ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ

Εργασία Εξαμήνου

ΓΙΑ ΤΟ ΜΑΘΗΜΑ

ΔΙΚΤΥΑ ΔΗΜΟΣΙΑΣ ΧΡΗΣΗΣ ΚΑΙ ΔΙΑΣΥΝΔΕΣΗ ΔΙΚΤΥΩΝ

ΣΧΕΔΙΑΣΗ ΠΡΩΤΟΚΟΛΛΩΝ

Ζαχαριά Αθανασία

Α.Μ 5295

Διδάσκων: Χρήστος Μπούρας

Πάτρα 2015

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ **4**

ΚΕΦΆΛΑΙΟ 1: ΕΙΣΑΓΩΓΉ ΣΤΑ ΠΡΩΤΌΚΟΛΛΑ **6**

1.1 ΟΡΙΣΜΌΣ ΠΡΩΤΟΚΌΛΛΟΥ ΚΑΙ ΠΡΩΤΟΚΌΛΛΟΥ ΕΠΙΚΟΙΝΩΝΊΑΣ **6**

1.2 ΠΑΡΑΔΟΧΈΣ ΓΙΑ ΤΟ ΣΧΕΔΙΑΣΜΌ ΠΡΩΤΟΚΌΛΛΩΝ **6**

ΚΕΦΆΛΑΙΟ 2: ΒΑΣΙΚΟΊ ΚΑΝΌΝΕΣ ΣΧΕΔΙΑΣΜΟΎ **8**

2.1 ΑΠΛΌΤΗΤΑ (SIMPLICITY) **8**

2.2 ΤΜΗΜΑΤΟΠΟΊΗΣΗ (MODULARITY) **8**

2.3 ΚΑΛΑ ΣΧΕΔΙΑΣΜΕΝΑ ΠΡΩΤΟΚΟΛΛΑ (WELL-FORMED PROTOCOLS) **8**

2.4 ΕΠΙΛΥΣΗ ΠΟΛΛΩΝ ΚΑΙ ΟΧΙ ΜΕΜΟΝΩΜΕΝΩΝ ΠΡΟΒΛΗΜΑΤΩΝ **9**

2.5 ΒΕΛΤΙΣΤΟΠΟΊΗΣΗ **9**

2.6 ΑΝΕΚΤΙΚΌΤΗΤΑ ΑΠΌ ΛΆΘΗ (ROBUST) **9**

2.7 ΣΥΝΈΠΕΙΑ (CONSISTENCY) **10**

ΚΕΦΆΛΑΙΟ 3: ΜΕΘΟΔΟΛΟΓΊΑ ΣΧΕΔΙΑΣΜΟΎ ΠΡΩΤΟΚΌΛΛΩΝ **11**

ΚΕΦΆΛΑΙΟ 4: ΈΛΕΓΧΟΣ ΡΟΉΣ **12**

ΚΕΦΆΛΑΙΟ 5: ΧΑΡΑΚΤΗΡΙΣΤΙΚΆ ΠΑΡΑΔΕΊΓΜΑΤΑ ΠΡΩΤΟΚΌΛΛΩΝ 14

5.1 ΠΡΩΤΌΚΟΛΛΟ ΧΩΡΊΣ ΈΛΕΓΧΟ ΡΟΉΣ 14

5.2 ΠΡΩΤΟΚΟΛΛΟ ΤΥΠΟΥ X – ON / X – OFF 15

5.3 ΠΡΩΤΌΚΟΛΛΑ ΜΕ ΠΑΡΆΘΥΡΟ 18

5.4 ΠΡΩΤΌΚΟΛΛΟ ΕΝΑΛΛΑΣΣΌΜΕΝΟΥ BIT ΜΕ TIMEOUT 23

5.5 ΠΡΟΒΛΉΜΑΤΑ ΠΡΩΤΟΚΌΛΛΩΝ ΜΕ ΠΑΡΆΘΥΡΟ 26

5.6 ΔΙΑΔΙΚΑΣΊΕΣ ΠΡΩΤΟΚΌΛΛΩΝ ΜΕ ΠΑΡΆΘΥΡΟ 27

5.7 ΠΡΩΤΌΚΟΛΛΑ ΜΕ ΑΡΝΗΤΙΚΈΣ ΕΠΙΒΕΒΑΙΏΣΕΙΣ 29

ΚΕΦΆΛΑΙΟ 6: ΔΟΜΉ ΚΑΙ ΠΡΟΔΙΑΓΡΑΦΈΣ ΠΡΩΤΟΚΌΛΛΩΝ 32

6.1 ΥΠΗΡΕΣΊΑ ΠΡΟΣ ΥΛΟΠΟΊΗΣΗ (SERVICE) 36

6.2 ΥΠΟΘΈΣΕΙΣ ΓΙΑ ΤΟ ΠΕΡΙΒΆΛΛΟΝ ΛΕΙΤΟΥΡΓΊΑΣ 36

6.3 ΛΕΞΙΛΌΓΙΟ ΜΗΝΥΜΆΤΩΝ 36

6.4 ΜΟΡΦΟΠΟΊΗΣΗ ΜΗΝΥΜΆΤΩΝ 36

6.5 ΔΙΑΔΙΚΑΣΤΙΚΟΊ ΚΑΝΌΝΕΣ 38

ΚΕΦΆΛΑΙΟ 7: ΥΠΗΡΕΣΊΕΣ ΚΑΙ ΠΕΡΙΒΆΛΛΟΝ ..... ERROR! BOOKMARK NOT DEFINED.

ΒΙΒΛΙΟΓΡΑΦΊΑ 40

Κεφαλαιο 1: Εισαγωγη στα Πρωτοκολλα

1.1 Ορισμός πρωτοκόλλου και πρωτοκόλλου επικοινωνίας

Πρωτόκολλο ονομάζεται ένα σύνολο κανόνων το οποίο καθορίζει το πώς θα εκτελεστεί μια διεργασία. Ένα πρωτόκολλο επικοινωνίας ορίζει την ακριβή μορφοποίηση των μηνυμάτων που ανταλλάσσονται (συντακτικό). Επίσης ορίζει τους διαδικαστικούς κανόνες για την ανταλλαγή των δεδομένων (Γραμματική). Τέλος ορίζει το λεξιλόγιο των μηνυμάτων που μπορούν να ανταλλάσσονται (σημασιολογικά).

1.2 Παραδοχές για το σχεδιασμό πρωτοκόλλων

Ένα πρωτόκολλο περιλαμβάνει παραδοχές και συμφωνίες για τις μεθόδους που θα χρησιμοποιούν για

- Αρχικοποίηση και τερματισμό των ανταλλαγών δεδομένων
- Συγχρονισμό αποστολέων και παραληπτών
- Αναγνώριση και διόρθωση σφαλμάτων μετάδοσης
- Μορφοποίηση και κωδικοποίηση των δεδομένων

Οι προδιαγραφές ενός πρωτοκόλλου περιλαμβάνουν τα ακόλουθα σημεία:

- Την υπηρεσία την οποία θα παρέχεται από το πρωτόκολλο. Για παράδειγμα ο σκοπός του πρωτοκόλλου είναι η μεταφορά αρχείων κειμένου ως ακολουθίες χαρακτήρων κατά μήκος μιας τηλεφωνικής γραμμής προσπαθώντας ταυτόχρονα να αντιμετωπίσουμε σφάλματα μετάδοσης. Το πρωτόκολλο ορίζεται για μετάδοση αρχείων σε full-duplex γραμμή δηλαδή επιτρέπει μεταδόσεις ταυτόχρονα και προς τις δυο κατευθύνσεις.
- Τις υποθέσεις σχετικά με το περιβάλλον στο οποίο εκτελείται το πρωτόκολλο. Συνήθως το περιβάλλον αυτό αποτελείται τουλάχιστον από δυο χρήστες που

ανταλλάζουν τα δεδομένα και ένα κανάλι επικοινωνίας. Οι χρήστες μπορεί να υποτεθεί ότι απλά υποβάλλουν μια αίτηση για την μεταφορά του αρχείου και περιμένουν την υλοποίηση της. Για το κανάλι μετάδοσης μπορούμε να υποθέσουμε ότι προκαλεί τυχαίες παραμορφώσεις στα μηνύματα αλλά δεν χάνει μηνύματα.

- Το λεξιλόγιο των μηνυμάτων που χρησιμοποιούνται για την υλοποίηση του πρωτοκόλλου

- Κωδικοποίηση κάθε μηνύματος στο λεξιλόγιο

- Οι κανόνες που επιβλέπουν την συνέπεια των μηνυμάτων που ανταλλάσσονται

ΚΕΦΑΛΑΙΟ 2: ΒΑΣΙΚΟΙ ΚΑΝΟΝΕΣ ΣΧΕΔΙΑΣΜΟΥ

Για την σωστή σχεδίαση ενός πρωτοκόλλου εφαρμόζουμε τους ακόλουθους βασικούς κανόνες:

2.1 Απλότητα (simplicity)

Ένα καλά δομημένο πρωτόκολλο μπορεί να κατασκευαστεί από ένα μικρό αριθμό καλά σχεδιασμένων τμημάτων. Για την κατανόηση της λειτουργιάς πρωτοκόλλων θα ήταν αρκετό να κατανοήσουμε την λειτουργία αυτών των τμημάτων καθώς και τον τρόπο με τον οποίο αυτά αλληλεπιδρούν μεταξύ τους. Σύνθετα πρωτόκολλα δεν είναι εύκολο να υλοποιηθούν ούτε να υλοποιηθούν ενώ δεν εγγυώνται την αποτελεσματικότητα. Η σωστή σχεδίαση μπορεί να βοηθήσει στην αναγνώριση προβλημάτων, στον επιμερισμό και στην επίλυση τους.

2.2 Τμηματοποίηση (Modularity)

Πρέπει να γίνει προσδιορισμός της υπηρεσίας που θα προσφέρεται προτού αποφασιστεί ποιες δομές θα χρησιμοποιηθούν για την υλοποίηση αυτής της υπηρεσίας. Ένα πρωτόκολλο το οποίο εκτελεί μια σύνθετη λειτουργία μπορεί να κατασκευαστεί από απλούστερα τμήματα τα οποία αλληλεπιδρούν μεταξύ τους με έναν καλά ορισμένο και καλό τρόπο. Το κάθε μικρότερο κομμάτι είναι ένα ‘‘ελαφρύ’ το οποίο μπορεί να αναπτυχθεί, να επιβεβαιωθεί, να υλοποιηθεί και να συντηρηθεί ξεχωριστά από τα υπόλοιπα. Ορθογώνιες λειτουργίες δεν αναμιγνύονται και σχεδιάζονται ως ανεξάρτητες οντότητες. Ο έλεγχος λαθών και η ροή ελέγχου για παράδειγμα είναι ορθογώνιες λειτουργίες.

2.3 Καλά σχεδιασμένα πρωτόκολλα (well-formed protocols)

Ένα καλά σχεδιασμένο πρωτόκολλο δεν πρέπει να περιέχει μη εκτελέσιμο κώδικα. Επίσης δεν πρέπει να είναι ελλιπές ούτε μη ολοκληρωμένο. Ακόμη πρέπει να περιορίζεται (bounded) δηλαδή να μην υπερβαίνει τα όρια του συστήματος όπως για

παράδειγμα την περιορισμένη χωρητικότητα μιας ουράς μηνυμάτων. Ένα επιπλέον χαρακτηριστικό του είναι ότι πρέπει να είναι αυτό-σταθεροποιούμενο (self-stabilizing). Εάν ένα προσωρινό λάθος αλλάξει τυχαία την κατάσταση του πρωτοκόλλου, τότε ένα αυτο-σταθεροποιούμενο πρωτόκολλο πρέπει πάντα να επιστρέφει σε μια επιθυμητή κατάσταση εντός ενός πεπερασμένου αριθμού μεταβάσεων και να επανακτά την κανονική λειτουργία του. Ένα καλά σχεδιασμένο πρωτόκολλο πρέπει να είναι αυτο-προσαρμοζόμενο (self-adapting). Αυτό σημαίνει ότι πρέπει να προσαρμόζει για παράδειγμα το ρυθμό που στέλνει τα δεδομένα στο ρυθμό που μπορεί να μεταδώσει το κανάλι και στο ρυθμό που μπορεί να τα καταναλώσει ο δέκτης.

