ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Συγκριτική μελέτη τεχνολογιών UPnP, Web Services, R-OSGi και αξιολόγηση της κλιμακωσιμότητας για την ανάπτυξη κατανεμημένων εφαρμογών

Σπουδαστής:Μαράντζας Πέτρος

pmarantzas@gmail.com

Επιβλέπων καθηγητής: Λέκτορας κ. Μηλιώνης Απόστολος

ΑΘΗΝΑ ΝΟΕΜΒΡΙΟΣ 2013

Ευχαριστίες

Θα ήθελα να αποδώσω θερμές ευχαριστίες σε όλους όσους συνέβαλαν με τον οποιονδήποτε τρόπο για την επιτυχή εκπόνηση της διπλωματικής μου εργασίας.

Αρχικά, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας Λέκτορα κ. Απόστολο Μηλιώνη για την πολύτιμη βοήθειά του και την καθοδήγηση που μου παρείχε σε όλα τα στάδια εκπόνησης της. Οφείλω να επισημάνω πως ήταν πάντα διαθέσιμος να μου προσφέρει την εμπειρία του, τις απαραίτητες γνώσεις και τις κατευθυντήριες για την καλύτερη δυνατή κατανόηση, προσέγγιση και κάλυψη του απαιτούμενου γνωστικού αντικειμένου.

Ευχαριστώ επίσης τους φίλους και συγγενείς από το στενό μου περιβάλλον οι οποίοι υπέδειξαν μια σημαντική ηθική υποστήριξη για την περάτωση των υποχρεώσεών μου τόσο στα πλαίσια της παρούσας διπλωματικής εργασίας όσο και γενικότερα των σπουδών μου στο παρόν Μεταπτυχιακό Πρόγραμμα του Πανεπιστημίου Πειραιώς.

Πάνω απ όλα, θα ήθελα να αναφέρω πως είμαι ευγνώμων στους γονείς μου Κωνσταντίνο και Μαρία Μαράντζα των οποίων η πίστη στις δυνατότητές μου, η ολόψυχη αγάπη και η στήριξή τους, αποτέλεσαν αρωγό και βασικό κίνητρο επίτευξης των στόχων μου.

Η παρούσα διπλωματική εργασία αφιερώνεται στους γονείς μου Κώστα - Μαρία και τον αδερφό μου Χρυσοβαλάντη.

Μαράντζας Πέτρος

Περιεχόμενα

Πρόλογος

Με τις αυξανόμενες απαιτήσεις της τεχνολογίας τα τελευταία έτη, ολοένα και περισσότερο καθίσταται αναγκαία η ευρεία χρήση κατανεμημένων εφαρμογών για την κάλυψη εταιρικών αναγκών αλλά και για την προσχώρησή τους στην οικιακή δικτύωση.

Η παρούσα διπλωματική εργασία πραγματεύεται την συγκριτική μελέτη των τεχνολογιών UPnP, Web Services και R-OSGi, η καθεμία εκ των οποίων συντελεί πρωταρχικό ρόλο για την ανάπτυξη κατανεμημένων εφαρμογών. Πιο συγκεκριμένα, θα αναλυθεί η δομή της αρχιτεκτονικής της κάθε τεχνολογίας, οι βασικές λειτουργίες που τις διέπουν καθώς επίσης οι τρόποι υλοποίησης τους. Θα γίνει αναφορά στα δικτυακά πρωτόκολλα και πρότυπα τα οποία χρησιμοποιούνται για την επίτευξη των αντίστοιχων διαδικασιών της κάθε τεχνολογίας σε επίπεδο κατανεμημένων εφαρμογών. Θα αναλυθούν ζητήματα ασφάλειας για την εκάστοτε αρχιτεκτονική καθώς και η πληρότητα στις απαιτήσεις που προκύπτουν στον εν λόγω τομέα. Παρατίθενται επίσης τα πλεονεκτήματα που απορρέουν από την χρήση των τριών τεχνολογιών για την ανάπτυξη κατανεμημένων εφαρμογών και τέλος εκτίθεται η καταλληλότητά τους στο πεδίο εφαρμογής βάση των χαρακτηριστικών που προκύπτουν από αυτές.

