ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΊΔΡΥΜΑ (Α.Τ.Ε.Ι.) ΗΠΕΙΡΟΥ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

-

ΤΜΗΜΑ ΤΗΛΕΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Πτυχιακή Εργασία:

«Η γλώσσα προγραμματισμού Perl, το πρότυπο CGI και το μοντέλο πελάτη-εξυπηρετητή (client-server).»

ΠΑΠΑΓΕΩΡΓΙΟΥ ΕΥΤΥΧΙΑ

Επιβλέπων καθηγητής: Λάππας Κωνσταντίνος

ΑΡΤΑ 2003

ΕΙΣΑΓΩΓΗ

Στην εργασία μου ,που ακολουθεί, ανέπτυξα το θέμα "Γλώσσα προγραμματισμού Perl και Μοντέλο client-server". Βασικό θέμα της εργασίας αποτελεί το client-server μοντέλο και ειδικότερα το πως υλοποιούνται οι εφαρμογές αυτού του τύπου μέσω του CGI και της Perl.

Πιο συγκεκριμένα στο πρώτο κεφάλαιο έγινε μια αναφορά με απαρίθμηση των βασικότερων σταθμών στη διαδρομή εξέλιξης της Perl. Επιπλέον δόθηκε μια γενική άποψη του τρόπου λειτουργίας και εκτέλεσης των προγραμμάτων που είναι γραμμένα σε αυτήν.

Στη συνέχεια της εργασίας στο επόμενο κεφάλαιο γίνεται μια αναφορά στο μοντέλο CGI, το οποίο είναι βασικό για την δημιουργία προγραμμάτων σε γλώσσες όπως η Perl, καθώς και στην αρχιτεκτονική CGI και την ασφάλεια που υπάρχει σε αυτά τα συστήματα.

Παρακάτω αναλύεται το μοντέλο client-server δίνοντας μεγάλη βάση και στα δυο μέρη. Δίνονται όλες οι κατάλληλες πληροφορίες για την κατανόηση της λειτουργίας καθώς και των εφαρμογών χρήσης του συγκεκριμένου μοντέλου. Αναλυτικότερα δίνονται πληροφορίες για το πως δημιουργήθηκε η τεχνολογία του πελάτη-εξυπηρετητή και ακολουθεί αναλυτική περιγραφή των συστατικών του δηλαδή λεπτομερής αναφορά και επεξήγηση του πελάτη (client) και του εξυπηρετητή (server) για το πως επιτυγχάνεται η μεταξύ τους επικοινωνία αλλά και για τα διάφορα στοιχεία από τα οποία αποτελούνται όπως το λειτουργικό σύστημα που χρησιμοποιούν και το υλικό και λογισμικό από το οποίο αποτελούνται και χρησιμοποιούν αντίστοιχα.

Στο αμέσως επόμενο κεφάλαιο γίνεται αναφορά στα κυριότερα εργαλεία υλοποίησης client-server εφαρμογών.

Τέλος, δίνεται ένα παράδειγμα λειτουργίας του client-server μοντέλου ως

εφαρμογή των όσων αναλύθηκαν στην εργασία. Το παράδειγμα δίνεται με τη μορφή κώδικα σε Perl, ο κώδικας είναι σχολιασμένος και επιπλέον πληροφορίες και λεπτομέρειες δίνονται σε μια παράγραφο που εξηγεί αναλυτικά τη διαδικασία του προγράμματος.

1. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PERL

1.1 Η ΙΣΤΟΡΙΑ ΤΗΣ PERL

Ο Larry Wall δημιούργησε την Perl το 1987. Από την αρχική της μορφή είχε εφαρμογή σε Unix συστήματα. Μέσα στα τέσσερα πρώτα χρόνια ύπαρξης της η Perl, στην μορφή που βρισκόταν , δεν είχε επιπλέον πράγματα να δώσει στις συνεχώς αυξανόμενες απαιτήσεις του αγοραστικού κοινού, αυτό συνέβαινε γιατί η Perl παρουσίαζε πολλά προβληματικά και συγκεχυμένα σημεία που έκαναν δυσχερή τη χρήση της. Το γεγονός αυτό οδήγησε στην έρευνα για την εξέλιξη της γλώσσας στα σημεία εκείνα που υστερούσε.

Σαν αποτέλεσμα όλων αυτών ήρθε η Perl 5 , έκδοση η οποία επιβεβαίωσε όσους είχαν πιστέψει στις δυνατότητες τις γλώσσας και που την καθιέρωσε στον χώρο του προγραμματισμού ως μια από τις καλύτερες και πιο εύχρηστες γλώσσες.

Χαρακτηριστικό είναι πως πολλοί την χαρακτηρίζουν ως «φυσική» γλώσσα και όχι γλώσσα μηχανής. Οι φυσικές γλώσσες έχουν την καταπληκτική ικανότητα να προϋποθέτουν σύστημα επικοινωνίας για ανθρώπους όλων των επιπέδων και ικανοτήτων. Η φυσική γλώσσα επιτρέπει το ίδιο την επικοινωνία ενός παιδιού μόλις τριών χρονών με τη μητέρα του για να της εκφράσει κάτι που επιθυμεί , αλλά και προσφέρει την δυνατότητα σε φιλόσοφους και μεγάλους στοχαστές να συγγράψουν. Είναι εμφανές το πόσο περιορισμένο λεξιλόγιο θα χρησιμοποιηθεί στον πρώτο διάλογο και πόσο εξεζητημένο στον δεύτερο.

Με παρόμοιο τρόπο συγκρίνουμε τα προγράμματα της Perl. Ακόμη και ένας αρχάριος που επιθυμεί να μάθει Perl μέσα σε πολύ λίγο χρονικό διάστημα είναι σε θέση να φτιάξει ένα απλό πρόγραμμα όμως υπάρχουν σύνθετα πολύπλοκα προγράμματα που απαιτούν πολλές γνώσεις αλλά και εμπειρία στον προγραμματισμό.

1.2 ΕΙΣΑΓΩΓΗ ΣΤΗΝ PERL

Η Perl μπορεί να λύσει κάποιο πρόβλημα που θα προκύψει. Φυσικά αν προγραμματίζεις το πρόβλημα αυτό μπορεί θεωρητικά να λυθεί από οποιαδήποτε ολοκληρωμένη γλώσσα προγραμματισμού. Όμως γνωρίζουμε πως οι γλώσσες προγραμματισμού διαφέρουν όχι τόσο στο τι κάνουν πιθανό όσο στο τι κάνουν εύκολο.

Η Perl διαφέρει από τις άλλες γλώσσες στο ότι έχει σχεδιαστεί έτσι ώστε να λύνει εύκολα τα «εύκολα» προβλήματα χωρίς όμως να κάνει τα «δύσκολα» προβλήματα να μοιάζουν αδύνατα.

Η γλώσσα στην οποία προγραμματίζεις πρέπει να μπορεί να υπολογίζει αριθμητικές πράξεις, να κάνει σύνταξη κειμένου , να φτιάχνει αρχεία , να υποστηρίζει δικτυακές εφαρμογές και γενικώς να μπορεί να λύσει πολύπλοκα ζητήματα. Βασικό όμως είναι να μπορείς να τρέξεις τα προγράμματά σου σε οποιοδήποτε καλό μηχάνημα.

Η Perl κάνει όλα αυτά και ακόμη πολλά περισσότερα.

Σημαντικό ρόλο στην αλματώδη εξέλιξη της Perl έπαιξε και η μεγάλη επιθυμία των χρηστών Unix να συνεχίσουν να προγραμματίζουν σε γλώσσα εξελιγμένη αλλά και συμβατή με το Unix. Γι' αυτούς η Perl τους δίνει την δυνατότητα να μπορέσουν να μεταφέρουν την συνέχιση του Unix. Από την άλλη μεριά όμως λειτουργεί αντίθετα στο γεγονός ότι τα προγράμματα που είναι γραμμένα σε Perl δεν μπορούν να εκτελεστούν χωρίς αλλαγές σε Unix προγράμματα.

Η Perl μπορεί να τρέξει σχεδόν σε οποιοδήποτε λειτουργικό σύστημα και δεν παρουσιάζει πολλά από τα προβλήματα που παρουσιάζουν άλλες γλώσσες, όπως να είναι απαραίτητο να χρησιμοποιείς σε κάθε πρόγραμμα που φτιάχνεις την εντολή #ifdef όταν προγραμματίζεις σε γλώσσα προγραμματισμού C ή C++ για να είναι δυνατό να μπορεί να διαβαστεί από όλα τα λειτουργικά συστήματα που μπορεί να χρησιμοποιηθούν. Επιπλέον συνδυάζει πολλά από τα πλεονεκτήματα που έχουν οι

άλλες γλώσσες με κάποια τα οποία έχει μόνο η Perl.

Η μαγεία της Perl βασίζεται στην χρησιμότητα της ίδιας της γλώσσας, την εφευρητικότητα της και την μεγάλη αφθονία που προσφέρει στις προγραμματιστικές κινήσεις που μπορείς να κάνεις χρησιμοποιώντας την Perl.

Η Perl είναι μία γλώσσα με αναμειγμένα κληρονομικά στοιχεία από διάφορες άλλες γλώσσες και πολλούς άλλους παράγοντες. Έχει καταφέρει να επικρατεί η άποψη πως η ποικιλία είναι περισσότερο προτέρημα παρά μειονέκτημα.

Η Perl απευθύνεται σε ευρύ κοινό εφόσον είναι μια γενικού σκοπού προγραμματιστική γλώσσα. Έχει πλούσιο και ολοκληρωμένο λογισμικό περιβάλλον , αναφορές , βιβλιοθήκες, συντακτικό καθώς και όλα τα άλλα που διαθέτει μια πλήρης γλώσσα. Όλα αυτά κάνουν τα δύσκολα να γίνονται πιθανά όμως η Perl είναι μοναδική στο ότι δεν έχασε (σε καμία από τις εκδόσεις της) την ικανότητα να κάνει με ευκολία τα εύκολα πράγματα.

Είναι εξίσου δυναμική και εύκολη στην πρόσβαση γλώσσα. Χρησιμοποιείται σε πάρα πολλές εφαρμογές από επισκευές σε διαστημόπλοια μέχρι σε εφαρμογές σε πειράματα βιολογίας! Ακόμη βρίσκει εφαρμογή στην επιστήμη των μαθηματικών, σε γραφιστικά, επεξεργασία κειμένου, υπολογισμό σε βάσεις δεδομένων και σε διαχείριση δικτύων.

Χρησιμοποιείται από ανθρώπους που χρειάζονται την γρήγορη και έγκυρη ανάλυση μεγάλης ποσότητας δεδομένων σε πολύ λίγο χρόνο. Μία επιχείρηση μπορεί να χρησιμοποιεί την Perl βασιζόμενη στο ότι θα μπορεί να έχει εξασφαλισμένη την λύση των προβλημάτων που θα θέλει να λύσει. Επιπλέον θα μπορεί να είναι σίγουρη πως ποτέ δεν θα είναι εκτός χρόνου εφόσον η Perl είναι μία γλώσσα που αναβαθμίζεται και θα μπορεί πάντα να χρησιμοποιείται.

Εκτός από το πλούσιο και δωρεάν λογισμικό πακέτο που προσφέρει μπορεί να εγγυηθεί και για την διασκέδαση των χρηστών. Κάνει τους χρήστες να νιώθουν

δημιουργικοί προσφέροντάς τους ελευθερία έκφρασης. Επιλέγουν τις συνθήκες που επιθυμούν να προγραμματίσουν και τις εφαρμόζουν. Τέτοιες συνθήκες είναι η ταχύτητα στην οποία θα προγραμματίζουν , η περιεκτικότητα η προσβασιμότητα και η ικανότητα ανάγνωσης.

Δίνει σε τέτοιο βαθμό ελευθερία γιατί είναι ξεχωριστή γλώσσα. Είναι συγχρόνως μία πολύ απλή αλλά και πολύ πλούσια γλώσσα. Συλλέχτηκαν οι πιο καλές ιδέες από διάφορα μέρη και έχουν προσαρμοστεί με τον καλύτερο δυνατό τρόπο στην Perl για την κάλυψη των αναγκών του σύγχρονου προγραμματιστή.

Εξ ορισμού η Perl είναι μία Πρακτική Εξαγωγική και Αναφορική Γλώσσα. Κατά πολλούς τρόπους η Perl είναι μία απλή γλώσσα. Δεν είναι απαραίτητο να έχεις εξεζητημένες γνώσεις για να μπορέσεις να εκτελέσεις ένα πρόγραμμα τμηματικά. Οι τύποι και οι περιορισμοί της γλώσσας είναι εύκολοι και κατανοητοί. Δεν επιβάλλει αυθαίρετα όρια στα δεδομένα και δεν περιορίζει το μέγεθος των strings και των πινάκων. Αντί να βάζει στη διαδικασία εκμάθησης νέου συντακτικού τους χρήστες η Perl δανείζεται στοιχεία από πολλές γλώσσες όπως η C, BASIC, τα Αγγλικά και τα Ελληνικά. Έτσι μπορείς να κατανοήσεις εύκολα κώδικα γραμμένο σε Perl.

