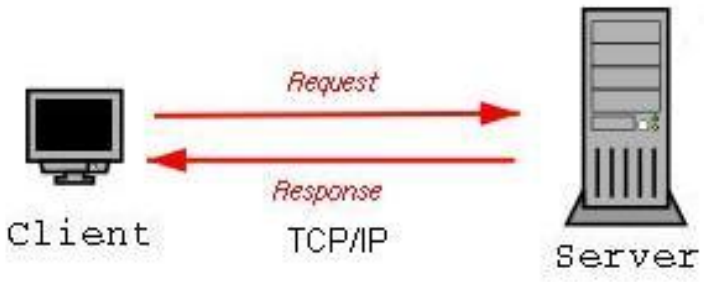
ΤΕΙ ΚΡΗΤΗΣ - ΠΑΡΑΡΤΗΜΑ ΧΑΝΙΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

''Το πρωτόκολλο TCP/IP σε Client Server εφαρμογές''



Εισηγητής : **Εμμανουήλ Αντωνιδάκης**

Σπουδαστής : **Φιλήμονας Παπαδιού**

Περιεχόμενα:

σελ.

Κεφάλαιο 1ο

’’Εισαγωγή’’

1.1 Η ιστορία του διαδικτύου και η δημιουργία των πρωτοκόλλων TCP και IP.

Την δεκαετία του 60’ στα πανεπιστήμια των ΗΠΑ οι ερευνητές ξεκινούν να πειραματίζονται με τη διασύνδεση απομακρυσμένων υπολογιστών μεταξύ τους. Το δίκτυο ARPANET γεννιέται το 1969 με πόρους του προγράμματος ARPA (Advanced Research Project Agency) του Υπουργείου Άμυνας, με σκοπό να συνδέσει το Υπουργείο με στρατιωτικούς ερευνητικούς οργανισμούς και να αποτελέσει ένα πείραμα για τη μελέτη της αξιόπιστης λειτουργίας των δικτύων. Στην αρχική του μορφή, το πρόγραμμα απέβλεπε στον πειραματισμό με μια νέα τεχνολογία γνωστή σαν μεταγωγή πακέτων (packet switching), σύμφωνα με την οποία τα προς μετάδοση δεδομένα κόβονται σε πακέτα και πολλοί χρήστες μπορούν να μοιραστούν την ίδια επικοινωνιακή γραμμή.

Στόχος ήταν η δημιουργία ενός διαδικτύου που θα εξασφάλιζε την επικοινωνία μεταξύ απομακρυσμένων δικτύων, έστω και αν κάποια από τα ενδιάμεσα συστήματα βρίσκονταν προσωρινά εκτός λειτουργίας. Κάθε πακέτο θα είχε την πληροφορία που χρειάζονταν για να φτάσει στον προορισμό του, όπου και θα γινόταν η επανασύνθεσή του σε δεδομένα τα οποία μπορούσε να χρησιμοποιήσει ο τελικός χρήστης.

Το παραπάνω σύστημα θα επέτρεπε σε υπολογιστές να μοιράζονται δεδομένα και σε ερευνητές να υλοποιήσουν το ηλεκτρονικό ταχυδρομείο.

Την δεκαετία του 70’ και συγκεκριμένα το 1973, ξεκινά ένα νέο ερευνητικό πρόγραμμα που ονομάζεται Internetting Project (Πρόγραμμα Διαδικτύωσης) προκειμένου να ξεπεραστούν οι διαφορετικοί τρόποι που χρησιμοποιεί κάθε δίκτυο για να διακινεί τα δεδομένα του. Στόχος είναι η διασύνδεση πιθανώς ανόμοιων δικτύων και η ομοιόμορφη διακίνηση δεδομένων από το ένα δίκτυο στο άλλο. Από την έρευνα γεννιέται μια νέα τεχνική, το Internet Protocol (**IP**) (Πρωτόκολλο Διαδικτύωσης). Διαφορετικά δίκτυα που χρησιμοποιούν το κοινό πρωτόκολλο IP μπορούν να συνδέονται και να αποτελούν ένα διαδίκτυο. Σε ένα δίκτυο IP όλοι οι υπολογιστές είναι ισοδύναμοι, οπότε τελικά οποιοσδήποτε υπολογιστής του διαδικτύου μπορεί να επικοινωνεί με οποιονδήποτε άλλον.

Επίσης, σχεδιάζεται μια άλλη τεχνική για τον έλεγχο της μετάδοσης των δεδομένων, το Transmission Control Protocol (**TCP**) (Πρωτόκολλο Ελέγχου Μετάδοσης). Ορίζονται προδιαγραφές για τη μεταφορά αρχείων μεταξύ υπολογιστών (FTP) και για το ηλεκτρονικό ταχυδρομείο (E-mail). Σταδιακά συνδέονται με το ARPANET ιδρύματα από άλλες χώρες, με πρώτα το University College of London (Αγγλία) και το Royal Radar Establishment (Νορβηγία).

Το 1983, το πρωτόκολλο **TCP/IP** (δηλ. ο συνδυασμός των TCP και IP) αναγνωρίζεται ως πρότυπο από το Υπουργείο Άμυνας των ΗΠΑ. Η έκδοση του λειτουργικού συστήματος Berkeley UNIX το οποίο περιλαμβάνει το TCP/IP συντελεί στη γρήγορη εξάπλωση της διαδικτύωσης των υπολογιστών. Εκατοντάδες Πανεπιστήμια συνδέουν τους υπολογιστές τους στο ARPANET, το οποίο επιβαρύνεται πολύ και το 1983, χωρίζεται σε δύο τμήματα: στο MILNET (για στρατιωτικές επικοινωνίες) και στο νέο ARPANET (για χρήση αποκλειστικά από την πανεπιστημιακή κοινότητα και συνέχιση της έρευνας στη δικτύωση).

Το 1985, το National Science Foundation (NSF) δημιουργεί ένα δικό του γρήγορο δίκτυο, το NSFNET χρησιμοποιώντας το πρωτόκολλο TCP/IP, προκειμένου να συνδέσει πέντε κέντρα υπολογιστών μεταξύ τους και με την υπόλοιπη επιστημονική κοινότητα. Στα τέλη της δεκαετίας του ‘80, όλο και περισσότερες χώρες συνδέονται στο NSFNET. Χιλιάδες πανεπιστήμια και οργανισμοί δημιουργούν τα δικά τους δίκτυα και τα συνδέουν πάνω στο παγκόσμιο αυτό δίκτυο το οποίο αρχίζει να γίνεται γνωστό σαν INTERNET και να εξαπλώνεται με τρομερούς ρυθμούς σε ολόκληρο τον κόσμο. Το 1990, το ARPANET πλέον καταργείται.

Κεφάλαιο 2ο

’’ Το πρωτόκολλο TCP/IP ’’

2.1 Γενικά

Ο όρος TCP/IP χρησιμοποιείται σήμερα για να περιγράψει ένα σύνολο από διαφορετικές έννοιες. Η συνήθεις χρήση του όρου όμως είναι όταν χρησιμοποιείται για να περιγράψει το πρωτόκολλο του διαδικτύου. Παρόλο που αναφέρεται σαν μια οντότητα στην πραγματικότητα πρόκειται για ένα σύνολο πρωτοκόλλων. Η ανάλυση του TCP/IP σαν ένα σύνολο πρωτοκόλλων θα παρουσιαστεί σε αυτό το κεφάλαιο.

2.2 Σχέση OSI και TCP/IP

Το μοντέλο OSI είναι μια αρχιτεκτονική δικτύου με επτά επίπεδα που περιγράφει όλα τα θέματα που αφορούν στην επικοινωνία δικτύου.
Το πρωτόκολλο TCP/IP δεν έρχεται σε σύγκρουση με το μοντέλο OSI. Υπάρχουν όμως μερικές ουσιαστικές διαφορές μεταξύ τους όπως φαίνεται στο παρακάτω σχήμα

<table>
<tr><th>OSI</th><th>TCP/IP</th></tr>
<tr><td>Επίπεδο εφαρμογής</td><td rowspan="3">Επίπεδο εφαρμογής</td></tr>
<tr><td>Επίπεδο παρουσίασης</td></tr>
<tr><td>Επίπεδο συνόδου</td></tr>
<tr><td>Επίπεδο μεταφοράς</td><td>Επίπεδο μεταφοράς (TCP)</td></tr>
<tr><td>Επίπεδο δικτύου</td><td>Επίπεδο δικτύου (IP)</td></tr>
<tr><td>Επίπεδο σύνδεσης δεδομένων</td><td rowspan="2">Επίπεδο πρόσβασης δικτύου</td></tr>
<tr><td>Φυσικό επίπεδο</td></tr>
</table>

Σχήμα 1. Σχέση OSI και TCP/IP

Στο μοντέλο OSI, το επίπεδο εφαρμογής παρέχει στους χρήστες πρόσβαση στις υπηρεσίες δικτύου, το επίπεδο παρουσίασης φροντίζει για την κατάλληλη αναπαράσταση των δεδομένων, το επίπεδο συνόδου ελέγχει τη διαδικασία της επικοινωνίας, το επίπεδο μεταφοράς φροντίζει για την αξιόπιστη μεταφορά των δεδομένων, το επίπεδο δικτύου απομονώνει τα υψηλότερα στρώματα και φροντίζει για τη μεταφορά των δεδομένων στον προορισμό τους, το επίπεδο σύνδεσης δεδομένων εξασφαλίζει την αξιόπιστη μεταφορά

πληροφορίας στη φυσική γραμμή σύνδεσης μεταδίδοντας πλαίσια με τον κατάλληλο συγχρονισμό, έλεγχο λαθών και έλεγχο ροής και τέλος το φυσικό επίπεδο με θέματα καλωδίωσης και φυσικής μεταφοράς των δεδομένων.
Στο TCP/IP στο επίπεδο εφαρμογής που είναι πάνω από τα πρωτοκολλά TCP και IP βρίσκονται οι υπηρεσίες και τα πρωτόκολλα εφαρμογής. Αυτά είναι δομημένα με τέτοιο τρόπο ώστε να χρησιμοποιούν για την επικοινωνία τα πρωτόκολλα TCP και IP αλλά και κάποια επιμέρους πρωτόκολλα που θα δούμε παρακάτω.

Το επίπεδο εφαρμογής παρέχει εφαρμογές που χρησιμοποιούν τα πρωτόκολλα του επιπέδου μεταφοράς και αντιπροσωπεύει το σημείο επαφής του χρήστη με το TCP/IP.

Το επίπεδο μεταφοράς υλοποιεί τις συνδέσεις μεταξύ των υπολογιστών του δικτύου χρησιμοποιώντας ως βασικό πρωτόκολλο το TCP.

Το επίπεδο δικτύου είναι υπεύθυνο για την μετάδοση στο φυσικό δίκτυο των πακέτων που δημιουργήθηκαν από τα πρωτόκολλα του επιπέδου μεταφοράς. Το βασικό πρωτόκολλο του επιπέδου αυτού είναι το IP και το οποίο εξασφαλίζει την διασυνδεσιμότητα.

Το επίπεδο πρόσβασης παρέχει την πρόσβαση στο φυσικό δίκτυο στο οποίο μεταδίδεται η πληροφορία με την μορφή πακέτων και αντιπροσωπεύει το χαμηλότερο επίπεδο λογικό επίπεδο. Το επίπεδο αυτό περιλαμβάνει τα στοιχεία φυσικών συνδέσεων όπως καλώδια, κάρτες δικτύου κ.α.

2.3 Επιμέρους πρωτοκολλά του TCP/IP

Στον παρακάτω πίνακα φαίνονται εκτός από τα TCP και IP και μερικά από τα επιμέρους πρωτόκολλα του πρωτοκόλλου TCP/IP στο επίπεδο που ανήκει το καθένα.

	Εφαρμογές	
Επίπεδο εφαρμογής	(Telnet, FTP, SMTP)	TFTP
Επίπεδο μεταφοράς	TCP	UDP
Επίπεδο δικτύου	IP / ICMP	

Σχήμα 2. Πρωτόκολλα του TCP/IP

Τα πρωτόκολλα απομακρυσμένης σύνδεσης Telnet (Telecommunications Network) μεταφοράς αρχείων FTP (File Transfer Protocol) και μεταφοράς απλού ταχυδρομείου SMTP (Simple Mail Transfer Protocol) χρησιμοποιούν το TCP ενώ

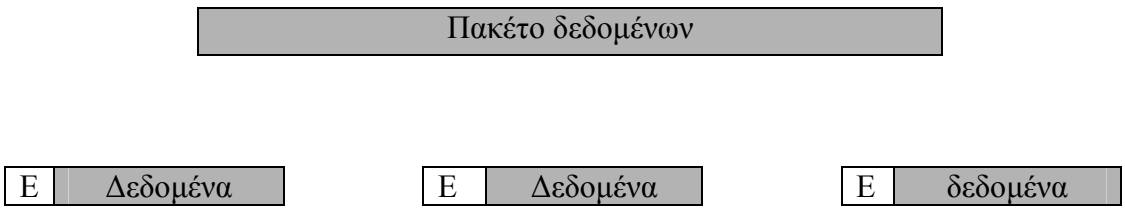
άλλα όπως η απλή μεταφορά αρχείων TFTP (Trivial File Transfer Protocol) χρησιμοποιούν το πρωτόκολλο UDP.
Εκτός από τα παραπάνω υπάρχουν το πρωτόκολλο Μηνύματος Ελέγχου Διαδικτύου ICMP (Internet Control Message Protocol) το οποίο λειτουργεί χωριστά στο επίπεδο δικτύου και να αναλαμβάνει να αναφέρει προβλήματα και ασυνήθιστες καταστάσεις που σχετίζονται με το πρωτόκολλο IP και το πρωτόκολλο μετατροπής διευθύνσεων ARP (Address Resolution Protocol) για το οποίο θα αναφερθούν περισσότερα στην ανάλυση του IP.

2.3.1 Το πρωτόκολλο UDP

Το πρωτόκολλο UDP χρησιμοποιείται σε περιπτώσεις που δεν χρειαζόμαστε την πολυπλοκότητα και την αξιόπιστη μετάδοση του TCP.
Κυρίως χρησιμοποιείται σε εφαρμογές στις οποίες σημαντικός παράγων είναι η ταχύτητα και όχι η επαναμετάδοση των δεδομένων , λειτουργία που εξασφαλίζει το TCP, όπως στο χαρακτηριστικό παράδειγμα της μετάδοσης φωνής. Το UDP δεν εκτελεί της λειτουργίες του TCP. Η επικοινωνία του με τα προγράμματα εφαρμογής γίνεται μέσω θυρών επικοινωνίας που ονομάζονται UDP θύρες (UDP ports).

2.4 Πρωτόκολλο TCP

Το πρωτόκολλο ελέγχου μετάδοσης δεδομένων TCP είναι το βασικό πρωτόκολλο του επιπέδου μεταφοράς. Το πρωτόκολλο TCP λαμβάνει από τα πρωτόκολλα ανωτέρου επιπέδου τα προς μετάδοση δεδομένα και τα μεταδίδει, μόνο όταν συμπληρωθεί πακέτο με μέγεθος ίσο με αυτό που έχει συμφωνηθεί κατά την εγκατάσταση της σύνδεσης.
Αντίστοιχα όταν το TCP λαμβάνει μηνύματα με μέγεθος μεγαλύτερο από το συμφωνηθέν το σπάει σε μικρότερα. Καθένα από αυτά τα πακέτα ονομάζεται TCP τμήμα (segment) η μορφή των οποίων φαίνεται στο σχήμα 3.



