Πανεπιστήμιο Πατρών
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

Τελική Έκθεση Προόδου
για την εργασία του μαθήματος
Προηγμένες Τεχνικες Προγραμματισμού

Εργασία 18:
Ανάπτυξη TCP agent
(send-receive fieldbus data packet).

Θωμάς Ρεπαντής
Έτος: Δ΄
Α.Μ.: 4218
E-mail: darkzero@otenet.gr

Πάτρα, 10.1.2001

Εισαγωγή. Σκοπός της εργασίας είναι η ανάπτυξη ενός TCP agent. Αυτή περιλαμβάνει το σχεδιασμό και την υλοποίησή του. Απαραίτητη προϋπόθεση είναι προφανώς η ανάλυση, μελέτη και κατανόηση των εμπλεκόμενων διαδικασιών και κατόπιν των μεθόδων υλοποιήσης στη γλώσσα προγραμματισμού που απαιτήθηκε (Java). Ακολουθεί ο σχεδιασμός του TCP agent και του αντίστοιχου client και τέλος η υλοποίησή τους.

Περιγραφή των ενεργειών. Αρχικά κατανοήθηκε και μελετήθηκε το θέμα. Κατόπιν αναλύθηκαν οι απαιτούμενες διαδικασίες. Ακολούθησε η μελέτη των μεθόδων υλοποίησης και των προσφερόμενων εργαλείων από τη γλώσσα. Ακολούθησε ο σχεδιασμός των εφαρμογών και έπειτα η υλοποίησή της στη Java. Πιο συγκεκριμένα:

Κατανόηση θέματος. Λέγοντας agent (πράκτορα) εννοούμε ένα πρόγραμμα που -δρώντας συνήθως στο υπόβαθρο σε ένα δίκτυο ή στο Διαδίκτυο- συλλέγει ή επεξεργάζεται πληροφορίες. Συνήθως η εργασία που εκτελεί ένας πράκτορας είναι μικρή και σαφώς καθορισμένη, εκτελείται δε χωρίς την ανάγκη χειριστή και μάλιστα συχνά με βάση ένα συγκεκριμένο χρονοδιάγραμμα. Συχνά και ανάλογα με τη λειτουργία, μπορεί να χρησιμοποιηθούν για έναν agent και οι παρακάτω ονομασίες: intelligent agent, bot και robot.

Σε ορισμένες περιπτώσεις οι πράκτορες συσχετίζονται με την τεχνολογία push ή server-push (παράδοση πληροφοριών στο χρήστη). Στη συγκεκριμένη περίπτωση μας ενδιαφέρει ο τελικός agent να στέλνει και να λαμβάνει πακέτα σε ένα βιομηχανικό τοπικό δίαυλο-δίκτυο που περιλαμβάνει αυτόματες ψηφιακές συσκευές μετρήσεων και ελέγχου (fieldbus).

Απαιτούμενες διαδικασίες. Ουσιαστικά ένα τμήμα του agent δρα ως TCP server και ένα άλλο ως TCP client. Επομένως μας ενδιαφέρουν οι μέθοδοι υλοποίησης για τέτοιες κατηγορίες προγραμμάτων. Ο agent σε μια γενικότερη υλοποίηση θα δέχεται πακέτα από ένα βιομηχανικό δίκτυο (client) και θα τα προωθεί σε έναν υπολογιστή (server). Ωστόσο στο δικό μας παράδειγμα που χρησιμεύει περισσότερο ως πρώτο βήμα και κύριο στόχο έχει την επίδειξη των διαδικάσιών ο agent δέχεται αιτήσεις από έναν ειδικό για αυτόν client, συγκεντρώνει πληροφορίες από τα time και chargen services ενός server και τις προωθεί στον client.