Επίσης θα πρέπει να γίνει σχεδιασμός της εξωτερικής λειτουργικότητας πριν την εσωτερική λειτουργικότητα. Η πρώτη θεώρηση της λύσης θα γίνει ως ένα «μαύρο κουτί» (black box) και μετά θα ληφθεί η απόφαση για το πώς θα αλληλεπιδρά με το περιβάλλον. Κατόπιν θα ληφθεί απόφαση για την εσωτερική οργάνωση του μαύρου κουτιού και τον καταμερισμό του σε μικρότερα ανάλογα δομικά στοιχεία. Τέλος θα πρέπει να αποφεύγεται η χρήση επουσιωδών τμημάτων.

2.4 Επίλυση πολλών και όχι μεμονωμένων προβλημάτων

Ένας καλός σχεδιασμός είναι εύκολα επεκτάσιμος και πρέπει να επιλύει μια κατηγορία προβλημάτων και όχι κάποιο μεμονωμένο πρόβλημα. Πριν την υλοποίηση του πρωτοκόλλου θα πρέπει να γίνεται σχεδιασμός του σε υψηλό επίπεδο και να επαληθεύεται ότι εκπληρώνει όλα τα κριτήρια σχεδιασμού.

2.5 Βελτιστοποίηση

Αφού γίνουν πολλές δοκιμές του πρωτοκόλλου, θα πρέπει να γίνεται μέτρηση των επιδόσεων του και στο βαθμό που είναι εφικτό, βελτιστοποίηση αυτών. Επίσης θα πρέπει να γίνεται έλεγχος αναφορικά με το ότι η τελική βελτιστοποιημένη υλοποίηση είναι ισοδύναμη με τον σχεδιασμό υψηλού επιπέδου που έτυχε της επαλήθευσης.

2.6 Ανεκτικότητα Από Λάθη (Robust)

Δεν είναι δύσκολο να σχεδιάσουμε πρωτόκολλα τα οποία λειτουργούν κάτω από κανονικές συνθήκες. Όμως η πρόκληση είναι να σχεδιάσουμε πρωτόκολλα που αντιμετωπίζουν το μη αναμενόμενο. Αυτό σημαίνει ότι το πρωτόκολλο πρέπει να

είναι προετοιμασμένο ώστε να αντιμετωπίζει κατάλληλα κάθε πιθανή κατάσταση κάτω από οποιεσδήποτε συνθήκες. Το πρωτόκολλο πρέπει να κάνει μόνο ελάχιστες υποθέσεις σχετικά με το περιβάλλον του προκειμένου να αποφύγει εξαρτήσεις από συγκεκριμένα χαρακτηριστικά τα οποία θα μπορούσαν δυνητικά να τροποποιηθούν. Ένας αξιόπιστος σχεδιασμός θα μπορεί να προσαρμόζεται σε οποιαδήποτε νέα τεχνολογία χωρίς να απαιτεί σημαντικές αλλαγές. Η καλύτερη ανεκτικότητα από λάθη επιτυγχάνεται όταν δεν απαιτείται να συμπληρώσουμε λειτουργικότητα για την αντιμετώπιση των νέων συνθηκών.

2.7 Συνέπεια (Consistency)

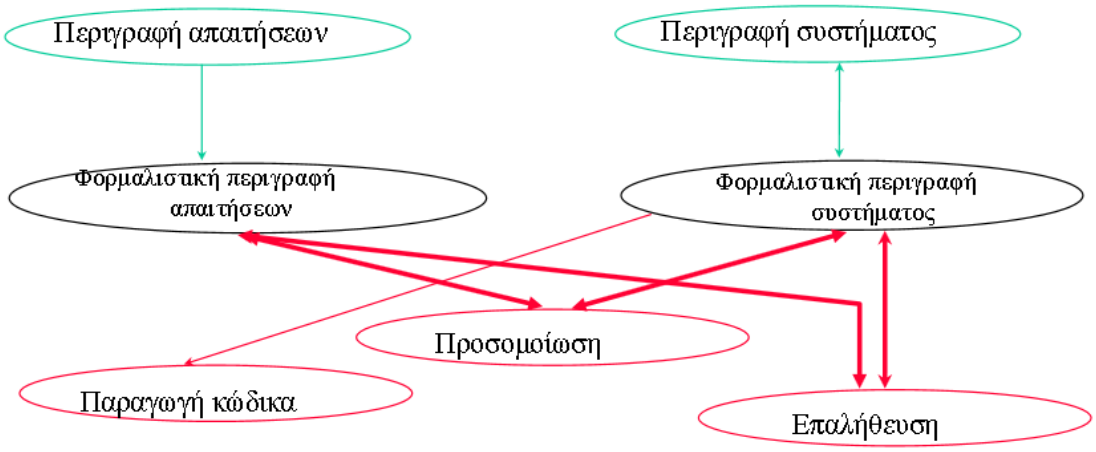
Υπάρχουν μερικοί standard τρόποι για τους οποίους ένα πρωτόκολλο θα μπορούσε να αποτύχει. Αναφέρουμε ακολούθως τους πιο σημαντικούς από αυτούς:

- Αδιέξοδα (deadlock). Αυτά είναι καταστάσεις στις οποίες δεν είναι δυνατή καμία περαιτέρω καμία ενέργεια του πρωτοκόλλου. Για παράδειγμα αυτό εμφανίζεται όταν όλες οι διεργασίες ενός πρωτοκόλλου περιμένουν για κάποιες συνθήκες οι οποίες ποτέ δεν θα εμφανιστούν
- Λιμοκτονίες (Livelocks). Αυτές είναι ακολουθίες εκτέλεσης οι οποίες μπορούν να επαναλαμβάνονται πάρα πολύ συχνά χωρίς να προκαλούν κάποια εξέλιξη
- Μη σωστοί τερματισμοί (Improper terminations). Αυτό είναι η ολοκλήρωση της εκτέλεσης ενός πρωτοκόλλου χωρίς να ικανοποιούνται όλες οι απαιτούμενες συνθήκες τερματισμού

Ένα πρωτόκολλο μπορεί να περιγραφεί σαν μια μηχανή καταστάσεων. Με τον όρο κατάσταση αναφερόμαστε στις ενέργειες που μπορεί να εκτελέσει μια διεργασία, στα γεγονότα που περιμένει να συμβούν καθώς και το πώς θα αντιδράσει σε αυτά τα γεγονότα. Επίσης πρέπει να γίνει σαφής προσδιορισμός του προβλήματος και απαρίθμηση όλων των κριτηρίων, απαιτήσεων και περιορισμών πριν από την έναρξη του σχεδιασμού.

Κεφαλαιο 3: Μεθοδολογια Σχεδιασμου Πρωτοκολλων

Η μεθοδολογία σχεδίασης πρωτοκόλλων απεικονίζεται στο ακόλουθο διάγραμμα:



Όπως παρατηρούμε η περιγραφή των απαιτήσεων που θα πρέπει να ικανοποιεί το πρωτόκολλο ουσιαστικά ξεκινάει ταυτόχρονα με την περιγραφή του συστήματος. Πρώτα περιγράφονται φορμαλιστικά (τυπικά και με λεπτομέρεια) οι απαιτήσεις του πρωτοκόλλου καθώς και οι απαιτήσεις του συστήματος που το χρησιμοποιεί. Στη συνέχεια γίνεται η υλοποίηση του συστήματος κατασκευάζοντας τον αντίστοιχο κώδικα. Μετά γίνεται η προσομοίωση του συστήματος για να διαπιστώσουμε το πώς λειτουργεί στην πράξη ελέγχοντας ταυτόχρονα αν πληροί τις απαιτήσεις του πρωτοκόλλου καθώς και τις δικές του απαιτήσεις. Με βάση τα αποτελέσματα της προσομοίωσης είναι δυνατόν να επαναπροσδιοριστούν τόσο οι απαιτήσεις του πρωτοκόλλου όσο και οι απαιτήσεις του συστήματος. Αυτό μπορεί να οδηγήσει σε τροποποίηση – βελτιστοποίηση του ήδη υπάρχοντος κώδικα. Σε κάθε περίπτωση η προσομοίωση μπορεί να βοηθήσει σε σημαντικές βελτιώσεις στη σχεδίαση του πρωτοκόλλου και των απαιτήσεων.

Κεφαλαιο 4: Έλεγχος Ροης

Η απλούστερη μορφή μιας διάταξης ελέγχου ροής ελέγχει τον ρυθμό βάση του οποίου ένας αποστολέας παράγει δεδομένα σε σχέση με τον ρυθμό που μπορεί να τα δέχεται ένας παραλήπτης. Πρέπει να εξετάζεται η απλούστερη εκδοχή του προβλήματος που χρησιμοποιείται έτσι ώστε να ικανοποιούνται οι ακόλουθοι στόχοι:

1. Να παρέχεται διαβεβαίωση ότι τα δεδομένα δεν αποστέλλονται με γρηγορότερο ρυθμό από αυτόν που μπορούν να επεξεργάζονται

2. Να παρέχεται βελτιστοποίηση της χρήσης του καναλιού

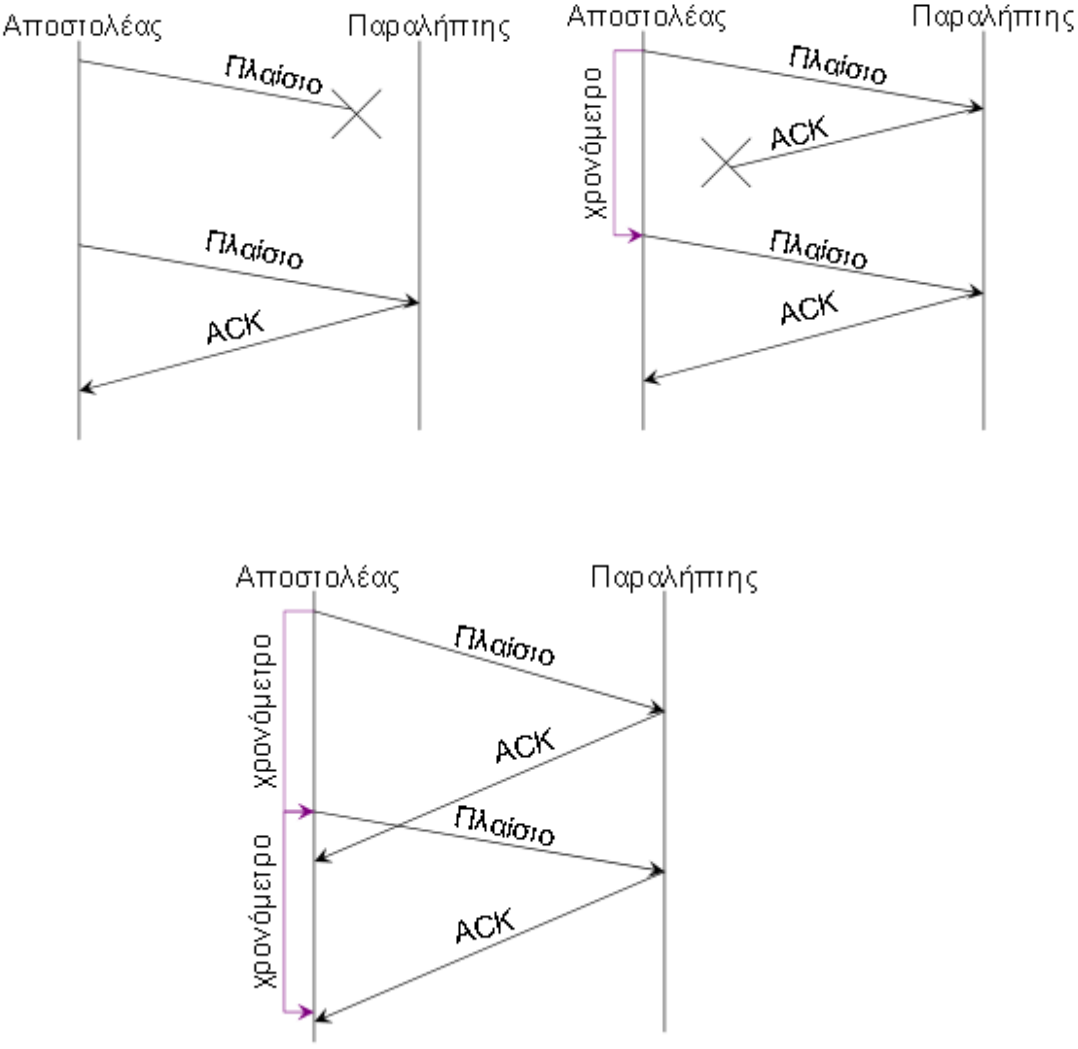
3. Να αποφεύγεται η υπερφόρτωση των ζεύξεων μετάδοσης με δεδομένα

Από τους τρεις στόχους που προαναφέραμε ο δεύτερος και ο τρίτος είναι συμπληρωματικοί, δηλαδή η αποστολή δεδομένων με πολύ αργό ρυθμό είναι άσκοπη. Από την άλλη μεριά η πολύ γρήγορη αποστολή δεδομένων μπορεί να προκαλέσει συμφόρηση.