Σχήμα 3. Διάσπαση δεδομένων σε TCP τμήματα

Όταν τα TCP τμημάτων φθάσουν στον προορισμό τους το πρωτόκολλο είναι υπεύθυνο να τα τοποθετήσει στη σωστή σειρά και να τα επανασυνδέσει έτσι ώστε να σχηματίσουν και πάλι το αρχικό πακέτο. Ο προσδιορισμός της σειράς των τμημάτων γίνεται με βάση το πεδίο της επικεφαλίδας το οποίο ονομάζεται *αριθμός σειράς* και προσδιορίζει τη θέση του τμήματος μέσα στο αρχικό πακέτο.
Σε περίπτωση που έχουμε κάποιο σφάλμα στην μετάδοση και ένα τμήμα δεν φτάσει στον προορισμό του τότε το πρωτόκολλο είναι υπεύθυνο για την επαναμετάδοση του. Για να πραγματοποιηθεί η μετάδοση των τμημάτων το TCP τα διαβιβάζει στο πρωτόκολλο δικτύου IP.
Προκειμένου να εξασφαλίσουμε ότι ένα τμήμα έφθασε στον προορισμό του ο παραλήπτης πρέπει να στείλει πίσω επιβεβαίωση. Η λειτουργία αυτή πραγματοποιείται ως εξής : όταν ο παραλήπτης πρέπει να στείλει ένα τμήμα στον αποστολέα τοποθετεί ένα πεδίο της επικεφαλίδας του τμήματος έναν αριθμό που δηλώνει ότι τα δεδομένα μέχρι και αυτόν τον αριθμό έχουν φτάσει σωστά στον παραλήπτη. Το πεδίο αυτό ονομάζεται *Αριθμός Επιβεβαίωσης.*

Άλλη λειτουργία που εκτελεί το πρωτόκολλο TCP είναι ο έλεγχος της ποσότητας δεδομένων που μπορούν να μεταδίδονται κάθε φορά. Η λειτουργία αυτή ονομάζεται *Έλεγχος Ροής* και πραγματοποιείται με πεδίο που βρίσκεται στη επικεφαλίδα του τμήματος και ονομάζεται *Παράθυρο*.

2.4.1 TCP θύρες (TCP ports)

Το πρωτόκολλο TCP πρέπει να παραδίδει τα πακέτα στις εφαρμογές στις οποίες κατευθύνονται. Τα TCP πρέπει να γνωρίζει σύνδεση ανήκει κάθε τμήμα. Η πληροφορία που χρειάζεται για να πραγματοποιηθεί η αποπολύπλεξη των τμημάτων βρίσκεται στην επικεφαλίδα τους. Έτσι για να μπορέσει το TCP να συσχετίσει τα διάφορα τμήματα με τις συνδέσεις για τις οποίες προορίζονται χρησιμοποιεί τις *TCP θύρες (TCP ports).* Οι TCP θύρες είναι αφηρημένα σημεία επικοινωνίας που η καθεμιά αντιπροσωπεύεται από ένα αριθμό 16 bit και αποτελούν πεδία των επικεφαλίδων των τμημάτων.
Κάθε φορά που πραγματοποιείτε μια νέα σύνδεση δημιουργούνται οι TCP θύρες πηγής και προορισμού που γίνονται γνωστά και στα δυο άκρα τις σύνδεσης. Για συγκεκριμένες εφαρμογές οι θύρες είναι προκαθορισμένες , για παράδειγμα η εφαρμογή μεταφοράς αρχείων FTP χρησιμοποιεί την θύρα 21.

2.4.2 TCP συνδέσεις

Οι TCP συνδέσεις εγκαθίστανται από το πρωτόκολλο και χρησιμοποιούνται για να συνδεθούν δυο σημεία. Κάθε σύνδεση περιγράφεται πλήρως από τέσσερις αριθμούς. Της IP διευθύνσεις της πηγής και του προορισμού και τις TCP θύρες κάθε άκρου. Αυτοί είναι οι αριθμοί που βοηθούν το TCP να προσδιορίσει ποια τμήματα ανήκουν σε κάθε σύνδεση.

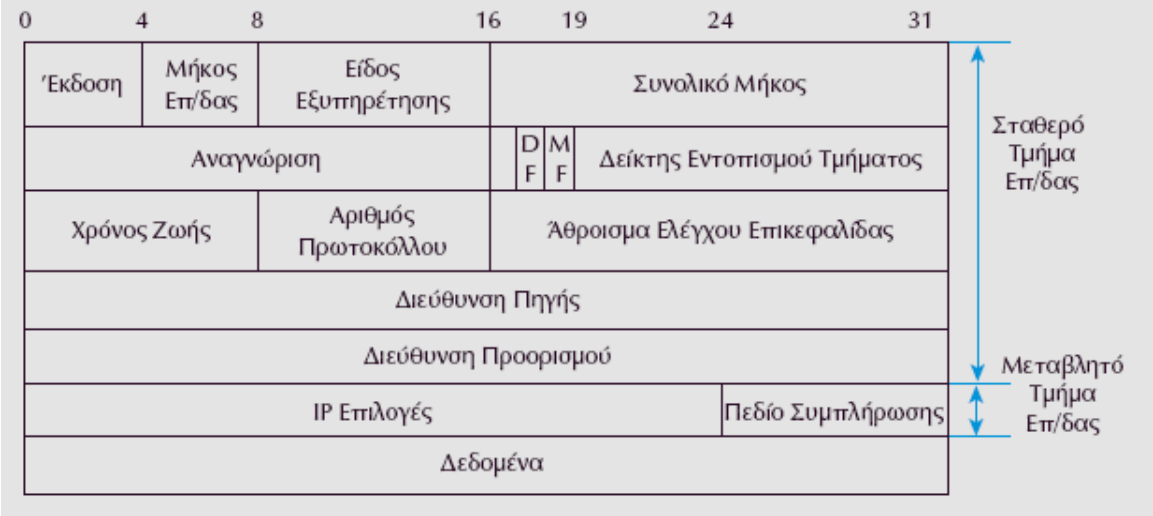
2.5 Το πρωτόκολλο IP

Το πρωτόκολλο IP είναι το βασικό πρωτόκολλο του επιπέδου δικτύου. Η λειτουργία του βασίζεται στη ιδέα των *αυτοδύναμων πακέτων (datagrams)* τα οποία μεταφέρονται ανεξάρτητα το ένα με το άλλο χωρίς να εξασφαλίζεται η αξιόπιστη μετάδοση τους.

Κάθε φορά που το πρωτόκολλο TCP ή UDP θέλει να μεταδώσει ένα πακέτο το προωθεί στο πρωτόκολλο IP προσδιορίζοντας τη διεύθυνση προορισμού. Αυτή η διεύθυνση αποτελεί και το μόνο στοιχείο που χρειάζεται το πρωτόκολλο IP.

Κάθε φορά που το IP λαμβάνει ένα TCP η UDP τμήμα προσθέτει και αυτό την δική του επικεφαλίδα και σχηματίζει ένα αυτοδύναμο πακέτο το οποίο έχει καθοριστεί να μην έχει μεγαλύτερο μέγεθος από 64 Kbytes.
Από την στιγμή που έχει σχηματιστεί ένα αυτοδύναμο πακέτο το IP αναλαμβάνει να βρει την κατάλληλη διαδρομή και να το προωθήσει στον προορισμό του.

Η μορφή του αυτοδύναμου πακέτου IP φαίνεται στο παρακάτω σχήμα.



Σχήμα 4. IP αυτοδύναμο πακέτο (datagram)

Από την στιγμή που προσδιορισθεί η διαδρομή του αυτοδύναμου πακέτου μεταδίδεται μέσω τον φυσικών δικτύων.

Τα φυσικά δίκτυα υπάρχει ενδεχόμενο να μην χρησιμοποιούν μέγιστο μήκος μονάδας μεταφοράς διαφορετικό από αυτό των IP αυτοδύναμων πακέτων.
Για να αντιμετωπιστεί ένα τέτοιο ενδεχόμενο το πρωτόκολλο IP έχει την δυνατότητα να διασπά τα αυτοδύναμα πακέτα σε μικρότερα πακέτα τα λεγόμενα *κομμάτια (fragments)* . Τα κομμάτια αυτά όταν φτάσουν στον προορισμό του ανασυντίθενται και σχηματίζουν το αρχικό IP αυτοδύναμο πακέτο.

2.5.1. Ανάλυση IP αυτοδύναμου πακέτου

Προκειμένου το πρωτόκολλο IP του υπολογιστή προορισμού να προσδιορίσει σε ποιο αυτοδύναμο πακέτο ανήκει το κάθε κομμάτι χρησιμοποίει το πεδίο *Αναγνώριση* της IP επικεφαλίδας. Τα πακέτα που έχουν την ίδια τιμή σε αυτό το πεδίο ανήκουν στο ίδιο αυτοδύναμο πακέτο.
Για να καταλάβει το πρωτόκολλο IP εάν ένα κομμάτι είναι ξεχωριστό ή ανήκει σε ένα μεγαλύτερο αυτοδύναμο πακέτο χρησιμοποιείται το πεδίο *ύπαρξης περισσοτέρων κομματιών (More Fragment).* Εάν το πεδίο τεθεί σε τιμή 1 σημαίνει πως το αυτοδύναμο πακέτο έχει διασπαστεί σε περισσότερα κομμάτια. Όλα τα κομμάτια του αυτοδύναμου πακέτου θέτουν αυτό το πεδίο σε 1 εκτός από το τελευταίο.
Σε περίπτωση που ο υπολογιστής προορισμού δεν μπορεί να ανασυνθέσει ένα διασπασμένο αυτοδύναμο πακέτο τότε θέτει το πεδίο *Απαγόρευσης διάσπασης (Don't Fragment)* στην τιμή 1.

Για να εντοπιστεί η θέση του κάθε κομματιού μέσα στο αυτοδύναμο πακέτο χρησιμοποιείται το πεδίο *Δείκτης εντοπισμού τμήματος*. Το πεδίο αυτό προσδιορίζει σε πιο σημείο του αρχικού πακέτου ανήκει το συγκεκριμένο κομμάτι.

Το πεδίο *Διεύθυνση πηγής* προσδιορίζει την IP διεύθυνση του υπολογιστή που στέλνει το πακέτο.

Το πεδίο *Διεύθυνση προορισμού* προσδιορίζει την IP διεύθυνση του υπολογιστή που πρέπει να παραδοθεί το πακέτο.

Το πεδίο *Αριθμός Πρωτοκόλλου* πληροφορεί το πρωτόκολλο IP στον υπολογιστή προορισμού σε ποιο πρωτόκολλο υψηλότερου επιπέδου θα παραδοθεί το πακέτο (TCP ή UDP).

Το πεδίο *Άθροισμα ελέγχου* βοήθα το πρωτόκολλο IP του υπολογιστή προορισμού να ελέγξει την ορθότητα της επικεφαλίδας του αυτοδύναμου πακέτου.

Το πεδίο *Έκδοση* χρησιμοποιείται για να προσδιορίσει την έκδοση του πρωτοκόλλου IP στην οποία ανήκει το αυτοδύναμο πακέτο.

Το πεδίο *Μήκος επικεφαλίδας* δηλώνει το μήκος της επικεφαλίδας σε λέξεις των 32 bits. Επειδή το μεταβλητό μήκος της επικεφαλίδας δεν έχει απαραίτητα μήκος πολλαπλάσιο των 32 bits χρησιμοποιείται το πεδίο *Συμπλήρωσης* έτσι ώστε το μήκος της επικεφαλίδας να είναι πολλαπλάσιο των 32 bits.

Το πεδίο Σ*υνολικό μήκος* δίνει το μήκος όλου του αυτοδύναμου πακέτου (επικεφαλίδας και δεδομένων). Το συνολικό μήκος είναι 64 Kbyte.
Εάν το αυτοδύναμο πακέτο έχει διασπαστεί τότε το πεδίο αυτό δείχνει το μήκος του συγκεκριμένου κομματιού.

Το πεδίο *Είδος εξυπηρέτησης* χρησιμοποιείται για να δηλώσει ο υπολογιστής τι είδους εξυπηρέτηση ζητάει από το δίκτυο.

Το πεδίο *IP επιλογές* για ειδικές λειτουργίες του πρωτοκόλλου.

Το πεδίο *Χρόνος ζωής* είναι ένας μετρητής που χρησιμοποιείται για να προσδιορίσει το χρόνο ζωής των αυτοδύναμων πακέτων.
Κάθε φορά που το πακέτο περνά από δρομολογητή το πεδίο μειώνεται τουλάχιστον κατά ένα. Όταν το πεδίο αυτό πάρει την τιμή 0 τότε το πακέτο απορρίπτεται.

2.6 IP διευθύνσεις

Το πρωτόκολλο TCP/IP χρησιμοποιεί διευθύνσεις των 32 bits προκειμένου να προσδιορίσει ένα υπολογιστή η ένα δίκτυο.

Η μορφή μιας IP διεύθυνσης φαίνεται στο σχήμα 4.

Σχήμα 5. Μορφή IP διεύθυνσης

Το πεδίο *Δίκτυο* προσδιορίζει το δίκτυο με το οποίο είναι συνδεμένος ο υπολογιστής και το πεδίο *Υπολογιστής* προσδιορίζει τον συγκεκριμένο υπολογιστή.

Οι IP διευθύνσεις έχουν μήκος 32 bits και παρουσιάζονται με τη μορφή τεσσάρων ομάδων των 8 bit που διαχωρίζονται από τελεία.

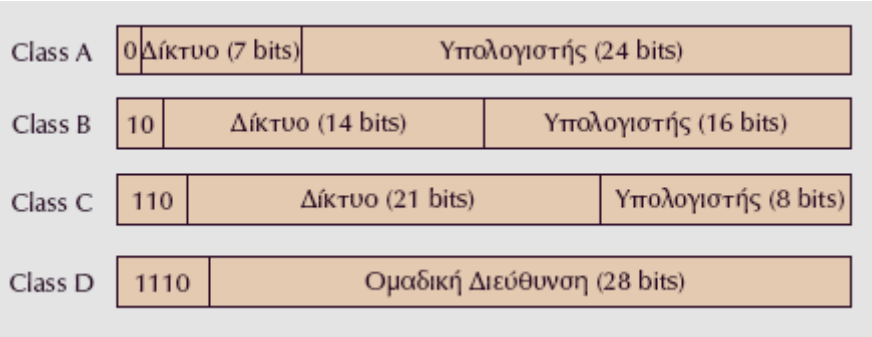
Τα δυο πεδία της IP διεύθυνσης εξαρτώνται από την κλάση του δικτύου που ανήκουν.

Οι διευθύνσεις IP είναι της μορφής ‘’194.177.198.33’’.