Μέθοδοι υλοποίησης. Μελετήθηκε το βιβλίο Internetworking with TCP/IP vol. III - Client-Server Programming and Applications (D. Comer- D. Stevens) (ιδίως τα κεφάλαια 1-12 και 16). Έτσι φάνηκαν οι διάφορες επιλογές στους αλγόριθμους υλοποίησης:

Το μέρος του agent που αναλαμβάνει το ρόλο του server θεωρούμε ότι προϋποτίθεται να είναι αξιόπιστο (connection-oriented) (TCP) και concurrent, αν και για χαμηλό φόρτο και απαιτήσεις επεξεργασίας ένας iterative server ίσως να απέδιδε και καλύτερα. Το ερώτημα που τίθεται είναι αν η πολυδιεργασία θα επιτευχθεί μέσω ταυτόχρονα εκτελούμενων διεργασιών (concurrent process implementation) ή μέσω μιας διεργασίας (single-process implementation), πολλαπλών συνδέσεων και ασύγχρονης εισόδου και εξόδου. Αν και η πρώτη υλοποίηση είναι πιο συνηθισμένη, η δεύτερη μπορεί να αποδώσει καλά, εφόσον η συγκεκριμένη εφαρμογή την επιβάλλει, λόγω ανάγκης για κοινή πρόσβαση σε δεδομένα, ή όταν ο απαιτούμενος χρόνος επεξεργασίας των αιτήσεων είναι μικρός (μόνο τότε όμως). Σημαντικό μειονέκτημα της υλοποίησης μέσω μοναδικής διεργασίας είναι η πιθανότητα να οδηγηθεί η διεργασία αυτή και μαζί της όλος ο εξυπηρετητής σε αδιέξοδο (deadlock), σε περίπτωση που ένας συνδεδεμένος πελάτης δε στέλνει αιτήσεις ή δε διαβάζει τις απαντήσεις. Σχετικά με τον εξυπηρετητή πρέπει τέλος να σημειώσουμε ότι όπως ορίστηκε η εφαρμογή μας δεν επιβάλλει την ανάπτυξη multiprotocol ή multiservice server. Τελικά η εργασία που καλείται ο agent να εκτελέσει μας ωθεί να επιλέξουμε το μοντέλο των ταυτόχρονα εκτελούμενων διεργασιών, μιας και δεν έχουμε κοινή πρόσβαση στα δεδομένα μεταξύ αυτών. Επιπλέον αποφεύγουμε τον κίνδυνο του αδιεξόδου, ενώ ακολουθούμε και το συνηθισμένο τρόπο υλοποίησης τέτοιων εφαρμογών σε Java με χρήση νημάτων (threads)

Το μέρος του agent που αναλαμβάνει το ρόλο του client θεωρούμε ότι προϋποτίθεται να είναι επίσης αξιόπιστο (connection-oriented) (TCP). Η συνήθης αντιμετώπιση θέλει τους πελάτες να μη χρησιμοποιούν ταυτοχρονισμό, μιας και αυτός εισάγει επιπλέον επεξεργαστικό κόστος. Ωστόσο αυτός δεν αποκλείεται, σε περιπτώσεις κυρίως που η εφαρμογή το επιβάλλει. (Αν λόγου χάρη υπάρχει η ανάγκη επικοινωνίας με πολλούς εξυπηρετητές ταυτόχρονα ή γενικότερα ασύγχρονης πολυδιεργασίας.) Κάτι τέτοιο δεν απαιτούνταν στην εφαρμογή μας και έτσι ο agent επικοινωνεί διαδοχικά με τον time και με τον chargen server, δηλαδή δε χρησιμοποιεί ταυτοχρονισμό στο μέρος του που δρα ως πελάτης.

Προσφερόμενα εργαλεία. Αρχικά μελετήθηκαν οι σχετικές με το θέμα υλοποιήσεις σε C που περιέχει το βιβλίο που αναφέραμε (Internetworking with TCP/IP vol. III - Client-Server Programming and Applications). Αυτές

