ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ανάπτυξη λογισμικού ανοικτής αρχιτεκτονικής διαχείρισης πόρων και υπηρεσιών σε συστήματα ενσωματωμένης λογικής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Λ. Καρρά

Επιβλέπων: Ευστάθιος Συκάς, Καθηγητής

Αθήνα, Οκτώβριος 2004

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ανάπτυξη λογισμικού ανοικτής αρχιτεκτονικής διαχείρισης πόρων και υπηρεσιών σε συστήματα ενσωματωμένης λογικής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Λ. Καρρά

Επιβλέπων: Ευστάθιος Συκάς
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Οκτωβρίου 2004.

.....	.....	.....
Ευστάθιος Συκάς Καθηγητής	Μιχάλης Θεολόγου Καθηγητής	Γεώργιος Στασινόπουλος Καθηγητής

Αθήνα, Οκτώβριος 2004.

Περιεχόμενα

Εισαγωγή

Η εργασία έχει ως αντικείμενο την διαχείριση πόρων και υπηρεσιών σε συστήματα ενσωματωμένης λογικής. Ειδικότερα, στόχο αποτέλεσε η ανάπτυξη ενός συστήματος λογισμικού ανοιχτής αρχιτεκτονικής για ενοποιημένη διαχείριση συστημάτων ενσωματωμένης λογικής.

Ως συστήματα ενσωματωμένης λογικής ορίζουμε τα υπολογιστικά συστήματα που κατασκευάζονται με στόχο την ικανοποίηση συγκεκριμένων αναγκών σε καλά καθορισμένα περιβάλλοντα λειτουργίας. Κλασικά παραδείγματα τέτοιων συστημάτων αποτελούν τα κινητά τηλέφωνα, τα συστήματα ελέγχου των σύγχρονων μέσων μεταφοράς, τα μηχανήματα αυτόματης ανάληψης στις τράπεζες, οι Ψηφιακοί Προσωπικοί Βοηθοί (PDA) και οι δρομολογητές σε πολλά δίκτυα υπολογιστών. Τα συστήματα ενσωματωμένης λογικής, που αποτελούν το μεγαλυτερο μέρος της υπολογιστικής δύναμης του σύγχρονου κόσμου, καλύπτουν διαρκώς μεγαλύτερο φάσμα καθημερινών ανθρώπινων ανάγκων. Η μεγάλη ανάπτυξη στην τεχνολογία των υπολογιστών έχει επηεάσει σημαντικά τις δυνατότητές τους μετατοπίζοντας την στατικότητα του τρόπου σχεδίασης τους σε χαρακτηριστικά που μέχρι σήμερα είχαν μόνο τα ολοκληρωμένα υπολογιστικά συστήματα, όπως μεγάλη σχετικά υπολογιστική ισχύ, ακόμα και ολοκληρωμένα λειτουργικά συστήματα.

Οι αυξημένες ανάγκες διαχείρισης των συστημάτων, λογική συνέπεια της αυξανόμενης λειτουργικότητάς τους, και μια ενοποιημένη προσέγγιση στον τρόπο εξυπηρέτησής τους αποτελούν το αντικείμενο της παρούσας εργασίας.

Η εκπόνηση της εργασίας πραγματοποιήθηκε στα πλαίσια του ευρωπαϊκού ερευνητικού προγράμματος Openrouter IST-2000-28503 στο οποίο συμμετείχε η εταιρία InAccess Networks SA που παρείχε και την πλατφόρμα ανάπτυξης.

Ευχαριστίες

Δημήτρης Οικονόμου <decon@inaccessnetworks.com>
Για την υποστήριξη στη σχεδίαση και ανάπτυξη του συστήματος.

InAccess Networks SA <www.inaccessnetworks.com>
Για την παροχή της απαιτούμενης υλικοτεχνικής υποδομής.

Ευστάθιος Συκάς <sykas@cslab.ntua.gr>
Για την επίβλεψη της εργασίας.

Κεφάλαιο 1

Προδιαγραφές

Η τελευταία εικοσαετία χαρακτηρίστηκε από μια εκρηκτική ανάπτυξη στο χώρο των υπολογιστικών συστημάτων. Τα ακριβά, μικρής υπολογιστικής ισχύος και μεγάλου μεγέθους συστήματα με λογισμικό περιορισμένης λειτουργικότητας της αρχής δεν έδιναν δυνατότητα ευρείας χρήσης, περιορίζοντας ετσι το ενδιαφέρον σε χώρους ακαδημαϊκούς, ερευνητικούς και μεγάλων εταιριών. Με την κατασκευή του πρώτου προσωπικού υπολογιστή (αρχή της δεκαετίας του 1980), την έρευνα στο χώρο των μικροϋπολογιστών και την διάδοση λειτουργικών συστημάτων με αυξημένη λειτουργικότητα(UNIX apo SUN, IBM, HP) και ευχρηστία(Mac, Windows) η κατάσταση άλλαξε. Η έννοια ενός ολοκληρωμένου υπολογιστικού συστήματος άρχισε να αποκτά πρακτικό ενδιαφέρον σε περιβάλλοντα που μέχρι τότε ήταν άγνωστη, όπως σπίτια και μικρές εταιρίες, ενώ η ανάπτυξη αλληλεπιδραστικών υπηρεσιών για αυτά απέκτησε ουσιαστικό νόημα με την ευρεία χρήση τους. Καθοριστικό όλο έπαιξε και η εξάπλωση της χρήσης του διαδικτύου που έδωσε τη δυνατότητα επικοινωνίας μεταξύ απομακρυσμένων συστημάτων και πρόσβαση σε πληοφορίες και υπηρεσίες άγνωστες μεχρι πρότινος. Ενώ λοιπόν σημαντικό μέρος της ιομηχανίας στράφηκε πρός αυτό το χώρο, η αυξανόμενη και πλέον πιο προσιτή υπολογιστική ισχύς πρόσφερε αυξανόμενες δυνατότητες και στα υπολογιστικά συστήματα ενσωματωμένης λογικής(embedded systems), συστήματα με χαρακτηριστικά περισσότερο καθορισμένο σκοπό λειτουργίας και μικρότερη αλληλεπίδραση με τον ανθρώπινο παράγοντα. Δυνατότητες οι οποίες τους έδωσαν σταδιακά χαρακτηριστικά ολοκληρωμένων συστημάτων, όπως λειτουργικά συστήματα γενικής χρήσης και κατ' επέκταση αλληλεπιδραστικές υπηρεσίες. Χαρακτηριστικό παράδειγμα τέτοιων λειτουργικών συστημάτων είναι τα λειτουργικά συστήματα τύπου UNIX που μέχρι πρόσφατα απευθύνονταν μόνο σε ολοκληρωμένα συστήματα. Η προσφορά νέων αλληλεπιδραστικών υπηρεσιών των συστημάτων ενσωματωμένου λογισμικού δημιούργησε την ανάγκη για ύθμιση των χαρακτηριστικών των υπηρεσιών

αυτών. Η διαχείριση αυτή όμως απαιτεί για την κάλυψή της ειδική τεχνογνωσία, γεγονός που έρχεται σε αντίθεση με την φιλοσοφία και τις αρχές των εν λόγω συστημάτων. Δείχνοντας έτσι την ανάγκη για μια νέα λύση προσανατολισμένη στις αρχές και στις κατευθύνσεις αυτές. Ακολουθεί μια σύντομη εξέταση των συστημάτων ενσωματωμένης λογικής σε συνδυασμό με τα λειτουργικά συστήματα -πιο συγκεκριμένα τα λειτουργικά τύπου UNIX- και τις ανάγκες διαχείρισης των συστημάτων αυτών που καταδεικνύει αρχικά την ανάγκη για ένα ειδικό σύστημα διαχείρισης και και κατ΄ επέκταση προδιαγράφει τα χαρακτηριστικά ενός τέτοιου συστήματος.

1.1 Συστήματα ενσωματωμένης λογικής

Τα συστήματα ενσωματωμένης λογικής είναι υπολογιστικά συστήματα που ορίζονται κυρίως με άση την διαφορά τους από τα ολοκληρωμένα υπολογιστικά συτήματα γενικής χρήσης. Ανέκαθεν η αρχιτεκτονική των υπολογιστικών συστημάτων καθοριζόταν από τις ανάγκες τις οποίες σκόπευε να καλύψει και τις υπολογιστικές ανάγκες των εν λόγω αναγκών. Διαφορετικές είναι οι ανάγκες ενός συστήματος που χρησιμοποιείται για την επίλυση μαθηματικών εξισώσεων και ενός συστήματος που χρησιμοποιείται για τον έλεγχο ενός διακόπτη ή ενός συστήματος που χρησιμοποιείται για δακτυλογράφηση. Οι ανάγκες είναι διαφορετικές τόσο στο επίπεδο του επεξεργαστή όσο και στην αρχιτεκτονική του, στα περιφερειακά που ελέγχονται από τον επεξεργαστή αλλά και στις απαιτούμενες ανάγκες από μνήμη και τον τύπο της (πολλά συστήματα ενσωματωμένης λογικής για παράδειγμα δεν χρειάζονται καθόλου αποθηκευτικό χώρο). Από την αρχιτεκτονική των συστημάτων επίσης καθορίζεται και το κόστος τους, οπότε μια αρχιτεκτονική που καλύπτει ακριβώς τις ανάγκες για τις οποίες σχεδίαστηκε οηθά και προς την μείωση του κόστους υλοποίηση της, κάτι ιδιαίτερα σημαντικό σε περιπτώσεις μαζικής παραγωγής.

1.2 Λειτουργικά συστήματα UNIX

Το UNIX είναι ένα λειτουργικό σύστημα που αναπτύχθηκε στις αρχές της δεκαετίας του 1970 στην εταιρία AT/T. Επειδή ο κώδικας των πρώτων εκδόσεων διατέθηκε και εκτός AT/T, στην πορεία εξελίχθηκε σε διόφορες μορφές από το πανεπιστήμιο του Berkley(BSD) και άλλες εταιρίες όπως η Sun Microsystems(SunOS) , IBM(AIX) , HP(HPUX) διατηρώντας πάντα τη ασική φιλοσοφία σχεδίασης του και συνήθως τα περισσότερα χαρακτηριστικά του. Είναι ένα λειτουργικό για πολλούς χρήστες και διεργασίες με

πλούσια χαρακτηριστικά και πολύπλοκες δυνατότητες, ασισμένο όμως σε απλά και δυνατά στοιχεία. Είναι χαρακτηριστικό πως σχεδόν όλα τα σύγχρονα λειτουργικά συστήματα έχουν δανειστεί έννοιες και μηχανισμούς από αυτό ενώ η πλέον γνωστή γλώσσα προγραμματισμού συστημάτων C σχεδιάστηκε και αναπτύχθηκε ταυτόχρονα με την προσπάθεια δημιουργίας του για να λυσει τις ανάγκες που προέκυπταν από αυτό. Μεγάλο πλεονέκτημα του είναι και η δυνατότητα του να προσαρμόζεται σε πολλούς τύπους και αρχιτεκτονικές συστημάτων αλλά και να παραμετροποείται ανάλογα με τις ανάγκες του κάθε συστήματος. Λόγω ακριβώς της εξάπλωσης χρήσης των διαφόρων μορφών στις οποίες έχει εξελιχθεί το UNIX και τα κοινά χαρακτηριστικά τους που ασίζονται στην φιλοσοφία σχεδίασης του, αναφερόμαστε σε αυτά με τον όρο συστήματα UNIX .

1.3 Διαχείριση συστημάτων UNIX

Τα συστήματα UNIX λόγω ακριβώς των πλούσιων χαρακτηριστικών τους αλλά και της μεγάλης δυνατότητας παραμετροποίησής τους απαιτούν γνώσεις για τον τρόπο με τον οποίο μπορούν αυτά να αξιοποιηθούν και να χρησιμοποιηθούν. Αν συνυπολογίσει κανείς και τα εξωτερικά προγράμματα που μπορούν να γίνουν μέρος του και να χρησιμοποιηθούν ως υπηρεσίες, τελικά καταλήγει σε ένα σύστημα με αυξημένες ανάγκες διαχείρισης. Αυτές παραδοσιακά καλύπτονταν από άτομα με ειδικές γνώσεις για συστήματα UNIX καθώς συνήθως τα εν λόγω συστήματα δεν χρησιμοποιούνταν ως υπολογιστές προσωπικής χρήσης. Λόγω της μεγάλης όμως εξάπλωσης της χρήσης των συστημάτων αυτών, σε τελείως διαφορετικούς τομείς και περιβάλλοντα, ακόμα και και ως υπολογιστές προσωπικής χρήσης, άρχισε να διαφαίνεται η ανάγκη για μια διαφορετική προσέγγιση στον τρόπο διαχείρισής τους. Είναι χαρακταριστικό πως στην παρούσα χρονική στιγμή, οι λύσεις που δίνονται στην ανάγκη αυτή είναι σε μεγάλο αθμό εξάρτημένες από την έκδοση για την οποία προορίζονται με αποτέλεσμα να υπάρχουν πολλά διαφορετικά εργαλεία που προσφέρουν σε μεγάλο αθμό και διαφορετικές κατευθύνσεις λύσεων. Κάτι που όπως είναι φυσικό δεν διευκολύνει τον χρήστη, αφού για κάθε διαφορετική διανομή θα πρέπει να εξοικειώνεται με διαφορετικά εργαλεία. Οι λύσεις που κατευθύνονται προς μια γενικότερη αντιμετώπιση των υπηρεσίων που μπορούν να υπάρχουν σε UNIX συστήματα απέχουν αρκετά από μια ανοιχτή αρχιτεκτονική που θα έδινε τη δυνατότητα εισαγωγής νέων υπηρεσιών από τρίτους, ενώ οι δυνατότητες που δίνουν είναι περιορισμένες από την έλλειψη ενός κοινού τρόπου αντιμετώπισης των πόρων διαχείρισης.

1.4 Προδιαγραφές ειδικού συστήματος διαχείισης

Το υπό μελέτη ειδικό σύστημα διαχείρισης για συστήματα UNIX πρέπει να έχει δύο κύρια χαρακτηριστικά. Την ενοποίηση του τρόπου διαχείρισης όλων των πόρων και υπηρεσιών με έναν κοινό τρόπο αντιμετώπισής τους και την διαμόρφωση ενός ανοιχτής και κατανεμημένης αρχιτεκτονικής συστήματος για την εισαγωγή και διαχείριση νέων υπηρεσιών. Ο σκοπός είναι τελικά, να μπορεί ο πάροχος της υπηρεσίας με εύκολο τρόπο χρησιμοποιώντας μια συγκεκριμένη διαδικασία να ορίζει τον τρόπο με τον οποίο η υπηρεσία μπορεί να γίνει αντικείμενο διαχείρισης χωρίς την ανάγκη για επεκτάσεις στην αρχιτεκτονική του συστήματος ή στην υλοποίηση του. Η διαδικασία αυτή θα πρέπει να περιλαμβάνει χρήση ήδη υπάρχοντων τεχνολογιών ώστε να μπορεί να χρησιμοποιηθεί χωρίς κόστος εκπαίδευσης από οποιονδήποτε έχει κάποιες βασικές γνώσεις πάνω στον τρόπο λειτουργίας των συστημάτων UNIX . Επίσης πρέπει να έχει ως αποτέλεσμα την προσφορά μιας ή περισσοτέρων διεπαφών στον χρήστη, ώστε με εύκολο τρόπο να γίνεται τελικά η διαχείριση των συστημάτων. Πολλές διεπαφές χρειάζονται ώστε να μπορεί να χρησιμοποιηθεί από πολλά περιβάλλοντα, ενώ η χρήση ήδη υπάρχοντων προγραμμάτων θα μπορούσε να διευκολύνει και την αποφυγή χρήσης ειδικού λογισμικού από την πλευρά του χρήστη.

Κεφάλαιο 2

Πλατφόρμα ανάπτυξης

Το σύστημα διαχείρισης υλοποιήθηκε στο πρωτότυπο σύστημα Openrouter που αναπτύχθηκε από την inAccess Networks . Το σύστημα Openrouter αναπτύχθηκε στην πλατφόρμα Linux/StrongARM με ανοιχτή αρχιτεκτονική τόσο στο επίπεδο του λογισμικού (software) όσο και στο επίπεδο του υλικού (hardware). Ακολουθεί μια σύντομη περιγραφή της πλατφόρμας και των προδιαγραφών της καθώς και μια συνοπτική περιγραφή των αναγκών σε επίπεδο διαχείρισης.

2.1 Υλικό

Η σχεδίαση σε επίπεδο υλικού φαίνεται στο σχήμα 2.1.1 με κύρια χαρακτηριστικά τον επεξεργαστή StrongARM , μνήμες τύπου flash , θύρες τύπου Ethernet καθώς και σύγχρονες και ασύγχρονες σειριακές θύρες. Πιο αναλυτικά το σύστημα περιλαμβάνει:

- επεξεργαστή RISC 200MHz Intel StrongARM SA1100
- 64 MB μνήμης τύπου SDRAM
- 32 MB μνήμης τύπου Flash
- 4 θύρες Ethernet 10/100 Mbit/sec 802.3 IEEE
- 2 σύγχρονες σειριακές θύρες υψηλής ταχύτητας (εώς 2 Mbit/sec)
- 2 ασύγχρονες σειριακές θύρες τύπου RS-232
- 2 θύρες PCMCIA

Για την υλοποίηση των λογικών κυκλωμάτων επικοινωνίας μεταξύ των υποσυστημάτων και των διεπαφών τους χρησιμοποιήθηκε μια προγραμματιόμενη μονάδα Xilinx FPAGA .

Σχήμα 2.1.1: Υλικό Openrouter

2.2 Λογισμικό

Το λειτουργικό σύστημα της πλατφόρμας είναι το λειτουργικό σύστημα Linux , ένα σύστημα με κύρια χαρακτηριστικά την παραμετροποίηση του, το μεγάλο εύρος υλικού που καλύπτει, τη διαρκή ανάπτυξη και εξέλιξη του, τον ανοιχτό και ελεύθερο κώδικα του καθώς και τη δυνατότητα χρήσης του για εμπορικές και μη εφαρμογές χωρίς κοστος.

Ορισμένα από τα χαρακτηριστικά του λειτουργικού συστήματος Linux είναι τα ακόλουθα:

- υποστήριξη πολλαπλών χρηστών

- multithreading αρχιτεκτονική

- multitasking

- εικονική μνήμη

- παραμετροποίηση της λειτουργίας με χρήση ανεξάρτητων οντοτήτων λογισμικού (modules)

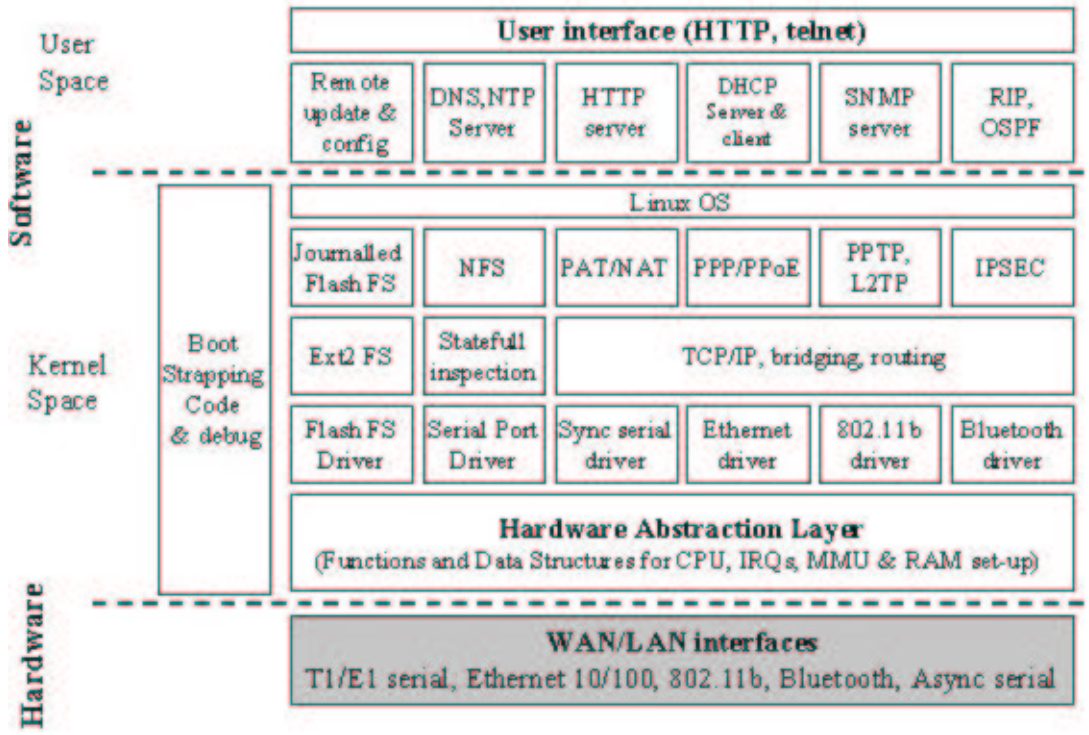
- υποστήριξη πλήθους δικτυακών πρωτοκόλλων (tcp, ipv4, ipv6, ethernet, ppp, bluetooth, isdn, xdsl)

- υποστήριξη πολλών συστημάτων διαχείρισης αρχείων (ext, ext2, ext3, hfs, hpfs, jfs, minix, msdos, nfs, reiserfs, vfat, xfs)

Το Linux μεταξύ των άλλων πλατφόρμων υλικού, υποστηρίζει και τα συστήματα ασισμένα στη σειρά επεξεργαστών ARM τύπου RISC της εταρείας Intel , όπως είναι και ο επεξεργαστής SA 1100 που χρησιμοποιεί ο Openrouter . Ο πυρήνας που έχει χρησιμοποιηθεί είναι η έκδοση 2.4.18 του Linux για τον arm ο οποίος διατηρεί όλα τα χαρακτηριστικά που αναφέρθηκαν, εκτός από μικρές αλλαγές σε ορισμένους οδηγούς συσκευών για καλύτερη υποστήριξη των υποσυστημάτων υου Openrouter .