Για να είναι σε θέση να κάνει πιθανά τα δύσκολα πράγματα η Perl είναι απαραίτητο να διαθέτει και πλούτο εκτός από απλότητα. Είναι δύσκολο και αρκετά χρονοβόρο να μπορέσεις να απορροφήσεις όλα αυτά που μπορεί να κάνει η Perl αλλά είναι πολύτιμες οι προεκτάσεις αυτής της γλώσσας και αξίζει πραγματικά τον όποιο κόπο και θυσίες για να γευθείς την χαρά και ικανοποίηση της απόκτησης μιας τόσο σημαντικής γνώσης.

Η Perl διέθετε όλο αυτόν τον πλούτο και πριν την μεγάλη της εξέλιξη όπου έγινε γλώσσα ικανή για χειρισμό αρχείων, διαχείριση βάσεων δεδομένων, για προγράμματα επικοινωνίας πελάτη-διαχειριστή, προγραμματισμό ασφαλείας, διαδικτυακό και αντικειμενοστραφή λειτουργικό προγραμματισμό.

Κάθε δυνατότητα που προσφέρει η Perl είναι αρμονική και με σχέση συνεργασίας με

τις υπόλοιπες πράγμα που βοηθά και παροτρύνει την ένταξη νέων δυνατοτήτων.

Και όμως η Perl έχει τη δυνατότητα και για περαιτέρω επέκταση. Επιτρέποντας την αυτόματη σχεδόν σχεδίαση προγραμμάτων. Η Perl μπορεί να εισχωρήσει σε άλλες γλώσσες και το αντίστροφο.

1.3 Ο ΤΡΟΠΟΣ ΕΚΤΕΛΕΣΗΣ ΤΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ

Η Perl αρχικά μεταφράζει όλο το πρόγραμμα γρήγορα σε ένα διαμεσολαβητικό σχήμα. Ο compiler δίνει στη στιγμή ανατροφοδότηση από το συντακτικό μέχρι και σε σημαντικά λάθη όπως αυτά που γίνονται στις βιβλιοθήκες.

Αν το πρόγραμμα αντιμετωπιστεί σωστά περνάει από την διαμεσολάβηση στον μεταφραστή για να το εκτελέσει. Μπορεί να ακούγονται λίγο μπερδεμένα όλα αυτά αλλά το σίγουρο είναι πως η διαδικασία από τον compiler μέχρι την εκτέλεση είναι αποδοτική και διαρκεί ελάχιστα δευτερόλεπτα μόνο.

Με αυτή την γλώσσα προγραμματισμού είναι επίσης πιο εύκολο να γράφεις ασφαλή προγράμματα.

Επιπρόσθετα σε όλες τις τυπικές διαδικασίες προστασίας που ακολουθούν όλες οι γλώσσες η Perl δρα εναντίων των τυχαίων λαθών μέσω ενός μοναδικού μηχανήματος που απορρίπτει τα δεδομένα που έρχονται από επισφαλής πηγές. Έτσι προλαμβάνει τις προβληματικές καταστάσεις .

Διαθέτει ειδικά προφυλαγμένα «διαμερίσματα» στα οποία μπορούν να εκτελεστούν τα προγράμματα χωρίς να είναι δυνατή η πρόσβαση σε επιβλαβή στοιχεία.

2. ΤΟ ΜΟΝΤΕΛΟ CGI (*COMMON GATEWAY INTERFACE*)

Το CGI αποτελεί μια προδιαγραφή για τη μεταφορά δεδομένων μεταξύ ενός World

Wide Web server και ενός προγράμματος. Τα προγράμματα CGI είναι ο δημοφιλέστερος τρόπος αμφίδρομης, δυναμικής επικοινωνίας μεταξύ ενός Web server και του χρήστη.

Για παράδειγμα, πολλές από τις σελίδες HTML που περιέχουν φόρμες προς συμπλήρωση, χρησιμοποιούν ένα πρόγραμμα CGI για την επεξεργασία των δεδομένων που εισάγει ο χρήστης. Ένα πρόγραμμα CGI είναι σχεδιασμένο για να δέχεται και να επιστρέφει δεδομένα που πληρούν τις αντίστοιχες προδιαγραφές. Το πρόγραμμα μπορεί να είναι γραμμένο σε οποιαδήποτε γλώσσα προγραμματισμού, όπως C, Perl, Java ή Visual Basic.

Όταν η πληροφορία συλλέγεται από τον browser, στέλνεται σ'ενα HTTP server (Hyper Text Transfer Protocol) ο οποίος καθορίζεται από την HTML φόρμα. Αυτός ο server ξεκινάει ένα πρόγραμμα CGI, το οποίο περιγράφεται επίσης στην HTML φόρμα και επεξεργάζεται την πληροφορία. Όταν το πρόγραμμα εκτελείται προετοιμάζει ένα HTML έγγραφο και στέλνει αυτό το έγγραφο στον client, ο οποίος το απεικονίζει όπως ακριβώς θα έκανε με κάθε άλλο HTML έγγραφο.

Σε μερικούς HTTP servers αυτά τα CGI προγράμματα αποθηκεύονται σε ένα κατάλογο με το όνομα cgi-bin, γιαυτό καλούνται "cgi-bin scripts".

Οι servers έχουν υποστεί μετατροπές, έτσι ώστε να μπορούν να ξεκινήσουν αμέσως το CGI πρόγραμμα το οποίο καθορίζεται στη φόρμα και να προωθήσουν τα συγκεντρωμένα δεδομένα στο πρόγραμμα, το οποίο μπορεί να ετοιμάσει μια απάντηση (πιθανότατα με το να ανατρέξει σε μια υπάρχουσα βάση δεδομένων) και να επιστρέψει ένα έγγραφο στο χρήστη.

Για παράδειγμα ένας client, ο οποίος τρέχει σε κάποιο υπολογιστή, παίρνει μια φόρμα από κάποιον server που τρέχει σε άλλο υπολογιστή.Ο client εμφανίζει τη φόρμα, ο χρήστης εισάγει δεδομένα και ο client στέλνει τις εισαχθείσες πληροφορίες στον server. Εκεί τα δεδομένα μεταβιβάζονται σε ένα CGI πρόγραμμα το οποίο προετοιμάζει ένα έγγραφο και το στέλνει στον client. Ο client τότε εμφανίζει αυτό το

έγγραφο.

Μία απο τις δυνάμεις πίσω απο την ανάπτυξη του Common Gateway Interface ήταν η επιθυμία για ολοκληρωμένες βάσεις δεδομένων μέσα στο Δίκτυο. Υπάρχουν πολλές διαφορετικές προσεγγίσεις και τα CGI είναι ευρέως διαδεδομένα.

Τα πλεονεκτήματα μιας CGI προσέγγησης είναι :

1 Ένας client μπορεί να χρησιμοποιηθεί σαν front end μιας πολλαπλής βάσης δεδομένων
2 Μια βάση δεδομένων μπορεί να επικοινωνήσει με πολλαπλούς clients, καθένας με τα αρχικά χαρακτηριστικά της πλατφόρμας του.
3 Αλλάζοντας το query μοντέλο της βάσης δεδομένων δεν απαιτεί αλλαγή όλων των clients στο πεδίο, παρά μόνο την φόρμα δεδομένων η οποία προσπελαύνεται απο τους χρήστες.

Και, φυσικά υπάρχουν και ορισμένες δυσκολίες :

1 Το interface δεν υποστηρίζει άπειρο set τύπων δεδομένων
2 Το interface φορμών είναι οργανωμένο σε φόρμες και όχι σε πεδία, με αποτέλεσμα να μην είναι τόσο εύρωστο όσο θα μπορούσε να γίνει :
    - δεν υποστηρίζει απο την μεριά του client εκτεταμένο έλεγχο για τιμές δεδομένων, και
    - απαιτεί, ο χρήστης να πατήσει ένα submit πλήκτρο για κάθε εμπλοκή του server.
3 Περιήγηση μεταξύ διαφόρων πεδίων εισόδου μπορεί να είναι αδέξια σε ορισμένες πλατφόρμες

Ένα ακόμη πρόβλημα των προγραμμάτων CGI είναι ότι σε κάθε εκτέλεση ενός script, ξεκινά μια νέα διαδικασία. Εάν υπάρχουν πολλοί χρήστες ταυτόχρονα σε έναν Web server και εκτελούνται πολλά scripts, ο server μπορεί να υπερφορτωθεί.

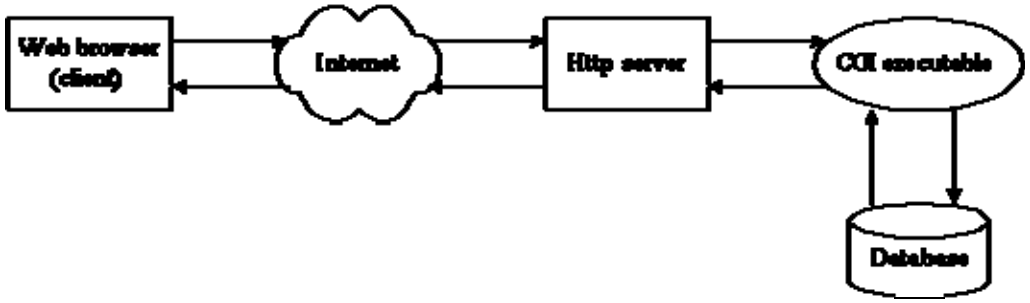
Το CGI μπορεί να υλοποιηθεί χρησιμοποιώντας διάφορα εργαλεία ανάπτυξης εφαρμογών βάσεων δεδομένων όπως για παράδειγμα το ESQL/C API, η C++ και η

Perl.

Η πρόσβαση σε βάσεις δεδομένων μέσω κατάλληλων CGIs έχει το πλεονέκτημα ότι είναι απλή στην υλοποίησή της. Επίσης, δεν επιφέρει καμία αλλαγή ή τροποποίηση στον Web server. Έτσι ένα CGI executable είναι ικανό να τρέξει σε όλους τους Web servers.

2.1 Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ CGI

Στο μοντέλο αυτό ο client (χρήστης) τοποθετεί σε μία HTML φόρμα το query (ερώτηση προς την βάση) που επιθυμεί και την στέλνει (την ερώτηση) στον Web server. Ο server καλεί κάποιο CGI script το οποίο παίρνει το request. Η αρχιτεκτονική του μοντέλου αυτού φαίνεται στο παρακάτω σχήμα :



Σχήμα 1. Η αρχιτεκτονική CGI

Το CGI πρόγραμμα επεξεργάζεται το query προκειμένου να δημιουργήσει την αντίστοιχη standard SQL ερώτηση προς την βάση. Έπειτα το CGI προωθεί την SQL ερώτηση στην βάση και περιμένει για τα αποτελέσματα. Καθώς το CGI λαμβάνει τα αποτελέσματα από την βάση τα μετατρέπει σε standard HTML έτσι ώστε να γίνουν ορατά από τον Web browser.Το CGI κατόπιν παραδίδει τα επεξεργασμένα αποτελέσματα στον Http server ο οποίος με τη σειρά του τα στέλνει στον browser.

2.2 Η ΑΣΦΑΛΕΙΑ ΣΤΟ CGI

Τα περισσότερα "παραθυράκια" στην ασφάλεια ενός συστήματος δεν είναι σκόπιμα.

Συνήθως οφείλονται σε άτομα που δεν έχουν την απαιτούμενη εμπειρία να γράψουν ασφαλή CGI scripts. Όσο περισσότεροι είναι οι χρήστες που έχουν την δυνατότητα να γράψουν scripts, τόσο μεγαλύτερη είναι η πιθανότητα ότι ένα από αυτά τα scripts θα περιέχει ένα σημαντικό λάθος. Γι' αυτό το λόγο κανένα script δεν πρέπει να εγκαθίσταται στον server, εάν δεν το έχει ελέγξει πρώτα κάποιος ειδικός.