Στο πρακτικό μέρος υλοποιείται η προσομοίωση ενός δικτύου μιας εμπορικής εφαρμογής όπου περιλαμβάνει 9 περιφερειακές αποθήκες Web Services και μια κεντρική όπου έχει το ρόλο του Main Web Service και ενημερώνεται για την πληροφορία των υπολοίπων. Η προσομοίωση υλοποιήθηκε σε περιβάλλον Eclipse ενώ χρησιμοποιήθηκε ένας Tomcat server κατά την εκκίνηση του οποίου φορτώνονται τα Web Services και περιμένουν από τους Clients να επικοινωνήσουν μαζί τους.

Στο σύστημα δημιουργούνται threads πελατών ο αριθμός των οποίων είναι σκόπιμο να μεταβάλλεται συνάγοντας έτσι στον σκοπό της υλοποίησης όπου αφορά στην αξιολόγηση της κλιμακωσιμότητας της υποδομής των Web Services για την ανάπτυξη κατανεμημένων εφαρμογών.

Ακρωνύμια

Ακρωνύμιο	Ορισμός	Ακρωνύμιο	Ορισμός
ARP	Address Resolution Protocol	SOAP	Simple Object Protocol
CP	Control Point	SSDP	Simple Service Discovery Protocol
DCP	Device Control Protocol	UDA	UPnP Device Architecture
DDD	Device Description Document	UPC	Universal Product Code
DHCP	Dynamic Host configuration Protocol	URI	Uniform Resource Identifier
DNS	Domain Name System	URL	Uniform Resource Locator
GENA	General Event Notification Architecture	URN	Uniform Resource Name
HTML	Hypertext Markup Language	UUD	Universally Unique Identifier
HTTP	Hypertext Transfer Protocol	XML	Extensible Markup Language
SCPD	Service Control Protocol Description	UDDI	Universal Discovery Description and Integration
WSDL	Web Services Definition Language	SMTP	Simple Mail Transport Protocol
FTP	File Transfer Protocol	DTD	Document Type Definition
RPC	remote procedure call	TCP	Transmission Control Protocol
UBR	UDDI Business Registry	HTTPS	Hypertext Transfer Protocol Secure
API	Application programming interface	SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture	JVM	Java virtual machine
LDAP	Lightweight Directory Access Protocol	SCR	Service Component Runtime
SLP	Service Location Protocol	RMI	Remote Method Invocation
CDC	Connection Device Configuration	UI	User Interface

Εισαγωγή στις κατανεμημένες εφαρμογές

Με την σταδιακή εξέλιξη των δυνατοτήτων των υπολογιστικών συστημάτων αλλά και με την ταυτόχρονη αύξηση της πολυπλοκότητας των εφαρμογών, οι τελευταίες άρχισαν να απαρτίζονται από συνθετικά μέρη ικανά για την μεταξύ τους συνεργασία. Έτσι με την ραγδαία προσχώρηση και καθιέρωση της χρήσης των υπολογιστικών δικτύων και των επικείμενων δικτυακών πρωτοκόλλων και τεχνολογιών, τα συνθετικά μέρη των εφαρμογών προκειμένου να υπάρξει καλύτερη και αποτελεσματικότερη εκμετάλλευση των νέων δυνατοτήτων, άρχισαν να διαμοιράζονται στους υπολογιστές του δικτύου. Με τον τρόπο αυτό αυξανόταν η απόδοση και η εκμετάλλευση των πόρων του δικτύου.

Ο όρος κατανεμημένες εφαρμογές ουσιαστικά αναφέρεται στο λογισμικό το οποίο εκτελείται σε πολλαπλά υπολογιστικά περιβάλλοντα ενός δικτύου ενώ επίσης καθίταται δυνατή η αλληλεπίδραση των συνθετικών μερών του με σκοπό την επίτευξη συγκεκριμένων στόχων ή εργασιών. Παραδοσιακά μια εφαρμογή βασιζόταν σε ένα μοναδικό σύστημα για την εκτέλεση της. Ακόμη και στο κλασσικό μοντέλο πελάτη – εξυπηρετητή η εκτέλεση ήταν αναγκαίο να επιτευχθεί είτε στον πελάτη είτε στον εξυπηρετητή στον οποίο ο πελάτης είχε πρόσβαση. Ωστόσο μια κατανεμημένη εφαρμογή είναι δυνατό να τρέχει ταυτόχρονα και στις δύο οντότητες.