Πάνω από το λειτουργικό σύστημα τρέχουν εφαρμογές που στην πλειοψηφία τους αοπτελούν εξυπηρετητές για δικτυακές και μη υπηρεσίες. Και οι εφαρμογές αυτές είναι ελεύθερου λογισμικού και το μεγαλύτερο ποσοστό προέρχονται από το GNU project . Στο επίπεδο αυτό τρέχουν και εφαρμογές εγκατάστασης καινούριων πακέτων λογισμικού, όπως και το σύστημα διαχείισης του Openrouter .

Στο σχήμα 2.2.1 φαίνεται η αρχιτεκτονική του λογισμικού του συστήματος.



Σχήμα 2.2.1: Σχηματικό διάγραμμα λογισμικού Openrouter

2.3 Προδιαγραφές

Ο Openrouter σχεδιάστηκε για να καλύψει τις δικτυακές ανάγκες μια μικρής εώς και μεσαίας επιχείρησης για πρόσβαση σε δίκτυα ευρείας ώνης. Οι κυριότερες λειτουργίες που πρέπει να παρέχει είναι η δρομολόγηση(routing), η επιλεκτική πρόσβαση(firewall) και η δυνατότητα εύκολης διαχείρισης.

Η λειτουργία δρομολόγησης αφορά διασύνδεση ετερογενών και μη δικτύων, που σημαίνει μετατροπή της πληροφορίας ανάλογα με το χρησιμοποιηούμενο πρωτόκολλο και προώθηση αυτής διαμέσου των δικτυακών επαφών. Η πλατφόρμα του Openrouter παρέχει όλη την απαιτούμενη υποδομή τόσο σε επίπεδο υλικού (σειριακές θύρες, θύρες PCMCIA , θύρες Ethernet) όσο και σε επίπεδο λογισμικού με το λειτουργικό σύστημα Linux .

Η λειτουργία του ελέγχου πρόσβασης είναι εξίσου σημαντική και αφορά την έλεγχο της πρόσβασης στο εσωτερικό δίκτυο για λόγους ασφάλειας. Οι υπολειτουργίες που απαιτούνται για τον έλεγχο είναι ο χωρισμός του δικτύου, ο μετασχηματισμός διευθύνσεων και ο έλεγχος των συνδέσεων(firewall).

Στην υπολειτουργία του χωρισμού του δικτύου, το δίκτυο χωρίζεται σε

ώνες πρόσβασης ανάλογα με τις ανάγκες. Συνήθως υπάρχει μια ενδιάμεση ώνη που ρίσκονται οι εξυπηρετητές που πρέπει να είναι προσβάσιμοι από το εξωτερικό δίκτυο και μία εσωτερική ώνη όπου ρίσκεται τερματικοί σταθμοί και εξυπηρετητές που δεν χρειάζεται να είναι προσβάσιμοι από έξω.

Στην υπολειτουργία του μετασχηματισμού διευθύνσεων υπάρχει η δυνατότητα να οριστούν στο εσωτερικό δίκτυο διευθύνσεις που έχουν τοπική σημασία μόνο. Έτσι επιτυγχάνεται και οικονομία στο χώρο διευθύνσεων αλλά και εξασφαλίζεται μεγαλύτερο επίπεδο ασφαλείας για το εσωτερικό δίκτυο αφού οι κόμβοι γίνονται απροσπέλαστοι από το εξωτερικό δίκτυο.

Στην υπολειτουργία του ελέγχου των συνδέσεων υλοποιείται στον μεγαλύτερο αθμό η πολιτική ασφαλείας του δικτύου, με τον ορισμό κανόνων πρόσβασης από και προς το εσωτερικό η και το ενδιάμεσο δίκτυο. Για την υλοποίηση αυτών των κανόνων απαιτείται έλεγχος όλων των εισερχόμενων και εξερχόμενων συνδέσεων. Ανάλογα με το αν η ητούμενη σύνδεση συμφωνεί ή όχι με τους κανόνες που έχουν τεθεί, έχουμε πραγματοποίηση ή απόρριψη αυτής αντίστοιχα.

Η λειτουργία της διαχείρισης και ύθμισης των παραμέτρων λειτουργίας του συστήματος αφορά την εγκατάσταση καινούριων πακέτων λογισμικού στο σύστημα καθώς και τη ύθμιση των παραμέτρων όλων των προαναφερθέντων λειτουργιών με έναν εύκολο και αυτοματοποιημένο τρόπο. Η εγκατάσταση νέων ή ενημερωμένων εκδόσεων λογισμικού γίνεται χρησιμοποιώντας ειδικό σύστημα διανομής και εγκατάστασης πακέτων λογισμικού. Η ύθμιση των παραμέτρων λειτουργίας όλων των προηγούμενων λειτουργιών καθώς και μεικών ακόμα υλοποιείται με το σύστημα διαχείρισης που αναπτύχθηκε στα πλαίσια της διπλωματικής. Στην επόμενη ενότητα περιγράφονται πιο αναλυτικα οι ανάγκες διαχείρισης που προκύπτουν από τις προαναφερθείσες λειτουργίες.

2.4 Ανάγκες διαχείρισης

Οι ανάγκες διαχείρισης της συγκεκριμένης πλατφόρμας όπως αυτές προέκυψαν από τις προδιαγραφές του συστήματος αλλά και τις δυνατότητες που προσέφερε το ίδιο το λειτουργικό σύστημα συνοψίζονται στα εξής ασικά σημεία:

- Δυνατότητα διαχείρισης της χρήσης του συστήματος από πολλούς χρήστες.

Η ύπαρξη πολλών χρηστών στο σύστημα εκτός από τη δυνατότητα που δίνει για μεγαλύτερη ανεξαρτησία και διαχωρισμό των δυνατοτήτων χρήσης

του μηχανήματος από διαφορετικούς χρήστες, αποτελεί και μια συνήθη πρακτική ασφαλείας σε τέτοιου είδους συστήματα. Έτσι η ασφάλεια του συνόλου των υπηρεσίων και του ίδιου του συστήματος δεν εξαρτάται αποκλειστικά από τυχόν αδυναμίες σε υλοποιήσεις υπηρεσιών σε λογισμικό. Είναι αναγκαία η δυνατότητα πρόσθεσης και αφαίρεσης χρηστών, ο καθορισμός των δυνατοτήτων τους μέσα στο σύστημα καθώς και καθορισμός κοινών ομάδων χρηστών με κοινά προνόμια και περιορισμούς.

- Δυνατότητα διαχείρισης των υπάρχοντων δικτυακών και σειριακών διεπαφών του συστήματος.

Η ύπαρξη πολλών δικτυακών και σειριακών διεπαφών στο σύστημα, όπως αυτή καθορίζεται από τις προδιαγραφές του συστήματος, καθιστά απαραίτητη τη δυνατότητα διαχείρισης των χαρακτηριστικών τους, όπως για παράδειγμα την διεύθυνση που αντιστοιχεί σε μια δικτυακή διεπαφή, το πρωτόκολλο που χρησιμοποιείται για επικοινωνία και αν αυτή είναι ενεργή ή όχι.

- Δυνατότητα διαχείρισης της επιλεκτικής πρόσβασης, του χωρισμού του δικτύου και του μετασχηματισμού διευθύνσεων.

Η ύπαρξη της λειτουργίας της επιλεκτικής πρόσβασης καθορίζεται από ένα σύνολο κανόνων που αφορούν ένα μεγάλο και δύσκολα διαχειρίσιμο πλήθος παραμέτρων. Είναι απαραίτητη λοιπόν η εμφάνιση αυτών με έναν πιο ενοποιημένο και αυτόματο τρόπο, που να δίνει τη δυνότητα εύκολης προσθήκης και αφαίρεσης κανόνων, καθώς και ομαδοποιήσης τους.

Για την καλύτερη ομαδοποίηση των παραπάνω κανόνων σημαντικό όλο παίζει και ο χωρισμός του δικτύου που επίσης πρέπει να υλοποείται με έναν πιο αυτοματοποιημένο τρόπο, δίνοντας ταυτόχρονα τη δυνατότητα για καλύτερη ομαδοποίηση αυτών.

Επίσης πρέπει να δίνεται δυνατότητα διαχείρισης της λειτουργίας του μετασχηματισμού διευθύνσεων.

- Δυνατότητα διαχείρισης της δρομολόγησης

Η λειτουργία της δρομολόγησης, όπως αυτή προκύπτει από τις προδιαγραφές του συστήματος, πρέπει επισης να αποτελεί ως ασική λειτουργία του μηχανήματος μια εύκολα διαχειρίσιμη λειτουργία, ανεξάρτητα από τις διαφορές των υποδικτύων στα οποία υλοποιείται και τις ιδιαιτερότητες των τμήματα του εσωτερικού δικτύου.

- Δυνατότητα διαχείρισης άλλων υπηρεσιών

Δυνατότητα διαχείρισης πρέπει να προσφέρεται και σε άλλες υπηρεσίες που τυχόν τρέχουν στο σύστημα. Παράδειγμα η υπηρεσία αυτόματης ανάθεσης IPδιευθύνσεων σε μηχανήματα του εσωτερικού δικτύου για την οποία.

Πρέπει να σημειωθεί στο σημείο αυτό πως σημαντικό όλο στη λειτουργία της διαχείρισης του συστήματος αποτελεί η δυνατότητα ενοποιημένης αντιμετώπισης τυχόν υπηρεσιών του συστήματος, με στόχο την εύκολη επέκταση του συστήματος από τρίτους ώστε να υποστηρίζει νέες υπηρεσίες.

Κεφάλαιο 3

Περιγραφή συστήματος

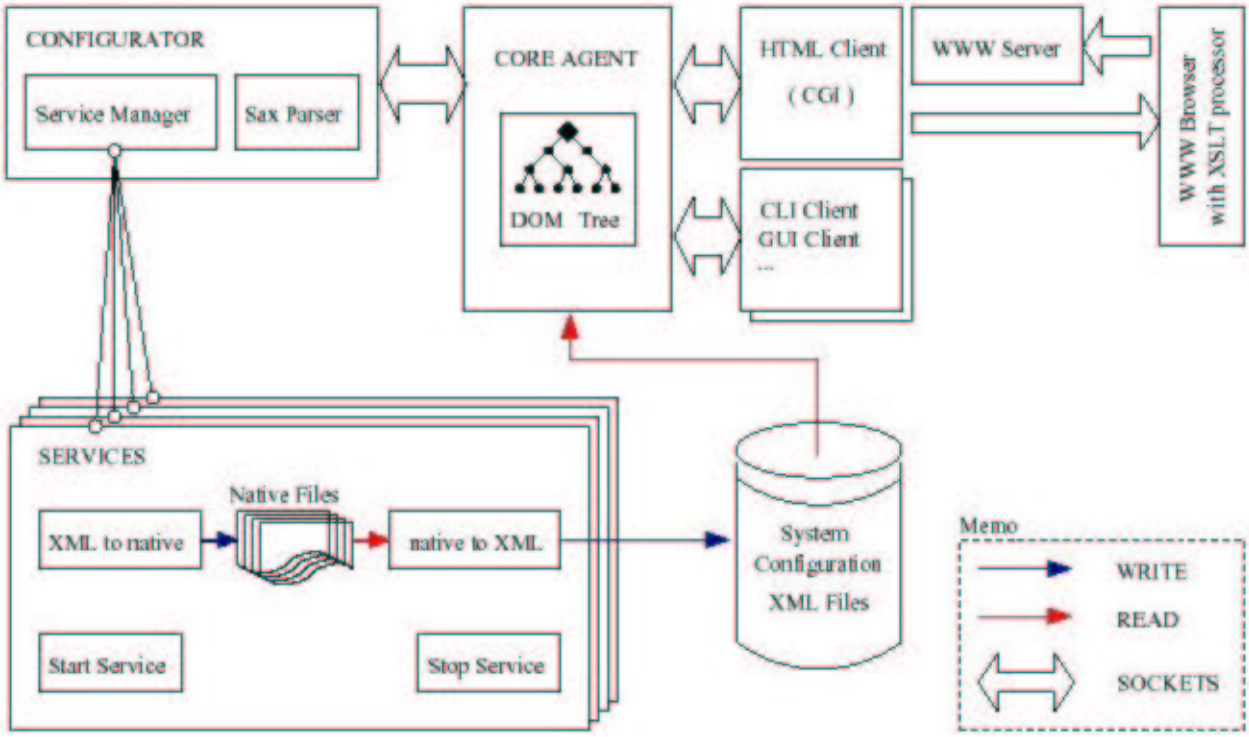
Η υλοποίηση του συστήματος ασίστηκε στη χρησιμοποίηση διάφορων καθιερωμένων τεχνολογιών. Οι τεχνολογίες XML, XSL DTD χρησιμοποιήθηκαν για την οργάνωση, ομαδοποίηση, αποθήκευση, έλεγχο και διαμόρφωση των δεδομένων, προσφέροντας στο σύστημα μια ανοιχτή αρχιτεκτονική και τη δυνατότητα για εύκολη επέκταση του. Δεδομένου ότι όλο το σύστημα αναπτύχθηκε στη γλώσσα προγραμματισμού C++ με την χρησιμοποίηση της ιβλιοθήκης libstdc++, για την υλοποίηση των τεχνολογιών αυτών χρησιμοποιήθηκε η ευρέως διαδεδομένη ιβλιοθήκες λογισμικού xerces που προσφέρεται ως ελεύθερο λογισμικό από το Apache Software Foundation (www.apache.org) . Για την χρήση του προγράμματος από τελικούς χρήστες χρησιμοποιήθηκαν ο εξυπηρετητής για http thttpd με κύρια πλεονεκτήματα τις πολύ καλές μετρήσεις για την απόδοσή του καθώς και την εύκολη χρησιμοποίησή του σε περιβάλλοντα ενσωματωμένης λογικής. Τέλος για την μετατροπή των δεδομένων του λειτουργικού συστήματος και των υπηρεσιών σε δεδομέναν του συστήματος καθώς και για πολλές άλλες ενέργειες διαχείρισης χρησιμοποιήθηκαν διάφορα εργαλεία λογισμικού για UNIX όπως bash, awk.

Η χρησιμοποίηση των παραπάνω τεχνολογιών εκτός από τα άλλα πλεονεκτήματα δίνει και στο σύστημα τη δυνατότητα να πετύχει τον αρχικό του στόχο αναφορικά με εύκολη επέκταση του για διαχείριση νέων υπηρεσιών. Η ανεξάρτητη οργάνωση των επιμέρους τμημάτων της αρχιτεκτονικής δίνει τη δυνατότητα σε τρίτους με ασικές γνώσεις των τεχνολογιών αυτών να ενσωματώσουν στο σύστημα υποστήριξη για νέες υπήρεσιες χωρίς την ανάγκη για επιπλέον κώδικα ή επεμβάσεις στον κώδικα του συστήματος. Γιατον λόγο αυτό η περιγραφή του συστήματος χωρίζεται σε δύο μέρη :την περιγραφή της αρχιτεκτονικής και της ασικής λειτουργίας και την περιγραφή των τμημάτων που απαρτίζουν την υποστήριξη μιας υπηρεσίας όπως και τον τρόπο υλοποίησης τους.

3.1 Αρχιτεκτονική συστήματος

Το σύστημα αποτελείται από τέσσερες ασικές οντότητες:

- Τον πυρήνα του συστήματος διαχείρισης (Core Configurator)
- Τον κεντρικό πράκτορα (Core Agent)
- Τον εξυπηρετητή πελατών (Client Agent)
- Τον πελάτη (Client)



Σχήμα 3.1.1: Αρχιτεκτονική συστήματος διαχείρισης

- Ο πυρήνας του συστήματος (τον οποίο στο εξής θα αναφέρουμε ως Configurator) είναι το μέρος του συστήματος που αναλύει τις οντότητες προς διαχείριση, συγκεντρώνει τις απαραίτητες πληροφορίες για αυτές, προσφέρει τα δεδομένα που προκύπτουν στο υπόλοιπο σύστημα, δέχεται διαχειριστικές εντολές από αυτό, τις εφαρμόζει και ενημερώνει το υπόλοιπο σύστημα για το αποτέλεσμα των εντολών αυτών.

- Ο κεντρικός πράκτορας (τον οποίο στο εξής θα αναφέρουμε ως Core Agent) συγκεντρώνει με τη οήθεια του πυρήνα σε μια κεντρική άση δεδομένα για όλες τις υπηρεσίες προς διαχείριση και τις προσφέρει στους ε-

ξυπηρετητές πελατών, υπό τη μορφή συγκεκριμένων μεταβλητών με τιμές ή ειδικών λειτουργιών που μπορεί ο κάθε πελάτης να χρησιμοποιήσει με παραμετροποιήσιμο τρόπο.

- Ο εξυπηρετητής πελατών που έχει ως σκοπό την εξηπηρέτηση μιας συγκεκριμένης κατηγορίας πελατών, με κύριο χαρακτηριστικό κατηγοριοποίησης το είδος διαμόρφωσης των δεδομένων. Έτσι μπορεί να υπάρχουν πολλοί εξυπερητές πελατών ταυτόχρονα, όπως εξηπηρετητής για HTTP πελάτες, εξυπηρετητής για απλούς command-lineπελάτες και άλλοι. Ουσιαστικά, ο κάθε είδους εξυπηρετητής πελατών αναλαμβάνει να μεταφράσει την εντολή που δέχεται από τον πελάτη στην γενική της μορφή που απαιτείται από τον κεντρικό πράκτορα και αντίστροφα τα δεδομένα που προσφέρονται από το σύστημα στην ειδική διαμόρφωση που απαιτεί ο πελάτης.

- Ο πελάτης είναι το πρόγραμμα που χρησιμοποιεί ο χρήστης του συστήματος και μπορεί να είναι ένας ειδικός πελάτης κατασκευασμένος για το συγκεκριμένο σκοπό, απαιτώντας έτσι έναν ειδικό εξηπηρετητή πελατών, είτε ένας γενικής χρήσης πελάτης όπως ένας HTTP πελάτης.

3.1.1 Πυρήνας συστήματος διαχείρισης

Ο Core Configurator αποτελείται από δύο βασικές οντότητητες, τον Service Managerκαι τον Agent Handler .