2.6.1 Κλάση δικτύων

Η κλάση της διεύθυνσης καθορίζονται από τα πρώτα τέσσερα πιο σημαντικά bits της διεύθυνσης. Υπάρχουν τέσσερις κλάσης οι A,B,C,D αλλά και η κλάση E που προορίζεται για μελλοντική χρήση .

Οι διευθύνσεις της κλάσης A αρχίζουν με 0 , της κλάσης B με 10 , της κλάσης C με 110, της κλάσης D με 1110 ενώ στην κλάση E ξεκινούν με 1111 (σχήμα 5).



Σχήμα 6. Κλάσεις IP διευθύνσεων

Η κλάση A είναι για μεγάλα δίκτυα με πολλούς υπολογιστές και για το λόγο αυτό δεσμεύονται 24 bits για τον υπολογιστή και 7 bits για το δίκτυο.

Η κλάση B είναι για μεσαία δίκτυα. Για το τμήμα υπολογιστή χρησιμοποιούνται 16 bits ενώ για το τμήμα δικτύου 14 bits.

Η κλάση C είναι για μικρά δίκτυα. Χρησιμοποιούνται μόνο 8 bits για το τμήμα υπολογιστή και 21 bits για το τμήμα δικτύου.

Η κλάση D επιτρέπει την ύπαρξη ομαδικών διευθύνσεων (multicast).

Για παράδειγμα η διεύθυνση 160.90.89.32 ανήκει στην κλάση B αφού 160=10100000 δηλαδή η διεύθυνση αρχίζει με 10.

2.7 MAC διευθύνσεις και Πρωτόκολλο ARP

Κάθε σε ένα δίκτυο εκτός από την διεύθυνση IP διαθέτει και μια φυσική διεύθυνση. Οι φυσικές διευθύνσεις είναι μοναδικές και είναι συνήθως ενσωματωμένες στην κάρτα στην κάρτα δικτύου από τον κατασκευαστή.
Σύμφωνα με το μοντέλο OSI οι διευθύνσεις αυτές αναφέρονται στο υποεπίπεδο Ελέγχου Προσπέλασης στο Μέσο (Media Access Control) και για ονομάζονται και MAC διευθύνσεις.

Το πρωτόκολλο ARP (Address resolution protocol) αναλαμβάνει την αντιστοιχίσει των IP διευθύνσεων στις φυσικές διευθύνσεις έτσι ώστε οι εφαρμογές να απαλλαγούν από αυτή την εργασία.
Όταν το πρωτόκολλο ARP λαμβάνει την IP διεύθυνση διερευνά των ARP πίνακα (πίνακας με αντιστοιχίες IP και φυσικών διευθύνσεων) για να δει αν υπάρχει εγγραφή. Εάν υπάρχει επιστρέφει τη φυσική διεύθυνση.
Εάν δεν υπάρχει το πρωτόκολλο ARP στέλνει μια ARP αίτηση στο τοπικό δίκτυο. Η αίτηση περιέχει την IP διεύθυνση του υπολογιστή προορισμού.

Εάν ένας υπολογιστής αναγνωρίσει την δική του IP διεύθυνση τότε απαντά στην συσκευή που έκανε τη αίτηση. Ο υπολογιστής που έκανε την αίτηση λαμβάνει την απάντηση και την καταχωρεί στον πίνακα ARP.

2.8 Σύστημα Ονομάτων Τομέων (DNS – Domain Name System)

Οι IP διευθύνσεις είναι της μορφής τεσσάρων δεκαδικών αριθμών που χωρίζονται με τελείες (π.χ. 194.177.198.2).
Επειδή οι χρήστες είναι δύσκολο να θυμούνται διευθύνσεις αυτής της μορφής χρησιμοποιούνται αντί αυτών συμβολικά ονόματα.
Έτσι λοιπόν για να επικοινωνήσουμε με μια συσκευή χρησιμοποιούμε τα συμβολικά ονόματα που είναι εύκολο να απομνημονεύσουμε.

Η μετατροπή από το συμβολικό όνομα στην πραγματική IP διεύθυνση της συσκευής προορισμού πραγματοποιείται από τον υπολογιστή του αποστολέα. Κάθε υπολογιστής διατηρεί ένα αρχείο όπου κάθε συμβολικό όνομα αντιστοιχεί σε IP. Έτσι κάθε φορά που ένα πακέτο πρέπει να σταλεί σε ένα υπολογιστή το λογισμικό αναζητά στο αρχείο αυτό τη διεύθυνση του υπολογιστή με λέξι κλειδί το όνομα του.

Επειδή όμως αυτός ο τρόπος λειτουργεί σε μικρά δίκτυα λόγω του ότι έχουμε λίγους υπολογιστές χρησιμοποιούμε το Σύστημα Ονομάτων Τομέων (DNS).

Για να λειτουργήσει το DNS χρησιμοποιεί του Εξυπηρετητές Ονόματος Τομέων (Domain Name Servers) οι οποίοι βρίσκονται σε διάφορα σημεία στο δίκτυο , παρέχουν πληροφορίες αντιστοίχησης των ονομάτων σε διευθύνσεις και συνεργάζονται μεταξύ τους. Κάθε εξυπηρετητής εξυπηρετεί συγκεκριμένο τμήμα του δικτύου.

2.9 Εφαρμογές που χρησιμοποιούν το πρωτόκολλο TCP/IP

Ηλεκτρονικό Ταχυδρομείο

Το ηλεκτρονικό ταχυδρομείο είναι εφαρμογή που επιτρέπει την αποστολή μηνυμάτων και επιστολών μεταξύ δυο η περισσοτέρων χρηστών με ηλεκτρονικό τρόπο. Για την μεταφορά του ηλεκτρονικού ταχυδρομείου χρησιμοποιείται το Πρωτόκολλο Μεταφοράς Απλού Ταχυδρομείου (Simple Mail Transfer Protocol , SMTP). Το SMTP χρησιμοποιεί σαν TCP port την θύρα 25.

Πρωτόκολλο μεταφοράς αρχείων (File Transfer Protocol, FTP)

Πρωτόκολλο για την μεταφορά αρχείων μεταξύ υπολογιστών. Η λειτουργία του βασίζεται στο μοντέλο πελάτη – εξυπηρετητή. Για την επικοινωνία με τον FTP εξυπηρετητή και την αποστολή εντολών χρησιμοποιείται η TCP port 21 ενώ για την μεταφορά δεδομένων χρησιμοποιείται το TCP port 20.

Πρωτόκολλο Απομεμακρυσμένης Σύνδεσης (Telnet)

Το πρόγραμμα Απομακρυσμένης Σύνδεσης επιτρέπει την προσπέλαση εφαρμογών που υπάρχουν σε διάφορους υπολογιστές του δικτύου, από οποιοδήποτε υπολογιστή συνδεμένο στο δίκτυο. Με αυτό τον τρόπο ένας χρήστης που εργάζεται σε ένα υπολογιστή μπορεί να συνδεθεί σε ένα άλλο υπολογιστή και να εκτελέσει προγράμματα σε αυτόν. Η λειτουργία του Telnet επιτυγχάνεται με το πρωτόκολλο Telnet.

Παγκόσμιος Ιστός (World Wide Web)

Ο Παγκόσμιος Ιστός είναι ένα σύστημα που δημιουργήθηκε αρχικά για τη διακίνηση ακαδημαϊκών πληροφοριών μέσω του δικτύου. Το σύστημα με την τεχνολογία των υπερκειμένων σχημάτισε ένα εύκολο προς τον χρήστη σύστημα πληροφοριών το οποίο περιλαμβάνει κείμενο, εικόνες, αρχεία ήχου, αρχεία εικόνας και γενικότερα διάφορα πολυμέσα.
Για την μεταφορά του υπερκειμένου χρησιμοποιείται το Πρωτόκολλο Μεταφοράς Υπερκειμένου (Hypertext Transfer Protocol).

Επίσης το TCP/IP βρίσκει εφαρμογή, στα *ασύρματα δίκτυα,* στην *τηλεφωνία μέσω διαδικτύου,* στην *μεταφορά εικόνας και ήχου* και στην *συνομιλία πραγματικού χρόνου με τη μορφή κειμένου.*

Κεφάλαιο 3ο

’’Client – Server’’

3.1 Το μοντέλο client-server (πελάτη – εξυπηρετητή)

Το μοντέλο client-server είναι ένα μοντέλο στο οποίο το δίκτυο ενώνει διάφορους υπολογιστικούς πόρους, ώστε οι clients να μπορούν να ζητούν υπηρεσίες ή πληροφορίες από έναν server.

Με άλλα λόγια, στο client-server μοντέλο, ο client θέτει μια αίτηση και ο server επιστρέφει μια ανταπόκριση ή κάνει μια σειρά από ενέργειες. Ο server μπορεί να ενεργοποιείται άμεσα με την αίτηση αυτή ή να προσθέτει την αίτηση σε μια ουρά αναμονής. Η τοποθέτηση της αίτησης σε μια ουρά μπορεί να σημαίνει ότι η αίτηση πρέπει να τεθεί σε αναμονή για να εξυπηρετηθεί. Μετά επεξεργάζεται την αίτηση με βάση την σειρά προτεραιότητας, η οποία , σε αυτή την περίπτωση, καθορίζεται από τη σειρά με την οποία ο server παρέλαβε την απαίτηση. Η πλευρά του client πρώτα στέλνει ένα μήνυμα για να καλέσει σε ετοιμότητα τον server. Από τη στιγμή που ο client και ο server έχουν επικοινωνία μεταξύ τους, ο client μπορεί να υποβάλλει την αίτησή του.

Επειδή ο client επικοινωνεί με τον server μέσω ενός καθορισμένου συστήματος διασύνδεσης, δεν χρειάζεται να γνωρίζει που ανήκει ο server ή πως ενεργεί. Ο σταθμός εργασίας τρέχει την εφαρμογή και εμφανίζει τις πληροφορίες στον χρήστη. Μόνο όταν ο client προσπελάζει πληροφορίες, τότε εγκαθίσταται επικοινωνία με τον server.

Στο client-server μοντέλο, η client εφαρμογή τρέχει σε έναν πλήρη σταθμό εργασίας. Αυτός ο σταθμός μπορεί να είναι ένας προσωπικός υπολογιστής, ένας UNIX σταθμός εργασίας ή ένας Mac. Η client εφαρμογή βασίζεται στις υπηρεσίες που προσφέρει ο server και επικοινωνούν μέσω πρωτοκόλλων, όπως το πρωτόκολλο του Internet (TCP/IP) ή του Novell (IPX/SPX).

Ο Server είναι και αυτός ένας πλήρης σταθμός εργασίας που μπορεί να λειτουργεί σε ένα προσωπικό υπολογιστή ή σε ένα μηχάνημα που έχει συγκεκριμένη κατασκευή και λειτουργικό σύστημα για να εκτελεί τη λειτουργία του εξυπηρετητή.

3.1.1 Ο Client

Ο client είναι ο αιτών των υπηρεσιών. Οι υπηρεσίες που ζητούνται από τον client μπορεί να υπάρχουν στους ίδιους σταθμούς εργασίας ή σε απομακρυσμένους σταθμούς εργασίας που συνδέονται μεταξύ τους μέσω ενός δικτύου. Ο client είναι αυτός που ξεκινάει πάντα την επικοινωνία.

Ο Client είναι το κέντρο της client-server εφαρμογής. Ο χρήστης αλληλεπιδρά με τον client, ο οποίος ξεκινάει το μεγαλύτερο μέρος της ανάπτυξης της εφαρμογής, και ο server υπάρχει για να απαντάει στις ανάγκες του client.

Οι λειτουργίες του client είναι :

- Να τρέχει το λογισμικό των γραφικών διεπαφών χρηστών (GUI).
- Να δημιουργεί τις αιτήσεις για πληροφορίες και να τις στέλνει στον server.
- Να αποθηκεύει τις επιστρεφόμενες πληροφορίες.

Το λογισμικό του Client :

- Είναι ένα πρόγραμμα εφαρμογής το οποίο γίνεται πελάτης μόνο περιστασιακά όταν απαιτείται απομακρυσμένη πρόσβαση.
- Καλείται απ’ ευθείας από τον χρήστη και τρέχει μόνο για μια σύνοδο
- Τρέχει τοπικά στον υπολογιστή του χρήστη.
- Ενεργεί για την επικοινωνία με τον εξυπηρετητή.
- Μπορεί να έχει πρόσβαση σε πολλαπλούς εξυπηρετητές, αλλά επικοινωνεί μόνο με ένα κάθε φορά.
- Δεν απαιτεί εξειδικευμένο υλικό ή κάποιο περίπλοκο λειτουργικό σύστημα.

3.1.2 Ο Server

Ο server απαντάει στις αιτήσεις που γίνονται από τους clients. Ένας client μπορεί να ενεργεί ως server εάν λαμβάνει και επεξεργάζεται αιτήσεις όπως ακριβώς και τις στέλνει (για παράδειγμα, ένας σταθμός εργασίας που χρησιμοποιείται και ως server εκτυπώσεων από άλλους). Οι server είναι ενεργοί και περιμένουν τις αιτήσεις των clients.

Οι λειτουργίες του server είναι:

- Να αποθηκεύει, να ανακτά και να προστατεύει πληροφορίες.
- Να επιθεωρεί τις αιτήσεις των clients.
- Να δημιουργεί εφαρμογές διαχείρισης πληροφοριών, όπως δημιουργία αντιγράφων, ασφάλεια κτλ.
- Να διαχειρίζεται πληροφορίες.

Το λογισμικό του εξυπηρετητή:

- Είναι ένα πρόγραμμα ειδικού σκοπού, αφιερωμένο στο να παρέχει μία υπηρεσία. Μπορεί, όμως, να χειρίζεται πολλαπλές απομακρυσμένους πελάτες την ίδια στιγμή.
- Ξεκινάει αυτόματα κάθε φορά που το σύστημα επαναλειτουργεί και συνεχίζει να λειτουργεί χωρίς να σταματάει όταν τελειώνει κάποια σύνοδος.
- Τρέχει σε ένα διαμοιραζόμενο υπολογιστή (συνήθως όχι στον προσωπικό υπολογιστή του χρήστη)
- Περιμένει παθητικά την επικοινωνία από τυχαίους απομακρυσμένους πελάτες.
- Δέχεται αιτήσεις από πολλούς τυχαίους απομακρυσμένους πελάτες.
- Απαιτεί συνήθως ισχυρό υλικό και περίπλοκο λειτουργικό σύστημα.

3.2 Διαδικασία σύνδεσης

Η διαδικασία σύνδεσης ενός client και ενός server μπορεί να απλοποιηθεί στα ακόλουθα βήματα:

1. Ο χρήστης δημιουργεί μια αίτηση ή ένα ερώτημα.
2. Ο client μορφοποιεί το ερώτημα και το στέλνει στο server.
3. Ο server ελέγχει την δυνατότητα πρόσβασης του χρήστη.
4. Ο server επεξεργάζεται το ερώτημα και επιστρέφει τα αποτελέσματα.
5. Ο client λαμβάνει την ανταπόκριση και τη μορφοποιεί για τον χρήστη.
6. Ο χρήστης βλέπει και χειρίζεται την πληροφορία.

