικά στάδιο στις αρχιτεκτονικές των συστημάτων λογισμικού. Πρώτα είχαν εφαρμοστεί σειριακές αρχιτεκτονικές διασύνδεσης των εφεδρικών-πλεοναζόντων προγραμμάτων. Η σειριακή επιλογή είναι η μέθοδος επικύρωσης της ορθής λειτουργίας ενός προγράμματος. Στην περίπτωση που είναι ελαττωματικό, επιλέγεται ένα νέο και στην περίπτωση που ξανά αποτύχει, επιλέγεται καινούργιο μέχρι να μην υπάρχουν αλλά διαθέσιμα.

Πολλές φορές γίνεται η χρήση υβριδικών τεχνικών για την σύνδεση των προγραμμάτων για την εκμετάλλευση των ιδιοτήτων που προσφέρουν οι

παράλληλες και σειριακές τεχνικές. Οπότε εξαρτάται από το βαθμό του πλεονασμού που επιθυμεί ο σχεδιαστής ενός συστήματος λογισμικού να αποδώσει σε αυτό. Η πρόσθεση πλεονασμού με την μορφή λογισμικού αυξάνει το κόστος υλοποίησης και συντήρησης οπότε πρέπει να χρησιμοποιείται στα απαραίτητα επίπεδα, γιατί ένα σύστημα με επιπλέον κώδικα και προγράμματα αυξάνει την πολυπλοκότητα του συστήματος στο σύνολο του.



Εικόνα 5.1: Βασικές αρχιτεκτονικές διασύνδεσης πλεονάζοντων εξαρτημάτων. Παράλληλη επικύρωση (Α), Παράλληλη επιλογή (Β), Σειριακή προσέγγιση (Γ)

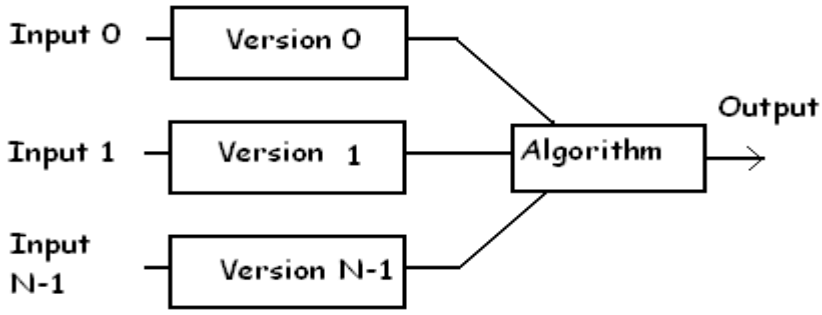
5.1 N-Version Programming

Η μεθοδολογία αυτή αποτελεί μια από τις κλασσικές μεθόδους πλεονασμού για συστήματα ανοχής σφαλμάτων λογισμικού. Η λογική του στηρίζεται στην ανάπτυξη αρκετών προγραμμάτων, που σχεδιάζονται ανεξάρτητα και εκτελούνται παράλληλα. Τα αποτελέσματα τους συγκρίνονται ώστε να εντοπιστούν πιθανά λάθη των εξόδων και με τη σειρά τους να αποφευχθούν. Οι πολλαπλές εκδόσεις λογισμικού πρέπει να διαφέρουν μεταξύ τους όσο το δυνατόν περισσότερο, όσον αφορά, το σχεδιασμό, την υλοποίηση των εργασιών τους και τα εργαλεία που χρησιμοποιούν. Αυτό γίνεται εφικτό με την εφαρμογή διαφορετικών γλωσσών προγραμματισμού και αλγορίθμων, ανάλογα πάντα με την εφαρμογή που εξυπηρετούν. Η τεχνική αυτή ονομάζεται *σχεδιαστική ποικιλομορφία λογισμικού* και διασφαλίζει την λειτουργία ενός συστήματος λογισμικού σε περίπτωση αποτυχίας των διάφορων εκδόσεων από κοινή αιτία.

Κάθε πρόγραμμα υλοποιεί την ίδια υπηρεσία και οι έξοδοι τους συγκρίνονται σε έναν συμψηφιστή λογισμικού, ο οποίος χρησιμοποιεί έναν αλγόριθμο πλειοψηφίας για τα αποτελέσματα. Οπότε το πλήθος των προγραμμάτων καθορίζει και το ποσοστό της αξιοπιστίας του συστήματος. Ένα απλό παράδειγμα αποτελεί η υλοποίηση 3VP(Triple Version Programming) η οποία χρησιμοποιεί τρείς εκδόσεις προγραμμάτων και αποφεύγεται η αποτυχία του συστήματος για σφάλμα μιας

έκδοσης. Το μειονέκτημα όμως είναι ότι για την υλοποίηση των εκδόσεων χρειάζονται εξειδικευμένες γνώσεις από πολλούς ανθρώπους. Διότι έτσι αυξάνεται και η πολυπλοκότητα του συνολικού έργου. Επιπλέον το έργο αυτό θα έχει διαφορετικούς οδηγούς συντήρησης για κάθε έκδοση που σημαίνει ότι χρειάζονται και ανάλογα άτομα που να γνωρίζουν τους τρόπους αντιμετώπισης σε μια πιθανή αποτυχία. Όμως παρόλο αυτά ο γενικός κανόνας για την ανοχή K αποτυχιών είναι η ανάπτυξη 2K+1 εκδόσεων λογισμικού.

Οι μηχανισμοί της διαδικασίας αυτής έχουν επεκταθεί και σε άλλα πεδία με την εξέλιξη της τεχνολογίας, όπως ο σχεδιασμός διαδικτυακών συστημάτων και τις κατανεμημένες εφαρμογές προσφοράς υπηρεσιών. Ένα σύστημα που προσφέρει τέτοιες δυνατότητες ονομάζεται WS-FTM [3]. Υποστηρίζει την παράλληλη εκτέλεση αρκετών ανεξάρτητων εκδόσεων, υλοποιώντας την ίδια υπηρεσία και η συμφωνία των αποτελεσμάτων καθορίζεται από ερωτήματα. Μια ακόμη αποτελεί η WS-BPEL [4] , που υλοποιεί την ίδια λειτουργικότητα όμως στο τέλος εφαρμόζεται ψηφοφορία πλειοψηφίας με έναν συμψηφιστή λογισμικού. Ένα σύστημα αναπτύχθηκε πάλι με την ίδια λογική όμως εξυπηρετούσε την εφαρμογή NVP σε διακομιστές SQL[5]. Στην περίπτωση αυτή, υπάρχει ένα πλεονέκτημα καθώς μια βάση δεδομένων είναι αυστηρά καθορισμένη και υπάρχουν διαθέσιμοι τρόποι υλοποίησης. Ωστόσο η συμφωνία σε συγκεκριμένη συνθήκη από πολλούς διακομιστές δεν είναι ακανόνιστη. Χρησιμοποιούνται τεχνικές σύγχρονου προγραμματισμού των εργασιών και άλλες μορφές μη ντετερμινιστικές. Η μορφή αυτή πλεονασμού μπορεί να αντιμετωπίσει αναπτυξιακά σφάλματα και ορισμένα φυσικά σφάλματα με το συνδυασμό συγκεκριμένου υλικού για κάθε έκδοση.



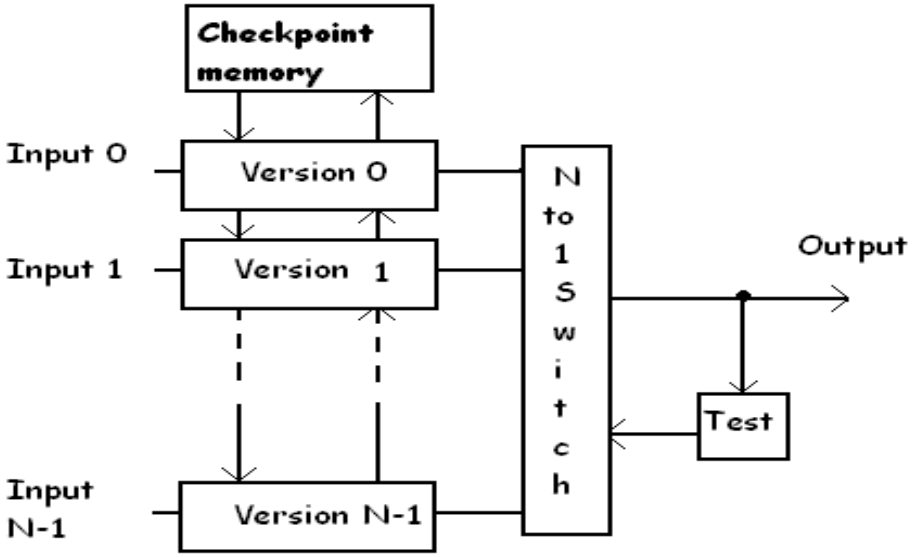
Εικόνα 5.2: Η τεχνική N-εκδόσεων λογισμικού.

5.2 Recovery Blocks

Η μορφή αυτή πλεονασμού είχε προταθεί από τον Randell [6] και στηρίζεται στην ποικιλομορφία των προγραμμάτων για την προσφορά της ίδιας υπηρεσίας. Η διαφορά του από το NVP είναι ότι οι πολλαπλές εκδόσεις εκτελούνται σειριακά και όχι παράλληλα. Όταν μια έκδοση αποτύχει, μια άλλη εφεδρική αναλαμβάνει την εκτέλεση. Αν και η επόμενη εφεδρική αποτύχει, τότε επιλέγεται εκ νέου μια νέα εφεδρική μονάδα και εφόσον οι αποτυχίες συνεχιστούν, το σύστημα θα λειτουργήσει ως το σημείο εκείνο που να μην υπάρχει άλλη εφεδρική μονάδα να το υποστηρίξει. Αυτές οι εφεδρικές μονάδες αναγνωρίζουν τις αποτυχίες με την εκτέλεση ελέγχων επιβεβαίωσης και αξιοποιούν έναν μηχανισμό επιστροφής (rollback) κάθε φορά για

την επαναφορά του συστήματος σε προγενέστερη κατάσταση ελεύθερη από σφάλματα. Έτσι αντιμετωπίζεται και η αδιάκοπη αλλαγή σε εφεδρικές μονάδες σε συνεχόμενες αποτυχίες.

Οι εφεδρικές μονάδες ως τεχνική έχει επεκταθεί σε αρκετά συστήματα πραγματικού χρόνου και στα κατανεμημένα συστήματα υπολογιστών. Η ιδιότητα της επαναπροσπάθειας με επιστροφή αποτελεί βασικό μέτρο ανοχής σφαλμάτων και ενσωματώνεται κατά την σχεδίαση του λογισμικού. Τέλος είναι κατάλληλα να αντιμετωπίσουν σφάλματα ανάπτυξης αλλά όχι τόσο τα φυσικά σφάλματα τα οποία δύναται να συμβούν καθώς αποκλείουν την περίπτωση της παράλληλης εκτέλεσης.



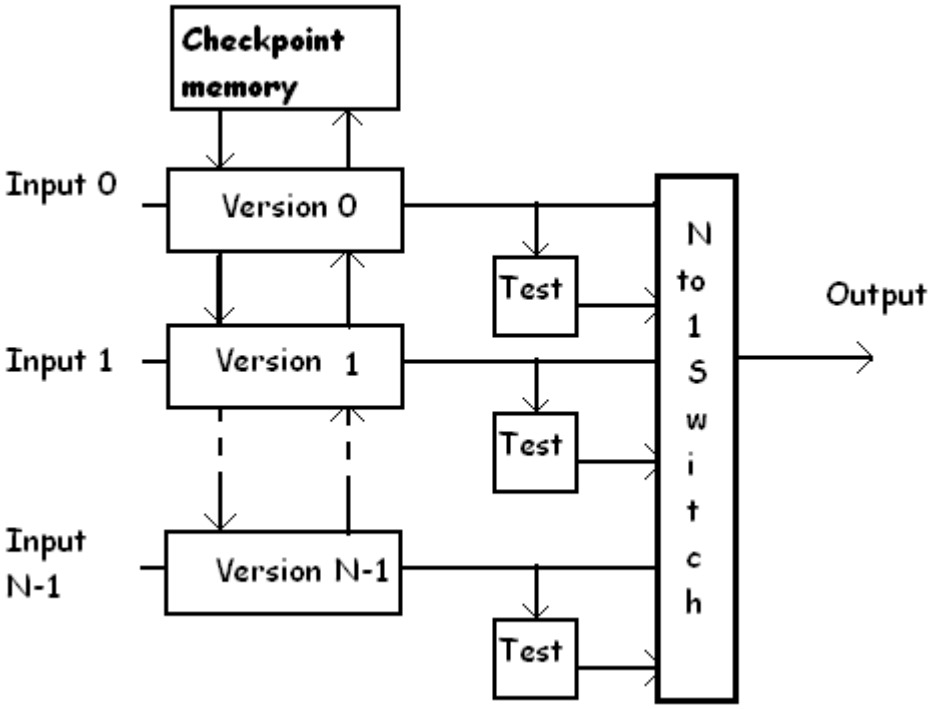
Εικόνα 5.3: Η τεχνική των Recovery Blocks.