Τα πολλαπλά μέρη μιας κατανεμημένης εφαρμογής μπορούν να βρίσκονται σε διαφορετικές εικονικές μηχανές οι οποίες είναι δυνατό συνυπάρχουν στο ίδιο υπολογιστικό σύστημα ή σε διαφορετικά. Η αρχιτεκτονική των κατανεμημένων εφαρμογών επιτρέπει στους χρήστες του δικτύου να έχουν πρόσβαση σε πληροφορίες, εφαρμογές και υπηρεσίες και να ανταλλάσουν πληροφορία με άλλους χρήστες μέσα από ένα ενιαίο συνεκτικό περιβάλλον χρήστη. Επιτρέπει την κατασκευή νέων υπηρεσιών και εφαρμογών καθώς επίσης παρέχει τη δυνατότητα ενσωμάτωσης ήδη υπαρχουσών εφαρμογών.

Ένα ολοκληρωμένο σύστημα το οποίο βασίζεται στην αρχιτεκτονική κατανεμημένων εφαρμογών, περιλαμβάνει δύο συνιστώσες. Την υποστήριξη των υπηρεσιών όπου παρέχονται ως μέρος της υποδομής και μια σειρά από συμβάσεις και πολιτικές όπου καθορίζουν την αλληλεπίδραση μεταξύ των παρεχόμενων υπηρεσιών. Μέσω των συμβάσεων αυτών καθίσταται δυνατή και η ενοποίηση τους σε ένα ενιαίο πλαίσιο.

Οι κατανεμημένες εφαρμογές μπορούν να χρησιμοποιηθούν σε διάφορους τύπους δικτύωσης, με περισσότερο συχνή την παρουσία τους σε δίκτυα πελάτη - εξυπηρετητή. Σε αυτόν το τύπο δικτύου, ο πελάτης προσπελαύνει προγράμματα και πληροφορία από τον διακομιστή. Ο διακομιστής επίσης είναι υπεύθυνος για την

επίτευξη όλων των απαιτούμενων διεργασιών με σκοπό την ομαλή εκτέλεση του προγράμματος.

Η χρήση κατανεμημένων εφαρμογών είναι γενικά επωφελής διότι καθίσταται εύκολη η ταυτόχρονη πρόσβαση και χρήση ενός προγράμματος. Ωστόσο αυτό θα μπορούσε να αποτελεί πρόβλημα σε περίπτωση αδυναμίας ενός διακομιστή. Ο διακομιστής είναι υπεύθυνος για την εκτέλεση και την απόδοση κάτι το οποίο έχει ως συνέπεια τη συχνή δημιουργία συνθηκών πίεσης. Αν ο διακομιστής δεν πληρεί τις αντίστοιχες δυνατότητες αυτό θα μπορούσε να οδηγήσει σε καθυστέρηση είτε περισσότερο σοβαρά προβλήματα σε υπολογιστικά περιβάλλοντα τα οποία προσπελαύνουν την εφαρμογή.