Ο server περιμένει παθητικά να φτάσουν οι αιτήσεις του client μέσω του δικτύου. Ο server πρέπει πάντα να απαντάει στους clients, ακόμα και όταν πολλοί clients κάνουν ταυτόχρονες αιτήσεις. Εάν πολλοί clients κάνουν αιτήσεις ταυτόχρονα, ο server πρέπει να είναι ικανός να βάζει σε προτεραιότητα τις αιτήσεις των clients, και να επεξεργάζεται πολλές αιτήσεις την στιγμή.

Από την στιγμή που ο server δέχεται από τον client την αίτηση, ο server πρέπει να βεβαιωθεί ότι ο client είναι εξουσιοδοτημένος να λάβει την πληροφορία ή την απάντηση. Αν ο client δεν είναι εξουσιοδοτημένος, ο server απορρίπτει την αίτηση και στέλνει μήνυμα στον client. Εάν ο client είναι εξουσιοδοτημένος, ο server συνεχίζει και επεξεργάζεται την αίτηση.

Η επεξεργασία της αίτησης περιλαμβάνει την παραλαβή της αίτησης του client, την μετατροπή του σε μια μορφή που μπορεί ο server να χρησιμοποιήσει και την επεξεργασία της ίδιας της αίτησης.

Όταν η επεξεργασία ολοκληρώνεται, ο server στέλνει τα αποτελέσματα πίσω στον client. Μετά, ο client μπορεί να μεταφράσει και να χρησιμοποιήσει τις πληροφορίες.

Δεν υπάρχει προκαθορισμένος διαχωρισμός στις ευθύνες για τις client-server εφαρμογές. Ανάλογα με τις ανάγκες μας, μπορούμε και να διαχωρίσουμε την εφαρμογή. Το ισχυρό client μοντέλο δίνει περισσότερες λειτουργίες στον client, ενώ το ισχυρό server μοντέλο δίνει περισσότερες λειτουργίες στον server. Οι servers εφαρμογών και συναλλαγών τείνουν να είναι ισχυροί servers, ενώ οι servers βάσεων δεδομένων και αρχείων τείνουν να έχουν ισχυρούς clients.

Ανεξάρτητα του πώς διαχωρίζουμε την εφαρμογή, η βασική ευθύνη του server παραμένει η ίδια: να εξυπηρετεί τους clients που κάνουν αιτήσεις. Ο server πρέπει να είναι ικανός να ανταποκριθεί στην αίτηση του client αμέσως. Όταν ο server τελειώνει την επεξεργασία των αποτελεσμάτων και είναι έτοιμος να

επιστρέψει τα αποτελέσματα στον client, πρέπει να μορφοποιήσει τα αποτελέσματα και να τα στείλει με ένα τρόπο που μπορεί ο client να καταλάβει.

Ο server παραδίδει τις πληροφορίες στο πρωτόκολλο, που διευθύνει ένα πακέτο, μορφοποιεί τις πληροφορίες για να τις τοποθετήσει στο πακέτο και περνάει το πακέτο στο δίκτυο. Το δίκτυο μετά βεβαιώνεται ότι το πακέτο πηγαίνει στον client.

3.3 Η σημασία του client-server στην ανάπτυξη των εφαρμογών

Η client-server αρχιτεκτονική επιτρέπει την εκμετάλλευση των δυνατοτήτων που παρέχουν οι clients, οι servers και το δίκτυο, όταν αναπτύσσεται μια εφαρμογή. Για να επωφεληθούμε από αυτές τις δυνατότητες, πρώτα πρέπει να γνωρίζουμε ότι η πιο συνηθισμένη λειτουργία μιας client-servers εφαρμογής είναι η παροχή δυνατότητας πρόσβασης του χρήστη στις πληροφορίες, αποτελεσματικά και εύκολα. Είναι αναγκαίο να ενοποιούνται ομαλά τα GUIs, οι κατανεμημένες εφαρμογές, οι συγγενικές βάσεις δεδομένων και τα δίκτυα.

Από την στιγμή που το μοντέλο client-server είναι διαφορετικό από τα κλασσικά μοντέλα, αυτή ενισχύει τις αδυναμίες των παραδοσιακών μεθοδολογιών ανάπτυξης συστημάτων. Οι πληροφορίες που ελέγχονται από αυτό το σύστημα είναι περισσότερες από τις προηγούμενες αρχιτεκτονικές. Η ασφάλεια ρυθμίζεται σε διάφορα επίπεδα συμπεριλαμβανομένου των σταθμών εργασιών, των πληροφοριών και του χρόνου. Οι clients και οι servers προσδιορίζονται από το λογισμικό και όχι από το υλικό. Τα RPCs, που επιτρέπουν στον client να απαιτήσουν μια υπηρεσία από τον server, είναι πολύ σημαντικά στο client-server computing.

3.4 Ανάπτυξη Client – Server εφαρμογών

Η ανάπτυξη client-server εφαρμογών διαφέρει από τον παραδοσιακό προγραμματισμό. Για να αναπτύξουμε μια εφαρμογή, ακολουθούμε τα παρακάτω γενικά βήματα:

1. Προσδιορισμός το πρωτόκολλο επικοινωνίας του client και server.
2. Ανάπτυξη των clients και servers προγραμμάτων.
3. Μεταγλωτισμός των προγραμμάτων.
4. Διασυνδέση των βιβλιοθηκών.
5. Εξέταση των εφαρμογών τοποθετώντας και τρέχοντας των server σε απομακρυσμένο σύστημα και των client τοπικά.

Εξαιτίας της πολυπλοκότητας της client-server αρχιτεκτονικής, η ανάπτυξη client-server εφαρμογών απαιτεί πιο λεπτομερή σχεδιασμό. Ειδικότερα για το πως να διαχωριστεί η εφαρμογή ανάμεσα στον client και τον server και πως να κατανεμηθούν οι πληροφορίες μεταξύ client και server.

Το σύστημα διασύνδεσης των χρηστών ανήκει στον client. Εργασίες που δουλεύουν καλά στον client είναι η μορφοποίηση ερωτημάτων για τον server,η δημιουργία αναφορών και ο έλεγχος των σφαλμάτων. Η λογική της εφαρμογής στον client ονομάζεται front end.

Ο server, από την άλλη, ευθύνεται για την ανάκτηση, τον χειρισμό και την ασφάλεια των πληροφοριών. Φυσικά, αυτό εξαρτάται από το πώς οι πληροφορίες κατανέμονται. Η λογική της εφαρμογής στον server ονομάζεται back end.

Οι clients-server εφαρμογές μετακινούν το επίκεντρο του προγραμματισμού από τις μηχανές προς τους χρήστες. Οι τελικοί χρήστες προσδοκούν τα ακόλουθα:

- Πρόσβαση σε πολλαπλές πληροφορίες, δηλαδή, οι πληροφορίες να είναι διαθέσιμες σε όλους τους εξουσιοδοτημένους χρήστες.
- Ολοκληρωμένες υπηρεσίες.
- Πρόσβαση σε πόρους μέσω διαφόρων πλατφορμών.
- Ανταλλαγή και εκμετάλλευση πληροφοριών.
- Πρόσβαση σε οποιεσδήποτε πληροφορίες.
- Ευκολότερη διατήρηση και συντήρηση των πληροφοριών.

3.5 RPC (Remote Procedure Calls)

Γενικά, οι clients χρησιμοποιούν RPCs για να υποβάλλουν αιτήσεις απομακρυσμένων υπηρεσιών. Οι clients στέλνουν στις απομακρυσμένες υπηρεσίες κάποιες παραμέτρους εισόδου και λαμβάνουν παραμέτρους εξόδου, που είναι τα αποτελέσματα των αιτήσεων. Ένα RPC είναι η μέθοδος με την οποία ένας επεξεργαστής ενεργοποιεί έναν άλλο επεξεργαστή, ο οποίος ανήκει σε ένα απομακρυσμένο σύστημα. Στη διάρκεια ενός RPC, ο client στέλνει μια αίτηση στον δίκτυο. Ο server περιμένει για αιτήσεις. Όταν λάβει μια αίτηση, ο server εκτελεί την απαιτούμενη διαδικασία και δημιουργεί την απάντηση. Η απάντηση μεταβιβάζεται μέσω του δικτύου στον client. Τα RPCs είναι η ραχοκοκαλιά της client-server αρχιτεκτονικής.

Πολλοί προγραμματιστές αναπτύσσουν κώδικες, χρησιμοποιώντας δομημένες τεχνικές και υπορουτίνες εδώ και χρόνια. Σήμερα αυτές οι υπορουτίνες πρέπει να τοποθετούνται κάπου, ώστε να είναι δυνατόν να χρησιμοποιούνται από τον καθένα. Οι RPCs προσφέρουν αυτή την δυνατότητα να καθορίζουν τον τρόπο με τον οποίο πρέπει οι προγραμματιστές να στέλνουν αιτήσεις σε απόμακρους σταθμούς και οι σταθμοί να τις αναγνωρίζουν και να ανταποκρίνονται σωστά. Εάν μία εφαρμογή αποστέλλει μία αίτηση και αυτή είναι ενσωματωμένη σε μία RPCs, η αίτηση μπορεί να είναι τοποθετημένη οπουδήποτε μέσα στο δίκτυο στο οποίο έχει την δυνατότητα να προσπελάσει ο χρήστης. Οι συνδέσεις μεταξύ των Clients και των Servers μέσω μίας RPC είναι εγκαταστημένες στο στρώμα μεταφοράς του OSI μοντέλου. Επιπλέον, η ευκολία των RPCs προσφέρει την κλήση και την εκτέλεση αιτήσεων από επεξεργαστές, που λειτουργούν υπό διαφορετικά λειτουργικά συστήματα και που χρησιμοποιούν διαφορετικές πλατφόρμες υλικού από αυτά του χρήστη.

3.6 Γραφική Διεπαφή Χρήστη (GUI)

Η γραφική διεπαφή χρήστη (Graphical User Interface) προσφέρει στον χρήστη μια εύκολη στη χρήση διασύνδεση. Με τη γραφική διεπαφή χρήστη (GUI), οι χρήστες δεν έχουν να κάνουν τίποτα περισσότερο από το να ’’σημειώνουν και να επιλέξουν’’ για να κάνουν την δουλεία τους. Οι χρήστες μπορούν να αλληλεπιδράσουν με γραφικές απεικονίσεις γρηγορότερα και ευκολότερα από ότι μπορούν όταν έχουν να αντιμετωπίσουν μόνο κείμενο.
Οι χρήστες είναι συνήθως ήδη εξοικειωμένοι με τη γραφική διεπαφή χρήστη (GUI) από την απασχόληση τους με τους προσωπικούς υπολογιστές τους. Η διασύνδεση καθορίζει πως οι χρήστες εισάγουν πληροφορίες και πως οι εφαρμογές επιστρέφουν πληροφορίες στους χρήστες.

3.7 Είδη Εξυπηρετητών (Server)

Οι σταθμοί εξυπηρέτησης ταξινομούνται ανάλογα με την υπηρεσία που προσφέρουν σε:

Server Platforms

Ένας όρος που χρησιμοποιείται συχνά συνώνυμα με λειτουργικό σύστημα, μια πλατφόρμα είναι το βασικό υλικό ή λογισμικό για ένα σύστημα και είναι έτσι η μηχανή που οδηγεί τον εξυπηρετητή.

Application Servers

Οι εξυπηρετητές αυτοί κατέχουν ένα μεγάλο μέρος της περιοχής υπολογισμού μεταξύ database servers και του τελικού χρήστη και πολλές φορές τους συνδέουν.

Audio/Video Servers

Οι εξυπηρετητές Audio/Video εισάγουν δυνατότητες πολυμέσων στις ιστοσελίδες, με το να τις καθιστούν ικανές να
μεταδώσουν περιεχόμενο πολυμέσων με σταθερό ρυθμό.

Chat Servers

Οι Chat servers δίνουν τη δυνατότητα σε έναν μμεγάλο αριθμό χρηστών να ανταλλάξουν πληροφορίες σε ένα περιβάλλον παρόμοιο με τα newsgroups του Internet και προσφέρει δυνατότητες συζήτησης πραγματικού χρόνου.

Fax Servers

Ένας fax server είναι ιδανικός για επιχειρήσεις που επιθυμούν να μειώσουν εισερχόμενους και εξερχόμενους τηλεφωνικούς πόρους, αλλά χρειάζονται να στέλνουν με fax διάφορα έγγραφα.

FTP Servers

Μια από τις παλαιότερες υπηρεσίες διαδικτύου, το πρωτόκολλο μεταφοράς αρχείων(File Transfer Protocol) καθιστά δυνατή την ασφαλή μεταφορά ενός ή περισσοτέρων αρχείων μεταξύ υπολογιστών, παρέχοντας ασφάλεια και οργάνωση αρχείων, καθώς και έλεγχο μεταφοράς.

Groupware Servers

Ένας groupware server είναι λογισμικό σχεδιασμένο να δίνει τη δυνατότητα σε απομακρυσμένους χρήστες να συνεργάζονται σε πραγματικό περιβάλλον, ανεξάρτητα από την τοπολογία τους, διαμέσου διαδικτύου(Internet) ή ενός εταιρικού εσωτερικού δικτύου(corporate intranet).

IRC Servers

Μια επιλογή για αυτούς που αναζητούν δυνατότητα συνομιλίας σε πραγματικό χρόνο. Το Internet Relay Chat αποτελείται από διάφορα ξεχωριστά δίκτυα εξυπηρετητών, που επιτρέπουν στους χρήστες να συνδεθούν διαμέσου δικτύου IRC.

List Servers

Οι εξυπηρετητές αυτοί προσφέρουν έναν τρόπο για την καλύτερη διαχείριση mailing lists, είτε είναι αλληλεπιδραστικές συζητήσεις ανοιχτές στο κοινό είτε μονόπλευρες λίστες που παραδίδουν αγγελίες, εξειδικευμένες εκδόσεις ή διαφημιστικά.

Mail Servers

Σχεδόν τόσο διαδεδομένοι και αποφασιστικοί όσο οι Web servers. Οι mail server μετακινούν και αποθηκεύουν ηλεκτρονικό ταχυδρομείο σε εταιρικά δίκτυα(μέσω LAN και WAN) και διαμέσου διαδικτύου(Internet).

News Servers

Δρουν ως ένας διανομέας για χιλιάδες δημόσιες ομάδες ειδήσεων που είναι προσβάσιμες αυτή τη στιγμή στο δίκτυο νέων USENET.

Proxy Servers

Οι Proxy server εντοπίζονται μεταξύ ενός προγράμματος του πελάτη (Web browser) και ενός εξωτερικού εξυπηρετητή (ένας άλλος εξυπηρετητής στον Ιστό) και φιλτράρουν αιτήσεις, βελτιώνουν την απόδοση και μοιράζουν τις συνδέσεις.

Telnet Servers

Ένας εξυπηρετητής Telnet καθιστά ικανούς τους χρήστες να συνδεθούν σε έναν υπολογιστή που είναι host και να εργαστούν όπως θα εργάζονταν αν καθόταν στο ίδιο τον απομακρυσμένο υπολογιστή.

Web Servers

Ένας εξυπηρετητής Ιστού (Web server) προωθεί στατικό περιεχόμενο σε έναν προβολής ιστοσελίδων, φορτώνοντας ένα αρχείο από το δίσκο και προωθώντας το διαμέσου δικτύου στον χρήστη. Αυτή η επικοινωνία μεταξύ του προγράμματος και του εξυπηρετητή γίνεται χρησιμοποιώντας το HTTP.