5.3 Self-checking programming

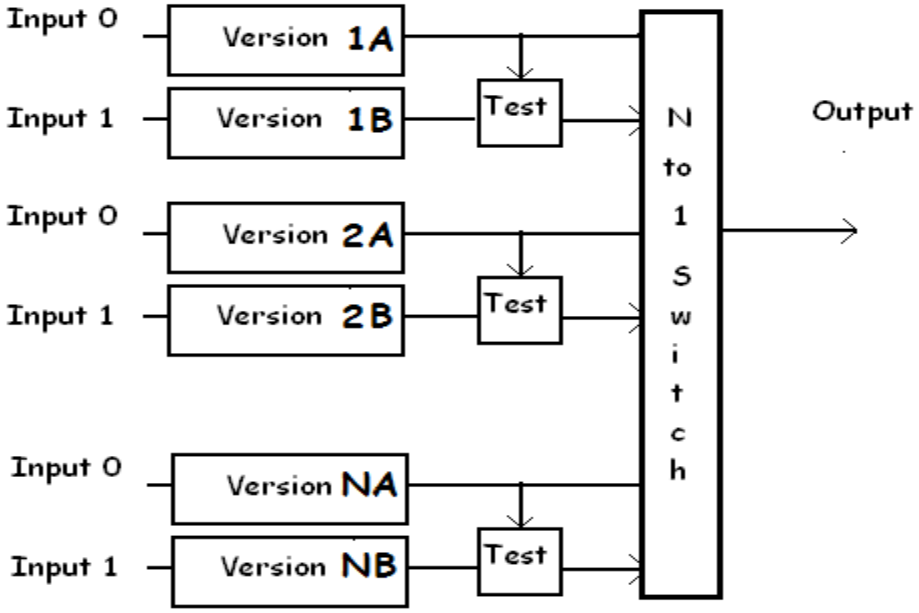
Για τον έλεγχο της κατάστασης του λογισμικού κατά το χρόνο εκτέλεσης ο Laprie [17] πρότεινε την επέκταση των δύο προηγουμένων τεχνικών, έτσι αναπτύχθηκε μια υβριδική προσέγγιση. Τα προγράμματα έγιναν αυτό-ελεγχόμενα με την προσθήκη επιπλέον κώδικα για τον έλεγχο της δυναμικής τους συμπεριφοράς κατά την λειτουργία τους. Μια αυτοελεγχόμενη μονάδα μπορεί να είναι λογισμικό που ενσωματώνει μηχανισμούς επιβεβαίωσης, ανάλογους με αυτές για τις εφεδρικές μονάδες ή ένα ζεύγος από ανεξάρτητες μονάδες, που εκτελούν σύγκριση των εξόδων τους, όπως στο NVP. Κάθε μια από τις παραπάνω περιπτώσεις πρέπει να αποτελείται τουλάχιστον από δύο μονάδες οι οποίες σχεδιάζονται διαφορετικά, αλλά εκτελούνται παράλληλα. Αν μια η κύρια μονάδα αποτύχει τότε αυτόματα γίνεται έλεγχος του αποτελέσματος από της εναλλακτικής μονάδας για την επιλογή σωστού αποτελέσματος.

Κατά τον χρόνο εκτέλεσης η διαδικασία αυτή γίνεται με τις μονάδες που εκτελούν τον υπολογισμό και τις μονάδες που αναμένουν την περίπτωση σφάλματος σε συνθήκες παραλληλίας. Όταν μια κύρια μονάδα αποτυγχάνει, αντικαθίσταται από μια εφεδρική. Έτσι δεν είναι απαραίτητο να χρησιμοποιηθεί ένας μηχανισμός επιστροφής. Η βασική ιδέα των αυτό-ελεγχόμενων μονάδων λογισμικού ήταν από τις

πρώτες μεθόδους έλεγχου της συμπεριφοράς και κατοχύρωσης της αξιοπιστίας των συστημάτων λογισμικού[7]. Ο Dobson [8] αναφέρει την χρήση αυτοελεγχόμενου προγραμματισμού για εφαρμογές προσφοράς υπηρεσιών, με την παράλληλη κλήση διεργασιών και την επιλογή αποτελέσματος από τις εφεδρικές μόνο στην περίπτωση αποτυχίας. Οι δυνατότητες του προσφέρει είναι η αντιμετώπιση αναπτυξιακών σφαλμάτων.



Εικόνα 5.4 Η τεχνική των αυτοελεγχόμενων μονάδων.



Εικόνα 5.5: Η τεχνική των αυτοελεγχόμενων μονάδων με σύγκριση.

5.4 Self-optimizing code

Ο όρος αυτο-βελτιώσιμος κώδικας [15] χρησιμοποιείται στα αυτο-διαχειριζόμενα συστήματα, αναφέρεται στην αυτοματοποιημένη αντίδραση ενός συστήματος με σκοπό την υποβοήθηση ή επαναφορά της απόδοσης του. Η υλοποίηση ίδιων λειτουργιών με αρκετά διαφορετικά τμήματα-μονάδες, τα οποία είναι βελτιστοποιημένα για ανάλογες καταστάσεις λειτουργίας. Οι εφαρμογές μπορούν να προσαρμοστούν σε διαφορετικές προδιαγραφές απόδοσης και συνθήκες εκτέλεσης κατά τον χρόνο λειτουργίας τους (runtime), με την επιλογή και ενεργοποίηση των κατάλληλων μονάδων ανάλογα με το περιβάλλον λειτουργίας. Μετέπειτα εφάρμοστηκαν στο πεδίο των Web υπηρεσιών.

5.5 Exception Handling και rule engines-registries

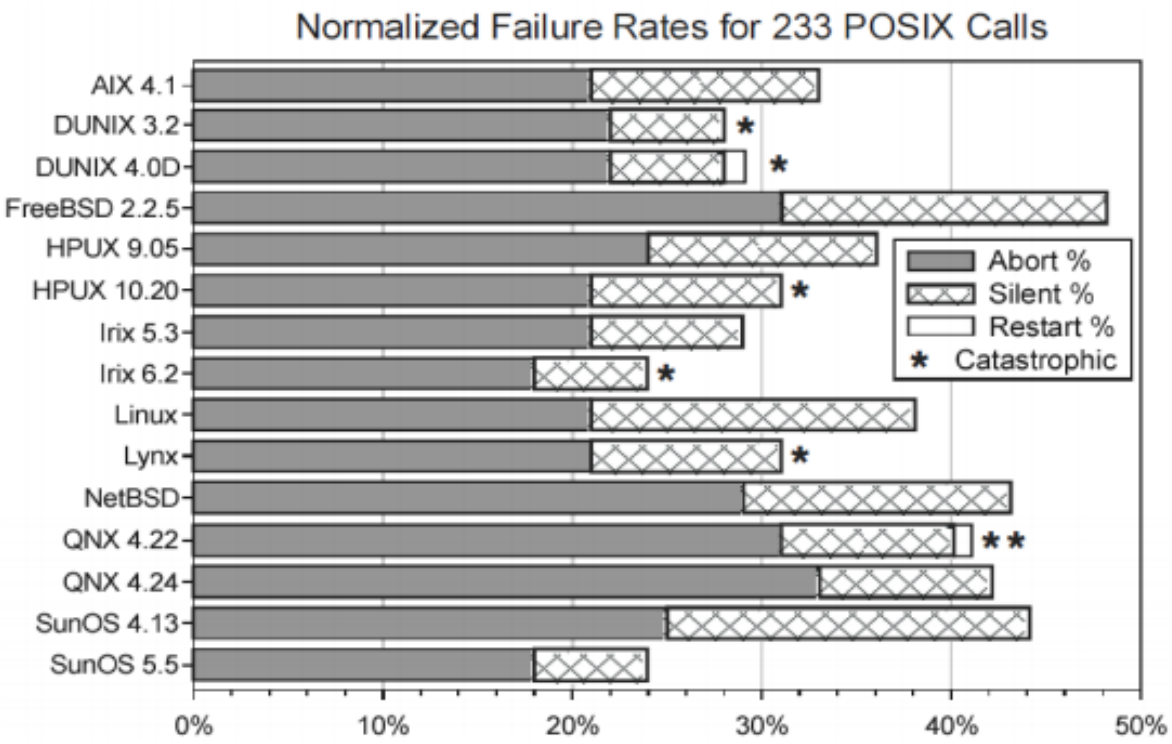
Η διαχείριση εξαιρέσεων αποτελεί την μέθοδο [8]κατασκευής ενός συστήματος με δυνατότητες εντοπισμού και επαναφοράς από ασυνήθιστες συνθήκες. Μερικές από αυτές είναι ενδεικτικά, τα ελαττώματα σχεδίασης λογισμικού, οι λάθασμένοι είσοδοι από τον χρήστη και περιβαλλοντολογικές μεταβολές. Οι χειριστές εξαιρέσεων είναι μηχανισμοί που μελετούν τον κώδικα προγραμμάτων με σκοπό να παγιδεύσουν προκαθορισμένες κατηγορίες σφαλμάτων και να ξεκινούν τις διαδικασίες επιδιόρθωσης τους κατά τον χρόνο σχεδίασης. Οι μηχανές κανόνων, που είναι γνωστές και ως μητρώα, επεκτείνουν τις δυνατότητες των χειριστών εξαιρέσεων ενσωματώνοντας ένα μητρώο από κανόνες με προκαθορισμένες διορθωτικές κινήσεις για την αντιμετώπιση κάθε πιθανού σφάλματος. Το μητρώο συμπληρώνεται κατά την σχεδίαση από τους προγραμματιστές και περιέχει μια λίστα από αποτυχίες με την ανάλογο μέθοδο επίλυσης της για την εκτέλεση της κατά τον χρόνο λειτουργίας του προγράμματος. Ο τρόπος με τον οποίο γίνεται πρακτικά γίνεται με την προσθήκη επιπλέον κώδικα όπως μπορεί να μελετηθεί στο [18] για την υλοποίηση στην γλωσσα C. Περίπου τα δύο τρίτα του κώδικα που θα γραφούν για ένα σύστημα αφορά την διαχείριση εξαιρέσεων. Οι αποτυχίες που οφείλονται σε τέτοιες αιτίες καταλαμβάνουν τα δυο τρίτα των συγκρούσεων-κολλημάτων λογισμικού και πενήντα τοις εκατό των ατελειών ασφάλειας[9]. Με τις μεθόδους αυτούς αντιμετωπίζονται σφάλματα ανάπτυξης και αλληλεπιδράσεων.

5.6 Wrappers

Ο όρος wrapper, που στα ελληνικά σημαίνει περιτύλιγμα, δείχνει στοιχεία που μεσολαβούν στις αλληλεπιδράσεις μεταξύ των συστατικών ενός συστήματος για να λύσουν προβλήματα ένταξης μεταξύ των τμημάτων κώδικα ενός συστήματος.

Οι Popov κ.α [10] προτείνουν περιτυλίγματα στο πλαίσιο του σχεδιασμού των συστημάτων που ενσωματώνουν συστατικά COTS να αντιμετωπίσουν προβλήματα ένταξης που προκύπτουν από ελλείπεις προδιαγραφές. Ατελώς καθορισμένα COTS συστατικά μπορούν να χρησιμοποιηθούν λανθασμένα ή σε περιπτώσεις που διαφέρουν από αυτές για τις οποίες έχουν σχεδιαστεί. Τα περιτυλίγματα που προτείνει ο Popov κ.ά.. ανιχνεύουν κλασσικές αναντιστοιχίες και ενεργοποιεί τις

κατάλληλες ενέργειες αποκατάστασης, για παράδειγμα, αλλάζουν σε εφεδρικά περιττά εξαρτήματα. Οι Chang κ.α[11] απαιτούν από τους προγραμματιστές να απελευθερώσουν τα εξαρτήματα με βλάβες και ταυτόχρονα να ενεργοποιηθούν οι διορθωτικοί μηχανισμοί που θα ασχοληθούν με αποτυχίες, από κοινές καταχρήσεις των εξαρτημάτων. Η διαδικασία αυτή είναι αυτοματοποιημένη. Ο Sales [12] πρότεινε περιτυλίγματα (wrappers) για off-the-shelf συστατικά στοιχεία των λειτουργικών συστημάτων. Τα περιτυλίγματα βελτιώνουν την αξιοπιστία των πυρήνων των λειτουργικών συστημάτων, που ενσωματώνουν συστατικά COTS με διαφορετικά επίπεδα αξιοπιστίας. Ο Fetzer[13] πρότεινε θεραπευτές, που είναι περιτυλίγματα που ενσωματώνουν όλες τις κλήσεις συναρτήσεων της βιβλιοθήκης C, που γράφουν στο σωρό και να πραγματοποιούν τους κατάλληλούς ελέγχους για την πρόληψη υπερχειλίσεων μνήμης. Έτσι λοιπόν εισάγεται σκόπιμα περιττός κώδικας για να προληφθούν πιθανές αποτυχίες από Bohrbugs και κακόβουλες επιθέσεις.



Εικόνα 5.6: Συγκριση ρυθμών αποτυχιας σε συστηματα POSIX.

5.7 Data structures and redundancy

Στις αρχές της δεκαετίας του εβδομήντα αναπτύχθηκαν συστήματα λογισμικού που είχαν δυνατότητες έλεγχου της ακεραιότητας του συστήματος κατά τον χρόνο λειτουργίας. Επίσης αναπτύχθηκαν και οι τεχνικές πλεονασμού στα δεδομένα, εισάγοντας πολύπλοκες δομές δεδομένων ως μέτρο ασφάλειας. Αυτές αποτελούνταν από επιπλέον κώδικα για την ανίχνευση του πλήθους των κόμβων, στις δομές δεδομένων, και την απόδοση αναγνωριστικών σε κάθε νέο, ώστε να υπάρχει ευρωστία. Αυτές οι τεχνικές σκόπιμα προσθέτουν κώδικα για να αντιμετωπίσουν σφάλματα σχεδίασης. Μια άλλη τεχνική είναι η χρήση πλεονασμού στα δεδομένα για την αντιμετώπιση επιθέσεων από μη εξουσιοδοτημένους χρήστες. Εφαρμόζεται σε

συστήματα λογισμικού που περιέχουν σφάλματα τα οποία ενεργοποιούνται με την λανθασμένη είσοδο δεδομένων σε και αποφέρουν την αποτυχία. Ο τρόπος είναι ο έξης, το σύστημα έχει προκαθορισμένες δομές δεδομένων για κάθε είσοδο, οπότε στην περίπτωση λάθους γίνεται διόρθωση της εισόδου αυτόματα και το σύστημα συνεχίζει την λειτουργία του. Η αξιοπιστία της διόρθωσης δεν είναι πάντα πίστη, όμως είναι σε αποδεκτά επίπεδα ώστε να διαμορφωθεί αποδεκτή έξοδος.