Ο Service Manager συγκεντρώνει την απαραίτητη πληροφορία για όλες τις προς διαχείριση υπηρεσίες. Ο σκοπός του είναι αφενός να μπορέσει τελικά να προσφέρει χρήσιμη πληροφορία για μια υπηρεσία (όπως αυτή έχει προδιαγραφεί από τον δημιουργό της) στο υπόλοιπο σύστημα, προδιαγράφοντας ταυτόχρονα και τις διαχειριστικές εντολές για αυτήν, και αφετέρου να δεχτεί και να εφαρμόσει τις εντολές αυτές από τον κεντρικό πράκτορα. Η απαραίτητη πληροφορία για κάθε υπηρεσία είναι καλά ορισμένη, ενώ ταυτόχρονα αφήνει την επιλογή στον δημιουργό της υπηρεσίας για τα δεδομένα που θέλει να προσφέρει στους πελάτες και τις διαχειριστικές εντολές που μπορεί να δεχτεί η υπηρεσία αυτή. Συγκεκριμένα η διαχείριση μιας υπηρεσίας αποτελείται από τα δεδομένα που τελικά προσφέρονται στον πελάτη, την αλλαγή των δεδομένων αυτών από τον πελάτη σε μια μορφή κλειδιού-τιμής, την εφαρμογή ειδικών εντοων στην υπηρεσία και την εκκίνηση ή τερματισμό της. Κάθε ένα από αυτά τα μέρη διαχείρισης της υπηρεσίας είναι ανεξάρτητα το ένα από το άλλο, ενώ ένα μέρος μπορεί να υπερισχύει των άλλων σε χρησιμότητα, που μπορεί και να μην υπάρχουν. Η διαχείριση μιας υπηρεσίας μπορεί για παράδειγμα να εξαντλείται στην εκκίνηση και στον τερματισμό της, χωρίς την ανάγκη για αλλαγή άλλων δεδομένων ή χρήση άλλων διαχειριστικών εντολών,

ή να αποτελείται από την δυνατότητα εφαρμογής απλά κάποιων εντολών. Η πληροφορία της υπηρεσίας για να ενσωματωθεί στο σύστημα πρέπει πρώτα να συγκεντρωθεί σε μορφή XML , ενώ για να αλλάξει πρέπει αντίστροφα να υπάρξει μετατροπή από την XML μορφή στην μορφή της υπηρεσίας. Για τον σκοπό αυτό χρησιμοποιούνται δύο εξωτερικά προγράμματα για την μετατροπή της πληροφορίας. Το πρώτο, που ονομάζεται native2xml , αναλαμβάνει την μετατροπή των δεδομένων από την μορφή με την οποία ρίσκονται στο σύστημα σε XML μορφή η οποία πρέπει να συμφωνεί με τους περιορισμούς που επιβάλλονται από τον Service Manager . Το δεύτερο, που ονομάζεται xml2native , αναλαμβάνει την μετατροπή των δεδομένων από την XML μορφή στην μορφή του συστήματος. Περισσότερες πληροφορίες για τα προγράμματα αυτά θα αναφερθούν στην ενότητα των υπηρεσιών. Εκτός όμως από την πληροφορία που προσφέρεται στον διαχειριστή, η υπηρεσία μπορεί να καθορίσει και ειδικές διαχειστικές εντολές, με δυνατότητα παραμετροποίησης, για τις περιπτώσεις που μια απλή αλλαγή τιμών των χαρακτηριστικών της υπηρεσίας δεν αρκεί. Οι εντολές αυτές καταλήγουν στην εκτέλεση εξωτερικών προγραμμάτων που ορίζονται από τον παροχό της υπηρεσίας και θα αναλύονται εκτενέστερα στην ενότητα των υπηρεσιών. Τέλος, πρέπει να προσφέρεται και η δυνατότητα δύο ειδικών διαχειριστικών εντολών που ενεργοποιούν και απενεργοποιούν την ίδια την υπηρεσία και μπορούν να χρησιμοποιούνται μετά από κάθε διαχειριστική εντολή αν αυτή το απαιτεί. Όλος αυτός ο ορισμός των διαχειρίσιμων δυνατοτήτων για μια υπηρεσία γίνεται με τη οήθεια ενός XML αρχείου του οποίου η μορφή είναι καλά ορισμένη και περιγράφεται και αυτή στην ενότητα των υπηρεσιών.

Με την εκκίνηση του, ο Service Manager συλλέγει λοιπόν την περιγραφή αυτή των υπό διαχείριση υπηρεσιών και εκτελεί την μετατροπή των προσφερόμενων δεδομένων σε XML μορφή με τη οήθεια του εξωτερικού native2xml προγράμματος. Έτσι η πληροφορία υπάρχει στο σύστημα και μπορεί να χρησιμοποιηθεί από τον Core Agent ώστε να προσφερθεί στους πελάτες.

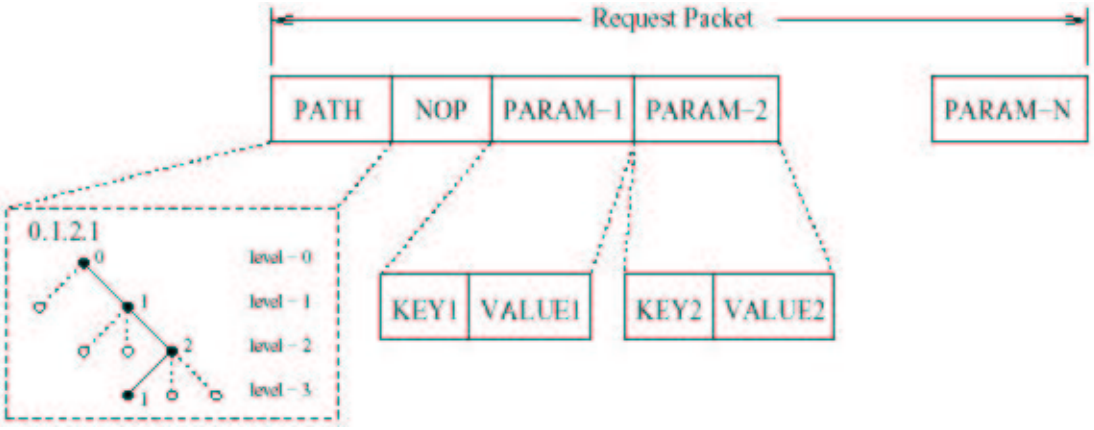
Η δεύτερη οντότητα του Core Configurator , ο Agent Handler αναλαμβάνει την εξύπηρέτηση του Core Agent υπό τη μορφή εκτέλεσης διαχειριστικών εντολών ή αλλαγή των χαρακτηριστικών μιας υπηρεσίας. Ελέγχει αν κάποιο από τα δεδομένα της υπηρεσίας έχουν αλλάξει ή αν έχει κληθεί κάποια διαχειριστική εντολή και χρησιμοποιεί τον Service Manager για να εφαρμόσει τις εντολές αυτές. Ο Service Manager έχοντας ορισμένη όλη την απαραίτητη πληροφορία για την εκτεέλεση των εντολών αυτών, μπορεί να καλέσει τα εξωτερικά προγράμματα με τις κατάλληλες παραμέτρους, να ελέγξει τα αποτελεσματά τους και να τα προσφέρει στον Agent Hanler . Ο Agent Handler εξυπηρετεί σειριακά τις αιτήσεις του Core Agent έχοντας κάποιο μέγιστο χρόνο εξυπηρέτησης. Αν ο μέγιστος χρόνος εξηπηρέτησης χρησιμοποιηθεί χωρίς να έχει ολοκληρωθεί η εκτέλεση της εντολής, ο Agent Handler ενημερώνει

τον Core Agent και περιμένει νέα εντολή. Έτσι επαφείεται στον πελάτη και στα εξωτερικά προγράμματα η εκ των υστέρων ενημέρωση για το πραγματικό αποτέλεσμα της πράξης.

Για την επικοινωνία Agent Handler - Core Agent χρησιμοποείται ένα απλό πρωτόκολλο μεταφοράς XML πληροφορίας πάνω από το UNIX domain socketsπου μπορούν να χρησιμοποιηθεί μόνο για τοπικές συνδέσεις, εφόσον και δύο οντότητες ρίσκονται στο ίδιο μηχάνημα.

3.1.2 Κεντρικός Πράκτορας

Ο Core Agent ενεργοποιείται μετά την συγκέντρωση από τον Core Configurator της πληροφορίας για τις υπό διαχείριση υπηρεσίες και την αποθήκευση τους στην κεντρική άση. Παίρνει από την κεντρική άση την πληροφορία όλου του συστήματος, όπως αυτή έχει διαμορφωθεί από τον Core Configurator και αρχίζει να δέχεται αιτήσεις εξυπηρέτησης από τους υπάρχοντες Client Agents . Οι αιτήσεις γίνονται με ένα απλό πρωτόκολλο που περιγράφεται στο ακόλουθο σχήμα.



Σχήμα 3.1.2: Πρωτοκόλλο επικοινωνίας κεντρικού πράκτορα με τον εξυπηρετητή πελάτη

Το πρώτο πεδίο που ονομάζεται PATHπεριγράφει τη θέση μέσα στην κεντρική άση του συστήματος στην οποία αναφέρονται οι αλλαγές που ακολουθούν. Είναι ένα αλφαριθμητικό πεδίο που περιγράφει με αριθμούς χωρισμένους με τελείες το άθος μέσα στην XML άση και την θέση του κλειδιού.

Το δεύτερο πεδίο που ονομάζεται NOPπεριγράφει τον αριθμό των αλλαγών στη συγκεκριμένη θέση που ακολουθούν με την μορφή κλειδιού-τιμής.

Από το τρίτο πεδίο και μετά ακολουθούν τόσα πεδία όσα αναφέρονται στο πεδίο NOPμε τη μορφή κλειδιού-τιμής. Το πρώτο αλφαριθμητικό πεδίο που

αναφέρεται στο κλειδί πρέπει να είναι μοναδικό ώστε να μπορεί να ρεθεί μέσα στη συγκεκριμένη θέση της άσης.

Αφού ο Core Agent δεχτεί την αίτηση, ρίσκει μέσα στο XML δέντρο της άσης την θέση στην οποία αναφέρονται οι αλλαγές. Μέσα στην συγκεκριμένη θέση, που είναι ένας συγκεκριμένος κόμβος του δέντρου, ψάχνει για τα κλειδιά που αναφέρονται στην αίτηση. Σε κάθε κλειδί που ρίσκει αλλάζει την τιμή του, που είναι συνήθως ένα απλό ASCII αλφαριθμητικό, ή ένα αλφαριθμητικό μέσα σε έναν κόμβο που ονομάζεται value. Αφού ελέγξει όλα τα κλειδιά που αναφέρονται στην αίτηση, και μία τουλάχιστον πραγματική αλλαγή έχει γινει σε σύγκριση με τις προηγούμενες τιμές των κλειδιών, στέλνει μια νέα αίτηση στον Core Configurator περιγράφοντας τις εν λόγω αλλαγές. Η αίτηση χρησιμοποιεί το αλλαγμένο μέρος της άσης και ουσιαστικά μεταδίδει απλά XMLδεδομένα πάνω από το υπάρχον κανάλι. Μετά την επεξεργασία της αίτησης του Core Agent ο Configurator επιστρέφει το αποτέλεσμα, με τη μορφή ενός κενού αλφαριθμητικού, σε περίπτωση επιτυχίας, ή μια περιγραφή του λάθους σε περίπτωση αποτυχίας.

Σε αυτό το σημείο ο Core Agent πρέπει να έχει διαχωρίσει αν η αλλαγή στη άση αφορούσε ειδική διαχειριστική εντολή ή αλλαγή χαρακτηριστικών της υπηρεσίας. Στην περίπτωση αλλαγής των χαρακτηριστικών της υπηρεσίας και αν η αλλαγή ήταν επιτυχής, ο Core Agent ελέγχει αν είναι απαραίτητη η επανεκκίνηση της υπηρεσίας, που δηλώνεται σε κάθε μέρος των δεδομένων της άσης. Αν είναι, την εκτελεί και επιστρέφει επιτυχές αποτελέσμα στον Client Agent που έκανε την αίτηση. Σε περίπτωση αποτυχίας, το μέρος της άσης που έχει αλλαχτεί πρέπει να ανανεωθεί από τη άση του συστήματος εφόσον δεν ανταποκρίνεται στην πραγματικότητα. Στην δεύτερη περίπτωση που η αίτηση αφορούσε ειδική διαχειριστική εντολή, αν αυτή ήταν επιτυχής, το μέρος της άσης που αφορούσε την εντολή αυτή πρέπει να ανανεωθεί αφού πιθανόν έχει αλλάξει. Αποτυχία στην εντολή πρέπει να σημαίνει ότι δεν έχει αλλάξει κάτι από τα δεδομένα της υπηρεσίας, επομένως απλά πρέπει να καθαριστούν οι παράμετροι της διαχειριστικής εντολής στη άση από τις τιθείσες τιμές.

3.1.3 Εξυπηρετητής πελατών

Ο Client Agent αντιμετωπίζει την ανάγκη για πολλών ειδών πελάτες, είτε ειδικά κατσκευασμένους για το εν λόγω σύστημα, ή ήδη υπαρχόντων, όπως είναι οι πελάτες HTTP. Χρησιμοποίηση ήδη υπάρχοντων πελατών έχει το πλεονέκτημα ότι μειώνει τον χρόνο και την πολυπλοκότητα εγκατάστασης του συστήματος, αυξάνοντας ταυτόχρονα την ευκολία αποδοχής του εφόσον για τη διαχείριση ενός συστήματος χρειάζεται εγκατάσταση λογισμικού μόνο σε ένα σύστημα.

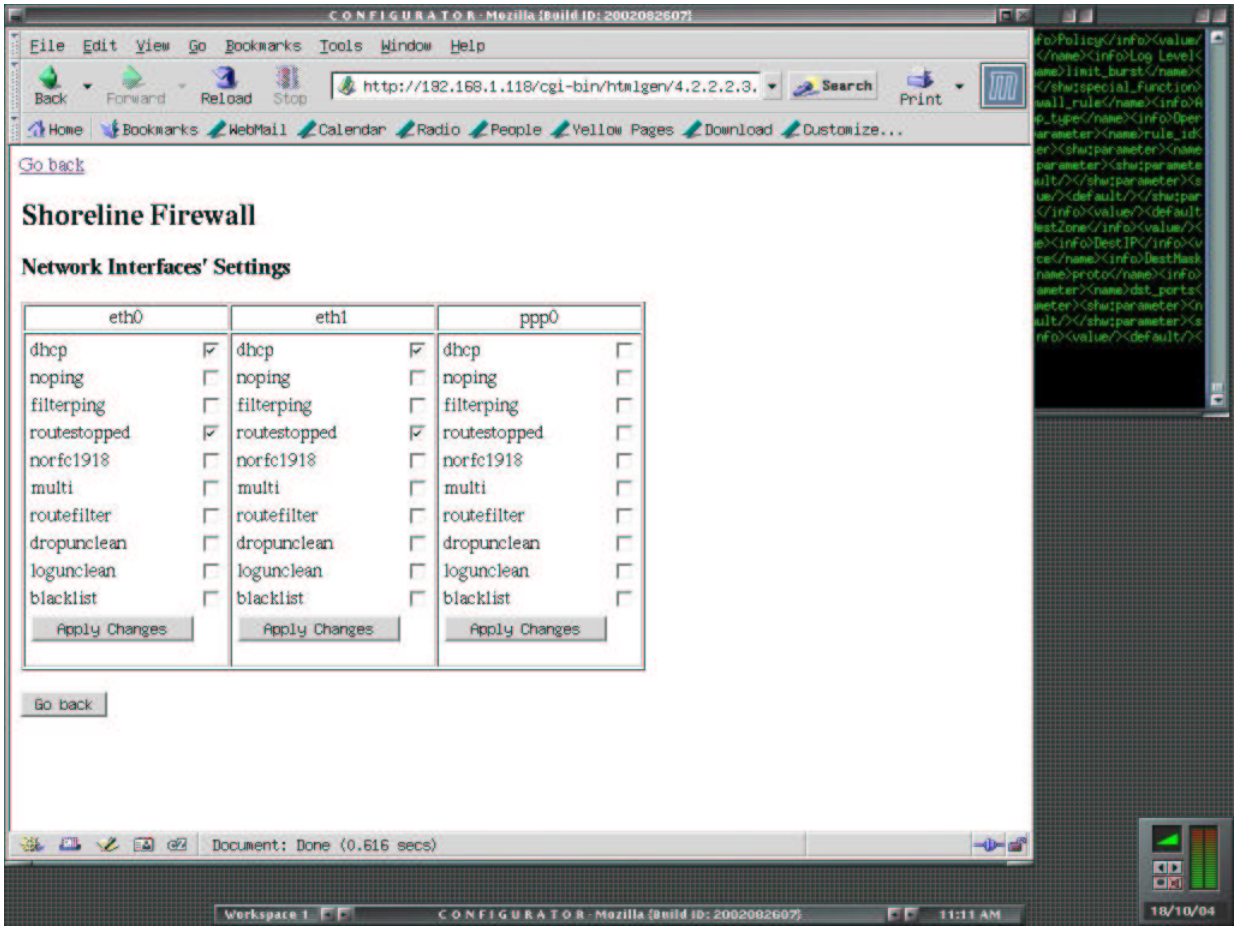
Για το συγκέκριμενο σύστημα υλοποιήθηκε μόνο ένας Client Agent για HTTP clients , όπως είναι οι Web browsers. Ο HTTP Client Agentαποτελείται από έναν εξυπηρετητή HTTP, μια οντότητα που ονομάζεται HTMLGenerator που τρέχει στον εξυπηρετητή χρησιμοποιώντας το πλέον γνωστό CGI interface και κάποια αρχεία XSL που αναλαμβάνουν την διαμόρφωση των XMLδεδομένων σε μορφή HTML. Η διαμόρφωση γίνεται στο μέρος του πελάτη εφόσον αυτός την υποστηρίζει. Ως HTTP εξηπηρετητής χρησιμοποιήθηκε ο ευρέως διαδεδομένος Apache Web Serverπου έχει δημιουργηθεί από το γνωστό για τη δραστηριότητα του στο χώρο του ελεύθερου κώδικα Apache Software Foundation. Επίσης ως εναλλακτική λύση χρησιμοποιήθηκε ο thttpd web server με κύρια πλεονεκτήματα το μικρό μέγεθος και την ταχύτητα του, που ταίριαζε καλύτερα στο υπό μελέτη σύστημα. Ο HTMLGenerator υλοποιήθηκε με τη οήθεια μιας μικρής ιβλιοθήκης λογισμικού για CGI την cgihtml. Ο HTMLGeneratorεκτελείται στο περιεχόμενο μιας αίτησης στον http server με παράμετρο την εντολή που πρέπει να σταλεί στον Core Agent σε μορφή μιας HTTP POST request . Η φόρμα της αίτησης αναλύεται και σχηματίζεται η αίτηση που πρέπει να σταλεί στονCore Agent. Αφού σταλεί χρησιμοποιώντας το πρωτόκολλο που έχει ήδη αναφερθεί, ο HTMLGenerator περιμένει μια απάντηση από τον Core Agent. Η απάντηση είναι σε μορφή XMLπεριέχοντας σαν Processing Instruction το XSL αρχείο που πρέπει να χρησιμοποιηθεί για το μετασχηματισμό σε HTML.

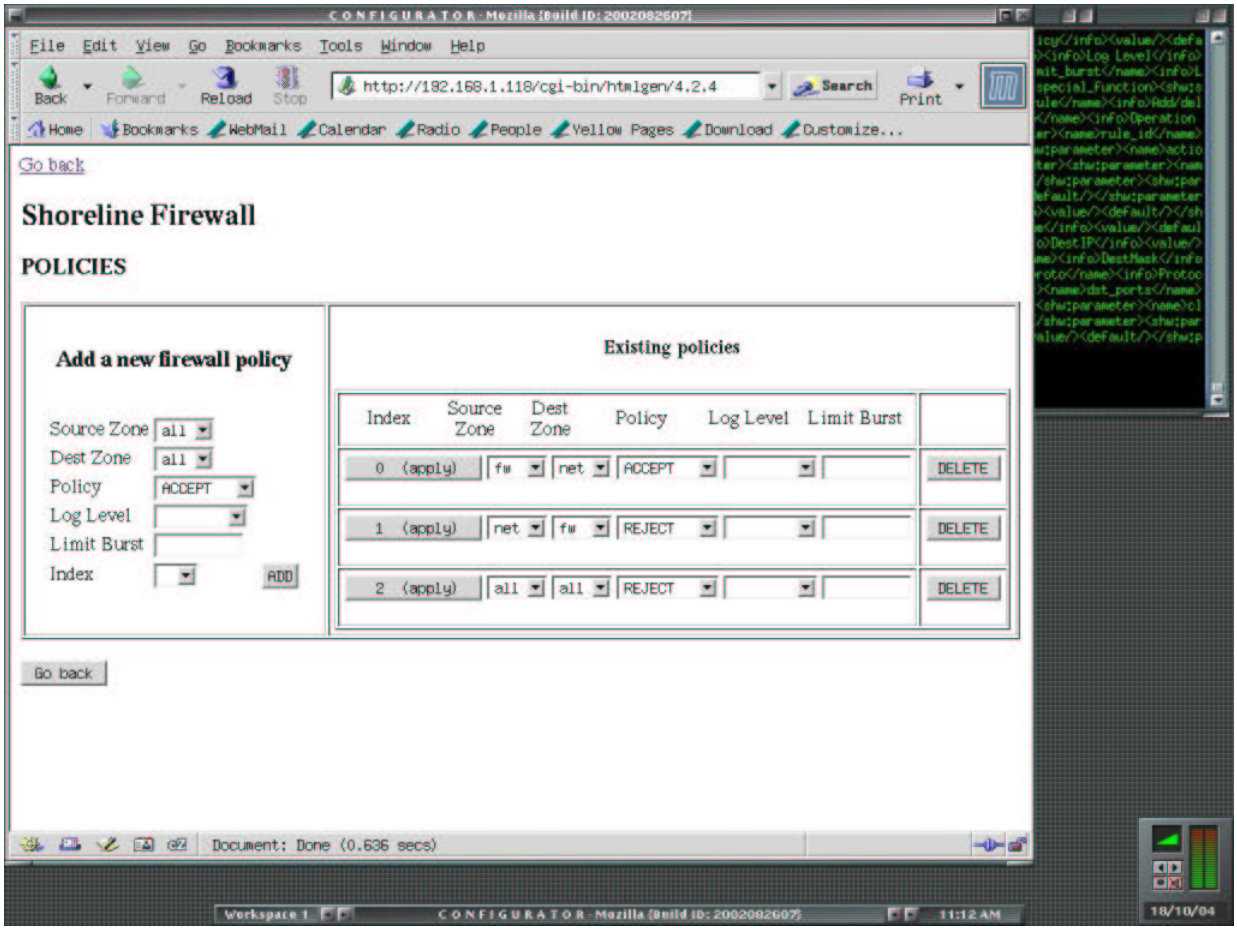
Η διαμόρφωση των δεδομένων σε HTMLγίνεται με τη οήθεια XSLαρχείων που είτε το σύστημα παρέχει για κάθε Client Agent , είτε κάθε υπηρεσία περιγράφει στον ορισμό της. Για να είναι δηλαδή δυνατόν μια υπηρεσία να είναι τελικά διαχειρίσιμη από ένα πελάτη που χρησιμοποιεί έναν συγκεκριμένο Client Agent θα πρέπει να προσφέρει τα αντίστοιχα XSL αρχεία.