Για την αποφυγή λαθών πρέπει να ακολουθούνται τα παρακάτω βήματα:

1 Ο Web server πρέπει να ρυθμιστεί ώστε όλα τα CGI scripts να αποθηκεύονται σε ένα κατάλογο (τυπικά ο κατάλογος αυτός είναι ο cgi-bin). Η πρακτική σύμφωνα με την οποία όλα τα αρχεία με επέκταση ".cgi" λαμβάνονται σαν CGI scripts δεν συνιστάται.
2 Συνιστάται η χρήση του προγράμματος Tripwire (ή οποιοδήποτε παρόμοιου) για επίβλεψη των αλλαγών που γίνονται στα scripts.
3 Η πρόσβαση στο κατάλογο με τα CGI scripts πρέπει να είναι περιορισμένη. Δεν πρέπει να επιτρέπεται στους τοπικοί χρήστες να εγκαθιστούν ή να αφαιρούν script ή να τροποποιούν τα υπάρχοντα χωρίς την επίβλεψη του διαχειριστή. Επίσης, καλή ιδέα είναι και η αφαίρεση του δικαιώματος ανάγνωσης τους, ώστε οι χρήστες του Διαδικτύου να μην έχουν την δυνατότητα να ανιχνεύσουν τυχόν ατέλειες.
4 Τα backup αρχεία που αυτόματα παράγουν κάποιοι διορθωτές κειμένου, πρέπει να σβήνονται. Ξεχασμένα τέτοια αρχεία μπορούν να εκτελεστούν από ένα εισβολέα με ανεπιθύμητα αποτελέσματα.

3. ΤΟ ΜΟΝΤΕΛΟ ΠΕΛΑΤΗ-ΕΞΥΠΗΡΕΤΗΤΗ (CLIENT-SERVER)

Γενικά, το client-server computing αναφέρεται σε μία βασική αλλαγή στο στυλ των υπολογιστών, την αλλαγή από τα συστήματα που βασίζονται στα μηχανήματα στα συστήματα που βασίζονται στον χρήστη.

Ειδικότερα ένα σύστημα client-server είναι ένα σύστημα στο οποίο το δίκτυο ενώνει

διάφορους υπολογιστικούς πόρους, ώστε οι clients (ή αλλιώς front end) να μπορούν να ζητούν υπηρεσίες από έναν server (ή αλλιώς back end), ο οποίος προσφέρει πληροφορίες ή επιπρόσθετη υπολογιστική ισχύ.

Με άλλα λόγια στο client-server μοντέλο, ο client θέτει μία αίτηση και ο server επιστρέφει μία ανταπόκριση ή κάνει μία σειρά από ενέργειες. Ο server μπορεί να ενεργοποιείται άμεσα για την αίτηση αυτή ή να προσθέτει την αίτηση σε μία ουρά. Η άμεση ενεργοποίηση για την αίτηση μπορεί , για παράδειγμα, να σημαίνει ότι ο server υπολογίζει έναν αριθμό και τον επιστρέφει αμέσως στον client. Η τοποθέτηση της αίτησης σε μία ουρά μπορεί να σημαίνει ότι η αίτηση μπορεί να τεθεί σε αναμονή για να εξυπηρετηθεί. Ένα καλό παράδειγμα για αυτό είναι όταν εκτυπώνουμε ένα κείμενο σε έναν εκτυπωτή δικτύου. Ο server τοποθετεί την αίτηση σε μία ουρά μαζί με αιτήσεις εκτυπώσεων και από άλλους clients. Μετά επεξεργάζεται την αίτηση με βάση την σειρά προτεραιότητας, η οποία , σε αυτή την περίπτωση, καθορίζεται από την σειρά με την οποία ο server παρέλαβε την αίτηση.

Το client-server computing είναι πολύ σημαντικό διότι επιτυγχάνει τα εξής :

1 Αποτελεσματική χρήση της υπολογιστικής ισχύος
2 Μείωση του κόστους συντήρησης, δημιουργώντας συστήματα client-server που απαιτούν λιγότερη συντήρηση και κοστίζουν λιγότερο στην αναβάθμιση.
3 Αύξηση της παραγωγικότητας, προσφέροντας στους χρήστες πρόσβαση στις αναγκαίες πληροφορίες μέσω σταθερών και εύκολων στη χρήση διασυνδέσεων.
4 Αύξηση της ευελιξίας και της δυνατότητας δημιουργίας συστημάτων που υποστηρίζουν πολλά περιβάλλοντα.

Με βάση αυτούς τους σκοπούς , οι οργανισμοί που κινούνται προς την κατεύθυνση της client-server τεχνολογίας αυξάνουν κατά πολύ την ανταγωνιστική τους θέση.

3.1 ΤΟ ΒΑΣΙΚΟ CLIENT-SERVER ΜΟΝΤΕΛΟ

Η πλευρά του client πρώτα στέλνει ένα μήνυμα για να καλέσει σε ετοιμότητα τον server. Από την στιγμή που ο client και ο server έχουν επικοινωνία μεταξύ τους, ο

client μπορεί να υποβάλλει την αίτησή του.

Ο client είναι ο αιτών των υπηρεσιών. Ο client δεν μπορεί παρά να είναι ένας υπολογιστής. 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 ξεκινά πάντα την επικοινωνία.

Τα συστατικά του client είναι πολύ απλά. Μια client μηχανή πρέπει να μπορεί να κάνει τα ακόλουθα:

1 Να τρέχει το λογισμικό των γραφικών διεπαφών χρηστών (GUIs)
2 Να δημιουργεί τις αιτήσεις για πληροφορίες και να τις στέλνει στον server.
3 Να αποθηκεύει τις επιστρεφόμενες πληροφορίες.

Αυτές οι αιτήσεις καθορίζουν πόση μνήμη χρειάζεται, ποια ταχύτητα επεξεργασίας θα μπορούσε να βελτιώσει τον χρόνο ανταπόκρισης, και πόση χωρητικότητα αποθήκευσης απαιτείται.

Ο server απαντάει στις αιτήσεις που γίνονται από τους clients. Ένας client μπορεί να ενεργεί ως server εάν λαμβάνει και επεξεργάζεται αιτήσεις όπως ακριβώς τις στέλνει (για παράδειγμα, ένας σταθμός που χρησιμοποιείται και ως server εκτυπώσεων από άλλους). Οι server δεν ξεκινάνε τις επικοινωνίες -περιμένουν τις αιτήσεις των clients.
Τα συστατικά του server είναι πολύ απλά. Μια server μηχανή μπορεί να κάνει τα ακόλουθα:

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

3.2 ΠΩΣ ΑΝΑΠΤΥΧΘΗΚΕ Η CLIENT-SEVER ΤΕΧΝΟΛΟΓΙΑ;

Η τεχνολογία των υπολογιστών αναπτύχθηκε βαθμιαία, με τέτοιο τρόπο που κάθε

καινούργια αρχιτεκτονική έπαιρνε τα πλεονεκτήματα από τις τεχνικές που ήδη υπήρχαν, ώστε να εκμεταλλεύεται όλες τις δυνατότητες των υπολογιστών. Σήμερα οι υπολογιστές είναι μικρότεροι, γρηγορότεροι και φθηνότεροι από ότι παλιότερα. Σαν αποτέλεσμα, η γενική κατεύθυνση είναι η διανομή της επεξεργασίας της πληροφορίας αλλά και της ίδια της πληροφορίας σε ένα πλήθος αυτών των νέων υπολογιστών.

Ο όρος αρχιτεκτονική συνήθως χρησιμοποιείται για να περιγράφει συστήματα διαχείρισης βάσεων δεδομένων, λειτουργικά συστήματα και άλλους υπολογιστικούς μηχανισμούς λογισμικού και υλικού. Οι αρχιτεκτονικές περιγράφουν πως οι συσκευές και τα λογισμικά πακέτα ταιριάζουν για να φτιάξουν ένα εύκολο στη χρήση και διαχείριση σύνολο.

Η κλασσική αρχιτεκτονική αποτελείται από έναν υπολογιστή μεγάλης ισχύος (που παίζει τον ρόλο του οικοδεσπότη) με ένα ή περισσότερα απλά τερματικά. Οι εφαρμογές ελέγχονται και διανέμονται από τον υπολογιστή «οικοδεσπότη». Σε αυτόν πραγματοποιούνται όλες οι διαχειρίσεις πληροφοριών, η λογική των εφαρμογών και η μορφοποίηση της εμφάνισης τους. Οι χρήστες αλληλεπιδρούν με το κεντρικό σύστημα μέσω των τερματικών, τα οποία εμφανίζουν μόνο πληροφορίες. Αυτή είναι η πιο συνηθισμένη αρχιτεκτονική σήμερα.

Στην client-server αρχιτεκτονική, 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).

Το περιβάλλον του client-server έχει πολλά πλεονεκτήματα σε σχέση με τις κλασσικές αρχιτεκτονικές. Η διαχείριση της διασύνδεσης των χρηστών και άλλες επεξεργασίες είναι αποφορτισμένα από τον «οικοδεσπότη», ενώ ο server ακόμη προσφέρει συγκεντρωμένο έλεγχο των κοινών πόρων. 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. Ο φόρτος εργασίας μειώνεται δραματικά στον υπολογιστή «οικοδεσπότη» όσο αυξάνεται η ισχύς κάθε σταθμού εργασίας.

Οι οργανισμοί έχουν να κάνουν με συνεχώς περισσότερα δεδομένα, τα οποία πρέπει να τα διαχειρίζονται και να τα εκμεταλλεύονται στις εργασίες τους . Η αύξηση του όγκου των δεδομένων, σε συνδυασμό με την προσπάθεια των οργανισμών να μειώσουν το κόστος, να αυξήσουν την παραγωγικότητα και να βελτιώσουν τις υπηρεσίες των πελατών(με καλύτερη χρήση πληροφοριών και ταχύτερο χρόνο ανταπόκρισης στους πελάτες ταυτόχρονα), έχουν συμβάλει σε μία ώθηση για δημιουργία και χρήση client-server εφαρμογών.
Η αλματώδης εξέλιξη στην τεχνολογία του μοντέλου client-server έχει οδηγήσει στην πρόοδο του υλικού, λογισμικού και δικτύου.

3.3 ΣΥΣΤΑΤΙΚΑ ΤΟΥ CLIENT-SERVER COMPUTING : Ο CLIENT

Για να σχεδιάσουμε το client τμήμα μιας εφαρμογής, που είναι γνωστό και ως front end, είναι απαραίτητο να καταλάβουμε τα διάφορα συστατικά που το απαρτίζουν. Το υλικό(hardware), το λειτουργικό σύστημα(operating system), το δίκτυο (network), η γραφική διεπαφή του χρήστη(graphical user interface) και το λογισμικό(software) είναι απαραίτητα για να υποστηρίξουν και να δημιουργήσουν μία εφαρμογή.

3.3.1 ΤΟ ΥΛΙΚΟ

Ο client πρέπει να είναι σε θέση να χειρίζεται την εφαρμογή. Με άλλα λόγια, ο client πρέπει να έχει αρκετή δύναμη για να απαιτήσει, να παρουσιάσει και να χειριστεί τις πληροφορίες. Παρατηρούμε δηλαδή ότι υπάρχουν τέσσερις σημαντικές προϋποθέσεις όταν καθορίζονται οι ανάγκες σε υλικό:

1 Η ισχύς του επεξεργαστή
2 Η ταχύτητα του επεξεργαστή

3 Η ποσότητα της RAM
4 Η κάρτα οθόνης (VGA)

3.3.2 ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ

Το λειτουργικό σύστημα κρύβει τις λεπτομέρειες του υλικού του υπολογιστή από τον client.

Τα λειτουργικά συστήματα είναι προγράμματα που διαχειρίζονται τους πόρους του υπολογιστή, ελέγχουν την εκτέλεση εφαρμογών και ενεργούν ως μία διασύνδεση μεταξύ του χρήστη και του ίδιου του υλικού του υπολογιστή. Τα λειτουργικά συστήματα κάνουν τον υπολογιστή πιο αποτελεσματικό και κατάλληλο για χρήση, παρά το γεγονός ότι τα ίδια λειτουργικά συστήματα δεν είναι τίποτα περισσότερο από προγράμματα. Τα λειτουργικά συστήματα εκτελούν τις ακόλουθες λειτουργίες:

1 Ελέγχουν την διαχείριση των πόρων που μετακινούν, αποθηκεύουν, επεξεργάζονται και ελέγχουν πληροφορίες.
2 Φορτώνουν οδηγίες και πληροφορίες στην κύρια μνήμη, αρχικοποιούν αρχεία και συσκευές I/O(εισόδου / εξόδου) και προετοιμάζουν τους πόρους.
3 Ελέγχουν την πρόσβαση στα αρχεία, συμπεριλαμβανομένου την μορφοποίηση και την διαθεσιμότητα των πληροφοριών, όπως και την πρόσβαση σε απασχολούμενους πόρους.
4 Παρέχει τις οδηγίες και τα συνθήματα ελέγχου των διαφόρων συσκευών I/O (εισόδου/ εξόδου).
5 Ελέγχει την πρόσβαση στο σύστημα σαν σύνολο.