Η διαδρομή δεδομένων μεταξύ αποστολέα και παραλήπτη μπορεί να περιέχει σημεία μεταφοράς με μια περιορισμένη χωρητικότητα, συγκριτικά με την αποθήκευση μηνυμάτων μεταξύ ορισμένων ζευγών αποστολέα – παραλήπτη. Μια πιο «συνετή» διάταξη ελέγχου ροής παρεμποδίζει ένα τέτοιο ζεύγος από το να καταλαμβάνει όλο το διαθέσιμο εύρος ζώνης.

Η αξιόπιστη μετάδοση περιλαμβάνει τη χρήση πακέτων επαλήθευσης (ACK) και χρονομέτρων (timers). Ο δέκτης κάθε φορά που παραλαμβάνει ένα πακέτο χωρίς σφάλματα στέλνει ένα πακέτο επιβεβαίωσης-επαλήθευσης στον αποστολέα. Ο αποστολέας συνεχίζει να στέλνει πλαίσια ενόσω λαμβάνει ACK. Εάν περάσει μια χρονική περίοδος κατά την οποία ο αποστολέας δεν λάβει κανένα ACK ή αν λάβει NACK τότε ξαναστέλνει τα προηγούμενα πακέτα.

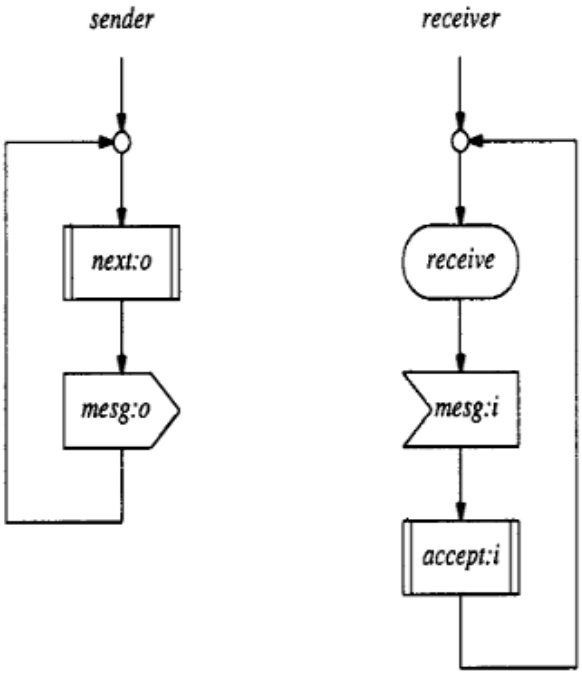
Το τι ακριβώς μπορεί να πάει λάθος φαίνεται στο ακόλουθο παράδειγμα:


Αποστολέας
Παραλήπτης
Πλαίσιο
Πλαίσιο
ACK
Αποστολέας
Παραλήπτης
χρονόμετρο
Πλαίσιο
ACK
Πλαίσιο
ACK
Αποστολέας
Παραλήπτης
χρονόμετρο
Πλαίσιο
ACK
Πλαίσιο
χρονόμετρο
ACK

ΚΕΦΑΛΑΙΟ 5:
ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΠΑΡΑΔΕΙΓΜΑΤΑ ΠΡΩΤΟΚΟΛΛΩΝ

5.1 Πρωτόκολλο χωρίς Έλεγχο Ροής

Έστω ότι χρησιμοποιούμε ένα απλό πρωτόκολλο, χωρίς καμία μορφή ελέγχου ροής που χρησιμοποιείται για την μεταφορά δεδομένων μόνο στην μια κατεύθυνση. Το πρωτόκολλο αυτό λειτουργεί αξιόπιστα μόνο αν παρέχονται εγγυήσεις ότι η διεργασία του παραλήπτη είναι ταχύτερη από εκείνη του αποστολέα. Επίσης δεν υπάρχει κανένας έλεγχος ούτε από το δέκτη σχετικά με το αν έχει ήδη λάβει το μήνυμα αυτό ούτε από τον πομπό ο οποίος στέλνει νέο πακέτο χωρίς να έχει λάβει κάποια επιβεβαίωση για το προηγούμενο.



Εικόνα 1-Πρωτόκολλο χωρίς Έλεγχο Ροής

5.2 Πρωτόκολλο Τύπου X – on / X – off

Η παλιότερη τεχνική ελέγχου ροής που μπορεί να χρησιμοποιηθεί για την αντιμετωπίσει αυτού του προβλήματος συγχρονισμού δεν απαιτεί καμία προηγούμενη διαπραγμάτευση μεταξύ αποστολέα και παραλήπτη σχετικά με τον ρυθμό που μπορούν να εκπέμπονται τα δεδομένα. Η μέθοδος αυτή χρησιμοποιεί δυο μηνύματα ελέγχου:

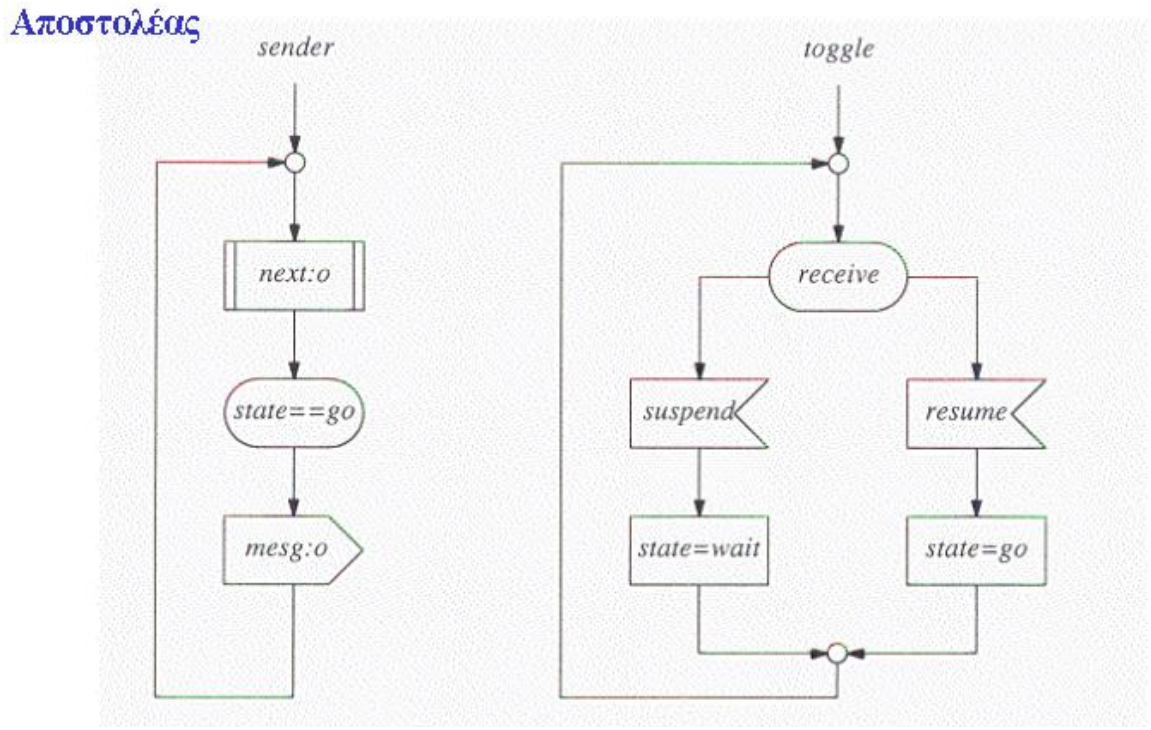
- Ένα μήνυμα για αναστολή της λειτουργίας (suspend) και
- ένα μήνυμα για συνέχιση της πληροφορίας (resume)

Τα δυο αυτά μηνύματα είναι γνωστά και ως x – off και x – on αντίστοιχα

Ας υποθέσουμε ότι έχουμε ένα κανάλι το οποίο δεν δημιουργεί σφάλματα κατά την μετάδοση και ένα λεξιλόγιο πρωτοκόλλου με τους ακόλουθους τύπους μηνυμάτων:

V = {message, suspend, resume}

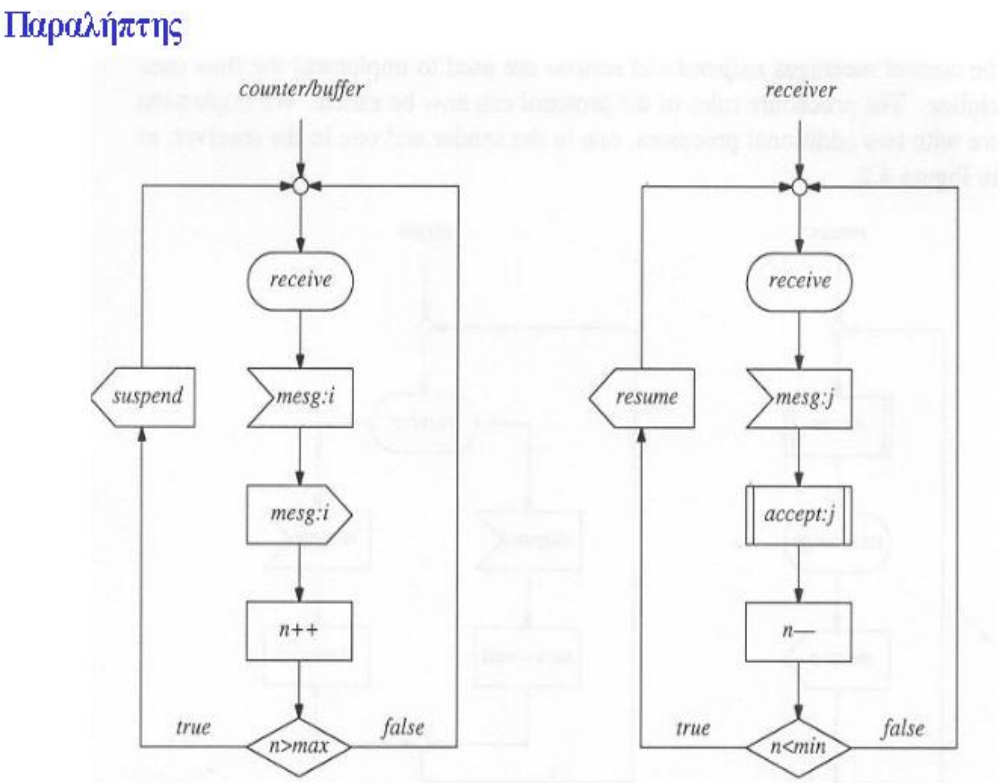
Τα δυο μηνύματα suspend και resume χρησιμοποιούνται για την υλοποίηση της μεθόδου ελέγχου ροής. Στη συνέχεια μπορούμε να προσθέσουμε τους διαδικαστικούς κανόνες στο πρωτόκολλο. Για την υλοποίηση του πρωτοκόλλου θα προσθέσουμε δύο επιπλέον διαδικασίες, μια στο αποστολέα και μια στο παραλήπτη, όπως φαίνεται στις ακόλουθες εικόνες:



Εικόνα 2 – Πρωτόκολλο X-on/X-off: Λειτουργία Πομπού

Στο αριστερό διάγραμμα ροής του αποστολέα παρατηρούμε ότι η διεργασία του αποστολέα εκτελεί ένα βρόγχο κάθε φορά που έχει μήνυμα προς μετάδοση. Συγκεκριμένα αν η κατάσταση του είναι «ενεργή» (state=go) τότε στέλνει το μήνυμα αυτό και εξετάζει αν έχει νέο μήνυμα προς μετάδοση.

Στο δεξιό διάγραμμα ροής του αποστολέα παρατηρούμε ότι η διεργασία του αποστολέα εκτελεί ένα βρόγχο εξετάζοντας αν έχει μήνυμα προς μετάδοση. Κάθε φορά που έχει ένα τέτοιο μήνυμα αρχίζει τη λειτουργία του (resume) που είναι η αποστολή του μηνύματος και η κατάσταση του δηλώνεται ως «ενεργή» (state=go). Αν δεν έχει τέτοιο μήνυμα, τότε μπαίνει σε αναστολή εκτέλεσης (suspend) και η κατάσταση του δηλώνεται ως «αναμένουσα» (state=wait).