περιελάμβαναν concurrent, connection oriented TCP ECHO server, single-process concurrent, connection oriented TCP ECHO server και TCP DAYTIME client. Κατόπιν μελετήθηκε το τμήμα του Java Tutorial της Sun που αναφέρεται σε Custom Networking και ειδικά το μέρος εκείνο που μιλάει για sockets, μιας και ενδιαφερόμαστε για συνδέσεις TCP (connection oriented). Τα παραδείγματα βοήθησαν στην εξοικείωση με τις κλάσεις του πακέτου java.net και ιδιαίτερα τις απαραίτητες Socket και ServerSocket. Επιπλέον πληροφορίες παρείχαν το Java API (Application Programming Interface) και η ιεραρχία των κλάσεων από τη Sun. Τα θέματα διαχείρισης εξαιρέσεων και ταυτόχρονου προγραμματισμού στη Java θίγονται στα παραπάνω παραδείγματα, αλλά έχουν αναλυθεί και στις παραδόσεις και στα συγγράματα του μαθήματος ¨-Προηγμένες Τεχνικές Προγραμματισμού¨. Επιπλέον βιβλιογραφία σε θέματα - ταυτόχρονου κυρίως - προγραμματισμού για δίκτυα σε Java ζητήθηκε και δώθηκε προς μελέτη. Τέλος μελετήθηκαν οι κλάσεις του πακέτου java.io που ήταν απαραίτητες στην επικοινωνία με sockets, αλλά και στη δημιουργία του καταγραφέα συμβάντων σε αρχείο (logger) που συμπεριλάβαμε στον πράκτορα για βοήθεια στη διαχείριση του. Παραδείγματα και για τις δύο διαφορετικές αυτές χρήσεις κλάσεων του πακέτου μελετήθηκαν επίσης από τη βιβλιογραφία.

Σχεδιασμός των εφαρμογών. Να σημειώσουμε ότι ο agent αν και περιλαμβάνει ένα μέρος που δρα ως πελάτης και ένα άλλο που δρα ως εξυπηρετητής έχει επιπλέον λειτουργικότητα από το σύνολο αυτών. Επεξεργάζεται τα δεδομένα που έλαβε πριν τα αποστείλει (όπως περιγράφεται παρακάτω) και έχει εντολές που τον παραμετροποιούν κατά την εκκίνηση της λειτουργίας του.

Σχετικά με τον client του πράκτορα που επίσης υλοποιήθηκε έγιναν εμφανείς δύο διαφορετικές προσεγγίσεις που θα μπορούσαν να ακολουθηθούν. Αυτή ενός πολύπλοκου πελάτη που επεξεργάζεται τα δεδομένα που απλώς του προωθεί ο πράκτορας και αυτή ενός απλού πελάτη που επαφίεται στον πράκτορα για την επεξεργασία των δεδομένων. Ακολουθήθηκε το δεύτερο παράδειγμα, μιας και εκτός του ότι προσδίδει απλότητα στην ανάπτυξη -όπως θα εξηγηθεί και παρακάτω- συμφωνεί κατά τη γνώμη μας και περισσότερο με τη φιλοσοφία των πρακτόρων, στους οποιούς ανατίθενται εργασίες για να δώσουν αποτελέσματα. Κατά συνέπεια ο -απλόσ- πελάτης δεν απαιτεί ταυτοχρονισμό στην υλοποίησή του. Ένα χαρακτηριστικό που του προσδώσαμε όμως και ανεβάζει κάπως την πολυπλοκότητα είναι η δυνατότητα παραμετροποίησης κατά την εκκίνηση του.

Ένα ακόμη δίλημμα που συναντήσαμε κατά το σχεδιασμό ήταν ο τρόπος επικοινωνίας με το χρήστη του πράκτορα, αλλά και του πελάτη αυτού. Ο παραδοσιακός για απλές δικτυακές εφαρμογές τέτοιου είδους τρόπος επικοινωνίας

είναι με περάσμα ορισμάτων στη γραμμή εντολών κατά την κλήση του προγράμματος. Μπορεί να μην είναι τόσο φιλικός για μη εξοικειωμένους χρήστες όσο ίσως μια γραφική διεπαφή (Graphical User Interface (GUI) και να μη θεωρείται 100Pure Java ωστόσο προσδίδει απλότητα και στην ανάπτυξη. Για τους λόγους αυτούς προτιμήθηκε στη συγκεκριμένη εφαρμογή και στην παρούσα φάση ανάπτυξης. Φυσικά ακολουθήθηκαν οι συμβάσεις POSIX για τα ορίσματα γραμμής εντολών.