Ακόμη τα λειτουργικά συστήματα χαρακτηρίζονται από τρεις βασικές δυνατότητες.

→ Την δυνατότητα να διευθύνει την RAM
→ Την δυνατότητα να φορτώνει και να εκτελεί εφαρμογές ταυτόχρονα.
→ Την δυνατότητα να υποστηρίζει ή να παρέχει σταθερή διασύνδεση στους χρήστες.

Το GUI(Graphical User Interface) παρέχει στον χρήστη την όψη και την αίσθηση της εφαρμογής. Το GUI μπορεί να είναι τμήμα του ίδιου του λειτουργικού

συστήματος. Σε μερικές περιπτώσεις, η εφαρμογή δημιουργείται για ένα συγκεκριμένο λειτουργικό σύστημα. Ωστόσο, μερικές φορές το λειτουργικό σύστημα επιλέγεται, αφού επιλεχθεί το περιβάλλον διασύνδεσης των χρηστών.

Τα πιο συνηθισμένα λειτουργικά συστήματα που χρησιμοποιούνται σε client μηχανές είναι τα : DOS, Windows, OS/2, System 7, UNIX.

3.3.3 ΤΟ ΔΙΚΤΥΟ

Ένα δίκτυο είναι ένα σύστημα επικοινωνίας που επιτρέπει την μεταφορά των πληροφοριών μεταξύ των επεξεργαστών. Το δίκτυο έχει κανόνες ή πρωτόκολλα, που καθορίζουν πόσες πληροφορίες μεταφέρονται. Χρησιμοποιώντας σταθερά πρωτόκολλα και τυποποιημένες πληροφορίες, διαφορετικές πλατφόρμες υλικού και λογισμικού μπορούν να επικοινωνήσουν η μία με την άλλη.

Τα δίκτυα έχουν λογισμικά λειτουργικών συστημάτων, όπως συμβαίνει με τους clients και τους servers. Τα λειτουργικά συστήματα δικτύων καλύπτουν τις client εφαρμογές από την άμεση επικοινωνία με τον server. Αν και το λειτουργικό σύστημα δικτύου είναι εγκατεστημένα στον server, μέρος του πρέπει να τρέξει σε κάθε client.

Το λειτουργικό σύστημα δικτύου συνδέει το λειτουργικό σύστημα του client με το δίκτυο, που σημαίνει ότι εφαρμογές μπορούν να προσπελάσουν το δίκτυο μέσω των λειτουργικών συστημάτων των clients. Για παράδειγμα, μπορούμε να σώσουμε ένα αρχείο κατευθείαν σε μία τοπική διαδρομή στον server του δικτύου.

3.3.4 ΓΡΑΦΙΚΗ ΔΙΕΠΑΦΗ

Η γραφική διεπαφή χρήστη (Graphical User Interface) προσφέρει στον χρήστη μία εύκολη στη χρήση διασύνδεση. Με τη γραφική διεπαφή χρήστη (GUI) οι χρήστες δεν έχουν να κάνουν τίποτε περισσότερο από το να «σημειώνουν και να επιλέξουν» για να κάνουν τη δουλειά 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 απασχόλησή τους με τους προσωπικούς υπολογιστές τους. Η διασύνδεση καθορίζει πώς οι χρήστες εισάγουν πληροφορίες και πως οι εφαρμογές επιστρέφουν πληροφορίες στους χρήστες.

3.3.5 ΛΟΓΙΣΜΙΚΟ

Το λογισμικό μπορεί να υπάρχει στον client. Εδώ το λογισμικό σημαίνει την λογική του client στην client-server εφαρμογή, όπως τα άλλα προγράμματα, όπως τα λογιστικά φύλλα, τα γραφικά και τα προγράμματα του υπολογιστή που μπορούν ή δεν μπορούν να χρησιμοποιηθούν σε σύνδεση με την client-server εφαρμογή.

Συνήθως η λογική της client εφαρμογής προσφέρει ερώτημα για μορφοποίηση πληροφοριών και υπηρεσίες αναφοράς ώστε ο χρήστης να μπορεί να απαιτήσει πληροφορίες, να λάβει πληροφορίες, να μεταβάλλει τις πληροφορίες αυτές και να πάρει αναφορές που συνοψίζουν ή καταγράφουν λεπτομερώς αυτές τις πληροφορίες.

3.3.6 Ο ΡΟΛΟΣ ΤΟΥ CLIENT

Η διαδικασία client-server μπορεί να απλοποιηθεί στα ακόλουθα βήματα:

i. Ο χρήστης δημιουργεί μία αίτηση ή ένα ερώτημα.
ii. Ο client μορφοποιεί το ερώτημα και το στέλνει στο server.
iii. Ο server ελέγχει την δυνατότητα πρόσβασης του χρήστη.
iv. Ο server επεξεργάζεται το ερώτημα και επιστρέφει τα αποτελέσματα.
v. Ο client λαμβάνει την ανταπόκριση και την μορφοποιεί για τον χρήστη.
vi. Ο χρήστης βλέπει και χειρίζεται την πληροφορία.

Πέρα από τα έξι αυτά βήματα, ο client παίζει τέσσερις βασικούς ρόλους. Ο client είναι στην πραγματικότητα το κέντρο της client-server εφαρμογής. Ο χρήστης αλληλεπιδρά με τον client, ο client ξεκινάει το μεγαλύτερο μέρος της ανάπτυξης της εφαρμογής, και ο server υπάρχει για να απαντάει στις ανάγκες του client.

Ο client εκτελεί τις ακόλουθες λειτουργίες :

1 Προσφέρει μια εύκολη στη χρήση διασύνδεση χρηστών
2 Στέλνει απαιτήσεις.
3 Δέχεται ανταποκρίσεις.
4 Επιτρέπει στον χρήστη να βλέπει και να χειρίζεται τις πληροφορίες.

3.3.7 ΠΑΡΟΧΗ ΜΙΑ ΕΥΚΟΛΗΣ ΣΤΗ ΧΡΗΣΗ ΔΙΑΣΥΝΔΕΣΗΣ

Μια εύκολη στη χρήση διασύνδεση αποτελείται από δύο σημαντικές εργασίες: αποδοχή των εισερχομένων και εμφάνιση των εξερχόμενων. Για παράδειγμα, ο client δέχεται τα εισερχόμενα, επιτρέποντας σε κάποιον που τροφοδοτεί κάποια πράγματα να διαθέσει μία ειδική παραγγελία σε έναν πελάτη. Ο client μπορεί επίσης να εμφανίσει τις πληροφορίες του πελάτη στον τροφοδότη.

Η διασύνδεση των χρηστών είναι ένα από τα πιο σημαντικά κομμάτια της client εφαρμογής. Ελέγχει την όψη (τα στοιχεία της οθόνης) και την αίσθηση (τον τρόπο που ο χρήστης κάνει αιτήσεις και παίρνει απαντήσεις) του προγράμματος.

Οι αρχές σχεδίασης στις οποίες βασίζεται η ανάπτυξη του client είναι οι εξής:
Διατηρεί τη διασύνδεση συνεπή, ώστε οι χρήστες να πάρουν μία οικεία όψη και αίσθηση από τις εφαρμογές και τις πλατφόρμες.

Δεν ξεχνά ότι ο υπολογιστής εξυπηρετεί τον χρήστη. Ο χρήστης θα πρέπει να ελέγχει την σειρά των εργασιών. Ο υπολογιστής δεν θα πρέπει ποτέ να αγνοεί τον χρήστη, αλλά να τον διατηρεί ενήμερο και να του προσφέρει άμεσες απαντήσεις.

Χρησιμοποιεί μεταφορές, τόσο φραστικές όσο και οπτικές για να βοηθήσει τους χρήστες να αναπτύξουν θεμελιώδεις απεικονίσεις. Για παράδειγμα, η αποθήκευση

αρχείων σε φακέλους στον υπολογιστή, ώστε ο χρήστης να μπορεί να συγκεντρωθεί στη δουλειά του.

Δεν ζητάει από τον χρήστη να θυμάται εντολές. Οι εντολές μπορούν να είναι διαθέσιμες στον χρήστη για να τις επιλέγει, ώστε ο χρήστης να μπορεί να βασίζεται στην αναγνώριση ,παρά στην απομνημόνευση.

Επιτρέπει στην διασύνδεση να συγχωρεί τα λάθη του χρήστη. Οι καταστροφικές ενέργειες απαιτούν επιβεβαίωση, και οι χρήστες μπορούν να ανατρέψουν ή να ακυρώσουν την τελευταία ενέργεια.

Πρότυπα: Μία πρότυπη εγκατεστημένη διασύνδεση παρέχει εγγύηση ότι οι κατευθυντήριες γραμμές έχουν δοκιμαστεί για συνέπεια και εύκολη αποτελεσματική χρήση. Η IBM, η Microsoft και η Macintosh ανήκουν στα διαθέσιμα πρότυπα.

Στον υπολογιστή, οι εφαρμογές και τα αντικείμενα αναπαρίστανται ως εικονίδια(μικρές γραφικές εικόνες). Το εικονίδιο είναι προκαθορισμένο να ξεκινάει το πρόγραμμα, το οποίο εμφανίζει το ίδιο παράθυρο, στο οποίο η εφαρμογή ή η συγκεκριμένη εργασία μπορεί να εκτελεστεί. Για παράδειγμα, ένα πρόγραμμα εκτύπωσης μπορεί να αναπαρασταθεί από μια μικρή εικόνα ενός εκτυπωτή. Χρησιμοποιώντας έναν δείκτη για να κάνουμε διπλό κλικ στο εικονίδιο, η εφαρμογή μπορεί τότε να ξεκινήσει. Η εφαρμογή εκτύπωσης ανοίγει ένα παράθυρο στο οποίο η ουρά εκτύπωσης μπορεί να εμφανιστεί. Τα εικονίδια , οι δείκτες και τα παράθυρα είναι μερικά από τα στοιχεί του GUIs. Άλλα στοιχεία είναι: οι μπάρες ολίσθησης, οι κέρσορες και η βοήθεια.

3.3.8 ΑΠΟΣΤΟΛΗ ΑΙΤΗΣΕΩΝ

Το να σταλεί μία αίτηση σημαίνει την μορφοποίηση της αίτησης και την αποστολή της στον server, με τρόπο τέτοιο που να μπορεί ο τελευταίος να καταλάβει. Εάν ο τροφοδότης θέλει να εμφανίσει όλες τις απλήρωτες ειδικές παραγγελίες, ο client

μορφοποιεί την αίτηση σε SQL που χρησιμοποιείται από τον server DBMS (Data Management System) και την στέλνει μέσω του δικτύου στον server.

Το βασικό πλεονέκτημα των συστημάτων client-server είναι ότι η λογική της εφαρμογής και η βάση δεδομένων είναι χωρισμένα. Ο διαχωρισμός αυτός προσφέρει πέντε ευδιάκριτα αποτελέσματα:

Από τη στιγμή που η βασική επεξεργασία γίνεται στον server, ο client δεν χρειάζεται τόση πολύ ισχύ, και οι πόροι δεν δεσμεύονται μετά την αποστολή των απαιτήσεων.

Διαχωρίζοντας τη λογική, μειώνεται ο φόρτος στο δίκτυο, επειδή το δίκτυο δεν χρειάζεται να διαβιβάζει ολόκληρα αρχεία πίσω και μπροστά. Χρησιμοποιώντας SQL απλοποιούνται τα προβλήματα στο δίκτυο με ερωτήματα και απαντήσεις. Αυτή η απλοποίηση μπορεί να είναι τεράστια, ειδικά σε μεγάλα δίκτυα με πολλούς clients.

Οι χρήστες δεν περιορίζονται σε μια client πλατφόρμα. Η SQL τυποποιεί τα ερωτήματα ώστε οι πληροφορίες να μπορούν να μεταφέρονται από μια πλατφόρμα σε μια άλλη με ξεκάθαρο τρόπο.

Έχοντας τη βάση δεδομένων στον server, διαφυλάσσεται η ακεραιότητα των πληροφοριών. Είναι δυνατό να προσφέρονται υπηρεσίες, όπως ασφάλεια και προστασία πληροφοριών, δημιουργία αντιγράφων ασφαλείας κτλ.. Αυτό το επίπεδο προστασίας των πληροφοριών γίνεται πιο δύσκολο, όσο ο έλεγχος γίνεται πιο αποκεντρωμένος(όπως με ένα πιο κατανεμημένο σύστημα).