Εικόνα 3 – Πρωτόκολλο X-on/X-off: Λειτουργία Δέκτη

Στο δεξιό διάγραμμα ροής του παραλήπτη παρατηρούμε ότι η διεργασία του παραλήπτη εκτελεί ένα βρόγχο κάθε φορά που λαμβάνει ένα μήνυμα. Συγκεκριμένα ανοίγει-χρησιμοποιεί το μήνυμα που έχει λάβει και μειώνει το πλήθος των ληφθέντων μηνυμάτων που έχει λάβει στο buffer του κατά ένα. Μετά ελέγχει αν το πλήθος των μηνυμάτων που έχει λάβει στο buffer του είναι μικρότερο από ένα ελάχιστο όριο μηνυμάτων που πρέπει να λάβει και αν ναι τότε συνεχίζει τη λήψη μηνυμάτων. Αν όχι αυτό σημαίνει ότι έχει ήδη λάβει τον ελάχιστο αριθμό μηνυμάτων που πρέπει και ουσιαστικά (χωρίς αυτό να φαίνεται άμεσα στο σχήμα) μπαίνει σε κατάσταση αναστολής (resume).

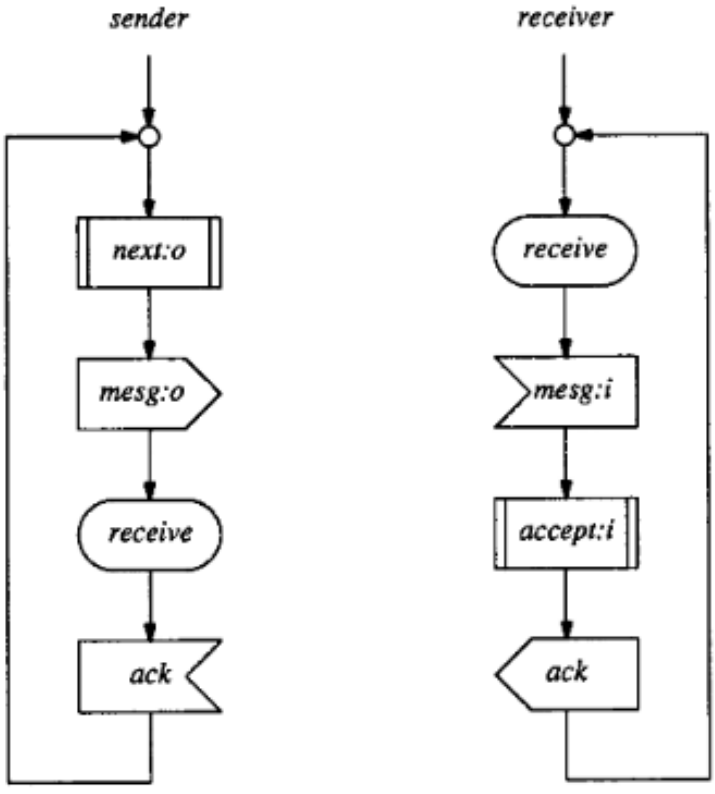
Στο αριστερό διάγραμμα ροής του παραλήπτη παρατηρούμε ότι κάθε φορά που παραλαμβάνει ένα μήνυμα από το κανάλι μετάδοσης χρησιμοποιεί το μήνυμα αυτό και αυξάνει το πλήθος των ληφθέντων μηνυμάτων στο buffer του. Αν ο αριθμός αυτός υπερβαίνει το μέγιστο πλήθος μηνυμάτων που μπορεί να αποθηκεύει ο buffer του, τότε ο παραλήπτης μπαίνει σε κατάσταση αναστολής (suspend), αλλιώς, αν ο buffer του παραλήπτη δεν είναι γεμάτος, τότε αυτός συνεχίζει τη λειτουργία του.

Βασικά προβλήματα που πρέπει να επιλυθούν στο πρωτόκολλο αυτό είναι τα εξής:

- Η ορθή λειτουργία του πρωτοκόλλου εξαρτάται κατά βάση από το κανάλι μετάδοσης (εάν χαθεί ή καθυστερήσει ένα μήνυμα αναστολής)
- Η λειτουργία πρωτοκόλλου δεν θα πρέπει να εξαρτάται από τον χρόνο που χρειάζεται για να φτάσει στον παραλήπτη ένα μήνυμα ελέγχου
- Στη χειρότερη περίπτωση αν χαθεί ένα μήνυμα resume το σύστημα επεξεργασίας το οποίο αποτελείται από τέσσερις επιμέρους διεργασίες σταματάει την λειτουργία του

Για αυτό θα πρέπει να υπάρχει μια προστασία έναντι προβλημάτων καθυστέρησης κατά την μετάδοση με ένα περισσότερο αξιόπιστο τρόπο. Επίσης θα πρέπει να υπάρχει προστασία έναντι της απώλειας μηνυμάτων.

5.3 Πρωτόκολλα με Παράθυρο



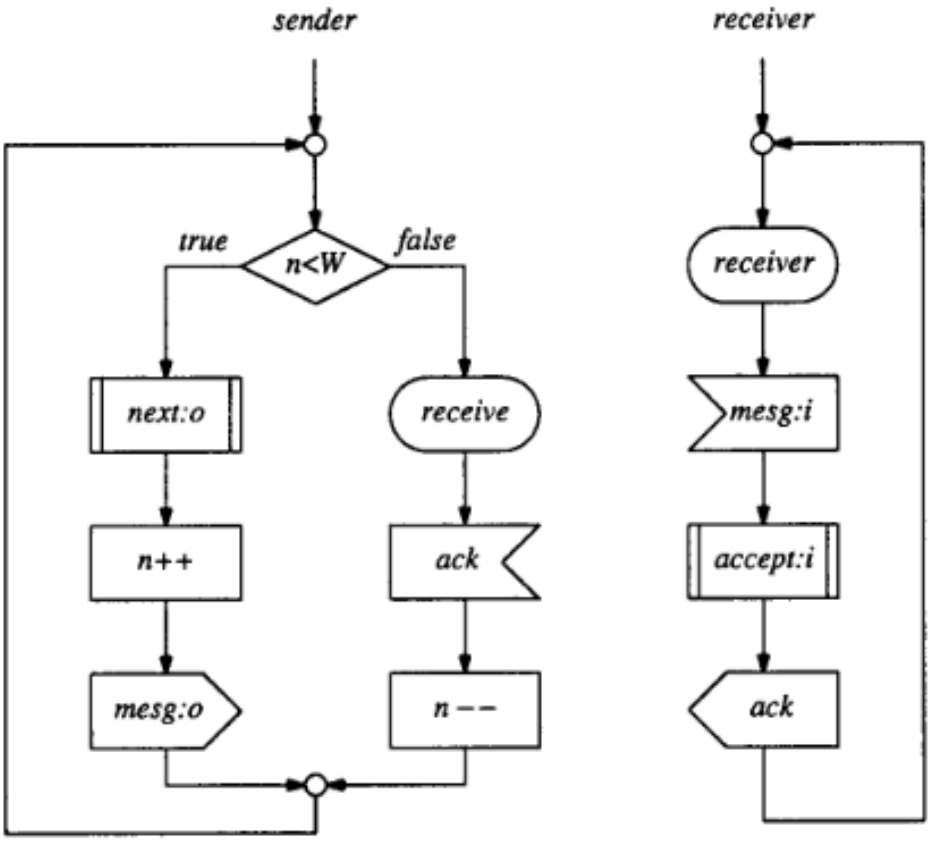
Εικόνα 4-Πρωτόκολλο Ping Pong

Στο δύο διαγράμματα ροής φαίνεται η λειτουργία ενός κλασσικού πρωτοκόλλου που ονομάζεται Stop-and-Wait στο οποίο ο αποστολέας στέλνει ένα πακέτο και περιμένει να λάβει μήνυμα επιβεβαίωσης από τον παραλήπτη ότι έλαβα το πακέτο του προτού στείλει το επόμενο πακέτο. Αντίστοιχα ο παραλήπτης-δέκτης

περιμένει να λάβει ένα μήνυμα και αφού το λάβει στέλνει μήνυμα επιβεβαίωσης στον πομπό. Το πρόβλημα που λύνουμε με το πρωτόκολλο αυτό είναι το πρόβλημα συγχρονισμού του αποστολέα με τον παραλήπτη το οποίο επιτυγχάνεται με την αποστολή των επιβεβαιώσεων (ACK) και από τους δύο. Αυτό όμως που παραμένει ως πρόβλημα είναι η απώλεια πακέτων μέσα στο κανάλι καθώς αν ένα πακέτο χαθεί κατά τη μετάδοση του, τότε ο δέκτης δεν θα το λάβει ποτέ και δεν θα στείλει καμία επιβεβαίωση γιαυτό, ενώ ο πομπός θα περιμένει συνεχώς να λάβει επιβεβαίωση για να στείλει το επόμενο πακέτο.

Στο πρωτόκολλο με παράθυρο υπάρχει μια φάση αρχικοποίησης στην οποία ο παραλήπτης ενημερώνει τον αποστολέα σχετικά με το μέγεθος του buffer του για τα εισερχόμενα μηνύματα. Στον αποστολέα δίνεται ένα μέγεθος για ένα σταθερό αριθμό μηνυμάτων που μπορεί να αποθηκεύσει. Το μέγεθος αυτό μπορεί να μεταβάλλεται δυναμικά όταν μεταβάλλεται και το μέγεθος της διαθέσιμης μνήμης στον παραλήπτη. Κάθε μήνυμα που λαμβάνεται από τον παραλήπτη επιβεβαιώνεται με ένα μήνυμα επιβεβαίωσης (ACK). Είναι απαραίτητο να λαμβάνονται επιτυχώς όλα τα μηνύματα που βρίσκονται μέσα στο παράθυρο του αποστολέα προτού ο αποστολέας «ολισθήσει» στο παράθυρο του και μεταδώσει νέα μηνύματα. Το παράθυρο εκφράζει το μέγιστο αριθμό μηνυμάτων που μπορεί να μεταδώσει ο αποστολέας χωρίς να περιμένει επιβεβαίωση.

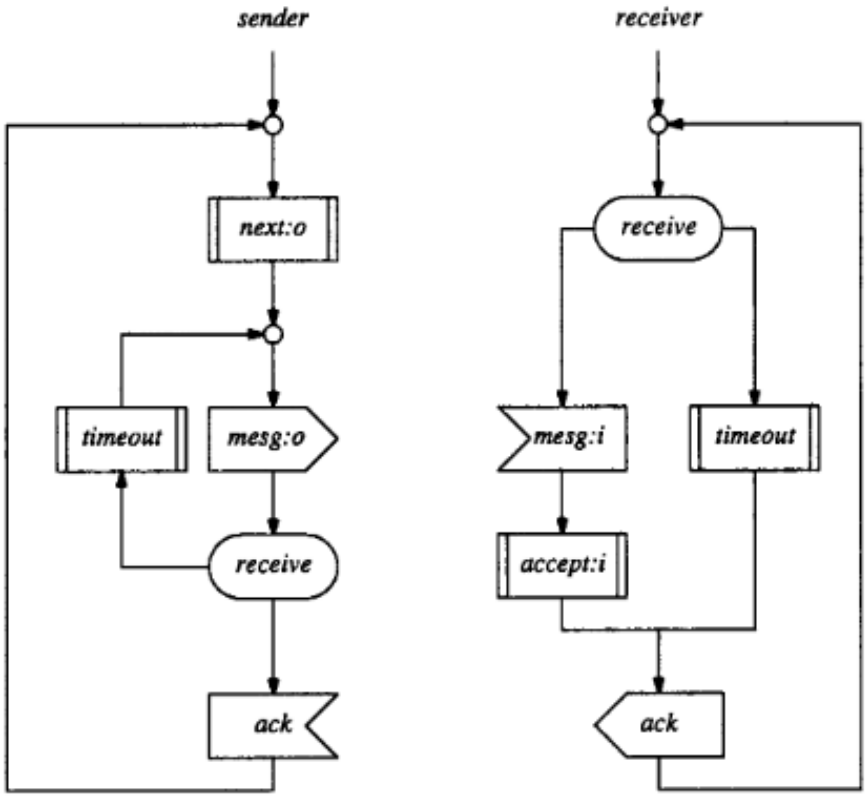
Η λειτουργία του πρωτοκόλλου με παράθυρο φαίνεται στην ακόλουθη εικόνα:



Εικόνα 5-Πρωτόκολλο Παραθύρου για Ιδανικό Κανάλι

Όπως παρατηρούμε στην διεργασία του αποστολέα όταν το πλήθος των μηνυμάτων που έχει στείλει δεν υπερβαίνει το μέγεθος του παραθύρου τότε στέλνει ένα μήνυμα και αυξάνει τον μετρητή των απεσταλμένων μηνυμάτων. Διαφορετικά αν έχει ολοκληρώσει την μετάδοση όλων των πακέτων του παραθύρου του τότε λαμβάνει νέα πακέτα στο παράθυρό του προκειμένου να τα μεταδώσει. Ο αποστολέας καθυστερεί μόνο όταν το μέγεθος του παραθύρου του φτάσει στο μηδέν. Αντίστοιχα ο δέκτης κάθε φορά που λαμβάνει ένα μήνυμα στέλνει ένα μήνυμα επιβεβαίωσης στον πομπό. Το πρόβλημα στην επικοινωνία αποστολέα και παραλήπτη είναι ότι ο αποστολέας δεν γνωρίζει για πόσο χρονικό διάστημα πρέπει να περιμένει μέχρι να λάβει επιβεβαίωση. Για την αντιμετώπιση της πιθανής απώλειας μηνυμάτων ή επιβεβαιώσεων θα πρέπει να τοποθετηθεί ένας χρονιστής (timer) στην μεριά του αποστολέα ο οποίος να καθορίζει το χρονικό διάστημα που πρέπει να περιμένει ο αποστολέας μέχρι να λάβει επιβεβαίωση. Το χρονικό αυτό διάστημα ονομάζεται timeout και αν συμπληρωθεί χωρίς να πάρει επιβεβαίωση τότε επαναμεταδίδει το

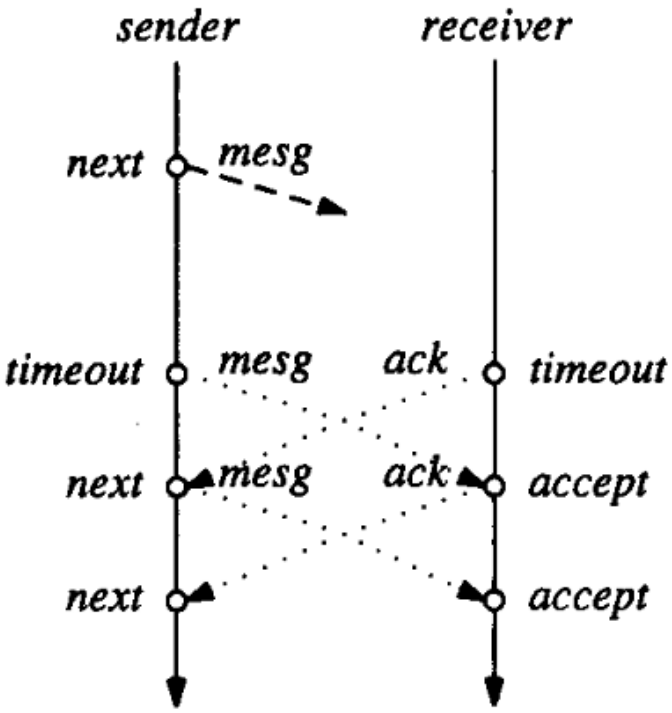
μήνυμα αυτό. Αντίστοιχα μπορούμε να χρησιμοποιήσουμε και έναν timer στην μεριά του δέκτη ο οποίος θα μετράει το χρονικό διάστημα εντός του οποίου θα περιμένει να λάβει μήνυμα. Αν συμπληρωθεί το διάστημα αυτό χωρίς να πάρει μήνυμα τότε ο παραλήπτης στέλνει ένα δικό του μήνυμα ACK στον πομπό ζητώντας του να στείλει πακέτο. Η επικοινωνία αποστολέα παραλήπτη με χρήση timeout φαίνεται στην ακόλουθη εικόνα.



Εικόνα 6 Πρωτόκολλο με χρήση Timeout

Ένα επιπλέον χαρακτηριστικό της επικοινωνίας με παράθυρο είναι ότι θα πρέπει οι επιβεβαιώσεις του δέκτη να περιλαμβάνουν και έναν αριθμό ακολουθίας (sequence number) ώστε να γνωρίζει ο πομπός ποιο θα είναι το επόμενο πακέτο που θα πρέπει να στείλει. Το εύρος των διαθέσιμων αριθμών ακολουθίας θα πρέπει να είναι μεγαλύτερο από το εύρος του παραθύρου προκειμένου να αποφεύγονται προβλήματα σύγχυσης με ανακυκλωμένους αριθμούς ακολουθίας.

Η επικοινωνία πομπού-δέκτη σε ένα πρωτόκολλο παύσης και αναμονής με αποστολή μηνυμάτων και επιβεβαιώσεων φαίνεται στην ακόλουθη εικόνα:



Εικόνα 7- Πρωτόκολλο με χρήση Timeout και Ανταλλαγή Μηνυμάτων

Στην εικόνα αυτή παρατηρούμε τα ακόλουθα:

1.Ο πομπός στέλνει αρχικά ένα μήνυμα το οποίο χάνεται κατά τη μετάδοση του στο κανάλι. Ο πομπός αρχικοποιεί ένα χρονιστή (timer) τη στιγμή που αρχίζει να στέλνει το πακέτο και περιμένει ένα συγκεκριμένο χρονικό διάστημα (timeout) να λάβει επιβεβαίωση (ack) από το δέκτη. Αφού παρέλθει το χρονικό διάστημα χωρίς να λάβει κάποια επιβεβαίωση από το δέκτη, επαναμεταδίδει το πακέτο του.

2.Ο δέκτης είναι συντονισμένος με τον πομπό και τη στιγμή που ο πομπός ξεκινά τη μετάδοση, ο δέκτης αρχικοποιεί από τη μεριά του ένα δικό του timer περιμένοντας ένα συγκεκριμένο χρονικό διάστημα να λάβει το πακέτο του πομπού. Όπως φαίνεται και στο σχήμα, το timeout στο δέκτη ολοκληρώνεται χωρίς να λάβει πακέτο, οπότε ο δέκτης στέλνει ένα μήνυμα ack στον πομπό ζητώντας του να του στείλει πακέτο

3.Το πακέτο που στέλνει ο πομπός λαμβάνεται τώρα από το δέκτη (accept) εντός του timeout, οπότε ο δέκτης απαντά πάλι με ένα μήνυμα ack.

4.Ο πομπός κάθε φορά που λαμβάνει ack από το δέκτη μεταδίδει το επόμενο πακέτο.

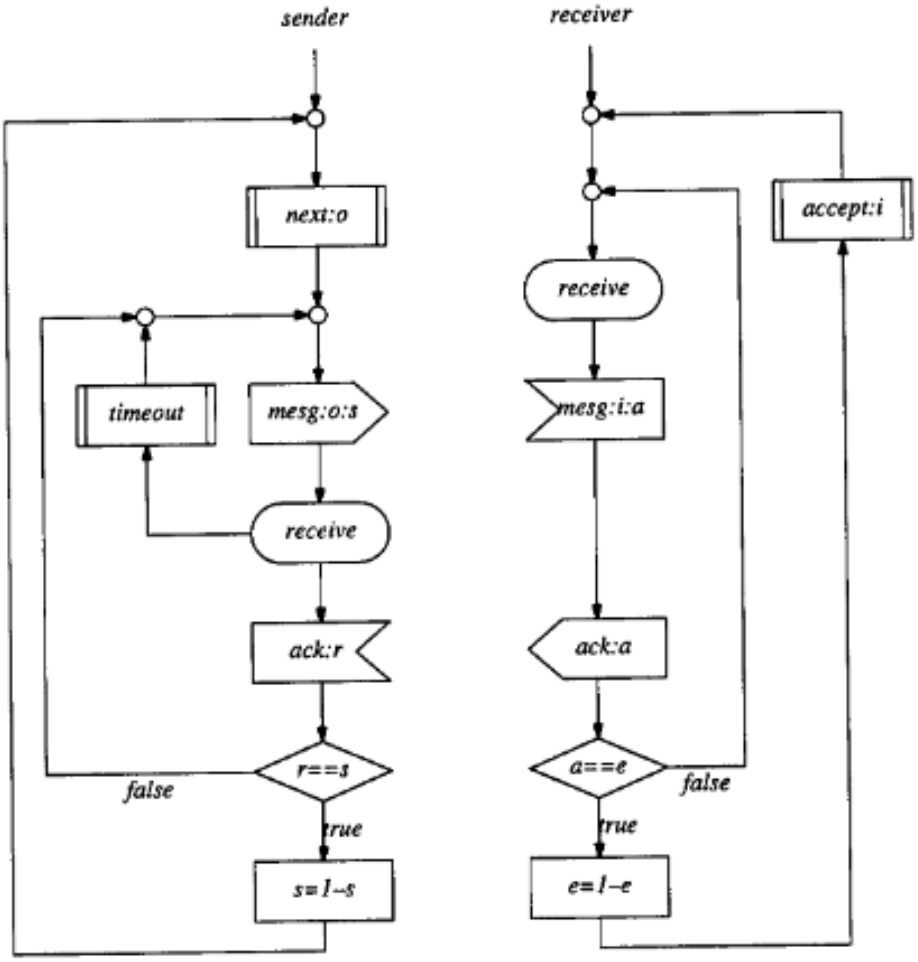
Βέβαια το βασικό πρόβλημα του πρωτοκόλλου αυτού είναι ότι δεν υπάρχουν αριθμοί (sequence numbers) τόσο στα πακέτα που στέλνονται από τον πομπό όσο και στις επιβεβαιώσεις που λαμβάνονται από το δέκτη με αποτέλεσμα ο πομπός να επαναμεταδίδει τα ίδια πακέτα ή ο δέκτης να επιβεβαιώνει πακέτα τα οποία έχει ήδη λάβει. Με τη χρήση χρονιστή στον πομπό και στο δέκτη αντιμετωπίζουμε την πιθανή απώλεια μηνυμάτων-πακέτων αλλά δεν αντιμετωπίζουμε την αποστολή ίδιων μηνυμάτων και επιβεβαιώσεων, γεγονός που επιβαρύνει το δίκτυο με αποστολή της ίδιας πληροφορίας πολλές φορές, Τα προβλήματα αυτά λύνονται στη συνέχεια με το πρωτόκολλο εναλλασσόμενου bit.

5.4 Πρωτόκολλο Εναλλασσόμενου Bit με Timeout

Στο πρωτόκολλο αυτό χρησιμοποιούνται δυο τύποι μηνυμάτων, mesg και ack τα οποία έχουν την ακόλουθη μορφή

- **{mesg, data, sequence number}** όπου η λέξη mesg καθορίζει τον τύπο του μηνύματος και συγκεκριμένα ότι είναι πακέτο δεδομένων, η λέξη data περιλαμβάνει το περιεχόμενο του πακέτου και η λέξη sequence number είναι ο αριθμός του πακέτου δεδομένων που στέλνεται

- **{ack, sequence number}** όπου η λέξη ack καθορίζει τον τύπο του μηνύματος και συγκεκριμένα ότι είναι μήνυμα επιβεβαίωσης και η λέξη sequence number είναι ο αριθμός του πακέτου επιβεβαίωσης



Εικόνα 8-Πρωτόκολλο Εναλλασσόμενου Bit με Timeout

Για τη λειτουργία του συγκεκριμένου πρωτοκόλλου χρησιμοποιούνται 4 μεταβλητές του ενός bit: οι a, e, r, s με το εξής νόημα (και οι 4 μεταβλητές αρχικοποιούνται στο μηδέν):

s (sender) → χρησιμοποιείται από τον αποστολέα για να αποθηκεύει τον τελευταίο αριθμό ακολουθίας πακέτου που στάλθηκε

r (received) → χρησιμοποιείται από τον αποστολέα για να αποθηκεύει τον τελευταίο αριθμό ακολουθίας πακέτου που παραλήφθηκε

e (expected) → χρησιμοποιείται από τον παραλήπτη για να συγκρατήσει τον επόμενο αριθμό πακέτου που αναμένεται να αφιχθεί

a (last) → χρησιμοποιείται από τον παραλήπτη για να αποθηκεύσει τον τελευταίο πραγματικό αριθμό ακολουθίας πακέτου που έχει ληφθεί

Στην Εικόνα αυτή παρατηρούμε ότι ο πομπός περιμένει όπως και πριν για ένα συγκεκριμένο χρονικό διάστημα (timeout) από τη στιγμή που έστειλε το πακέτο του προκειμένου να λάβει μια επιβεβαίωση. Όταν λάβει μια επιβεβαίωση (ack) χρησιμοποιεί τη μεταβλητή r και τη μεταβλητή s προκειμένου να επαληθεύσει ότι η επιβεβαίωση που έλαβε αφορά το σωστό πακέτο που έστειλε και στην περίπτωση που δεν συμβεί αυτό, τότε επαναμεταδίδει το πακέτο. Αν έλαβε επιβεβαίωση για το σωστό πακέτο τότε ο αριθμός του επόμενου πακέτου που θα στείλει είναι ο s=1-s. Επειδή αρχικά το s=0, τότε το επόμενο πακέτο που θα στείλει είναι το 1, ενώ αν λάβει επιβεβαίωση για το πακέτο με αριθμό 1 τότε το επόμενο πακέτο που θα στείλει θα είναι το s=1-s δηλ. το πακέτο με αριθμό 0. Άρα ο πομπός θα στέλνει είτε το πακέτο με αριθμό ακολουθίας 0 είτε το πακέτο με αριθμό ακολουθίας 1.