Οι τεχνικές αυτές προέκυψαν από τον NVP και τα recovery blocks, απλώς όταν χρησιμοποιηθούν στα δεδομένα ονομάζονται retry blocks και N-copy programming. Πρόσφατα Knight κ.ά. [14] επέκτειναν την ποικιλομορφία των δεδομένων για την αντιμετώπιση προβλημάτων ασφάλειας. Χρησιμοποιεί N-παραλλαγές από εικασίες, για την παροχή υψηλής αξιοπιστίας, εναντίον μιας κατηγορίας επιθέσεων που μπορεί να δεχτεί ένα σύστημα. Τα δεδομένα μετατρέπονται σε παραλλαγές με την ιδιότητα ότι πανομοιότυπες τιμές δεδομένων έχουν διαφορετικές ερμηνείες. Με τον τρόπο αυτό οι εισβολείς θα χρειαστεί να αλλάξουν τα αντίστοιχα δεδομένα σε κάθε παραλλαγή με διαφορετικό τρόπο κατά την αποστολή των ίδιων εισόδων σε όλες τις παραλλαγές.

5.7 Genetic Programming

Πρόσφατα διερευνήθηκε η χρήση γενετικού προγραμματισμού ως τρόπο διόρθωσης λαθών λογισμικού. Και οι δύο προσεγγίσεις υποθέτουν την ύπαρξη ενός συνόλου από περιπτώσεις δοκιμών που θα χρησιμοποιηθούν ως βασικές για την εξαγωγή αποτελεσμάτων. Όταν το λογισμικό αποτύχει, το πλαίσιο εκτέλεσης δημιουργεί αυτόματα ένα πληθυσμό παραλλαγών του αρχικού ελαττωματικού προγράμματος. Έπειτα οι γενετικοί αλγόριθμοι εξελίσσουν τον αρχικό πληθυσμό, που καθοδηγείται από τα αποτελέσματα της προηγούμενης δοκιμής και επιλέγουν μια νέα έκδοση του προγράμματος η οποία περιέχει λιγότερα λάθη από την αρχική.

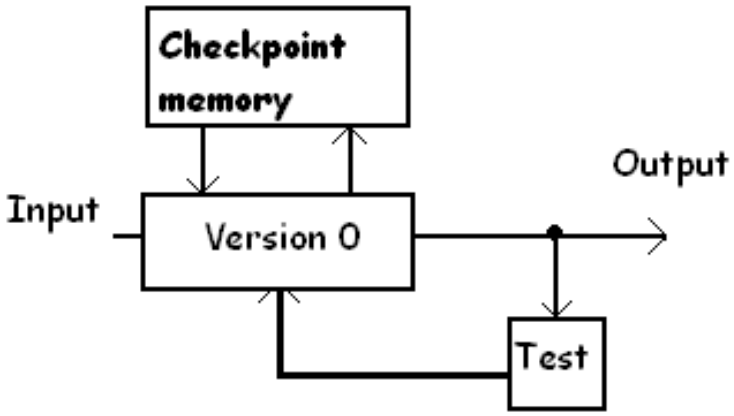
Ο γενετικός προγραμματισμός δεν προϋποθέτει τη σκόπιμη ανάπτυξη επιπλέον λειτουργικότητας, αλλά εκμεταλλεύεται τον κώδικα ώστε να παράγει παραλλαγές των αρχικών προγραμμάτων και επιλέξει μια "σωστή" παραλλαγή. Αυτή η γενετική προσέγγιση αντιδρά στις αδυναμίες που εντοπίζονται από σουίτες δοκιμών, κατά την σχεδίαση και είναι κατάλληλοι για τον εντοπισμό και διόρθωση Bohrbugs.

Χρησιμοποιώντας ως αναλογία αυτή των αλλεργιών σε ανθρώπους και ζώα, και της θεραπείας τους. Υπάρχουν μηχανισμοί επαναφοράς με μερική ή εκ νέου εκτέλεση προγραμμάτων, όταν αποτυγχάνουν υπό διαφοροποιημένες συνθήκες του περιβάλλοντος εκτέλεσης, για να ανακτηθεί η ντετερμινιστική φύση του λογισμικού και να γίνει η επαναφορά από μη ντετερμινιστικά σφάλματα [14].

Ο μηχανισμός βασίζεται σε αλλαγές του περιβάλλοντος, που αφορούν διαφορετικές στρατηγικές διαχείρισης της μνήμης και ανάγνωσης μηνυμάτων μεταβολή της προτεραιότητας διεργασιών και μείωσης των αιτημάτων των χρηστών. Αυτές οι αλλαγές στο περιβάλλον εκτέλεσης μπορεί να δημιουργήσουν αποτυχίες, όπως υπερχειλίσεις μνήμης και αδιέξοδα, προβλήματα συνταύτισης, σφάλματα αλληλεπίδρασης από κακόβουλα αιτήματα. Παρόλα αυτά αποτελούν ένα καινοτόμο και νέο τρόπο διαχείρισης αποτυχιών.

5.7 Checkpointing and recovery

Οι τεχνικές αυτές καθορίζουν κατά την περίοδο λειτουργίας σημεία ελέγχου, που είναι καταστάσεις του λογισμικού, που αποθηκεύονται ώστε να υπάρχει η δυνατότητα επαναφοράς σε μια από αυτές σε περίπτωση μετέπειτα αποτυχίας[]. Όταν το σύστημα αποτυγχάνει, επαναφέρεται πίσω σε προγενέστερη σταθερή κατάσταση και επανεκλεεί τις πράξεις, να λύσει προσωρινά προβλήματα που έχουν προκληθεί από τυχαίες, μεταβατικές συνθήκες στο περιβάλλον του συστήματος. Αυτές οι προσεγγίσεις καιροσκοπικά εκμεταλλεύονται το περιβάλλον , δεδομένου ότι σύστημα θα εκτελέσει ξανά τον ίδιο κώδικα, χωρίς να προσπαθεί να τροποποιήσει το περιβάλλον, αλλά αντίθετα βασίζεται σε αυθόρμητες αλλαγές στο περιβάλλον για να αποφευχθεί οι συνθήκες που δημιούργησε τη αποτυχία. Αυτές οι τεχνικές είναι αποτελεσματικές στην αντιμετώπιση Heisenbugs, που εξαρτώνται από τις συνθήκες προσωρινής εκτέλεσης, αλλά δεν λειτουργούν καλά για Bohrbugs που εξακολουθούν να υπάρχουν στον κώδικα και στο περιβάλλον εκτέλεσης.



Εικόνα 5.7: Η τεχνική των αυτοελεγχόμενων μονάδων με σύγκριση.

Ανακεφαλαιώνοντας, τόσο οι κοινότητες της ανοχής σφαλμάτων και της αυτό-διόρθωσης βλαβών εργάζονται σε νέες τεχνικές για τη μείωση των επιπτώσεων των σφαλμάτων, κατά την εκτέλεση του λογισμικού, ώστε να εξασφαλίζεται αξιοπιστία και στην παρουσία των σφαλμάτων εκτέλεσης του λογισμικού. Οι τεχνικές που ερευνήθηκαν μέχρι αντιμετωπίζουν διάφορα προβλήματα, εκτελούνται υπό διαφορετικές παραδοχές και ασχολούνται με διάφορες μορφές επιπτώσεων, τόσο για την ανάπτυξη όσο και για την εκτέλεση του λογισμικού. Η αντιμετώπιση των διαφορετικών τύπων από εφαρμογές, μπορεί να επιλύσουν θέματα της αρχιτεκτονικής των συστημάτων λογισμικού. Οι προσπάθειες να πλαισιωθούν οι προσεγγίσεις κάτω από μια ενοποιημένη άποψη, έχουν κατηγοριοποιηθεί είτε στην ανοχή σφαλμάτων είτε στις αυτό-θεραπευτικές τεχνικές, γεγονός το οποίο δημιουργεί σύγχυση μεταξύ των δύο κοινοτήτων. Έτσι, χάνονται πολλές φορές σημαντικές σχέσεις ενός έργου λογισμικού, σε δύο τομείς που είναι αυστηρά αλληλένδετες και αποτελούν όψεις του ίδιου νομίσματος.

	Intention	Type	Adjudicator	Faults
N-version programming	deliberate	code	reactive implicit	development
Recovery blocks	deliberate	code	reactive explicit	development
Self-checking programming	deliberate	code	reactive expl./impl.	development
Self-optimizing code	deliberate	code	reactive explicit	development
Exception handling, rule engines	deliberate	code	reactive explicit	development
Wrappers	deliberate	code	preventive	Bohrbugs malicious
Robust data structures, audits	deliberate	data	reactive implicit	development
Data diversity	deliberate	data	reactive expl./impl.	development
Data diversity for security	deliberate	data	reactive implicit	malicious
Rejuvenation	deliberate	environment	preventive	Heisenbugs
Environment perturbation	deliberate	environment	reactive explicit	development
Process replicas	deliberate	environment	reactive implicit	malicious
Dynamic service substitution	opportunistic	code	reactive explicit	development
Fault fixing, genetic programming	opportunistic	code	reactive explicit	Bohrbugs
Automatic workarounds	opportunistic	code	reactive explicit	development
Checkpoint-recovery	opportunistic	environment	reactive explicit	Heisenbugs
Reboot and micro-reboot	opportunistic	environment	reactive explicit	Heisenbugs

Εικόνα 5.8: Συνοπτικός πίνακας με αρκετές μεθόδους ανοχής σφαλμάτων λογισμικού [15] .

ΚΕΦΆΛΑΙΟ 6 Time Redundancy

Ο χρονικός πλεονασμός αφορά τις τεχνικές διαχείρισης του χρόνου σε ένα υπολογιστικό σύστημα. Ο χρόνος είναι απαραίτητος για τον καθορισμό εργασιών προς εκτέλεση καθώς και για τον προγραμματισμό γεγονότων, που πρέπει να γίνουν από το σύστημα. Ακόμη είναι χρήσιμος για τον συγχρονισμό μηνυμάτων που μεταδίδονται σε ένα δίκτυο υπολογιστών, σε μεγαλύτερη κλίμακα, και επιπλέον για την ενδοεπικοινωνία εξαρτημάτων, σε μικρότερη κλίμακα, για την υλοποίηση υπολογισμών.

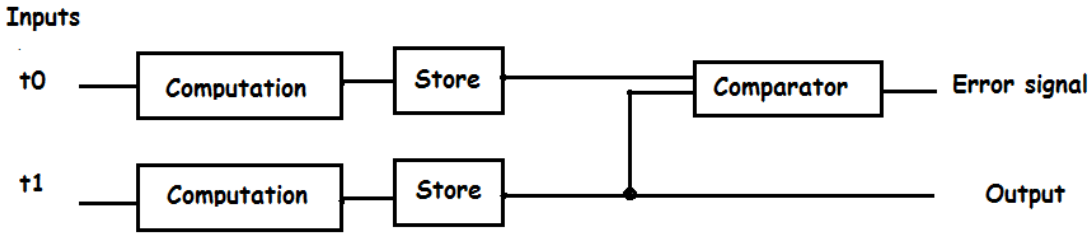
Τα σφάλματα που μπορεί να αντιμετωπίσει είναι δύο ειδών τα *περιστασιακά* και τα *μόνιμα σφάλματα (transient and permanent faults)*. Τα περιστασιακά σφάλματα μπορούν να συμβούν μια τυχαία χρονική στιγμή αλλά με την επανάληψη του υπολογισμού να εξαλείφουν από το σύστημα. Σε αντίθεση με αυτά τα μόνιμα σφάλματα επιφέρουν την αποτυχία του συστήματος. Όμως πρέπει να διευκρινιστεί και ένα από τα ελαττώματα του χρονικού πλεονασμού. Ο επαναυπολογισμός προϋποθέτει την διαθεσιμότητα των εισόδων κάθε χρονική στιγμή, πράγμα που δεν είναι δεδομένο. Υπάρχει η περίπτωση περιστασιακού σφάλματος που μπορεί να επιφέρει την αποτυχία του συστήματος, αν τα δεδομένα, που είναι απαραίτητα για τον υπολογισμό, δεν είναι πλέον διαθέσιμα με την μορφή εισόδου ή αντιγράφου.