Η εκτέλεση της επεξεργασίας ενημερώνει όλες τις αλλαγές που έγιναν στη βάση δεδομένων σε μια χρονική περίοδο για να βεβαιωθεί ότι οι τροποποιήσεις έχουν καταγραφεί κανονικά. Η ενημέρωση μπορεί επίσης να χρησιμοποιηθεί για να ανακτηθεί η βάση δεδομένων σε περίπτωση που «πέσει» το σύστημα.

Το πώς ο χρήστης κάνει την αίτηση εξαρτάται από τον client. Γενικά, ο client προτρέπει τον χρήστη να μπει στα πεδία για να ψάξει και μετά δημιουργεί την

πραγματική απαίτηση σε SQL. Για παράδειγμα , ένας τροφοδότης θέλει να δημιουργήσει μια λίστα από συγκεκριμένους πελάτες που περιμένουν να παραλάβουν ειδικές παραγγελίες που πραγματοποιήθηκαν πάνω από 30 ημέρες πριν. Τα κριτήρια του ερωτήματος θα περιλαμβάνουν συγκεκριμένους πελάτες που περιμένουν να παραλάβουν ειδικές παραγγελίες που πραγματοποιήθηκαν πάνω από 30 ημέρες πριν.

Οι πληροφορίες μπορούν να τοποθετηθούν με τρεις βασικούς τρόπους:

1 Επικεντρωμένες στον server.
2 Τοποθετημένες σε πολλαπλά πανομοιότυπα αντίγραφα σε διαφορετικές τοποθεσίες.
3 Χωρισμένες σε διάφορες τοποθεσίες.

Η συγκέντρωση των πληροφοριών στον server προσφέρει περισσότερο έλεγχο, διότι υπάρχει μόνο ένα αντίγραφο για να προσπελάσουμε και να διατηρήσουμε, και περιέχει όλες τις απαραίτητες πληροφορίες. Ωστόσο, εάν ο server «πέσει», οι πληροφορίες δεν είναι διαθέσιμες. Επίσης, συγκεντρώνοντας τις πληροφορίες, η δραστηριότητα στο δίκτυο αυξάνεται, διότι όλοι πρέπει να προσπελάσουν τις ίδιες πληροφορίες στην ίδια περιοχή.

Τα βασικά πλεονεκτήματα του είναι να έχουμε πολλαπλά πανομοιότυπα αντίγραφα, είναι η βελτίωση στην επίδοση και η ασφάλεια της ύπαρξης περισσοτέρων από ένα αντιγράφων σε περίπτωση που ο server «πέσει». Όταν δύο ομάδες χρηστών σε διαφορετικούς κόμβους σε ένα δίκτυο, προσπελάσουν την ίδια πληροφορία, μπορεί να βελτιωθεί σημαντικά η επίδοση όταν υπάρχουν δύο αντίγραφα της πληροφορίας, ένα σε κάθε κόμβο.

Υπάρχουν δύο επιλογές όταν υπάρχουν πολλαπλά αντίγραφα:

1 Οι χρήστες μπορούν να αντιγράφουν τις απαιτούμενες πληροφορίες στις δικές τους μηχανές και να δουλέψουν πάνω σε αυτές εκεί.
2 Ένα σύστημα διαχείρισης βάσεων δεδομένων μπορεί να διανείμει αντίγραφα της πληροφορίας σε τακτικά διαστήματα.

3.3.9 ΛΗΨΗ ΑΠΟΚΡΙΣΗΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

Ο server ελέγχει τη δυνατότητα πρόσβασης του χρήστη, επεξεργάζεται το ερώτημα, και επιστρέφει την αιτούμενη πληροφορία. Σε αυτό το σημείο, ο client δέχεται τα αποτελέσματα και τα μετατρέπει σε μία μορφή που μπορεί να χρησιμοποιήσει ο client.

Η διαδικασία της λήψης της απόκρισης είναι παρόμοια με τη διαδικασία που χρησιμοποιεί ο client για να στείλει αιτήσεις, αλλά αντιστροφή.

Μετά, το API (Application Programming Interface) εξάγει την αιτούμενη πληροφορία και την περνάει στην εφαρμογή. Με άλλα λόγια, αντί να μορφοποιεί την αίτηση και να την στέλνει στον client, ο client δέχεται την απόκριση και την μορφοποιεί για τον χρήστη. Το τελευταίο βήμα είναι η δυνατότητα στον χρήστη να βλέπει και να χειρίζεται την πληροφορία.

Από την στιγμή που η πληροφορία επιστρέφει, το πλεονέκτημα είναι ότι ο client θα θέλει να πάρει κάποια πρωτοβουλία. Σε αυτό το σημείο η εφαρμογή υιοθετεί ξανά τον πρώτο της ρόλο: προσφέρει μια εύκολη στη χρήση αλληλεπίδραση για να δεχθεί εισερχόμενα δεδομένα και να εμφανίσει εξερχόμενα.

3.4 ΣΥΣΤΑΤΙΚΑ ΤΟΥ CLIENT-SERVER COMPUTING: Ο SERVER

Αν και ο client κατέχει ένα μεγάλο μέρος της προσοχής, ο server είναι η καρδιά του client-server συστήματος. Οι servers είναι τα σημεία όπου αποθηκεύονται οι πληροφορίες και εκτελούνται οι εργασίες. Σήμερα, ο server μπορεί να είναι οποιαδήποτε μορφή υπολογιστή. Ωστόσο η αύξηση της ισχύος και η μείωση του κόστους των προσωπικών υπολογιστών τους κάνει γενικά την πιο συμφέρουσα οικονομικά επιλογή. Ακόμα και αν ο server είναι ένας σταθερός προσωπικός υπολογιστής αυτό που κάνει τη διαφορά από ένα σταθερό προσωπικό σύστημα είναι ότι είναι εξειδικευμένος και έχει συγκεκριμένες πρωτοβουλίες.

3.4.1 ΤΥΠΟΙ ΤΩΝ SERVERS

Οι servers μπορούν να διαιρεθούν σε έξι τύπους:

1 Server Εφαρμογών (Application servers)
2 Server Πληροφοριών (Data servers)
3 Server Υπολογισμών (Compute servers)
4 Server Βάσεων Δεδομένων (Database servers)
5 Server Πόρων ή Επικοινωνιών (Resource or Communication servers)

Ο τύπος του server που χρησιμοποιείται εξαρτάται από την απαιτούμενη εργασία. Επίσης, αυτοί οι έξι ρόλοι μπορούν να συνδυαστούν σε ένα σύστημα ή να διαιρεθούν σε περισσότερα. Για παράδειγμα, η ίδια μηχανή μπορεί να εξυπηρετήσει σαν ένας server εφαρμογών και ένας server βάσεων δεδομένων.
Οι περισσότεροι servers που χρησιμοποιούνται σήμερα στις επιχειρήσεις είναι servers αρχείων (file servers). Οι servers αρχείων επιτρέπουν στους clients να προσπελάσουν αρχεία και να μοιραστούν πληροφορίες και λογισμικό. Αυτοί οι servers είναι συνήθως ένας προσωπικός υπολογιστής ή ένα UNIX σύστημα με έναν επεξεργαστή. Πολλοί άνθρωποι μπορούν να προσπελάσουν τον server αρχείων την ίδια στιγμή, που σημαίνει ότι ο server έχει πολλαπλές μονάδες δίσκων και κάρτες προσαρμογής δικτύου, αλλά μόνο ένα άτομο μπορεί να προσπελάσει ένα συγκεκριμένο αρχείο εκείνη τη στιγμή.

Για να σχεδιάσουμε το server τμήμα της εφαρμογής, που είναι γνωστό και ως back end, πρέπει να καταλάβουμε τα διάφορα συστατικά που απαρτίζουν τον server. Το υλικό, το λειτουργικό σύστημα, η βάση δεδομένων και το λογισμικό πρέπει να υποστηρίζουν και να δουλέψουν για την εφαρμογή.

3.4.2 ΤΟ ΥΛΙΚΟ

Μια μηχανή δεν απαιτεί πολύ ειδικό υλικό για να μετατραπεί σε server, αν και ορισμένοι servers έχουν κάποιες συγκεκριμένες απαιτήσεις. Συνήθως ο server που επιλέγεται, εξαρτάται από τις εφαρμογές που θα τρέξουμε και από το κόστος της μηχανής.

Τα βασικά χαρακτηριστικά του server είναι τα ακόλουθα:

1 Ο server μπορεί να ανταποκριθεί σε ταυτόχρονες αιτήσεις για εξυπηρέτηση πολλών clients.
2 Ο server είναι αξιόπιστος, διότι οι clients εξαρτώνται από αυτόν.
3 Ο server μπορεί να αυξομειώνεται, διότι οι client-server εφαρμογές τείνουν να έχουν ανάγκη όλο και περισσότερης μνήμης και ισχύς για επεξεργασία.

3.4.3 ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ

Τα λειτουργικά συστήματα των servers διαχειρίζονται τους πόρους του υπολογιστή, ελέγχουν την εκτέλεση των εφαρμογών και ενεργούν ως προστασία ανάμεσα στις servers εφαρμογές και στους clients.

Οι servers για client-server εφαρμογές λειτουργούν καλύτερα όταν τα λειτουργικά τους συστήματα υποστηρίζουν την πολυεπεξεργασία, την προτεραιότητα, την διαδικασία επικοινωνίας, την πολυνηματική επεξεργασία, την διαχείριση μνήμης, την απομόνωση της εφαρμογής και τις εκτεταμένες υπηρεσίες.

3.4.4 ΛΟΓΙΣΜΙΚΟ

Το λογισμικό στον server είναι το λειτουργικό σύστημα του server, το λειτουργικό σύστημα του δικτύου και το τμήμα του server της client-server εφαρμογής. Συνήθως, η λογική της server εφαρμογής προσφέρει υπηρεσίες έρευνας, υπολογισμού και προτεραιότητας, ώστε ο server να μπορεί να ανταποκριθεί στις απαιτήσεις του client για πληροφορίες.

Ποιος είναι ο ρόλος του server;
Η client-server διαδικασία μπορεί να απλοποιηθεί στα παρακάτω βήματα:

1. Ο χρήστης στέλνει μία αίτηση ή ένα ερώτημα, μέσω του client, στον server.
2. Ο server ακούει την αίτηση του client.
3. Από την στιγμή που ο server ακούει την αίτηση, ελέγχει την δυνατότητα πρόσβασης του χρήστη.

4. Ο server επεξεργάζεται το ερώτημα.
5. Ο server επιστρέφει τα αποτελέσματα στον client.
6. Ο client δέχεται τα αποτελέσματα και τα παρουσιάζει στον χρήστη.

Από τα έξι βήματα, ο server παίζει τέσσερις σημαντικούς ρόλους. Όπως είδαμε, ο server είναι η καρδιά της client-server εφαρμογής. Ο server υπάρχει για να απαντήσει στις ανάγκες του client, και ο client εξαρτάται από την αξιοπιστία και την έγκυρη απάντηση του server.
Ο server πρέπει να εκτελέσει τις ακόλουθες λειτουργίες:

1 Να ακούσει την αίτηση του client.
2 Να ελέγξει την δυναμικότητα πρόσβασης του χρήστη.
3 Να επεξεργαστεί την αίτηση.
4 Να επιστρέψει αποτελέσματα.

Ο server δεν εγκαινιάζει καμία ενέργεια. Αντίθετα, 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. Από την στιγμή που ο server δέχεται από τον client την απαίτηση, 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 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 πως διαχωρίζουμε την εφαρμογή, η βασική ευθύνη του server παραμένει η ίδια: να εξυπηρετεί τους clients που κάνουν αιτήσεις.

3.5 Η ΕΠΙΤΕΥΞΗ ΤΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ CLIENT-SERVER

3.5.1 ΑΚΟΥΓΟΝΤΑΣ ΤΗΝ ΑΙΤΗΣΗ ΤΟΥ CLIENT

Ο server δεν ξεκινάει καμία αλληλεπίδραση με τον client, απλά περιμένει τον client να κάνει την αίτηση του. Όταν ο client κάνει την αίτηση, ο server ανταποκρίνεται το συντομότερο δυνατό. Η κάρτα προσαρμογής στο δίκτυο συνδέει φυσικά τον server, με το δίκτυο και καθορίζει εάν οι εισερχόμενες απαιτήσεις είναι κατανοητές για τον κόμβο του προσαρμογέα. Εάν ναι, το πρωτόκολλο τις αποδέχεται και τις αποκωδικοποιεί ώστε μετά ο server να μπορεί να τις επεξεργαστεί.