Συνοψίζοντας ο πομπός κάνει δύο βασικές επιβεβαιώσεις:

- Πρώτον ότι δεν έχει παρέλθει το timeout εντός του οποίου πρέπει να λάβει επιβεβαίωση και
- Δεύτερον ότι η επιβεβαίωση που λαμβάνει αφορά το σωστό πακέτο που έστειλε

Άρα με το πρωτόκολλο εναλλασσόμενου bit λύνονται τα προβλήματα που αντιμετώπιζε ο πομπός στο πρωτόκολλο παραθύρου.

Επίσης στην Εικόνα αυτή παρατηρούμε αντίστοιχα ότι όταν ο δέκτης λάβει ένα πακέτο εξετάζει αν ο αριθμός του πακέτου που έλαβε είναι ίσος με τον αριθμό πακέτου που αναμένει να λάβει και αν όχι τότε απορρίπτει το ληφθέν πακέτο. Αν ο δέκτης έλαβε το σωστό πακέτο τότε ο αριθμός του επόμενου πακέτου που θα λάβει είναι ο e=1-e.

Επειδή αρχικά το e=0, τότε το επόμενο πακέτο που θα λάβει είναι το 1, ενώ αν λάβει επιβεβαίωση για το πακέτο με αριθμό 1 τότε το επόμενο πακέτο που θα αναμένει θα είναι το e=1-e δηλ. το πακέτο με αριθμό 0. Άρα ο δέκτης θα λαμβάνει είτε το πακέτο με αριθμό ακολουθίας 0 είτε το πακέτο με αριθμό ακολουθίας 1.

Συνοψίζοντας ο δέκτης μια βασική επιβεβαίωση:

- Ότι το πακέτο που παραλαμβάνει αφορά ακριβώς αυτό που ανέμενε

Άρα με το πρωτόκολλο εναλλασσόμενου bit λύνονται τα προβλήματα που αντιμετώπιζε και ο δέκτης στο πρωτόκολλο παραθύρου.

5.5 Προβλήματα Πρωτοκόλλων με Παράθυρο

Σε δίκτυα μεταγωγής πακέτων υπάρχουν περιπτώσεις αντιγραφής (duplication) και επαναδιάταξης (reordering) μηνυμάτων. Συγκεκριμένα ο δέκτης μπορεί να λάβει πολλές φορές το ίδιο πακέτο ή μπορεί να λάβει τα πακέτα ενός μηνύματος με λάθος σειρά. Μια προφανής λύση που αντιμετωπίζει τα παραπάνω προβλήματα είναι η χρήση ενός μεγάλου αριθμού ακολουθίας ο οποίος προσαρτάται σε κάθε μήνυμα. Πχ με 16 bit αριθμούς ακολουθίας μπορούμε να αναπαραστήσουμε 65536 διαφορετικά πακέτα. Επίσης ο αποστολέας πρέπει να θυμάται κάθε μήνυμα που είναι σε εκκρεμότητα ώστε να το επαναμεταδώσει. Για το σκοπό αυτό χρησιμοποιεί δύο πίνακες:

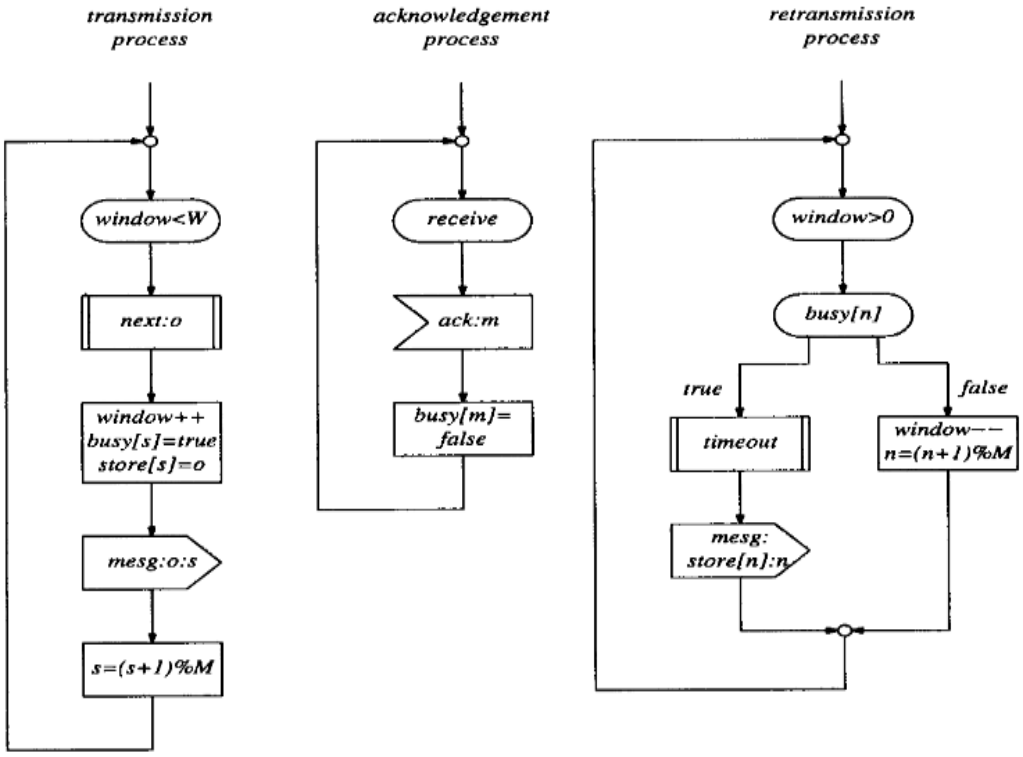
- **busy[s]** → true εάν ένα μήνυμα με αριθμό ακολουθίας εστάλη και δεν έχει ακόμα γνωστοποιηθεί
- **store[s]** →αποθηκεύουμε το μήνυμα με αριθμό ακολουθίας s, που εκπέμφθηκε

Επίσης χρησιμοποιούνται δύο σταθερές η M η οποία είναι μια περιοχή – εύρος των διαθέσιμων πηγών αριθμών ακολουθίας και η W η οποία ορίζει μια αρχική πίστωση μηνυμάτων. Το M είναι επαρκώς μεγαλύτερο από το W, για την αποφυγή σύγχυσης με τους ανακυκλωμένους αριθμούς ακολουθίας (recycled sequence numbers). Επιπλέον με τις σταθερές W και M χρησιμοποιούνται και οι ακόλουθες μεταβλητές οι οποίες αρχικά τίθενται όλες στο μηδέν:

- **s**: είναι ο αριθμός ακολουθίας του επόμενου μηνύματος που πρόκειται να σταλεί
- **window**: είναι ο αριθμός των μηνυμάτων που βρίσκονται σε εκκρεμότητα, δηλαδή δεν έχουν ακόμα επιβεβαιωθεί
- **n**: είναι ο αριθμός ακολουθίας του παλαιότερου μηνύματος ακολουθίας που δεν έχει επιβεβαιωθεί
- **m**: είναι ο αριθμός ακολουθίας του τελευταίου μηνύματος ακολουθίας που έχει επιβεβαιωθεί

5.6 Διαδικασίες Πρωτοκόλλων με Παράθυρο

Οι βασικές διαδικασίες που εκτελούνται σε πρωτόκολλα με παράθυρο περιγράφονται στις επόμενες εικόνες.

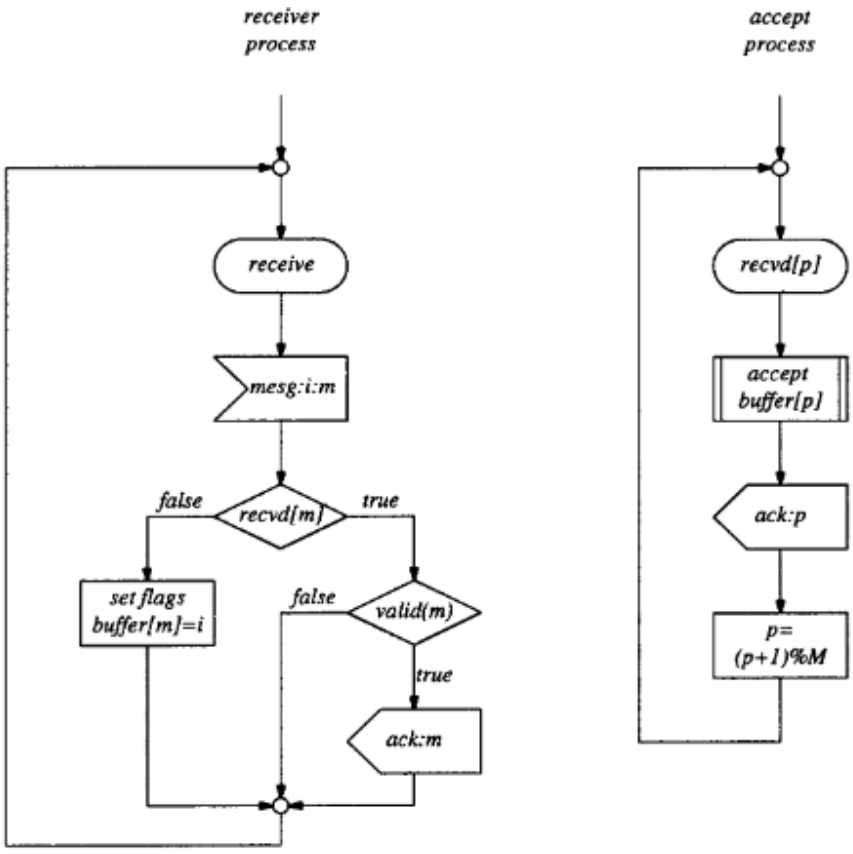


Εικόνα 9-Διεργασίες Αποστολέα σε πρωτόκολλα παραθύρου

Ο πομπός πρώτα εξετάζει πρώτα αν έχει να στείλει πακέτο και αν ναι τότε αυξάνει το πλήθος των πακέτων που έχει στείλει και θέτει στη μεταβλητή busy[s] την τιμή true επειδή έστειλε ένα μήνυμα το οποίο ακόμα δεν έχει γνωστοποιηθεί. Κατόπιν αυξάνει τον αριθμό ακολουθίας των πακέτων που στέλνει και υπολογίζει το υπόλοιπο της διαίρεσης με τη σταθερά M ώστε οι αριθμοί ακολουθίας των αποστελλόμενων πακέτων να βρίσκονται στο εύρος 0..M-1.

Όταν ο πομπός λάβει επιβεβαίωση για το πακέτο με αριθμό m, τότε θέτει στη θέση m του πίνακα busy την τιμή false ως ένδειξη ότι το πακέτο έχει επιβεβαιωθεί.

Αν ο πομπός δεν λάβει επιβεβαίωση για το πακέτο n που έστειλε, τότε το επαναμεταδίδει εξετάζοντας τη θέση n του πίνακα busy και αν τη βρει true (που σημαίνει ότι το πακέτο με αριθμό ακολουθίας n δεν έχει επιβεβαιωθεί) τότε περιμένει για ένα συγκεκριμένο διάστημα (timeout) και το επαναμεταδίδει αποθηκεύοντας στον πίνακα store το μήνυμα με αριθμό ακολουθίας n που εκπέμφθηκε.