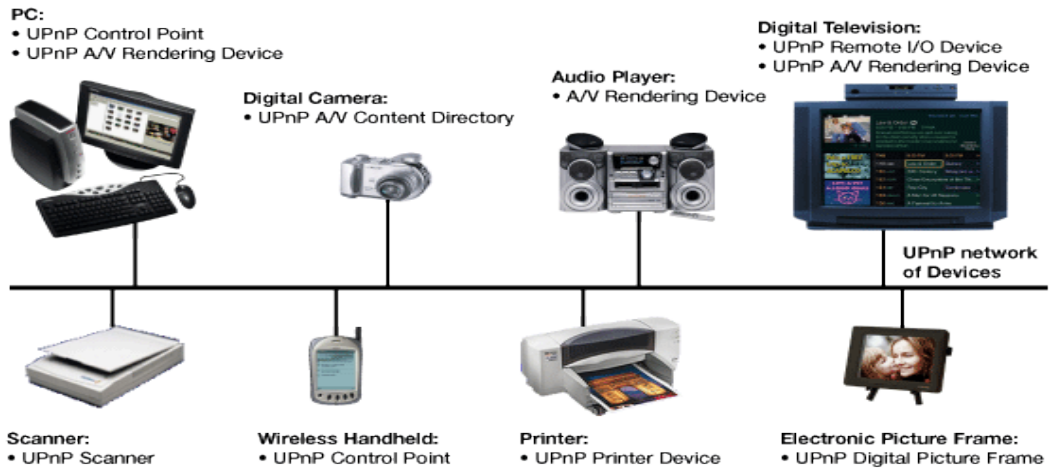
Η καθεμία από τις τεχνολογίες UPnP, Web Services και R-OSGi αποτελούν ένα ολοκληρωμένο μοντέλο ανάπτυξης κατανεμημένων εφαρμογών. Βασίζονται σε μια αρχιτεκτονική προδιαγραφών που αποσκοπούν στην κάλυψη των εκάστοτε απαιτήσεων σε ένα κατανεμημένο περιβάλλον, μέσω μιας διαφορετικής προσέγγισης που επικεντρώνει τόσο στο πεδίο υποδομής και υλοποίησης όσο και στο πεδίο εφαρμογής.

1. UPNP

1.1Εισαγωγή

Η τεχνολογία UPNP ορίζει μια αρχιτεκτονική η οποία παρέχει διεισδυτική και ομότιμη δικτύωση για όλους τους τύπους υπολογιστικών συστημάτων, είτε αυτά είναι απλές έξυπνες συσκευές είτε περισσότερο πολύπλοκα υπολογιστικά συστήματα.

Έχει σχεδιαστεί για να παρέχει εύχρηστη, ευέλικτη και βασισμένη στα πρότυπα διασύνδεση, σε ad-hoc και μη διαχειριζόμενα δίκτυα, σε δίκτυα που αφορούν το οικιακό περιβάλλον και μικρές επιχειρήσεις. Η UPnP τεχνολογία προσφέρει μία κατανεμημένη, ανοιχτή αρχιτεκτονική δικτύωσης που εκμεταλλεύεται τις ιδιότητες του TCP/IP και των τεχνολογιών του διαδικτύου προκειμένου να καταστήσει δυνατή την απρόσκοπτη μεταφορά δεδομένων ελέγχου και πληροφορίας μεταξύ των διασυνδεδεμένων στο ίδιο δίκτυο συσκευών.

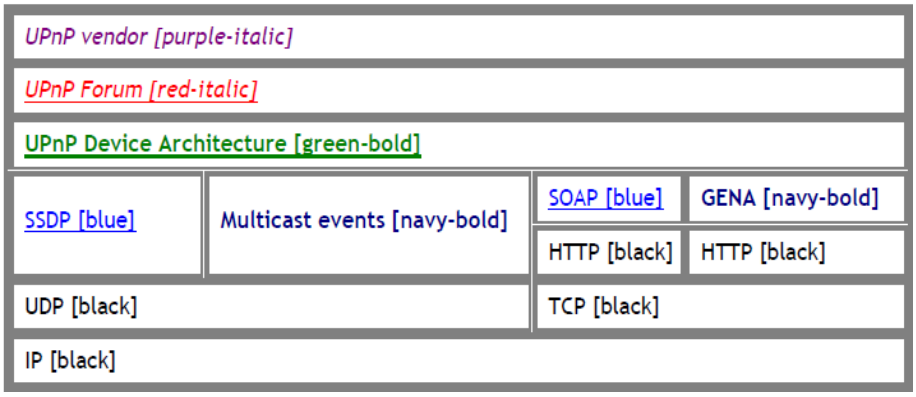


Σχήμα 1. UPNP Network

Η αρχιτεκτονική αυτής της τεχνολογίας σχεδιάστηκε για να υποστηρίζει ρύθμιση από μηδενική βάση, διαφανή δικτύωση και αυτόματη αναγνώριση για ένα μεγάλο εύρος κατηγοριών συσκευών από ένα ευρύτατο πεδίο προμηθευτών. Αυτό σημαίνει ότι μια συσκευή μπορεί να συνδεθεί δυναμικά με το δίκτυο να ανακτήσει διεύθυνση IP ,να ανακοινώσει τις δυνατότητές της καθώς και να πληροφορηθεί για την ύπαρξη και τις δυνατότητες των άλλων συνδεδεμένων συσκευών. Τέλος μια

UPNP συσκευή μπορεί να αποσυνδέεται από το δίκτυο αυτόματα, χωρίς προβλήματα και χωρίς να αφήνει οποιαδήποτε ανεπιθύμητη κατάσταση πίσω της.