Είναι χρήσιμη σαν μέθοδος επειδή μπορεί να αντικαταστήσει την επιπλέον προσθήκη υλικού ανοχής σφαλμάτων με απλές διαδικασίες επανυπολογισμού των αποτελεσμάτων. Τα πλεονεκτήματα που προσφέρει σαν διαδικασία είναι η απλοποίηση των κυκλωμάτων και η μείωση των εξόδων κατασκευής του συστήματος

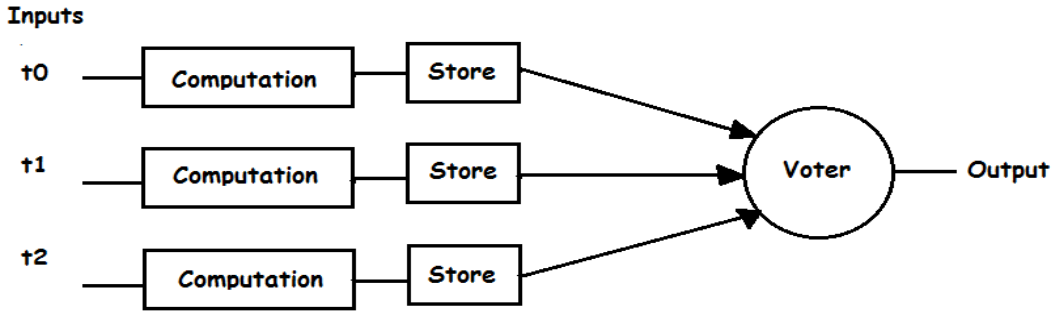
Το μειονέκτημα είναι ότι θα χρειαστούν παραπάνω κύκλοι εκτέλεσης, άρα και επιπλέον χρόνο για την παραγωγή του ίδιου αποτελέσματος. Το πρόβλημα αυτό όμως δεν είναι και τόσο τραγικό καθώς αν συνδυαστεί η τεχνική αυτή με κατάλληλες αρχιτεκτονικές διασωλήνωσης εργασιών ελαχιστοποιείται σε μεγάλο βαθμό ο χρόνος του επαναυπολογισμού. Χρησιμοποιεί έννοιες από τον πλεονασμό πληροφορίας που αναλύθηκε σε προηγούμενη ενότητα, όπως κωδικοποιητές και αποκωδικοποιητές, καθώς και ορισμένους κώδικες εντοπισμού και επιδιόρθωσης.

Έτσι λοιπόν στην ενότητα αυτή θα αναφερθούν οι τρόποι αντιμετώπισης των περιστασιακών και μόνιμων σφαλμάτων αλλά και κάποιες αξιόλογες διαδικασίες χρονικού πλεονασμού, μαζί με αναλυτικά παραδείγματα προς κατανόηση.

Τα περιστασιακά σφάλματα αντιμετωπίζονται με την επανάληψη των υπολογισμών ή της μετάδοσης της πληροφορίας και την μετέπειτα σύγκριση τους με προσωρινά αποθηκευμένα αντίγραφα αυτών. Αν ο υπολογισμός έχει υποστεί αλλοίωση από λάθος την χρονική στιγμή t0, το αντίγραφό του θα αποκαλύψει το πρόβλημα αυτό με την επανάληψή του την χρονική στιγμή t1. Ο συγκριτής θα παράξει το κατάλληλο σήμα λάθους και με τον τρόπο αυτό θα εντοπιστεί. Οπότε χρειάζονται δύο υπολογισμοί για τον εντοπισμό ενός σφάλματος όπως φαίνεται στο σχήμα 6.1. Αν ο υπολογισμός πραγματοποιηθεί τρεις φορές τότε θα υπάρχει η δυνατότητα επιδιόρθωσις του όπως φαίνεται και στο σχήμα 6.2.



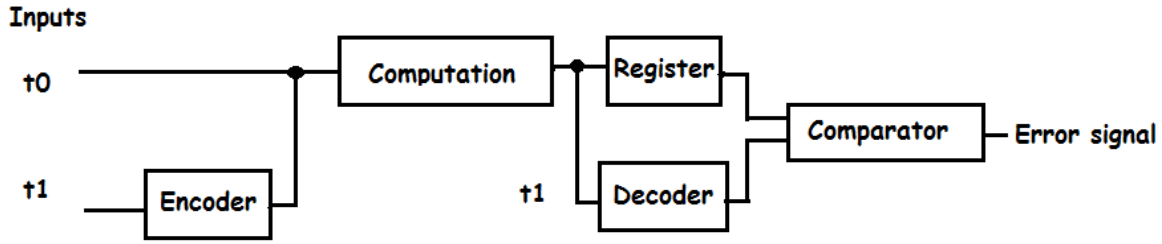
Εικόνα 6.1: Μέθοδος εντοπισμού περιστασιακών σφαλμάτων με χρονικό πλεονασμό



Εικόνα 6.2: Μέθοδος επιδιόρθωσης περιστασιακών σφαλμάτων με χρονικό πλεονασμό

Με τον χρονικό πλεονασμό γίνεται και ο κατάλληλος χαρακτηρισμός ενός σφάλματος και οι κατάλληλες ενέργειες για την επαναφορά του συστήματος. Αν μετά από επανάληψη του υπολογισμού το σύστημα συνεχίζει την ορθή του λειτουργία, τότε υπέστη ένα περιστασιακό σφάλμα και δεν χρειάζονται περαιτέρω ενέργειες για την αντιμετώπισή του, όπως για παράδειγμα αντικατάσταση του με νέα εφεδρική μονάδα υλικού. Παραμένει σε λειτουργία για την εξοικονόμηση πόρων του συστήματος και για την αύξηση της διαθεσιμότητας του.

Για την αντιμετώπιση των μόνιμων σφαλμάτων γίνεται η χρήση κωδικοποίησης των δεδομένων. Η κωδικοποίηση μπορεί να γίνει με κυκλώματα η με κώδικα λογισμικού ανάλογα με την εφαρμογή. Οι κώδικες εντοπισμού EDC και επιδιόρθωσης ECC, είναι τα μέσα για την επίτευξη αξιόπιστων υπολογισμών και των κατάλληλων ενεργειών συντήρησης σε ένα σύστημα. Ο υπολογισμός στην περίπτωση αυτή πραγματοποιείται από την λειτουργική μονάδα την χρονική στιγμή t0 και αποθηκεύεται προσωρινά σε έναν καταχωρητή ή μια μεταβλητή σε συστήματα λογισμικού. Το επόμενο βήμα είναι η επανάληψη του υπολογισμού με τα ίδια δεδομένα την χρονική στιγμή t1 αφού πρώτα έχουν κωδικοποιηθεί με κωδικοποιητή. Μετά την εκτέλεση του υπολογισμού η κωδικολέξη αποκωδικοποιείται και συγκρίνεται με το καταχωρημένο αποτέλεσμα. Ο παραπάνω τρόπος αποτελεί έναν γενικό τρόπο αντιμετώπισης μονίμων σφαλμάτων με χρονικό πλεονασμό, όπως φαίνεται στο σχήμα 6.3.



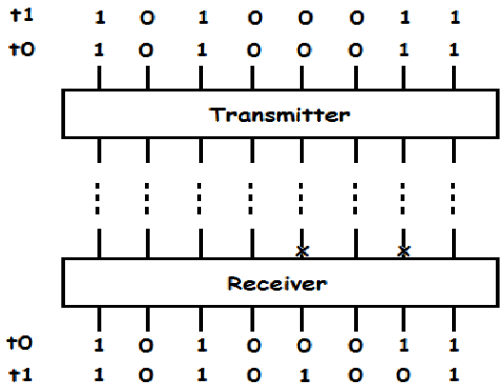
Εικόνα 6.3: Μέθοδος επιδιόρθωσης μονίμων σφαλμάτων με χρονικό πλεονασμό

Ειδικότερα, έχουν αναπτυχθεί πρακτικές διαδικασίες πρόληψης μόνιμων σφαλμάτων που θα αναλυθούν παρακάτω και είναι οι εξής: η Αντίστροφη λογική (Alternating Logic) και ο επανυπολογισμός με μεταβολές των συντελεστών (REMO, RESO, RESWO).

6.1 Alternating Logic

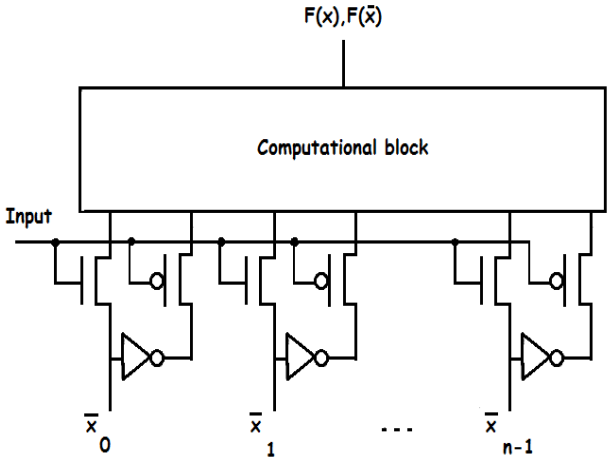
Η αντίστροφη λογική αποτελεί μια έξυπνη τεχνική πλεονασμού που μπορεί να χρησιμοποιηθεί με επιτυχία για την αναγνώριση μονίμων σφαλμάτων κατά την μετάδοση δεδομένων και στα λογικά κυκλώματα. Χρησιμοποιεί αντίθετη λογική κατά την επανάληψη του υπολογισμού, όπως πύλες NOT και αριθμητικά συμπληρώματα. Στηρίζεται στην διπλή φύση των λογικών κυκλωμάτων: Συμπληρώματα και αντιστροφές για την παραγωγή της αρχικής λογικής συνάρτησης.

Κατά την μετάδοση δεδομένων συμβαίνουν σφάλματα που μεταβάλλουν τις τιμές των ψηφίων γνωστές ως stuck-at-faults. Έστω σε ένα δίαυλο επικοινωνίας, ο πομπός επιθυμεί να αποστείλει μια πληροφορία στον δέκτη την χρονική στιγμή t0. Ο δέκτης λαμβάνει τα δεδομένα με επιτυχία . Την επόμενη χρονική στιγμή t1 ο πομπός συμπληρώνει τα δεδομένα και τα μεταδίδει στον δίαυλο. Με την σειρά του ο δέκτης την λαμβάνει και την συμπληρώνει εκ νέου. Τέλος οι δύο πληροφορίες συγκρίνονται ψηφίο ανά ψηφίο για πιθανά σφάλματα. Επομένως στην περίπτωση που υπάρχει σφάλμα, εντοπίζεται και με τον απλό αυτό τρόπο αντιμετωπίζονται και πολλαπλά κολλήματα ψηφίων. Παρακάτω παρουσιάζεται ο τρόπος μετάδοσης της πληροφορίας 10100011 την στιγμή t0. Την στιγμή t1 επαναλαμβάνεται η μετάδοση και ο πομπός συμπληρώνει την πληροφορία. Με την παραλαβή της συγκρίνονται αν η μία είναι συμπλήρωμα της άλλης. Παρατηρούμε ότι έχουν συμβεί δυο σφάλματα μετάδοσης στο bit-4 και bit-6 αντίστοιχα.



Εικόνα 6.4: Μετάδοση ενός byte με stuck-at-1 και stuck-at-0, στο bit-4 και bit-6.

Για την επιδιόρθωση πρέπει να εισαχθούν οι κατάλληλοι κωδικοποιητές και αποκωδικοποιητές στον πομπό και δέκτη αντίστοιχα. Το πιο απλό παράδειγμα είναι η εισαγωγή ενός κώδικα ισοτιμίας, για τον έλεγχο της πληροφορίας. Ένα παράδειγμα σε υλικό αποτελεί το παρακάτω που για είσοδο 0 παράγει την F(x) ενώ για 1 την αντίστροφή της[3].



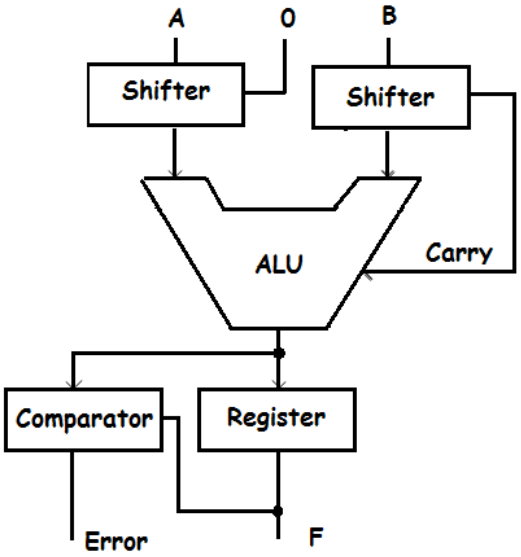
Εικόνα 6.5: Κύκλωμα μεταβαλλόμενης λογικής για την παραγωγή συμπληρωματικής συνάρτησης

6.2 Επανυπολογισμός με Μεταβολές των Συντελεστών - Recompute with Modified Operands

Σύμφωνα με την μέθοδο αυτή οι συντελεστές χωρίζονται σε δύο η περισσότερα τμήματα ώστε να υπάρχει η δυνατότητα απλοποίησης μονίμων σφαλμάτων σε με κάθε επαναυπολογισμό. Υπάρχουν δυο βασικές περιπτώσεις [1] της τεχνικής αυτής. Η πρώτη είναι γνωστή ως RESO και χρησιμοποιεί πρακτικές ολίσθησης για την επικύρωση των αποτελεσμάτων. Η δεύτερη είναι όμοια με την πρώτη μόνο που χρησιμοποιεί τεχνικές εναλλαγής μεταξύ των τμημάτων των συντελεστών.