3.8 Συνηθισμένα CLIENT-SERVER εργαλεία

Υπάρχει μια μεγάλη επιλογή από εργαλεία ανάπτυξης client-server που διατίθενται στην σημερινή αγορά των υπολογιστών. Παρακάτω βλέπουμε ορισμένα από τα πιο δημοφιλή εργαλεία. Αυτό που τα περισσότερα εργαλεία έχουν κοινό είναι ότι γενικά είναι βελτιστοποιημένα για ανάπτυξη εφαρμογών .

Borland Delphi

Η Delphi είναι ένα εργαλείο αντικειμένων Άμεσης Ανάπτυξης Εφαρμογών. Η γλώσσα προγραμματισμού είναι βασισμένη στην Borland's Object Pascal και δημιουργεί απλό κώδικα για τον προσωπικό υπολογιστή. Το περιβάλλον ανάπτυξης είναι οπτικό, βασισμένο στις ιδέες που πρώτα χρησιμοποίησε η Visual Basic.

Visual Basic

Η Visual Basic είναι η καθαρά οπτική γλώσσα προγραμματισμού. Προσφέρει τα εργαλεία για να δημιουργήσει client εφαρμογές. Μπορεί να προσπελάσει βάσεις δεδομένων μέσω ODBC (Open Database Connectivity). Το μόνο μεγάλο πρόβλημα των εφαρμογών σε Visual Basic είναι η έλλειψη ταχύτητας επειδή είναι interpreted..

PowerBuilder

Το PowerBuilder είναι ένα εργαλείο για ανάπτυξη GUI (Graphical User Interface) σε εφαρμογές βάσεων δεδομένων. Είναι ένα από τα πιο παλιά εργαλεία δημιουργίας client. Το PowerBuilder έχει δυνατότητες χρήσης Windows 98, Windows NT στο Intel και Alpha, Solaris (UNIX), και Macintosh clients και με αρχή την έκδοση 5 έχει ενσωματωμένο Netscape.

C / C++

Οι παλιότερες αξιόπιστες γλώσσες client-server είναι οι C και C++. Οι πρόσφατοι PC Compilers σε C/C++ παρέχουν ένα οπτικό περιβάλλον προγραμματισμού για ανάπτυξη client μέσω των γραμμών της Visual Basic. Δυνατότητες επικοινωνιών και βάσεων δεδομένων προσφέρονται μέσω βιβλιοθηκών υπορουτίνων. Πριν την προσθήκη της οπτικής δημιουργίας client, εργαλεία, όπως το PowerBuilder, κρατούσαν ένα καθαρό πλεονέκτημα ανάπτυξης.

Developer/2000

Το Developer/2000 είναι ένα εργαλείο ανάπτυξης client-server που δημιουργήθηκε από το Oracle. Περιλαμβάνει μονάδες για σχεδιασμό και δημιουργία φορμών εισαγωγής δεδομένων και παράγει αναφορές από μια βάση δεδομένων της Oracle.

Access

Η Access είναι ένα προϊόν της Microsoft βασισμένο σε SQL. Η Access συχνά χρησιμοποιείται ως front-end client σε άλλα SQL συστήματα βάσεων δεδομένων, όπως η Oracle και Sybase.

Java

Η Java είναι μία γλώσσα προγραμματισμού ειδικά σχεδιασμένη για χρήση σε κατανεμημένα περιβάλλοντα όπως είναι το Internet (ή αλλιώς Διαδίκτυο στα ελληνικά). Δημιουργήθηκε με την προοπτική να μοιάζει σε πολλά σημεία με την παλιότερη γλώσσα C++, αλλά με το πλεονέκτημα να είναι πιο εύκολη στη χρήση και επιβάλει μία ολοκληρωτικά αντικειμενοστραφή αντιμετώπιση όλων των πραγμάτων. Μία εφαρμογή σε Java μπορεί είτε να εκτελεστεί σε έναν μεμονωμένο Η/Υ, είτε να κατανεμηθεί μέσω ενός δικτύου σε πολλούς Η/Υ. 7Επίσης με την Java μπορεί να δημιουργηθούν μικρές εφαρμογές γνωστές ως applets που προσαρτώνται σε σελίδες στο Web. Με την χρήση αυτών των μικροεφαρμογών (όχι μικρές ως προς το μέγεθος ή τη λειτουργικότητα, αλλά ως προς την πληρότητα στοιχείων που χαρακτηρίζουν μια κανονική εφαρμογή), είναι δυνατή η αλληλεπίδραση με τον χρήστη μέσα από έναν απλό Web browser.

Άλλες γλώσσες

Και πολλές άλλες γλώσσες προγραμματισμού, όπως η Smalltalk και η Eiffel, έχουν χρησιμοποιηθεί με επιτυχία για ανάπτυξη client-server. Οποιαδήποτε γλώσσα που μπορεί να δημιουργήσει βιβλιοθήκες, μπορεί να χρησιμοποιηθεί αποτελεσματικά σε εφαρμογές client–server.

Κεφάλαιο 4ο

“Ανάπτυξη εφαρμογής Client – Server”

4.1 Γενικά

Η Visual Basic μας προσφέρει πολλές δυνατότητες προγραμματισμού εφαρμογών δικτύου. Με τα διάφορα εργαλεία και αντικείμενα αλλά και το γραφικό περιβάλλον που μας προσφέρει μπορούμε να φτιάξουμε εφαρμογές φιλικές προς τον χρήστη.

Σκοπός της εφαρμογής μας θα είναι η δημιουργία ενός client και ενός server.

Θα δημιουργήσουμε ένα γραφικό περιβάλλον λειτουργικό προς τον χρήστη με έλεγχο σφαλμάτων και μηνύματα που βοηθούν τον χρήστη να διαχειριστή και να κατανοήσει τυχόν σφάλματα που θα κάνει κατά την χρήση της εφαρμογής.

Συγκεκριμένα η εφαρμογή μας θα είναι ένας Client ο οποίος θα συνδέεται σε ένα Server μέσω δικτύου.

Της παραμέτρους σύνδεσης του Client στον Server θα τις καθορίζει ο χρήστης.

4.2 Η ανάπτυξη της εφαρμογής

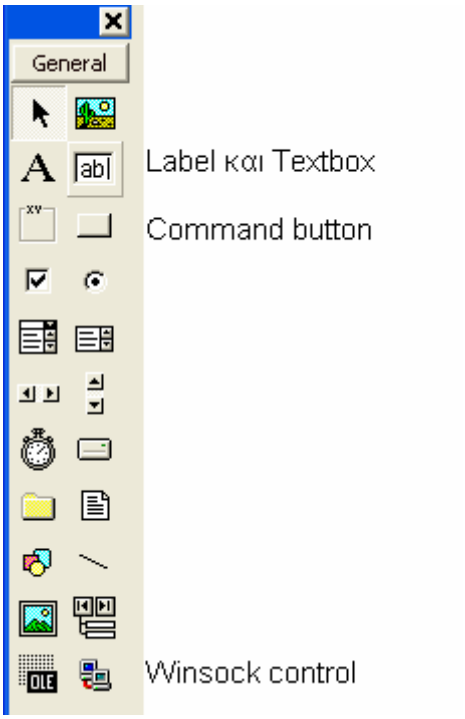
Τα εργαλεία της Visual Basic που θα χρησιμοποιήσουμε είναι τα εξής :

Textbox, Label, Command Button και το Winsock Control που είναι και το βασικότερο εργαλείο της εφαρμογής.

Στην εικόνα 1 φαίνεται η γραμμή εργαλείων της Visual Basic.

Στην Visual Basic προγραμματίζουμε συνδέοντας τα αντικείμενα που τοποθετούμε σε μια φόρμα με τον κώδικα που γράφουμε. Για κάθε αντικείμενο γράφουμε στον κώδικα του προγράμματος μια συνάρτηση όπου μέσα σε αυτήν καθορίζουμε τις ιδιότητες του αντικειμένου και γράφουμε εντολές για να συνδέουμε το αντικείμενο με άλλα αντικείμενα.

Κάθε εργαλείο που χρησιμοποιούμε χαρακτηρίζεται έχει συγκεκριμένες λειτουργίες και χαρακτηρίζεται από συγκεκριμένες ιδιότητες.



Εικόνα 1. Η γραμμή εργαλείων της Visual Basic

4.2.1 Textbox , Label και CommandButton

Textbox

Δημιουργεί ένα πλαίσιο κειμένου στο οποίο μπορούμε να τυπώσουμε το περιεχόμενο μιας μεταβλητής μνήμης, ενός στοιχείου σειράς, ή ενός τομέα. Όλα τα τυποποιημένα οπτικά χαρακτηριστικά γνωρίσματα όπως η περικοπή, το αντίγραφο, και η επικόλληση, είναι διαθέσιμα στο παράθυρο κειμένου.

Label

Ένας έλεγχος που περιέχει το κείμενο για να περιγράψει τους τομείς της φόρμας ή άλλων ελέγχων. Μερικές φορές αναφερόμενος ως υπαγόρευση.

Command Button

Ένας έλεγχος που συνδέεται με μια εντολή. Όταν χτυπάτε το κουμπί εντολής κατά την εκτέλεση του προγράμματος , η εντολή που συνδέεται με το κουμπί εκτελείτε.

4.2.2 Winsock Control

Το winsock control αποτελεί το βασικό εργαλείο της εφαρμογής client – server. Το winsock control είναι αόρατο στο χρήστη και παρέχει εύκολη πρόσβαση στις υπηρεσίες δικτύων TCP και UDP. Για να γράψουμε τις εφαρμογές πελατών ή εξυπηρετητών δεν χρειάζεται να καταλαβαίνουμε τις λεπτομέρειες του TCP ή να καλέσουμε το χαμηλό επίπεδο Winsock APIs. Με τον καθορισμό των ιδιοτήτων και την επίκληση των μεθόδων του ελέγχου, μπορούμε εύκολα να συνδέσουμε ένα μακρινό υπολογιστή με ένα τοπικό υπολογιστή.

Το WinSock μας επιτρέπει να συνδεόμαστε με ένα μακρινό υπολογιστή χρησιμοποιώντας είτε το πρωτόκολλο (UDP) είτε το πρωτόκολλο ελέγχου μετάδοσης (TCP). Και τα δύο πρωτόκολλα μπορούν να χρησιμοποιηθούν για να δημιουργήσουν τις εφαρμογές πελατών και εξυπηρετητών υπολογιστών.

Πιθανές χρήσεις

Δημιουργία μιας εφαρμογής client που συλλέγει τις πληροφορίες χρηστών και τις στέλνει σε έναν κεντρικό υπολογιστή.

Δημιουργία μιας εφαρμογή κεντρικών υπολογιστών που λειτουργεί ως κεντρικό σημείο συλλογής για τα στοιχεία από διάφορους χρήστες.

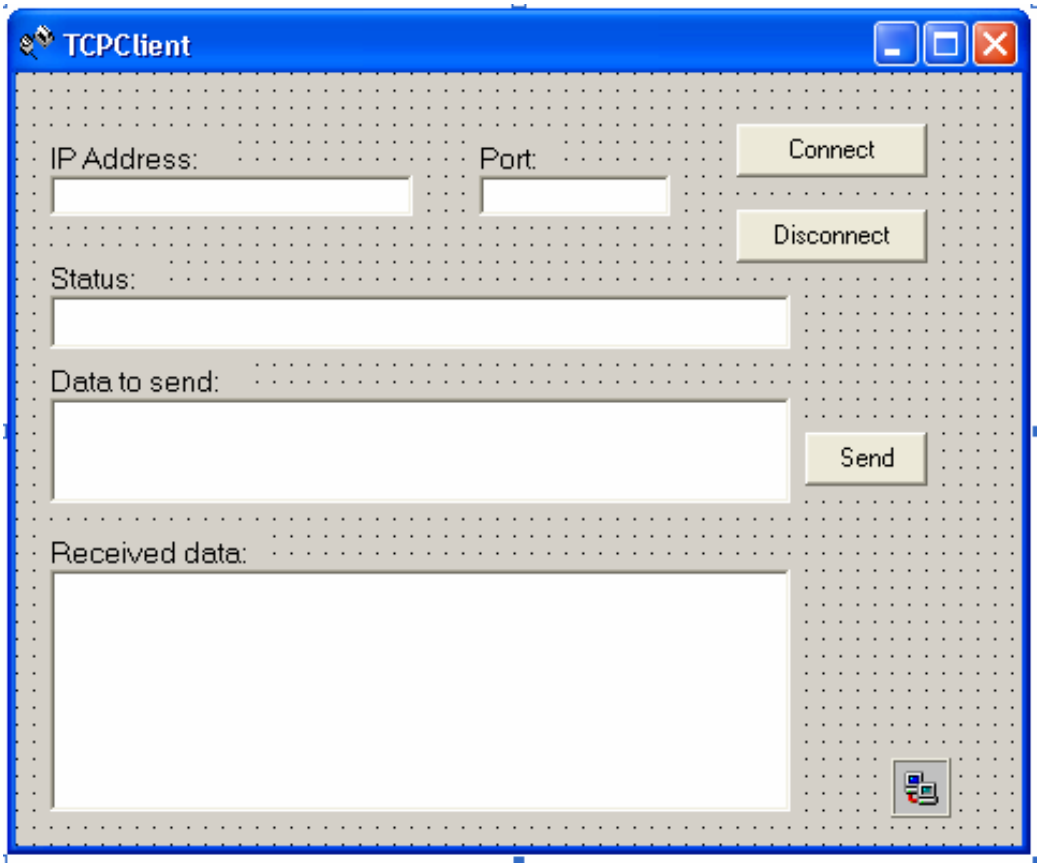
Δημιουργία μιας εφαρμογής "συνομιλίας".

Το Winsock Control είναι το εργαλείο της Visual Basic που μας δίνει την δυνατότητα να διασύνδεουμε την εφαρμογή μας με το επίπεδο μεταφοράς και ανάλογα με το πρωτόκολλο TCP ή UDP. Χρησιμοποιώντας το winsock control και ρυθμίζοντας της παραμέτρους, τις μεθόδους και τις εντολές που θα εκτελεί ανάλογα με τα γεγονότα που λαμβάνουν χώρα και φυσικά συνδέοντας το με τα υπόλοιπα αντικείμενα τις εφαρμογής φτιάχνουμε μια πλήρη εφαρμογή δικτύου.

Η ανάλυση του αντικειμένου Winsock Control δίνεται στο παράρτημα Α’.