Το UPNP βασίζεται σε πρωτόκολλα διαδικτύου και πρότυπα όπως τα IP, TCP, UDP, HTTP, και η XML. Με τη χρήση πρωτοκόλλων διαδικτύου και με την εφαρμογή της τεχνολογίας σε διαφορετικά φυσικά μέσα (ενσύρματα και ασύρματα), δίνεται η δυνατότητα διαλειτουργικότητας μεταξύ διαφόρων τύπων συσκευών προερχόμενες από διαφορετικούς προμηθευτές καθώς και η επίτευξη συνεργασίας με το διαδίκτυο και το intranet κάποιου ιδιωτικού χώρου. Μάλιστα το UPNP έχει ρητά σχεδιαστεί ώστε να είναι συμβατό με τις απαιτήσεις που υπάρχουν σε αυτά τα περιβάλλοντα.



Σχήμα 2. Στοίβα πρωτοκόλλου UPNP

Στο υψηλότερο επίπεδο της στοίβας πρωτοκόλλου UPNP τα μηνύματα περιέχουν μόνο ειδικές ανά προμηθευτή πληροφορίες σχετικά με τις συσκευές τους. Χαμηλότερα στην στοίβα πρωτοκόλλων, τα μηνύματα συμπληρώνονται από πληροφορία που προέρχεται από το UPNP Forum και την προτυποποιημένη UPNP αρχιτεκτονική. Τα μηνύματα από τα παραπάνω στρώματα εισάγονται σε πρωτόκολλα που σχετίζονται με την αρχιτεκτονική της τεχνολογίας UPNP, όπως είναι το SSDP (Simple Service Discovery Protocol) και το General Event Notification Architecture (GENA). Το SSDP μεταφέρεται unicast είτε multicast μέσω UDP ενώ το GENA μέσω HTTP. Τελικά, όλα τα παραπάνω μηνύματα ανταλλάσσονται κατά τη διάρκεια των UPNP λειτουργιών πάνω από το IP πρωτόκολλο.

Η έννοια Universal στην τεχνολογία UPNP σημαίνει πως δεν είναι απαραίτητη η χρήση drivers για τις συσκευές. Επίσης η εφαρμογή των συσκευών UPNP είναι

ανεξάρτητη από το τη γλώσσα προγραμματισμού και το λειτουργικό σύστημα που χρησιμοποιείται. Η αρχιτεκτονική UPNP δεν είναι περιοριστική για τον σχεδιασμό API για τις εφαρμογές. Ωστόσο οι προμηθευτές μπορούν να δημιουργήσουν APIs που ταιριάζουν στις ανάγκες του πελάτη. Σημαντικό χαρακτηριστικό του πρωτοκόλλου UPNP είναι το γεγονός ότι βασίζεται σε τεχνολογίες ομότιμων δικτύων (peer-to-peer). Η UPNP αρχιτεκτονική ορίζει δύο βασικές κατηγορίες συσκευών. Η πρώτη αποτελείται από τις συσκευές (Devices), ενώ η δεύτερη από τα σημεία ελέγχου (Control Points). Ουσιαστικά η διαχείριση των συσκευών (Devices) γίνεται μέσω των σημείων ελέγχων (Access Points). Καμία από τις δύο κατηγορίες δεν διαδραματίζει το ρόλο του εξυπηρετητή ή του πελάτη στο κλασσικό πλέον μοντέλο client – server. Αντίθετα, οι δύο συσκευές είναι εντελώς ομότιμες, ενώ με κατάλληλες τεχνικές είναι πολύ εύκολο να γίνει αντιστροφή των ρόλων τους. Επίσης στο ίδιο μηχάνημα μπορούν να λειτουργούν ταυτόχρονα τόσο η UPNP συσκευή όσο και το UPNP σημείο ελέγχου επιτρέποντας έτσι στις συνδρομητικές συσκευές να είναι ταυτόχρονα διαχειριστές αλλά και διαχειριζόμενοι. Οι συσκευές και των δύο κατηγοριών μπορούν να υλοποιηθούν σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των προσωπικών υπολογιστών και των ενσωματωμένων συστημάτων.

