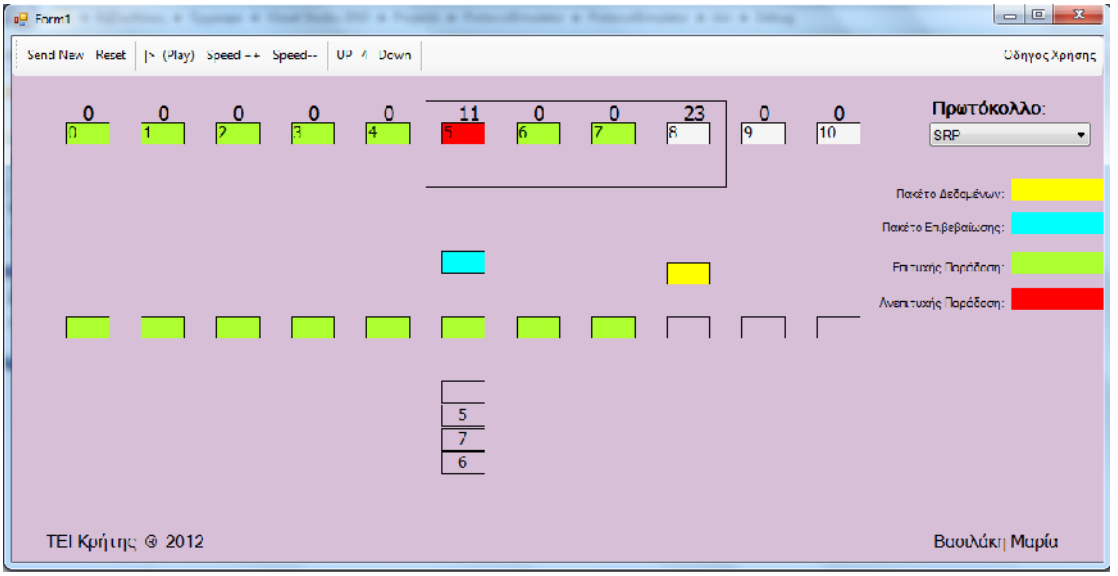
Τ.Ε.Ι. ΚΡΗΤΗΣ / ΠΑΡΑΡΤΗΜΑ ΧΑΝΙΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

Ανάπτυξη κώδικα σε γλώσσα C# για προσομοίωση πρωτοκόλλων επιπέδου ζεύξης.



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βασιλάκη Μαρία

Επιβλέπων : Δρ. Μπαρμπουνάκης Ιωάννης
Καθηγητής Εφαρμογών

Χανιά 2012

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα κ. Μπαρμπουνάκη Ιωάννη, Καθηγητή Εφαρμογών του τμήματος Ηλεκτρονικής του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Κρήτης για τις πολύτιμες συμβουλές και κατευθύνσεις που μου παρείχε για την ολοκλήρωση αυτής της πτυχιακής εργασίας, αλλά και για την άψογη συνεργασία που είχαμε καθ' όλη την διάρκεια των σπουδών μου.

Επίσης, θα ήθελα να ευχαριστήσω θερμά τον φοιτητή Βακάκη Δημήτρη, του τμήματος Εφαρμοσμένης Πληροφορικής και Πολυμέσων του ΤΕΙ Κρήτης για την βοήθειά του στη συγγραφή του κώδικα.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την ηθική αλλά και οικονομική στήριξη που μου παρείχαν.

Περίληψη

Η παρούσα πτυχιακη εργασία επιχειρεί μια προσομοίωση πρωτοκόλλων στο $2^{ο}$ επίπεδο της αρχιτεκτονικής OSI. Συγκεκριμένα προσομοιώνονται τα πρωτόκολλα ABP, GBN και SRP.
Ο κώδικας έχει αναπτυχθεί σε γλώσσα C#.NET στην πλατφόρμα ανάπτυξης Microsoft Visual Studio 2010. Ο φοιτητης έχει την ευκαιρία να κατανοήσει καλύτερα την λειτουργία των πρωτοκόλλων και να αλληλεπιδράσει με την εφαρμογή για να εξάγει χρήσιμα συμπεράσματα.

Abstract

The present study attempts to simulate the second level protocol architecture OSI. It simulated the protocols ABP, GBN and SRP.
The code has been developed in language C #. NET development platform Microsoft Visual Studio 2010. The student has the opportunity to better understand the operation of protocols and interact with the application to draw useful conclusions.

Εισαγωγή

Στο μάθημα Δικτύων Η/Υ σε μια από τις διαλέξεις αναφερθήκαμε στα πρωτόκολλα ζεύξης δεδομένων 2^{ου} επιπέδου OSI τα οποία διασφαλίζουν μια αξιόπιστη επικοινωνία. Αυτά είναι το πρωτόκολλο εναλλασσόμενου bit (alternating bit protocol – ABP), το πρωτόκολλο οπισθοχώρησης N (Go Back N - GBN) και το πρωτόκολλο επιλεκτικής επανάληψης (selective repeat protocol – SRP).

Στο ABP ο αποστολέας στέλνει ένα πακέτο και περιμένει την επιβεβαίωσή του σε χρόνο προθεσμίας, από τον παραλήπτη, για να στείλει το επόμενο πακέτο. Στο GBN ο αποστολέας επιτρέπεται να στέλνει περισσότερα από ένα πακέτα και αν χαθεί κάποιο, ξαναστέλνει αυτό και τα επόμενά του ενώ ο παραλήπτης στέλνει επιβεβαίωση για κάθε πακέτο σε χρόνο προθεσμίας, και απορρίπτει τα πακέτα που λαμβάνει εκτός σειράς. Στο SRP ο αποστολέας στέλνει πολλά πακέτα και αν χαθεί κάποιο, ξαναστέλνει μόνο αυτό, ο παραλήπτης στέλνει επιβεβαίωση σε χρόνο προθεσμίας για κάθε πακέτο και αποθηκεύει σ' έναν καταχωρητή τα πακέτα που λαμβάνει εκτός σειράς. Περισσότερες λεπτομέρειες για την λειτουργία των πρωτοκόλλων αυτών θα δώσουμε στη συνέχεια της εργασίας μας.

Η παρούσα πτυχιακή εργασία αναφέρεται σε ένα προσομοιωτή πρωτοκόλλων επιπέδου ζεύξης σε γλώσσα C#. Η ιδέα για την εκπόνηση αυτής της πτυχιακής εργασίας προήλθε από μια εργασία που μας είχε ανατεθεί στο πλαίσιο του μαθήματος "Δίκτυα Η/Υ" προερχόμενη από την ιστοσελίδα http://www.eecis.udel.edu/~amer/450/TransportApplets/GBN/GBNindex.html [4]

όπου υπήρχε κώδικας java προσομοιωτή, που επέτρεπε την παρουσίαση της λειτουργίας του πρωτοκόλλου GBN. Σε συζήτηση με τον επιβλέποντα καθηγητή, κ. Μπαρμπουνάκη Ιωάννη για το θέμα καταλήξαμε να επεκτείνουμε την εργασία και να ξεπεράσουμε τα σφάλματα που εμφάνιζε ο συγκεκριμένος προσομοιωτής. Ο κώδικας που τροποποιήσαμε βρίσκεται στην ιστοσελίδα http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/go-back-/go-back-n.html [5]

Αρχικά, εκτελέσαμε τον κώδικα (java) στην πλατφόρμα ανάπτυξης Netbeans με δυνατότητα καθορισμού του μεγέθους του παραθύρου από σταθερή τιμή 5 σε μεταβλητή τιμή εύρους {1,..,7} ανάλογα με τις επιλογές του χρήστη. Επίσης απλοποιήσαμε την διαδικασία εισαγωγής σφαλμάτων σε ένα πακέτο δεδομένων ή την επιβεβαίωσή του κάνοντας απλά 'κλικ' πάνω στο αντίστοιχο κελί με προσωρινή παύση της προσομοίωσης.

Στη συνέχεια της πτυχιακής μου, επέλεξα να υλοποιήσω τον προσομοιωτή των πρωτοκόλλων επιπέδου ζεύξης σε γλώσσα C#. Η επιλογή μου βασίστηκε αφ' ενός στο γεγονός ότι στο πρόγραμμα σπουδών υπάρχει μάθημα σε γλώσσα C ενώ δεν υπάρχει για java, αφ' ετέρου στην δυσκολία τροποποίησης κώδικα που δεν έχει οργάνωση και πλήρη τεκμηρίωση. Η συγγραφή του κώδικα έγινε σε γλώσσα C# στην πλατφόρμα ανάπτυξης Microsoft Visual Studio 2010 και εκτός από το πρωτόκολλο GBN, υλοποιήσαμε και τα άλλα δύο πρωτόκολλα το ABP και το SRP.

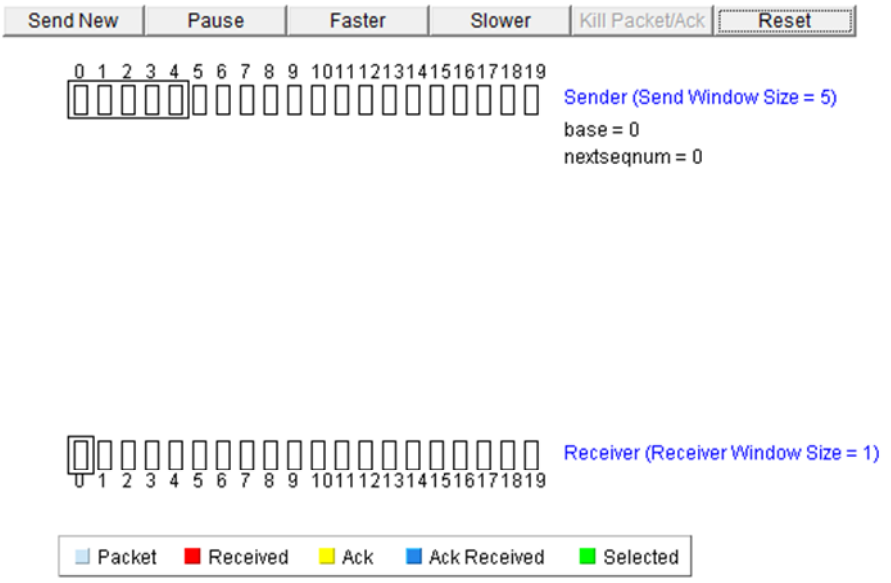
Η παρούσα πτυχιακή εργασία διαιρείται σε 6 κεφάλαια. Στο κεφάλαιο 1 γίνεται αναφορά στο πρόγραμμα προσομοίωσης της java για το πώς λειτουργεί το πρωτόκολλο GBN . Στο κεφάλαιο 2 δίνεται μια σύντομη περιγραφή της γλώσσας προγραμματισμού C#, της πλατφόρμας ανάπτυξης εφαρμογών .NET και της πλατφόρμας ανάπτυξης Microsoft Visual Studio 2010. Το 3^{ο} κεφάλαιο αναφέρεται στην προσομοίωση της πρακτικής της C# και στα χαρακτηριστικά της καθώς και στις κλάσεις PacketN, Form1 που έχουν αναπτυχθεί στο πλαίσιο της πτυχιακής. Το κεφάλαιο 4 εξηγεί το πρωτόκολλο εναλλασσομένου Bit (Alternating Bit Protocol ABP), το πρωτόκολλο οπισθοχώρησης κατά N (Go back N) και το πρωτόκολλο επιλεκτικής επανάληψης (Selective Repeat Protocol SRP) κάνοντας αναφορά στον αντίστοιχο κώδικα υλοποίησης του προσομοιωτή. Το 5^{ο} κεφάλαιο περιέχει τα συμπεράσματα και το 6^{ο} περιέχει την βιβλιογραφία με τα βιβλία και τους ιστότοπους που χρησιμοποιήθηκαν για την εκπόνηση της παρούσας πτυχιακής εργασίας.

Περιεχόμενα

Κεφάλαιο 1

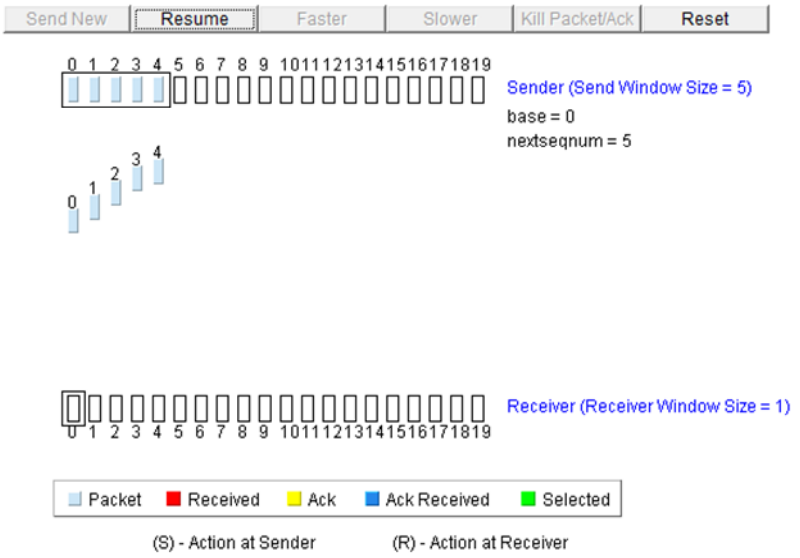
Πρόγραμμα προσομοίωσης πρωτοκόλλων java

Στην ιστοσελίδα του πανεπιστημίου Delaware http://www.eecis.udel.edu/~amer/450/TransportApplets/GBN/GBNindex.html [4] υπάρχει μια αρκετά καλή έκδοση ενός κώδικα που παρουσιάζει τις λειτουργίες του πρωτοκόλλου Go-Back-N με την βοήθεια ενός φυλλομετρητή. (Εικόνα 1.1)



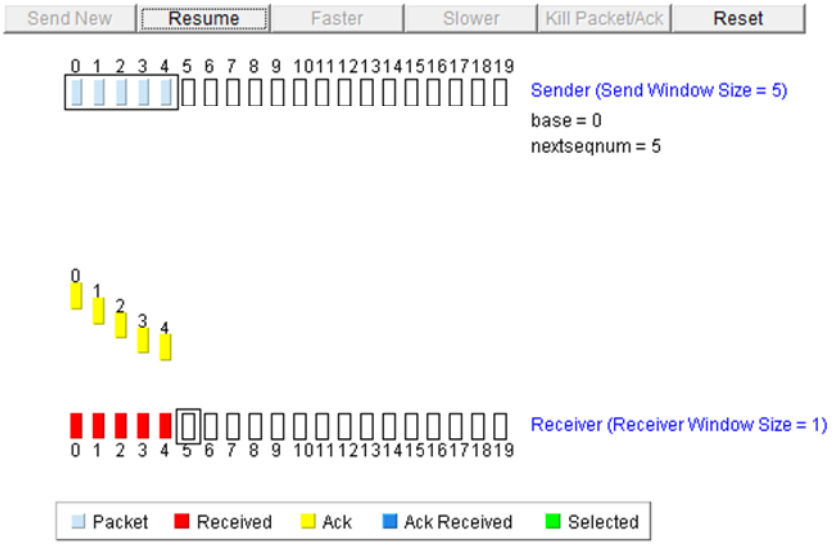
Εικόνα 1.1: Προσομοιωτής πρωτοκόλλου Go-Back-N

Ουσιαστικά πρόκειται για ένα java applet που ορίζει για τον αποστολέα ένα σταθερό παράθυρο 5 ανεπιβεβαίωτων πακέτων. Επιπλέον, δίνεται η δυνατότητα στο χρήστη πατώντας το ειδικό πλήκτρο "Send New" κάθε φορά να εκκινεί την αποστολή πακέτων προς τον παραλήπτη. Τα πλαίσια μπλε ανοικτής απόχρωσης που απεικονίζουν τα πακέτα δεδομένων, φέρουν αρίθμηση π.χ. {0,…,4} (Εικόνα 1.2)



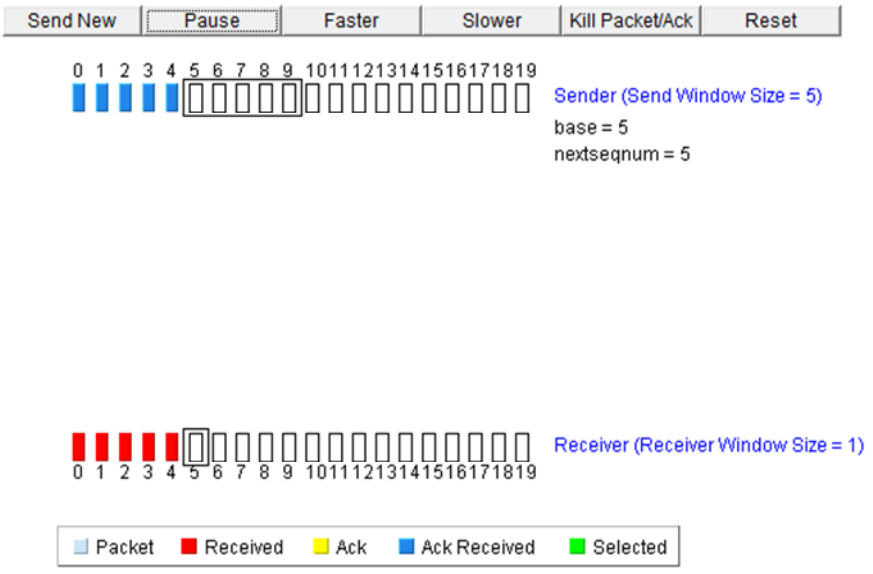
Εικόνα 1.2: Λειτουργία "Send New"

Όταν δεν υπάρχουν σφάλματα τα πακέτα φτάνουν στον παραλήπτη με τη σειρά που εστάλησαν. Αυτός τα δέχεται χρωματίζοντας τα συγκεκριμένα πλαίσια λήψης των πακέτων με κόκκινο χρώμα, ένδειξη ότι τα πακέτα φτάσανε σωστά δηλαδή χωρίς σφάλματα. Στη συνέχεια αφού ο παραλήπτης λάβει τα πακέτα, στέλνει πίσω στον αποστολέα τις επιβεβαιώσεις τους με κίτρινο χρώμα και αριθμεί τα αντίστοιχα πλαίσια λήψης με τους αριθμούς 0,1,2,3,4 αντίστοιχα (Εικόνα 1.3).