Περιγραφή των εφαρμογών που υλοποιήθηκαν. Τόσο ο πράκτορας όσο και ο πελάτης του είναι σχετικά μικρές εφαρμογές. Έτσι περιλαμβάνουν λίγες μεθόδους και κλάσεις, αφού και οι μεταβλητές που χρησιμοποιούν αυτές είναι κοινές συνήθως. Ο TCPAgent περιλαμβάνει δύο κλάσεις την TCPAgent και την TCPAgentThread, ενώ ο TCPAgentClient μία, την TCPAgentClient.

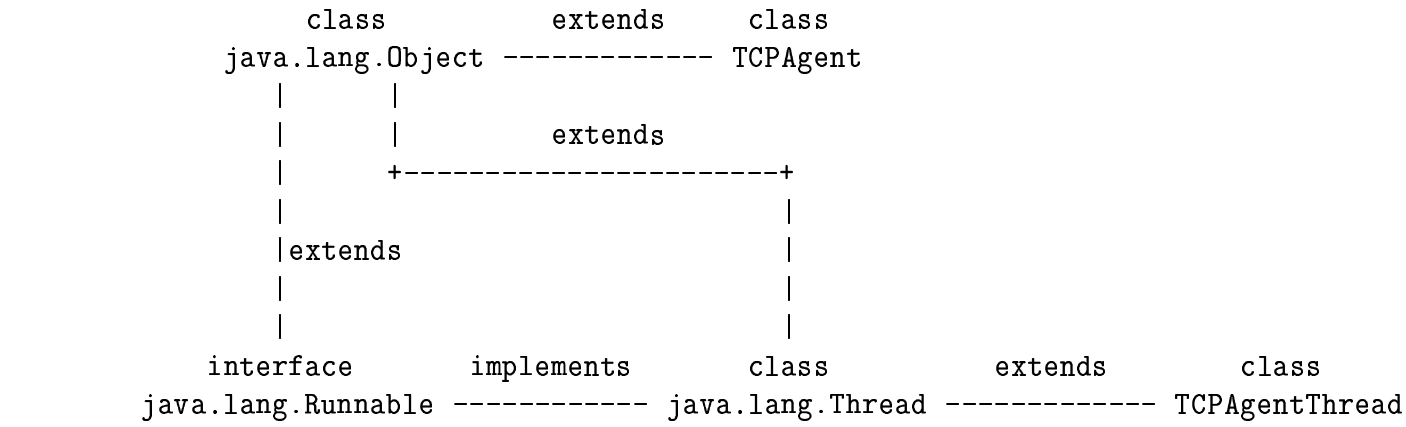
Από τη μεριά του πράκτορα, ένα στιγμιότυπο της κλάσης TCPAgent τυπώνει τα εισαγωγικά μηνύματα, διαχωρίζει τα ορίσματα γραμμής εντολών, θέτει τις αντίστοιχες μεταβλητές και δημιουργεί ένα serverSocket που περιμένει πελάτες να συνδεθούν. Μόλις συμβεί αυτό, δημιουργεί ένα στιγμιότυπο της κλάσης TCPAgentThread. Αυτό συνδέεται με τους time chargen servers, παίρνει τις αποκρίσεις τους, τις επεξεργάζεται, δηλαδή μετατρέπει τον 32-bit integer και το string σε strings, ενώ με βάση το πως κλήθηκε ο TCPAgent δέχεται όσους τυχαίους χαρακτήρες ορίζει ο χρήστης. Κατόπιν περνά τις επεξεργασμένες απαντήσεις στον TCPAgentClient και κρατά αρχείο της σύνδεσης, αν έχει εκκινηθεί με αυτή την εντολή. Να σημειώσουμε ότι με τη μετατροπή σε αλφαριθμητικά και των δύο απαντήσεων απλοποιούμε τον πελάτη που δε βλέπει διαφοροποίηση, αλλά και τη διαδικασία μετάδοσης, αφού ένας τύπος stream αρκεί, ενώ δε χρειάζεται να εισάγουμε και χαρακτήρες διαχωρισμού των απαντήσεων. Επίσης η επιλογή του αριθμού των χαρακτήρων γίνεται στον πράκτορα και μόνο, ώστε να μην ανακύψουν και επιπλέον πολύπλοκα προβλήματα τύπου παραγωγού - καταναλωτή, όπως περίσσεια ή έλλειμα χαρακτήρων στον ενδιάμεσο αποθηκευτικό χώρο.