Το UPNP πρωτόκολλο περιλαμβάνει τις ακόλουθες έξι βασικές λειτουργίες:

Addressing Κατά τη φάση αυτή μία συσκευή είτε ένα σημείο ελέγχου πρέπει να λάβει μία IP διεύθυνση για να εισέλθει στο τοπικό IP δίκτυο.

Discovery Τα σημεία ελέγχου αναζητούν τις UPnP συσκευές αλλά και τις υπηρεσίες που αυτές προσφέρουν. Αντίστοιχα, οι UPnP συσκευές ανακοινώνουν την παρουσία τους κάθε φορά που εισέρχονται στο τοπικό δίκτυο.

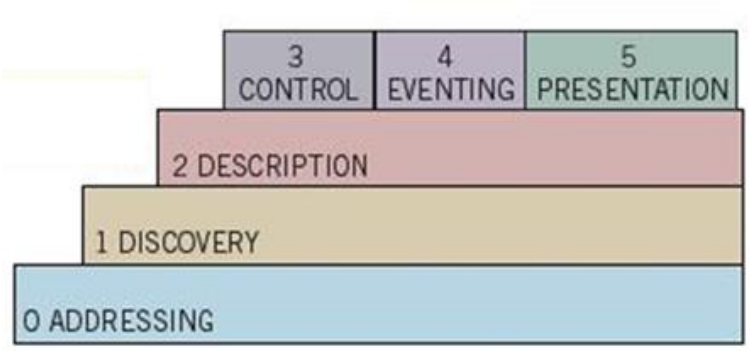
Description Μόλις ένα σημείο ελέγχου ανακαλύψει στη φάση discovery μια συσκευή με υπηρεσίες που το ενδιαφέρει, ζητά από τη συσκευή να του αποστείλει μια λεπτομερή περιγραφή. Η περιγραφή αυτή συνίσταται στις λειτουργικές μονάδες της συσκευής και στις υπηρεσίες ελέγχου και διαχείρισης που προσφέρει η κάθε μονάδα. Έτσι εκτός από υπηρεσίες, οι πληροφορίες αυτές μπορεί να αφορούν και άλλες ενσωματωμένες συσκευές τις οποίες περιέχει ή άλλες πληροφορίες κατάστασης.

Control Αφότου έχει ανακτήσει μια περιγραφή της συσκευής, ένα σημείο ελέγχου μπορεί να επικαλεστεί ενέργειες προς τις υπηρεσίες και εν συνεχεία να λαμβάνει απαντήσεις που υποδεικνύουν το αποτέλεσμα της κάθε ενέργειας.

Eventing Τα σημεία ελέγχου ενημερώνονται για την αλλαγή κατάστασης των υπηρεσιών για τις οποίες ενδιαφέρονται. Η φάση αυτή λοιπόν επιτρέπει στο

σημείο ελέγχου να διατηρείται σε συγχρονισμό με την κατάσταση των υπηρεσιών της συσκευής την οποία ελέγχει.

Presentation Κατά τη φάση της παρουσίασης η UPNP συσκευή προσφέρει στο χρήστη μια διεπαφή ή αλλιώς το γραφικό περιβάλλον με όλες τις ελεγχόμενες υπηρεσίες της.



Σχήμα 3. Οι έξι βασικές λειτουργίες του UPNP

1.2 Addressing

Η πρώτη φάση λειτουργίας (Βήμα 0) στην τεχνολογία UPNP είναι η διαδικασία addressing. Μέσω της διαδικασίας αυτής οι συσκευές και τα σημεία ελέγχου αποκτούν IP στο δίκτυο. Η διαδικασία addressing επιτρέπει την ανίχνευση *discovery* (βήμα 1) των συσκευών ενδιαφέροντος από τα σημεία ελέγχου, την περιγραφή *description* (βήμα 2) κατά την οποία τα σημεία ελέγχου λαμβάνουν μια λεπτομερή περιγραφή των δυνατοτήτων των συσκευών, τον έλεγχο *control* (βήμα 3) όπου τα σημεία ελέγχου στέλνουν εντολές στις συσκευές, την λειτουργία *eventing* (βήμα 4) όπου τα σημεία ελέγχου ακούν τις αλλαγές των συσκευών και τέλος την λειτουργία presentation (βήμα 5) όπου τα σημεία ελέγχου παρέχουν μια διεπαφή ή αλλιώς το γραφικό περιβάλλον της συσκευής προς το χρήστη.