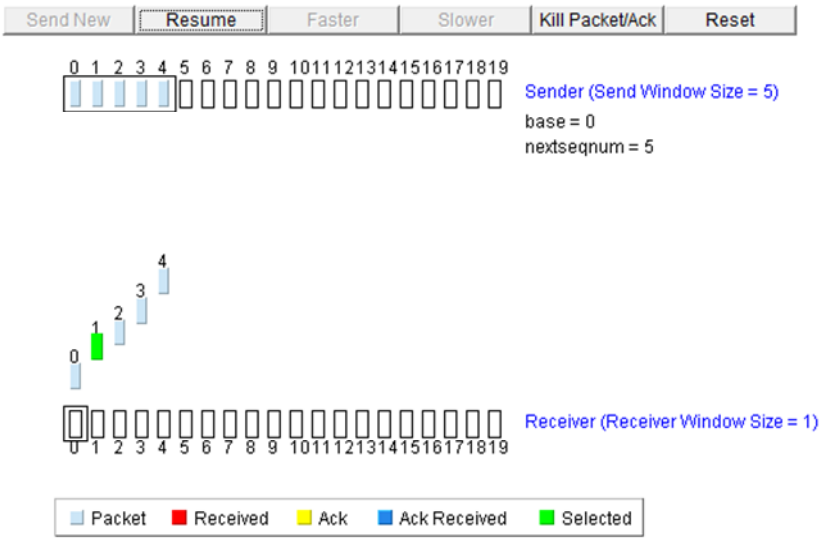
Εικόνα 1.3: Λειτουργία παράδοσης και επιβεβαίωσης

Όταν ο αποστολέας λαμβάνει τις επιβεβαιώσεις των αντίστοιχων πακέτων χωρίς σφάλματα, χρωματίζει τα αντίστοιχα πλαίσια με μπλε σκούρο χρώμα. Το παράθυρο που περιβάλλει τα ανεπιβεβαίωτα πακέτα ολισθαίνει κατά μια θέση δεξιότερα κάθε φορά που ο αποστολέας λαμβάνει μια επιβεβαίωση. Έτσι δίνεται η δυνατότητα στον αποστολέα να αποστείλει τα αμέσως επόμενα πακέτα με αρίθμηση 5,6,7,8 κτλ. (Εικόνα 1.4)



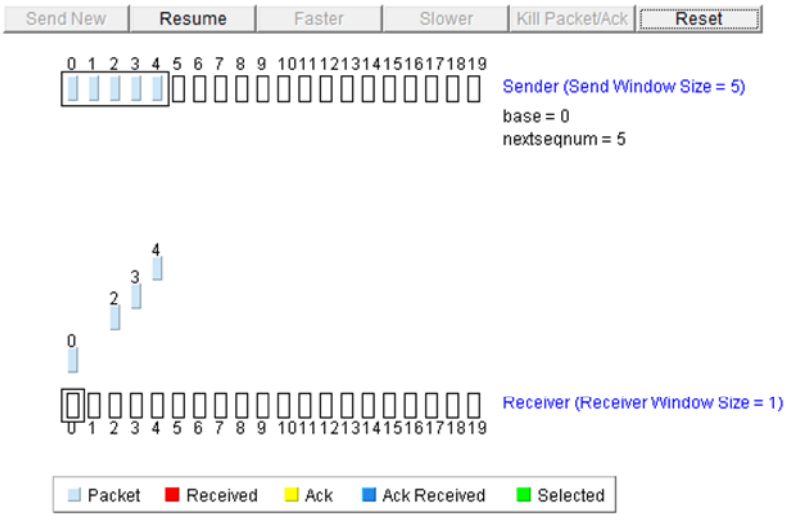
Εικόνα 1.4: Λειτουργία λήψης επιβεβαιώσεων και ολίσθηση παραθύρου

Στο συγκεκριμένο java applet εκτός των άλλων, δίνεται η δυνατότητα στο χρήστη μέσω του ειδικού πλήκτρου "Pause" να παγώνει τη λειτουργία του πρωτοκόλλου έτσι ώστε να μπορεί να επιλέγει ένα πακέτο (ή την επιβεβαίωσή του) κάνοντας ‘κλικ’ πάνω του με τη βοήθεια του ποντικιού. Το πακέτο με αυτή την κίνηση γίνεται πράσινο. Στην συνέχεια πατώντας το ειδικό πλήκτρο "kill" το επιλεγμένο πράσινο πακέτο εξαφανίζεται για να προσομοιώσει απώλεια λόγω σφαλμάτων. Για παράδειγμα στέλνονται τα πρώτα ανεπιβεβαίωτα πακέτα πατώντας το πλήκτρο "Send New" 5 φορές. Έτσι, ο αποστολέας στέλνει 5 ανεπιβεβαίωτα πακέτα, τα οποία πριν φτάσουν στον παραλήπτη, μπορούν να επιλεγούν για να «σκοτωθούν» εφόσον προηγηθεί η ενεργοποίηση του πλήκτρου "Pause" (Εικόνα 1.5)



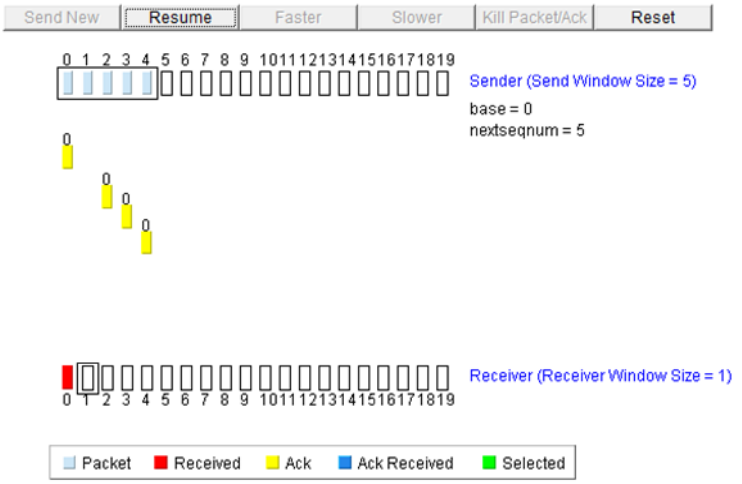
Εικόνα 1.5: Επιλογή πακέτου "1"

Στην συνέχεια πατώντας το πλήκτρο "kill" το επιλεγμένο πράσινο πακέτο με τον αριθμό "1" χάνεται, υποτίθεται λόγω σφαλμάτων. (Εικόνα 1.6)



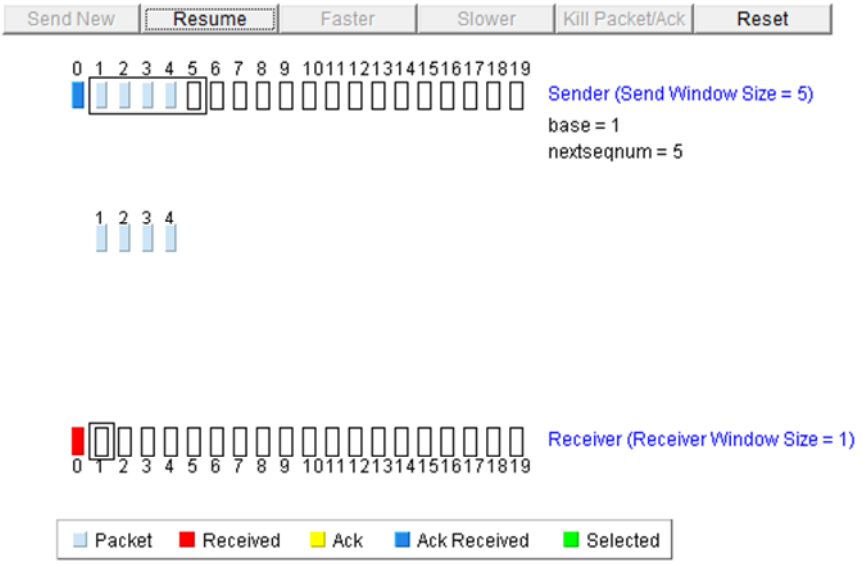
Εικόνα 1.6: Λειτουργία πλήκτρου "kill" στο πακέτο "1"

Για να επαναφέρουμε τη λειτουργία του προσομοιωτή πρωτοκόλλου, πατάμε το ειδικό πλήκτρο "Resume". Ο παραλήπτης δέχεται το πακέτο με τον αριθμό "0" χρωματίζοντας το αντίστοιχο πλαίσιο κόκκινο και στέλνει στον αποστολέα την επιβεβαίωσή του, δεν λαμβάνει το πακέτο με τον αριθμό "1" και δεν χρωματίζει το αντίστοιχο πλαίσιο του για να δείξει ότι υπάρχει σφάλμα, επίσης δεν αποδέχεται τα πακέτα 2, 3 και 4 γιατί είναι εκτός σειράς, δεν χρωματίζει τα αντίστοιχα πλαίσια τους και στέλνει στον αποστολέα επιβεβαιώσεις του πακέτου "0" (Εικόνα 1.7)



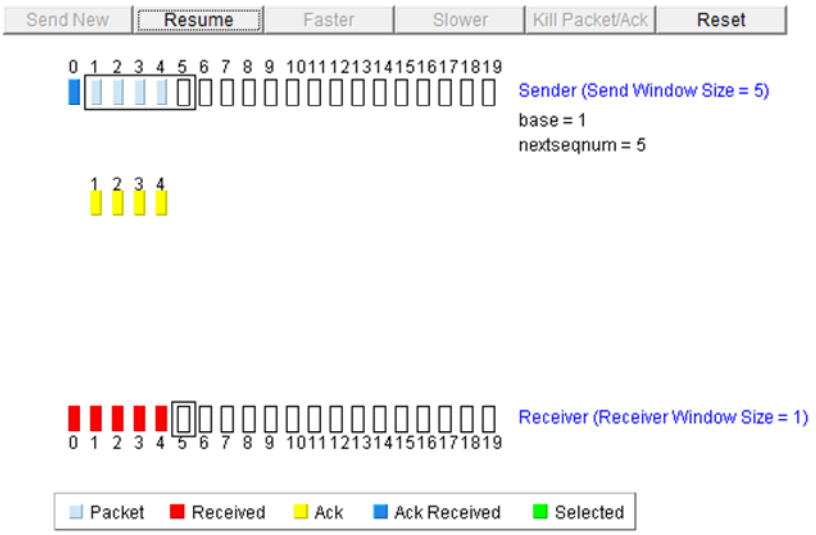
Εικόνα 1.7: Απόρριψη πακέτων εκτός σειράς στον παραλήπτη

Μετά από ένα προκαθορισμένο χρόνο προθεσμίας ο αποστολέας ξαναστέλνει το χαμένο πακέτο και τα επόμενά του, πιο συγκεκριμένα στο παράδειγμά μας ο αποστολέας ξαναστέλνει τα ανεπιβεβαίωτα πακέτα 1, 2, 3 και 4 με μπλε ανοικτό χρώμα στον παραλήπτη. Τα πακέτα ξαναστέλνονται στην ίδια ευθεία σαν να έχουν εκπεμφθεί ταυτόχρονα κάτι που είναι λάθος από τεχνικής πλευράς (Εικόνα 1.8).



Εικόνα 1.8: Επανεκπομπή πακέτων

Ο παραλήπτης δέχεται τα πακέτα εφόσον έχουν φτάσει σωστά, χρωματίζει τα αντίστοιχα πλαίσια τους με κόκκινο χρώμα και στέλνει πίσω τις επιβεβαιώσεις τους με κίτρινο χρώμα αριθμημένες 1,2,3,4 (Εικόνα 1.9)



Εικόνα 1.9: Αποστολή επιβεβαιώσεων

Αν ο χρήστης πατήσει το πλήκτρο "Send New" , ο αποστολέας μπορεί να στείλει το πακέτο με το αριθμό "5" αφού έχει λάβει την επιβεβαίωση του πακέτου "0" και το παράθυρο που περιβάλλει ανεπιβεβαίωτα πακέτα έχει ολισθήσει δεξιότερα γιατί τα ανεπιβεβαίωτα πακέτα είναι 5.

Όπως, ήδη έχουμε αναφέρει, δίνεται η δυνατότητα στο χρήστη μέσω ειδικού πλήκτρου "Pause" να παγώνει τη λειτουργία του πρωτοκόλλου έτσι ώστε να μπορεί να επιλέγει ένα πακέτο ή την επιβεβαίωσή του. Αν επιλέξει την επιβεβαίωση ενός πακέτου τότε το πακέτο της επιβεβαίωσης θα γίνει πράσινο χρώμα και πατώντας το πλήκτρο "kill", το πακέτο επιβεβαίωσης θα χαθεί λόγω σφαλμάτων. Ο αποστολέας ως αποτέλεσμα πρέπει να ξαναστείλει το χαμένο πακέτο και τα επόμενά του αλλά στη συγκεκριμένη εφαρμογή δεν το κάνει και συνεχίζει τη λειτουργία κάτι που από τεχνικής σκοπιάς είναι λάθος.

Κεφάλαιο 2

C#.NET

2.1 Τι είναι η C# [2]

Η C# είναι μια σχετικά νέα αντικειμενοστραφής γλώσσα προγραμματισμού η οποία δημιουργήθηκε από την Microsoft. Δανείζεται πολλά στοιχεία, και έχει παρόμοια σύνταξη, με την C++ και την Java, κάνοντας την εκμάθησή της σχετικά εύκολη. Κατά την εκτέλεση της εφαρμογής, το .NET, το περιβάλλον δηλαδή μέσα στο οποίο θα τρέξει το πρόγραμμα, διαβάζει τον ενδιάμεσο κώδικα (MSIL) και τον μεταγλωττίζει σε κώδικα Windows. Είναι γλώσσα ειδικά σχεδιασμένη για να υποστηρίζει το .NET framework της ίδιας εταιρείας. Βασικό χαρακτηριστικό της είναι ότι δεν παράγει απευθείας κώδικα μηχανής όπως η C++, άλλα ένα ενδιάμεσο κώδικα που στοχεύει το .NET. Επίσης κάθε εντολή στην C#, όπως και στην C++ πρέπει να τερματίζεται με το ελληνικό ερωτηματικό. Η C# (προφέρεται σι σαρπ) είναι η γλώσσα επιλογής για ανάπτυξη .NET εφαρμογών. Εν αντιθέσει με την Visual Basic που προσαρμόστηκε για να υποστηρίζει το .NET, η C# έχει σχεδιαστεί εξαρχής με γνώμονα την πλατφόρμα αυτή και συνεπώς μπορεί να την αξιοποιήσει καλύτερα.

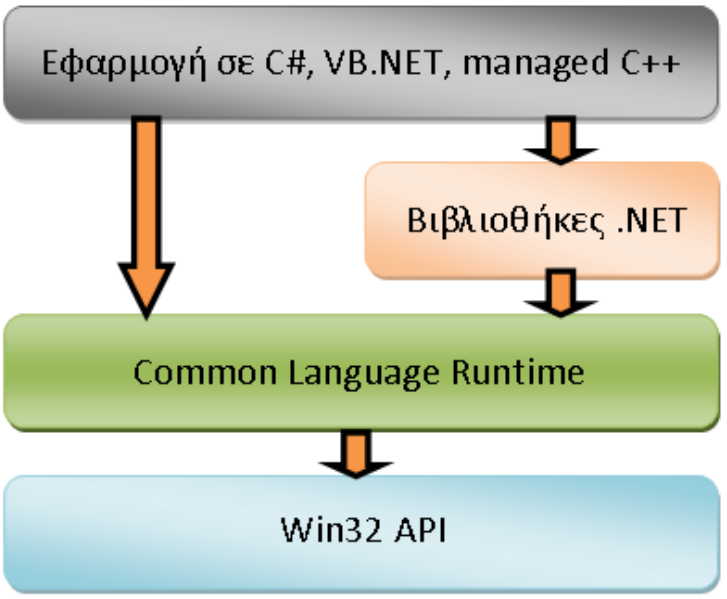
2.2 Τι είναι το .NET [2]

Το .NET είναι μια νέα πλατφόρμα ανάπτυξης εφαρμογών σε περιβάλλοντα Windows. Σύμφωνα με την Microsoft, έχει ως σκοπό την απλοποίηση της ανάπτυξης εφαρμογών «κρύβοντας» τις τεχνικές λεπτομέρειες υλοποίησης πολλών λειτουργιών, όπως διαχείριση μνήμης, επικοινωνία μέσω δικτύου, είσοδο/έξοδο από συσκευές και αφήνοντας το προγραμματιστή ελεύθερο να επικεντρωθεί στην ανάπτυξη της εφαρμογής του.

Το .NET χαρακτηρίζεται ως διαχειρίσιμη πλατφόρμα με την έννοια ότι δημιουργεί ένα ελεγχόμενο και ασφαλές περιβάλλον μέσα στο οποίο μπορεί να εκτελεί μια εφαρμογή. Η ασφάλεια έγκειται για παράδειγμα στον έλεγχο στην δέσμευση και προσπέλαση της μνήμης (δεν υπάρχουν pointers, δεν μπορείς να προσπελάσεις μια θέση μνήμης εκτός πίνακα), στο τύπο των μεταβλητών και δεδομένων (δεν μπορείς να θέσεις μια float τιμή σε μια ακέραια μεταβλητή) ή στην αυτόματη υλοποίηση δικλείδων ασφαλείας.

Το .NET υποστηρίζει πληθώρα γλωσσών προγραμματισμού οι οποίες είναι ειδικά σχεδιασμένες για αυτό, όπως C#, Visual Basic.NET, J++ και managed C++. Στην πραγματικότητα, το .NET καταλαβαίνει μόνο μια γλώσσα προγραμματισμού την Microsoft Intermediate Language (MSIL). Συνεπώς, οποιαδήποτε γλώσσα προγραμματισμού μπορεί να μεταγλωττιστεί σε MSIL μπορεί να τρέξει στην πλατφόρμα .NET. Ο χρήστης μπορεί ακόμα να γράψει απευθείας ένα πρόγραμμα σε MSIL στο Notepad να το κάνει compile και να το τρέξει στο .NET.

Καρδιά του .NET αποτελεί το λεγόμενο Common Language Runtime (CLR). Η οντότητα αυτή είναι το διαχειρίσιμο περιβάλλον μέσα στο οποίο τρέχουν οι εφαρμογές .NET. Κατά μια έννοια κρύβει το δύσχρηστο Win32 API που χρησιμοποιείται συχνά για προγραμματισμό εφαρμογών Windows και παρουσιάζει στο χρήστη ένα απλούστερο και περισσότερο εύχρηστο Win32 API. (Σχήμα 1)



Σχήμα 1: .NET

Επίσης το .NET παρέχει και μια πληθώρα βιβλιοθηκών με έτοιμες λειτουργίες που μπορεί να χρησιμοποιήσει ο χρήστης για την ανάπτυξη των εφαρμογών.

Το .NET, όπως και η Java χαρακτηρίζονται από την λεγόμενη Just in Time μεταγλώττιση. Ο κώδικας, στη γλώσσα προγραμματισμού που χρησιμοποιεί ο χρήστης, μεταγλωττίζεται αρχικά σε MSIL η οποία αποθηκεύεται σε ένα εκτελέσιμο .exe αρχείο. Όταν ο χρήστης τρέξει το πρόγραμμα που ανέπτυξε, το Common Language Runtime (CLR) διαβάζει τον MSIL κώδικα του αρχείου και Just In Time (JIT) το μεταγλωττίζει σε κώδικα Windows (native) έτοιμο προς εκτέλεση, και στην

συνέχεια τον εκτελεί. Αυτό το επιπλέον βήμα πριν την εκτέλεση του κώδικα διαφοροποιεί μια διαχειρίσιμη εφαρμογή σε .NET από μια μη διαχειρίσιμη σε C++ για παράδειγμα. Το πρόγραμμα σε C++ είναι ήδη μεταγλωττισμένο σε native Windows κώδικα και τρέχει απευθείας.

Ένα ακόμα χαρακτηριστικό του CLR είναι η αυτοματοποιημένη διαχείριση μνήμης. Σε κλασσικές γλώσσες προγραμματισμού (unmanaged) όπως η C++, όταν ο χρήστης δεσμεύσει μια περιοχή μνήμης για να αποθηκεύσει ένα αντικείμενο πρέπει να είναι πολύ προσεκτικός στο να την αποδεσμεύσει, να την επιστρέψει στο σύστημα δηλαδή, όταν δεν την χρειάζεται. Αν το αγνοήσει αυτό συστηματικά, τότε θα δημιουργηθεί το επονομαζόμενο memory leak, δηλαδή η διαθέσιμη μνήμη του συστήματος θα ελαττώνεται διαρκώς και σε κάποιο σημείο τα Windows θα στερέψουν από ελεύθερη μνήμη.