3.5.2 ΕΛΕΓΧΟΝΤΑΣ ΤΗΝ ΔΥΝΑΤΟΤΗΤΑ ΠΡΟΣΒΑΣΗΣ ΤΟΥ ΧΡΗΣΤΗ

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

3.5.3 ΕΠΕΞΕΡΓΑΖΟΝΤΑΣ ΤΗΝ ΑΙΤΗΣΗ

Ο server πρέπει να είναι ικανός να ανταποκριθεί στην αίτηση του client αμέσως. 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 επιβεβαιώνει ότι ο χρήστης είναι εξουσιοδοτημένος να κάνει αιτήσεις στον server, ο server μπορεί να αποκαλύψει την απαίτηση και να την επεξεργαστεί.

Η αίτηση μπορεί να έχει μια από τις ακόλουθες τέσσερις μορφές:

1 Μια απόμακρη αίτηση είναι μια απλή αίτηση για πληροφορίες από έναν απλό client.
2 Μια απόμακρη συναλλαγή περιλαμβάνει πολλαπλές αιτήσεις για πληροφορίες από έναν απλό client.
3 Μια κατανεμημένη συναλλαγή περιλαμβάνει πολλαπλές αιτήσεις για πληροφορίες από έναν απλό client, οι οποίες πληροφορίες ανήκουν σε πολλούς server.
4 Μια κατανεμημένη αίτηση είναι μια συναλλαγή που σχηματίζεται από πολλαπλές αιτήσεις για πληροφορίες από πολλαπλούς clients, οι οποίες πληροφορίες ανήκουν σε πολλαπλούς servers.

Αυτές οι αιτήσεις πρέπει να περάσουν από το λεγόμενο ACID τεστ: Ατομικότητα (Atomicity), Συνέπεια (Consistency), Απομόνωση (Isolation) και Αντοχή(Durability).

Η ατομικότητα σημαίνει ότι ολόκληρη η συναλλαγή πρέπει να πετύχει ή να αποτύχει, δεν μπορεί να ολοκληρωθεί ως προς ένα κομμάτι της. Η συνέπεια σημαίνει ότι το σύστημα πάει από ένα σταθερό σημείο σε ένα άλλο σταθερό σημείο. Η απομόνωση σημαίνει ότι, από την στιγμή που μία συναλλαγή ολοκληρώνεται με επιτυχία, τα αποτελέσματά της δεν είναι ορατά σε άλλες συναλλαγές. Η αντοχή σημαίνει ότι από την στιγμή που η συναλλαγή ολοκληρώνεται με επιτυχία, δεσμεύεται μόνιμα από το σύστημα και επακόλουθες αποτυχίες δεν θα το επηρεάσουν. Εάν η συναλλαγή αποτύχει, το σύστημα οπισθοχωρεί στο σημείο που ήταν πριν προσπαθήσει να επεξεργαστεί την συναλλαγή.

Η γλώσσα καθορισμού των πληροφοριών καθορίζει τις δομές των πληροφοριών, η γλώσσα διαχείρισης των πληροφοριών μετακινεί και ενημερώνει τις πληροφορίες και η γλώσσα έλεγχου των πληροφοριών καθορίζει τους περιορισμούς πρόσβασης και ασφάλειας. Η SQL επεξεργάζεται πληροφορίες σε ομάδες. Έτσι, ο server μπορεί να στείλει πολλαπλές εγγραφές για να ικανοποιήσει την αίτηση του client. Η SQL

μπορεί επίσης να φιλτράρει, να μετασχηματίσει, ή να συνδυάσει πληροφορίες πριν τις στείλει στον client.

3.5.4 ΕΠΙΣΤΡΕΦΟΝΤΑΣ ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

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 έναν 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.

3.5.5 ΤΙ ΕΠΙΔΡΑ ΣΤΗΝ ΣΥΝΔΕΣΗ ΜΕΤΑΞΥ CLIENT ΚΑΙ SERVER;

Εκείνο που παίζει σημαντικό ρόλο στο client-server computing είναι η σύνδεση του client με τον server, δηλαδή ουσιαστικά το δίκτυο στο οποίο εντάσσονται. Οι χρήστες θέλουν να αισθάνονται ότι οι υπηρεσίες που χρειάζονται οι ίδιοι, είναι διαθέσιμες και προσπελάσιμες στο δίκτυο, χωρίς να πρέπει να λαμβάνουν υπόψη μόνο την τεχνολογία. Όταν χρειάζεται να χρησιμοποιήσουν client-server εφαρμογές, είναι απαραίτητο να προσδιορίζεται το θέμα της σύνδεσης. Αρχικά, οι περισσότεροι χρήστες ανακάλυψαν την ανάγκη της πρόσβασης σε έναν εκτυπωτή, ο οποίος δεν ήταν συνδεδεμένος φυσικά με τον client σταθμό εργασίας. Μπορεί αρχεία δεδομένων σε μη-δικτυωμένους υπολογιστές να χρησιμοποιούνται από κοινού, π.χ. με την μεταφορά μέσω δισκετών κτλ, αλλά στην εκτύπωση τα πράγματα είναι διαφορετικά. Τα πρώτα LANs (Local Area Network) που εγκαταστάθηκαν ήταν βασικά υπηρεσίες δικτύων για υποστήριξη των αναγκών εκτύπωσης. Τώρα ένας εκτυπωτής, οπουδήποτε και αν βρίσκεται μπορεί να χρησιμοποιείται για από κοινού χρήση.

Ο φυσικός τρόπος για να επιτευχθεί αυτή η σύνδεση client και server είναι η LAN καλωδίωση. Κάθε σταθμός εργασίας συνδέεται με ένα καλώδιο που οδηγεί την μετάδοση είτε αμέσως στον επόμενο σταθμό εργασίας του LAN είτε σε ένα κομβικό σημείο που οδηγεί την μετάδοση στην κατάλληλη διαδρομή. Υπάρχουν δύο βασικοί

LAN σχεδιασμοί, που χρησιμοποιούν Ethernet και Token Ring.

Υπάρχει μια βασική λειτουργική διαφορά στον τρόπο που οι σχηματισμοί Ethernet και Token Ring τοποθετούν τα δεδομένα στο καλώδιο. Με το πρωτόκολλο του Ethernet, ο επεξεργαστής επιχειρεί να αποθηκεύσει δεδομένα στο καλώδιο , οπότε απαιτεί εξυπηρέτηση. Το πρωτόκολλο αυτό περιλαμβάνει και την κατάλληλη λογική για να επιλύσει τις συγκρούσεις, όποτε αυτές εμφανίζονται. Από την άλλη πλευρά, με το πρωτόκολλο του Token Ring, ο επεξεργαστής επιχειρεί μόνο να τοποθετήσει δεδομένα πάνω στο καλώδιο, όταν υπάρχει χώρος για να δεχθεί την μετάδοση.

3.5.6 ΠΡΩΤΟΚΟΛΛΟ ΕΠΙΚΟΙΝΩΝΙΩΝ

Ταυτόχρονα ο client και ο server πρέπει να ακολουθούν ένα συγκεκριμένο σύνολο κανόνων όταν μετασχηματίζονται και εκπέμπονται απαιτήσεις μέσα σε ένα δίκτυο. Οι κανόνες που ακολουθούνται ονομάζονται πρωτόκολλα.
Υπάρχουν τρεις βασικές κατηγορίες πρωτοκόλλων: τα ασύγχρονα (asynchronous), τα δυαδικά συγχρονισμένα (binary-synchronous) και τα ψηφιακά προσαρμοσμένα (bit-oriented). Τα δύο πρώτα πρωτόκολλα βασίζονται σε χαρακτήρες, που σημαίνει ότι το πρωτόκολλο χρησιμοποιεί ένα συγκεκριμένο σύνολο κώδικα για μετάδοση, με κάποιους χαρακτήρες του συνόλου του κώδικα να δεσμεύονται για λειτουργίες ελέγχου. Το πρωτόκολλο που βασίζεται σε ψηφία αντιστοιχεί στο πρωτόκολλο που είναι ανεξάρτητο από συγκεκριμένο σύνολο κώδικα και δεν δεσμεύονται χαρακτήρες του κώδικα για λειτουργίες ελέγχου.

Όλα τα πρωτόκολλα επικοινωνιών εκτελούν τις ακόλουθες λειτουργίες:

1 Συγχρονισμό : εγκαθιστούν και καθοδηγούν μια σύνδεση μεταξύ client και server ώστε ο client και ο server δεν αποσυνδέονται όσο εκπέμπονται οδηγίες ή πληροφορίες.
2 Οριοθέτηση : σημειώνουν την αρχή και το τέλος κάθε εκπομπής, ώστε ο παραλήπτης να γνωρίζει που ξεκινάει και που τελειώνει η εκπομπή.
3 Έλεγχος : εκτελούν κάποιες λειτουργίες ελέγχου ανάλογα με τον τύπο της επικοινωνίας.
4 Εντοπισμό σφαλμάτων : εντοπίζουν σφάλματα και εκτελούν επανορθώσεις.

Για παράδειγμα το δίκτυο εντοπίζει μια διακοπή στην επικοινωνία μεταξύ του client και του server. Το σύστημα στέλνει μήνυμα στους χρήστες ρωτώντας τους αν θέλουν να εγκαταστήσουν ξανά τη επικοινωνία ή το σύστημα απλά συνεχίζει και εγκαθίσταται ξανά η επικοινωνία αυτόματα.

Κάποια πρωτόκολλα επικοινωνίας εκτελούν επίσης τις ακόλουθες λειτουργίες:

1 *Αποστολή* : διαχειρίζονται διευθύνσεις του δικτύου προετοιμάζοντας την αποστολή της απαίτησης του client στον κατάλληλο server.
2 *Αναμετάδοση* : ξαναστέλνουν εκπομπές όταν εκτελούνται σφάλματα, που μπορεί να συμβούν εάν οι πληροφορίες αλλοιώνονται κατά την μετάδοσή τους.
3 *Καθορισμό ρυθμού* : ελέγχουν τον ρυθμό με τον οποίο μεταδίδονται οι πληροφορίες για να σιγουρευτούν ότι ο client δεν στέλνει πληροφορίες γρηγορότερα από ότι μπορεί να λάβει ο server ή το αντίστροφο.
4 *Αναζήτηση πληροφοριών* : ερευνούν για την κατάσταση των άλλων clients ή servers-για παράδειγμα, να δουν εάν ένας server είναι διατεθειμένος ή όχι να λάβει μια απαίτηση.

3.6 ΙΣΧΥΡΟΙ SERVERS ΚΑΙ ΙΣΧΥΡΟΙ CLIENTS (FAT SERVERS ΚΑΙ FAT CLIENTS)

Εκτός από τον διαχωρισμό ανάλογα με τις υπηρεσίες που προσφέρουν, οι client-server εφαρμογές μπορούν να διακριθούν ανάλογα με το πώς η εφαρμογή κατανέμεται μεταξύ του client και του server. Το μοντέλο του ισχυρού server προσδίδει περισσότερες λειτουργίες στον server. Το μοντέλο του ισχυρού client προσδίδει περισσότερες λειτουργίες στον client. Οι Web servers είναι παραδείγματα ισχυρών servers, ενώ οι servers των βάσεων δεδομένων και αρχείων είναι παραδείγματα ισχυρών clients.

Οι ισχυροί clients είναι πιο παραδοσιακοί τύποι των clients-servers. Το κύριο σώμα της εφαρμογής τρέχει στην πλευρά της εξίσωσης, που ανήκει στον client. Ταυτόχρονα στον server αρχείων και τον server βάσεων δεδομένων , οι clients γνωρίζουν το πώς είναι οργανωμένες και αποθηκευμένες οι πληροφορίες στην

πλευρά του server. Προσφέρουν ευλυγισία και ευκαιρίες για δημιουργία εργαλείων που επιτρέπουν στους τελικούς χρήστες να δημιουργήσουν τις δικές τους εφαρμογές.

Οι εφαρμογές των ισχυρών servers είναι πιο εύκολο να διαχειρίζονται και να αναπτύσσονται στο δίκτυο διότι το μεγαλύτερο μέρος του κώδικα τρέχει στους servers. Οι ισχυροί servers προσπαθούν να ελαχιστοποιήσουν τις ανταλλαγές στο δίκτυο δημιουργώντας πιο ουσιώδη επίπεδα υπηρεσιών. Οι servers «συναλλαγών», για παράδειγμα, συμπυκνώνουν τη βάση δεδομένων. Αντί να εξάγουν ανεπεξέργαστες πληροφορίες, εξάγουν τις διαδικασίες που χειρίζονται αυτές τις πληροφορίες. Ο client στο μοντέλο του ισχυρού client προσφέρει το GUI και αλληλεπιδρά με τον server μέσω των RPCs (Remote Procedure Calls).
Κάθε client-server μοντέλο έχει την χρησιμότητά του. Σε πολλές περιπτώσεις, τα μοντέλα αλληλοσυμπληρώνονται και δεν είναι ασυνήθιστο να συνυπάρχουν σε μια εφαρμογή. Για παράδειγμα, μια εφαρμογή θα μπορούσε να απαιτεί έναν server, ο οποίος να συνδυάζει τους servers αρχείων, βάσεων δεδομένων και συναλλαγών.