6.2.1 Recompute with Shifted Operands

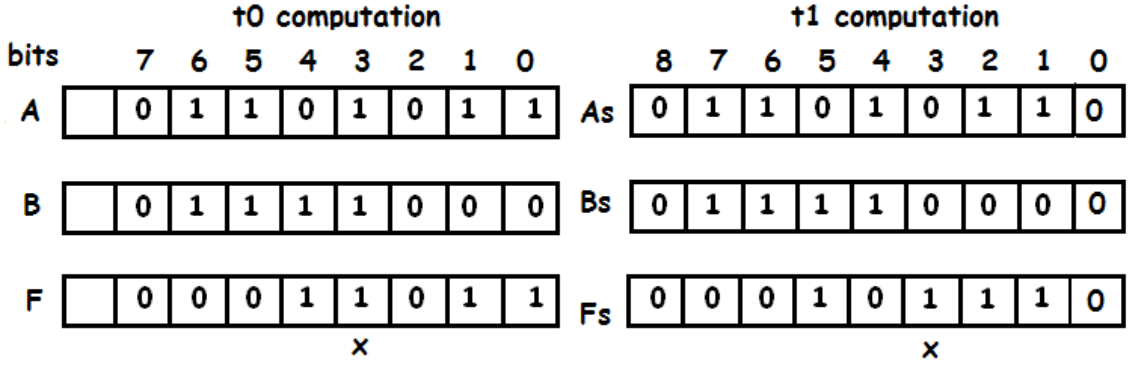
Σύμφωνα με την μέθοδο αυτή την χρονική στιγμή t0 πραγματοποιείται ο υπολογισμός με τους αρχικούς συντελεστές. Σε επόμενο χρόνο t1 οι συντελεστές ολισθαίνουν αριστερά, πραγματοποιείται ο υπολογισμός με αυτούς και το αποτέλεσμα ολισθαίνεται δεξιά. Τα αποτελέσματα τω δύο υπολογισμών του αρχικού και του ολισθημένου πρέπει να συμφωνούν. Σε αντίθετη περίπτωση έχει συμβεί ένα σφάλμα. Η ολίσθηση πρέπει να είναι ένα ψηφίο για λογικές πράξεις ή δύο για αριθμητικές πράξεις. Λόγω της απλότητας του οποιοδήποτε κύκλωμα μπορεί να χρησιμοποιήσει αυτή την μέθοδο με απλή ανάλυση των συστατικών του. Καλύπτονται οι βασικές αριθμητικές πράξεις καθώς και πράξεις προσημασμένων και δεκαδικών αριθμών[1].



Εικόνα 6.6: Η μεθοδολογία RESO με χρήση ALU για αριθμητικές και λογικές πράξεις.

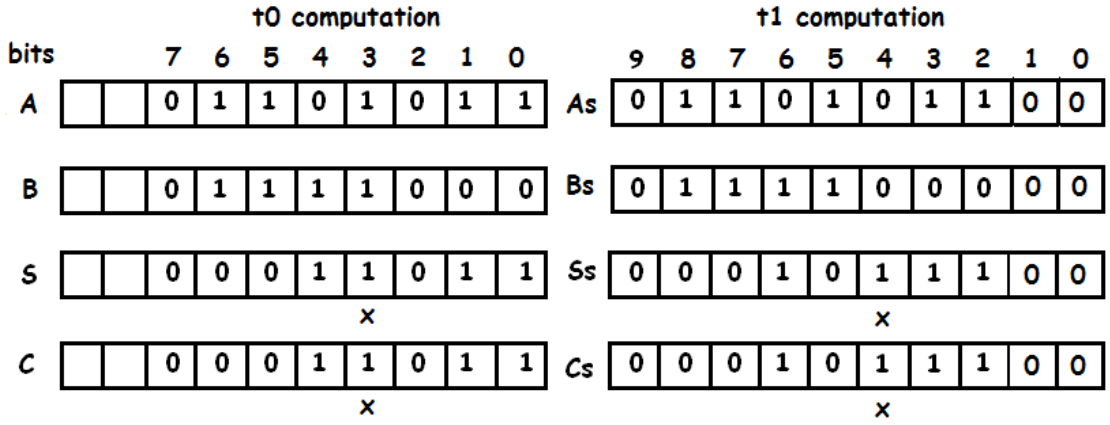
Η RESO όπως είναι γνωστή επιτυγχάνει σύγχρονο εντοπισμό σφαλμάτων σε αριθμητικές λογικές μονάδες. Στην περίπτωση αυτή οι ολισθητές χρησιμοποιούνται σαν μέθοδο κωδικοποίησης των συντελεστών πριν εισαχθούν στην ALU. Ένας αριστερός ολισθητής μπορεί να είναι ο κωδικοποιητής και ένας δεξιός ο αποκωδικοποιητής του. Το παραπάνω σχήμα αποτελεί μια πρακτική υλοποίηση της τεχνικής αυτής. Η λειτουργικότητα της ALU εμπεριέχει την οργάνωση των συντελεστών σε bit slices.

Έστω δύο συντελεστές A και B, μήκους 8-bit και F η έξοδος ενός τέτοιου κυκλώματος. Υποθέτουμε ότι έχει προκύψει ένα μόνιμο σφάλμα σε κάποιο ψηφίο των συντελεστών και η πράξη που πρέπει να υλοποιηθεί είναι λογική XOR. Έστω συντελεστής A=01101011 και B=01111000, επομένως το αποτέλεσμα την χρονική στιγμή t0 είναι F=00010011. Αν όμως έχει συμβεί ένα μόνιμο stuck-at fault στο bit-3, το αποτέλεσμα είναι F=00011011. Κατά τον υπολογισμό την επόμενη στιγμή t1, ολισθαίνουν οι συντελεστές αριστερά και το μήκος γίνεται εννέα ψηφία. Το αποτέλεσμα παρατηρούμε ότι επηρεάζει την δεύτερη θέση της ολισθημένης πληροφορίας, πράγμα που υποδηλώνει ότι έχει συμβεί μόνιμο σφάλμα στη τρίτη θέση του αρχικού αποτελέσματος F. Με τον τρόπο αυτό εντοπίζονται μόνιμα σφάλματα σε λογικές πράξεις.



Εικόνα 6.7: RESO για την λογική πράξη XOR του παραδείγματος.

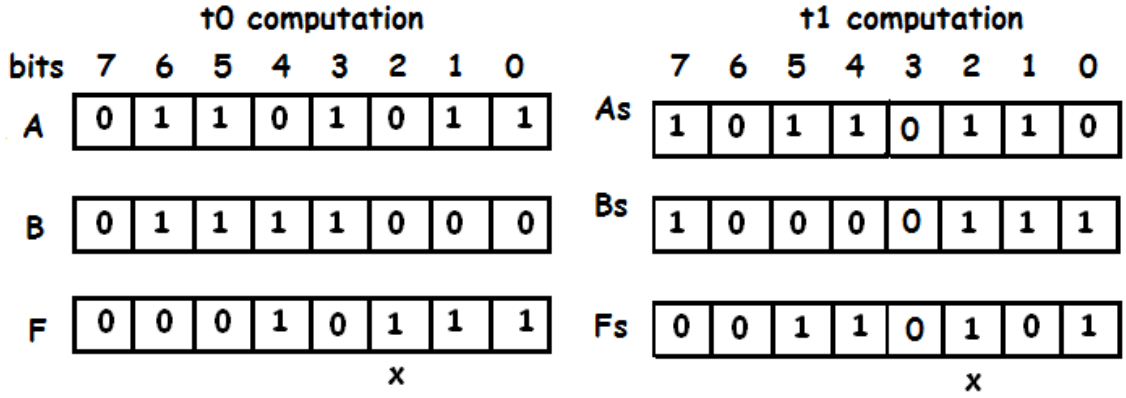
Για τις αριθμητικές πράξεις ο τρόπος είναι παρόμοιος[3]. Για παράδειγμα σε έναν αθροιστή κρατούμενου χρειάζονται δύο επιπλέον ψηφία για να πραγματοποιηθεί ο εντοπισμός μονίμων σφαλμάτων σε ένα ψηφίο της πληροφορίας.



Εικόνα 6.8: RESO για αριθμητικές πράξεις.

6.2.2 Recompute with Swapped Operands

Η τεχνική αυτή χρησιμοποιείται για τον εντοπισμό σφαλμάτων σε κάθε bit πληροφορίας τεμαχίζοντας την πληροφορία σε δύο τμήματα. Αυτά εναλλάσσονται κατά τον επόμενο υπολογισμό και συγκρίνονται τα αποτελέσματα, με τον αρχικό υπολογισμό. Χρησιμοποιείται για την αντιμετώπιση μονίμων σφαλμάτων τόσο στις λογικές όσο και στις αριθμητικές πράξεις. Για παράδειγμα η λογική πράξη XOR, με τους τελεστές A=01101011 και B=01111000. Έστω η περίπτωση που έχουμε μόνιμο σφάλμα στο bit 2. Η διαδικασία εντοπισμού είναι η έξης, πραγματοποιείται ο υπολογισμός την χρονική στιγμή t0 με τους δύο τελεστές A και B κανονικά τοποθετημένους. Την επόμενη χρονική στιγμή, τα τέσσερα πρώτα ψηφία αντιμεταθέτονται με τα τέσσερα τελευταία του ίδιου τελεστή και εκτελείται εκ νέου ο υπολογισμός. Έτσι λοιπόν παρατηρείται εκ νέου το σφάλμα στην θέση δύο.



Εικόνα 6.9: RESWO για την λογική πράξη XOR.

Η μέθοδος αυτή προϋποθέτει και την χρήση επιπλέον συσκευών όπως οι πολυπλέκτες για την εναλλαγή των ψηφίων και ένα αποθηκευτικό μέσο για την

αποθήκευση του προτού υπολογισμού και έναν συγκριτή για την εξέταση των αποτελεσμάτων.

6.2.3 Recompute by Duplicating and Comparing Operands

Σε αυτή την μέθοδο οι τελεστές χωρίζονται σε δύο τμήματα και πραγματοποιούνται οι πράξεις ξεχωριστά. Πρώτα εκτελούνται οι πράξεις μεταξύ των λιγότερο σημαντικών τμημάτων, αφού πρώτα διπλασιαστούν και αποθηκεύεται το αποτέλεσμα τους. Έπειτα πραγματοποιείται και η πράξη μεταξύ των υπόλοιπων τμημάτων και στο τέλος ενσωματώνονται για την εξαγωγή του αποτελέσματος. Ο τρόπος αυτός εκτέλεσης πράξεων είναι χρήσιμος τόσο για αριθμητικές αλλά και λογικές πράξεις[3]. Οι συσκευές που πραγματοποιούν την επιλογή των δύο τμημάτων συνήθως είναι οι πολυπλέκτες.

6.3 Συγχρονισμός

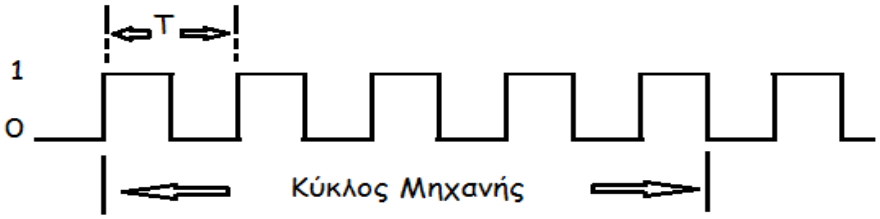
Σε όλα τα υπολογιστικά συστήματα υπάρχει η έννοια του ρολογιού συστήματος. Το ρολόι αυτό είναι απαραίτητο για την παραγωγή συγκεκριμένου αριθμού βημάτων με την μορφή παλμών, που χρησιμοποιούνται για την σύγκριση γεγονότων και τον συγχρονισμό των εξαρτημάτων. Είναι γνωστό και με την ονομασία *λογικό ρολόι* καθώς δεν έχει την καθιερωμένη χρήση ενός κλασσικού ρολογιού, που χρησιμοποιείται για την αναφορά της ώρας της ημέρας. Χρησιμοποιείται για τον καθορισμό των βημάτων εκτέλεσης εντολών ανά μονάδα χρόνου στους μικροεπεξεργαστές και τον χρονοπρογραμματισμό εργασιών στα λειτουργικά και κατανεμημένα συστήματα.

Το βασικό πρόβλημα είναι ότι δεν υπάρχει συγχρονισμός στην ιδανική του μορφή. Στην καθημερινότητά μας χρησιμοποιούμε τις ώρες της ημέρας για την ομαλή λειτουργία της κοινωνίας και της προσωπικής μας ζωής. Η πρώτη μας σκέψη λοιπόν θα ήταν να ενσωματώσουμε αυτή την λειτουργικότητα σε ένα σύστημα. Για τα συστήματα υπολογιστών δεν είναι τόσο εύκολο το έργο. Έστω η περίπτωση που έχουμε δύο υπολογιστές, Α και Β. Ο Α επιθυμεί να στείλει ένα μήνυμα στον Β μια συγκεκριμένη χρονική στιγμή. Το πρώτο πρόβλημα είναι ότι δεν υπάρχει κάποια διαβεβαίωση ότι τα ρολόγια δύο διαφορετικών υπολογιστών είναι συγχρονισμένα, δηλαδή δείχνουν την ίδια ώρα. Έστω ότι ο Α μεταδίδει ένα μήνυμα στις 12:00:00 και ο Β μεταδίδει ένα μήνυμα στις 12:00:20, είναι πολύ πιθανό ότι το μήνυμα του Β να έχει παραχθεί πριν από αυτό του Α, αν το ρολόι του υπολογιστή Β είναι 20 δευτερόλεπτα γρηγορότερο από αυτό του Α. Προβλήματα αυτού του είδους μελέτησε ο Lamport και έδωσε τις πρώτες λύσεις [8].