Αντιθέτως το CLR προσφέρει ένα μηχανισμό Garbage Collection. Ο χρήστης μπορεί να ζητήσει όση μνήμη χρειάζεται από το σύστημα και να μην ασχοληθεί με την απελευθέρωσή της. Ο Garbage Collector υλοποιεί μηχανισμούς που του επιτρέπουν να «καταλάβει» πότε μια δεσμευμένη ποσότητα μνήμης δεν χρησιμοποιείται πλέον και αυτόματα την απελευθερώνει για μετέπειτα χρήση.

2.3 Πλεονεκτήματα και Μειονεκτήματα .NET [2]

Το .NET έχει πολλά πλεονεκτήματα για την ανάπτυξη εφαρμογών:

- Είναι εγγενώς αντικειμενοστραφής πλατφόρμα.
- Είναι ανεξάρτητο από γλώσσα προγραμματισμού. Σε μια εφαρμογή ένας προγραμματιστής μπορεί να γράφει κώδικα σε C#, άλλος σε VB.NET και άλλος σε managed C++ και τα τμήματα που αναπτύσσει ο καθένας να συνεργάζονται μεταξύ τους χωρίς προβλήματα.
- Κάνει με τη χρήση βιβλιοθηκών (assemblies) πολύ εύκολη την επαναχρησιμοποίηση κώδικα.
- Παρέχει πολύ εύκολη εγκατάσταση. Αρκεί να αντιγράψουμε το κατάλογο της εφαρμογής σε ένα άλλο υπολογιστή και αυτή θα τρέξει άμεσα.
- Παρέχει πληθώρα έτοιμων λειτουργιών που κάνουν την ανάπτυξη κώδικα πολύ εύκολη.
- Έχει αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

Το .NET έχει 2 μειονεκτήματα που αφορούν ειδικά την ανάπτυξη βιντεοπαιχνιδιών και όχι την γενική ανάπτυξη εφαρμογών:

- Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.
- Το CLR εισάγει μια (μικρή) καθυστέρηση στην εκτέλεση της εφαρμογής.

Το πρώτο, αν και είναι πολύ χρήσιμο χαρακτηριστικό για γενικό προγραμματισμό, σε ένα βιντεοπαιχνίδι μπορεί να δημιουργήσει πρόβλημα. Η απελευθέρωση μνήμης που δεν χρησιμοποιείται πλέον είναι μια ακριβή εργασία (από πλευράς χρόνου), η οποία επιπλέον είναι μη-ντετερμινιστική, μπορεί να συμβεί δηλαδή οποτεδήποτε. Αυτό μπορεί να έχει σαν αποτέλεσμα ένα παιχνίδι που τρέχει σε ένα σταθερό ρυθμό 60 καρέ το δευτερόλεπτο, να δει μια δραματική πτώση στο ρυθμό ανανέωσης για 1 δευτερόλεπτο, πράγμα ανεπίτρεπτο στην ανάπτυξη βιντεοπαιχνιδιών.

Το δεύτερο αφορά τη Just in Time μεταγλώττιση που υποστηρίζει το CLR. Από τη μία εισάγει μια καθυστέρηση στην εκκίνηση του παιχνιδιού μιας και πρέπει να μεταγλωττιστεί ο κώδικας από την άλλη ο μεταγλωττιστής ο ίδιος δεν είναι βέλτιστος με την έννοια ότι δεν παράγει το καλύτερο δυνατό native κώδικα για Windows.

Και τα δυο προβλήματα αυτά μπορούν να αντιμετωπιστούν αν τα λάβουμε υπ’ όψη κατά την ανάπτυξη της εφαρμογής.

2.4 Visual Studio 2010 [3]

Το Microsoft Visual Studio 2010 είναι ένα ενοποιημένο περιβάλλον που απλοποιεί ολόκληρο τον κύκλο προγραμματισμού, από τη σχεδίαση έως την ανάπτυξη. Είναι ένα ενοποιημένο περιβάλλον, στο οποίο οι προγραμματιστές μπορούν να χρησιμοποιούν τις ικανότητές τους, για να δημιουργούν μοντέλα ενός ολοένα μεγαλύτερου αριθμού εφαρμογών, να συγγράφουν κώδικα για αυτές, να εντοπίζουν τα σφάλματά τους, να τις δοκιμάζουν και να τις αναπτύσσουν. Το Visual Studio 2010 απλοποιεί συνήθεις εργασίες και βοηθά τους προγραμματιστές να εξερευνήσουν την πλατφόρμα εις βάθος. Παρέχει ισχυρά εργαλεία για τη διαχείριση των έργων, τη συντήρηση του κώδικα προέλευσης και τον εντοπισμό των σφαλμάτων. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν αυτοματοποιημένες και μη δοκιμές, καθώς και σύνθετα εργαλεία εντοπισμού σφαλμάτων, για να διασφαλίσουν ότι δημιουργούν τη σωστή εφαρμογή με τον σωστό τρόπο.

Κεφάλαιο 3
Πρωτόκολλα ζεύξης δεδομένων

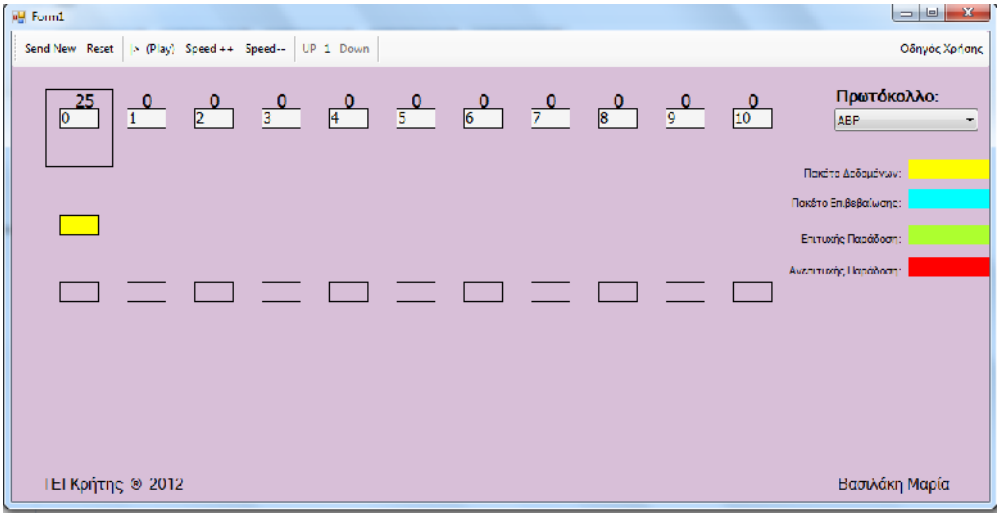
3.1 Προσομοίωση πακέτων δεδομένων και διαδικασιών.

Τα πακέτα του αποστολέα αναπαριστούνται από κάποια σταθερά πλαίσια σε μια οριζόντια γραμμή στο πάνω άκρο της εφαρμογής και πλαισιώνονται από ένα παράθυρο, το παράθυρο αυτό μας δείχνει πόσα πακέτα επιτρέπεται να στέλνουμε ανεπιβεβαίωτα. Με την ενεργοποίηση του πλήκτρου "Send New", ξεκινάει ένα πακέτο δεδομένων από την αφετηρία "ως πακέτο κίτρινης απόχρωσης" και κατευθύνεται προς τον παραλήπτη, ο οποίος αναπαριστάται επίσης από μια σειρά με σταθερά πλαίσια στο κάτω μέρος της εφαρμογής μας. Όταν το πακέτο δεδομένων φτάσει στον παραλήπτη τότε το συγκεκριμένο πλαίσιο χρωματίζεται πράσινο και ξεκινάει το πακέτο επιβεβαίωσης "ως πακέτο μπλε απόχρωσης" και κατευθύνεται προς τον αποστολέα. Όταν το πακέτο επιβεβαίωσης φτάσει στον αποστολέα τότε το συγκεκριμένο πλαίσιο χρωματίζεται πράσινο και το παράθυρό που πλαισιώνει τα πακέτα ολισθαίνει μια θέση δεξιότερα.

Αν ένα πακέτο χαθεί κατά τη μετάδοσή του και δεν το λάβει σωστά ο παραλήπτης, τότε το πλαίσιο του παραλήπτη γίνεται κόκκινης απόχρωσης. Προκειμένου να προσομοιώσουμε την απώλεια πακέτων, έχουμε υλοποιήσει την δυνατότητα ο χρήστης να μην επιλέγει απαραίτητα το ίδιο το πακέτο δεδομένων ή το πακέτο επιβεβαίωσης αλλά να κάνει διπλό κλικ στην στήλη μεταξύ αποστολέα και παραλήπτη. Κάνοντας διπλό κλικ την ώρα που στέλνεται το πακέτο, προσομοιώνουμε την περίπτωση εισαγωγής σφαλμάτων στο πακέτο δεδομένων. Έτσι, ο αποστολέας ξαναστέλνει το πακέτο μόλις μηδενίσει ο χρόνος προθεσμίας (timer). Για κάθε χαμένο πακέτο, ο παραλήπτης γίνεται κόκκινης απόχρωσης στο μισό χρόνο προθεσμίας (timer).

3.2 Πρωτόκολλο εναλλασσομένου Bit (Alternating Bit Protocol ABP)

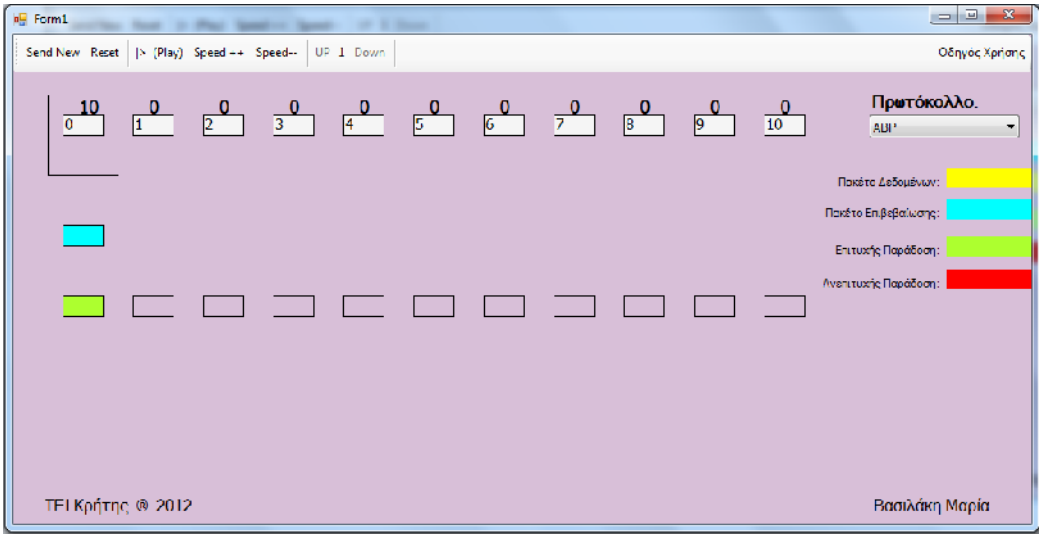
Στο πρωτόκολλο ABP στέλνεται 1 πακέτο κάθε φορά και πρέπει να έρθει η επιβεβαίωση για να σταλεί το επόμενο. Επιλέγοντας το πρωτόκολλο ABP από το σχετικό μενού επιλογής, το παράθυρο που πλαισιώνει τα ανεπιβεβαίωτα πακέτα περιορίζεται σε ένα μόνο πακέτο. Στο σχήμα 3.1 βλέπουμε τη μετάδοση ενός ανεπιβεβαίωτου πακέτου.



Σχήμα 3.1: Μετάδοση ανεπιβεβαίωτου πακέτου (ABP).

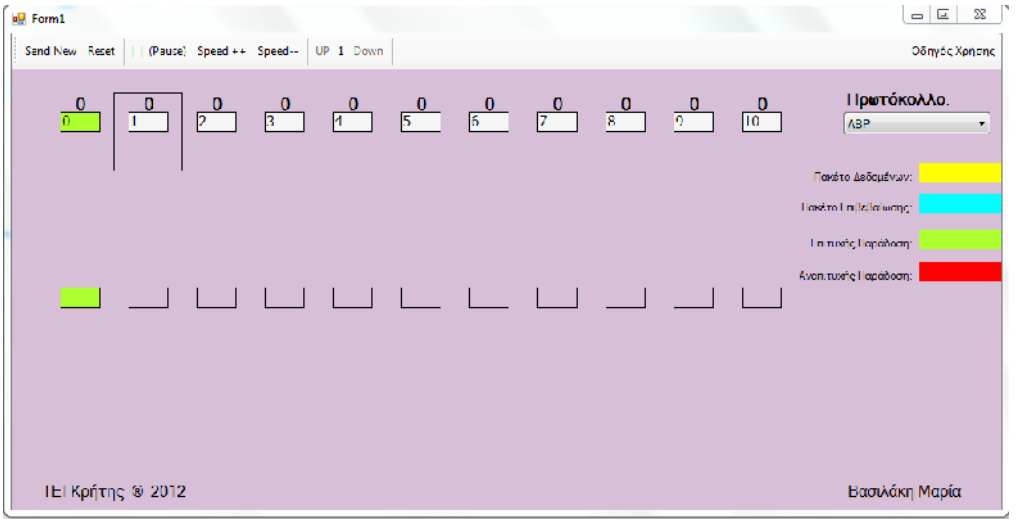
Μόλις ο αποστολέας μεταδώσει το πρώτο πακέτο, ξεκινάει ένας χρόνος προθεσμίας που διασφαλίζει ότι η επιβεβαίωση του πακέτου θα φτάσει εγκαίρως. Ο αποστολέας περιμένει την ACK από τον παραλήπτη πριν προχωρήσει στην αποστολή του επόμενου. Την ώρα που το πακέτο στέλνεται στον παραλήπτη είναι κίτρινου χρώματος όπως φαίνεται στο σχήμα 3.1. Αν το πακέτο δεν χαθεί κατά τη μετάδοσή του και το λάβει σωστά ο παραλήπτης τότε το πλαίσιο του παραλήπτη γίνεται πράσινου χρώματος (σχήμα 3.2) και ο παραλήπτης στέλνει πακέτο επιβεβαίωσης ACK μπλε χρώματος.

.



Σχήμα 3.2: Λειτουργία επιτυχούς παράδοσης και επιβεβαίωσης.

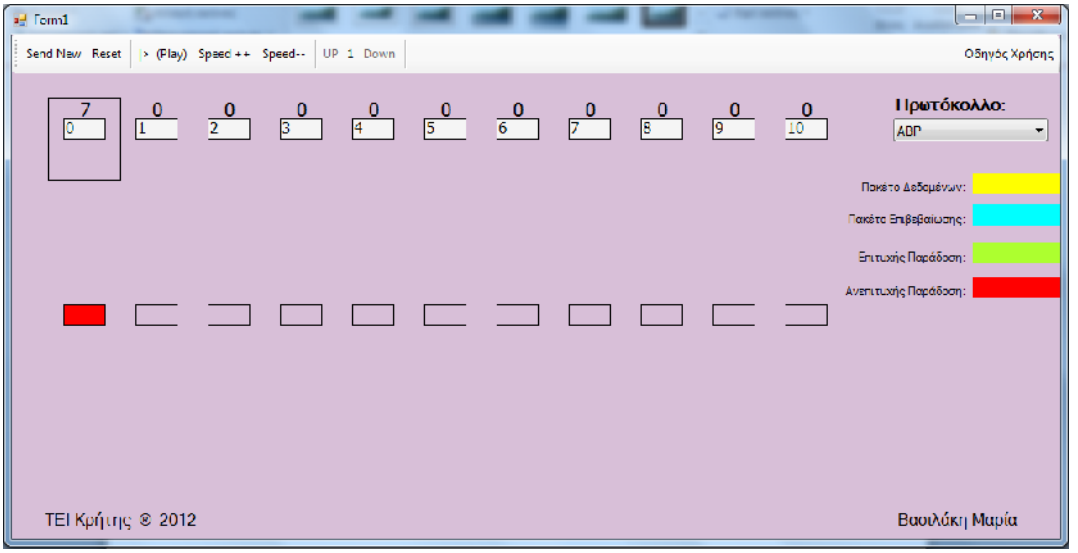
Εάν η επιβεβαίωση ACK φτάσει μέσα στον προσυμφωνημένο χρόνο προθεσμίας (timeout) και δεν χαθεί, τότε το πλαίσιο του αποστολέα γίνεται πράσινο, ο χρόνος προθεσμίας (timeout) μηδενίζει και από πλευράς αποστολέα, μετατοπίζεται στο επόμενο πακέτο. (Σχήμα 3.3)



Σχήμα 3.3: Λειτουργία λήψης επιτυχούς επιβεβαίωσης και ολίσθηση παραθύρου.

3.2.1 Σφάλμα πακέτου δεδομένων στο ABP πρωτόκολλο.

Αν το πακέτο χαθεί κατά τη μετάδοση του και δεν το λάβει σωστά ο παραλήπτης τότε το πλαίσιο του παραλήπτη γίνεται κόκκινου χρώματος. Προκειμένου να προσομοιώσουμε την απώλεια πακέτων, έχουμε υλοποιήσει την δυνατότητα στον χρήστη να κάνει διπλό κλικ όχι απαραίτητα στο ίδιο το πακέτο δεδομένων ή το πακέτο επιβεβαίωσης αλλά στην στήλη που κινείται μεταξύ αποστολέα και παραλήπτη. Κάνοντας διπλό κλικ την ώρα που στέλνεται το πακέτο, προσομοιώνουμε την περίπτωση σφαλμάτων στο πακέτο δεδομένων (σχήμα 3.4)

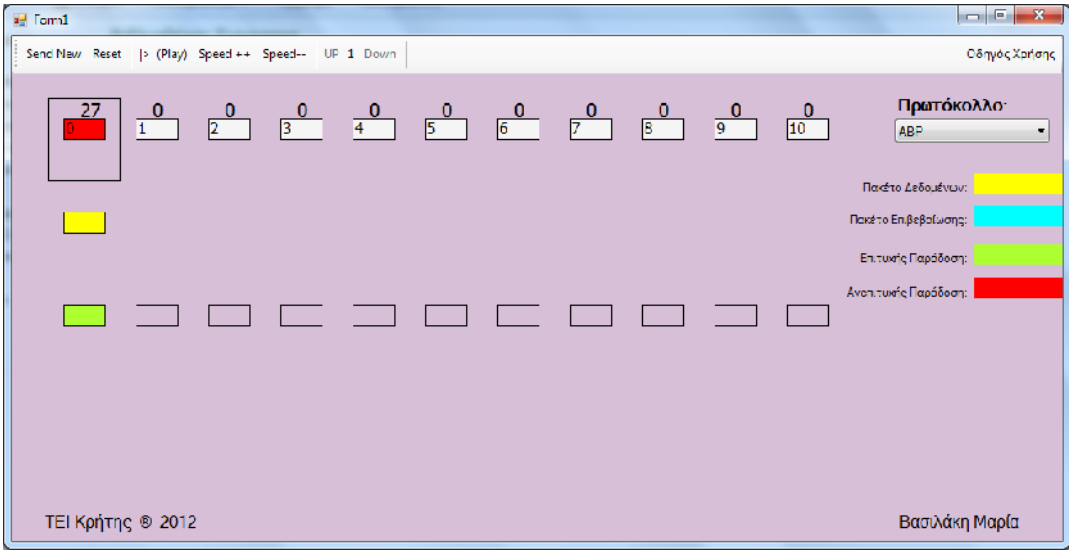


Σχήμα 3.4: Ανεπιτυχής παράδοση

Έτσι, ο αποστολέας ξαναστέλνει το πακέτο μόλις μηδενίσει ο χρόνος προθεσμίας (timer) και χρωματίζει το συγκεκριμένο πλαίσιο κόκκινο. Για κάθε χαμένο πακέτο, ο παραλήπτης γίνεται κόκκινου χρώματος στο μισό χρόνο προθεσμίας (timer) και ο αποστολέας μόλις μηδενίσει ο χρόνος προθεσμίας (timer)

3.2.2 Σφάλμα πακέτου επιβεβαίωσης στο ABP πρωτόκολλο.