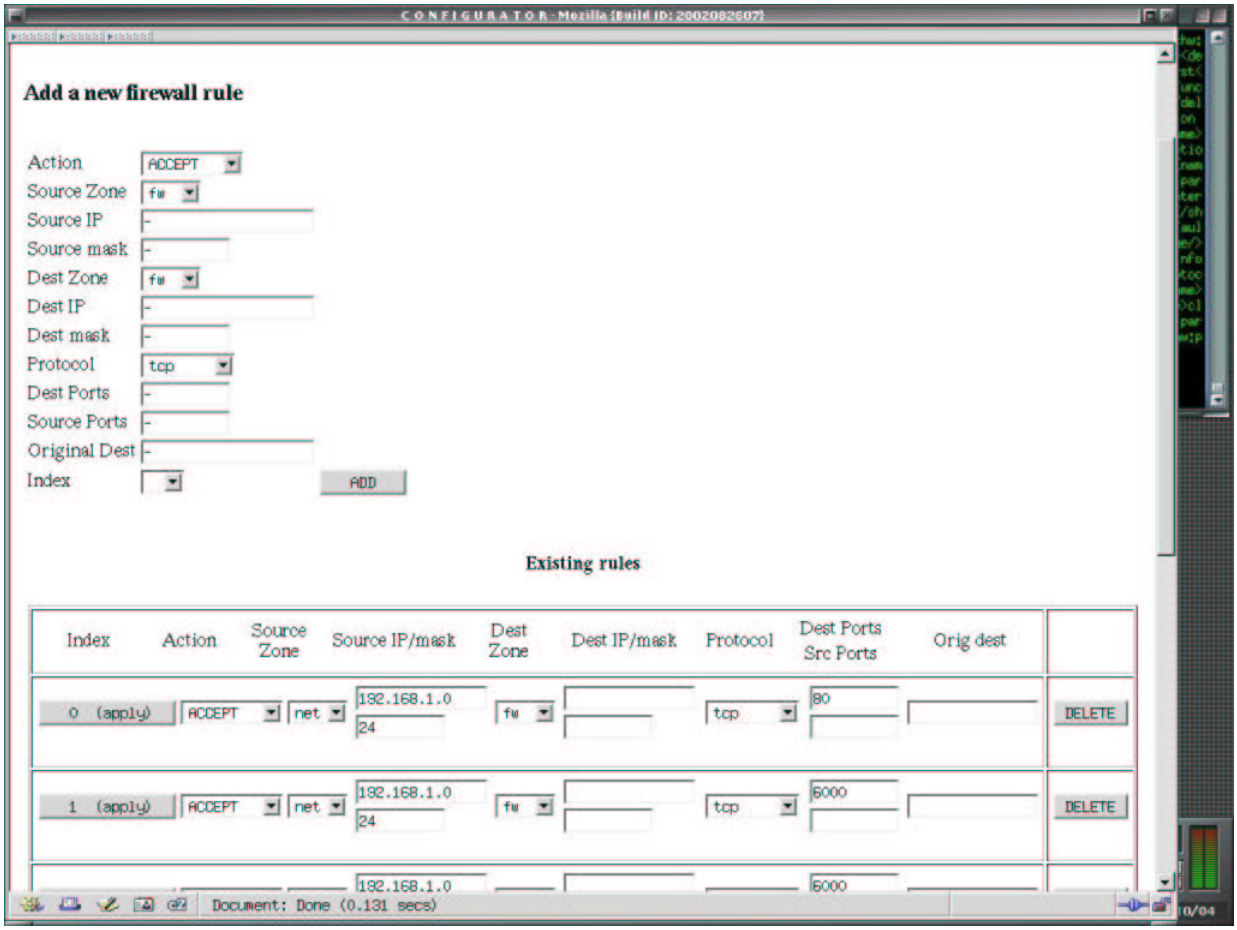
3.1.4 Πελάτης

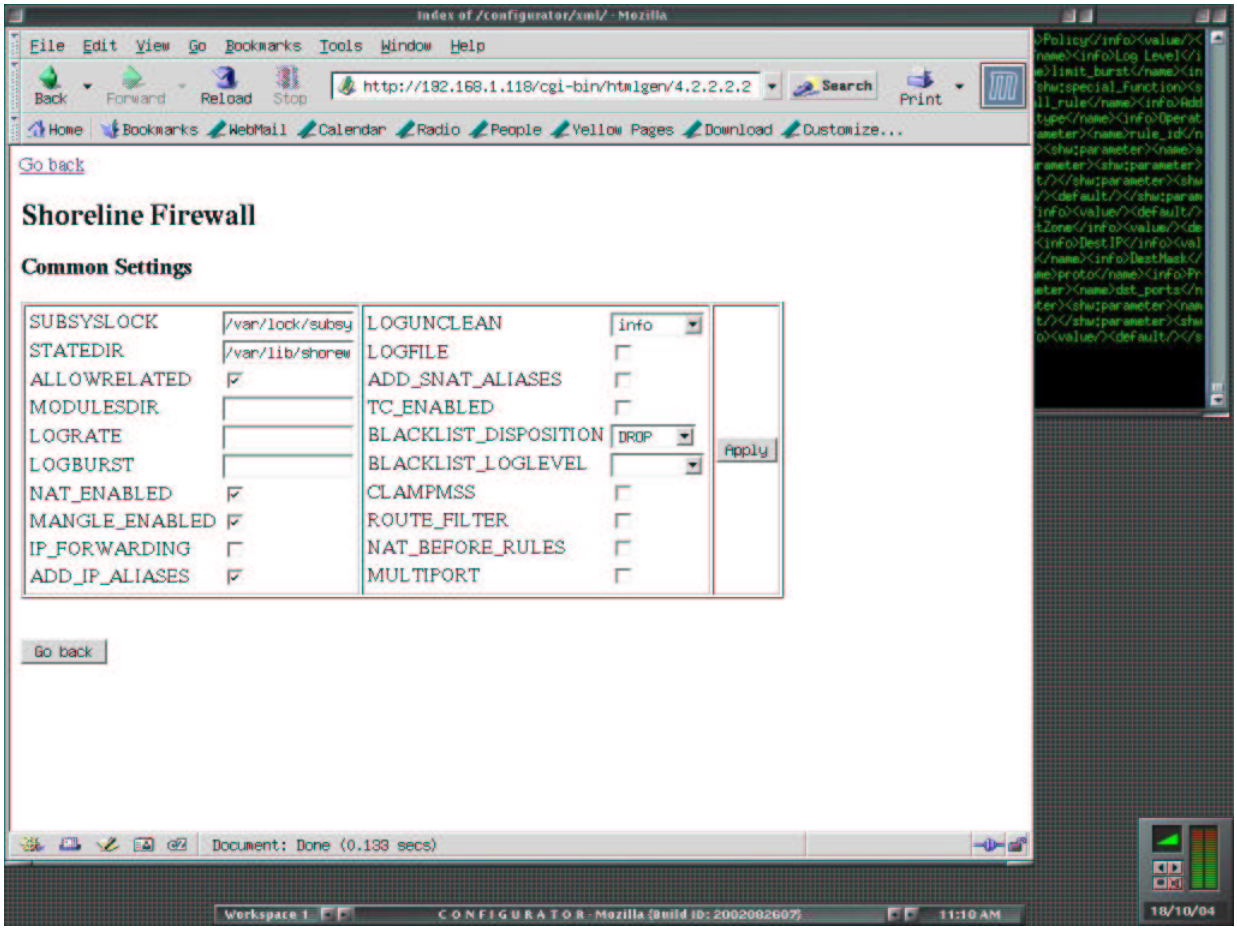
Ο Client υλοποιεί την διεπαφή μεταξύ του τελικού χρήστη και του συστήματος. Στην συγκεκριμένη φάση ανάπτυξης του συστήματος χρησιμοποιήθηκε σαν Client ένας web browser που υποστηρίζει διαμόρφωση XML δεδομένων σε HTML μέσω XSL , ο Mozilla. Ο Mozilla μπορεί να δεχτεί σαν είσοδο ένα XML αρχείο με ορισμένο το XSL αρχείο που μπορεί να το διαμορφώσει σε HTML και να εκτελέσει τη διαμόρφωση. Το XML αρχείο προσφέρεται από τον HTML Generator σε μορφή raw data .

Σχήμα 3.1.3: Παράδειγμα διεπαφής HTML πελάτη


CONFIGURATOR - Mozilla {Build ID: 2002082607}
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://192.168.1.118/cgi-bin/htmlgen/4.2.2.2.3.
Search Print
Home Bookmarks WebMail Calendar Radio People Yellow Pages Download Customize...
Go back
Shoreline Firewall
Network Interfaces' Settings
eth0 eth1 ppp0
dhcp noping filterping routestopped norfc1918 multi routefilter dropunclean logunclean blacklist
Apply Changes
Apply Changes
Apply Changes
Go back
Document: Done (0.616 secs)
Workspace 1
CONFIGURATOR - Mozilla {Build ID: 2002082607}
11:11 AM
18/10/04


CONFIGURATOR - Mozilla {Build ID: 2002082607}
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://192.168.1.118/cgi-bin/htmlgen/4.2.4
Search Print
Home Bookmarks WebMail Calendar Radio People Yellow Pages Download Customize...
Go back
Shoreline Firewall
POLICIES
Add a new firewall policy
Source Zone all
Dest Zone all
Policy ACCEPT
Log Level
Limit Burst
Index
ADD
Existing policies
Index Source Zone Dest Zone Policy Log Level Limit Burst
0 (apply) fw net ACCEPT DELETE
1 (apply) net fw REJECT DELETE
2 (apply) all all REJECT DELETE
Go back
Document: Done (0.636 secs)
Workspace 1
CONFIGURATOR - Mozilla {Build ID: 2002082607}
11:12 AM
18/10/04


CONFIGURATOR - Mozilla {Build ID: 2002082607}
Add a new firewall rule
Action ACCEPT
Source Zone fw
Source IP -
Source mask -
Dest Zone fw
Dest IP -
Dest mask -
Protocol tcp
Dest Ports -
Source Ports -
Original Dest -
Index
ADD
Existing rules
Index
Action
Source Zone
Source IP/mask
Dest Zone
Dest IP/mask
Protocol
Dest Ports
Src Ports
Orig dest
0 (apply) ACCEPT net 192.168.1.0 24 fw tcp 80 DELETE
1 (apply) ACCEPT net 192.168.1.0 24 fw tcp 6000 DELETE
192.168.1.0 6000
Document: Done (0.131 secs)


Index of /configurator/xml/ - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://192.168.1.118/cgi-bin/htmlgen/4.2.2.2.2
Search
Print
Home Bookmarks WebMail Calendar Radio People Yellow Pages Download Customize...
Go back
Shoreline Firewall
Common Settings
SUBSYSLOCK /var/lock/subsy
STATEDIR /var/lib/shorew
ALLOWRELATED
MODULESDIR
LOGRATE
LOGBURST
NAT_ENABLED
MANGLE_ENABLED
IP_FORWARDING
ADD_IP_ALIASES
LOGUNCLEAN info
LOGFILE
ADD_SNAT_ALIASES
TC_ENABLED
BLACKLIST_DISPOSITION DROP
BLACKLIST_LOGLEVEL
CLAMPMSS
ROUTE_FILTER
NAT_BEFORE_RULES
MULTIPORT
Apply
Go back
Document: Done (0.133 secs)
Workspace 1
CONFIGURATOR - Mozilla {Build ID: 2002082607}
11:10 AM
18/10/04

3.2 Υπηρεσίες

Σημαντικό κομμάτι του συστήματος είναι τέλος η υπηρεσία που αποτελεί αντικείμενο διαχείρισης. Με τον ίδιο τρόπο που περιγράφεται παρακάτω, οποιαδήποτε υπηρεσία μπορεί να διαχειριστεί εκτελώντας συγκεκριμένες ενέργειες σε ένα UNIX λειτουργικό και αλλάζοντας συγκεκριμένες παραμέτρους της, μπορεί να γίνει μέρος του υπό μελέτη συστήματος διαχείρισης.

3.2.1 Δημιουργία διαχειριστικής οντότητας υπηρεσίας

Η υπηρεσία πρέπει καταρχήν να υποδηλώσει την παρουσία της στο σύστημα με ένα XML configurationαρχείο. Η μορφή του αρχείου αυτού ορίζεται από ένα DTD αρχείο το cfg.dtd . Το αρχείο αυτό καθορίζει ότι κάθε υπηρεσία πρέπει να δηλώσει ένα όνομα και μια περιγραφή για τον εαυτό της, τα χαρακτηριστικά των XML αρχείων της, τα εξωτερικά προγράμματα που χρησιμοποιεί και το σύνολο των οντοτήτων από τις οποίες αποτελείται.

Τα XML χαρακτηριστικά των αρχείων της υπηρεσίας συνοψίζονται στα :
* xml_path: Το μέρος της κεντρικής άσης στο οποίο θα τοποθετηθεί η υπηρεσία.
* xsl_path: Το μέρος της κεντρικής άσης που θα τοποθετηθούν τα XSL αρχεία της υπηρεσίας.
* dtd_path Το μέρος της κεντρικής άσης που θα τοποθετηθούν οι ορισμοί των XML αρχείων της υπηρεσίας σε μορφή DTD αρχείων.
* ns_name: Το χαρακτηριστικό όνομα όλων των στοιχείων της υπηρεσίας ώστε να αποφευχθούν δηλωσεις ίδιων στοιχείων σε διαφορετικές υπηρεσίες.
* sections_no: Ο αριθμός των ενοτήτων στις οποίες χωρίζεται μια υπηρεσία.

Τα εξωτερικά προγράμματα που μια υπηρεσία μπορεί να χρησιμοποιήσει χωρίζονται στα :

* format_bin : Τα προγράμματα που χρησιμοποιούνται για το μετασχηματισμό από XML στη μορφή του συστήματος και αντίστροφα και ονομάζονται native2xml, xml2native

* config_bin: Τα προγράμματα που χρησιμοποιούνται για τις διαχειριστικές ενέργειες, όπως το κεντρικό πρόγραμμα επανεκκίνησης service_script και τα ειδικά προγράμματα special_functions . Τα special_functions ορίζονται με ένα όνομα, το μέρος της υπηρεσίας που εφαρμόζονται και των αριθμό των παραμέτρων που δέχονται.

Η υπηρεσία μπορεί επίσης να χωριστεί σε ξεχωριστές οντότητες με ξεχωριστά προγράμματα ως format_bin και ξεχωριστό service_script .

Ο ορισμός του configuration αρχείου που πρέπει να παρέχει κάθε υπηεσία ακολουθεί.

<!– Service configuration document definition –>
<!ELEMENT service (name, desc, xml, binaries, sections?)>
<!– Name contains the name of the service's xml file root element. –>
<!ELEMENT name (#PCDATA)>
<!– A description for the service –>
<!ELEMENT desc (#PCDATA)>
<!– XML related info –>
<!ELEMENT xml (xml_path , xsl_path , ns_name , ns_uri , dtd_path , dtd_mount , xml_mount , sections_no?)>
<!– Service's binaries related info –>
<!ELEMENT binaries (format_bin, config_bin)>
<!– Service's sections' (if any) –>
<!ELEMENT sections (section*)>

<!– Service's xml file name –>
<!ELEMENT xml_path (#PCDATA)>
<!– Service's xsl file name –>
<!ELEMENT xsl_path (#PCDATA)>
<!– Service's namespace alias –>
<!ELEMENT ns_name (#PCDATA)>
<!– Service's namespace alias –>
<!ELEMENT ns_uri (#PCDATA)>
<!– Service's dtd file name –>
<!ELEMENT dtd_path (#PCDATA)>
<!– Service's dtd and xml mount points. Where in the main dtd and xml files the service's entity shall be placed. (under which child node, e.g. services) –>
<!ELEMENT dtd_mount (#PCDATA)>
<!ELEMENT xml_mount (#PCDATA)>
<!– Number of sections consisting the service. A service can be divided in sections in cases where many configuration files are present. A section has its' own transformation and service_script scripts to be used instead of service's when a change related to the specific section is detected –>
<!ELEMENT sections_no (#PCDATA)>
<!– Service's transformation scripts –>
<!ELEMENT format_bin (native2xml, xml2native)>

<!ELEMENT config_bin (service_script?, special_function*)>

<!– Transformation scripts' paths (relative to $CONFIGURATOR_ROOT)–>
<!ELEMENT native2xml (#PCDATA)>
<!ELEMENT xml2native (#PCDATA)>

<!– Service's generic start-stop script –>
<!ELEMENT service_script (#PCDATA)>
<!– Service's special function. Special functions implement less static actions than changing a file by running a script; path is the relative to $CONFIGURATOR_ROOT path of the script and args_no the number of arguments that shall pass to the script. –>
<!ELEMENT special_function (name, path, args_no)>
<!ATTLIST special_function no CDATA #REQUIRED>
<!ELEMENT path (#PCDATA)>
<!ELEMENT args_no (#PCDATA)>

<!– A section has its' own start-stop and transformation scripts. –>
<!ELEMENT section (sname, sreload, sn2x, sx2n)>
<!ELEMENT sname (#PCDATA)>
<!ELEMENT sreload (#PCDATA)>
<!ELEMENT sn2x (#PCDATA)>
<!ELEMENT sx2n (#PCDATA)>

3.2.2 Υπάρχουσες υπηρεσίες υπό διαχείριση

Στο υπό μελέτη σύστημα ενσωματώθηκαν πολλές υπηρεσίες με τον τρόπο που μόλις αναφέρθηκε:
- dhcpd (για την δυναμική ανάθεση IP διευθύνσεων στο εσωτερικό δίκτυο)
- NetConf (για την διαχείριση των δικτυακών διεπαφών)
- shorewall (για την ασφάλεια, δρομολόγηση και επιλεκτική πρόσβαση)
- UserAccount (για την διαχείριση λογαριασμών χρηστών)

Για μια καλύτερη επίδειξη του τρόπου με τον οποίο μια υπηρεσία μπορεί να ενσωματωθεί στο σύστημα θα περιγραφεί η διαδικασία ενσωμάτωσης μιας από τις παραπάνω, του shorewall (Shoreline Firewall) .