1.2.1Περίπτωση χρήσης AUTO-IP

Κάθε συσκευή ή σημείο ελέγχου όπου δεν υλοποιούν από μόνα τους έναν DHCP εξυπηρετητή, θα πρέπει να έχουν έναν DHCP (Dynamic Host Configuration Protocol) πελάτη όπου θα ψάχνει για DHCP εξυπηρετητή όταν συνδέονται για πρώτη φορά στο δίκτυο.

Μια συσκευή ή ένα σημείο ελέγχου όπου υποστηρίζει τη λειτουργία αυτόματης ανάθεσης IP διεύθυνσης (AUTO-IP), ξεκινά τη δυναμική ανάθεση IP με την αίτηση μιας διεύθυνσης μέσω DHCP. Το παραπάνω επιτυγχάνεται με την αποστολή ενός μηνύματος DHCPDISCOVER. Εάν ληφθεί έγκαιρα ένα μήνυμα απόδοσης διεύθυνσης DHCPOFFER ολοκληρώνεται η διαδικασία δυναμικής ανάθεσης IP ενώ εάν κανένα μήνυμα DHCPOFFER δεν παραληφθεί, τότε το σημείο ελέγχου ή η συσκευή θα πρέπει αυτόματα να διαμορφώσουν μια διεύθυνση IP με χρήση της αυτόματης IP διευθυνσιολόγησης (AUTO-IP). Η συγκεκριμένη μέθοδος απόδοσης διευθύνσεων καθιστά τη συσκευή ικανή να μετακινείται εύκολα ανάμεσα σε διαχειριζόμενα ή μη δίκτυα.

1.2.2 Επιλογή διεύθυνσης μέσω αυτόματης ανάθεσης (AUTO-IP)

Η αυτόματη διαμόρφωση διεύθυνσης μέσω της διαδικασίας AUTO-IP από τη συσκευή ή το σημείο ελέγχου γίνεται με τη χρήση αλγόριθμου. Η επιλεγμένη διεύθυνση θα πρέπει να δοκιμαστεί ώστε να επιβεβαιωθεί πως δεν βρίσκεται ήδη σε χρήση. Σε περίπτωση που η διεύθυνση χρησιμοποιείται από άλλη συσκευή ή σημείο ελέγχου στο δίκτυο, θα πρέπει να επιλεγεί και να δοκιμαστεί άλλη διεύθυνση. Η επιλογή διεύθυνσης θα πρέπει να είναι τυχαία έτσι ώστε να αποφευχθεί η σύγκρουση διευθύνσεων όταν ταυτόχρονα άλλες συσκευές ή σημεία ελέγχου προσπαθούν να δεσμεύσουν κάποια IP. Η συσκευή ή το σημείο ελέγχου επιλέγει μια διεύθυνση κάνοντας χρήση ενός ψευδοτυχαίου αλγορίθμου κατανεμημένου σε όλο το διάστημα διευθύνσεων με εύρος 169.254.1.0 έως 169.254.254.255, έτσι ώστε να ελαχιστοποιηθεί η πιθανότητα τα σημεία ελέγχου ή οι συσκευές που εισέρχονται στο δίκτυο την ίδια στιγμή να επιλέξουν την ίδια διεύθυνση καθώς επίσης να υπάρχει η δυνατότητα να γίνει άμεσα επιλογή νέας διεύθυνσης όταν μια σύγκρουση ανιχνευθεί.