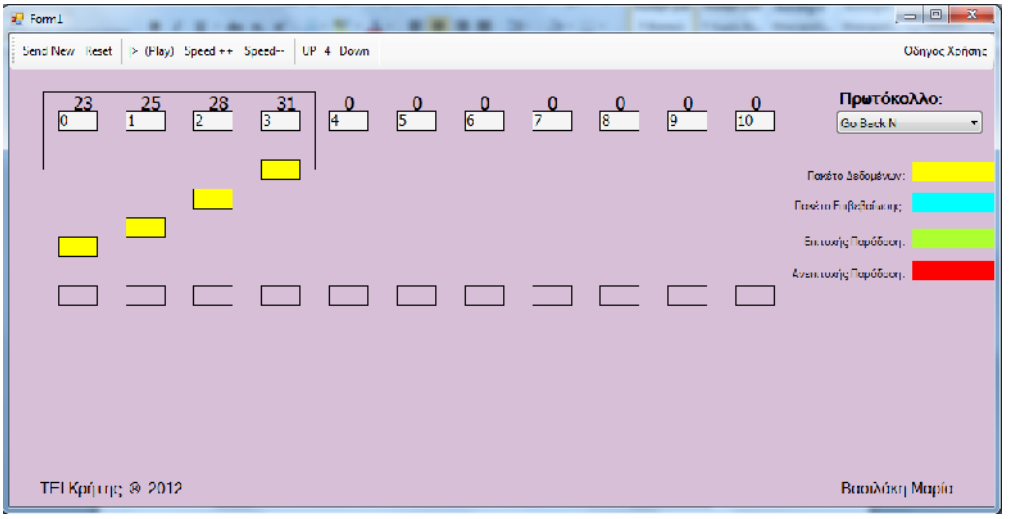
Αν η επιβεβαίωση ACK χαθεί, κάνοντας διπλό κλικ στη στήλη μεταξύ αποστολέα και παραλήπτη, ο αποστολέας ξαναστέλνει το πακέτο μόλις μηδενίσει ο χρόνος προθεσμίας (timer), ο παραλήπτης που είναι ήδη πράσινος γιατί το πακέτο είχε φτάσει σωστά και το έχει αποδεχτεί, δεν αλλάζει χρώμα. (Σχήμα 3.5)



Σχήμα 3.5: Επανεκπομπή πακέτου

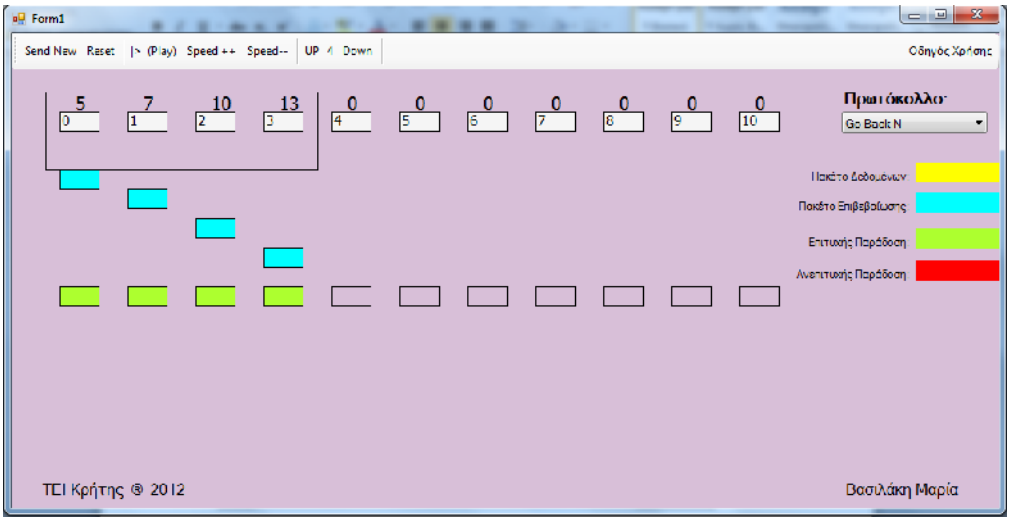
3.3 Πρωτόκολλο οπισθοχώρησης κατά N (Go back N)

Στο πρωτόκολλο GBN ο αποστολέας μπορεί να στέλνει ανεπιβεβαίωτα πακέτα μέχρι και το πλήθος του παραθύρου εκπομπής. Στο σχήμα 3.6 βλέπουμε τη μετάδοση 4 ανεπιβεβαίωτων πακέτων. Την ώρα που τα πακέτα στέλνονται στον παραλήπτη έχουν κίτρινη απόχρωση όπως και στο πρωτόκολλο ABP.



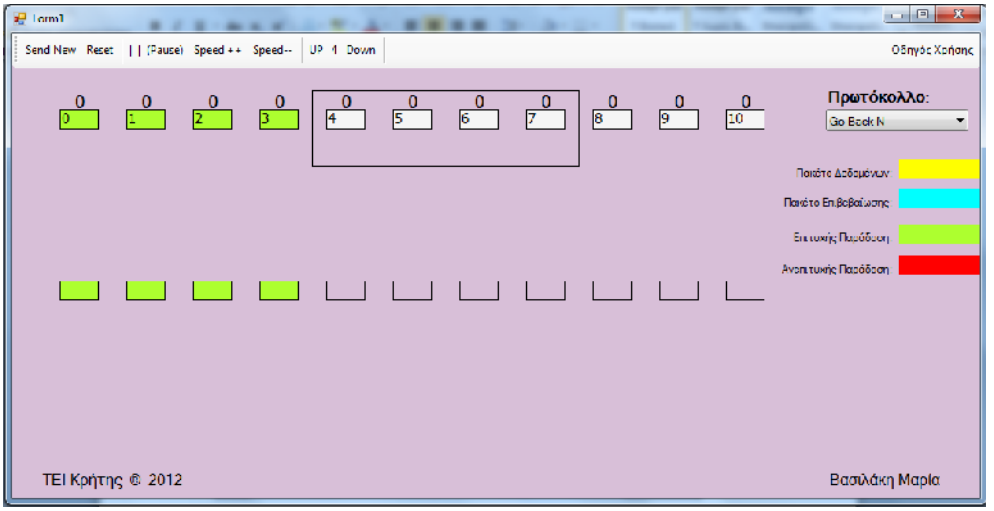
Σχήμα 3.6: Μετάδοση ανεπιβεβαίωτων πακέτων (GBN)

Αν τα πακέτα φτάσουν σωστά στον παραλήπτη και δεν χαθεί κανένα κατά τη μετάδοσή τους, τότε ο παραλήπτης των συγκεκριμένων πακέτων χρωματίζει πράσινα τα πλαίσια λήψης (σχήμα 3.7) και στέλνει τα αντίστοιχα πακέτα επιβεβαίωσης ACK στον αποστολέα με μπλε απόχρωση.



Σχήμα 3.7: Λειτουργία παράδοσης και επιβεβαίωσης

Εάν τα πακέτα επιβεβαίωσης ACK φτάσουν μέσα στον προσυμφωνημένο χρόνο προθεσμίας (timeout) και δεν χαθούν, τότε τα πλαίσια του αποστολέα γίνονται πράσινο χρώμα, ο χρόνος προθεσμίας (timeout) μηδενίζει και από πλευράς αποστολέα, μετατοπίζεται στα επόμενα πακέτα το παράθυρο που πλαισιώνει τα εκάστοτε ανεπιβεβαίωτα πακέτα. (Σχήμα 3.8)

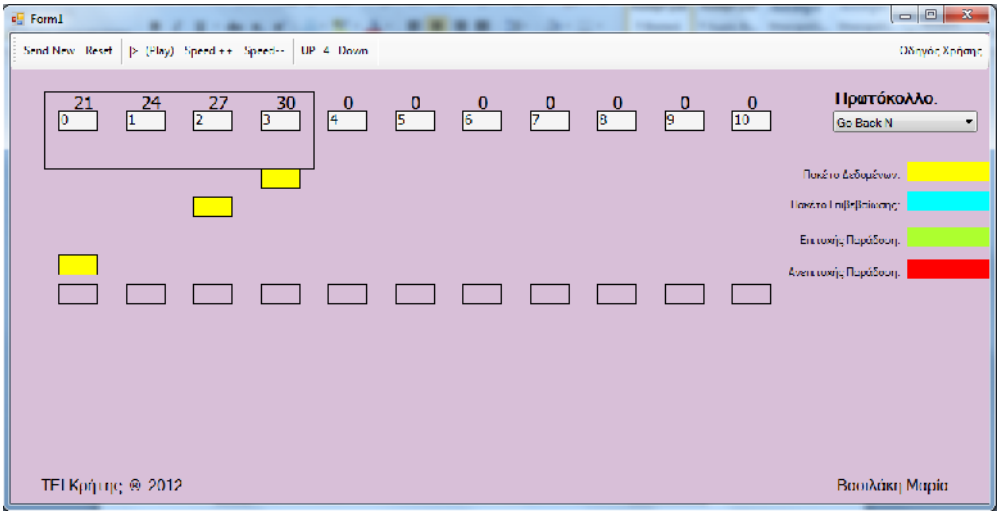


Σχήμα 3.8: Λειτουργία λήψης επιτυχών επιβεβαιώσεων και ολίσθηση παραθύρου

3.3.1 Σφάλμα πακέτου δεδομένων στο GBN πρωτόκολλο.

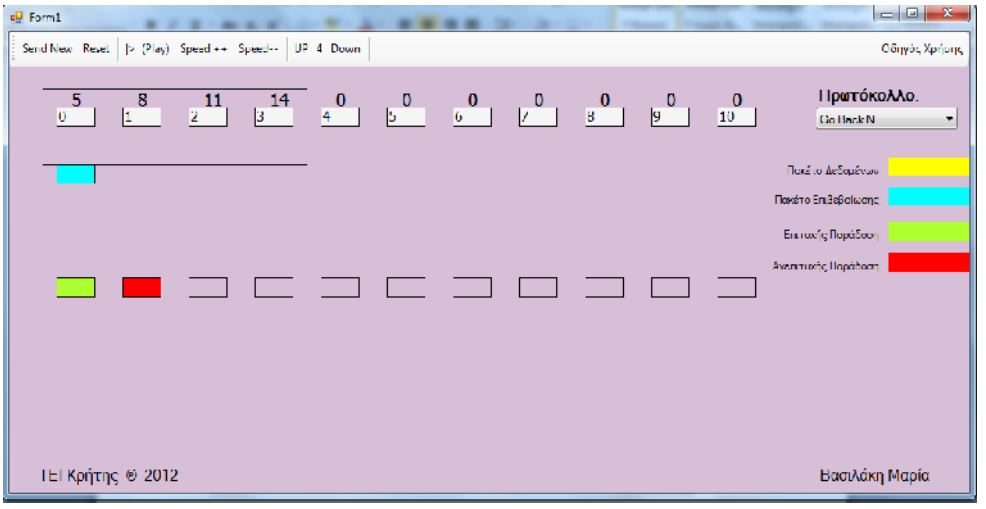
Αν κάποιο πακέτο χαθεί κατά την αποστολή του από τον αποστολέα στον παραλήπτη τότε το πλαίσιο του παραλήπτη του συγκεκριμένου πακέτου γίνεται κόκκινου χρώματος (όπως και στο ABP) και τα επόμενα πακέτα που έχουν σταλεί μετά από αυτό δεν γίνονται δεκτά από τον παραλήπτη. Ο παραλήπτης χρωματίζει το πλαίσιο του χαμένου πακέτου κόκκινο για να δείξει ότι υπάρχει σφάλμα, απορρίπτει τα επόμενα πακέτα γιατί είναι εκτός σειράς και δεν χρωματίζει τα αντίστοιχα πλαίσια τους. Ο αποστολέας επαναμεταδίδει τα πακέτα για τα οποία δεν λαμβάνει επιβεβαίωση πριν τη λήξη του χρόνου προθεσμίας καθώς και τα επόμενα και χρωματίζει το πλαίσιο του χαμένου πακέτου κόκκινο.

Για παράδειγμα στο σχήμα 3.9, δίνεται η δυνατότητα στο χρήστη πατώντας το πλήκτρο "Send new" 4 φορές να στείλει τα πρώτα 4 ανεπιβεβαίωτα πακέτα με αριθμούς 0, 1, 2 και 3, κάνοντας κλικ στη στήλη που κινείται το πακέτο 1, αυτό χάνεται λόγω σφαλμάτων. (Σχήμα 3.9).



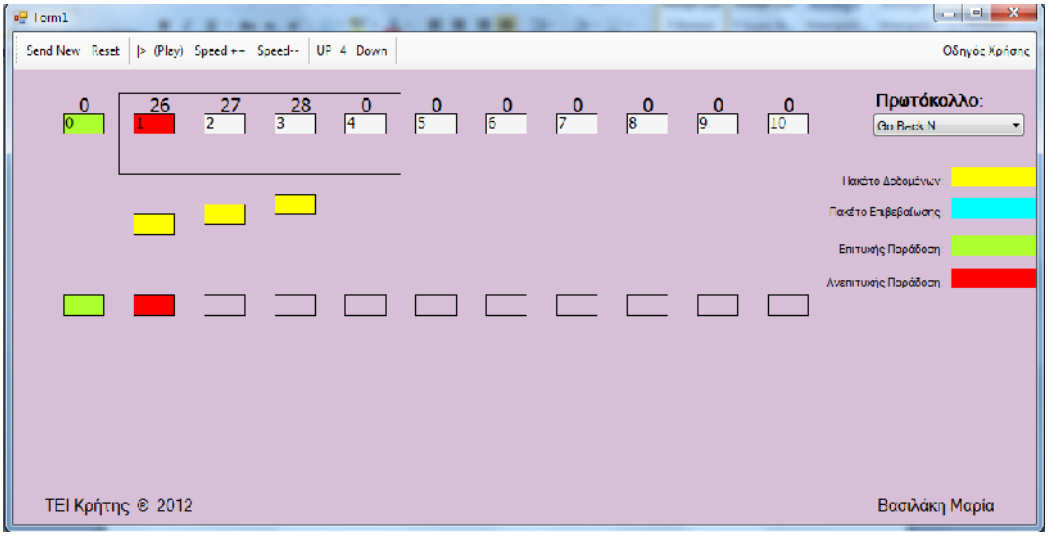
Σχήμα 3.9: Σφάλμα στην αποστολή πακέτων δεδομένων.

Αρχικά φτάνει επιτυχής το πακέτο 0, χρωματίζεται πράσινο το αντίστοιχο πλαίσιο του παραλήπτη και στέλνεται επιτυχής η επιβεβαίωση. Στο μισό χρόνο προθεσμίας timeout το αντίστοιχο πλαίσιο του παραλήπτη του πακέτου 1 γίνεται κόκκινο και τα επόμενα πλαίσια 2 και 3 δεν χρωματίζονται γιατί είναι εκτός σειράς και δεν στέλνονται επιβεβαιώσεις στον αποστολέα (σχήμα 3.10).



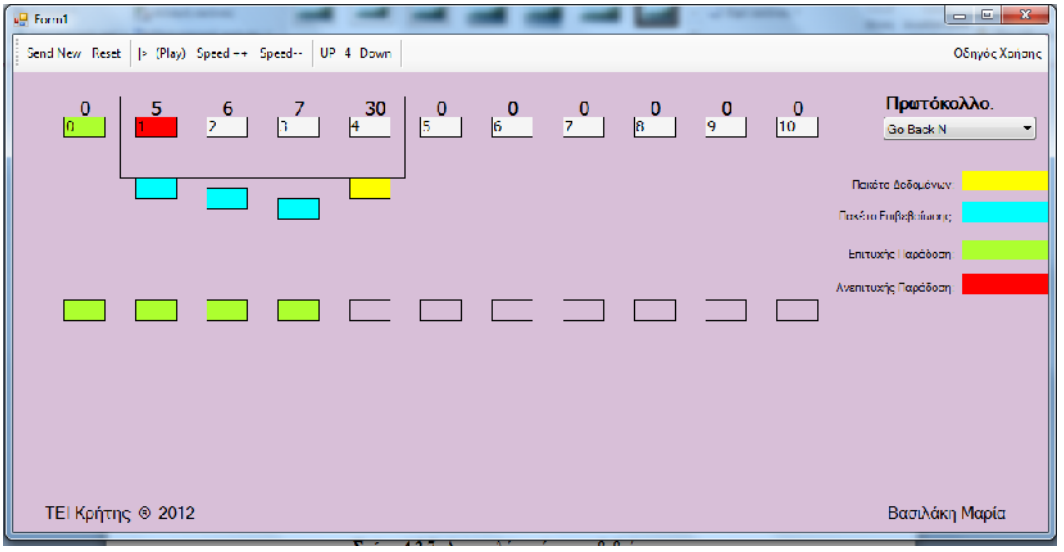
Σχήμα 3.10: Ανεπιτυχής παράδοση πακέτου δεδομένων και απόρριψη επόμενων.

Ο αποστολέας ξαναστέλνει τα πακέτα 1, 2 και 3 αφού δεν έχει λάβει τα αντίστοιχα πακέτα επιβεβαίωσης. (σχήμα 3.11)



Σχήμα 3.11: Επανεκπομπή πακέτων δεδομένων

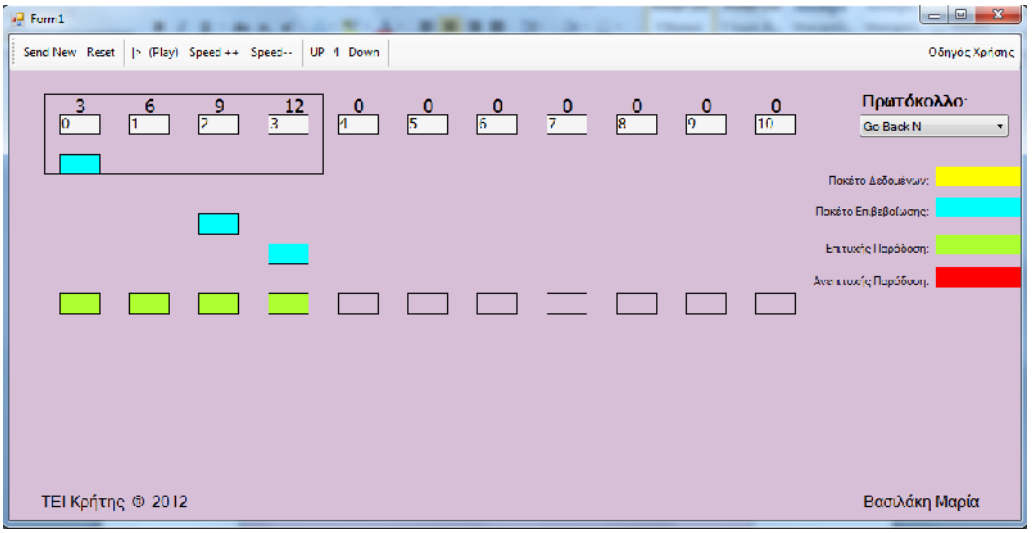
Ο παραλήπτης δέχεται τα πακέτα 1, 2, 3 αφού έχουν φτάσει σωστά, χρωματίζει τα αντίστοιχα πλαίσια πράσινα και στέλνει πακέτα επιβεβαίωσης. Το παράθυρο που πλαισιώνει τα πακέτα ολισθαίνει μια θέση δεξιά και έτσι δίνεται η δυνατότητα στο χρήστη πατώντας το πλήκτρο "Send new" να στείλει και το πακέτο 4. (Σχήμα 3.12)



Σχήμα 3.12: Λήψη πακέτων επιβεβαίωσης για δεύτερης εκπομπής πακέτα δεδομένων

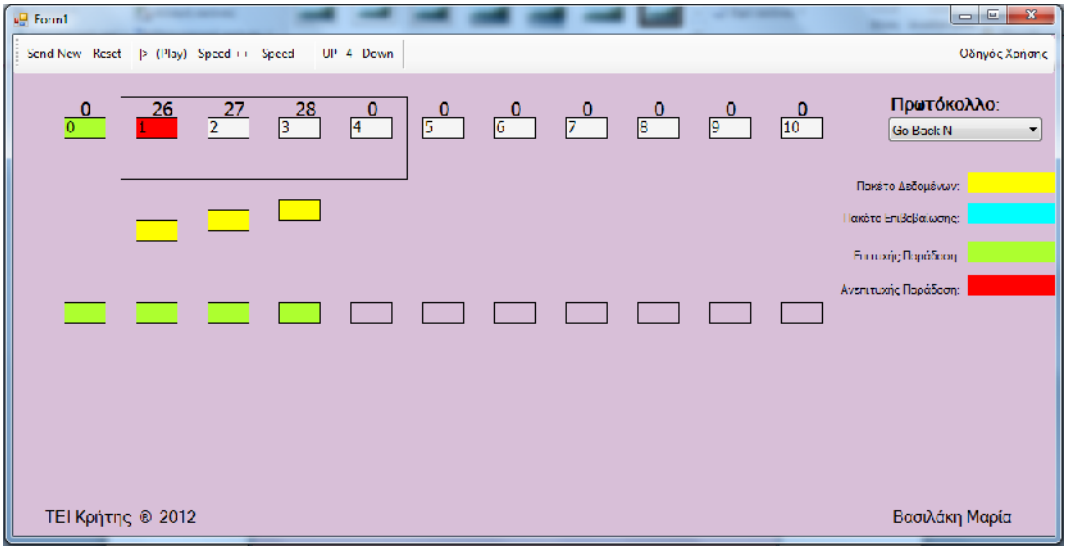
3.3.2 Σφάλμα πακέτου επιβεβαίωσης στο GBN πρωτόκολλο.