Το δεύτερο πρόβλημα που προκύπτει είναι ότι ακόμα και στην περίπτωση που συγχρονίζαμε τους υπολογιστές Α και Β ανά τακτά χρονικά διαστήματα, δεν υπάρχει διαβεβαίωση ότι τα εσωτερικά ρολόγια των υπολογιστών τρέχουν σε ίδιες συχνότητες. Η διαφορά αυτή προκαλεί την ολίσθηση του χρόνου, έστω και μερικών χιλιοστών του δευτερολέπτου ανά έτος. Το πρόβλημα που προκύπτει είναι η αναφορά διαφορετικών χρόνων στα μηνύματα, γνωστό και ως timestamp. Κάποιος θα μπορούσε να αναρωτηθεί τι γίνεται στην περίπτωση που οι υπολογιστές είναι του ίδιου κατασκευαστή ή χρησιμοποιούν κοινό ρολόι αναφοράς. Το αποτέλεσμα θα είναι πάλι το ίδιο καθώς πάντα θα υπάρχουν έστω και χιλιοστές του χιλιοστού διαφορές

στα ρολόγια από κατασκευαστικής άποψης και διαφορές από την σκοπιά της μετάδοσης της πληροφορίας, όπως το μήκος διαύλου και η ταχύτητα των ρολογιών μεταξύ συστημάτων. Ενδεικτικά γίνεται γνωστή αυτή η διάφορα στο [5] Ο συγχρονισμός αποτελεί βασικό πρόβλημα για τα υπολογιστικά συστήματα από τις αρχές δημιουργίας τους, όμως από τότε έχουν αναπτυχθεί αρκετές μέθοδοι που προσφέρουν λύσεις σε αυτό. Μερικά ενδεικτικά αποτελούν τα έξης: το πρόβλημα παραγωγού- καταναλωτή, το δείπνο των πέντε φιλοσόφων και αυτό των αναγνωστών-συγγραφέων.

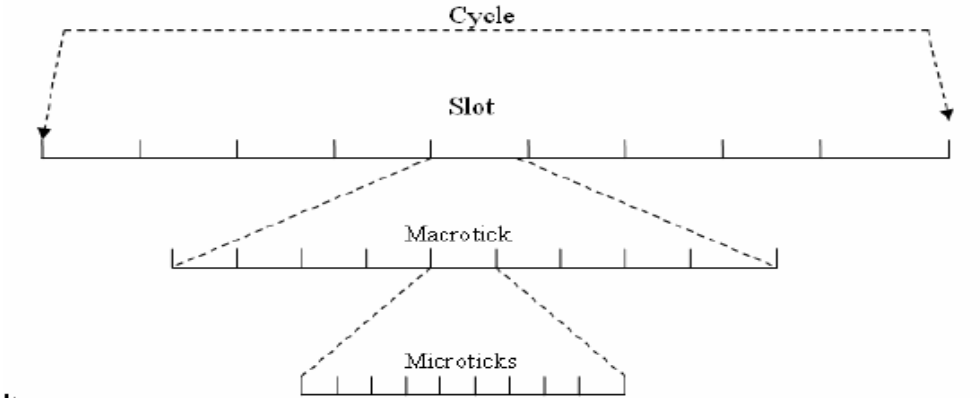
Οι μικροεπεξεργαστές είναι πολύ καλά παραδείγματα περίπλοκων συγχρονισμένων εξαρτημάτων. Η λειτουργία τους βασίζεται σε ένα εσωτερικό ρολόι το οποίο είναι υπεύθυνο για τον χρονοπρογραμματισμό των εντολών. Κάθε εντολή έχει και συγκεκριμένο χρόνο εκτέλεσης, που είναι υποδιαίρεση του ρολογιού συστήματος. Ο χρόνος μετριέται σε περιόδους T (ns) και η ταχύτητα εκτέλεσης σε f (GHz). Είναι απαραίτητος καθώς όλες οι λειτουργίες να είναι συγχρονισμένες, όπως η ανάγνωση, η αποκωδικοποίηση, η εκτέλεση και η εγγραφή στην μνήμη. Όλες οι παραπάνω λειτουργίες είναι ένας κύκλος μηχανής. Ο συγχρονισμός επιτυγχάνεται με δύο τρόπους, είτε στην εναλλαγή 0 σε 1 γνωστό ως rising edge, είτε στην εναλλαγή 1 σε 0 γνωστό ως falling edge. Οι κατασκευαστές μικροεπεξεργαστών συμφωνούν εκ των προτέρων και το αποσαφηνίζουν στα manual τον τρόπο και την μέθοδο που χρησιμοποιούν.



Εικόνα 6.10: Ο χρονοπρογραμματισμός σε μια CPU.

Στα κατανεμημένα συστήματα [4] ο συγχρονισμός ομοίως βασίζεται στην ακρίβεια των εσωτερικών χρονικών ταλαντωτών και η αξιοπιστία τους ορίζεται από τους κατασκευαστές των συστημάτων. Επιπλέον παράμετρος αποτελεί η μακροχρόνια ευστάθεια των ταλαντωτών από μαζικές εξωτερικές διαταραχές. Επομένως σε ένα κατανεμημένο δίκτυο ο συγχρονισμός μπορεί να εφαρμοστεί τοπικά και καθολικά. Τοπικά για κάθε υπολογιστικό σύστημα και καθολικά για τις λειτουργίες του δικτύου. Παραδείγματα αποτελούν τα FlexRay, Time Triggered Protocol TTP, Time triggered Controller Area Network-TTCAN, DACAPO.

Όπως γίνεται προφανές από το σχήμα 6.11 κάθε κύκλος μηχανής διαιρείται σε τμήματα και αυτά τα τμήματα σε υποτμήματα. Η μικρότερη μονάδα ονομάζεται Microticks (μTs). Η επόμενη μονάδα ονομάζεται Macroticks. Η σχέση μεταξύ των δύο είναι 9/1. Στην επόμενη βαθμίδα ανάλυσης ενός παλμού του ταλαντωτή βρίσκονται τα slots με την ίδια αναλογία. Τέλος ένας κύκλος μηχανής στο FlexRay αναλύεται σε 9 slots. Όμοια ανάλυση του χρόνου γίνεται για την κατανόηση της μέτρησης του χρόνου στα υπόλοιπα συστήματα. Η διαίρεση αυτή του χρόνου μειώνει σε ικανοποιητικά επίπεδα την πιθανότητα σφάλματος λόγω συγχρονισμού και αυτόματα δημιουργούνται και οι τρόποι αντιμετώπισης.



Εικόνα 6.11: Η δομή του χρονισμού στο FlexRay.

Ο Christian [6] ανέπτυξε τον ομώνυμο αλγόριθμο που απαιτεί έναν κεντρικό διακομιστή ως λογικό ρολόι και αποτελεί την βάση ή σύμβολο αναφοράς για όλο το δίκτυο. Ο χρόνος μεταξύ της αίτησης, της επεξεργασίας και αποστολής ενός μηνύματος από αυτόν ορίζεται ως *Round Trip Time-RTT. Η διαδικασία γίνεται με τον παρακάτω τρόπο:*

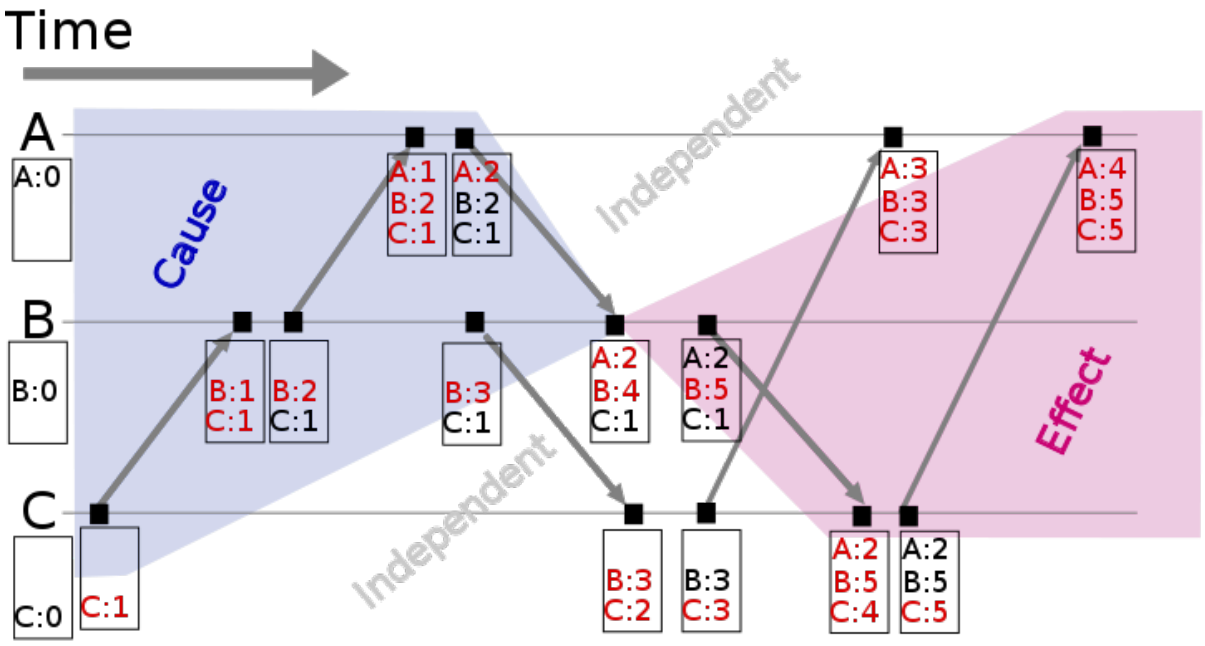
- Μία διεργασία P ζητά να εξυπηρετηθεί από έναν διακομιστή S.
- S λαμβάνοντας το αίτημα, αποστέλλει τον δικό του χρόνο T πίσω στην P.
- Η P αφού λάβει την απάντηση εκτελεί την πράξη T+RTT/2 για τον συγχρονισμό.

Η μέθοδος αυτή ουσιαστικά εκφράζει την πιθανότητα εξυπηρέτησης ενός αιτήματος ως τον μέσο όρο του χρόνου, πράγμα που δεν είναι απόλυτα ακριβές αλλά ικανοποιητικό σε ένα δίκτυο LAN. Περισσότερη ακρίβεια μπορεί να επιτευχθεί με την αποστολή περισσότερων αιτημάτων και την εκτέλεση της πράξης για το συντομότερο RTT. Πρακτικά η διαφορά στην ακρίβεια γίνεται αντιληπτή με την θεώρηση: Έστω τα ο χρόνος μέχρι το αίτημα της P φτάσει στον S. Βρίσκεται ανάμεσα στο (RTT - τ) και (T+ RTT - τ). Το εύρος αυτού του διαστήματος είναι (RTT-2*τ). Οπότε η ακρίβεια είναι (RTT/2 - τ).

Μια ακόμη μέθοδος είναι ο αλγόριθμος του Berkeley [7]. Σύμφωνα με αυτόν επιλέγεται ένας υπολογιστής ως master και οι υπόλοιποι θεωρούνται ως slave. Γίνονται εκλογές κάθε φορά και επιλέγεται διαφορετικός υπολογιστής ως master. Αυτός έχει την ευθύνη να συγκεντρώσει τις τιμές του χρόνου των slave, σε RTT και να πραγματοποιεί σύγκριση των αποτελεσμάτων με το δικό του ρολόι. Η μέθοδος που χρησιμοποιείται είναι ίδια με αυτή του Christian [6] για τις αιτήσεις. Έπειτα ο master υπολογίζει τις μέσες τιμές των χρόνων που έλαβε, μόνο για τους πλησιέστερους χρόνους των υπολοίπων. Μεγάλες αποκλίσεις δεν λαμβάνονται υπόψην. Τέλος αντί ο υπολογιστής αφέντης να στείλει τον ενημερωμένο χρόνο στους υπόλοιπους, στέλνει την απόκλιση που πρέπει κάθε υπολογιστής να διορθώσει στο εσωτερικό του ρολόι, για να επιτευχθεί ο συγχρονισμός. Με τον τρόπο αυτό καταργούνται οι χρονικές αποκλίσεις μεταξύ υπολογιστών.

Επέκταση της αρχικής ιδέας του Lamport [8] αποτελεί ο αλγόριθμος των vector clocks. Η λειτουργία του είναι η εξής:

- Αρχικά όλα τα ρολόγια σε ένα δίκτυο είναι μηδενικά.
- Όταν ένας υπολογιστής υποστεί μεταβολή της κατάστασης του ρολογιού του αυξάνει το διάνυσμα του κατά ένα.
- Όταν ένας υπολογιστής θέλει να στείλει σε άλλον, στέλνει και το διάνυσμα του μαζί με το μήνυμα.
- Όταν ένας υπολογιστής θέλει να λάβει ένα μήνυμα, αυξάνει το λογικό του ρολόι και ενημερώνει τις υπόλοιπες τιμές του διανύσματος, με την μέγιστη τιμή του δικού του διανύσματος και των υπολοίπων ξεχωριστά.



Εικόνα 6.12: Λειτουργία των Vector clocks

Η μέθοδος αυτή είναι αιτιοκρατική, δηλαδή εκτός από τον συγχρονισμό των ρολογιών προσφέρει και την μερική οργάνωση του τρόπου που θα πραγματοποιηθεί ,όπως παρουσιάζεται παραπάνω. Με μπλε είναι η ροή του μηνύματος και με κόκκινο το αποτέλεσμα της ενημέρωσης του διανύσματος Β με την τιμή 4.