Από τη μεριά του πελάτη, ένα στιγμιότυπο της κλάσης TCPAgentClient τυπώνει τα εισαγωγικά μηνύματα, διαχωρίζει τα ορίσματα γραμμής εντολών, θέτει τις αντίστοιχες μεταβλητές, συνδέεται στον πράκτορα και παίρνει την απόκριση από το time service (32 - bit integer που αλλάζει συνεχώς) και από το chargen service (ένας ή περισσότεροι χαρακτήρες, ανάλογα με το πως καθορίστηκε η λειτουργία του TCPAgent κατά την εκκίνησή του: Οι χαρακτήρες αυτοί δίνονται από τον chargen server και είναι πάντοτε οι ίδιοι, εμείς επιλέγουμε μερικούς από την αρχή του stream. Προεπιλεγμένη είναι η εμφάνιση ενός

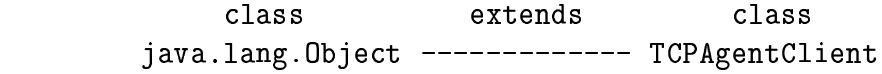
μόνο χαρακτήρα, που αν είναι το κενό μπορεί να δώσει την εσφαλμένη εντύπωση ότι υπάρχει κάποιο πρόβλημα. Αν θέλαμε να αλλάζουν οι χαρακτήρες που παρουσιάζονται, ενώ ο εξυπηρετητής που τους παρέχει θα έμενε ο ίδιος, αρκούσε η εισαγωγή μιας απλής τυχαίας μεθόδου επιλογής μερικών στην πλευρά του πολύπλοκου TCPAgent, μιας και διατηρούμε τον TCPAgentClient) απλό.) Κατόπιν τυπώνει στην οθόνη τις δύο αυτές αποκρίσεις.

Διάγραμμα κλάσεων. Η παραπάνω περιγραφή καθιστά σαφή τα παρακάτω απλά διαγράμματα κλάσεων για τις δύο εφαρμογές, στα οποία φαίνονται και οι σχέσεις γενίκευσης - εξειδίκευσης και συνάθροισης.:

TCPAgent



TCPAgentClient



Υλοποίηση. Ο κώδικας ελέχγθηκε με τους αντίστοιχους Java compilers και Java virtual machines στα ακόλουθα λειτουργικά συστήματα: FreeBSD 4.2-STABLE (jdk1.1.8), Linux 2.2.17 (jdk1.2.2), Microsoft Windows 98 (j2sdk1.3).

Η άδεια κάτω από την οποία τέθηκε ο κώδικας επιλέχθηκε να είναι η GNU General Public License. Ακολουθούν σύντομες οδηγίες χρήσης του εκτελέσιμου κώδικα, παρμένες από τη βοήθεια που δίνουν οι εφαρμογές στο χρήστη:

TCPAgent
Copyright (C) 2000 Thomas Repantis darkzero@otenet.gr
Terminate with Ctrl+C.
java TCPAgent -h for help

Usage: java TCPAgent [-h] [-c NumberOfChars] [-host hostname] [-chargenport portnumber] [-timeport portnumber] [-port2listen portnumber] [-log]

TCPAgent will listen for TCPAgentClients' connections on the specified port (or on port 14000 by default) and once triggered will connect to a time and chargen server on the specified host (or on the localhost by default), on the specified ports (or on timeport 37 and on chargenport 19 by default), will get time and chargen output (specified number of characters or one character by default), send them to the TCPAgentClient and log this action in file TCPAgent.log, if invoked with the -log argument.

TCPAgentClient
Copyright (C) 2000 Thomas Repantis darkzero@otenet.gr
java TCPAgentClient -h for help

Usage: java TCPAgentClient [-h] [-host hostname] [-port portnumber]

TCPAgentClient will connect to a TCPAgent on the specified host (or on the localhost by default), on the specified port (or on port 14000 by default) and will get time and chargen output.