Την ώρα που ο παραλήπτης στέλνει πακέτα επιβεβαίωσης στον αποστολέα δίνεται η δυνατότητα στο χρήστη κάνοντας διπλό κλικ, για παράδειγμα στη στήλη που κινείται το πακέτο 1, να εξαφανίσει το συγκεκριμένο πακέτο επιβεβαίωσης λόγω σφαλμάτων. (σχήμα 3.13)



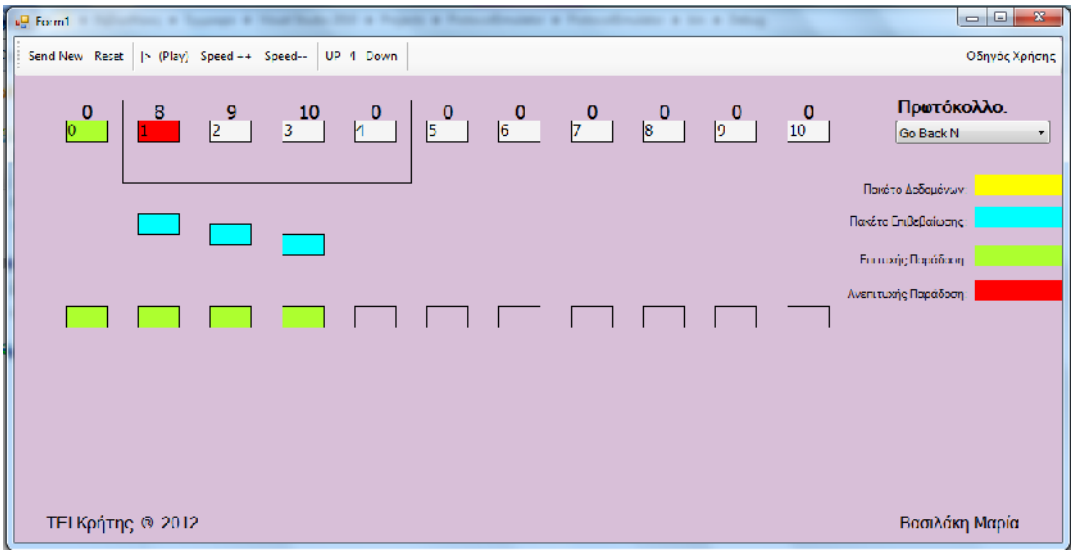
Σχήμα 3.13: Σφάλμα στην αποστολή επιβεβαιώσεων

Ο αποστολέας χρωματίζει το αντίστοιχο πλαίσιο κόκκινο και ξαναστέλνει τα πακέτα 1, 2 και 3. (σχήμα 3.14)



Σχήμα 3.14: Επανεκπομπή πακέτων δεδομένων λόγω σφαλμάτων στην μετάδοση πακέτων επιβεβαίωσης

Το πλαίσιο του παραλήπτη δεν αλλάζει αφού έχουν ήδη ληφθεί τα πακέτα 1, 2 και 3 και απλά ξαναστέλνονται τα αντίστοιχα πακέτα επιβεβαίωσης. (σχήμα 3.15)

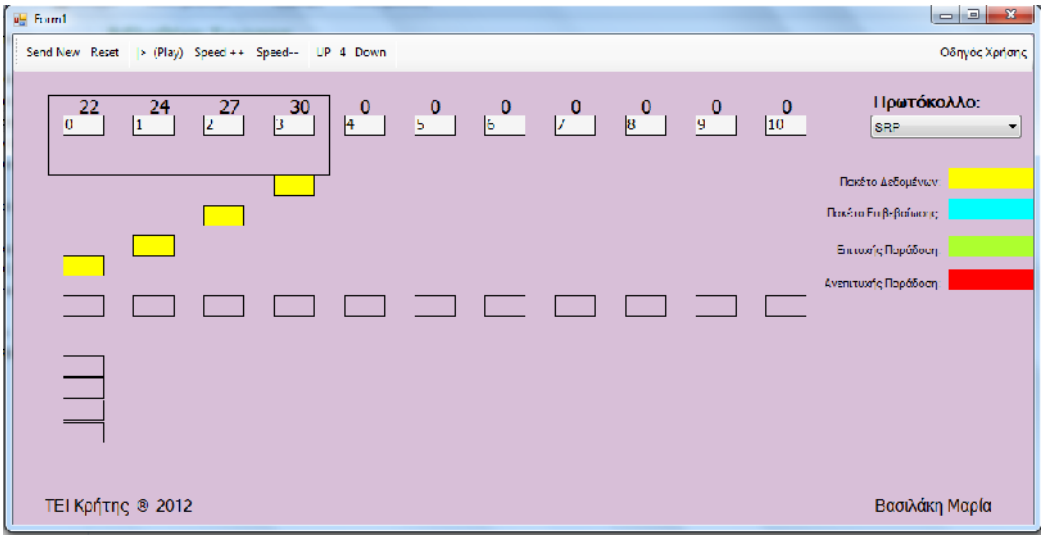


Σχήμα 3.15: Επανεκπομπή πακέτων επιβεβαίωσης

3.4 Πρωτόκολλο επιλεκτικής επανάληψης (Selective Repeat Protocol SRP)

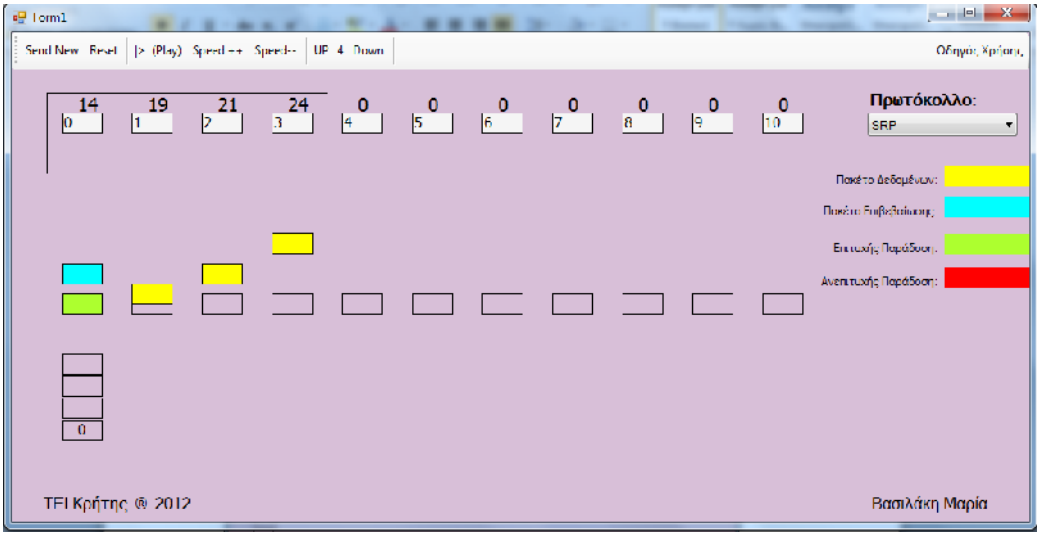
Το πρωτόκολλο επιλεκτικής επανάληψης βασίζεται στους μηχανισμούς επιβεβαίωσης λήψης και χρόνου προθεσμίας και επιτρέπει περισσότερα από ένα ανεπιβεβαίωτα πλαίσια να βρίσκονται κάθε στιγμή υπό μεταφορά. Σε αντίθεση με το GBN χρησιμοποιεί ένα καταχωρητή στον παραλήπτη. Εκεί αποθηκεύονται προσωρινά τα πακέτα που λαμβάνονται εκτός σειράς, μέχρι να προωθηθούν προς επεξεργασία, βελτιώνοντας την απόδοση του πρωτοκόλλου. Τα αποθηκευμένα πακέτα προωθούνται στα ανώτερα επίπεδα αρχιτεκτονικής OSI του παραλήπτη, όταν ληφθεί το πακέτο που έλειπε για να υπάρχει συνέχεια.

Στο πρωτόκολλο SRP, ο αποστολέας μπορεί να στέλνει ανεπιβεβαίωτα πακέτα μέχρι και το πλήθος του παραθύρου εκπομπής. Επαναμεταδίδει μόνο τα πακέτα που δεν έχουν επιβεβαιωθεί πριν από το χρόνο προθεσμίας. Στο σχήμα 3.16 βλέπουμε τη μετάδοση 4 ανεπιβεβαίωτων πακέτων. Την ώρα που τα πακέτα στέλνονται στον παραλήπτη έχουν κίτρινη απόχρωση όπως και στα πρωτόκολλα ABP και GBN.



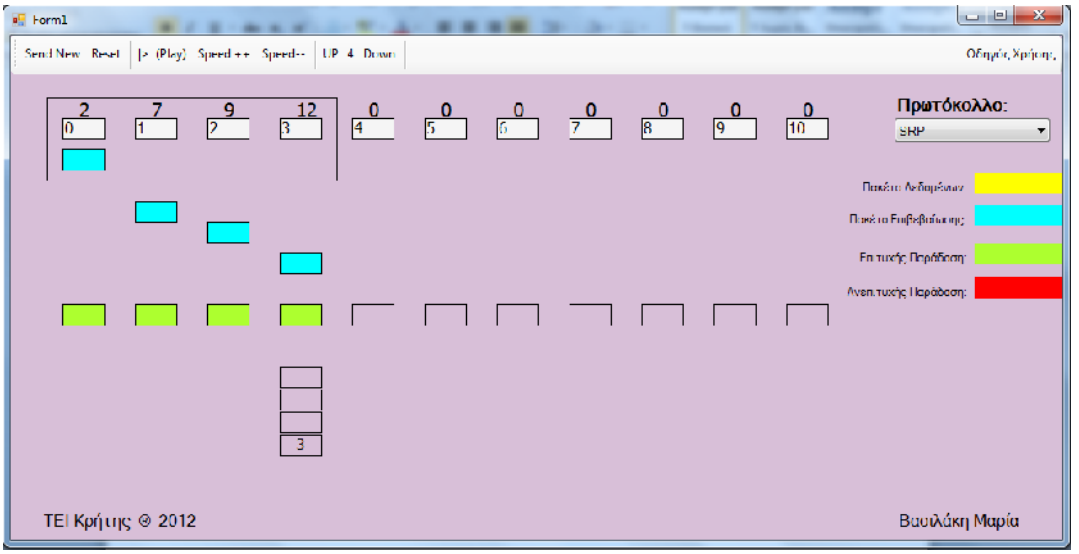
Σχήμα 3.16: Μετάδοση πακέτων δεδομένων (SRP).

Όταν φτάσει το πρώτο πακέτο "0", ο παραλήπτης το αποθηκεύει στον καταχωρητή και στη συνέχεια το παραδίδει στα ανώτερα στρώματα, το ίδιο κάνει και με τα επόμενα πακέτα που δεν χάνονται και παραδίνονται στη σωστή σειρά. (Σχήμα 3.17)



Σχήμα 3.17: Αποθήκευση πρώτου πακέτου στον καταχωρητή.

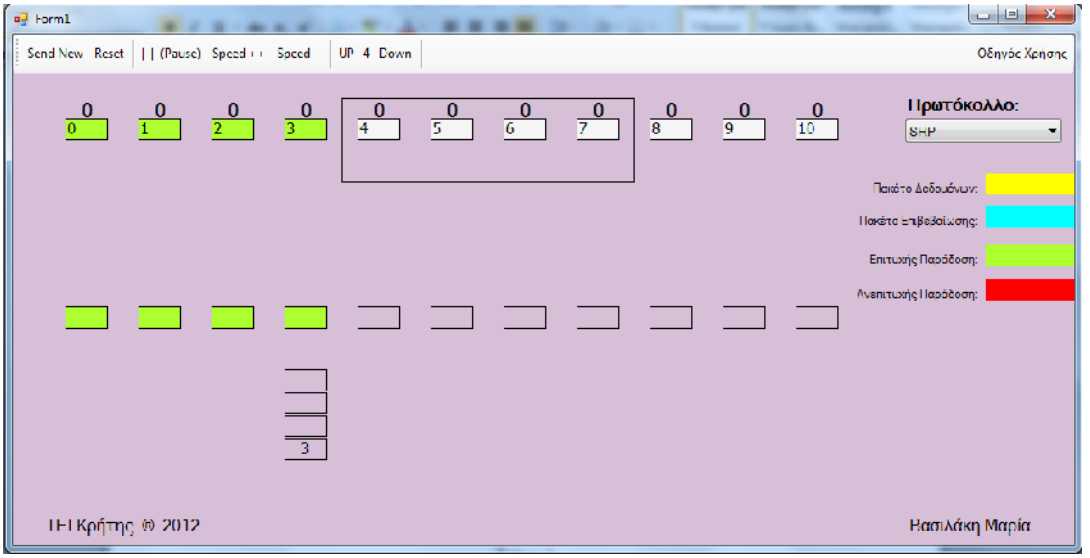
Στη συνέχεια, αν τα πακέτα φτάσουν σωστά στον παραλήπτη και δεν χαθεί κανένα κατά τη μετάδοσή τους τότε τα πλαίσια του παραλήπτη των συγκεκριμένων πακέτων γίνονται πράσινα. Ο παραλήπτης στέλνει πακέτα επιβεβαίωσης ACK στον αποστολέα που αναπαρίστανται με πλαίσια μπλε απόχρωσης. (σχήμα 3.18)



Σχήμα 3.18: Λειτουργία επιτυχούς παράδοσης πακέτου δεδομένων και πακέτου επιβεβαίωσης

Ο παραλήπτης αποθηκεύει τα πακέτα που λαμβάνονται εκτός σειράς σε έναν καταχωρητή ώστε να τα παραδώσει στη σωστή σειρά στα ανώτερα στρώματα.

Εάν τα πακέτα επιβεβαίωσης ACK φτάσουν μέσα στον προσυμφωνημένο χρόνο προθεσμίας (timeout) και δεν χαθούν, τότε τα πλαίσια του αποστολέα γίνονται πράσινη απόχρωση, ο χρόνος προθεσμίας (timeout) μηδενίζει και από πλευρας αποστολέα, το παράθυρο, που πλαισιώνει τα ανεπιβεβαίωτα πακέτα, μετατοπίζεται δεξιότερα. (Σχήμα 3.19) Προγραμματιστικά, ο καταχωρητής παραμένει κάτω από το πλαίσιο παραλαβής του τελευταίου πακέτου μέχρι να λάβει το επόμενο πακέτο.

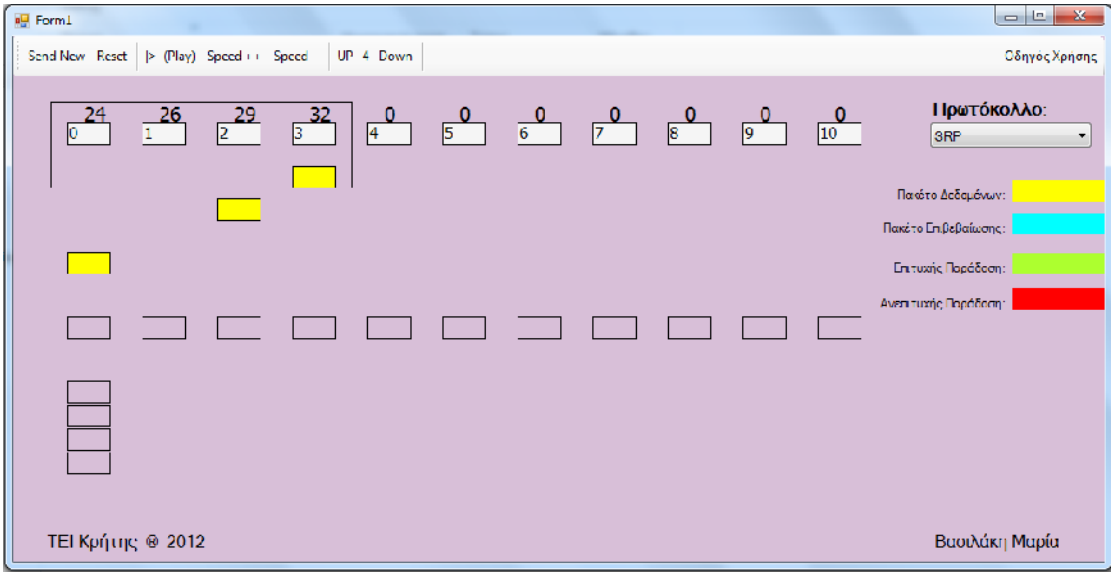


Σχήμα 3.19: Λειτουργία λήψης επιβεβαίωσης και ολίσθηση παραθύρου

3.4.1 Σφάλμα πακέτου δεδομένων στο SRP πρωτόκολλο.