Επομένως όπως γίνεται αντιληπτό οι παραπάνω μέθοδοι εξυπηρετούν τον συγχρονισμό για την αποστολή και λήψη δεδομένων πάνω σε κανάλια επικοινωνίας, είτε εσωτερικά μεταξύ εξαρτημάτων είτε σε μεγαλύτερη κλίμακα όπως σε ένα δίκτυο υπολογιστών. Όμως με το πέρασμα στα πολυπύρηνα συστήματα δημιουργήθηκαν νέα προβλήματα συγχρονισμού μεταξύ των πυρήνων του μικροεπεξεργαστή, όπως ο παράλληλος χρονοπρογραμματισμός εντολών και διεργασιών. Μια καλή αναφορά γίνεται από τον Grammoli[9], που δοκιμάζει τις μεθόδους συγχρονισμού με την χρήση λογισμικού, που έχουν αναπτυχθεί ως τις μέρες μας.

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

9.1 Σύνοψη της πτυχιακής εργασίας

Η εργασία αυτή ξεκίνησε από προσωπική μου ενδιαφέρον σχετικά με τους τρόπους που μπορούμε να δημιουργήσουμε ανθεκτικότερα υπολογιστικά συστήματα. Η διερεύνηση αυτή στο πεδίο της Ανοχής σφαλμάτων μου έμαθε αρκετούς από τους τρόπους που μπορούμε να το επιτύχουμε αυτό.
Αναλυθήκαν οι έννοιες σφάλμα λάθος και αποτυχία καθώς και η σχέση τους σε ένα σύστημα. Η έννοια της αξιοπιστίας σε ένα σύστημα είναι τόσο σημαντική όσο και η επίδοση του, σε βαθμό που πλέον είναι αλληλένδετες και μη διαχωρίσιμες για τα υπολογιστικά συστήματα. Εκτός αυτού ο πυλώνας όλων των συστημάτων είναι τα RAMS(Reliability, Availability, Maintainability Safety)

Στο επίπεδο της πληροφορίας οι κώδικες επιδιόρθωσης και εντοπισμού είναι τα χρήσιμα εργαλεία. Αναφέρονται στην ενσωμάτωση πρόσθετων πληροφοριών στα δεδομένα με σκοπό τον εντοπισμό και την επιδιόρθωση σφαλμάτων. Η μορφή αυτή πλεονασμού υλοποιείται με την χρήση κωδικών εντοπισμού σφαλμάτων (EDC) και κωδικών επιδιόρθωσης σφαλμάτων (ECC) που έχουν μεγάλη εφαρμογή στα λογικά κυκλώματα. Χαρακτηρίζονται από την διαχωρισιμότητα τους και κατηγοριοποιούνται με βάση την εφαρμογή. Ευρέως εφαρμοσμένες τεχνικές αποτελούν οι κωδικες Hamming, Reed Solomon, CRC, Convolution κώδικες. Οι τεχνικές ARQ χρησιμοποιούνται στις επικοινωνίες δεδομένων για την επιτυχή αποστολή και λήψη πακέτων.

Ο πλεονασμός υλικού ενδείκνυται στις περιπτώσεις που οι υπόλοιπες τεχνικές δεν είναι σε θέση να βελτιώσουν την αξιοπιστία του συστήματος. Επιτυγχάνεται με την προσθήκη δύο ή περισσότερων φυσικών αντιγράφων ενός εξαρτήματος υλικού. Πολλές φορές πραγματοποιείται σύγκριση των αποτελεσμάτων και ψηφοφορία των αποτελεσμάτων ώστε να αποκρύβονται σφάλματα από το σύστημα. Με τον τρόπο αυτό έχουμε αύξηση της διαθεσιμότητας των υπολογιστικών συστημάτων. Οι τεχνικές που αξιοποιούνται εδώ είναι ο Ενεργός και ο Παθητικός πλεονασμός και οι μέθοδοι NMR και PNS.

Ο πλεονασμός στο χώρο του λογισμικού εφαρμόζεται με μεγάλη επιτυχία εδώ και δεκαετίες . Οι βασικές ιδέες αν και απλές είναι χρήσιμες στην αντιμετώπιση σφαλμάτων λογισμικού. Οι πιο σπουδαίες είναι οι τεχνικές του NVP, Τα Recovery Blocks, Self checking programming και Checkpoint and Recovery.

Ο χρονικός πλεονασμός αφορά τις τεχνικές διαχείρισης του χρόνου σε ένα υπολογιστικό σύστημα. Τα σφάλματα που μπορεί να αντιμετωπίσει είναι δύο ειδών τα περιστασιακά και τα μόνιμα σφάλματα. Τα περιστασιακά σφάλματα μπορούν να συμβούν μια τυχαία χρονική στιγμή αλλά με την επανάληψη του υπολογισμού να εξαλείφουν από το σύστημα. Σε αντίθεση με αυτά τα μόνιμα σφάλματα επιφέρουν την αποτυχία του συστήματος. Οι μέθοδοι στο πεδίο αυτό είναι οι Αντίστροφη Λογική, οι REMO,RESWO και RDCO.

Συνοψίζοντας,Η χρήση πλεονασμού και των μεθόδων ανοχής σφαλμάτων σε ένα σύστημα δεν σημαίνει ότι θα έχουμε και άμεση βελτίωση της αξιοπιστίας του. Ο πλεονασμός αυξάνει την πολυπλοκότητα του συστήματος σε αισθητά επίπεδα,

επομένως πρέπει να χρησιμοποιείται με αυστηρότητα και σε εξαρτήματα τα οποία το έχουν ανάγκη. Ειδάλλως δημιουργούμε περισσότερα προβλήματα παρά λύνουμε.

Γνωρίζοντας τις παραπάνω μεθοδολογίες κάποιος έχει την δυνατότητα να εφαρμόσει όπου κρίνει τις τεχνικές για την δημιουργία αξιόπιστων συστημάτων με μικρότερο κόστος από μη αξιόπιστα υλικά και πολλές φορές είναι μικρότερου κόστους από ότι ακριβότερα εξαρτήματα που θα χρησιμοποιηθούν στην ίδια εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΊΑ

ΚΕΦΆΛΑΙΟ 1

[1] John Von Neumann, "Probabilistic logic and synthesis of reliable organisms from unreliable components", 1956

[2]David A. Rennels, "Fault tolerant Computing: Issues, Examples and methdodology", 1987.

[3]http://courses.cs.vt.edu/professionalism/Therac_25/Therac_1.html

[4]http://atomicinsights.com/accident-at-chernobyl-caused-explosion/

[5]http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html

[6]http://edition.cnn.com/TECH/space/9909/30/mars.metric.02/

[7]http://www.spiegel.de/international/airbus-confirms-delay-a380-hit-by-new-production-problems-a-438408.html

[8]http://www.businessweek.com/stories/2008-05-13/airbus-confirms-further-a380-delaybusinessweek-business-news-stock-market-and-financial-advice

[9]http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/ak16/

[10]http://voyager.jpl.nasa.gov/

[11] http://www.zdnet.com/the-top-10-it-disasters-of-all-time-3039290976/

[12] http://www.cs.ucla.edu/~rennels/article98.pdf

[13]http://www.world-nuclear.org/information-library/safety-and-security/safety-of-plants/fukushima-accident.aspx

[14] http://www.isro.gov.in/pslv-c25-mars-orbiter-mission

[15]http://www.theguardian.com/business/2012/oct/14/black-monday-sowed-seeds-financial-crisis

[16] A. Avizienis, H Kopetz, C. Laprie, "The Evolution of Fault-Tolerant Computing",1991

ΚΕΦΑΛΑΙΟ 2

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell,"Fundamental Concepts of Dependability», 2001

[2] Elena Dubrova, "Fault Tolerant Design", 2013.

[3] Mostafa Abd el Barr, 'Design And Analysis Of Reliable And Fault-Tolerant Computer Systems', 2007

[4] Alessandro Birolini, "Reliability Engineering Theory and Practice", Springer ,2013

[4] Lahoz, C. H. N., Romani, M. A. S. , Yano, E. T, "Dependability Attributes for Space Computer Systems: Quality Factors Approach"

[5]http://www.eventhelix.com/realtimemantra/faulthandling/reliability_availability_basics.htm#Hardware Failures

[6] MIL-STD-721RevC

ΚΕΦΑΛΑΙΟ 3

[1] Eiji Fujiwara, "Code Design for Dependable Systems: Teory and Practical Applications", (2006)

[2] Elena Dubrova, "Fault Tolerant Design", Springer, 2013

[3] Parag K. Lala, "Self-checking and Fault-tolerant Digital Design" ,Academic Press, 2001

[4] http://www.slideshare.net/sandeep101026/crc-java-code.

[5] http://www.repairfaq.org/filipg/LINK/F_crc_v32.html#CRCV_002

[6] http://nutaq.com/en/blog/hybrid-arq-part-2

[7] Mostafa Abd el Barr, 'Design And Analysis Of Reliable And Fault-Tolerant Computer Systems', Imperial College Press, 2007

ΚΕΦΑΛΑΙΟ 4

[1] Mostafa Abd el Barr, "Design And Analysis Of Reliable And Fault-Tolerant Computer Systems", Imperial College Press 2007

[2] Elena Dubrova, "Fault Tolerant Design", 2013

[3] Moore, E., Shannon, C., “Reliable circuits using less reliable relays”. J. Frankl. Inst. 262(3), 191–208 , 1956

[4] Smith, D.J.: “Reliability Engineering”, Barnes & Noble Books, New York, 1972

[5] Dickinson, M., Jackson, J., Randa, G., “Saturn V launch vehicle digital computer and data adapter”, Proceedings of the Fall Joint Computer Conference, pp. 501–516 1964

[6] Parhami, B. ”Voting algorithms”, IEEE Trans. Reliab. 43, 617–629, 1994

[7] Yeh, Y., “Triple-triple redundant 777 primary flight computer” , Aerospace Applications Conference,Proceedings, 1996 IEEE, vol. 1, pp. 293–307, 1996

Κεφαλαιο 5

[1] McAllister, D., Vouk, M.A. “Fault-tolerant software reliability engineering” In: Lyu, M.R. (ed.)Handbook of Software Reliability. McGraw-Hill, New York, pp. 567–614 1996

[2] Elena Dubrova, “Fault Tolerant Design”, Springer, 2013

[3] Looker, N., Munro, M., Xu, J.: Increasing web service dependability through consensus voting. In: COMPSAC 2005: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC 2005), vol. 2, pp. 66–69. IEEE Computer Society, Washington (2005) & «WS-FTM: A Fault Tolerance Mechanism for Web Services» Nik Looker Malcolm Munro

[4] Dobson, G.: Using WS-BPEL to implement software fault tolerance for web services.In: EUROMICRO 2006: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 126–133. IEEE Computer Society, Washington, 2006

[5] Gashi, I., Popov, P., Stankovic, V., Strigini, L.: On designing dependable services with diverse off-the-shelf SQL servers. In: de Lemos, R., Gacek, C., Romanovsky, A. (eds.) Architecting Dependable Systems II. LNCS, vol. 3069, pp. 191–214. Springer, Heidelberg, 2004

[6] Randell, B., Xu, J.: The evolution of the recovery block concept. In: Lyu, M.R. (ed.) Software Fault Tolerance. Wiley, New York, pp. 1–21, 1995

[7] Yau, S.S., Cheung, R.C.: Design of self-checking software. In: Proceedings of the International Conference on Reliable software, pp. 450–455. ACM, New York, 1975

[8] http://users.ece.cmu.edu/~koopman/des_s99/exceptions/

[9] Maxion, Roy A.; Olszewski, Robert T., "Improving Software Robustness With Dependability Cases." Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, June 1998, p. 346-355.

[10] Popov, P., Riddle, S., Romanovsky, A., Strigini, L.: On systematic design of protectors for employing OTS items. In: Euromicro 2001: in Proceedings of the 27th Euromicro Conference, pp. 22–29, 2001

[11] Chang, H., Mariani, L., Pezz`e, M.: In-field healing of integration problems with COTS components. In: ICSE 2009: Proceeding of the 31st International Conference on Software Engineering, pp. 166–176, 2009

[12] Salles, F., Rodrguez, M., Fabre, J.C., Arlat, J.: Metakernels and fault containment wrappers. In: International Symposium on Fault-Tolerant Computing. IEEEComputer Society Press, Los Alamitos, 1999

[13] Fetzer, C., Xiao, Z.: Detecting heap smashing attacks through fault containment wrappers. In: Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems, pp. 80–89 , 2001

[14] Qin, F., Tucek, J., Zhou, Y., Sundaresan, J.: Rx: Treating bugs as allergies—a safe method to survive software failures. ACM Transactions on Computer Systems 25(3), 7, 2007

[15] Antonio Carzaniga, Alessandra Gorla, and Mauro Pezze, “Handling Software Faults with Redundancy”,

[16] Koopman, P. & DeVale, J. Comparing the Robustness of POSIX Operating Systems; IEEE Computer Society, 1999

[17] LAPRIE, J. -C., et al., “Hardware and Software Fault Tolerance: Definition and Analysis of Architectural Solutions,” Proceedings of FTCS-17, Pittsburgh, PA, 1987, pp. 116–112.