Εικόνα 10- Διεργασίες Δέκτη σε πρωτόκολλα παραθύρου

Πριν περιγράψουμε την λειτουργία του δέκτη ας αναφέρουμε μερικά βασικά μεγέθη:

- Recvd[m]: είναι ο αριθμός ακολουθία μηνυμάτων που έχουν ληφθεί αλλά δεν έχουν γίνει ακόμα αποδεκτά

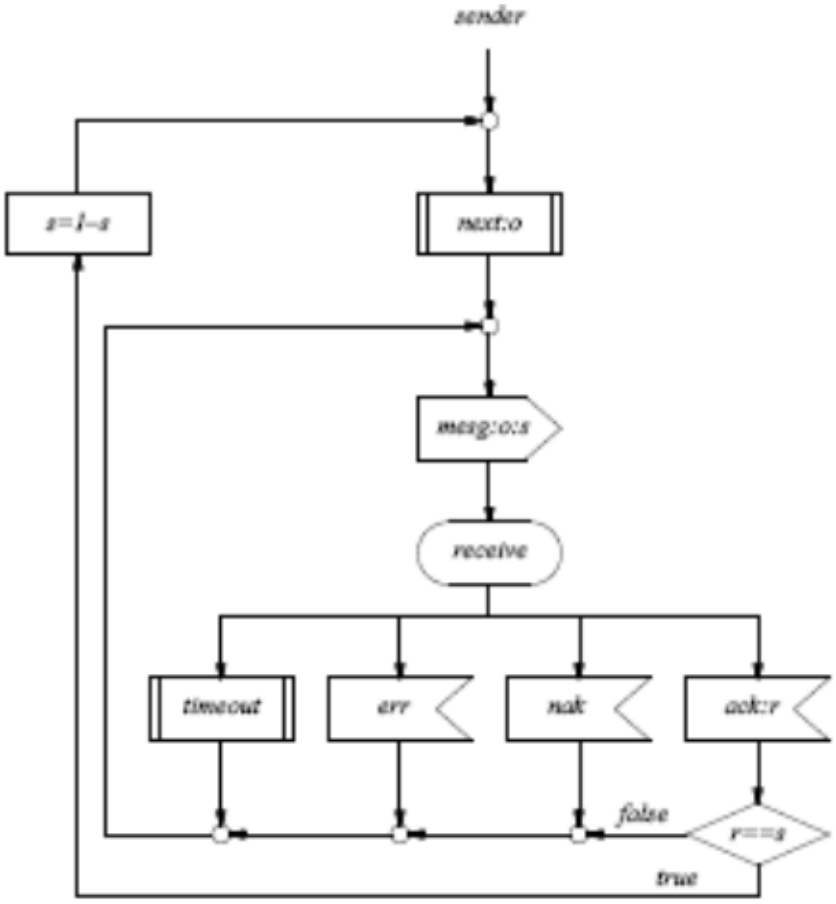
- Buffer[m]: είναι η θέση του πίνακα buffer στην οποία φυλάσσεται το περιεχόμενο του μηνύματος που γίνεται αποδεκτό
- P: είναι ο αριθμός ακολουθίας του επόμενου μηνύματος που πρέπει να γίνει αποδεκτό

Ο δέκτης αρχικά λαμβάνει ένα μήνυμα κα ελέγχει αν το μήνυμα που έλαβε έχει γίνει αποδεκτό ή όχι. Εάν δεν έχει γίνει αποδεκτό τότε το αποδέχεται καταχωρώντας το στον πίνακα buffer στην θέση m. Εάν έχει γίνει ήδη αποδεκτό τότε εξετάζει εάν το μήνυμα που έλαβε είναι έγκυρο και εφόσον είναι στέλνει επιβεβαίωση (ACK) για το μήνυμα αυτό. Αν το μήνυμα δεν είναι έγκυρο, τότε δεν στέλνει καμία επιβεβαίωση στον πομπό.

Η διαδικασία αποδοχής από τον δέκτη περιλαμβάνει αποστολή εκ νέου της επιβεβαίωσης μόνο αν το μήνυμα έχει γίνει αποδεκτό και απλά χάθηκε η αρχική επιβεβαίωση. Αν δεν έχει γίνει αποδεκτό τότε δεν στέλνεται καμία επιβεβαίωση.

5.7 Πρωτόκολλα με Αρνητικές Επιβεβαιώσεις

Στα πρωτόκολλα Stop-And-Wait και Ολισθαίνοντος Παραθύρου (Go-Back-N και Selective-Repeat) υπάρχει η δυνατότητα αποστολής αρνητικής επιβεβαίωσης (Negative Acknowledgement – NAK). Στην επόμενη εικόνα παρατηρούμε τη συμπεριφορά του Αποστολέα στο Πρωτόκολλο Εναλλασσόμενου bit με Timeout.



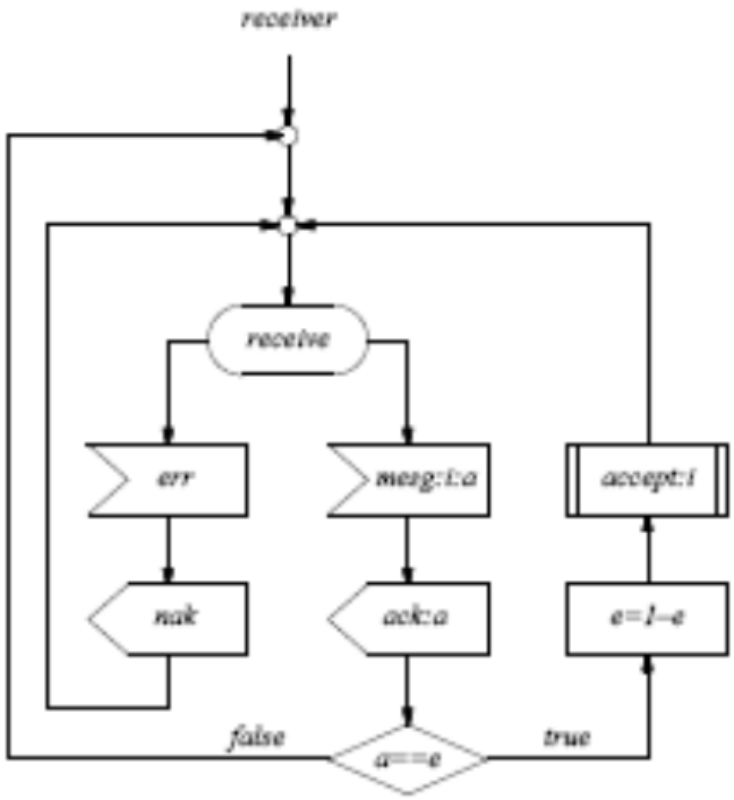
Εικόνα 11-Αποστολέας στο Πρωτόκολλο Εναλλασσόμενου bit με Timeout

Παρατηρούμε ότι ο πομπός κάθε φορά που λαμβάνει ένα μήνυμα από το δέκτη εξετάζει 4 περιπτώσεις:

- Αν το μήνυμα αφορά λήξη του timeout
- Αν το μήνυμα αφορά λάθος μετάδοση πακέτου
- Αν το μήνυμα αφορά λήψη αρνητικής επιβεβαίωσης (NACK) από το δέκτη
- Αν το μήνυμα αφορά λήψη θετικής επιβεβαίωσης (ACK) από το δέκτη

Στις 3 πρώτες περιπτώσεις ο πομπός ξαναστέλνει το ίδιο πακέτο ενώ στην $4^{η}$ περίπτωση που πήρε θετική επιβεβαίωση (ACK) από το δέκτη στέλνει το επόμενο πακέτο εφόσον η επιβεβαίωση αφορά το πακέτο το οποίο στάλθηκε.

Στην επόμενη εικόνα παρατηρούμε τη συμπεριφορά του Δέκτη στο Πρωτόκολλο Εναλλασσόμενου bit με Timeout.



Εικόνα 12- Δέκτης στο Πρωτόκολλο Εναλλασσόμενου bit με Timeout

Επίσης στην Εικόνα αυτή παρατηρούμε αντίστοιχα ότι όταν ο δέκτης λάβει ένα πακέτο εξετάζει αν ο αριθμός του πακέτου που έλαβε είναι ίσος με τον αριθμό πακέτου που αναμένει να λάβει και αν όχι τότε απορρίπτει το ληφθέν πακέτο. Αν ο δέκτης έλαβε το σωστό πακέτο τότε ο αριθμός του επόμενου πακέτου που θα λάβει είναι ο e=1-e. Αν δεν λάβει το σωστό πακέτο τότε το απορρίπτει.

Κεφαλαιο 6: Δομη και προδιαγραφες πρωτοκολλων

Η δομή ενός πρωτοκόλλου είναι σύνολα κανόνων που ορίζουν την αλληλεπίδραση μεταξύ παράλληλων διαδικασιών και αφορούν τις ακόλουθες έννοιες:

- Ανταλλαγή δεδομένων
- Συγχρονισμός διεργασιών
- Ανακάλυψη και διόρθωση λαθών
- Μορφοποίηση και κωδικοποίηση
- Λήξη ανταλλαγής δεδομένων

Μια προδιαγραφή πρωτοκόλλου αποτελείται από πέντε διακριτά τμήματα που περιγράφονται με παραδείγματα στις επόμενες ενότητες.

- Την Υπηρεσία (service) που πρόκειται να υλοποιηθεί
- Τις υποθέσεις (assumptions) για το περιβάλλον λειτουργίας που εφαρμόζεται το πρωτόκολλο
- Το λεξιλόγιο (vocabulary) των μηνυμάτων που χρησιμοποιούνται για την υλοποίηση του πρωτοκόλλου
- Την μορφοποίηση (formatting)
- Τους διαδικαστικούς κανόνες επικοινωνίας αναφορικά με τη συνέπεια

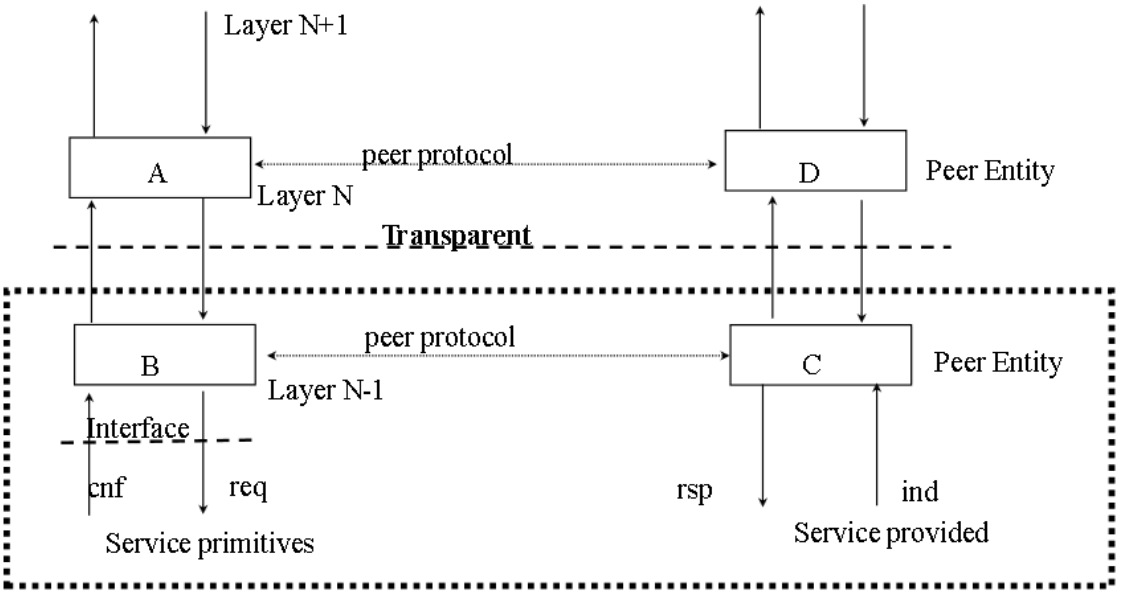
Μερικά από τα χαρακτηριστικά των πρωτοκόλλων είναι τα ακόλουθα:

- **Άμεση/Εμμεση Επικοινωνία**
    - Επικοινωνία σημείο-προς-σημείο (point-to-point link)
        - Δύο οντότητες διαμοιράζονται μια γραμμή (link) επιτρέποντας έτσι απευθείας επικοινωνία
        - Δύο οντότητες μπορούν να επικοινωνήσουν μεταξύ τους μέσω ενός αριθμού ενδιάμεσων κόμβων (hosts). Στην περίπτωση αυτή προκύπτει το πρόβλημα ελέγχου προσπέλασης (access control) κάνοντας το πρωτόκολλο περισσότερο σύνθετο. Πιο συγκεκριμένα πρέπει να οριστούν κανόνες εξυπηρέτησης για το ποια αίτηση θα ικανοποιήσει ο ενδιάμεσος κόμβος όταν δέχεται ταυτόχρονα πολλές αιτήσεις
    - Δίκτυο επικοινωνίας με διακόπτες (switched communcations network)
        - Οι οντότητες βασίζονται σε άλλες οντότητες για ανταλλαγή δεδομένων και δεν κάνουν οι ίδιες την ανταλλαγή αυτή
        - Οι οντότητες μπορεί να συνδέονται στο ίδιο τοπικό δίκτυο (Ethernet) ή μπορούν να ανήκουν σε διαφορετικά δίκτυα (internet)