3.7 ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ TWO-TIER (2-ΣΤΡΩΜΑΤΩΝ) ΚΑΙ THREE-TIER(3-ΣΤΡΩΜΑΤΩΝ) CLIENT-SERVER

Πολλές φορές προτιμούνται να χρησιμοποιούνται όροι, όπως 2-tier, 3-tier client-server αρχιτεκτονικές αντί των όρων ισχυροί clients και ισχυροί servers. Αλλά ουσιαστικά αυτοί οι όροι βασίζονται στην ίδια βασική ιδέα. Έχουν να κάνουν με το πώς διαιρείται η client-server εφαρμογή σε λειτουργικές ενότητες, οι οποίες μετά μπορούν να ανατεθούν είτε στον client, είτε σε έναν ή περισσότερους servers.

3.7.1 TWO-TIER CLIENT-SERVER COMPUTING

Οι client-server εφαρμογές πρώτης γενιάς εκτελούνταν γενικά με δυο λογικά στρώματα. Αυτό το μοντέλο έχει συχνά δυο στρώματα υλικού. Αυτή δεν είναι η περίπτωση όπου ο client και ο server τρέχουν ταυτόχρονα στον ίδιο υπολογιστή. Ο two-tier client-server διαιρεί την εφαρμογή σε δυο συγκεκριμένα τμήματα (τα tiers), όπου ένα τμήμα τρέχει στον client υπολογιστή και ένα ξεχωριστό τμήμα τρέχει στον server . Αξίζει να σημειωθεί ότι ο κώδικας του client και του server δεν

ενημερώνει, ούτε καν γνωρίζει εάν αυτοί τρέχουν στον ίδιο υπολογιστή ή όχι. Επιπλέον, η εφαρμογή διαιρείται κατά μήκος του client και του server.

Η ποσότητα της λογικής εφαρμογής που λειτουργεί στον client ή στον server καθορίζει εάν αυτό είναι αδύνατο ή ισχυρό. Το αδύνατο υποδηλώνει ότι παρουσιάζεται μικρή ανάπτυξη της εφαρμογής και το ισχυρό ότι παρουσιάζεται ένα μεγάλο τμήμα της λογικής της εφαρμογής. Υπάρχουν ποικίλες διαβαθμίσεις αδυνάτου και ισχυρού. Οι αδύνατοι client είναι ελκυστικοί όταν ο client υπολογιστής έχει περιορισμένη επίδοση. Δεν υπάρχουν MIPS να χειριστούν ένα GUI, επικοινωνίες και ένα σημαντικό τμήμα της ανάπτυξης της εφαρμογής.

Οι two-tier clients-server εμφανίζονται να είναι πιο δύσκολο να αναπτυχθούν και να διατηρηθούν από ό,τι κανονικά προσδοκάται. Οι two-tier εφαρμογές δεν κλιμακώνουν καλά. Επίσης , τα εργαλεία των client-server πήραν χρόνο για να αναπτυχθούν. Η Visual Basic, η Delphi και το PowerBuilder είναι συγκεκριμένα παραδείγματα αυτού του τύπου των εργαλείων. Τα περισσότερα από αυτά τα εργαλεία είναι σε πορεία ανάπτυξης για υποστήριξη three-tier.

3.7.2 THREE-TIER CLIENT-SERVER COMPUTING

Ο πιο πρόσφατος τύπος client-server computing που αναπτύσσεται είναι ο three-tier. Μερικοί άνθρωποι επίσης χρησιμοποιούν πολλαπλούς tier για να περιλάβουν οποιαδήποτε προσέγγιση που χρησιμοποιεί περισσότερους από δυο λογικούς tiers.

Υπάρχουν πολλές προσεγγίσεις για ανάπτυξη multi-tier client-server σήμερα. Το πιο προφανές συνεχίζει να χρησιμοποιεί εργαλεία two-tier για το GUI. Μια δεύτερη προσέγγιση που μπορεί να κερδίσει έδαφος, ειδικά για μεγάλες εργασίες, είναι ένα ενσωματωμένο εργαλείο ανάπτυξης client-server, όπως το TI˙s IEF.

Μια πρωταρχική διαφορά μεταξύ two-tier και three-tier εφαρμογών είναι η επιπλέον επίστρωση λογισμικού στο server. Όπου οι two-tier εφαρμογές τείνουν να

τοποθετούν την λογική στον client και να περνούν εγγραφές στη βάση δεδομένων (ισχυρό client μοντέλο) ή να περνούν δεδομένα στη βάση δεδομένων, όπου αποθηκευμένες διαδικασίες εκτελούν την λογική της εφαρμογής (αδύνατο client μοντέλο), οι three-tier εφαρμογές τείνουν να περνούν μήνυμα μεταξύ των client και των server τμημάτων του κώδικα της εφαρμογής. Το τμήμα του server εφαρμόζει τη λογική της εφαρμογής, κατόπιν την στέλνει στη βάση δεδομένων. Η λογική της εφαρμογής συνήθως καλείται «Business Rules» στο χώρο των client -server.

Οι three-tier clients-servers προσθέτουν κάποια πολυπλοκότητα, διότι πρόκειται για ένα επιπρόσθετο κομμάτι του κώδικα που αναπτύσσεται. Τα εργαλεία και οι γλώσσες προγραμματισμού που χρησιμοποιούνται στον κώδικα του server εξαρτώνται από την πλατφόρμα του server. Η Visual Basic δεν είναι κατάλληλη για εφαρμογή tier σε έναν Sun Sparc Server. Ένας UNIX server ίσως υποστηρίζει C ή C++, ενώ ένας server που βασίζεται σε Intel και τρέχει τα Windows NT μπορεί επίσης να χρησιμοποιήσει Delphi αποτελεσματικά για εφαρμογή tier. Με λίγα λόγια, δεν υπάρχει μια απλή απάντηση για το ποια γλώσσα χρησιμοποιείται για το τμήμα του server μιας εφαρμογής client -server, όσο δεν μπορούμε να αναγκάσουμε τον server να χρησιμοποιεί την ίδια γλώσσα προγραμματισμού όπως ο client όταν δεν είναι κατάλληλη.

Το τμήμα του server της three-tier αρχιτεκτονικής προσθέτει κάτι σε όλη την πολυπλοκότητα της εφαρμογής. Ωστόσο υπάρχουν ορισμένα πλεονεκτήματα σε μια three-tier client-server προσέγγιση. Αυτά περιλαμβάνουν:

1 *Κλιμάκωση*
2 *Γενικότερα πιο χαμηλά προβλήματα στα δίκτυα διανομής*
3 *Ευλυγισία*

Η *κλιμάκωση* βελτιώθηκε, διότι ο κώδικας του server και η βάση δεδομένων είναι χωρισμένα, μπορούν να ξεκινήσουν από ένα απλό «υπολογιστή-οικοδεσπότη» και αργότερα να χωριστούν. Πολλαπλές εφαρμογές server μπορούν να επικοινωνήσουν με μια κεντρική βάση δεδομένων ή μια εφαρμογή server μπορεί ακόμα να εξυπηρετήσει τους clients ενώ προσπελάζονται οι πολλαπλές βάσεις δεδομένων όσο

το σύστημα αυξομειώνεται.

Τα χαμηλότερα προβλήματα στα δίκτυα απορρέουν από το πέρασμα μικρών μηνυμάτων στην εφαρμογή παρά από ολόκληρες εγγραφές πληροφοριών.

Η *ευλυγισία* κερδίζεται, διότι ο client, ο server και τα συστήματα βάσεων δεδομένων μπορούν το καθένα να αντικατασταθούν χωρίς να επηρεάζουν τα αλλά κομμάτια, δεδομένου ότι η διασύνδεση επίσης δεν αλλάζει. Για παράδειγμα, μετατρέποντας τη βάση δεδομένων από Sybase σε Oracle επηρεάζεται μόνο το τμήμα του server της εφαρμογής , όχι του client. Το να επαναδιατυπώνεις έναν client από Visual Basic σε Delphi δεν έχει επίδραση στο υπόλοιπο τμήμα της εφαρμογής, δεδομένου ότι υπάρχει αλληλεπίδραση ανάμεσα στον κώδικα του client και τον κώδικα του server.

Πολλές από τις σημερινές εμπορικές εφαρμογές, που βασίζονται σε βάσεις δεδομένων, συμπεριλαμβανόμενου και του SAP's R/3, χρησιμοποιούν το three-tiered client-server μοντέλο για να κερδίσουν τη δυνατότητα αυξομείωσης και ευλυγισίας. Επίσης οι Web εφαρμογές είναι γενικά three-tier client-server εφαρμογές.

3.8 Η ΣΗΜΑΣΙΑ ΤΟΥ CLIENT-SERVER COMPUTING

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.

Οι πληροφορίες δεν αποθηκεύονται ή ελέγχονται από κεντρικούς μεγάλους υπολογιστές (mainframes). Αντίθετα, είναι εύκολα προσπελάσιμες στους servers του δικτύου.

Από την στιγμή που το client-server computing 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.

3.9 Η ΑΝΑΠΤΥΞΗ CLIENT-SERVER ΕΦΑΡΜΟΓΩΝ

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

1 Προσδιορίζεται το πρωτόκολλο επικοινωνίας του client και server.
2 Αναπτύσσονται τα clients και servers προγράμματα.
3 Μεταγλωττίζονται τα προγράμματα.
4 Διασυνδέονται οι βιβλιοθήκες.
5 Εξετάζονται οι εφαρμογές τοποθετώντας τον server σε μια απομακρυσμένη μηχανή και τρέχοντας τον client τοπικά.

Εξαιτίας της πολυπλοκότητας της client-server αρχιτεκτονικής, η ανάπτυξη client-server εφαρμογών απαιτεί πιο λεπτομερή σχεδιασμό-ειδικότερα, πως να διαχωριστεί η εφαρμογή ανάμεσα στον client και τον server και πως να κατανεμηθούν οι πληροφορίες μεταξύ client και server. Οι εφαρμογές που χρησιμοποιούν RPCs είναι κατανεμημένες διότι τα RPCs είναι ουσιαστικά clients και απομακρυσμένοι server επεξεργαστές. Οι ίδιες οι εφαρμογές συνήθως διαιρούνται σε δυο τμήματα -το τμήμα του client και το τμήμα του server.
Οι clients κάνουν κλήσεις και οι servers εξυπηρετούν τις κλήσεις.

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, βασικά 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.

3.10 Η ΚΑΤΑΝΟΜΗ ΤΩΝ ΠΛΗΡΟΦΟΡΙΩΝ

Αφού αποφασιστεί ο διαχωρισμός της εφαρμογής μεταξύ του client και του server πρέπει να αποφασιστεί το πώς θα κατανέμονται οι πληροφορίες. Ένας από τους κύριους λόγους που χρειάζεται να κατανέμονται οι πληροφορίες είναι η ελαχιστοποίηση και συνεπώς ο περιορισμός των προβλημάτων στο δίκτυο.
Υπάρχουν τέσσερις βασικές επιλογές κατανομής των πληροφοριών, όπως φαίνονται παρακάτω:

1 Η ύπαρξη πολλαπλών πανομοιότυπων αντιγράφων.
2 Η διαίρεση των πληροφοριών σε διάφορες περιοχές.
3 Η αντίληψη πληροφοριών από άλλες περιοχές.
4 Η αντιγραφή των πληροφοριών σε περιοχές προσπέλασης, σε υψηλή ταχύτητα.

Η αρχή που παίζει μεγάλο ρόλο στον καθορισμό του πως κατανέμονται οι πληροφορίες είναι η κομβική χωροθέτηση (nodal residency). Αυτό σημαίνει, ότι οι πληροφορίες είναι αποθηκευμένες πλησιέστερα στους χρήστες που τις χρειάζονται. Έτσι, εάν ένα αρχείο αναφοράς που δεν πρόκειται να αλλάζει συχνά χρησιμοποιείται από πολλαπλούς χρήστες, είναι λογικό να υπάρχουν πολλαπλά αντίγραφα αυτού του αρχείου στους clients. Εάν κάποιες πληροφορίες χρησιμοποιούνται από όλους τους clients και άλλες πληροφορίες είναι ειδικές για συγκεκριμένους clients, είναι πιο πρακτικό να αποθηκεύονται οι πρώτες πληροφορίες στον server και οι δεύτερες πληροφορίες στον client. Εάν κάποιες πληροφορίες που αλλάζουν συχνά, μπορούν να υπολογιστούν γρήγορα από υπάρχουσες πληροφορίες στον server, δεν

χρειάζεται να σπαταλείται χρόνος ή χώρος για να αποθηκευτούν. Ή, εάν οι clients χρησιμοποιούν συνήθως το ίδιο τμήμα των αρχείων, είναι βολικό να αντιγράφονται τα τμήματα που χρειάζονται οι χρήστες κάθε φορά που απαιτούν τις πληροφορίες. Από τη στιγμή που αποφασιστεί το πώς θα διαιρεθούν οι εργασίες και οι πληροφορίες μεταξύ του client και του server, τότε αρχίζει η δημιουργία των προγραμμάτων.