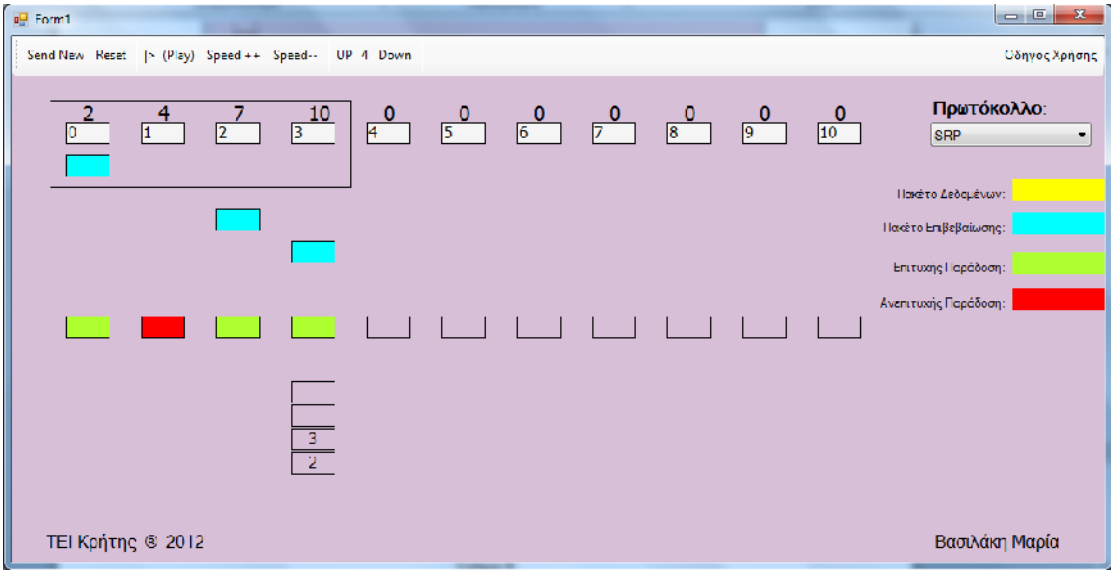
Αν κάποιο πακέτο χαθεί κατά την διαδρομή του από τον αποστολέα στον παραλήπτη τότε το πλαίσιο παραλαβής του συγκεκριμένου πακέτου γίνεται κόκκινου χρώματος για να δείξει ότι υπάρχει σφάλμα. (όπως και στα προηγούμενα πρωτόκολλα). Ο παραλήπτης δέχεται τα επόμενα πακέτα που έχουν σταλεί μετά από αυτό χρωματίζοντάς τα πράσινα και τα αποθηκεύει στον καταχωρητή περιμένοντας να έρθει και το χαμένο πακέτο ώστε να τα παραδώσει με τη σωστή σειρά. Ο αποστολέας επαναμεταδίδει μονο τα πακέτα για τα οποία δεν λαμβάνει επιβεβαίωση πριν τη λήξη του χρόνου προθεσμίας.

Για παράδειγμα στο σχήμα 3.20, δίνεται η δυνατότητα στο χρήστη πατώντας το πλήκτρο "Send new" 4 φορές να στείλει τα πρώτα 4 ανεπιβεβαίωτα πακέτα με αριθμούς 0, 1, 2 και 3, κάνοντας διπλό κλικ στη στήλη που κινείται το πακέτο 1, αυτό χάνεται λόγω σφαλμάτων. (Σχήμα 3.20).



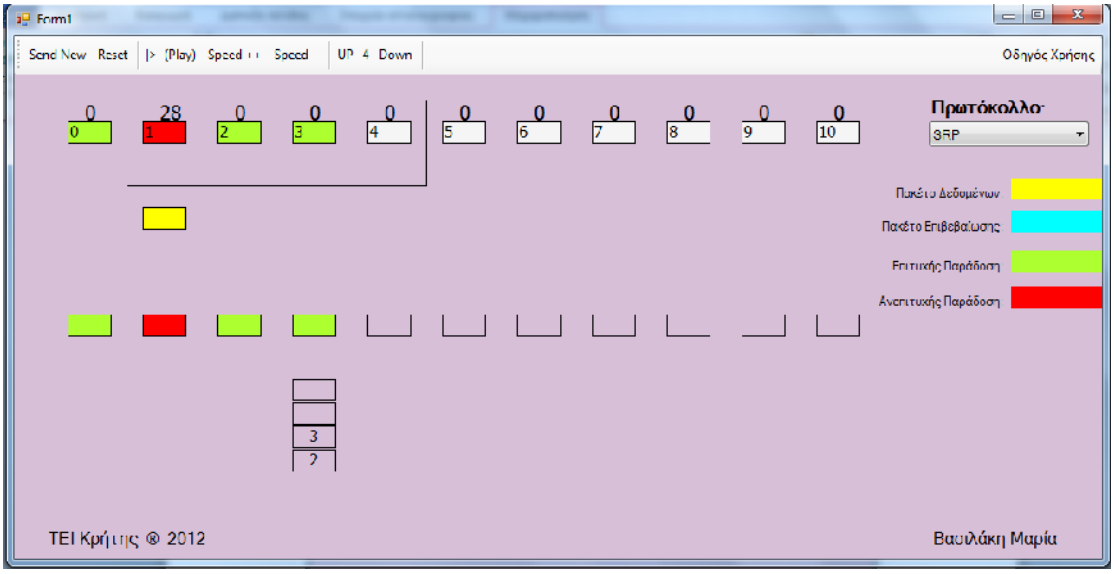
Σχήμα 3.20: Σφάλμα στην αποστολή πακέτων δεδομένων.

Στο μισό χρόνο προθεσμίας timeout το πλαίσιο του παραλήπτη με τον αριθμό 1 γίνεται κόκκινου χρώματος και τα κουτιά του παραλήπτη με τους αριθμούς 2 και 3 γίνονται πράσινα γιατί στο SRP πρωτόκολλο ο παραλήπτης δέχεται τα εκτός σειράς πακέτα στέλνοντας αντίστοιχα πακέτα επιβεβαίωσης στον αποστολέα και τα αποθηκεύει στον καταχωρητή περιμένοντας να έρθει το χαμένο πακέτο ώστε να τα παραδώσει με τη σωστή σειρά (σχήμα 3.21).



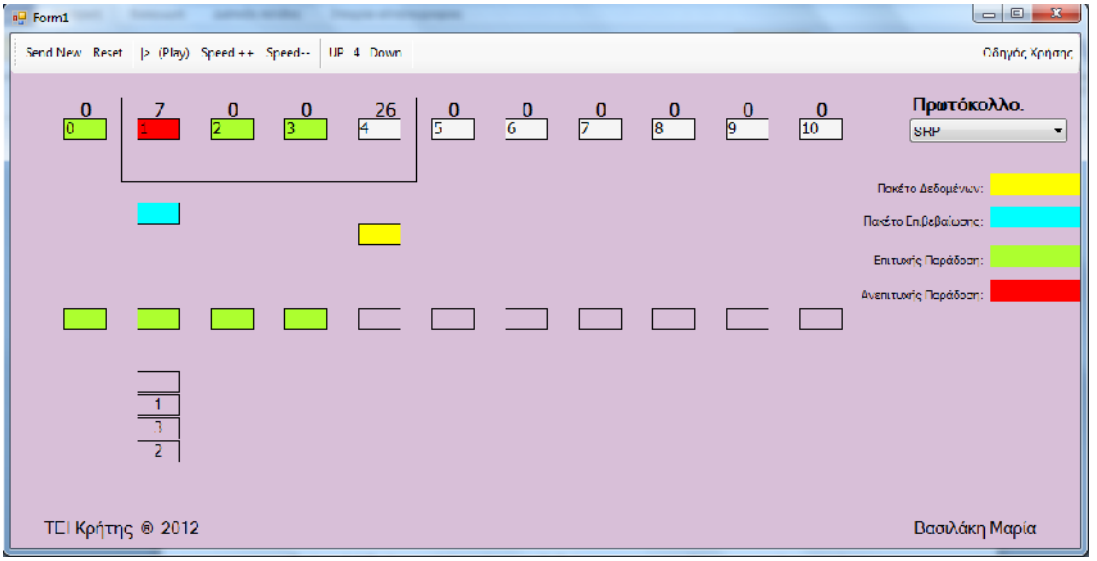
Σχήμα 3.21: Ανεπιτυχής παράδοση και αποθήκευση επόμενων πακέτων δεδομένων στον καταχωρητή.

Ο αποστολέας ξαναστέλνει μόνο το πακέτο 1 αφού δεν έχει λάβει το αντίστοιχο πακέτο επιβεβαίωσης και χρωματίζει το αντίστοιχο πλαίσιο κόκκινο (σχήμα 3.22). Το παράθυρο που πλαισιώνει τα πακέτα ολισθαίνει μια θέση δεξιά και έτσι δίνεται η δυνατότητα στο χρήστη πατώντας το πλήκτρο "Send new" να στείλει και το επόμενο πακέτο 4.



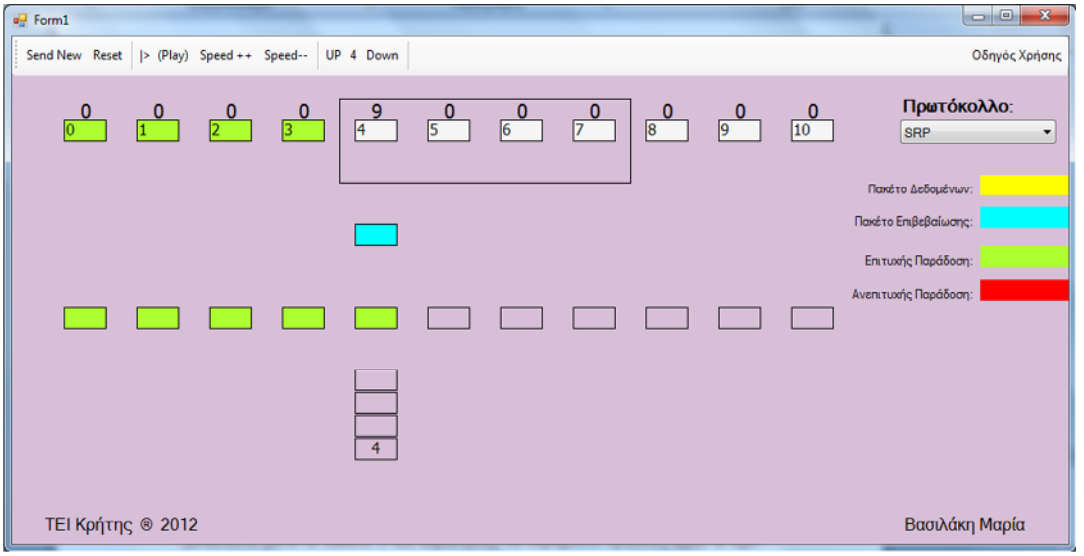
Σχήμα 3.22: Επανεκπομπή εσφαλμένου πακέτου δεδομένων.

Ο παραλήπτης δέχεται το πακέτο 1, χρωματίζοντάς το με πράσινο χρώμα, το αποθηκεύει στον καταχωρητή και στέλνει το πακέτο επιβεβαίωσής του. (Σχήμα 3.23)



Σχήμα 3.23: Αποστολή πακέτου επιβεβαίωσης και αποθήκευση επανεκπεμπόμενου πακέτου στον καταχωρητή.

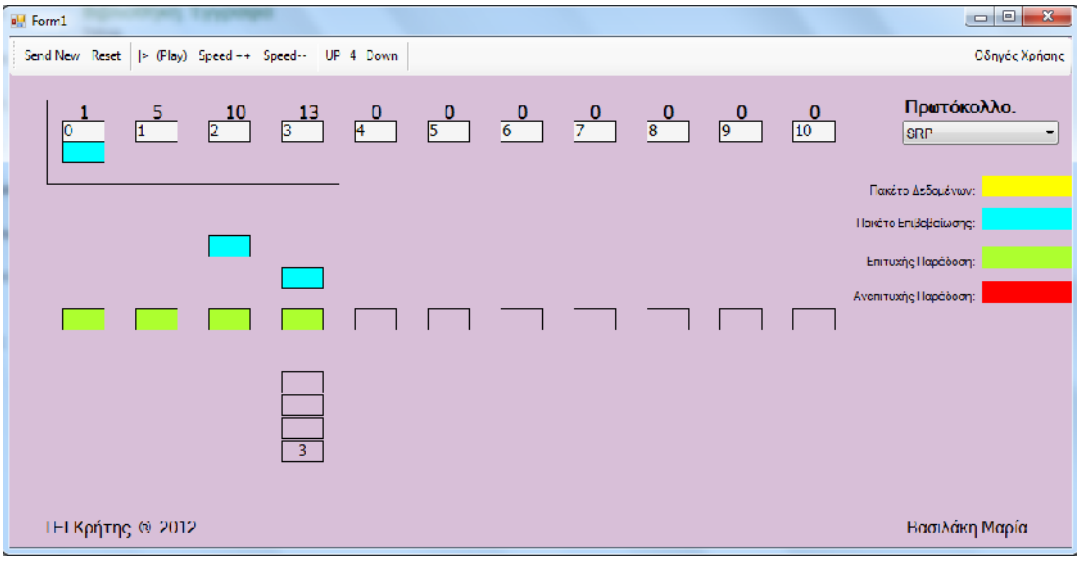
Όταν φτάσει η επανεκπομπή του πακέτου 1 ο καταχωρητής αδειάζει. Όταν φτάσει το πακέτο 4, ο παραλήπτης κάνει το αντίστοιχο πλαίσιο πράσινο και στέλνει το πακέτο επιβεβαίωσής του με μπλε χρώμα. (σχήμα 3.24)



Σχήμα 3.24: Αποθήκευση πακέτου 4 και εκπομπή πακέτου επιβεβαίωσης.

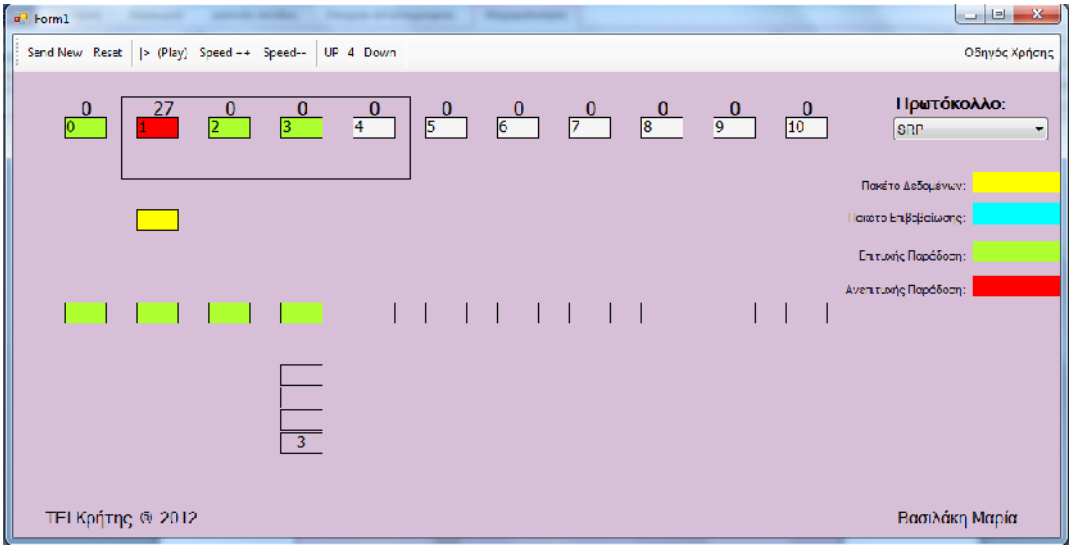
3.4.2 Σφάλμα πακέτου επιβεβαίωσης στο SRP πρωτόκολλο.

Την ώρα που ο παραλήπτης στέλνει πακέτα επιβεβαίωσης στον αποστολέα δίνεται η δυνατότητα στο χρήστη κάνοντας διπλό κλικ, για παράδειγμα στη στήλη που κινείται το πακέτο 1, να «σκοτώσει» το συγκεκριμένο πακέτο επιβεβαίωσης λόγω σφαλμάτων. (σχήμα 3.25)



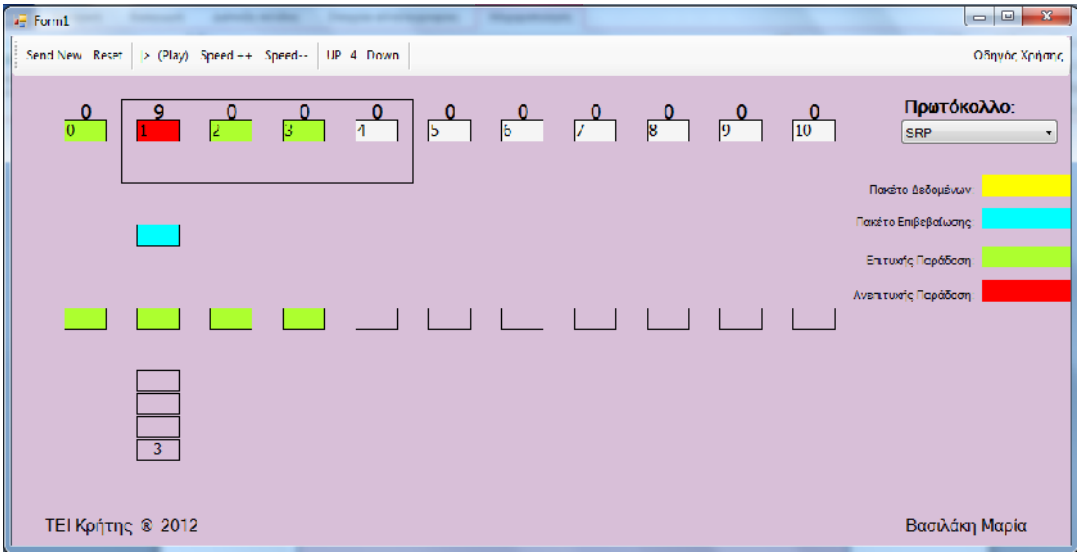
Σχήμα 3.25: Σφάλμα στην αποστολή πακέτων επιβεβαίωσης.

Ο αποστολέας θα ξαναστείλει το πακέτο 1 και χρωματίζει το αντίστοιχο πλαίσιο κόκκινο. (σχήμα 3.26)



Σχήμα 3.26: Μετάδοση εσφαλμένου πακέτου επιβεβαίωσης.

Το πλαίσιο του παραλήπτη δεν αλλάζει αφού έχει ήδη ληφθεί το πακέτο 1 και απλά ξαναστέλνεται το πακέτο επιβεβαίωσής του. (σχήμα 3.27)



Σχήμα 3.27: Επανεκπομπή πακέτου επιβεβαίωσης.

Το παράθυρο που πλαισιώνει τα πακέτα ολισθαίνει μια θέση δεξιά και έτσι δίνεται η δυνατότητα στο χρήστη πατώντας το “Send New” να στείλει το πακέτο 4.

3.5 Αναλυτική περιγραφή τμημάτων κώδικα.

Ο κώδικας που μας δείχνει πως κινούνται τα πακέτα είναι ο παρακάτω.

```
            if (stelnete || killed) {
                stepCount += 1;
                timer = ((((2 * travelLenght) / packetStep) - stepCount));//όλη η
απόσταση/πόσο προχωράει το πακέτο κάθε φορά)- πόσα βήματα έχουν γίνει)

            }
            if (stelnete)
            {
                if (einaiACK) {

                    if (y <= startY) //ελέγχω αν το πακέτο έφτασε πάνω
                    {
                        stelnete = false;
                        resend = false;
                        senderColor = Color.Green;
                        done = true;
                    }
                    else {
                        y -= packetStep; //πάω το πακέτο πιο πάνω
                        if (justACK)
                            justACK = false;
                    }

                }
                else {

                    if (y >= startY + travelLenght) //ελέγχω αν το πακέτο έφτασε
κάτω
                    {
                        eftase = true;
                        einaiACK = true;
                        justACK = true;
                        packetColor = Color.Aqua;

                        if (killed) {
                            receiverColor = Color.Red;
```

Σχήμα 3.28: Κώδικας μετατόπισης δεδομένων - επιβεβαίωσης

Το πακέτο ξεκινάει από τον αποστολέα να πηγαίνει προς τα κάτω και ελέγχουμε τη θέση του κάθε στιγμή. Όταν το πακέτο φτάσει επιτυχώς στον παραλήπτη, το συγκεκριμένο πλαίσιο γίνεται πράσινο και ένα πακέτο επιβεβαώσης χρώματος μπλε αποστέλλεται προς τον αποστολέα των δεδομένων. Αν το πακέτο δεδομένων δεν φτάσει επιτυχώς στον παραλήπτη τοτε το πλαίσιο γίνεται κόκκινο. Αυτή η διαδικασία ακολουθείται και στα 3 πρωτόκολλα. Κάθε πρωτόκολλο έχει διαφορετικό τρόπο λειτουργίας.