Προβλήματα και εμπειρίες που αποκτήθηκαν. Αρχικά φάνηκε δύσκολη η επιλογή των κατάλληλων αλγορίθμων υλοποίησης για τον πράκτορα και για τον πελάτη του. Αυτή η δυσκολία ξεπεράστηκε με το σαφή καθορισμό του έργου τους. Η επόμενη δυσκολία στο σχεδιασμό συναντήθηκε στην προσπάθεια να ορισθούν αντικείμενα, μιας και οι εργασίες είναι λίγες και χρησιμοποιούν στην πλειοψηφία τους κοινές μεταβλητές. Έτσι ακολουθήθηκε μια συντηρητική, απλή προσέγγιση, που πιθανώς με περισσότερη πείρα στην ανάπτυξη ολοκληρωμένων αντικειμενοστρεφών εφαρμογών να μπορεί να βελτιωθεί. Μια ακόμα δυσκολία υπήρξε στην προσπάθεια κατανόησης των διαφόρων κατασκευών που

περιλαμβάνει η γλώσσα Java για είσοδο και έξοδο, μιας και δεν υπήρχε μεγάλη προηγούμενη εμπειρία. Η δυσκολία αυτή ξεπεράστηκε με μελέτη της βιβλιογραφίας, του API και παραδειγμάτων. Οι κατασκευές της γλώσσας για υλοποιήση δικτυακών εφαρμογών κατανοήθηκαν εύκολα.

Προφανώς οι εμπειρίες που αποκτήθηκαν είναι πολλές και σημαντικότατες. Ενδεικτικά αναφέρουμε την πείρα που αποκτήθηκε στον ορισμό προβλημάτων, στην ανάλυση και το σχεδιασμό ολοκληρωμένων -έστω και μικρού μεγέθουσεφαρμογών, στην προγραμματιστική αντιμετώπιση πραγματικών προβλημάτων, καθώς βέβαια και την καλύτερη κατανόηση και την πρακτική εξοικείωση με τις κατασκευές της γλώσσας Java, ειδικά αυτές που αφορούν δίκτυα και είσοδο / έξοδο. Δε μπορούμε να παραλείψουμε τις γνώσεις που αποκτήθηκαν στον τομέα της ανάπτυξης δικτυακών εφαρμογών (servers, clients, agents) σε αλγοριθμικό επίπεδο, ανεξαρτήτως γλώσσας προγραμματισμού, όπως και εφαρμογών που χρησιμοποιούν ταυτοχρονισμό (concurrency).

Αναφορές .

1. D. Comer- D. Stevens, "Internetworking with TCP/IP vol. III - Client-Server Programming and Applications", 1996
2. Κ. Θραμπουλίδης, ¨Από τη C στην Java - Από τον διαδικαστικό στον object-oriented προγραμματισμό¨, 1999
3. Sun Microsystems Inc., "Java Platform Core API Specification", 2000
4. Sun Microsystems Inc., "Java Platform Class Hierarchy", 2000
5. Sun Microsystems Inc., "The Java Tutorial", 2000
6. Κ. Θραμπουλίδης, ¨Προηγμένες Τεχνικές Προγραμματισμού - Σημειώσεις Μαθήματος¨, 2000
7. H.M. Deitel-P.J. Deitel, "Java How To Program", 1998
8. K. Arnold, J. Gosling, "The Java Programming Language", 1996
9. D. Lea, "Concurrent Programming in Java: Design Principles and Patterns", 1997

Συνημμένα .

Επισυνάπτεται ο πηγαίος κώδικας σε μορφή κειμένου για τις εφαρμογές της εργασίας αυτής (18).

Επίσης σε δισκέττα επισυνάπτονται το παρόν κείμενο σε ηλεκτρονική μορφή, καθώς και ο πηγαίος και ο εκτελέσιμος κώδικας για τις εφαρμογές της εργασίας 18, καθώς και της συμπληρωματικής εργασίας.