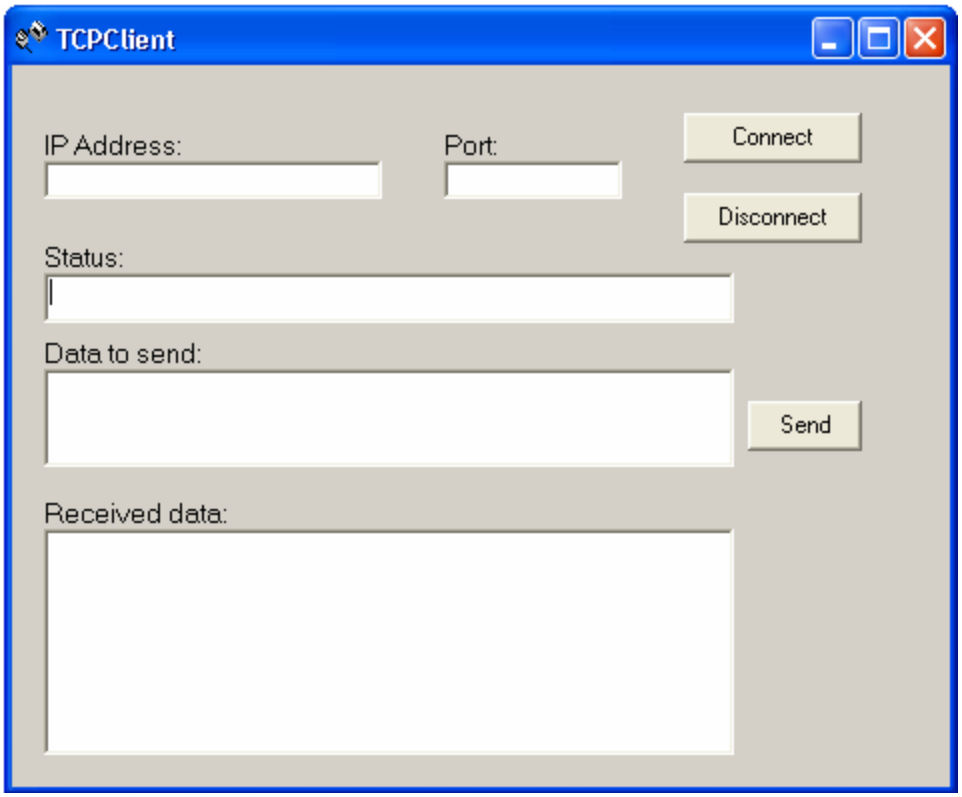
4.3 Ο Client

Αφού προσθέσουμε τα εργαλεία στην φόρμα η εφαρμογή του Client έχει την μορφή που φαίνεται στη εικόνα 2.



Εικόνα 2. Η μορφή της φόρμας στο περιβάλλον εργασίας της Visual Basic

Όταν θα τρέξει η εφαρμογή το Winsock Control θα είναι αόρατο και έτσι ο χρήστης θα δει στην οθόνη την εικόνα 3.



Εικόνα 3. Η εφαρμογή του Client όταν εκτελείτε

4.3.1 Ανάλυση κώδικα του Client

```
Private Sub cmdconnect_Click()


'Συνάρτηση για το αντικείμενο commandbutton cmdconnect
'Με τους τρεις ελέγχους if εξασφαλίζουμε το ότι ο χρήστης θα δώσει
'τα δεδομένα που χρειάζεται για να λειτουργήσει ο Client και να συνδεθεί
'Χρησιμοποιώντας την εντολή MsgBox τυπώνουμε στον χρήστη μηνύματα που τον βοηθούν

If txtip = "" And txtport = "" Then

  MsgBox "Please enter the IP address and port number",
vbInformation, "Warning"

  Exit Sub

 End If
```

'Ελέγχουμε εάν ο χρήστης έδωσε IP και Port και του τυπώνουμε το αντίστοιχο μήνυμα

```
If txtip = "" Then
  MsgBox "Please enter the IP address", vbInformation, "Warning"
  Exit Sub
 End If
```

'Ελέγχουμε εάν ο χρήστης έδωσε IP και του τυπώνουμε το αντίστοιχο μήνυμα

```
 If txtport = "" Then
  MsgBox "Please enter the port number", vbInformation, "Warning"
  Exit Sub
 End If
```

'Ελέγχουμε εάν ο χρήστης έδωσε το Tcp Port και του τυπώνουμε το αντίστοιχο μήνυμα

```
 Winsock1.RemoteHost = txtip
 Winsock1.RemotePort = txtport
```

'Αντιστοιχούμε τις ιδιότητες remotehost και remoteport με τα textbox txtip και txtport

'Έτσι το winsock παίρνει την IP η το Hostname και την πόρτα

'του υπολογιστή που θα συνδεθεί

```
Winsock1.Connect txtip, txtport
```

'Καλούμε την λειτουργία σύνδεσης του winsock

'με δεδομένα την IP και την πόρτα του Server

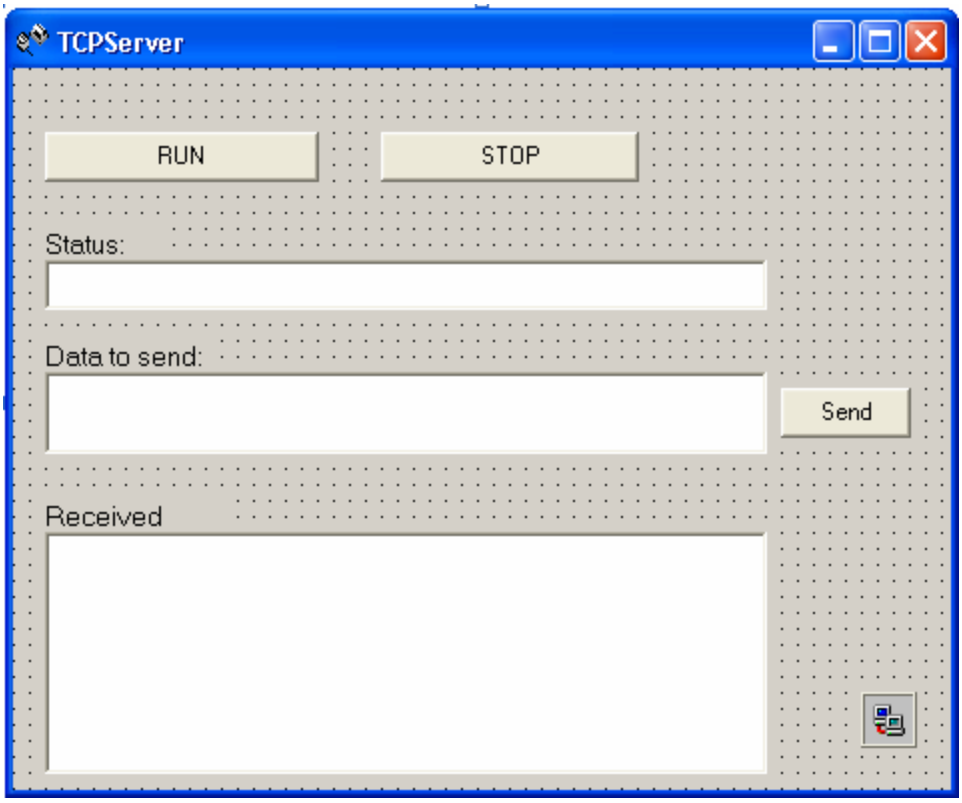
```
End Sub
```

```
Private Sub cmddisconnect_Click()
'Συνάρτηση για το commandbutton cmddisconnect
'Έλεγχος για την περίπτωση που ο χρήστης πατήσει το κουμπί αποσύνδεσης
'χωρίς να είναι συνδεμένος
If Winsock1.State = sckClosed Then
   MsgBox "You are not conected", vbInformation, "Warning"
  Exit Sub
 End If
Winsock1.Close
'Κλείνουμε την tcp πόρτα που είχε ανοίξει το winsock για να συνδεθεί στον server
txtstatus = "Disconnected from the server " & Winsock1.RemoteHostIP
'Τυπώνουμε στο πλαίσιο κειμένου status το παραπάνω μήνυμα
End Sub

Private Sub Winsock1_Connect()
'Όταν συνδεθούμε στον server τυπώνεται στο πλαίσιο κειμένου status
'το παρακάτω μήνυμα και ενεργοποιείται το commandbutton cmdsend
'ώστε να μπορούμε να στείλουμε δεδομένα
 txtstatus = "Connected to " & Winsock1.RemoteHostIP
 cmdsend.Enabled = True
 End Sub
```

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
'Συνάρτηση που διαχειρίζεται τα εισερχόμενα δεδομένα που έχουμε ορίσει να είναι
'τύπου χαρακτήρα
'Καλούμε την μέθοδο λήψης δεδομένων του winsock Getdata
'και τυπώνουμε τα εισερχόμενα δεδομένα
'στο πλαισιο κειμένου txtreceived
 Winsock1.GetData Data, vbString
 txtreceived = txtreceived & Data
End Sub

Private Sub cmdSend_Click()
'Συνάρτηση αποστολής δεδομένων , ελέγχουμε αν ο client είναι συνδεμένος και
'εάν δεν είναι του τυπώνουμε το αντίστοιχο μήνυμα
If Winsock1.State = sckClosed Then
   MsgBox "You are not conected", vbInformation, "Warning"
  Exit Sub
 End If
'Καλούμε την μέθοδο αποστολής δεδομένων
'και σαν δεδομένα θέτουμε το κείμενο στο πλαίσιο κειμένου
'txtdatasend
Winsock1.SendData ("" & txtdatasend.Text)
'Σβήνουμε το κείμενο που εισάγαμε για αποστολή
txtdatasend = ""
End Sub
```

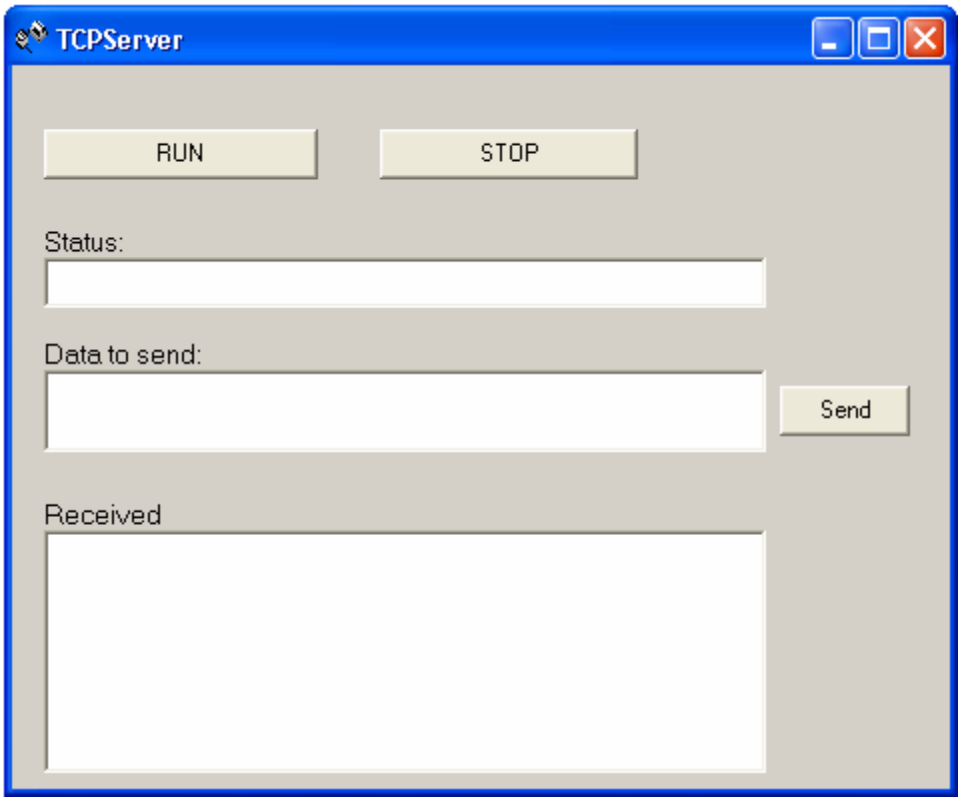
4.4 Ο Server

Αφού προσθέσουμε τα εργαλεία στην φόρμα η εφαρμογή του Server έχει την μορφή που φαίνεται στη εικόνα 4.



Εικόνα 4. Η εφαρμογή του Server στο περιβάλλον της Visual Basic

Όταν θα τρέξει η εφαρμογή το Winsock Control θα είναι αόρατο και έτσι ο χρήστης θα δει στην οθόνη την εικόνα 5.



Εικόνα 5. Η εφαρμογή του Server όταν εκτελείται

4.4.1 Ανάλυση κώδικα του Server

```
Private Sub cmdrun_Click()
```

΄Συνάρτηση για το command button cmdrun

```
 Winsock1.LocalPort = 1020
 Winsock1.Listen
```

΄Θέτουμε σαν local port στην οποία θα ακούει ο server την 1020
΄Καλούμε την ιδιότητα listen του server

```
 txtstatus.Text = "Server is listening for incoming connection
on port 1020"
```

΄Τυπώνουμε στο textbox το παραπάνω μήνυμα

```
End Sub
```

```
Private Sub cmdstop_Click()
```
΄Συνάρτηση για το command button cmdstop

```
 Winsock1.Close
```
΄Καλούμε την ιδιότητα close έτσι ώστε να κλείσει ο server

```
txtstatus.Text = "Server is closed"
```
΄Τυπώνουμε το αντίστοιχο μήνυμα στο textbox status

```
End Sub


Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
```
΄Συνάρτηση για την περίπτωση που ο server θα δεχτεί αίτηση σύνδεσης

```
 If Winsock1.State <> sckClosed Then Winsock1.Close

 txtstatus = "Incoming connection request from " &
Winsock1.RemoteHostIP

 Winsock1.Accept requestID
```

΄Ελέγχουμε αν η κατάσταση του server είναι διαφορετική από την κατάσταση ΄closed και τότε κλείνουμε το socket όπου έκανε listening ο server και ΄δεχόμαστε την σύνδεση τυπώνοντας και ένα μήνυμα

```
 cmdsend.Enabled = True
```

΄Ενεργοποιούμε το command button cmdsend για να μπορούμε να στείλουμε ΄δεδομένα

```
End Sub


Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
```
΄Συνάρτηση για την περίπτωση που ο server λάβει δεδομένα

```
txtstatus = "Conected with client " & Winsock1.RemoteHostIP
```
΄Τυπώνουμε μήνυμα με την IP του client που συνδέθηκε

```
 Winsock1.GetData Data, vbString
```

΄Καλούμε την ιδιότητα getdata για να λάβουμε τα δεδομένα
```
 txtdatareceived = txtdatareceived & Data
```
΄Τυπώνουμε τα δεδομένα στο textbox txtdatarecieved

```
End Sub
```

```
Private Sub cmdSend_Click()
```

΄Συνάρτηση για την περίπτωση που ο server στείλει δεδομένα

```
Winsock1.SendData ("" & txtdatasend.Text)
```

΄Καλούμε την ιδίοτητα senddata με δεδομένα το κείμενο που θα εισάγουμε στο ΄textbox txtdatasend