Η υπηρεσία shorewall αποτελείται από έναν αριθμό διαχειριστικών αρχείων που ορίζουν τα χαρακτηριστικά της υπηρεσίας. Όταν ένα από αυτά τα αρχεία αλλάξει από το διαχειριστή, τα χαρακτηριστικά της υπηρεσίας αλλάζουν και αυτή πρέπει να σταματήσει και να ξεκινήσει ξανά μέσα από ένα εξωτερικό πρόγραμμα, το shorewall . Επομένως ένα το service_script της υ-

πηρεσίας είναι το πρόγραμμα /etc/init.d/shorewall που είναι και το πλήρες path του προγράμματος στο σύστημα αρχείων του λειτουργικού. Το όνομα της υπηρεσίας μπορεί να τεθεί για προφανείς λόγους σε shorewall , ενώ η περιγραφή σε Shoreline Firewall . Τα ονόματα των xml, xsl, dtd αρχείων είναι σύμφωνα με τη σύμβαση αντίστοιχα shorewall.xml, shorewall.xsl, shorewall.dtd ενώ το ns_name μπορεί να τεθεί συντομογρφικά ως shw ώστε τα στοιχεία της υπηρεσίας να ξεχωρίζουν από άλλα διαφορετικών υπηρεσιών. Στην κεντρική άση, η θέση της υπηρεσίας είναι κάτω από την κατηγορία services .

Η xml περιγραφή της υπηρεσίας θα σχηματιστεί από ένα μεγάλο αριθμό αρχείων και πρέπει να ληθφεί υπόψην ο χρόνος που θα απαιτηθεί για την μετατροπή των αρχείων αυτών σε XML σε περίπτωση κάποιας αλλαγής. Έτσι λοιπόν αποφασίστηκε ο χωρισμός της υπηρεσίας σε τέσσερις ενότητες, με άση τα αρχεία που παρουσιάζουν μεγαλύτερο ενδιαφέρον. Για κάθε ενότητα ορίστηκαν ξεχωριστά format_bin αρχεία και με αυτόν τον τρόπο σε περίπτωση αλλαγών εκτελούνται μόνο οι απολύτως απαραίτητες μετατροπές. Το σύνολο των ενοτήτων είναι τέσσερις και χωρίζονται στις :
- conf
με format_bin τα αρχεία confn2x, confx2n
- zones
με format_bin τα αρχεία zonesn2x, zonesx2n
- policy
με format_bin τα αρχεία policyn2x, policyx2n
- rules
με format_bin τα αρχεία rulesn2x, rulesx2n

Για την μετατροπή του συνόλου της πληροφορίας της υπηρεσίας χρησιμοποιούνται τα κεντρικά scripts n2xscrpt, x2nscript , τα οποία στην πραγματικότητα απλά καλούν τα αντίστοιχα προγράμματα των επιμέρους ενοτήτων και επιπλέον διαχειρίζονται την πληροφορία των special_functions .
Οι special_functions της υπηρεσίας είναι τρεις και αφορούν την πρόσθεση νέων χαρακτηριστικών στο σύστημα.
Έτσι λοιπόν το configuration αρχείο της υπηρεσίας shorewall σχηματίζεται ως εξής :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE service SYSTEM "../dtd/cfg.dtd">
<service>
```

```
<name>shorewall</name>
<desc>Shoreline Firewall configuration.</desc>
<xml>
   <xml_path>shorewall.xml</xml_path>
   <xsl_path>shorewall.xsl</xsl_path>
   <ns_name>shw</ns_name>
   <ns_uri>http://localhost/</ns_uri>
   <dtd_path>shorewall.dtd</dtd_path>
   <dtd_mount>services</dtd_mount>
   <xml_mount>services</xml_mount>
   <sections_no>4</sections_no>
</xml>
<binaries>
   <format_bin>
      <native2xml>scripts/shorewall/n2xscrpt</native2xml>
      <xml2native>scripts/shorewall/x2nscrpt</xml2native>
   </format_bin>
   <config_bin>
      <service_script>/etc/shorewall/shorewall</service_script>
      <special_function no="0">
         <name>shorewall_zone</name>
         <path>scripts/shorewall/admin_zone</path>
         <args_no>8</args_no>
      </special_function>
      <special_function no="1">
         <name>shorewall_policy</name>
         <path>scripts/shorewall/admin_policy</path>
         <args_no>7</args_no>
      </special_function>
      <special_function no="2">
         <name>shorewall_rule</name>
         <path>scripts/shorewall/admin_rule</path>
         <args_no>13</args_no>
      </special_function>
   </config_bin>
</binaries>
<sections>
   <section>
      <sname>conf</sname>
      <sreload>/etc/init.d/shorewall</sreload>
      <sn2x>scripts/shorewall/confn2x</sn2x>
```

```
            <sx2n>scripts/shorewall/confx2n</sx2n>
        </section>
        <section>
            <sname>zones</sname>
            <sreload>/etc/init.d/shorewall</sreload>
            <sn2x>scripts/shorewall/zonesn2x</sn2x>
            <sx2n>scripts/shorewall/zonesx2n</sx2n>
        </section>
        <section>
            <sname>policy</sname>
            <sreload>/etc/init.d/shorewall</sreload>
            <sn2x>scripts/shorewall/policyn2x</sn2x>
            <sx2n>scripts/shorewall/policyx2n</sx2n>
        </section>
        <section>
            <sname>rules</sname>
            <sreload>/etc/init.d/shorewall</sreload>
            <sn2x>scripts/shorewall/rulesn2x</sn2x>
            <sx2n>scripts/shorewall/rulesx2n</sx2n>
        </section>
    </sections>
</service>
```

Για την μετατροπή των δεδομένων της υπηρεσίας από τα αρχεία του συστήματος σε XML δεδομένα δίνεται ένα από τα native2xml scripts των επιμέρους υπηρεσιών, γραμμένο στην scripting language bash που χρησιμοποιείται πολύ στα συστήματα τύπου UNIX . Αυτό το πρόγραμμα επεξεργάζεται ένα ασικό αρχείο της υπηρεσίας, το /etc/shorewall/zones και δημιουργεί ένα XML αρχείο με όλη την απαραίτητη πληροφορία όπως αυτή καθορίστηκε στο DTD αρχείο της υπηρεσίας shorewall.dtd.
Στο DTD αυτό αρχείο είναι προφανής και ο διαχωρισμός των δεδομένων της υπηρεσίας στα δεδομένα (configuration) και στις ειδικές διαχειριστικές εντολές special_function .

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:shw="http://localhost/shw">
<xsl:template match="shw:shorewall" mode="root">
```

```
<xsl:variable name="rootlink">
<xsl:value-of select="$base"/>
<xsl:value-of select="@elink"/>
</xsl:variable>
<xsl:variable name="newrootlink">
<xsl:value-of select="$base"/>
<xsl:value-of select="$blockpath"/>
<xsl:value-of select="'.'"/>
<xsl:value-of select="$relpath"/>
</xsl:variable>
<xsl:variable name="baselink">
<xsl:value-of select="$base"/>
<xsl:value-of select="$blockpath"/>
<xsl:value-of select="'.'"/>
</xsl:variable>
<h2>Shoreline Firewall</h2>
<xsl:choose>
<xsl:when test="string-length($relpath)=0">
<br/>
<a>
<xsl:attribute name="href">
<xsl:value-of select="$rootlink"/>
<xsl:value-of select="'2.2.2'"/>
</xsl:attribute>
<h3>COMMON SETTINGS</h3>
```

Κεφάλαιο 4

Σχεδιασμός και Υλοποίηση

Στο κεφάλαιο αυτό ακολουθεί μια σύντομη περιγραφή των κλάσεων που υλοποιούν τις οντότητες του συστήματος διαχείρισης στη γλώσσα C++ και του τρόπου λειτουργίας τους.

4.1 Πυρήνας Συστήματος Διαχείρισης

Ο Core Configurator υλοποιήθηκε με τη δημιουργία των κλάσεων
- ConfBase
- ConfigHandlers
- ConfService
- ConfXMLParser
- MainXMLHandlers
- MainXSLHandlers
- SAXCheckHandlers
- SAXCheck
- ServiceMgr
- Service
- SocketInputSource
- SpecialFunction

Η κλάση ConfBase αποτελεί τον πυρήνα του Core Configurator και περιέχει λειτουργίες για την αρχικοποίηση και τερματισμό του περιβάλλοντος του πυρήνα όπως και τον ασικό αλγόριθμο εξυπηρέτησης αιτήσεων του κεντρικού πράκτορα αλλά και διαχείρισης των υπηρεσιών. Ακολουθεί ο ορισμός της με μια σύντομη περιγραφή.

```
#include "ServiceMgr.hpp"

using namespace std;

/**
  *  This class is the heart of the configurator.
  * It handles configuration requests from the agent
  * by applying them to the system.
  * It uses a ServiceMgr to handle services and
  * and a ConfService to communicate with the agent.
  * The ConfXMLParser is used to to parse the request
  * sent from agent (in xml form) identify the changes
  * and apply them via ServiceMgr to the system.
  */
class ConfBase
{
public:
/**
  * Used to identify services for configuration,
  * keep them in a list and apply any changes to
  * them when needed.
  */
ServiceMgr * servMgr;
/**
  * Used to communicate with the agent
  */
ConfService * confService;
/**
  * Used to parse agent's requests and identify
  * configuration changes in services.
  */
ConfXMLParser * confParser;

/** @name Constructor and destructor */
```

```
//@{
/**
  * Default constructor
  */
ConfBase() ;
/**
  * Default destructor
  */
~ConfBase();
//@}

/** @name methods */
//@{
/**
  * Initialize the confParser and find existing
  * configurable services via ServiceMgr
  */
void initialize();
/**
  * Get a request from agent in the form of an
  * xml file, parsing it and identifying changes.
  */
void getAgentRequest();
/**
  * Apply the requested changes to the system via
  * ServiceMgr and get results.
  */
void processRequest();
/**
  * Notify agent about configuration processing results
  * and close this configuration session.
  */
void notifyAgent();
/**
  * Terminate the configurator
  */
void terminate();
//@}
};
```

Η κλάση ConfigHandlers υλοποιεί την επεξεργασία των ορισμών των υπηρεσιών μέσα από τα XML configuration αρχεία τους. Η πληροφορία που αποκτάται με τις λειτουργίες που προσφέρει αυτή η κλάση μπορεί να χρησιμοποιηθεί για την δημιουργία των υπηρεσιών. Ακολουθεί ο ορισμός της με μια σύντομη περιγραφή.

```
#include <sax/HandlerBase.hpp>
#include <framework/XMLFormatter.hpp>
#include "Service.hpp"

/**
  * This class implements the Document Handler needed
  * from the ServiceMgr to parse services' configuration
  * files and create Services' list. During parsing must
  * get all info from a file and store it to local
  * variables for use  by the ServiceMgr.
  */
class ConfigHandlers : public HandlerBase, public XMLFormatTarget
{
public:
    /** @name Constructor and destructor */
    //@{
    /**
     * Default constructor
     */
    ConfigHandlers(const char* const encodingName,
   const XMLFormatter::UnRepFlags unRepFlags) ;
    /**
      * Default destructor
      */
    ~ConfigHandlers() ;
    //@}

    /** @name Implementation of the format target interface */
    //@{
    void writeChars(const XMLByte* const toWrite) ;
    void writeChars(const XMLByte* const toWrite,
                    const unsigned int count,
                    XMLFormatter* const   formatter) ;
    //@}
```

```
/** @name Implementation of the SAX DocumentHandler interface
//@{
void endDocument() ;
void endElement(const XMLCh* const name) ;
void characters(const XMLCh* const chars, const unsigned int l
void ignorableWhitespace(const XMLCh* const chars,
                         const unsigned int length) ;
void processingInstruction(const XMLCh* const target,
                           const XMLCh* const    data) ;
void startDocument() ;
void startElement(const XMLCh* const name, AttributeList& attr
//@}
/** @name Implementation of the SAX ErrorHandler interface */
//@{
void warning(const SAXParseException& exception) ;
void error(const SAXParseException& exception) ;
void fatalError(const SAXParseException& exception) ;
//@}
/** @name Implementation of the SAX DTDHandler interface */
//@{
void notationDecl(const XMLCh* const name,
                  const XMLCh* const publicId,
                  const XMLCh* const systemId) ;
void unparsedEntityDecl(const XMLCh* const name,
                        const XMLCh* const publicId,
                        const XMLCh* const systemId,
                        const XMLCh* const notationName) ;
//@}

Service *service;
SpecialFunction *specialFunction;

int nameFlag;
int descFlag;
int xmlFFlag;
int xmlPathFlag;
int xslPathFlag;
int namespaceFlag;
int nsuriFlag;
int dtdPathFlag;
int xmlMountFlag;
```

```
    int dtdMountFlag;
    int sectionsNoFlag;
    int sections_no;
    int sectionsFlag;
    int sectionFlag;
    int sectionsCounter;
    int snameFlag;
    int sreloadFlag;
    int sn2xFlag;
    int sx2nFlag;
int binsFlag;
    int formatBinFlag;
    int nat2xmlFlag;
    int xml2natFlag;
    int configBinFlag;
    int servFlag;
    int sfFlag;
    int sfPathFlag;
    int sfNameFlag;
    int sfArgsFlag;

private:
    /**
      * This is the formatter object that is used to output
      * the data to the target. It is set up to format to the
      * standard output stream.
      */
    XMLFormatter fFormatter;
};
```

Η κλάση ConfService υλοποιεί τις απαραίτητες λειτουργίες για την επικοινωνία με τον Core Agent . Προσφέρει στον πυρήνα τη δυνατότητα να λαμβάνει αιτήσεις σε μορφή XML χρησιμοποιώντας μια υποκλάση της InputSource . Ακολουθεί ο ορισμός της με μια σύντομη περιγραφή.

```
#include "SocketInputSource.hpp"

/**
  * This class is responsible for the communication
  * with the agent.
  * It provides the configurator the capability to
  * receive an agents's request in xml form via an
  * InputSource derivative.
  */
class ConfService
{
char *response;

public:
/** Provides the configurator access to xml data from
  * the agent via a socket.
  */
SocketInputSource * xmlSocket;
/** @name Constructor and destructor */
//@{
/**
  * Default constructor
  */
ConfService();
/**
  * Default destructor
  */
~ConfService();
//@}

/** @name Configurator-agent communication methods */
//@{
/**
  * Listen for connections from agent.
  * When a connection is requested prepare the handling
  * of xml data via the SocketInputSource instance.
```

```
  */
int startServAgent();
/**
  * Set the response string to be sent to agent.
  */
void setResponse(char *response);
/**
  * Inform the agent about the result of a configuration
  * request made.
  */
int notifyAgent();
/**
  * Stop serving an agent’s request.
  */
int stopServAgent();
//@}
};
```

Η κλάση ConfXMLParser υλοποιεί την λειτουργία επεξεργασίας αιτήσεων από τον Core Agent , ανίχνευσης αλλαγών και αποθήκευσής τους για περαιτέρω επεξεργασία. Η ανίχνευση αλλαγών αποτελείται από τις επιμέρους λειτουργίες της ανίχνευσης της υπηρεσίας την οποία αφορά η αλλαγή και αν η αίτηση αφορά ειδική διαχειριστική εντολή ή αλλαγή στα χαρακτηριστικά της υπηρεσίας. Στην πρώτη περίπτωση αποθηκεύεται το XML κομμάτι της υπηρεσίας που άλλαξε, ενώ στη δεύτερη μόνο οι παράμετροι της ειδικής εντολής. Ακολουθεί ο ορισμός της κλάσης με μια σύντομη περιγραφή.

```
#include <parsers/SAXParser.hpp>
#include <util/PlatformUtils.hpp>
#include <util/TransService.hpp>
#include "SAXCheck.hpp"
#include "SAXCheckHandlers.hpp"
#include "ConfService.hpp"

using namespace std;

/**
  * This class implements the configurator's ability
  * to parse agent's requests, identify changes and
  * save them for further processing. Identify changes
  * means:
  *   (i) the service where changes are requested
  *   (ii) if changes are static or dynamic, that is
  *     if they involve special functions
  * (iii) if static save the xml portion of the service
  *   changed to a buffer
  *  (iv) if dynamic save the parameters of the special
  *   function
  */
class ConfXMLParser
{
public:
/** SAX parser used to parse requests */
SAXParser * parser;
/** Document handler for the parser to identify changes */
SAXCheckHandlers * handler;
/** Error counter */
int errorCount;
/** @name Constructor and destructor */
```

```
//@{
/**
  * Constructor
  */
ConfXMLParser();
/**
  * Destructor
  */
~ConfXMLParser();
//@}

/** @name Requests' handling methods */
//@{
/**
  * Initialize handling utilities
  */
void init();
/**
  * Parse a request and save changes.
  */
void parseStart(ConfService &);
/**
  * Stop handling a request.
  */
void parseStop();
/**
  * Terminate handling utilities
  */
void terminate();
//@}
};
```

Η κλάσεις MainXmlHandlers και MainXslHandlers χρησιμοποιούνται για την αρχικοποίηση της άσης του συστήματος τόσο όσον αφορά τις προς διαχείριση υπηρεσίες όσο και την διαμόρφωση των δεδομένων από τους Client Agents . Ακολουθεί ο ορισμός τους με μια σύντομη περιγραφή.

```
#include <sax/HandlerBase.hpp>
#include <framework/XMLFormatter.hpp>
#include "Service.hpp"
/**
  * This class implements the Document Handler needed
  * from the ServiceMgr to parse services' configuration
  * files and create Services' list. During parsing must
  * get all info from a file and store it to local
  * variables for use  by the ServiceMgr.
  */
class MainXmlHandlers : public HandlerBase, public XMLFormatTarget
{
public:
   /** @name Constructor and destructor */
   //@{
   /**
     * Default constructor
     */
    MainXmlHandlers(const char* const encodingName,
                    const XMLFormatter::UnRepFlags unRepFlags,
                    Service *services, int no_of_services) ;
    /**
      * Default destructor
      */
    ~MainXmlHandlers() ;
    //@}
    /** @name Implementation of the format target interface */
    //@{
    void writeChars(const XMLByte* const toWrite) ;
    void writeChars(const XMLByte* const toWrite,
                    const unsigned int count,
                    XMLFormatter* const   formatter) ;
    //@}
    /** @name Implementation of the SAX DocumentHandler interface
    //@{
    void endDocument() ;
```

```
    void endElement(const XMLCh* const name) ;
    void characters(const XMLCh* const chars, const unsigned int length)
    void ignorableWhitespace(const XMLCh* const chars,
                             const unsigned int length) ;
    void processingInstruction(const XMLCh* const target,
                               const XMLCh* const    data) ;
    void startDocument() ;
    void startElement(const XMLCh* const name, AttributeList& attributes
    //@}
    /** @name Implementation of the SAX ErrorHandler interface */
    //@{
    void warning(const SAXParseException& exception) ;
    void error(const SAXParseException& exception) ;
    void fatalError(const SAXParseException& exception) ;
    //@}
    /** @name Implementation of the SAX DTDHandler interface */
    //@{
    void notationDecl(const XMLCh* const name,
                      const XMLCh* const publicId,
                      const XMLCh* const systemId) ;
    void unparsedEntityDecl(const XMLCh* const name,
                            const XMLCh* const publicId,
                            const XMLCh* const systemId,
                            const XMLCh* const notationName) ;
    //@}
    Service *services;
    int filedes;
    char *currentMount;
    int mountFlag;
    int len;

private:
    /**
      * This is the formatter object that is used to output
      * the data to the target. It is set up to format to the
      * standard output stream.
      */
    XMLFormatter fFormatter;
    char *base_dtd;
    char *main_xsl;
    char *getHostName();
```

```
};

#include <sax/HandlerBase.hpp>
#include <framework/XMLFormatter.hpp>
#include "Service.hpp"
using namespace std;
/**
  * This class implements the Document Handler needed
  * from the ServiceMgr to parse services' configuration
  * files and create Services' list. During parsing must
  * get all info from a file and store it to local
  * variables for use  by the ServiceMgr.
  */
class MainXslHandlers : public HandlerBase, public XMLFormatTarget
{
public:
    /** @name Constructor and destructor */
    //@{
    /**
      * Default constructor
      */
    MainXslHandlers(const char* const encodingName,
                    const XMLFormatter::UnRepFlags unRepFlags,
                    Service *services, int len) ;
    /**
      * Default destructor
      */
    ~MainXslHandlers() ;
     //@}
     /** @name Implementation of the format target interface */
     //@{
    void writeChars(const XMLByte* const toWrite) ;
    void writeChars(const XMLByte* const toWrite,
                    const unsigned int count,
                    XMLFormatter* const   formatter) ;
    //@}
    /** @name Implementation of the SAX DocumentHandler interface
    //@{
    void endDocument() ;
    void endElement(const XMLCh* const name) ;
    void characters(const XMLCh* const chars, const unsigned int l
```

```
    void ignorableWhitespace(const XMLCh* const chars,
                             const unsigned int length) ;
    void processingInstruction(const XMLCh* const target,
                               const XMLCh* const    data) ;
    void startDocument() ;
    void startElement(const XMLCh* const name, AttributeList& attributes
    //@}
    /** @name Implementation of the SAX ErrorHandler interface */
    //@{
    void warning(const SAXParseException& exception) ;
    void error(const SAXParseException& exception) ;
    void fatalError(const SAXParseException& exception) ;
    //@}
    /** @name Implementation of the SAX DTDHandler interface */
    //@{
    void notationDecl(const XMLCh* const name,
                      const XMLCh* const publicId,
                      const XMLCh* const systemId) ;
    void unparsedEntityDecl(const XMLCh* const name,
                            const XMLCh* const publicId,
                            const XMLCh* const systemId,
                            const XMLCh* const notationName) ;
    //@}
    Service *services;
    int filedes;
    char *currentMount;
    int mountFlag;
    int len;

private:
    /**
      * This is the formatter object that is used to output
      * the data to the target. It is set up to format to the
      * standard output stream.
      */
     XMLFormatter fFormatter;
};
```