[18] http://www.on-time.com/ddj0011.htm

ΚΕΦΑΛΑΙΟ 6

[1] Janak H. Patel, Leona Y. Fung «*Concurrent Error Detection in ALU's by Recomputing with Shifted Operands» IEEE TRANSACTIONS ON COMPUTERS, VOL. C-31 NO.7, 1982*

[2] Mostafa Abd el Barr, “Design And Analysis Of Reliable And Fault-Tolerant Computer Systems”, Imperial College Press 2007

[3] Elena Dubrova, “Fault Tolerant Design”, Springer, 2013,

[4] Klaus Echtle and Soubhi Mohamed, Clock Synchronization Issues in Multi-Cluster Time-Triggered Networks,

[5] http://www.centralx.com/time/index.en.html

[6] Cristian, F. , "Probabilistic clock synchronization", *Distributed Computing* (Springer) 3 (3): 146–158, 1989

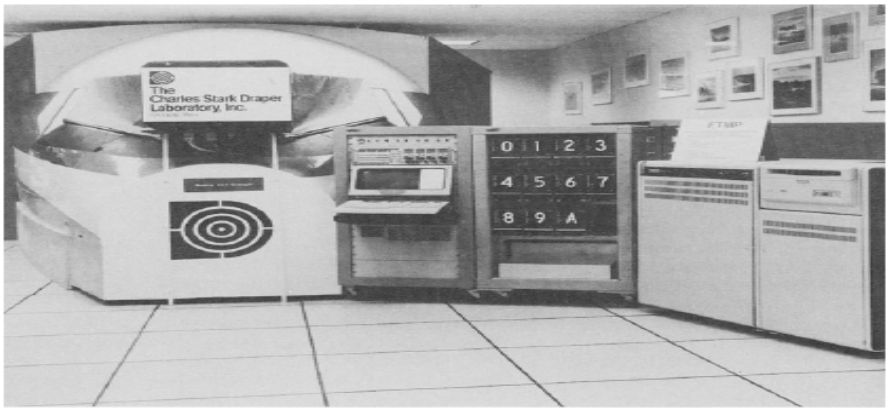
[7] Gusella, R.; Zatti, S. , "The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD", *Software Engineering, IEEE Transactions on* (IEEE) **15** (7): 847–853, 1989

[8] Lamport, L. , "Time, clocks, and the ordering of events in a distributed system" , Communications of the ACM 21 (7): 558–565, 1978

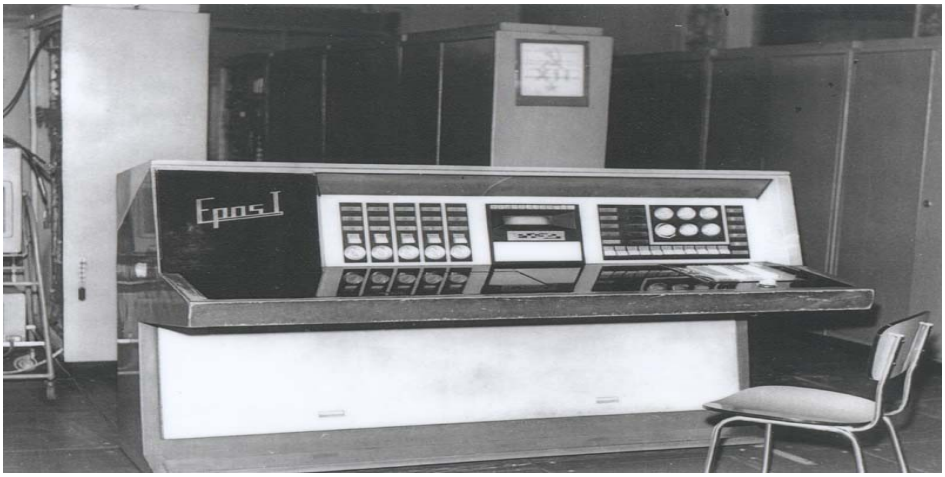
[9] More Than You Ever Wanted to Know about Synchronization,Synchrobench, Measuring the Impact of the Synchronization on Concurrent Algorithms
Vincent Gramoli

ΠΑΡΑΡΤΗΜΑ

Κεφάλαιο 1



FTMP



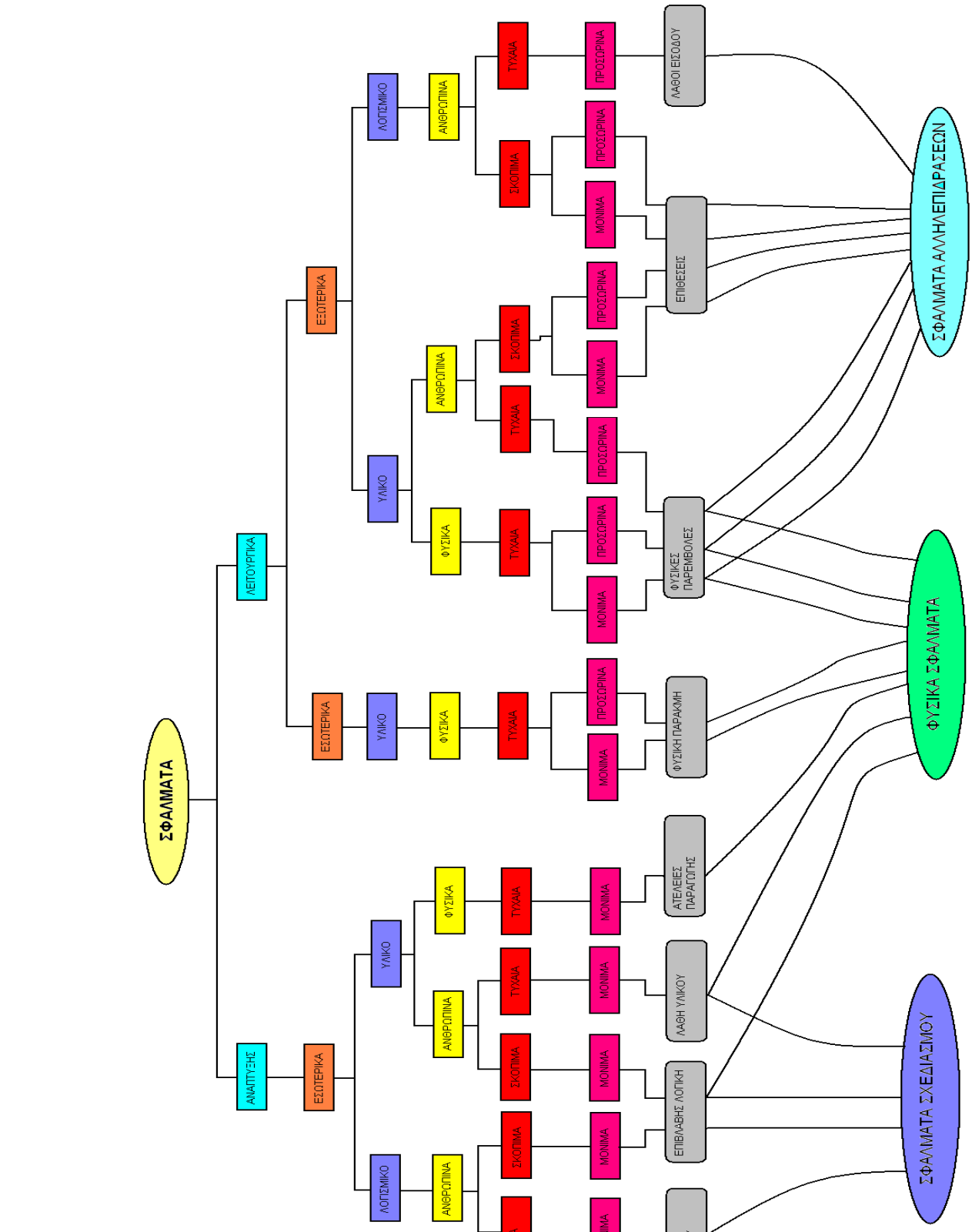
Chernobyl

Black Monday

India Mars Orbiter

Curiosity

No.	Hardware reliability	Software reliability
1	Wears out.	Does not wear out.
2	Many hardware parts fail according to the bathtub hazard rate curve.	Software does not fail according to the bathtub hazard rate curve.
3	A hardware failure is mainly due to physical effects.	A software failure is caused by programming error.
4	The failed system is repaired by performing corrective maintenance.	Corrective maintenance is really redesign.
5	Interfaces are visual.	Interfaces are conceptual.
6	The hardware reliability field is well established, particularly in the area of electronics.	The software reliability field is relatively new.
7	Obtaining good failure data is a problem.	Obtaining good failure data is a problem.
8	Usually redundancy is effective.	Redundancy may not be effective.
9	Potential for monetary savings.	Potential for monetary savings.
10	Preventive maintenance is performed to inhibit failures.	Preventive maintenance has no meaning in software.
11	It is possible to repair hardware by using spare modules.	It is impossible to repair software failures by using spare modules.
12	Hardware reliability has well-established mathematical concepts and theory.	Software reliability still lacks well-established mathematical concepts and theory.
13	Has a hazard function.	Has an error rate function.
14	Mean time to repair has significance.	Mean time to repair does not have any significance.
15	Reliability is time-related with failures occurring as a function of operational/storage time.	Reliability is not time-related and failures occur when a program step/path containing the fault is executed.


ΣΦΑΛΜΑΤΑ
ΑΝΑΠΤΥΞΗΣ
ΛΕΙΤΟΥΡΓΙΚΑ
ΕΣΩΤΕΡΙΚΑ
ΕΞΩΤΕΡΙΚΑ
ΕΣΩΤΕΡΙΚΑ
ΕΞΩΤΕΡΙΚΑ
ΥΛΙΚΟ
ΛΟΓΙΣΜΙΚΟ
ΥΛΙΚΟ
ΛΟΓΙΣΜΙΚΟ
ΥΛΙΚΟ
ΛΟΓΙΣΜΙΚΟ
ΦΥΣΙΚΑ
ΑΝΘΡΩΠΙΝΑ
ΦΥΣΙΚΑ
ΑΝΘΡΩΠΙΝΑ
ΦΥΣΙΚΑ
ΑΝΘΡΩΠΙΝΑ
ΤΥΧΑΙΑ
ΣΚΟΠΙΜΑ
ΜΟΝΙΜΑ
ΤΥΧΑΙΑ
ΣΚΟΠΙΜΑ
ΜΟΝΙΜΑ
ΤΥΧΑΙΑ
ΣΚΟΠΙΜΑ
ΜΟΝΙΜΑ
ΤΥΧΑΙΑ
ΣΚΟΠΙΜΑ
ΜΟΝΙΜΑ
ΤΥΧΑΙΑ
ΣΚΟΠΙΜΑ
ΜΟΝΙΜΑ
ΠΡΟΣΩΡΙΝΑ
ΜΟΝΙΜΑ
ΠΡΟΣΩΡΙΝΑ
ΜΟΝΙΜΑ
ΠΡΟΣΩΡΙΝΑ
ΜΟΝΙΜΑ
ΠΡΟΣΩΡΙΝΑ
ΜΟΝΙΜΑ
ΠΡΟΣΩΡΙΝΑ
ΜΟΝΙΜΑ
ΠΡΟΣΩΡΙΝΑ
ΜΟΝΙΜΑ
ΛΑΘΟΣ ΕΙΣΟΔΟΥ
ΕΠΙΘΕΣΕΙΣ
ΦΥΣΙΚΕΣ ΠΑΡΕΜΒΟΛΕΣ
ΦΥΣΙΚΗ ΠΑΡΑΜΜΗ
ΑΤΕΛΕΙΕΣ ΠΑΡΑΓΩΓΗΣ
ΛΑΘΗ ΥΛΙΚΟΥ
ΕΠΙΒΛΑΒΗΣ ΛΟΓΙΚΗ
ΑΤΕΛΕΙΕΣ ΛΟΓΙΣΜΙΚΟΥ
ΣΦΑΛΜΑΤΑ ΑΛΛΗΛΕΠΙΔΡΑΣΕΩΝ
ΦΥΣΙΚΑ ΣΦΑΛΜΑΤΑ
ΣΦΑΛΜΑΤΑ ΣΧΕΔΙΑΣΜΟΥ
ΦΑΣΗ ΔΗΜΙΟΥΡΓΙΑΣ
ΣΥΣΤΗΜΙΚΑ ΟΡΙΑ
ΠΕΔΙΟ
ΦΑΙΝΟΜΕΝΙΚΕΣ ΑΙΤΙΕΣ
ΠΡΟΘΕΣΗ
ΑΝΘΕΚΤΙΚΟΤΗΤΑ

ΣΥΝΑΡΤΗΣΗ	**ΚΑΤΑΝΟΜΗ**
Διωνυμική-Bernoulli $f(x) = \frac{n!}{k!\,(n-k)!} p^x q^{n-x}$	$F(x) = \sum_{k=0}^{x} \binom{n}{k} p^k q^{n-k}$, x=0,1…k p: πιθανότητα επιτυχίας δοκιμής q: πιθανότητα αποτυχίας δοκιμής x: αποτυχίες σε k δοκιμές p+q=1 **πάντα**
Εκθετική $f(t) = \lambda e^{-\lambda t}$, t>=0, λ>0	$F(t) = 1 - e^{-\lambda t}$
Poisson $f(m) = \frac{(\lambda t)^m e^{-\lambda t}}{m!}$, $m = 0,1,2 \ldots$	$F = \sum_{i=0}^{m} [(\lambda t)^i e^{-\lambda t}]/i\,!$
Weibull $f(t) = \left(\frac{\beta t^{\beta-1}}{\theta^\beta}\right) t e^{-\left(\frac{t}{\theta}\right)^\beta}$ t>=0, β>0, θ>0	$F(t) = 1 - e^{-\left(\frac{t}{\theta}\right)^\beta}$
Rayleigh $f(t) = \left(\frac{2}{\theta^2}\right) t e^{-\left(\frac{2}{\theta}\right)^2}$ θ: συντελεστής κατανομής	$F(t) = 1 - e^{-\left(\frac{2}{\theta}\right)^2}$
Κανονική-Gaussian $f(t) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) e^{-\frac{(t-\mu)^2}{2\sigma^2}}$	$F(t) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) \int_{-\infty}^{t} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dx$