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

1 *Πρόσβαση σε πολλαπλές πληροφορίες*, δηλαδή, οι πληροφορίες να είναι διαθέσιμες σε όλους τους εξουσιοδοτημένους χρήστες.
2 *Ολοκληρωμένες υπηρεσίες*.
3 *Πρόσβαση σε πόρους μέσω διαφόρων πλατφόρμων*.
4 *Ανταλλαγή και εκμετάλλευση πληροφοριών*.
5 *Πρόσβαση σε οποιεσδήποτε πληροφορίες*, η ακόμα και σε φαινομενικά απροσπέλαστες πληροφορίες.
6 *Ευκολότερη διατήρηση και συντήρηση των πληροφοριών*.

4. ΣΥΝΗΘΙΣΜΕΝΑ CLIENT-SERVER ΕΡΓΑΛΕΙΑ

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

4.1 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.

4.2 VISUAL BASIC

Η Visual Basic είναι η καθαρά οπτική γλώσσα προγραμματισμού. Προσφέρει τα εργαλεία για να δημιουργήσει client εφαρμογές. Η έκδοση 4 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. Η έκδοση 5 προσφέρει έναν απλό compiler, μεταξύ των άλλων καινοτομιών της.

4.3 POWER BUILDER

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.

4.4 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.

4.5 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 .

4.6 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.

4.7 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. Επίσης 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.

Τα κύρια χαρακτηριστικά της Java είναι:

1 Τα προγράμματά της είναι μεταφέρσιμα μέσα από το δίκτυο. Δηλαδή ένα πρόγραμμα μεταφράζεται σε αυτό που ονομάζουμε "Java bytecode", μία ενδιάμεση μορφή που επιτρέπει στο πρόγραμμα να εκτελεστεί σε οποιοδήποτε Η/Υ στο δίκτυο (server ή client) που έχει εγκατεστημένη την Java Virtual Machine. Αυτή η Java Virtual Machine μεταγλωττίζει το πρόγραμμά μας από την ενδιάμεση μορφή (Java bytecode) σε πλήρως εκτελέσιμο για τον εκάστοτε Η/Υ κώδικα. Αυτό σημαίνει ότι οι ιδιαιτερότητες της κάθε αρχιτεκτονικής και πλατφόρμας αναγνωρίζονται και αντιμετωπίζονται τοπικά, καθώς εκτελείται το πρόγραμμα. Έτσι απαλείφεται το πρόβλημα των πολλαπλών εκδόσεων για κάθε διαφορετική υπολογιστική πλατφόρμα.

2 Ο κώδικας είναι πιο "συμπαγής" και "αξιόπιστος", με την έννοια ότι αντίθετα με την C++, τα αντικείμενα στην Java δεν μπορούν να κάνουν αναφορές σε δεδομένα εξωτερικά ως προς τα ίδια ή συγγενή αντικείμενα. Αυτό διασφαλίζει ότι μία εντολή δεν μπορεί να περιέχει μία διεύθυνση μνήμης που ανήκει σε μία άλλη εφαρμογή ή ακόμη και στο ίδιο το λειτουργικό σύστημα, κάτι που μπορεί να οδηγήσει σε πολλά προβλήματα. Η virtual machine της Java κάνει πολλαπλούς ελέγχους, για να διασφαλίσει την ακεραιότητα κάθε αντικειμένου.

3 Η Java είναι αντικειμενοστραφής.. Ο χρήστης μπορεί να αντιλαμβάνεται ένα αντικείμενο στον προγραμματισμό σαν το γραμματικό όρο "υποκείμενο" σε

μία πρόταση, το οποίο ο χρήστης συνδέει με άλλα υποκείμενα με τη χρήση μεθόδων που αντιστοιχούν στα γραμματικά "ρήματα". Μία μέθοδος μπορεί να θεωρηθεί ως μία ενέργεια ή χαρακτηριστικό του αντικειμένου. Η αντικειμενοστρέφεια ανάμεσα σε πολλά άλλα πράγματα, σημαίνει ότι τα αντικείμενα μπορούν να εκμεταλλευτούν το γεγονός ότι ανήκουν στην ίδια κλάση και να κληρονομήσουν κάποιο κοινό κώδικα. Επιπλέον του γεγονότος ότι εκτελείται το πρόγραμμα στον εξυπηρετούμενο Η/Υ και όχι στον εξυπηρετητή, υπάρχουν και άλλα χαρακτηριστικά που προσδίδουν ταχύτητα σε ένα Java πρόγραμμα, λαμβάνοντας πάντα υπόψη την επιβάρυνση που του προσδίδει η ιδιότητα της διαπλατφορμικότητας.

4 Είναι μία γλώσσα πολύ ευκολότερη στην εκμάθηση από την C++.

4.8 ΆΛΛΕΣ ΓΛΩΣΣΕΣ

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.

5. ΈΝΑ ΠΑΡΑΔΕΙΓΜΑ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ CLIENT-SERVER ΜΟΝΤΕΛΟΥ

Στα πλαίσια της εργασίας υλοποιήθηκε και ένα απλό παράδειγμα επικοινωνίας μεταξύ client και server. Στην ενότητα αυτή θα περιγραφεί λεπτομερώς ο τρόπος λειτουργίας του συγκεκριμένου προγράμματος, το οποίο γράφτηκε στη γλώσσα προγραμματισμού Perl και σε περιβάλλον Linux.

Μέσω του προγράμματος αυτού, επιτυγχάνεται επικοινωνία ανάμεσα στον client και τον server σε μια απλή μορφή, η οποία άνετα μπορεί να επεκταθεί σε πιο σύνθετες λειτουργίες, ανάλογα με τις απαιτήσεις των εφαρμογών που καλείται να υποστηρίξει. Αυτό που υλοποιείται είναι η απλή ανταλλαγή μηνυμάτων κειμένου ανάμεσα τους.

Συγκεκριμένα, ο μηχανισμός επικοινωνίας που ακολουθήθηκε είναι αυτός των sockets. Πιο αναλυτικά τώρα, ο εξυπηρετητής καταρχήν δημιουργεί ένα socket, μέσω του διαθέσιμου από τη Perl τύπου *IO::Socket::INET* και της συνάρτησης δημιουργίας *new()*, δίνοντας τον αριθμό του port στο οποίο θα «ακούει», θα περιμένει δηλαδή τη σύνδεση από τους πελάτες, τον τύπο του socket και κάποια άλλα χαρακτηριστικά στοιχεία. Στο δικό μας παράδειγμα ο αριθμός του port στατικός (συγκεκριμένα το 32227) ενώ εύκολα θα μπορούσε να δίνεται σαν είσοδος από τον χρήστη που εκκινεί τον εξυπηρετητή. Σε περίπτωση που κάποιο πρόβλημα προκύψει σε αυτή τη διαδικασία δημιουργίας του socket από τη μεριά του εξυπηρετητή, ένα μήνυμα λάθους εμφανίζεται στο χρήστη ενημερώνοντάς τον για

το σφάλμα και το πρόγραμμα τερματίζει. Αν όλα πάνε κατ' ευχήν, ο εξυπηρετητής εισέρχεται σε μια ατέρμονο κατάσταση αναμονής αίτησης σύνδεσης από κάποιον πελάτη. Όταν μια τέτοια αίτηση καταφθάσει, στέλνει στον συνδεδεμένο (μέσω του socket) πελάτη ένα απλό μήνυμα χαιρετισμού (καλωσορίσματος).

Ο πελάτης από τη δική του μεριά, δημιουργεί ένα socket με αντίστοιχα χαρακτηριστικά (port number, τύπο κτλ) και σε περίπτωση σφάλματος στη σύνδεση, όπως και ο εξυπηρετητής, τερματίζει τη λειτουργία του εμφανίζοντας ένα μήνυμα λάθους. Αλλιώς η δημιουργία του socket από τον πελάτη δεδομένου ότι και ο εξυπηρετητής έχει από πριν κατασκευάσει το δικό του, καθιστά δυνατή τη μεταξύ τους επικοινωνία. Μετά από αυτό ο πελάτης λαμβάνει το μήνυμα που όπως είδαμε στέλνει ο εξυπηρετητής προκειμένου να «καλωσορίσει» την εισερχόμενη σύνδεση και το τυπώνει στην οθόνη του χρήστη, που εκτελεί την client-εφαρμογή.

Στη συνέχεια παρατίθενται σχολιασμένοι οι κώδικες υλοποίησης των παραπάνω εφαρμογών.

```
#!/usr/bin/perl -w

use sigtrap;
use Socket;

use IO::Socket::INET; # χρήση του ενσωματωμένου στην Perl τύπου
                                                    # socket

# δημιουργία socket – ορισμός παραμέτρων
$server = IO::Socket::INET->new(LocalPort => 32227, # αριθμός port
                        Type => SOCK_STREAM, # τύπος
                        Reuse => 1, # επαναχρησιμοποιήσιμο
                        Listen => 3 ) # μέγιστος αριθμός
                                        # εισερχόμενων συνδέσεων

      or die "couldn't serve\n"; # σε περίπτωση αποτυχίας
                        # έξοδος και σχετική ενημέρωση του χρήστη

while ($client = $server->accept()) { # αναμονή εισερχόμενου πελάτη

print $client "hello\n" # αποστολή μηνύματος στον πελάτη μέσω του
                                                    # socket
}

close($server); # τερματισμός σύνδεσης
```

Κώδικας 1. Η server-εφαρμογή

```
#!/usr/bin/perl -w

use sigtrap;
use Socket;

use IO::Socket::INET; # χρήση του ενσωματωμένου στην Perl τύπου
                                                    # socket

# εγκατάσταση socket προς τον server – ορισμός παραμέτρων
$socket = IO::Socket::INET->new(PeerAddr => localhost,# διεύθυνση
                            PeerPort => 32227,# αριθμός port
                            Proto => "tcp",# πρωτόκολλο
                            Type => SOCK_STREAM) # τύπος

      or die "couldn't connect to ...\n"; # σε περίπτωση αποτυχίας
                         # έξοδος και σχετική ενημέρωση του χρήστη

print $socket "hola!\n"; # αποστολή μέσω του socket μηνύματος

$answer = <$socket>; # αποθήκευση της απάντησης του server

print ("i received this : ", $answer); # εκτύπωση της απάντησης

close($socket); # τερματισμός της σύνδεσης
```

Κώδικας 2. Η client-εφαρμογή

5.1 ΔΥΝΑΤΕΣ ΕΠΕΚΤΑΣΕΙΣ - ΒΕΛΤΙΩΣΕΙΣ

Το παραπάνω πρόγραμμα αν και στοιχειώδες παρέχει τη βάση για ποικίλες και σύνθετες λειτουργίες. Αυτή η ανταλλαγή μηνυμάτων που επιτυγχάνει, μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές. Σε αυτήν την παράγραφο θα αναφέρουμε μερικές μόνο από τις επεκτάσεις που εύκολα μπορούν να γίνουν προκειμένου να επιτευχθούν πιο σύνθετες λειτουργίες.

Για παράδειγμα, ο εξυπηρετητής θα μπορούσε να αποθηκεύει τις πληροφορίες που του αποστέλλει ο πελάτης σε μια απομακρυσμένη βάση δεδομένων, κάτι που πολλές εφαρμογές απαιτούν.

Επίσης η επικοινωνία είναι δυνατό να πάρει τη μορφή ελέγχου της ταυτότητας του πελάτη, με την αποστολή εκ μέρους του κάποιου κωδικού (password) την ορθότητα του οποίου ελέγχει ο server.

Επιπρόσθετα, τα χαρακτηριστικά λειτουργίας του socket που στο παράδειγμα που εδώ παρουσιάσαμε είναι στατικά, θα μπορούσαν να δίνονται δυναμικά από τον χρήστη που το εκτελεί. Πιο συγκεκριμένα τα στοιχεία αυτά είναι ο αριθμός του port στο οποίο λειτουργεί ή και το πρωτόκολλο που χρησιμοποιείται, που αντί να αποτελούν μέρος του προγράμματος θα μπορούσαν να εισάγονται σαν ορίσματα από το χρήστη.