Οι κλάσεις SAXCheck και SAXCheckHandlers χρησιμοποιούνται από τον ConfXMLParser για τις λειτουργίες τις σχετικές με της επεξεργασία των εντολών από τον Agent .

```
#include <iostream>
#include "util/XMLString.hpp"

using namespace std;

/**
  * This is a simple class that lets us do easy transcoding
  * of XMLCh data to local code page for display.
  */
class StrX
{
public :
/** @name Constructor and destructor */
//@{
/**
  * Constructor
  */
StrX(const XMLCh* const toTranscode)
{
fLocalForm = XMLString::transcode(toTranscode);
}
/**
  * Destructor
  */
~StrX()
{
delete [] fLocalForm;
}
//@}

/**
  * Transcoding method
  */
const char* localForm() const
{
return fLocalForm;
}
```

```
private :
char* fLocalForm;

};


inline ostream&
operator<<(ostream& target, const StrX& toDump)
{
    target << toDump.localForm();
    return target;
}


#include <sstream>
#include    <sax/HandlerBase.hpp>
#include    <framework/XMLFormatter.hpp>

using namespace std;
/**
  * This class implements the Document Handler needed
  * from the configurator for handling agent’s requests.
  * It provides the ability to save changes in a specific
  * configurable service and points the way to be applied.
  */
class SAXCheckHandlers : public HandlerBase, public XMLFormatTarget
{
public:
/** @name Constructor and destructor */
//@{
/**
  * Constructor
  */
SAXCheckHandlers(const char* const encodingName,
 const XMLFormatter::UnRepFlags unRepFlags);
/**
  * Destructor
  */
~SAXCheckHandlers();
//@}
```

```
/** @name Implementation of the format target interface */
//@{
void writeChars(const XMLByte* const toWrite);

void writeChars(const XMLByte* const toWrite,
const unsigned int count,
XMLFormatter* const formatter);
//@}

/** @name Implementation of the SAX DocumentHandler interface */
//@{
void endDocument();

    void endElement(const XMLCh* const name);

    void characters(const XMLCh* const chars, const unsigned int l

    void ignorableWhitespace(const XMLCh* const chars,
 const unsigned int length);

    void processingInstruction(const XMLCh* const target,
   const XMLCh* const data);

    void startDocument();

    void startElement(const XMLCh* const name, AttributeList& attr
//@}


    /** @name Implementation of the SAX ErrorHandler interface */
//@{
    void warning(const SAXParseException& exception);

void error(const SAXParseException& exception);

    void fatalError(const SAXParseException& exception);
//@}

    /** @name Implementation of the SAX DTDHandler interface */
//@{
```

```
void notationDecl(const XMLCh* const name,
  const XMLCh* const publicId,
  const XMLCh* const systemId);

void unparsedEntityDecl(const XMLCh* const name,
const XMLCh* const publicId,
const XMLCh* const systemId,
const XMLCh* const notationName);
//@}
/**
  * Reset session specific variables for the next session
  */
void reset () ;

/** @name public members */
//@{
/** Changed configuration flag */
int changed;
/**
  * If there has not been any special function call
  * then the xml data must be saved in a buffer to
  * be transformed to native configuration files.
  * If a special function is called then no saving
  * to a buffer is needed. So :
  * saveToBuffer > 0 --> to save to a buffer
  * saveToBuffer = 0 --> not to use a buffer
  */
int saveToBuffer;
/**
  * A flag to indicate that parameters of a special
  * function must be saved.
  */
int saveParams;
int saveParamNow;
/**
  * A flag to indicate that this parameter of a special
  * function must not be saved.
  */
int saveParamNot;
/**
  * Changed service's name
```

```
  */
char *serviceName;
/**
  * Special function called
  */
char *sf_name;
/**
  * Buffer to store a service’s xml portion
  */
char *serviceXml;
/**
  *
  */
int tmp_fd;
/**
  *
  */
stringstream *outs;
/**
  * Used to indicate dynamic or static configuration
  * request.
  */
int changeID;
/**
  * Used to indicate service’s sections.
  */
int *sectionID;
/**
  * The number of changed sections
  */
int sectionIndex;
/**
  * Used to indicate a specific special function.
  */
int specialIndex;
/**
  * Parameters for special function
  */
char **params;
//@}
```

```
//private :
    /**
  * This is the formatter object that is used to output the data
      * to the target. It is set up to format to the standard output
  */
    XMLFormatter    fFormatter;
};
```

Η κλάση ServiceMgr είναι υπεύθυνη για την διαχείριση των πραγματικών υπηρεσιών και την αναπαράστασή τους στο σύστημα διαχείρισης. Χρησιμοποιεί έναν SAX XML Parser για να επεξεργαστεί τα αρχεία περιγραφής των υπηρεσιών που ρίσκονται σε έναν συγκεκριμένο κατάλογο του συστήματος. Έτσι δημιουργεί αντικείμενα νέων υπηρεσιών για κάθε πραγματική υπηρεσία. Όταν κάποια διαχειριστική ενέργεια εκτελεσθεί για μία από τις υπηρεσίες ο ServiceMgr διαλέγει την σωστή υπηρεσία και εκτελεί την ενέργεια. Ακολουθεί ο ορισμός της κλάσης και μια σύντομη περιγραφή της.

```
#include "ConfXMLParser.hpp"
#include "ConfigHandlers.hpp"
#include "MainXmlHandlers.hpp"
#include "MainXslHandlers.hpp"

using namespace std;

/**
  * This class is responsible for dealing with the real services
  * and their represantation in the configurator.
  * It uses a sax parser to parse the configuration xml files
  * of services in a specific directory. It extracts all info
  * for each service and creates instanses of the Service class
  * for each one.
  * When changes to a service have taken place, the ServiceMgr
  * applies changes or calls special functions through the rigth
  * Service object in memory.
  */
class ServiceMgr {

public:
/** @name public members */
//@{
/** The parser used for parsing the configuration files */
SAXParser *parser;
/** The handler used for parsing the configuration files */
ConfigHandlers *handler;
/** The handler used for parsing the main xml file */
MainXmlHandlers *mainXmlHandler;
/** The handler used for parsing the main xml file */
MainXslHandlers *mainXslHandler;
/** The system directory with all configuration files */
```

```
char *config_dir;
/** The number of services as found in config_dir */
int no_of_services;
/** The services that can be configured */
Service *service;
//@}

/** @name Constructor and destructor */
//@{
/**
  * Constructor
  */
ServiceMgr();
/**
  * Destructor
  */
~ServiceMgr();
//@}

/** @name Methods dealing with services */
//@{
/**
  * Method used to add a new Service object in the
  * list of Services.
  */
Service *addService();
/**
  * Method used to create Service object for each service
  * with the administrative info found in configuration
  * file.
  */
int getServices();
/**
  * Apply changes to services as given by ConfXMLParser
  */
char * applyChanges(ConfXMLParser *conf);
/**
  * Create mail.xml appending the entity references and
  * namespaces for services, main.xsl path, base.dtd path.
  */
int createMainXML();
```

```
/**
  * Create main.xsl appending the xsl-includes and
  * namespaces for services, main.xsl path, base.dtd path.
  */
int createMainXSL();
/**
  * Create base.dtd appending the 3-entity related lines to
  * the base.dtd and namespaces in nsbase.dtd
  */
int createBaseDTD(Service *, int no_of_services);
//@}
};
```

Αυτή η κλάση αντιπροσωπεύει την ύπαρξη μιας υπηρεσίας συστήματος στο σύστημα διαχείρισης και τον τρόπο με τον οποίο αυτή μπορεί να γίνει αντικείμενο διαχείρισης σύμφωνα με την πληροφορία που συγκεντρώθηκε από το διαχειριστικό αρχείο της υπηρεσίας. Ακολουθεί η περιγραφή της και ο ορισμός της.

```
#include <parsers/SAXParser.hpp>
#include "SpecialFunction.hpp"

#ifndef _SERVICE_HPP_
#define _SERVICE_HPP_

class ServiceSection {

public:
ServiceSection();
~ServiceSection();

char *name;
char *section_script;
char *native2xml;
char *xml2native;
};


/**
  * This class represents a specific system service and
  * the way to administer it from the configurator
  * as seen by its' configuration file:
  * Each service has native configuration files.
  * The way to transform them to a single xml file
  * and backwards must be provided (through scripts).
  * A DTD for such a file must also be provided and
  * optional are the service's start/stop script (if
  * existing) as the existense of Special Functions
  * (though usefull).
  */
class Service {

public:
/** @name public members */
```

```
//@{
/** The name of the service */
char *name;
/** A description for the service */
char *description;
/** A path to service’s XML file */
char *xml_path;
/** A path to service’s XSL file */
char *xsl_path;
/** The service’s main namespace */
char *ns_name;
/** The service’s main namespace URI */
char *ns_uri;
/** A path to service’s DTD file */
char *dtd_path;
/** Under which DTD definition to place the DTD entity for the ser
char *dtd_mountpoint;
/** Under which main XML’s tag to place this service’s xml portion
char *xml_mountpoint;
/** Number of sections the service is divided */
int sections_no;
/** Sections the service is divided */
ServiceSection *sections;
/** Binary to create XML for this service from known config-files
char *native2xml;
/** Binary to save xml data in the config-files */
char *xml2native;
/** Service’s start/stop/restart script/binary */
char *service_script;
/** Number of special functions */
int sf_no;
/** Special functions */
SpecialFunction *sf;
/** Result of a configuration change */
char *confResult;
/** Pointer to the next Service*/
Service *next;
//@}

/** @name Constructor and destructor */
//@{
```

```
/**
  * Constructor
  */
Service();
/**
  * Constructor with parameter
  */
Service(char *config);
/**
  * Destructor
  */
~Service();
//@}

/** @name Service handling methods */
//@{
/**
  * Restart the service
  */
int start_service();
/**
  * Stop the service
  */
int stop_service();
/**
  * Restart the service
  */
int restart_service(int);
/**
  * create service’s xml file from native configuration files
  */
int createXML(int);
/**
  * Create native configuration files from xml
  */
int createNative(char *, int);
/**
  * Restore native configuration files in case of a failure
  */
int restoreNative(int);
//@}
```

```
};

#endif
```

Αυτή η κλάση χρηησιμοποείται για να περιγράψει μια ειδική διαχειιστική ενέργεια που ανήκει σε μια υπηρεσία και λαμβάνει χώρα με την εκτέλεση ενός εξωτερικού προγράμματος.

```
#include "rootdir.h"

#ifndef _SPECIAL_FUNCTION_HPP_
#define _SPECIAL_FUNCTION_HPP_

/**
  * This class is used for administrative tasks,
  * such as running a script or executing commands.
  * It represents a specific script or command and
  * the way to execute it.
  */
class SpecialFunction {
public:
/** @name public members */
//@{
/** The name of this special function */
char *name;
/** The path to this service's executable */
char *path;
/** Parameters for the execution */
char **param;
/** Number of parameters */
int param_number;

char **param_name;

char **param_info;
/** A pointer to another special function */
SpecialFunction *next;
//@}

/** @name Constructor and destructor */
//@{
/**
  * Constructor
  */
SpecialFunction();
```

```
/**
  * Destructor
  */
~SpecialFunction();
//@}
/**
  * Method to execute the real script or command
  */
int exec (char **);
};

#endif
```

4.2 Κεντρικός Πράκτορας

Ο Core Agent υλοποιήθηκε με τη δημιουργία των κλάσεων
- AgentXMLParser
- ConfAgent
- DOMPrint
- DOMPrintFormatTarget
- DomService

Αυτή η κλάση αποτελεί ασικό μέρος του κεντρικού πράκτορα. Χρησιμοποιείται για να την επεξεργασία της κεντρικής άσης του συστήματος και των αιτήσεων των εξυπηρετητών πελατών. Ακολουθεί ο ορισμός της και μια σύντομη περιγραφή.

```
#include <iostream>
#include <dom/DOM.hpp>
#include <dom/DOM_NamedNodeMap.hpp>
#include <parsers/DOMParser.hpp>
#include <framework/XMLFormatter.hpp>

#include "DomService.hpp"
#include "DOMPrintFormatTarget.hpp"

using namespace std;

/**
  * This class implements the core part of the agent.
  * It is used to parse the main xml file and keep
  * the current configuration state of a system in
  * memory.
  * This is achieved by accepting configuration requests
  * from clients (such us the HTML client), examing them
  * and apply them in memory. The parsed xml document after
  * that is sent to the main configurator, responsible for
  * the actual changes in the system. Changes not accepted
  * by the configurator are finally discarded by the agent
  * by reparsing the main xml file from storage media.
  */
class AgentXMLParser
{
public:
bool changed;
bool redirection;
int curid;
int sf_flag;
DOMParser *parser;
DOMParser *redir;
DOM_Document redirect;
DOM_Node redirectNode;
dom_info *info;
```

```
char *confMessage;
DOM_Node root;
DOM_Document doc;
DOM_Node docNode;

/** @name Constructor and destructor */
//@{
/**
  * Default constructor
  */
AgentXMLParser();
/**
  * Default destructor
  */
~AgentXMLParser();
//@}

/** @name Methods used for parsing and applying changes */
//@{
/**
  * Make root the requested DOM_Node, as implied by path.
  * Given a specific root DOM_Node, path indicates the
  * relative to that node position in the DOM tree of the
  * node requested. Parsed from left to right every number
  * in path indicates the number of a child and every '.'
  * character one level deeper in the DOM tree. '/' indicates
  * the root document node.
  *
  * @param root A node taken as root
  * @param path A relative to that node path
  */
void pathToNode(DOM_Node& root, char *path);
/**
  * Method used to mark a node as requested
  * (attribute requested)
  *
  * @param req The node to be marked
  */
void markNode(DOM_Node& node);
/**
  * Method used to unmark a requested node
```

```
  * (attribute requested)
  *
  * @param req The node to be unmarked
  */
void unmarkNode(DOM_Node& node);
/**
  * Method used to apply changes requested in the DOM tree in memo
  *
  * @param doc The document node of the xml tree
  * @param changes  The dom_info holding the path and the changes
  * requested by the client
  */
int applyDomChanges(DOM_Document& doc,dom_info& changes);
/**
  * Apply changes to attributes in node and nodes above it
  * Provides the configurator the capability to identify changed
  * nodes in the xml data given from agent
  *
  * @param node Node under which all changes took place
  * @param flag Provides roll-back capability in applying
  *  attribute changes
  */
int applyAttrChanges(DOM_Node& node, int flag);
/**
  * Used to create a link to the ch-th child of node
  * indicated by path
  */
char *createLink(char *path, int ch);

/**
  * Start a new parsing
  */
void start();
/**
  * If DOM tree nodes changed indicate changes with attribute "cha
  */
bool domChanged();
/**
  * Reparse the xml file in case of configuration failure
  */
void reset();
```

```
/**
  * Send xml data to a socket stream
  */
char *save(DOM_Node &, int, char *);
/**
  * Stop proccessing client's request
  */
void stop();
//@}
};

/** @name Overriden stream out operators */
//@{
/**
  * Stream out a DOM string
  * @param toWrite The DOM string to stream out
  */
ostream&
operator<<(ostream& target, const DOMString& toWrite);
/**
  * Stream out a DOM node.
  * Stream out a DOM node and recursively all of its children
  * @param toWrite The DOM node to stream out
  */
ostream&
operator<<(ostream& target, DOM_Node& toWrite);
/**
  * Basic formatting on xml data.
  * Basic formatting required to turn the Unicode based xml data
  * from the parser into a form that can be used on non-Unicode
  * based systems, i.e. local or generic text encodings.
  */
XMLFormatter&
operator<< (XMLFormatter& strm, const DOMString& s);
//@}
```

Αυτή η κλάση αποτελεί το ασικό μέρος του κεντρικού πράκτορα. Υ-λοποιεί τον αλγόριθμο εξυπηρέτησης των διαφόρων εξυπηρετητών πελατών, επεξεργάζεται τις αιτήσεις με τη οήθεια του AgentXMLParser , αλλάζει τη άση του συστήματος και στέλνει τις αιτήσεις στον πυρήνα. Ακολουθεί ο ορισμός της και μια σύντομη περιγραφή της.

```
#include "AgentXMLParser.hpp"
#include "DomPrint.hpp"


/**
  * This class implements the agent for the configurator.
  * It receives requests from a client through the domServer,
  * processes the requests with the agParser and either replies
  * the client immediately or applies any changes to the memory
  * and sends the xml file to the main configurator and replies
  * after the proccessing of the configurator reply.
  */
class ConfAgent
{
public:
AgentXMLParser *agParser;
DomService *domServer;
DomPrint *confClient;

/** @name Constructor and destructor */
//@{
/**
  * Constructor
  */
ConfAgent();
/**
  * Destructor
  */
~ConfAgent();
//@}

/**
  * Method to get client request
  */
int getClientRequest();
```

```
/**
  * Process client’s request:
  * Checks for changes and if done, apply the changes
  * first in the dom tree in memory and via the main
  * part of the configurator in services.
  *
  * @return The number of uploaded changes.
  */
void processRequest();
/**
  * Return to the client a reply in the form raw
  * xml data, taken from the target node indicated
  * by the client’s request and the original declarations
  * and processing instuctions
  */
int replyClient();

};
```

Αυτή η κλάση υλοποιεί την επικοινωνία με τον εξυπηρετητή πελατών. Ακολοθεί ο ορισμός της και μια σύντομη περιγραφή της.

```
#include <sys/un.h>
#include "../Configurator/rootdir.h"
/**
  * Info received from client:
  * path of the node to the dom tree
  * & uploaded data
  */
struct dom_info{
char path[50];
char request[50];
char **data;
int valuesCounter;
};

/**
  * This class implements the communication of the agent
  * with the client.
  */
class DomService {

public:
int lsd;
int sd;
struct sockaddr_un sun;
unsigned int sun_size;
char *filename;
char *cgiRequest;

/** @name Constructor and destructor */
//@{
/**
  * Default constructor
  */
DomService();
/**
  * Default constructor
  */
```

```
~DomService();
//@}

/** @name methods for the communication with the client */
//@{
/**
  * Wait for a connection request from client
  */
int acceptConnection();
/**
  * Close a connection with a client
  */
int closeConnection();
/**
  * Receive a request from the client
  * @param dom_info The struct where the request will be saved
  */
int receiveHtmlInfo(struct dom_info &);
//@}
};
```

Αυτές οι δύο κλάσεις υλοποιούν την επικοινωνία με τον πυρήνα του συστήματος. Ακολοθεί ο ορισμός τους και μια σύντομη περιγραφή τους.

```
static const int errorMessageLen = 50 ;

/**
  * Agent - main configurator communication.
  * This class implements the communication between the
  * agent and the main configurator.
  */
class DomPrint
{
int sent;
public:
int sockd;
char *result;
/** @name Constuctor and destructor */
//@{
/**
  * Default constructor
  */
DomPrint();
/**
  * Default destructor
  */
~DomPrint();
//@}

/** @name agent - configurator communication methods */
//@{
/**
  * Try to connect to the configurator
  * @return A positive integer if connected
  */
int connectToConf();
/**
  * Get the reply of the configurator
  * @return A message from the configurator if something
  *  went wrong or nothing in case of success.
  */
```

```
char *getResultFromConf();
/**
  * Disconnect from configurator
  */
int disconnectFromConf();
//@}
};

#include <unistd.h>
#include <framework/XMLFormatter.hpp>

/**
  * This class implements the format target interface
  * needed to stream out a dom xml tree to the socket
  */
class DOMPrintFormatTarget : public XMLFormatTarget
{
public:
/**
  * The socket descriptor of the connection to the agent
  */
int sockd;

/** @name Constructors and destructor */
//@{
/**
  * Default constructor
  */
DOMPrintFormatTarget();
/**
  * Constructor with parameter
  *
  * @param sd The socket descriptor of the connection to the agent
  */
DOMPrintFormatTarget(int sd);
/**
  * Default destructor
  */
~DOMPrintFormatTarget();
//@}
```

```
/** @name Implementation of the format target interface */
//@{
/**
  * Implements the format target interface
  *
  */
void writeChars(const XMLByte* const toWrite,
const unsigned int count,
XMLFormatter * const formatter);
//@}

private:
/* @name Unimplemented methods */
//@{
    DOMPrintFormatTarget(const DOMPrintFormatTarget& other);
    void operator=(const DOMPrintFormatTarget& rhs);
//@}
};
```