Ο κώδικας που ξεχωρίζει τα πρωτόκολλα είναι ο εξής :

```
private void cmbProtocolo_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cmbProtocolo.SelectedIndex == 0) { // το 0 είναι το GBN
                protocolo = Protocol.GBN;
                btnUp.Enabled = true; //ενεργοποιημένο το up και το down
                btnDown.Enabled = true;
            }
            else if (cmbProtocolo.SelectedIndex == 1) { // το 1 είναι το SRP
                protocolo = Protocol.SRP;
                ChangePacketLeght(4); // έχει 4 πακέτα
                btnUp.Enabled = true; //ενεργοποιημένο το up και το down
                btnDown.Enabled = true;

            }
            else if (cmbProtocolo.SelectedIndex == 2) { // το 2 ειναι το ABP
                protocolo = Protocol.ABP;
                ChangePacketLeght(1); // έχει 1 πακέτο
                btnUp.Enabled =false;
                btnDown.Enabled =false; // απενεργοποιεί το up και το down

            }
            Reset();
        }
```

Σχήμα 3.29: Κώδικας επιλογής πρωτοκόλλου

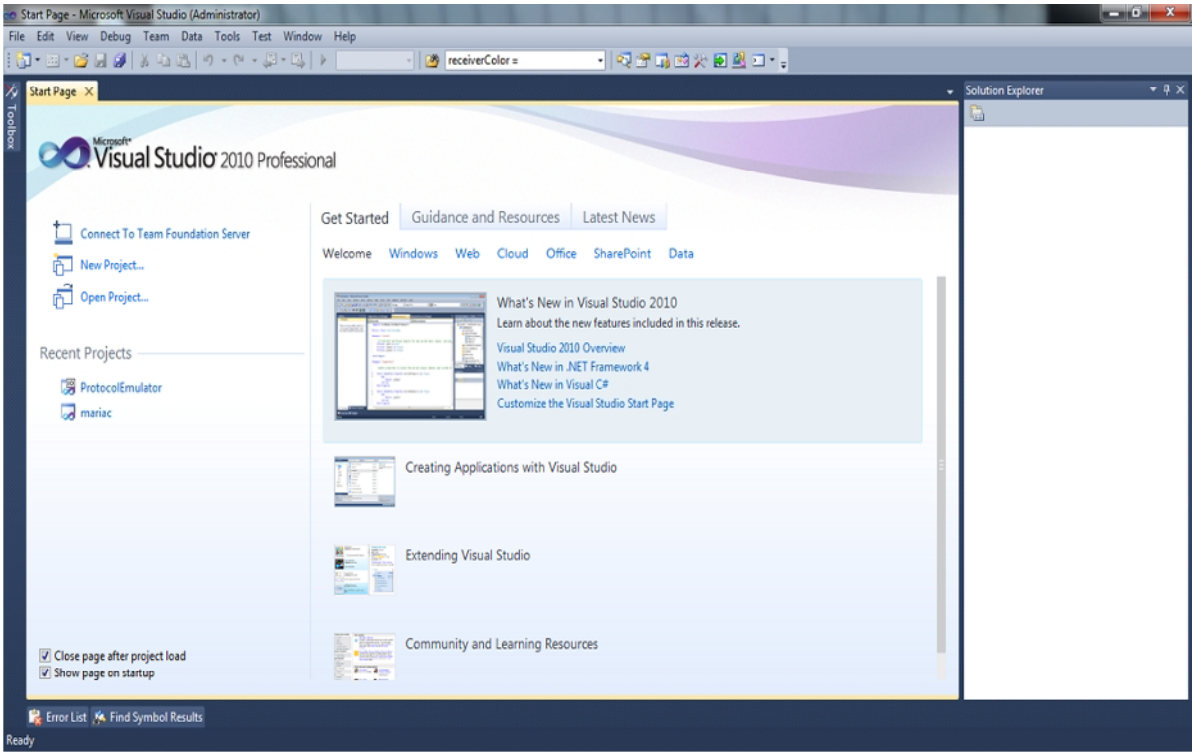
Το πρόγραμμα δεν μπορεί να καταλάβει τις λέξεις GBN, SRP και ABP ,έτσι συμβολίσαμε τα πρωτόκολλα μας με αριθμούς . Το GBN με τον αριθμό 0 . το SRP με τον αριθμό 1 και το με τον αριθμό 2.

Κεφάλαιο 4

Η C# στην προσομοίωση της πρακτικής

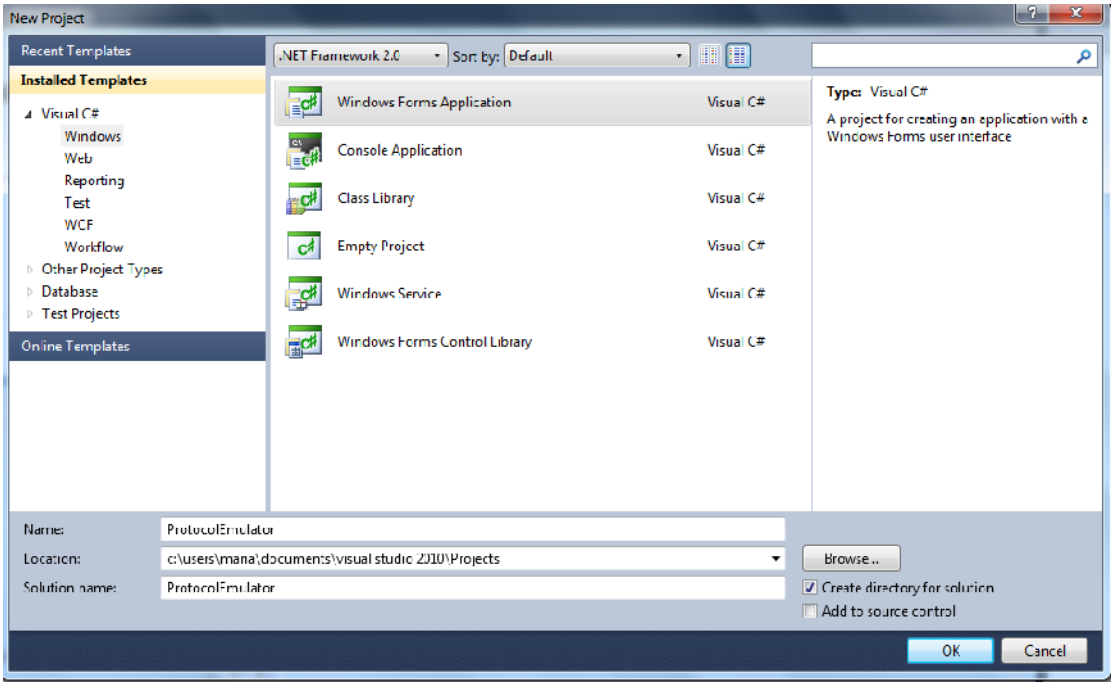
4.1 Η C# στην ανάπτυξη του κώδικα προσομοιωτή πρωτοκόλλων.

Για να δούμε την ανάπτυξη του κώδικά μας σε γλώσσα C#, ανοίγουμε την πλατφόρμα ανάπτυξης Microsoft Visual Studio 2010 όπου μας εμφανίζεται το κεντρικό παράθυρο της εφαρμογής. (Εικόνα 4.1)



Εικόνα 4.1: Οθόνη εισαγωγής Microsoft Visual Studio 2010

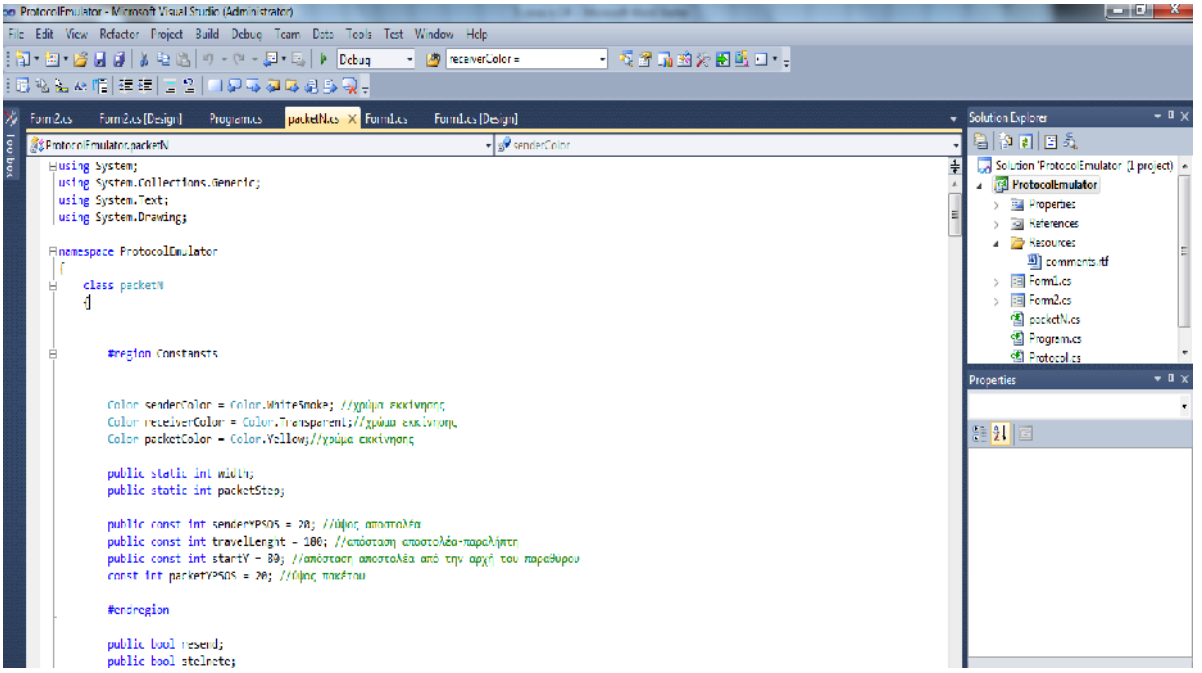
Πατάμε το "New Project" και μας εμφανίζει το παρακάτω παράθυρο. (Εικόνα 4.2)



Εικόνα 4.2: New Project

Εισάγουμε στο πλαίσιο ονόματος το όνομα " ProtocolEmulator", πατάμε το πλήκτρο "OK" και η πλατφόρμα είναι έτοιμη για ανάπτυξη κώδικα. Όταν εγκαταστήσαμε το Microsoft Visual Studio 2010 επιλέξαμε η γραφή του κώδικα να είναι σε C#.Net.

Για να εξακριβώσουμε πως λειτουργεί ο κώδικας που αναπτύξαμε, ανοίγουμε την εφαρμογή μας. (Εικόνα 4.3)



Εικόνα 4.3: ProtocolEmulator

4.2 Χαρακτηριστικά της C# στις κλάσεις PacketN και Form1

Οι δηλώσεις **using** στην αρχή του προγράμματος καθορίζουν ποιες βιβλιοθήκες θα χρησιμοποιήσουμε στην εφαρμογή. Το **namespace** ορίζει ένα «χώρο μνήμης» μέσα στο οποίο θα δημιουργούνται οι μεταβλητές του προγράμματος, θα αποθηκεύονται προσωρινά αποτελέσματα των ενδιάμεσων πράξεων. Το βασικό πρόγραμμα είναι μια κλάση (**class**). Στην C# όλος ο κώδικας της εφαρμογής πρέπει να είναι μέρος μιας κλάσης, εν αντιθέσει με την C++ που επιτρέπει δημιουργία κώδικα και εκτός κλάσης. Το συγκεκριμένο πρόγραμμα προσομοίωσης έχει υλοποιηθεί σε δύο κλάσεις, την κλάση **PacketN** και την κλάση **Form1.** Όταν τρέχει η εφαρμογή, εκτελείται η κλάση Form1 και μέσω αυτής καλείται η κλάση PacketN. (σαν αντικείμενο), δηλαδή όπου βλέπουμε `<packetN>` στη κλάση Form1 παίρνει αρχεία από την κλάση PacketN.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ProtocolEmulator
{

    public partial class Form1 : Form
    {

        const int kenoSenders = 30; // κενό ανάμεσα στα πακέτα
        const int kenoAristera = 50;


        #region Variables

        List<packetN> paketa;
        List<int> resetIndexes;

        bool pause;
        bool simulationRunning;
        bool stopSimulation; //όταν κλείνει το παράθυρο πρέπει να σταματάει το
simulation
        public static int simulationSpeed;
        public static int pluthosPaketon;
```

Σχήμα 4.1 : Πρώτες εντολές κλάση **Form1**

4.3 Κλάση PacketN

Στην κλάση PacketN έχουμε σχεδιάσει τα πακέτα μας, το πλαίσιο του αποστολέα, την αρίθμηση των πακέτων μέσα στο πλαίσιο του αποστολέα, το πλαίσιο του παραλήπτη, καθώς επίσης και τα χρώματά τους που αλλάζουν ανάλογα με την κατάσταση που βρίσκονται (εκπομπή, λήψη, σφάλμα μετάδοσης). Επίσης στην κλάση PacketN έχουμε σχεδιάσει έναν timer πάνω από κάθε πακέτο που μας δείχνει το χρόνο προθεσμίας που έχει το κάθε πακέτο μέχρι να επιβεβαιωθεί επιτυχώς από τον παραλήπτη. Τέλος στην ίδια κλάση εκτελείται η μεταβλητή reset η οποία αρχικοποιεί τις παραμέτρους λειτουργίας ώστε να εκτελεστεί η εφαρμογή προσομοίωσης από την αρχή.

```
public void DrawPacket(Graphics gfx)
        {
            //χρωματίζω  την επιφάνεια πλαισίου του αποστολέα
            // σχεδιάζω το περίγραμμα του πλαισίου του αποστολέα

            Rectangle recSender = new Rectangle(x, startY, width, senderYPSOS);
            gfx.FillRectangle(new SolidBrush(senderColor), recSender);
            gfx.DrawRectangle(new Pen(Color.Black), recSender); //το κουτι του
sender


          //ζωγραφίζω τον timer
            gfx.DrawString(timer.ToString(), new Font("Tahoma", 14), new
SolidBrush(Color.Black), x + (width / 2) - 7, startY - senderYPSOS);


            //Σχεδίαση του πλαισίου λήψης πακέτων στο δέκτη
            //χρωματίζω  την επιφάνεια πλαισίου του παραλήπτη

            gfx.DrawRectangle(new Pen(Color.Black), x, startY + travelLenght,
width, senderYPSOS);
            if (receiverColor != Color.Transparent) {
                gfx.FillRectangle(new SolidBrush(receiverColor), x + 1, startY +
travelLenght + 1, width - 1, senderYPSOS - 1);
            }
```

Σχήμα 4.2 : Απεικόνιση κελιών αποστολέα-παραλήπτη.Υλοποίηση και εμφάνιση timer μέσα στο κουτί του αποστολέα

```
  // Εδώ μπαίνει η αρίθμηση των πακέτων 1...2....3...4....5

            gfx.DrawString(No.ToString(), new Font("Tahoma", 11), new
SolidBrush(Color.Black), recSender);


            ////απο εδω και κάτω ζωγραφίζω το πακέτο. Μονο οταν το πακετο στελνετε
και δεν ειναι σκοτωμένο.

            if (stelnete && !killed && !ignore && y >= startY) {
                gfx.DrawRectangle(new Pen(Color.Black), x, y, width, packetYPSOS);


                {
                    gfx.FillRectangle(new SolidBrush(packetColor), x + 1, y + 1,
width - 1, packetYPSOS - 1);
                }
```

Σχήμα 4.3: Δημιουργία και αρίθμηση πακέτων

4.3.1 Reset

Το κουμπί reset επαναφέρει στις αρχικές ρυθμίσεις, έτσι μηδενίσαμε τις μεταβλητές που δείχνουν σε τι κατάσταση βρίσκεται το πακέτο κάθε φορά (εκπομπή ,λήψη, σφάλμα μετάδοσης, επιβεβαίωση κτλ.) και αρχικοποιήσαμε τα χρώματα αποστολέα, παραλήπτη και πακέτου.

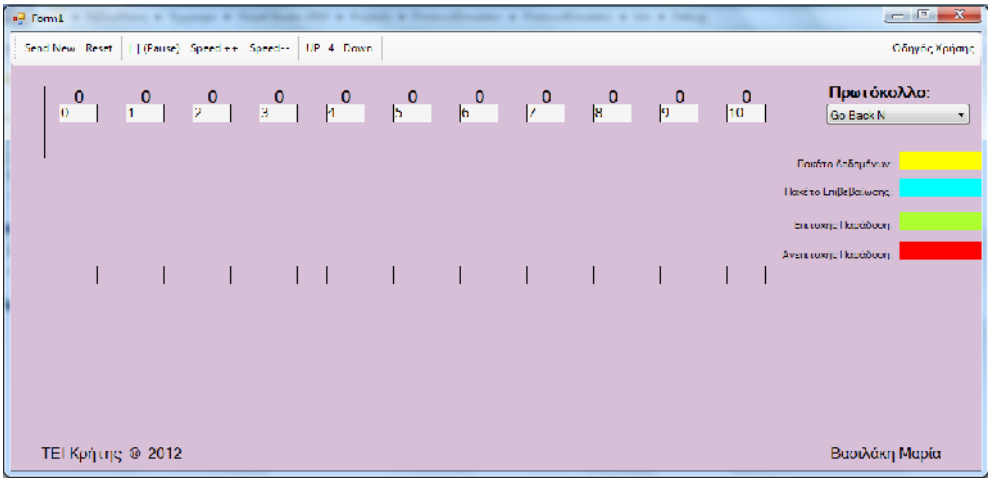
Ο κώδικας που υλοποιεί την λειτουργία του πλήκτρου reset :

```
      public void ResetPacket(bool resetReceiverColor) //μεταβλητή reset
      {