```
txtdatasend = ""
```

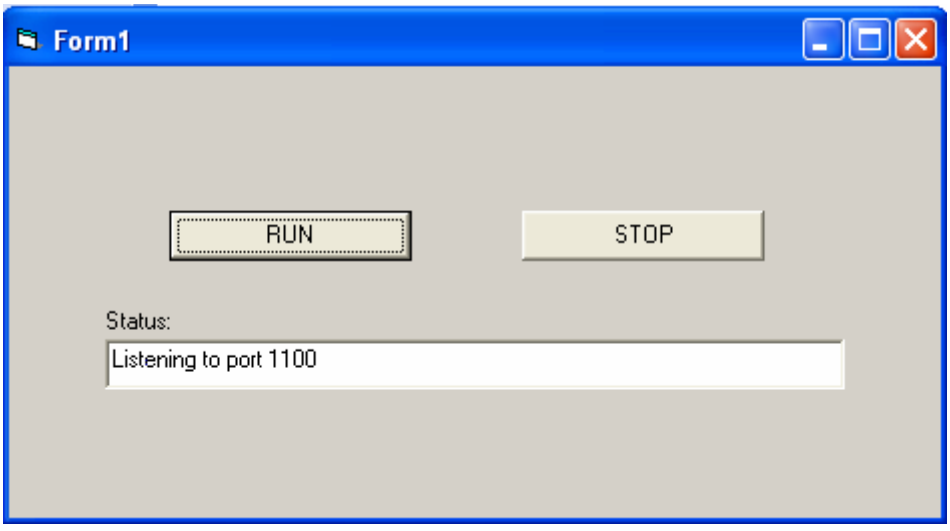
΄Σβήνουμε τα δεδομένα που εισάγαμε για αποστολή

```
End Sub
```

4.5 Server που δέχεται πολλαπλές συνδέσεις (δυναμικός τρόπος)

```
Option Explicit
'ορίζουμε μια γενική μεταβλητή που αντιπροσωπεύει τον αριθμό των winsock
Public NumSockets As Integer


Private Sub cmdrun_click()
    Winsock1(0).LocalPort = 1100
  'θέτουμε σαν localport την 1100
  txtstatus.Text = "Listening to port" + Str(Winsock1(0).LocalPort)
  'Τυπώνουμε μήνυμα στο textbox status
  Winsock1(0).Listen
  'καλούμε την ιδιότητα listen
End Sub

Private Sub cmdstop_click()
    Dim index As Integer
txtstatus.Text = "Connection Closed :" &
Winsock1(index).RemoteHostIP
    Winsock1(index).Close
  'κλείνουμε τη σύνδεση και τυπώνουμε το αντίστοιχο μήνυμα
End Sub


Private Sub Winsock1_ConnectionRequest(index As Integer, ByVal
requestID As Long)
```

```
    NumSockets = NumSockets + 1
  'αυξάνουμε τον αριθμό της πόρτας κατά 1
    Load Winsock1(NumSockets)
  'φορτώνουμε ένα νέο winsock με δείκτη την τιμή της μεταβλητής numsockets

    Winsock1(NumSockets).Accept requestID
  'αποδεχόμαστε την σύνδεση σε μια τυχαία πόρτα

End Sub

Private Sub Winsock1_DataArrival(index As Integer, ByVal
bytesTotal As Long)

Dim strData As String
  'ορίζουμε μια μεταβλητή όπου αποθηκεύουμε τα εισερχόμενα  δεδομένα
Winsock1(index).GetData strData, vbString
  'λαμβάνουμε τα δεδομένα και τα αποθηκεύουμε στη μεταβλητή

     Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\data.txt", True)

    txtfile.Write strData
  'αποθηκεύουμε τα δεδομένα μας σε ένα αρχείο κειμένου
End Sub
```



Εικόνα 6. Η εφαρμογή όταν εκτελείται

4.6 Server που δέχεται συγκεκριμένο αριθμό συνδέσεων (στατικός τρόπος)

```
Option Explicit
'ορίζουμε μια γενική μεταβλητή που αντιπροσωπεύει τον αριθμό της πόρτας
Public port As Integer

Private Sub cmdrun_Click()

  port = txtportstart.Text
'από την τιμή που θα εισάγουμε στο textbox txtportstart ορίζουμε την πόρτας από όπου ο server θα ανοίξει 10 πόρτες στις οποίες θα ακούει
 Winsock1.LocalPort = port
 Winsock1.Listen
 Winsock2.LocalPort = port + 1
 Winsock2.Listen
 Winsock3.LocalPort = port + 2
 Winsock3.Listen
 Winsock4.LocalPort = port + 3
 Winsock4.Listen
 Winsock5.LocalPort = port + 4
 Winsock5.Listen
 Winsock6.LocalPort = port + 5
 Winsock6.Listen
 Winsock7.LocalPort = port + 6
 Winsock7.Listen
 Winsock8.LocalPort = port + 7
 Winsock8.Listen
 Winsock9.LocalPort = port + 8
 Winsock9.Listen
 Winsock10.LocalPort = port + 9
 Winsock10.Listen
 'θέτουμε σε κάθε winsock την τιμή της πόρτας και καλούμε την ιδιότητα listen
 txtstatus.Text = "Server is listening for incoming conections"
 'τυπώνουμε το αντίστοιχο μήνυμα
End Sub

Private Sub cmdstop_click()

Winsock1.Close
Winsock2.Close
Winsock3.Close
Winsock4.Close
Winsock5.Close
Winsock6.Close
Winsock7.Close
```

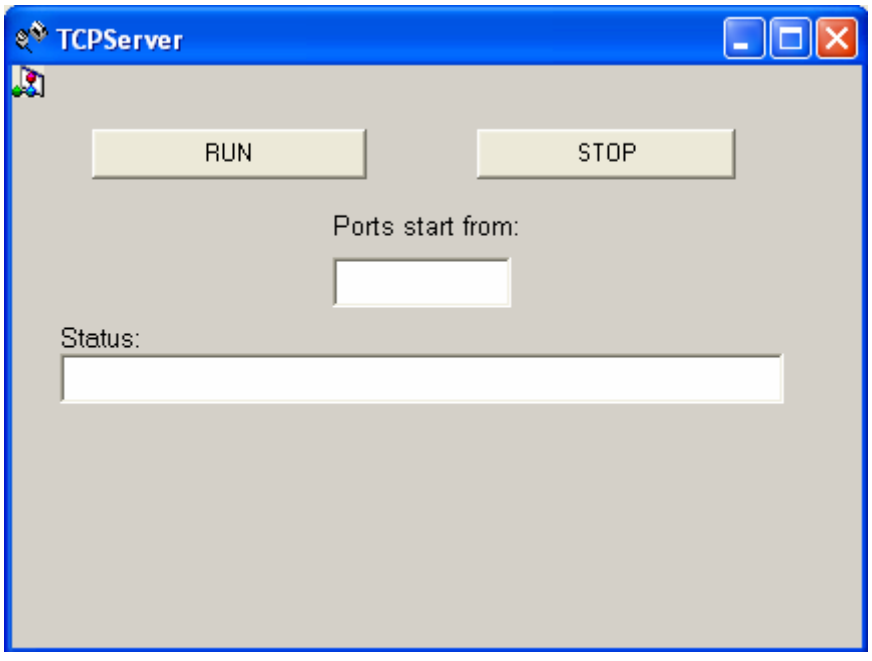
```
Winsock8.Close
Winsock9.Close
Winsock10.Close
txtstatus.Text = "Server is closed"
'κλείνουμε το server και τυπώνουμε το αντίστοιχο μήνυμα
End Sub


Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)

 If Winsock1.State <> sckClosed Then Winsock1.Close

 Winsock1.Accept requestID
'για κάθε winsock αποδεχόμαστε την αίτηση σύνδεσης από τον client
'τον ίδιο κώδικα τον επαναλαμβάνουμε για κάθε winsock
End Sub

Private Sub Winsock2_ConnectionRequest(ByVal requestID As Long)

 If Winsock2.State <> sckClosed Then Winsock2.Close

 Winsock2.Accept requestID

End Sub
Private Sub Winsock3_ConnectionRequest(ByVal requestID As Long)

 If Winsock3.State <> sckClosed Then Winsock3.Close

 Winsock3.Accept requestID

End Sub
Private Sub Winsock4_ConnectionRequest(ByVal requestID As Long)

 If Winsock4.State <> sckClosed Then Winsock4.Close

 Winsock4.Accept requestID

End Sub
Private Sub Winsock5_ConnectionRequest(ByVal requestID As Long)

 If Winsock5.State <> sckClosed Then Winsock5.Close

 Winsock5.Accept requestID

End Sub
Private Sub Winsock6_ConnectionRequest(ByVal requestID As Long)

 If Winsock6.State <> sckClosed Then Winsock6.Close
```

```
 Winsock6.Accept requestID

End Sub
Private Sub Winsock7_ConnectionRequest(ByVal requestID As Long)

 If Winsock7.State <> sckClosed Then Winsock7.Close

 Winsock7.Accept requestID

End Sub
Private Sub Winsock8_ConnectionRequest(ByVal requestID As Long)

 If Winsock8.State <> sckClosed Then Winsock8.Close

 Winsock8.Accept requestID

End Sub
Private Sub Winsock9_ConnectionRequest(ByVal requestID As Long)

 If Winsock9.State <> sckClosed Then Winsock9.Close

 Winsock9.Accept requestID

End Sub
Private Sub Winsock10_ConnectionRequest(ByVal requestID As Long)

 If Winsock10.State <> sckClosed Then Winsock10.Close

 Winsock10.Accept requestID

End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String
   'ορίζουμε μια μεταβλητή όπου αποθηκεύουμε τα εισερχόμενα δεδομένα
   Winsock1.GetData strData, vbString
   'καλούμε την ιδιότητα λήψης δεδομένων
   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client1.txt", True)
    txtfile.Write strData

    txtfile.Close
    'γράφουμε τα δεδομένα σε αρχείο κειμένου
    'τον ίδιο κώδικα επαναλαμβάνουμε για κάθε winsock
End Sub
```

```
Private Sub Winsock2_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock2.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client2.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock3_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock3.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client3.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock4_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock4.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client4.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock5_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock5.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
```

```
    Set txtfile = fso.CreateTextFile("c:\client5.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock6_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock6.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client6.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock7_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock7.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client7.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock8_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock8.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client8.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock9_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock9.GetData strData, vbString
```

```
   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client9.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
Private Sub Winsock10_DataArrival(ByVal bytesTotal As Long)
   Dim strData As String

   Winsock10.GetData strData, vbString

   Dim fso, txtfile

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile("c:\client10.txt", True)

    txtfile.Write strData
    txtfile.Close
End Sub
```



Εικόνα 7. Η εφαρμογή όταν εκτελείται

Κεφάλαιο 5ο

’’Παρατηρήσεις – Συμπεράσματα’’

5.1 Παρατηρήσεις και συμπεράσματα

Μια σημαντική παρατήρηση για το πρωτόκολλο TCP/IP είναι ότι δυο υπολογιστές εντελώς διαφορετικών χαρακτηριστικών και κατασκευαστών μπορούν επικοινωνήσουν κατευθείαν ο ένας με τον άλλον χωρίς να χρειάζεται μετατροπή δεδομένων από ένα πρωτόκολλο σε άλλο αφού και οι δυο χρησιμοποιούν το TCP/IP.

Επίσης κατανοώντας την λειτουργία του πρωτοκόλλου TCP/IP και στηριζόμενοι στο μοντέλο πελάτη – εξυπηρετητή μπορούμε να κατασκευάσουμε εφαρμογές χρησιμοποιώντας οποιαδήποτε από τα προγραμματιστικά εργαλεία που μας προσφέρουν δυνατότητα κατασκευής εφαρμογών δικτύου που στηρίζονται στο πρωτόκολλο TCP/IP.

Σε ότι αφορά την εφαρμογή της παρούσας πτυχιακής πέρα από το πεδίο που αφορά το δίκτυο μπορούμε να συμπεράνουμε πως η κατασκευή τέτοιων εφαρμογών χρειάζεται ιδιαίτερη προσοχή στα εξής πεδία:

- Το περιβάλλον της εφαρμογής πρέπει να είναι εύχρηστο για τον χρήστη.
- Κατά την χρήση της εφαρμογής θα πρέπει να γίνεται έλεγχος τυχόν σφαλμάτων του χρήστη και να προειδοποιείται για αυτά.
- Να ενημερώνεται ο χρήστης με αντίστοιχα μηνύματα σχετικά με την λειτουργία της εφαρμογής.

Έχοντας σαν βάση την εφαρμογή της παρούσας πτυχιακής μπορούμε να προγραμματίσουμε ποικίλες εφαρμογές που θα έχουν συγκεκριμένη λειτουργία όπως για παράδειγμα μια εφαρμογή όπου ο client θα στέλνει στο server συγκεκριμένα δεδομένα (π.χ. δεδομένα από μια σειριακή συσκευή η ένα αρχείο) και ο server με τη σειρά του θα δέχεται αυτά τα δεδομένα και θα τα αποθηκεύει σε συγκεκριμένη θέση και με συγκεκριμένη μορφή.

Παράρτημα Α’
’’Winsock control reference’’

BytesReceived Property

Returns the amount of data received (currently in the receive buffer). Use the **GetData** method to retrieve data.

Read-only and unavailable at design time.

Syntax

object.**BytesReceived**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Value

Long

LocalHostName Property

Returns the local machine name. Read-only and unavailable at design time.

Syntax

object.**LocalHostName**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Value

String

LocalIP Property

Returns the IP address of the local machine in the IP address dotted string format (xxx.xxx.xxx.xxx).Read-only and unavailable at design time.

Syntax

object.**LocalIP**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type

String

LocalPort Property

Returns or sets the local port to use. Read/Write and available at design time.

- For the client, this designates the local port to send data from. Specify port 0 if the application does not need a specific port. In this case, the control will select a random port. After a connection is established, this is the local port used for the TCP connection.
- For the server, this is the local port to listen on. If port 0 is specified, a random port is used. After invoking the **Listen** method, the property contains the actual port that has been selected.

Syntax

object.**LocalPort** = *long*

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type

Long

Remarks

Port 0 is often used to establish connections between computers dynamically. For example, a client that wishes to be "called back" by a server can use port 0 to procure a new (random) port number, which can then be given to the remote computer for this purpose.

Protocol Property (Winsock Control)

Returns or sets the protocol, either TCP or UDP, used by the **Winsock** control.

Syntax

object.**Protocol** *[=protocol]*

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Settings

The settings for *protocol* are:

Constant	Value	Description
sckTCPProtocol	0	Default. TCP protocol.
sckUDPProtocol	1	UDP protocol.

Return Value

Void

Remarks

The control must be closed (using the **Close** method) before this property can be reset.

RemoteHostIP Property

Returns the IP address of the remote machine.

- For client applications, after a connection has been established using the **Connect** method, this property contains the IP string of the remote machine.
- For server applications, after an incoming connection request (ConnectionRequest event), this property contains the IP string of the remote machine that initiated the connection.
- When using the UDP protocol, after the DataArrival event occurs, this property contains the IP address of the machine sending the UDP data.

Syntax

object.**RemoteHostIP**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type

String

SocketHandle Property

Returns a value that corresponds to the socket handle the control uses to communicate with the Winsock layer. Read-only and unavailable at design time.

Syntax

object.**SocketHandle**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type

Long

Remarks

This property was designed to be passed to Winsock APIs.

State Property (Winsock Control)

Returns the state of the control, expressed as an enumerated type. Read-only and unavailable at design time.

Syntax

object.**State**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type

Integer

Settings

The settings for the **State** property are:

Constant	Value	Description
sckClosed	0	Default. Closed
sckOpen	1	Open
sckListening	2	Listening
sckConnectionPending	3	Connection pending
sckResolvingHost	4	Resolving host
sckHostResolved	5	Host resolved
sckConnecting	6	Connecting
sckConnected	7	Connected
sckClosing	8	Peer is closing the connection
sckError	9	Error

Accept Method

For TCP server applications only. This method is used to accept an incoming connection when handling a ConnectionRequest event.

Syntax

object.**Accept** *requestID*

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type

Long

Return Value

Void

Remarks

The **Accept** method is used in the ConnectionRequest event. The ConnectionRequest event has a corresponding argument, the **RequestID** parameter, that should be passed to the **Accept** method. An example is shown below:

```
Private Sub Winsock1_ConnectionRequest _
(ByVal requestID As Long)
  ' Close the connection if it is currently open
  ' by testing the State property.
  If Winsock1.State <> sckClosed Then Winsock1.Close

  ' Pass the value of the requestID parameter to the
  ' Accept method.
  Winsock1.Accept requestID
End Sub
```

The **Accept** method should be used on a new control instance (other than the one that is in the listening state.)

Bind Method

Specifies the LocalPort and LocalIP to be used for TCP connections. Use this method if you have multiple protocol adapters.

Syntax

object.**Bind** *LocalPort, LocalIP*

The **Bind** method syntax has these parts

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
LocalPort	The port used to make a connection.
LocalIP	The local Internet address used to make a connection.

Remarks

You must invoke the **Bind** method before invoking the **Listen** method.

Close Method (Winsock Control)

Closes a TCP connection or a listening socket for both client and server applications.

Syntax

*object.***Close**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Arguments

None

Return Value

Void

GetData Method (WinSock Control)

Retrieves the current block of data and stores it in a variable of type variant.

Return Value

Void

Syntax

*object.***GetData** *data***,** [*type,*] [*maxLen*]

The **GetData** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.
type	Optional. Type of data to be retrieved, as shown in Settings.
maxLen	Optional. Specifies the desired size when receiving a byte array or a string. If this parameter is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this parameter is ignored.

Settings

The settings for *type* are:

Description	Constant
Byte	**vbByte**
Integer	**vbInteger**
Long	**vbLong**
Single	**vbSingle**
Double	**vbDouble**

Currency	**vbCurrency**
Date	**vbDate**
Boolean	**vbBoolean**
SCODE	**vbError**
String	**vbString**
Byte Array	**vbArray + vbByte**

Remarks

It's common to use the **GetData** method with the DataArrival event, which includes the *totalBytes* argument. If you specify a *maxlen* that is less than the *totalBytes* argument, you will get the warning 10040 indicating that the remaining bytes will be lost.

Listen Method

Creates a socket and sets it in listen mode. This method works only for TCP connections.

Syntax

object.**Listen**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Arguments

None

Return Value

Void

Remarks

The ConnectionRequest event occurs when there is an incoming connection. When handling ConnectionRequest, the application should use the **Accept** method (on a new control instance) to accept the connection.

PeekData Method

Similar to **GetData** except **PeekData** does not remove data from the input queue. This method works only for TCP connections.

Syntax

object.**PeekData** d*ata*, [t*ype*,] [*maxLen*]

The **PeekData** method syntax has these parts

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
data	Stores retrieved data after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.
type	Optional. Type of data to be retrieved, as described in Settings. Default Value: **vbArray** + **vbByte**.
maxLen	Optional. Length specifies the desired size when receiving a byte array or a string. If this argument is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this argument is ignored.

Settings

The settings for *type* are:

Type	**Constant**
Byte	**vbByte**
Integer	**vbInteger**

Long	**vbLong**
Single	**vbSingle**
Double	**vbDouble**
Currency	**vbCurrency**
Date	**vbDate**
Boolean	**vbBoolean**
SCODE	**vbError**
String	**vbString**
Byte Array	**vbArray** + **vbByte**

Return Value

Void

Remarks

If the type is specified as **vbString**, string data is converted to UNICODE before returning to the user.

SendData Method

Sends data to a remote computer.

Return Value

Void

Syntax

object.**SendData** *data*

The **SendData** method syntax has these parts

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	Data to be sent. For binary data, byte array should be used.

Remarks

When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

Close Event

Occurs when the remote computer closes the connection. Applications should use the **Close** method to correctly close a TCP connection.

Syntax

*object*_**Close( )**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Arguments

None

Connect Event (Winsock Control)

Occurs when a Connect operation is completed.

Syntax

object.Connect()

The *object* placeholder represents an object expression that evaluates to a **Winsock** control.

Remarks

Use the Connect event to confirm when a connection has been made successfully.

ConnectionRequest Event

Occurs when a remote machine requests a connection.

- For TCP server applications only. The event is activated when there is an incoming connection request. **RemoteHostIP** and **RemotePort** properties store the information about the client after the event is activated.

Syntax

*object*_**ConnectionRequest** (r*equestID* **As Long***)*

The ConnectionRequest event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
requestID	The incoming connection request identifier. This argument should be passed to the **Accept** method on the second control instance.

Remarks

The server can decide whether or not to accept the connection. If the incoming connection is not accepted, the peer (client) will get the Close event. Use the **Accept** method (on a new control instance) to accept an incoming connection.

DataArrival Event

Occurs when new data arrives.

Syntax

*object*_**DataArrival** (*bytesTotal* **As Long***)*

The DataArrival event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
bytesTotal	Long. The total amount of data that can be retrieved.

Remarks

This event will not occur if you do not retrieve all the data in one **GetData** call. It is activated only when there is new data. Use the **BytesReceived** property to check how much data is available at any time.

Error Event (Winsock Control)

Occurs whenever an error occurs in background processing (for example, failed to connect, or failed to send or receive in the background).

Syntax

*object*_**Error**(*number* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **as String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To

	list.
number	An integer that defines the error code. See Settings below for constants.
description	String containing error information.
Scode	The long SCODE
Source	String describing the error source.
HelpFile	String containing the help file name.
HelpContext	Help file context.
CancelDisplay	Indicates whether to cancel the display. The default is **False**, which is to display the default error message box. If you do not want to use the default message box, set CancelDisplay to **True**.

Settings

The settings for *number* are:

Constant	Value	Description
sckOutOfMemory	7	Out of memory
sckInvalidPropertyValue	380	The property value is invalid.
sckGetNotSupported	394	The property can't be read.
sckSetNotSupported	383	The property is read-only.
sckBadState	40006	Wrong protocol or connection state for the requested transaction or request.
sckInvalidArg	40014	The argument passed to a function was not in the correct format or in the specified range.
sckSuccess	40017	Successful.
sckUnsupported	40018	Unsupported variant type.
sckInvalidOp	40020	Invalid operation at current state
sckOutOfRange	40021	Argument is out of range.
sckWrongProtocol	40026	Wrong protocol for the requested

		transaction or request
sckOpCanceled	1004	The operation was canceled.
sckInvalidArgument	10014	The requested address is a broadcast address, but flag is not set.
sckWouldBlock	10035	Socket is non-blocking and the specified operation will block.
sckInProgress	10036	A blocking Winsock operation in progress.
sckAlreadyComplete	10037	The operation is completed. No blocking operation in progress
sckNotSocket	10038	The descriptor is not a socket.
sckMsgTooBig	10040	The datagram is too large to fit into the buffer and is truncated.
sckPortNotSupported	10043	The specified port is not supported.
sckAddressInUse	10048	Address in use.
sckAddressNotAvailable	10049	Address not available from the local machine.
sckNetworkSubsystemFailed	10050	Network subsystem failed.
sckNetworkUnreachable	10051	The network cannot be reached from this host at this time.
sckNetReset	10052	Connection has timed out when SO_KEEPALIVE is set.
sckConnectAborted	11053	Connection is aborted due to timeout or other failure.
sckConnectionReset	10054	The connection is reset by remote side.
sckNoBufferSpace	10055	No buffer space is available.
sckAlreadyConnected	10056	Socket is already connected.
sckNotConnected	10057	Socket is not connected.
sckSocketShutdown	10058	Socket has been shut down.
sckTimedout	10060	Socket has been shut down.

sckConnectionRefused	10061	Connection is forcefully rejected.
sckNotInitialized	10093	WinsockInit should be called first.
sckHostNotFound	11001	Authoritative answer: Host not found.
sckHostNotFoundTryAgain	11002	Non-Authoritative answer: Host not found.
sckNonRecoverableError	11003	Non-recoverable errors.
sckNoData	11004	Valid name, no data record of requested type.

SendComplete Event

Occurs when a send operation is completed.

Syntax

*object*_**SendComplete**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Arguments

None

SendProgress Event

Occurs while data is being sent.

Syntax

*object*_**SendProgress** (*bytesSent* **As Long**, *bytesRemaining* **As Long**)

The SendProgress event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
bytesSent	The number of bytes that have been sent since the last time this event was activated.
bytesRemaining	The number of bytes in the send buffer waiting to be sent.

Παράρτημα Β’
’’Using MsgBox on Visual Basic’’

Prompting the User with Dialog Boxes

In Windows-based applications, dialog boxes are used to prompt the user for data needed by the application to continue or to display information to the user. Dialog boxes are a specialized type of form object that can be created in one of three ways:

- *Predefined* dialog boxes can be created from code using the MsgBox or InputBox functions.
- *Customized* dialog boxes can be created using a standard form or by customizing an existing dialog box.
- *Standard* dialog boxes, such as Print and File Open, can be created using the common dialog control.

Figure 3.24 shows an example of a predefined dialog box created using the MsgBox function.

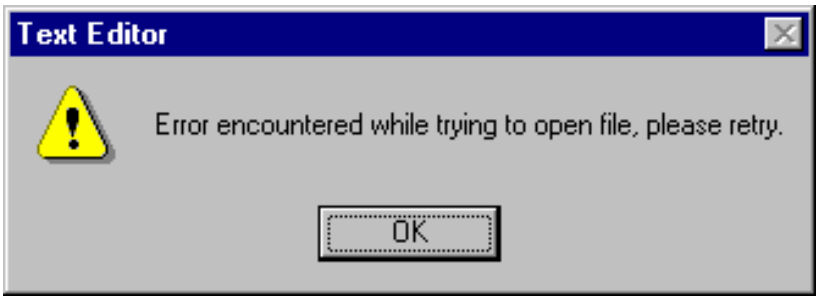


Figure 3.24 A predefined dialog box

This dialog is displayed when you invoke the MsgBox function in code. The code for displaying the dialog box shown in Figure 3.24 looks like this:

```
MsgBox "Error encountered while trying to open file," & vbCrLf & "please retry.", vbExclamation, "Text Editor"
```

You supply three pieces of information, or arguments, to the MsgBox function: the message text, a constant (numeric value) to determine the style of the dialog box, and a title. Styles are available with various combinations of buttons and icons to make creating dialog boxes easy.

Because most dialog boxes require user interaction, they are usually displayed as modal dialog boxes. A *modal* dialog box must be closed (hidden or unloaded) before you can continue working with the rest of the application. For example, a dialog box is

modal if it requires you to click OK or Cancel before you can switch to another form or dialog box.

Modeless dialog boxes let you shift the focus between the dialog box and another form without having to close the dialog box. You can continue to work elsewhere in the current application while the dialog box is displayed. Modeless dialog boxes are rare; you will usually display a dialog because a response is needed before the application can continue. From the Edit menu, the Find dialog box in Visual Basic is an example of a modeless dialog box. Use modeless dialog boxes to display frequently used commands or information.

MsgBox Function

Displays a message in a dialog box, waits for the user to click a button, and returns an **Integer** indicating which button the user clicked.

Syntax

MsgBox(*prompt*[, *buttons*] [, *title*] [, *helpfile*, *context*])

The **MsgBox** function syntax has these named arguments:

Part	Description
prompt	Required. String expression displayed as the message in the dialog box. The maximum length of ***prompt*** is approximately 1024 characters, depending on the width of the characters used. If ***prompt*** consists of more than one line, you can separate the lines using a carriage return character (**Chr(**13**)**), a linefeed character (**Chr(**10**)**), or carriage return – linefeed character combination (**Chr(**13**)** & **Chr(**10**)**) between each line.
buttons	Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If omitted, the default value for ***buttons*** is 0.
title	Optional. String expression displayed in the title bar of the dialog box. If you omit ***title***, the application name is placed in the title bar.
helpfile	Optional. String expression that identifies the Help file to use to provide context-sensitive Help for the dialog box. If ***helpfile*** is provided, ***context*** must also be provided.
context	Optional. Numeric expression that is the Help context number assigned to the appropriate Help topic by the Help author. If ***context*** is provided,

	helpfile must also be provided.

Settings

The ***buttons*** argument settings are:

Constant	Value	Description
vbOKOnly	0	Display **OK** button only.
vbOKCancel	1	Display **OK** and **Cancel** buttons.
vbAbortRetryIgnore	2	Display **Abort**, **Retry**, and **Ignore** buttons.
vbYesNoCancel	3	Display **Yes**, **No**, and **Cancel** buttons.
vbYesNo	4	Display **Yes** and **No** buttons.
vbRetryCancel	5	Display **Retry** and **Cancel** buttons.
vbCritical	16	Display **Critical Message** icon.
vbQuestion	32	Display **Warning Query** icon.
vbExclamation	48	Display **Warning Message** icon.
vbInformation	64	Display **Information Message** icon.
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default.
vbApplicationModal	0	Application modal; the user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal; all applications are suspended until the user responds to the message box.
vbMsgBoxHelpButton	16384	Adds Help button to the message box
VbMsgBoxSetForeground	65536	Specifies the message box window as the foreground window

vbMsgBoxRight	524288	Text is right aligned
vbMsgBoxRtlReading	1048576	Specifies text should appear as right-to-left reading on Hebrew and Arabic systems

The first group of values (0–5) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0, 256, 512) determines which button is the default; and the fourth group (0, 4096) determines the modality of the message box. When adding numbers to create a final value for the ***buttons*** argument, use only one number from each group.

Note These constants are specified by Visual Basic for Applications. As a result, the names can be used anywhere in your code in place of the actual values.

Return Values

Constant	Value	Description
vbOK	1	**OK**
vbCancel	2	**Cancel**
vbAbort	3	**Abort**
vbRetry	4	**Retry**
vbIgnore	5	**Ignore**
vbYes	6	**Yes**
vbNo	7	**No**

Remarks

When both ***helpfile*** and ***context*** are provided, the user can press F1 to view the Help topic corresponding to the **context**. Some host applications, for example, Microsoft Excel, also automatically add a **Help** button to the dialog box.

If the dialog box displays a **Cancel** button, pressing the ESC key has the same effect as clicking **Cancel**. If the dialog box contains a **Help** button, context-sensitive Help is provided for the dialog box. However, no value is returned until one of the other buttons is clicked.

Note To specify more than the first named argument, you must use **MsgBox** in an expression. To omit some positional arguments, you must include the corresponding comma delimiter.

Βιβλιογραφία :

Τεχνολογία δικτύων επικοινωνιών - Κ. Αρβανίτης , Γ. Κόλυβας , Σ. Ούτσιος
TCP/IP Network Administration - O' Reilly
MSDN Library Visual Studio 6.0a - Microsoft
Client Server computing – Sams Publishing

Πηγές διαδικτύου:

http://msdn.microsoft.com