4.3 Εξυπηρετητής Πελατών

Ο HTML Client Agent χρησιμοποιείται για την εξυπηρέτηση αιτήσεων από HTML πελάτες. Αναλύει τις αιτήσεις και τις προσαρμόζει στο πρωτόκολλο επικοινωνίας με τον κεντρικό πράκτορα.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/utsname.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <html-lib.h>
#include <cgi-lib.h>
#include "../Configurator/rootdir.h"
```

```
#define SFILE "/tmp/cgi2a"

int cgi_start();
int ag_connect();
int changed();
void send_request();
void get_response();
void ag_disconnect();
void cgi_end();

int path_len = 50;
int data_len = 50;
int entry_len = 100;
char *filename = SFILE;

char *xmlPath;
char *request;
int requestFlag = 0;
llist entries;
int newValuesLen;
char *newValues;
int sockd;

int main (int argc, char *argv[], char *envp[]) {

cgi_start();

ag_connect();

send_request();

get_response();

ag_disconnect();

cgi_end();

return 0;
}

int
```

```
cgi_start()
{
char *xml_header;
char *path;
xmlPath = (char *)malloc(path_len);
request = (char *)malloc(path_len);
strcpy(request, "");
path = PATH_INFO;
if ((path == NULL) || strlen(path) == 0) {
strcpy(xmlPath, "/");
} else {
if (strlen(path) < path_len) {
strcpy(xmlPath, path);
} else {
strcpy(xmlPath, "Too Long\n");
}
if(xmlPath[strlen(path)-1] == '.') {
xmlPath[strlen(path)-1] = '\0';
}
}
newValuesLen = read_cgi_input(&entries)-1;

xml_header = "Content-Type: text/xml\n\n";
write(1, xml_header, strlen(xml_header));
return 0;
}

int
ag_connect()
{
int k;
struct sockaddr_un server;
unsigned int sun_size;
sockd = socket(AF_UNIX, SOCK_STREAM, 0);
server.sun_family = AF_UNIX;
strncpy(server.sun_path, filename, sizeof(server.sun_path));
sun_size = SUN_LEN(&server);
k = connect(sockd, (struct sockaddr *)&server, sun_size);
if (k < 0) {
printf("cgi not connected to agent: %s", strerror(errno));
    }
```

```
    return k;
}

int
changed()
{
node *current;
char c;
char multiple[50] = "";
int first_multiple = 0;
int i = 0;
int k = 0;
int form_size = 0;
int pos = 0;
int incr_path = 0;
int last = 0;
int last_node = -1 ;
// Get elements & length
form_size = newValuesLen;

if (form_size > 0) {
current = entries.head;
newValues = (char *)malloc(2*data_len*form_size);
for (i=0; i<newValuesLen; i++) {
if ((!strcmp(current->entry.name, "submit")) ||
(!strcmp(current->entry.name, "SUBMIT")) ||
(!strcmp(current->entry.name, "Submit")) ) {
current = current->next;
}
if (strlen(current->entry.name) < data_len) {
if (incr_path > 0) {
last = strlen(current->entry.name) - 1;
// Provides the capability to load multiple name-value pairs
// with the same name. In the xml tree this is represented
// as child siblings one after another, so to distinquish
// them the last number of the path must be incremented
// each time. The client must provide as '.' chars in the
// end of the path as the digits of the the maximum same-name
// uploaded values.
while (current->entry.name[last] == '.') {
current->entry.name[last] = '\0';
```

```
last--;
}
while (current->entry.name[last] != '.') {
last--;
}
if (last_node == -1) {
last_node = atoi(current->entry.name + last + 1);
first_multiple = last_node;
}
last_node = first_multiple + incr_path;
if (last_node < 10) {
c = '0' + last_node;
current->entry.name[last + 1] = c;
current->entry.name[last + 2] = '\0';
} else {
k = last_node/10;
c = '0' + k;
current->entry.name[last + 1] = c;
k = last_node - k * 10;
c = '0' + k;
current->entry.name[last + 2] = c;
current->entry.name[last + 3] = '\0';
}
}
strcpy(newValues+i*entry_len, current->entry.name);
if (!strcmp(current->entry.name, "REQUEST")) {
requestFlag = 1;
}
} else {
strcpy(newValues+i*entry_len, "");
}
if (strlen(current->entry.value) < data_len) {
if (!strcmp(current->entry.name, "REQUEST")) {
strcpy(request, current->entry.value);
} else {
strcpy(newValues+i*entry_len+data_len, current->entry.value);
}
} else {
strcpy(newValues+i*entry_len+data_len, "");
}
pos += entry_len;
```

```
if (i < newValuesLen-1) {
if (incr_path == 0) {
strcpy(multiple, current->entry.name);
}
if (!strcmp(multiple, current->next->entry.name)) {
incr_path++;
} else {
incr_path = 0;
last_node = -1;
}
current = current->next;
}
}
} else {
newValuesLen = 0;
}
return newValuesLen;
}

void
send_request()
{
int i = 0;
int sent = 0;
sent += write(sockd, (char *)xmlPath, data_len);
if (changed()) {
if (requestFlag != 0) {
newValuesLen--;
sent += write(sockd, &newValuesLen , sizeof(newValuesLen));
newValuesLen++;
} else {
sent += write(sockd, &newValuesLen, sizeof(newValuesLen));
}
for (i=0; i<newValuesLen; i++) {
if (strcmp(newValues+i*entry_len, "REQUEST")) {
write(sockd, newValues+i*entry_len, 2*data_len);
sent += 2* data_len;
}
}
sent = 0;
sent += write(sockd, request, data_len);
```

```
} else {
sent += write(sockd, &newValuesLen, sizeof(newValuesLen));
}
requestFlag = 0;
}

void
get_response()
{
char *xmldata;
char *xmlData;
int length = 0;
int total = 0;
    xmldata = (char *)malloc(50000);
    xmlData = xmldata ;
    while ((length = read(sockd, xmldata, 1000)) > 0) {
        xmldata += length;
        total += length;
    }
write(1, xmlData, total);
}

void
ag_disconnect()
{
close(sockd);
}

void
cgi_end()
{
list_clear(&entries);
}
```

4.4 Πελάτης

Βιβλιογραφία

[1] Robin Burk, UNIX administration

[2] inAccess Networks, www.inaccessnetworks.com/projects/openrouter

[3] Apache Software Foundation, www.apache.org

```
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include "ConfBase.hpp"

using namespace std;

ConfBase::ConfBase()
{
confParser = new ConfXMLParser();
confService = new ConfService();
servMgr = new ServiceMgr();
}

ConfBase::~ConfBase()
{
delete confParser;
delete confService;
delete servMgr;
}

void
ConfBase::initialize()
{
confParser->init();
servMgr->getServices();
}

void
ConfBase::getAgentRequest()
{
confService->startServAgent();
confParser->parseStart(*confService);
}

void
ConfBase::processRequest()
{
char *confResult;
confResult = new char[50];
```

```
strcpy(confResult, servMgr->applyChanges(confParser));
confService->setResponse(confResult);
confParser->parseStop();
delete [] confResult;
}

void
ConfBase::notifyAgent()
{
confService->notifyAgent();
confService->stopServAgent();
}

void
ConfBase::terminate()
{
confParser->terminate();
}

/**
  * Configurator
  */
int
main(int argC, char* argV[])
{
ConfBase *configurator = new ConfBase();
/* Initialize the configurator */
configurator->initialize();


    if (argC == 2 && !strcmp(argV[1], "-d")) {
   if (daemon(1, 0) < 0) {
  cerr << "Error daemonizing:" << strerror(errno) << endl;
  return 1;
   }
}

while (true) {
/* Get a request from agent */
configurator->getAgentRequest();
/* Apply this request to the system */
```

```
configurator->processRequest();
/* Notify agent about results */
configurator->notifyAgent();

}
/* Terminate the configurator */
configurator->terminate();