        //1)μηδενισμός όλων των boolean μεταβλητών
(stelnete,einaiack,eftase,killed ktl)
        //2)μηδενισμός του Y του πακέτου
        //3)αρχικοποίηση των χρωμμάτων (αποστολέα, παραλήπτη, πακέτου)
        stelnete = false;
        eftase = false;
        einaiACK = false;
        killed = false;
        epilegmeno = false;
        resend = false;
        ksekinise = false;
        done = false;
        spasimo = false;
        ignore = false;
        mustBeIgnored = false;
        y = startY; //αν βάλω y=0 ξεκινάει μετά το reset με χρονοκαθυστέρηση,
           // γιατί το πακέτο ξεκινάει από την κορυφή και όχι από τον sender
        timer = 0;
        stepCount = 0; // μηδενίζω τα βήματα που έχουν γίνει

   if (resetReceiverColor)
      receiverColor = Color.Transparent;//χρώμα εκκίνησης


      senderColor = Color.WhiteSmoke; //χρώμα εκκίνησης


      packetColor = Color.Yellow;//χρώμα εκκίνησης
      }
```

Σχήμα 4.4 : Κώδικας λειτουργίας Reset

4.4 Κλάση Form1

Στην κλάση Form1 δημιουργούμε την φόρμα μας.



Σχήμα 4.5 : Οθόνη εμφάνισης προσομοιωτή

Δημιουργούμε τα πλήκτρα της εφαρμογής, το "send new", το "reset",το "pause", το "speed++",το "speed--",το "up", το "down",το κουμπί οδηγός χρήσης (βοήθεια). Τοποθετήσαμε ένα υπομενού όπου μπορούμε να επιλέξουμε το πρωτόκολλο που θέλουμε GBN ,SRP, και ABP. Στο πρωτόκολλο SRP χρειάζεται επιπλέον ένας καταχωρητής κάτω από τον παραλήπτη. Επίσης σχεδιάζουμε ένα παράθυρο που πλαισιώνει τα πακέτα. Με τα πλήκτρα up και down μας δίνεται η δυνατότητα αυξομείωσης του τρέχοντος παραθύρου των πρωτοκόλλων GBN ,SRP, ABP. Το παράθυρο αυτό μας δείχνει πόσα πακέτα επιτρέπεται να στέλνουμε ανεπιβεβαίωτα. Το παράθυρο απεικονίζεται με ένα πλαίσιο που περιβάλλει τα εκάστοτε τελευταία ανεπιβεβαίωτα πακέτα.Πατώντας το Send new ο αποστολέας στέλνει πακέτα στον παραλήπτη και ξεκινάει ένας χρόνος προθεσμίας (timer) για κάθε πακέτο που στέλνεται. Το κουμπί pause σταματά προσωρινά την εκτέλεση της εφαρμογής μας . Με τα κουμπιά το speed++ ή το speed— μας δίνεται η δυνατότητα να επιταχύνουμε ή να επιβραδύνουμε την προσομοίωση του πρωτοκόλλου. Το κουμπί reset επαναφέρει στις αρχικές ρυθμίσεις. Έχουμε φτιάξει ράβδους με χρώματα και εξηγούμε ποιο χρώμα απεικονίζει την εξέλιξη του κάθε πακέτου. "Πακέτο Δεδομένων: Κίτρινο, "Πακέτο επιβεβαίωσης: Μπλε", "Επιτυχής παράδοση: Πράσινο" και "Ανεπιτυχής παράδοση: Κόκκινο".

4.4.1 Send new

Με το πλήκτρο "Send new" αρχίζει να αποστέλλεται κάθε πακέτο από τον αποστολέα στον παραλήπτη και ξεκινάει ο χρόνος προθεσμίας (timer) για κάθε πακέτο που στέλνεται. Ο κώδικας που υλοποιεί την λειτουργία του πλήκτρου "Send new"

```
private void btnSendNew_Click(object sender, EventArgs e) // sendnew
        {
             AddCount += 1;
            if (!simulationRunning) {
                RunSimulation();
            }
        }
          private int YpologismosOlokliromenwn()
        {
            int athristis = 0;
            for (int i = 0; i < paketa.Count; i++) {

                if ( paketa[i].done) // ολοκληρώθηκε
                    athristis++;
         }
            return athristis;
        }
           packetN CreatePacket()  //δημιουργώ και φτιάχνω ένα πακέτο

        {
            int X;
            X = kenoAristera + ((packetN.width + kenoSenders) * paketa.Count);
            if (X + packetN.width < this.Width) {
                packetN p = new packetN();
                p.x = X;
                p.No = paketa.Count;
                paketa.Add(p);

                if (!simulationRunning)
                    RunSimulation();
                    return p;
            }
            else {
                return null;
            }
        }

        private int YpologismosAnepivevaiotonPaketon()
        {
            int athristis = 0;
            for (int i = 0; i < paketa.Count; i++) {

                if (paketa[i].stelnete == true)
                    athristis++;

            }
            return athristis;
        }
```

Σχήμα 4.6: Κώδικας λειτουργίας Send new

Έχουμε έναν πίνακα με 11 στοιχεία (0-10) και το κάθε στοιχείο έχει τις πληροφορίες του πακέτου. Ο παραπάνω κώδικας πηγαίνει στην επόμενη θέση του πίνακα, δείχνει το επόμενο πακέτο και τρέχει την εφαρμογή. Ελέγχει πόσα πακέτα έχουν ολοκληρώσει την αποστολή τους, ζωγραφίζει τα πακέτα και υπολογίζει τα ανεπιβεβαίωτα πακέτα έτσι ώστε αν σταλεί ένα ανεπιβεβαίωτο πακέτο να αυξηθεί ο μετρητής.

4.4.2 Speed++ και speed—

Με τα κουμπιά το speed++ ή το speed— μας δίνεται η δυνατότητα να επιταχύνουμε ή να επιβραδύνουμε την προσομοίωση του πρωτοκόλλου και έτσι με αυτόν τον τρόπο μεγαλώνει ή μικραίνει την ταχύτητα που προχωράνε τα πακέτα.
Ο κώδικας για το speed++ και το speed— είναι:

```
private void button3_Click(object sender, EventArgs e) // speed ++
      {
          if (simulationSpeed > 30)
              simulationSpeed -= 30;
      }

private void button4_Click(object sender, EventArgs e) // speed --
      {
          simulationSpeed += 30;
      }
```

Σχήμα 4.7: Κώδικας λειτουργίας speed++ και το speed—

4.4.3 Up και down

Το κουμπι up δίνει τη δυνατότητα στο χρήστη να αυξάνει τον αριθμό των ανεπιβεβαίωτων πακέτων και έτσι αυξάνεται ο αριθμός που βρίσκεται ανάμεσα στα πλήκτρα up και down. Στο πρωτόκολλο SRP χρησιμοποιούμε έναν καταχωρητή που αποθηκεύει πακέτα εκτός σειράς, με το πλήκτρο up δίνεται η δυνατότητα στο χρήστη να αυξήσει τις θέσεις του καταχωρητή. Το αντίθετο γίνεται με το κουμπί down.
Ο κώδικας που υλοποιεί την λειτουργία του πλήκτρου up είναι:

```
        private void button1_Click(object sender, EventArgs e) //up
        {
            for (int i = 0; i < paketa.Count; i++) {
                if (paketa[i].ksekinise)
                    return;
            }
            int anepivev =paketa.Count- YpologismosEpivevaiotonPaketon()-1;

            if (pluthosPaketon <=anepivev-1 ) {
                int newPlythos;
                newPlythos = pluthosPaketon + 1;
                ChangePacketLeght(newPlythos);
            }

        }
           private void ChangePacketLeght(int packetLenght)
        {

            pluthosPaketon = packetLenght;
            lblPaketa.Text = pluthosPaketon.ToString();
            if (protocolo == Protocol.SRP) {
                buffer.Clear();
                for (int i = 0; i < pluthosPaketon; i++) {
                    buffer.Add(-1);
                }
            }

            this.Refresh(); //προκαλώ ανανέωση της οθόνης
        }
```

Σχήμα 4.8: Κώδικας λειτουργίας up

Ο κώδικας που υλοποιεί την λειτουργία του πλήκτρου down είναι:

```
private void button5_Click(object sender, EventArgs e) //down
        {

            for (int i = 0; i < paketa.Count; i++) {
                if (paketa[i].ksekinise)
                    return;
            }
            if (pluthosPaketon > 1) {
                int newPlythos;
                newPlythos = pluthosPaketon - 1;
                ChangePacketLeght(newPlythos);
            }

        }

        private void lblPaketa_Click(object sender, EventArgs e)
        {
```

Σχήμα 4.9: Κώδικας λειτουργίας down

4.4.4 Pause

Το κουμπί pause σταματά προσωρινά την εκτέλεση της εφαρμογή μας .

Ο κώδικας που υλοποιεί την λειτουργία του πλήκτρου pause είναι:

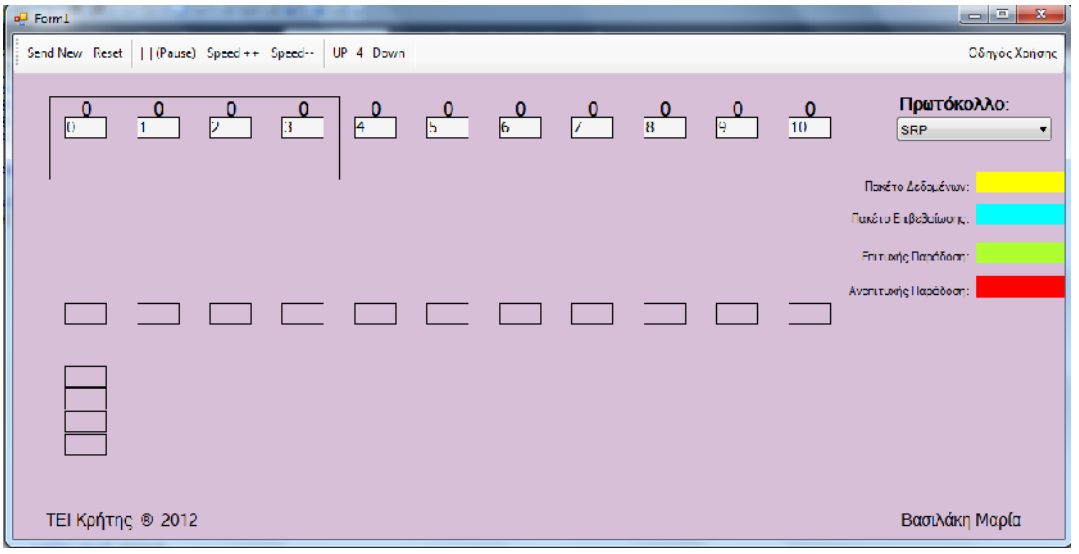
```
{
private void btnPause_Click(object sender, EventArgs e) // το κουμπί του pause

        {
            pause = !pause;
            if (pause) {
                btnPause.Text = "|>  (Play)";
            }
            else {
                btnPause.Text = "|  |  (Pause)";
            }
        }
```

Σχήμα 4.10: Κώδικας πλήκτρου pause

```
bool run;
            stopSimulation = false; // ξεκινάει
            run = true;
            while (run && !stopSimulation) {
                //εδώ εκτελείται ένα βήμα

                if (pause) {
                    Application.DoEvents();
                    continue;
                }

                run = false; // όταν πάταμε το Pause, στάματαει να τρέχει,
                lost = -1;
```

Σχήμα 4.11: Κώδικας λειτουργίας pause

4.4.5 Καταχωρητής παραλήπτη πρωτοκόλλου SRP



Σχήμα 4.12: Καταχωρητής-Παράθυρο αποστολέα

Ο παραλήπτης αποθηκεύει τα πακέτα που λαμβάνονται εκτός σειράς σε αυτόν τον καταχωρητή (σχήμα 4.12) ώστε να τα παραδώσει στη σωστή σειρά στα ανώτερα στρώματα . Όταν έρχεται το πρώτο πακέτο "0" ο παραλήπτης το αποθηκεύει στον καταχωρητή και στη συνέχεια το παραδίδει, το ίδιο κάνει και με τα επόμενα πακέτα όσο αυτά έρχονται κανονικά στη σειρά τους και δεν χάνονται. Αν χαθεί ένα πακέτο ο παραλήπτης αποθηκεύει τα επόμενα πακέτα στον καταχωρητή περιμένοντας να έρθει και το χαμένο πακέτο ώστε να τα αναδιατάξει και να παραδώσει με τη σωστή σειρά. Ο καταχωρητής εξαρτάται από τα πλήκτρα up και down και αυξομειώνεται ανάλογα με το πόσα ανεπιβεβαίωτα πακέτα υπάρχουν, μεταφέρεται κάτω από τον παραλήπτη την ώρα ακριβώς που φτάνει ένα πακέτο.

```
  if (protocolo == Protocol.SRP) {
                    //ζωγραφίζω τον buffer
                    if (buffer.Count > 0) {
                        int yBuffer = packetN.travelLenght + packetN.senderYPSOS + 120;
//από εδώ ξεκινάει ο buffer
                        Pen pen = new Pen(Color.Black);
                        Font bufferFont = new Font("Tahoma", 11);
                        Brush brBlack = (Brush)new SolidBrush(Color.Black);
                        RectangleF rectBox;
                        StringFormat strF=new StringFormat();
                        strF.Alignment = StringAlignment.Center;

                        int x=bufferLeft;
                        for (int i = 0; i < buffer.Count; i++) {
                            rectBox = new RectangleF((float) x,(float) yBuffer,(float)
packetN.width , 20.0f); // το πλάτος και το ύψος του buffer
                            e.Graphics.DrawRectangle(pen, Rectangle.Round( rectBox));
                            if (buffer[i] > -1) {
                                e.Graphics.DrawString(buffer[i].ToString(), bufferFont,
brBlack, rectBox, strF);
                            }
                            yBuffer += 22;
                        }
                    }
```

Σχήμα 4.13: Κώδικας δημιουργίας Καταχωρητή

4.4.6 Παράθυρο ανεπιβεβαίωτων πακέτων

Στο παραπάνω σχήμα απεικονίζεται το παράθυρο που πλαισιώνει τα πακέτα, ο κώδικας του είναι ο εξής:

```
//εδώ φτιάχνω το παραλληλόγραμμο (πλαίσιο)


                int epiveveomena; //υπολογίζω πόσα πακέτα στέλνονται ακόμα.
                //έτσι ξέρω από που θα αρχίσει το παραλληλόγραμμο. Θα τελείωσει από
εκεί που ξεκινάει + plithosPaketon
                epiveveomena = YpologismosEpivevaiotonPaketon();

                int paketa_apomenoun = paketa.Count - epiveveomena;


                if (pluthosPaketon > paketa_apomenoun && pluthosPaketon>1 ){
//περιορίζω το μήκος του πλαισίου ανάλογα με το πλήθος των πακέτων

                }

                int startPlaisio;
                int mikosPlaisio = pluthosPaketon * (packetN.width + kenoSenders) -
(kenoSenders / 2); //όλο το πλαίσιο

                if (paketa_apomenoun >= pluthosPaketon) {
                    startPlaisio= kenoAristera + (epiveveomena * packetN.width) +
((epiveveomena - 1) * kenoSenders) + (kenoSenders / 2); // από που ξεκινάει (κενό
ανάμεσα στον αποστολέα -1)
                }
                else {
                    startPlaisio = kenoAristera + ((paketa.Count - pluthosPaketon) *
packetN.width) + (((paketa.Count - pluthosPaketon) - 1) * kenoSenders) + (kenoSenders
/ 2); // από που ξεκινάει (κενό ανάμεσα στον αποστολέα -1)
                }


                if (pluthosPaketon == paketa_apomenoun) {
                    e.Graphics.DrawRectangle(new Pen(Color.Black), startPlaisio,
packetN.startY - 20, mikosPlaisio + (kenoSenders * 2), packetN.startY);
                }
                else {
                    e.Graphics.DrawRectangle(new Pen(Color.Black), startPlaisio,
packetN.startY - 20, mikosPlaisio + (kenoSenders / 2), packetN.startY);
                }
```

Σχήμα 4.14: Κώδικας λειτουργίας παράθυρου

Αρχική τιμή παράθυρου = 1. Σύνολο πακέτων προς αποστολή = 11

Έχουμε εισάγει μια σειρά από περιορισμούς στην αυξομείωση του παραθύρου ώστε σε όποιο σημείο της προσομοίωσης και να θέλουμε να αυξομειώσουμε το παράθυρο να είναι δυνατό να δούμε την ολίσθηση κατά μια θέση δεξιά.

Γενικά το πλαίσιο μπορεί να αλλάξει, μόνο όταν έχει ολοκληρωθεί επιτυχώς η λήψη των ήδη απεσταλμένων πακέτων. Έστω ότι έχουν αποσταλεί 5 πακέτα και το παράθυρο έχει τιμή = 3 , σημαίνει ότι το σχετικό πλαίσιο αφορά τα πακέτα (5, 6 και 7). Εάν θελήσουμε να αυξήσουμε το παράθυρο πριν την ενεργοποίηση αποστολής νέου πακέτου δεδομένων (Send new) μπορούμε να το αυξήσουμε μέχρι 5 πακέτα δηλαδή να περιλαμβάνει τα πακέτα (5, 6, 7, 8 και 9). Κατά αυτόν τον τρόπο με την επιτυχής αποστολής του πακέτου 5, το παράθυρο θα ολισθήσει μια θέση δεξιά να συμπεριλάβει και το πακέτο 10

Κεφάλαιο 5

Συμπεράσματα

Η παρούσα πτυχιακή εργασία ανατέθηκε με σκοπό να δημιουργηθεί μια εφαρμογή που θα μπορούσε να βοηθήσει στην όσο το δυνατό πιο πιστή αναπαράσταση της λειτουργίας των πρωτοκόλλων του επιπέδου ζεύξης στην δικτυακή αρχιτεκτονική OSI. Πρόκειται για μια σειρά από πρωτόκολλα που σκοπό έχουν να διασφαλίσουν την αξιοπιστία στην ροή της πληροφορίας στο $2^{ο}$ επίπεδο.

Πιστεύουμε ότι το τελικό αποτέλεσμα συνεισφέρει ουσιαστικά στην επεξήγηση του μηχανισμού λειτουργίας των πρωτοκόλλων αυτών, από το πιο απλό ABP μέχρι το πιο σύνθετο SRP με χρήση προσωρινού καταχωρητή σε αποστολέα και παραλήπτη.

Η προσπάθεια ξεκίνησε με βάση μια εργασία σε γλώσσα προγραμματισμού Java που ανατρέξαμε στο πλαίσιο του μαθήματος και ολοκληρώθηκε με ουσιαστικές επεμβάσεις σε κώδικα που δημιουργήθηκε από την αρχή σε γλώσσα C# στην οποία υπάρχει κατάλληλη προπτυχιακή εκπαίδευση και πρότερη εμπειρία.

Μεταξύ των ουσιαστικών παρεμβάσεων προς όφελος μιας καλύτερης και ορθότερης αναπαράστασης της λειτουργίας, έγιναν οι ακόλουθες παρεμβάσεις:

- Δυνατότητα αυξομείωσης του ενεργού παραθύρου πακέτων δεδομένων
- Προσομοίωση απώλειας πακέτου δεδομένων ή επιβεβαίωσης με μια κίνηση
- Εμφάνιση χρόνου προθεσμίας (timer) που συμβαδίζει με την επανεκπομπή πακέτου που δεν έχει επιβεβαιωθεί εγκαίρως.
- Χρωματικές αναπαραστάσεις των διαφόρων καταστάσεων, λήψη, απώλεια πακέτου με συνέπεια ως προς τα πιθανά σενάρια εξέλιξης της προσομοιωμένης λειτουργίας των πρωτοκόλλων.
- Κατά την αναπαράσταση επανεκπομπής σειράς πακέτων λαμβάνεται πρόνοια να μην απεικονίζονται ταυτόχρονα όλες αυτές οι επανεκπομπές αλλά να υπάρχει μικρή χρονική διαφορά που αναπαριστάται από μικρή απόσταση στην διαδρομή μεταξύ αποστολέα και παραλήπτη.

Κεφάλαιο 6

Βιβλιογραφία

Βιβλία

[1] «Δίκτυα Επικοινώνιων», Jean Walrand, Εκδόσεις Παπασωτηρίου 1997

Ιστότοποι

[2] «C#.NET»

http://videogameslab.wordpress.com/2009/01/08/intro-csharp-net/

[3] «Microsoft visual»

http://www.microsoft.com/business/smb/el-gr/servers-and-tools/visual-studio-pro.mspx

[4] «Παρουσίαση της λειτουργίας του πρωτοκόλλου Go-Back N»

http://www.eecis.udel.edu/~amer/450/TransportApplets/GBN/GBNindex.html

[5] «Κώδικας Java»

http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/go-back-n/go-back-n.html