- Μονολιθικά/Δομημένα Πρωτόκολλα
    - Μονολιθικά Πρωτόκολλα
        - Όλη η λειτουργία του πρωτοκόλλου (για κάθε επίπεδο) συνδυάζεται σε μια ενότητα. Θεωρείται ένας ξεπερασμένος τρόπος κατασκευής πρωτοκόλλων
        - Το βασικό μειονέκτημα αυτής της προσέγγισης είναι ότι είναι δύσκολη οποιαδήποτε αλλαγή στο πρωτόκολλο (όπως π.χ. μια αίτηση για χρήση εικονικού κυκλώματος) γιατί όλες οι λειτουργίες του πρωτοκόλλου είναι συγκεντρωμένες όπως αναφέραμε σε μια και μόνο μονάδα

- Δομημένα Πρωτόκολλα

  - Είναι πρωτόκολλα με ιεραρχική/διαστρωματωμένη δομή. Κάθε επίπεδο εκτελεί συγκεκριμένες λειτουργίες και επικοινωνεί με το αμέσως επόμενο και προηγούμενο του επίπεδο. Το λογισμικό ενός σταθμού εργασίας (είτε του αποστολέα είτε του παραλήπτη σε μια επικοινωνία) χωρίζεται ιεραρχικά σε επίπεδα τα οποία μπορούν να επικοινωνούν και να ανταλλάζουν πληροφορίες τόσο με τα προηγούμενα και τα επόμενα επίπεδα της ιεραρχίας όσο και με αντίστοιχα επίπεδα ιεραρχίας του σταθμού με τον οποίο επικοινωνεί. Η επικοινωνία αυτή φαίνεται στο ακόλουθο σχήμα:



Virtual Channel

  - Σε κάθε επίπεδο (layer) γίνεται ομαδοποίηση παρόμοιων διαδικασιών. Η διασύνδεση (interface) είναι ένα σύνολο από σημεία προσπέλασης υπηρεσίας. Οι υπηρεσίες προσφέρονται στο πιο πάνω επίπεδο και οι υποθέσεις αφορούν τις υπηρεσίες των πιο χαμηλών επιπέδων.

  - Δείχνουν ξεκάθαρα τη διαφορά μεταξύ διαφορετικών επίπέδων

  - Οποιαδήποτε αλλαγή/τροποποίηση στο πρωτόκολλο γίνεται συνήθωσε ένα συγκεκριμένο επίπεδο διευκολύνοντας έτσι τις αλλαγές

- Αρχιτεκτονική Επικοινωνίας
    - Χρησιμοποιείται υλικό και λογισμικό για την υλοποίηση της επικοινωνίας με δομημένα πρωτόκολλα

- Συμμετρικά/Ασύμμετρα Πρωτόκολλα
    - Συμμετρικά Πρωτόκολλα
        - Χρησιμοποιούν Επικοινωνία μεταξύ οντοτήτων σημείου προς σημείο. Όλες οι οντότητες είναι ισότιμες (ομότιμες) και μπορούν να ανταλλάσουν δεδομένα
    - Ασύμμετρα Πρωτόκολλα
        - Χρησιμοποιούν Επικοινωνία τύπου client/server, δηλαδή η επικοινωνία αρχίζει από συγκεκριμένη οντότητα (πελάτης-client) και καταλήγει σε συγκεκριμένη οντότητα (εξυπηρετητής-server)

- Standard/Nonstandard Πρωτόκολλα
    - Standard Πρωτόκολλα
        - Είναι κοινώς αποδεκτά πρωτόκολλα τα οποία έχουν συμφωνηθεί και έχουν τυποποιημένους κανόνες
    - Nonstandard Πρωτόκολλα
        - Τα πρωτόκολλα αυτά κατασκευάζονται για επικοινωνίες ειδικού σκοπού

- Λειτουργίες (Functions)
    - Βάση για όλα τα πρωτόκολλα
    - Ενθυλάκωση
        - Χρησιμοποιούνται Δεδομένα σε συνδυασμό με πληροφορίες ελέγχου σε κάθε μονάδα

6.1 Υπηρεσία προς υλοποίηση (service)

Ένα παράδειγμα υπηρεσίας που μπορεί να υλοποιηθεί από ένα πρωτόκολλο είναι το ακόλουθο:

1. Μεταφορά αρχείων ASCII ως μια σειρά χαρακτήρων
2. Προστασία από λάθη μετάδοσης
3. Αμφίδρομη επικοινωνία
4. Ύπαρξη θετικών και αρνητικών επιβεβαιώσεων (ACK και NACK αντίστοιχα)

6.2 Υποθέσεις για το περιβάλλον λειτουργίας

Ένα παράδειγμα υποθέσεων αναφορικά με το περιβάλλον λειτουργίας είναι το ακόλουθο:

1. Δυο χρήστες ζητούν μεταφορά δεδομένων και περιμένουν να ολοκληρωθεί
2. Το κανάλι επικοινωνίας μπορεί να καταστρέψει μηνύματα αλλά δεν χάνει μηνύματα, δεν δημιουργεί αντίγραφα και δεν αλλάζει τη σειρά τους
3. Υπάρχει αυτόματος μηχανισμός επαναμετάδοσης εσφαλμένων μηνυμάτων ή μηνυμάτων τα οποία χάνονται. Έτσι οι δυο χρήστες είναι σίγουροι ότι λαμβάνουν τα σωστά μηνύματα

6.3 Λεξιλόγιο μηνυμάτων

Υπάρχουν τρείς τύποι μηνυμάτων:

1. ACK : μήνυμα με θετική επιβεβαίωση
2. NACK : μήνυμα με αρνητική επιβεβαίωση
3. ERR : μήνυμα με λάθος μετάδοσης

6.4 Μορφοποίηση μηνυμάτων

Ένα μήνυμα μπορεί να υλοποιηθεί ως μια σύνθετη δομή δεδομένων που τα επιμέρους πεδία της ανήκουν σε σύνθετους ή απλούς τύπους δεδομένων. Ενώ ο τύπος message αναπαριστάνει το μήνυμα και το πεδίο tag περιγράφει τον τύπο του μηνύματος, ενώ το πεδίο data περιγράφει το περιεχόμενο του μηνύματος.

```
enum control {ack, nack, err}

struct message
```

{

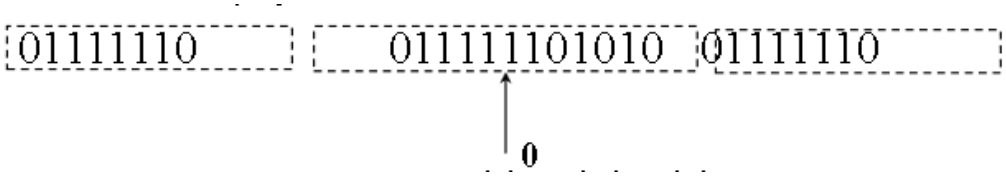
enum control tag;

unsigned char data;

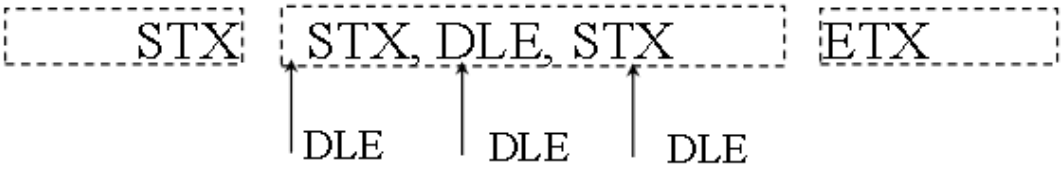
};

Οι μέθοδοι μορφοποίησης πρωτοκόλλων είναι είτε bit oriented (δηλαδή σειρές από bits) είτε character oriented (δηλαδή σειρές από χαρακτήρες)

Το ακόλουθο σχήμα δείχνει ένα παράδειγμα bit oriented πρωτοκόλλου με bit stuffing δηλ. μετά από 5 συνεχόμενους άσσους παρεμβάλλουμε ένα μηδέν προκειμένου να αποφύγουμε τη σημαία αρχής και τη σημαία τέλους 01111110 από τα καθαρά δεδομένα



Το ακόλουθο σχήμα δείχνει ένα παράδειγμα character oriented πρωτοκόλλου



Το πρωτόκολλο αυτό είναι Byte count oriented γιατί μετά το STX (start of text) έχει τον ακριβή αριθμό bytes

Τέλος στην επόμενη εικόνα παρουσιάζουμε ένα παράδειγμα λεξιλογίου και μορφοποίησης.

format = {header, data, trailer}

header = { type, destination, seq no, count}

trailer = {checksum, return address}

6.5 Διαδικαστικοί κανόνες

Οι διαδικαστικοί κανόνες είναι:

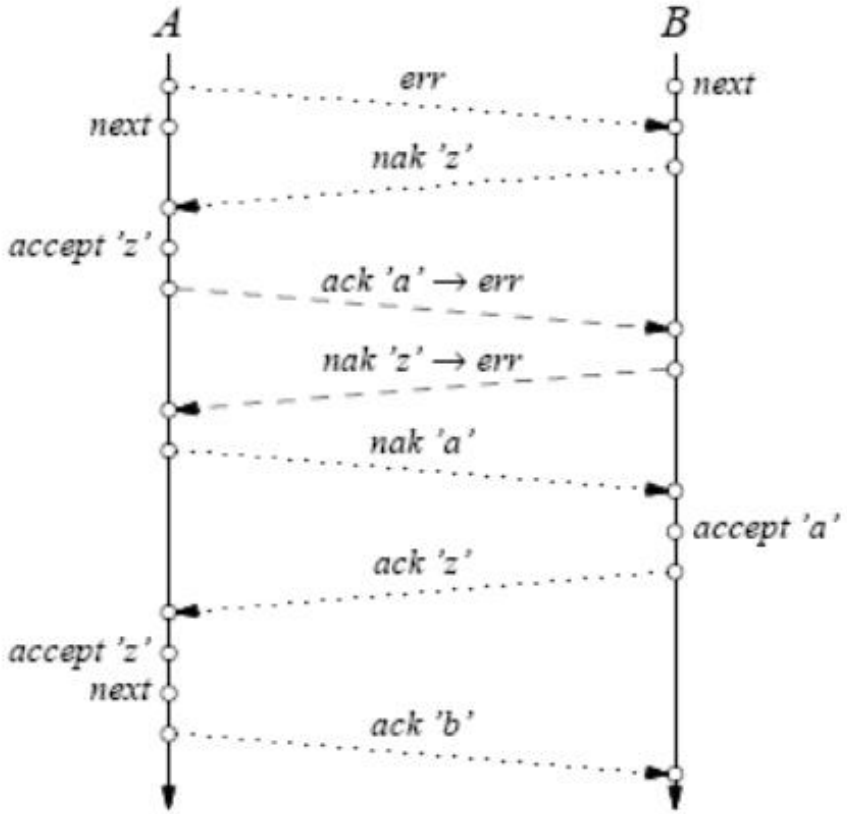
1. Να μην υπάρχει απροσπέλαστος κώδικας
2. Να μην υπάρχει μη ολοκληρωμένος κώδικας
3. Να μην υπάρχουν αδιέξοδα
4. Να μην υπάρχουν λιμοκτονίες
5. Να μην υπάρχουν μη σωστοί τερματισμοί

Ένα παράδειγμα σχεδιαστικών λαθών φαίνεται στο ακόλουθο παράδειγμα:

Υποθέτοντας ότι

1. Μεταφορά προς την μια κατεύθυνση μόνο αν γίνεται και από την άλλη
2. Το πώς αρχίζει και τελειώνει η μεταφορά
3. Μπορούν να γίνουν αποδεκτά αντίγραφα μηνυμάτων

ο αποστολέας (sender) επιχειρεί να στείλει την ακολουθία a-z και ο παραλήπτης (receiver) την ακολουθία z-a όπως φαίνεται στο ακόλουθο σχήμα:



Όπως παρατηρούμε από το σχήμα σε κάθε λάθος μετάδοση από τον Α στον Β ο Β απαντάει με ένα NAK και αντίστοιχα και ο Α απαντάει με ένα NAK όταν λάβει λάθος μήνυμα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία:

- Mohamed G. Gouda (1998), *Elements of Network Protocol Design*, Wiley Interscience
- Τσαουσίδης Βασίλειος (2004), *Διαδικτυακά Πρωτόκολλα*, Κλειδάριθμος
- Sharp Robin (2008), *Principles of Protocol Design*, Springer
- Tanenbaum S. Andrew, Wetherall J. David (2011), *Δίκτυα Υπολογιστών*, Κλειδάριθμος