return 0;
}
```

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<Configurator>
<nsbase/>
<name>IAN Configurator</name>
    <info>How could administration be divided?</info>
    <users>
     <name>Administration of users</name>
       <info>Administration of Users</info>
</users>
    <services>
<name>System Services</name>
       <info>Administration of several system services</info>
    </services>
<tasks>
     <name>Administrative Tasks</name>
     <info>Common administrative tasks</info>
    </tasks>
</Configurator>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XS
<xsl:output method="html"></xsl:output>
<xsl:variable name="base">
<xsl:value-of select="’/cgi-bin/htmlgen’"/>
</xsl:variable>
<xsl:variable name="result">
<xsl:value-of select="//@result"/>
</xsl:variable>
```

```
<xsl:variable name="elink">
<xsl:value-of select="//@elink"/>
</xsl:variable>
<xsl:variable name="blockpath">
<xsl:value-of select="substring-before($elink,'..')"/>
</xsl:variable>
<xsl:variable name="relpath">
<xsl:value-of select="substring-after($elink,'..')"/>
</xsl:variable>
<xsl:template match="/">
<xsl:variable name="back">
<xsl:choose>
<xsl:when test="string-length($blockpath)!=0">
<xsl:value-of select="$blockpath"></xsl:value-of>
<xsl:value-of select="'.'"></xsl:value-of>
<xsl:value-of select="$relpath"></xsl:value-of>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="//@elink"></xsl:value-of>
</xsl:otherwise>
</xsl:choose>
</xsl:variable>
<html>
<head>
<title>
C O N F I G U R A T O R
</title>
</head>
<body>
<xsl:variable name="backlink">
<xsl:if test="string-length($blockpath)!=0">
<xsl:for-each select="//*[@request]">
<xsl:value-of select="./@reqpath"/>
</xsl:for-each>
</xsl:if>
</xsl:variable>
<a>
<xsl:attribute name="href">
<xsl:choose>
<xsl:when test="string-length($backlink)!=0">
<xsl:value-of select="$base"></xsl:value-of>
```

```
<xsl:value-of select="$blockpath"></xsl:value-of>
<xsl:value-of select="$backlink"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$base"></xsl:value-of>
<xsl:value-of select="substring($back,0,string-length($back)-2)"/>
</xsl:otherwise>
</xsl:choose>
</xsl:attribute>
<xsl:value-of select="’Go back’"></xsl:value-of>
</a>
<xsl:choose>
<xsl:when test="string-length($result)!=0">
                    <h4>Last request failed</h4>
      <h4><xsl:value-of select="$result"></xsl:value-of></h4>
 </xsl:when>
</xsl:choose>
<td colspan="2">
<xsl:apply-templates select="." mode="root"></xsl:apply-templates>
</td>
<form method="POST">
<xsl:attribute name="action">
<xsl:choose>
<xsl:when test="string-length($backlink)!=0">
<xsl:value-of select="$base"></xsl:value-of>
<xsl:value-of select="$blockpath"></xsl:value-of>
<xsl:value-of select="$backlink"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$base"></xsl:value-of>
<xsl:value-of select="substring($back,0,string-length($back)-2)"/>
</xsl:otherwise>
</xsl:choose>
</xsl:attribute>
<div>
<p>
<input type="submit" name="submit" value="Go back"></input>
</p>
</div>
</form>
</body>
```

```
</html>
</xsl:template>
<xsl:template match="Configurator|admin|users|services|tasks" mode="root
<xsl:variable name="rootlink">
<xsl:value-of select="$base"></xsl:value-of>
<xsl:value-of select="@elink"></xsl:value-of>
</xsl:variable>
<h2>
<xsl:value-of select="child::name"></xsl:value-of>
</h2>
<h3>
<xsl:value-of select="child::info"></xsl:value-of>
</h3>
<p>
<xsl:for-each select="child::*[position()>2]" mode="child">
<a>
<xsl:attribute name="href">
<xsl:value-of select="$rootlink"></xsl:value-of>
<xsl:number value="position()+1"></xsl:number>
</xsl:attribute>
<xsl:value-of select="child::name"></xsl:value-of>
</a>
<h4>
<xsl:value-of select="child::info"></xsl:value-of>
</h4>
</xsl:for-each>
</p>
</xsl:template>
<xsl:template match="Configurator|admin|users|user_monitor|services|task
<xsl:param name="link">index.html</xsl:param>
<a>
<xsl:attribute name="href">
<xsl:value-of select="$link"></xsl:value-of>
<xsl:number value="position()-1"></xsl:number>
</xsl:attribute>
<xsl:value-of select="child::name"></xsl:value-of>
</a>
<h4>
<xsl:value-of select="child::info"></xsl:value-of>
</h4>
</xsl:template>
```

```
<xsl:template match="name|info|usera:user_admin" mode="child">
</xsl:template>
</xsl:stylesheet>


<!ELEMENT Configurator (nsbase,name,info,users,services,tasks)>
<!ELEMENT nsbase ANY>
<!ATTLIST nsbase xmlns:usera CDATA #REQUIRED xmlns:userb CDATA #RE
<!ELEMENT name (#PCDATA)>
<!ELEMENT info (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT default (#PCDATA)>
<!ELEMENT users (name,info)>
<!ELEMENT user_monitor (name,info)>
<!ATTLIST user_monitor changed (0 | 1 |2) #REQUIRED>
<!ELEMENT services (name,info,service_admin*)>
<!ELEMENT service_admin ANY>
<!ELEMENT tasks (name,info,task_admin*)>
<!ELEMENT task_admin ANY>
```

```
#include <new>
#include <cwchar>
#include <csignal>
#include <cmath>
#include <cstring>
#include <cstdlib>
#include <fstream>
#include <cerrno>

#include "ConfAgent.hpp"

using namespace std;

ConfAgent::ConfAgent()
{
agParser = new AgentXMLParser();
domServer = new DomService();
}

ConfAgent::~ConfAgent()
{
delete domServer;
delete agParser;
}

int
ConfAgent::getClientRequest()
{
agParser->start();
domServer->acceptConnection();
domServer->receiveHtmlInfo(*(agParser->info));
return 0;
}

void
ConfAgent::processRequest()
{
/* Check for uploaded changes */
if (agParser->info->valuesCounter > 0) {
/* If real changes exist, apply to the dom tree in memory */
```

```
agParser->applyDomChanges(agParser->doc, *(agParser->info)) ;

if (agParser->domChanged()) {

confClient = new DomPrint();

if (confClient->connectToConf() < 0) {
cout << "Cannot connect to main conf!\n";
}
agParser->save(agParser->docNode, confClient->sockd, "");
confClient->getResultFromConf();
strcpy(agParser->confMessage, confClient->result);
if (strlen(agParser->confMessage)) {
cout << "CONF MESSAGE :" << agParser->confMessage << "."<< endl;
}
confClient->disconnectFromConf();

if (agParser->sf_flag == 0) {
/* XML configuration context. */
/* != 0 -> ERROR. Special function has failed */
if (strlen(agParser->confMessage) != 0) {
agParser->reset();
} else {
agParser->applyAttrChanges(agParser->root, 1) ;
}
} else { /* Special Function context. */
if (strlen(agParser->confMessage) == 0) {
agParser->reset();
} else {
agParser->applyAttrChanges(agParser->root, 1) ;
}
}
delete confClient;
}
agParser->changed = false;
}
return;
}

int
ConfAgent::replyClient()
```

```
{
if (strlen(agParser->info->request)) {
cout << "Requested path #" << agParser->info->request << "#" << endl;
agParser->save(agParser->doc, domServer->sd, agParser->info->request);
} else {
cout << "Requested path #" << agParser->info->path << "#" << endl;
agParser->save(agParser->doc, domServer->sd, agParser->info->path);
}
agParser->stop();
domServer->closeConnection();
return 0;
}


int main(int argc, char**argv )
{

ConfAgent *agent =  new ConfAgent();

if (argc == 2 && !strcmp(argv[1], "-d")) {
if (daemon(1, 0) < 0) {
    cerr << "Error daemonizing:" << strerror(errno) << endl;
    return 1;
}
}

while (true) {

agent->getClientRequest();

agent->processRequest();

agent->replyClient();

}

delete agent;

    return 0;
}
```

```
  * +----------+-----------------+--------------+
  * | command  | no of arguments | arguments .. |
  * +----------+-----------------+--------------+
  * +----------+-----------------+--------------+
  * | 12 bytes |    4 bytes      |      ..      |
  * +----------+-----------------+--------------+
  * (2) reply
  * +---------+
  * |  reply  |
  * +---------+
  * +---------+
  * | 4 bytes |
  * +---------+
  *
  */

#include <sys/un.h>

#include <dom/DOM.hpp>

#define COMMAND_LEN 12

class Native2Xml
{
bool xmlDecl;
/**
  * The number of nodes created.
  * For debugging purposes.
  */
int stopFlag;
/** For server-client communication */
//@{
struct sockaddr_un sun;
unsigned int sun_size;
char *sfile;
char *base_dtd;
int lsd;
int sd;
//@}
/** @name Client-request related members */
//@{
```

```
/**
  * The name of an action requsted by a client
  */
char *command;
/**
  * The number of arguments sent by the client
  */
int commandArgsCount;
/**
  * The parameters of the action requested
  */
char **commandArgs;
/**
  * The reply to be sent to the client.
  */
int reply;
//@}
/** @name Members used for xml parsing */
//@{
/**
  * The xml document.
  */
DOM_Document doc;
/**
  * The root node of the xml document under construction
  */
DOM_Node root;
/**
  * The current parsed node.
  * Used as reference point for every action.
  */
DOM_Element *current;
DOM_Node parent;
DOM_Node child;
DOM_Node next;
DOM_Node previous;
//@}

public:
int nodesCreated;
/** @name Constructor and destuctor */
```

```
//@{
/**
  * Default constructor
  */
Native2Xml();
/**
  * Default destructor
  */
~Native2Xml();
//@}
/** @name Implementation of the protocol */
//@{
/**
  * Start listening for connections.
  */
int start_server();
/**
  * Stop listening for requests. In case of a failure or a
  * successfull xml file construction.
  */
int stop_server();
/**
  * Accept a request from a client implementing the specific proto
  * @return 0 in case of a successful request
  *   1 in case of a protocol-relatrd error in the request
  */
int receive_request();
/**
  * Send a reply to the client requested a change indicating the r
  */
int send_reply();
//@}

/** @name Implementation of the xml functions for the creation of
//@{
/**
  * Create an xml document.
  * @param namespaceURI The namespace of the xml dcoument to be cr
  * @param docElement The root element of the xml dcoument to be c
  * @param xml_decl Include or not an xml declaration.
  * @return 0 in case of a successful document creation
```

```
  *   1 in case of failure
  */
int createDoc(char *namespaceURI, char *docElement, int xml_decl);
/**
  * Decides the type of change requested and applies it to the xml
  * document.
  * A request may be valid in protocol terms but may fail due to
  * xml-related reasons.
  * (for example a CDATA node can not have siblings)
  * @return 0 if change requested was valid and succedded
  *  1 if change requsted was invalid or failed
  */
int handle_request();
private:
/**
  * Create a dom node as child of the current node.
  * The node created becomes the current node or not
  * depending to cur parameter.
  * @param name The name of the node to be created
  * @param cdata The character data inside the node.
  *              A '\0' character in case of no data.
  * @param cur   True to make the created node the current
  *              parsed node, false otherwise.
  * @return      0 in case of a successful node creation
  *              1 in case of failure
  */
int createNode(char *name, char *cdata, bool cur);
/**
  * Create an entity reference (which must be declared in a DTD)
  * @param name The name of the entity reference
  * @return 0 if change requested was valid and succedded
  *         1 if change requsted was invalid or failed
  */
int createEntityReference(char *name);
/**
  * Create an attribute for the current node.
  * @param name The name of the attribute to be created.
  * @param value The value of the attribute.
  * @return 0 in case of a successful attribute creation
  *         1 in case of failure
  */
```

```
int createAttr(char *name, char *value);
/**
  * Set current node the parent of the current node.
  * @return 0 in case of success
  *         1 in case of failure
  */
int setParent();
/**
  * Set current node the first child of the current node.
  * @return 0 in case of success
  *         1 in case of failure
  */
int setChild();
/**
  * Set current node the first sibling of the current node.
  * @return 0 in case of success
  *         1 in case of failure
  */
int setNext();
/**
  * Set current node the previous sibling of the current node.
  * @return 0 in case of success
  *         1 in case of failure
  */
int setPrevious();
/**
  * Print the xml document creted in a file.
  * @param xmlfile The name of the file for the xml document to be
  *                saved as.
  * @return 0 in case of success
  *         1 in case of failure
  */
int printDoc(char *xmlfile);
//@}

char *getHostName();
};



/**
```

```
* Through command-line arguments can implement various different
* clients requesting xml parsing and constructing actions from a
* a server.
*
* The server when started creates a new xml dom document in
* memory and a root element and then starts listening for client's
* requests. It provides the client the following capabilities:
* - dom node creation inside the dom tree
* - attribute node creation for a node
* - dom tree navigating -- get parent node
*                       -- get first child
*                       -- get next sibling
*                       -- get previous sibling
* - xml document print to a file
*
* The client-server communication is implemented through a simple
* command-oriented protocol which consists of two basic data units:
* 1. request sent from client to server (1)
* 2. reply from server to client        (2)
* (1) request
* +---------+-----------------+--------------+
* | command | no of arguments | arguments .. |
* +---------+-----------------+--------------+
* +----------+----------------+--------------+
* | 12 bytes | 4 bytes        |     ..       |
* +----------+----------------+--------------+
* (2) reply
* +---------+
* |  reply  |
* +---------+
* +---------+
* | 4 bytes |
* +---------+
*
* Seven(7) different types of clients exist, each one providing a diff
* request to the server:
* (1) addnode (addnode node_name node_cdata)
* To create a new node in the xml document. The node is created as a c
* of the current node. The current node is either the root node for th
* first action or the node we have specified via parsing actions like
* the ones that follow.
```

```
  * (2) addattr (addattr attr_name attr_value)
  * To create a new attribute for the current node.
  * (3) getparent (getparent)
  * To make current node the parent of the current node.
  * (4) getchild (getchild)
  *  To make current node the first child of the current node.
  * (5) getnext (getnext)
  *  To make current node the next sibling of the current node.
  * (6) getprevious (getprevious)
  * To make current node the previous sibling of the current node.
  * (7) dumpdoc (dumpdoc filename)
  * To print the xml document created in a file.
  */


class Native2XmlClient
{
int sockd;
char *cl_name;
char *command;
int args_no;
char *arguments;
int argsLen;
int result;

public:
/** @name Default constuctor and destructor */
//@{
Native2XmlClient();
~Native2XmlClient();
//@}
/**
  * Connect to the server and prepare the request.
  */
int start_client(int argc, char *argv[]);
/**
  * Disconnect from the server.
  */
int stop_client();
/**
  * Send request to the server.
```

```
  */
int send_request();
/**
  * Receive the reply from the server.
  */
int receive_reply();
};
```

```
#include <sax/HandlerBase.hpp>
#include <framework/XMLFormatter.hpp>
#include <util/PlatformUtils.hpp>
#include <util/TransService.hpp>
#include <parsers/SAXParser.hpp>
#include "../Configurator/rootdir.h"

#define TMP_FILE "/tmp/tmpconf.xml"
#define PVFILE   "/tmp/pav.txt"

/**
  * Implements a common interface for the transformation of xml data to
  * native service configuration files. The tmp xml file created by the
  * configurator is parsed and at the end of each element a service-spec
  * script is called with parameters the element’s name and the characte
  * data if any contained.
  * The service-specific script can easily choose how to construct a nat
  * based on these data.
  *
  * To store elements’names and values an array of (char *) is used.
  * This array’s size is relative to the possible depth of the xml.
  */

class Xml2NativeHandlers : public HandlerBase, private XMLFormatTarget
{
public:
   /** @name Default constructor and destructor */
   //@{
   /**
```

```
     * Default constuctor
     */
    Xml2NativeHandlers(const char* const encodingName,
                       const XMLFormatter::UnRepFlags unRepFlags);
    /**
      * Default destructor
      */
    ~Xml2NativeHandlers();
    //@}

    /** @name Implementations of the format target interface */
    //@{
    void writeChars(const XMLByte* const toWrite);

    void writeChars(const XMLByte* const toWrite,
                    const unsigned int count,
                    XMLFormatter* const formatter);
    //@}

    /** @name Implementations of the SAX DocumentHandler interface
    //@{
    void endDocument();

    void endElement(const XMLCh* const name);

    void characters(const XMLCh* const chars, const unsigned int l

    void ignorableWhitespace(const XMLCh* const chars,
                             const unsigned int length);

    void processingInstruction(const XMLCh* const target,
                               const XMLCh* const data);

    void startDocument();

    void startElement(const XMLCh* const name, AttributeList& attr
    //@}


    /** @name Implementations of the SAX ErrorHandler interface */
    //@{
```

```
    void warning(const SAXParseException& exception);
    void error(const SAXParseException& exception);
    void fatalError(const SAXParseException& exception);
    //@}


    /** @name Implementation of the SAX DTDHandler interface */
    //@{
    void notationDecl(const XMLCh* const name,
                      const XMLCh* const publicId,
                      const XMLCh* const systemId);

    void unparsedEntityDecl(const XMLCh* const name,
                            const XMLCh* const publicId,
                            const XMLCh* const systemId,
                            const XMLCh* const notationName);
    //@}
    /**
      * To execute the xml2native script
      */
    void natScript();
    /**
      * Whether a character data value has just been stored or not.
      * To be used when the service-specific script is called to
      * decide to include the last one or two stored values.
      * (When a character data value is stored then the script
      *  must be called with the name of the element and it's value)
      */
    bool valFlag;
    /**
      * The current path where the name of the node is appended
      * and then used for the native2xml script.
      */
    char *currentPath;
    /**
      * An array of character arrays to store parsed data.
      */
    char *value;

private :
```

```
    XMLFormatter    fFormatter;
};
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:shw="http://localhost/shw">
<xsl:template match="shw:shorewall" mode="root">
<xsl:variable name="rootlink">
<xsl:value-of select="$base"/>
<xsl:value-of select="@elink"/>
</xsl:variable>
<xsl:variable name="newrootlink">
<xsl:value-of select="$base"/>
<xsl:value-of select="$blockpath"/>
<xsl:value-of select="'.'"/>
<xsl:value-of select="$relpath"/>
</xsl:variable>
<xsl:variable name="baselink">
<xsl:value-of select="$base"/>
<xsl:value-of select="$blockpath"/>
<xsl:value-of select="'.'"/>
</xsl:variable>
<h2>Shoreline Firewall</h2>
<xsl:choose>
<xsl:when test="string-length($relpath)=0">
<br/>
<a>
<xsl:attribute name="href">
<xsl:value-of select="$rootlink"/>
<xsl:value-of select="'2.2.2'"/>
</xsl:attribute>
<h3>COMMON SETTINGS</h3>
</a>
<br/>
<h3>ZONES</h3>
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:zones/
<table><tr><td width="150">
<table bgcolor="gray" width="100%">
<tr><td>
<table><tr>
<td><h3>Zone: </h3></td>
<td><h3><xsl:value-of select="child::shw:zonename"/></h3></td>
<td><h3>(<xsl:value-of select="child::shw:display"/>)</h3></td>
```

```
</tr></table>
</td></tr>
<tr>
<td align="center">
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:value-of select="’2.2.3.’"/>
<xsl:value-of select="position()-1"/>
</xsl:attribute>
<table width="100">
<tr><td align="center">
<input type="submit" name="submit" value="Change Zone"/>
</td></tr>
</table>
</form>
</td>
</tr>
</table>
</td><td width="390">
<table border="1"  width="100%" bgcolor="#CCCCCA">
<tr width="100%">
<td width="100%">
<table border="1" width="100%"><tr width="100%">
<td align="center" width="20%">Interface</td>
<td align="center" width="20%">Hosts</td>
<td align="center" width="20%">Firewall stopped</td>
</tr></table>
</td>
</tr>
<xsl:for-each select="./shw:zone_entry">
<tr width="100%"><td align="middle" width="100%">
<table width="100%"><tr width="100%">
<td align="center" width="20%"><xsl:value-of select="./shw:ifname"
<xsl:variable name="ipstring">
<xsl:value-of select="./shw:host/shw:ifip/shw:ip"/>
</xsl:variable>
<xsl:variable name="maskstring">
<xsl:value-of select="./shw:host/shw:ifip/shw:mask"/>
</xsl:variable>
<td align="center" width="20%">
```

```
<xsl:if test="string-length($ipstring)!=0">
<xsl:value-of select="./shw:host/shw:ifip/shw:ip"/>
<xsl:if test="string-length($maskstring)!=0">
/<xsl:value-of select="./shw:host/shw:ifip/shw:mask"/>
</xsl:if>
</xsl:if>
</td>
<td align="center" width="20%">
<xsl:variable name="option1"><xsl:value-of select="./shw:host/shw:optior
<xsl:variable name="option2"><xsl:value-of select="./shw:ifname"/></xsl:
<xsl:for-each select="../../shw:interfaces/shw:interface">
<xsl:variable name="option3"><xsl:value-of select="./shw:ifname"/></xsl:
<xsl:if test="$option2=$option3">
<xsl:for-each select="./shw:options/shw:option[position()=4]">
<xsl:variable name="option2"><xsl:value-of select="./shw:value"/></xsl:v
<xsl:if test="string-length($ipstring)!=0 and $option1='yes'">
<xsl:value-of select="'ACTIVE'"/>
</xsl:if>
<xsl:if test="string-length($ipstring)=0 and $option2='yes'">
<xsl:value-of select="'ACTIVE'"/>
</xsl:if>
</xsl:for-each>
</xsl:if>
</xsl:for-each>
</td>
</tr></table>
</td></tr>
</xsl:for-each>
</table>
</td></tr></table>
</xsl:for-each>
<br/>
<table border="1"><tr>
<td width="180">
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
</xsl:attribute>
<table width="100%">
<tr><td valign="top"><h4>Add a new Zone</h4></td></tr>
```

```
<tr><td align="left">Zone name</td>
<td><input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
<xsl:value-of select="'.'"/>
<xsl:number value="2"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="'addzone'"/>
</xsl:attribute>
</input>
<input type="text" size="8">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
<xsl:value-of select="'.'"/>
<xsl:number value="4"/>
</xsl:attribute>
<xsl:attribute name="value">
</xsl:attribute>
</input></td></tr>
<tr><td align="left">Zone display</td>
<td><input type="text" size="10">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
<xsl:value-of select="'.'"/>
<xsl:number value="5"/>
</xsl:attribute>
<xsl:attribute name="value">
</xsl:attribute>
</input></td></tr>
<tr><td align="left">Zone comment</td>
<td><input type="text" size="12">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
<xsl:value-of select="'.'"/>
<xsl:number value="6"/>
</xsl:attribute>
```

```
<xsl:attribute name="value">
</xsl:attribute>
</input></td></tr>
<tr><td align="left">Zone position</td>
<td>
<select>
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
<xsl:value-of select="'.'"/>
<xsl:number value="3"/>
</xsl:attribute>
<option selected="yes">
<xsl:attribute name="value">
<xsl:number value="-1"/>
</xsl:attribute>
</option>
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:zones/
<option>
<xsl:value-of select="./shw:zoneid"/>
</option>
</xsl:for-each>
</select>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="'REQUEST'"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="'/3.3.3'"/>
</xsl:attribute>
</input></td>
</tr><br/><tr>
<td/><td valign="top"><input type="submit" name="submit" value="Add Zone
</tr></table>
</form>
</td>
<td width="180" valign="top" align="center">
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
```

```
</xsl:attribute>
<table>
<tr><td align="center"><h4>Delete Zone</h4></td></tr>
<tr><td align="center"><input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="3"/>
<xsl:value-of select="'.'"/>
<xsl:number value="2"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="'delzone'"/>
</xsl:attribute>
</input>
<select>
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:value-of select="'3.4'"/>
</xsl:attribute>
<option> </option>
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:
<option><xsl:value-of select="./shw:zonename"/></option>
</xsl:for-each>
</select>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="'REQUEST'"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select='"/3.3.3"'/>
</xsl:attribute>
</input></td></tr>
<br/>
<tr><td align="center"><input type="submit" name="submit" value="D
</table>
</form>
</td>
<td width="180" valign="top" align="center">
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
```

```
<xsl:value-of select="’2.2.3.’"/>
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:zones/
<xsl:if test="name(.)=’shw:interfaces’">
<xsl:value-of select="position()-1"/>
</xsl:if>
</xsl:for-each>
</xsl:attribute>
<table>
<tr align="center"><td><h4>Network Interfaces’ Settings</h4></td></tr>
<tr><td align="center">
<input type="submit" name="submit" value="Change"/>
</td></tr>
</table>
</form>
</td>
</tr></table>
<br/><br/>
<h3>POLICIES</h3>
<table width="450"><tr width="100%">
<td width="100%"><table border="1" width="100%" bgcolor="#CCCCCA">
<tr width="100%">
<td width="100%">
<table width="100%"><tr>
<td align="center" width="20%">Source Zone</td>
<td align="center" width="20%">Dest Zone</td>
<td align="center" width="20%">Policy</td>
<td align="center" width="20%">Log Level</td>
<td align="center" width="20%">Limit Burst</td>
</tr></table>
</td>
<td></td>
</tr>
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:polici
<tr width="100%">
<td width="100%">
<table width="100%"><tr>
<td align="center" width="20%"><xsl:value-of select="child::shw:sourcezo
<td align="center" width="20%"><xsl:value-of select="child::shw:destzone
<td align="center" width="20%"><xsl:value-of select="child::shw:pol"/></
<td align="center" width="20%"><xsl:value-of select="child::shw:logl"/><
<td align="center" width="20%"><xsl:value-of select="child::shw:limitb"/
```

```
</tr></table>
</td>
<td>
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:number value="4"/>
</xsl:attribute>
<table valign="bottom">
<tr><td>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="4"/>
<xsl:value-of select="'.'"/>
<xsl:number value="2"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="'delete'"/>
</xsl:attribute>
</input>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="4"/>
<xsl:value-of select="'.'"/>
<xsl:number value="3"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="./shw:policyid"/>
</xsl:attribute>
</input>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="'REQUEST'"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select='"/3.3.3"'/>
</xsl:attribute>
</input>
<input type="submit" name="submit" value="DELETE"/></td></tr>
```

```
</table>
</form>
</td>
</tr>
</xsl:for-each>
</table></td></tr><br/>
<tr><td><table>
<tr><td>
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:value-of select="4"/>
</xsl:attribute>
<input type="submit" name="submit" value="Add a new policy / Change poli
</form>
</td></tr>
</table></td>
</tr></table>
<br/>
<h3>RULES</h3>
<table width="450"><tr>
<td width="100%"><table border="1" bgcolor="#CCCCCA">
<tr width="100%">
<td width="100%">
<table width="100%"><tr>
<td align="center" width="14%">Action</td>
<td align="center" width="14%">Source</td>
<td align="center" width="14%">Dest</td>
<td align="center" width="14%">Protocol</td>
<td align="center" width="14%">Dest ports</td>
<td align="center" width="14%">Cl ports</td>
<td align="center" width="14%">OrigDest</td>
</tr></table>
</td>
<td></td>
</tr>
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:rules/
<tr width="100%">
<td width="100%">
<table width="100%"><tr>
<td align="center" width="14%"><xsl:value-of select="child::shw:action"/
```

```
<td align="center" width="14%"><xsl:value-of select="child::shw:so
<td align="center" width="14%"><xsl:value-of select="child::shw:de
<td align="center" width="14%"><xsl:value-of select="child::shw:pr
<td align="center" width="14%"><xsl:value-of select="child::shw:ds
<td align="center" width="14%"><xsl:value-of select="child::shw:cl
<td align="center" width="14%"><xsl:value-of select="child::shw:de
</tr></table>
</td>
<td>
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:number value="5"/>
</xsl:attribute>
<table valign="bottom">
<tr><td>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="5"/>
<xsl:value-of select="'.'"/>
<xsl:number value="2"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="'delete'"/>
</xsl:attribute>
</input>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$rootlink"/>
<xsl:number value="5"/>
<xsl:value-of select="'.'"/>
<xsl:number value="3"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="./shw:ruleid"/>
</xsl:attribute>
</input>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="'REQUEST'"/>
```

```
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select='"/3.3.3"'/>
</xsl:attribute>
</input>
<input type="submit" name="submit" value="DELETE"/></td></tr>
</table>
</form>
</td>
</tr>
</xsl:for-each>
</table></td></tr>
<br/>
<tr><td><table>
<tr><td>
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$rootlink"/>
<xsl:value-of select="5"/>
</xsl:attribute>
<input type="submit" name="submit" value="Add a new rule / Change rules'
</form>
</td></tr>
</table></td>
</tr></table>
<br/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="tmplink"><xsl:value-of select='substring-before($rel
<xsl:variable name="newtmp"><xsl:value-of select='substring-after($relpa
<xsl:if test="number($tmplink)=2">
<xsl:variable name="tmplink"><xsl:value-of select='substring-before($new
<xsl:variable name="newtmplink"><xsl:value-of select='substring-after($n
<xsl:if test="number($tmplink)=2">
<xsl:variable name="tmplink"><xsl:value-of select='substring-before($new
<xsl:variable name="newtmp"><xsl:value-of select='substring-after($newtm
<xsl:if test="number($tmplink)=2">
<h3>Common Settings</h3>
<form method="POST">
<xsl:attribute name="action">
<xsl:value-of select="$baselink"/>
```

```
<xsl:value-of select="’2.2.2’"/>
</xsl:attribute>
<table border="1"><tr>
<td><table width="50%">
<xsl:for-each select="./shw:configuration/shw:config_sections/shw:
<xsl:if test="position() &lt; 14 and position() &gt; 1 and positio
<tr>
<td><xsl:value-of select="./shw:name"/></td>
<td>
<xsl:choose>
<xsl:when test="position()=4 or position() &gt; 9">
<xsl:variable name="checked">
<xsl:value-of select="./shw:value"/>
</xsl:variable>
<input type="hidden">
<xsl:attribute name="name">
<xsl:value-of select="$baselink"/>
<xsl:value-of select="’2.2.2.’"/>
<xsl:value-of select="position()-1"/>
<xsl:value-of select="’.1.’"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:if test="$checked=’yes’ or $checked=’Yes’ or $checked=’no’ or
<xsl:value-of select="’no’"/>
</xsl:if>
<xsl:if test="$checked=’On’ or $checked=’on’ or $checked=’Off’ or
<xsl:value-of select="’off’"/>
</xsl:if>
</xsl:attribute>
</input>
<input type="checkbox">
<xsl:attribute name="name">
<xsl:value-of select="$baselink"/>
<xsl:value-of select="’2.2.2.’"/>
<xsl:value-of select="position()-1"/>
<xsl:value-of select="’.1’"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:if test="$checked=’yes’ or $checked=’Yes’ or $checked=’no’ or
<xsl:value-of select="’yes’"/>
</xsl:if>
```

```
<xsl:if test="$checked=’On’ or $checked=’on’ or $checked=’Off’ or $check
<xsl:value-of select="’on’"/>
</xsl:if>
</xsl:attribute>
<xsl:if test="$checked=’yes’ or $checked=’Yes’ or $checked=’On’ or $chec
<xsl:attribute name="checked">
<xsl:if test="$checked=’yes’ or $checked=’Yes’ or $checked=’no’ or $chec
<xsl:value-of select="’yes’"/>
</xsl:if>
<xsl:if test="$checked=’On’ or $checked=’on’ or $checked=’Off’ or $check
<xsl:value-of select="’on’"/>
</xsl:if>
</xsl:attribute>
</xsl:if>
</input>
</xsl:when>
<xsl:otherwise>
<input type="text" size="15">
<xsl:attribute name="name">
<xsl:value-of select="$baselink"/>
<xsl:value-of select="’2.2.2.’"/>
<xsl:value-of select="position()-1"/>
<xsl:value-of select="’.1’"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="./shw:value"/>
</xsl:attribute>
</input>
</xsl:otherwise>
</xsl:choose>
</td>
</tr>
</xsl:if>
</xsl:for-each>
</table></td>
<td>
</td></tr>
</xsl:if>
</xsl:for-each>
</table></td>
<td><input type="hidden">
```

```
<xsl:attribute name="name">
<xsl:value-of select="'REQUEST'"/>
</xsl:attribute>
<xsl:attribute name="value">
<xsl:value-of select="'/3.3.3.2.2'"/>
</xsl:attribute>
</input>
<input type="submit" name="submit" value="Apply"/></td>
</tr></table>
</form>
</xsl:if>
</td>
</tr></table>
</xsl:if>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
<xsl:template match="shw:shorewall" mode="child">
<h3>NAME</h3>
<h3>INFO</h3>
</xsl:template>
</xsl:stylesheet>
```

.....

ΙΩΑΝΝΗΣ Λ. ΚΑΡΡΑΣ
Ηλεκτρολόγος Μηχανικός