1.2.3 Έλεγχος της επιλεγμένης διεύθυνσης

Για τον έλεγχο της επιλεγμένης διεύθυνσης η συσκευή ή το σημείο ελέγχου θα πρέπει να χρησιμοποιήσουν έναν έλεγχο με βάση το πρωτόκολλο Address Resolution Protocol (ARP). Ο έλεγχος ARP είναι η αποστολή ενός ARP αιτήματος με διεύθυνση αποστολέα την hardware διεύθυνση της συσκευής ή του σημείου ελέγχου ενώ η διεύθυνση IP του αποστολέα τίθεται σε μηδενική. Έτσι πριν την έναρξη χρήσης μιας διεύθυνσης IP, η οποία θα ληφθεί είτε χειροκίνητα είτε από DCHP είτε με οποιοδήποτε μέσο, η συσκευή ή το σημείο ελέγχου με τον ARP έλεγχο δοκιμάζουν εάν η διεύθυνση χρησιμοποιείται ήδη στο δίκτυο, ώστε να αποφευχθεί η σύγκρουση κάνοντας broadcast ARP πακέτα.

Η συσκευή ή το σημεία ελέγχου θα πρέπει να μπορούν να ακούσουν στις απαντήσεις στον ARP έλεγχο ή σε άλλους ARP ελέγχους που αφορούν στην ίδια IP διεύθυνση. Μόλις ληφθεί κάποιο από τα πακέτα αυτά, θα πρέπει να διαπιστώσουν αν η διεύθυνση είναι σε χρήση και αν ναι, να δοκιμάσουν κάποια άλλη. Για μεγαλύτερη βεβαιότητα και τον αποκλεισμό του ενδεχομένου σύγκρουσης των IP, ο ARP έλεγχος μπορεί αν επαναληφθεί. Μια ιδανική περίπτωση είναι ο έλεγχος να σταλεί τέσσερις φορές σε διάστημα δύο δευτερολέπτων. Έπειτα από μια επιτυχή ανάθεση τοπικής IP, ή συσκευή η το σημείο ελέγχου θα πρέπει να αποστείλουν δύο ARP πακέτα σε διάστημα δύο δευτερολέπτων τα οποία σαν διεύθυνση αποστολέα θα έχουν πλέον την IP διεύθυνση. Η ενέργεια αυτή έχει ως σκοπό την επιβεβαίωση πως όλοι οι host στο δίκτυο δεν διατηρούν κάποια παλαιά καταχώρηση από hosts που έκαναν στο παρελθόν χρήση της ίδιας IP.

Επιπλέον για την αποφυγή συγκρούσεων των διευθύνσεων και αύξηση της σταθερότητας στη χρήση τους, οι συσκευές και τα σημεία ελέγχου που έχουν καταχωρήσει το παραπάνω αρχείο ελέγχου, μπορούν να χρησιμοποιήσουν και σε επόμενες λειτουργίες την ίδια IP ως πρώτη υποψήφια. Ο έλεγχος σύγκρουσης διευθύνσεων δε περιορίζεται μόνο στη φάση που η συσκευή και το σημείο ελέγχου στέλνουν ARP πακέτα και ακούν σε απαντήσεις τους, αλλά καθ' όλη τη διάρκεια χρήσης μιας τοπικής διεύθυνσης IP. Οποιαδήποτε στιγμή η συσκευή ή το σημείο ελέγχου λάβουν ένα ARP πακέτο με διεύθυνση αποστολέα την IP τους διεύθυνση αλλά η διεύθυνση υλικού αποστολέα δεν ταιριάζει με τη δική τους, αυτό θα πρέπει να θεωρηθεί σαν σύγκρουση και θα πρέπει να ενεργήσουν ως εξής:

α) Άμεση διαμόρφωση μιας IP διεύθυνσης με τους τρόπους που περιγράφησαν παραπάνω.

β) Αν το σημείο ελέγχου ή η συσκευή δεν έχει εντοπίσει πρόσφατα επιπλέον πακέτα σύγκρουσης ARP (μέσα σε 10 sec για παράδειγμα) και επιθυμεί να διατηρήσει την ίδια IP διότι μπορεί να έχει ενεργές TCP συνδέσεις είτε για άλλους παρόμοιους λόγους, στην περίπτωση αυτή ίσως επιλέξει να προσπαθήσει να διατηρήσει τη διεύθυνση καταγράφοντας στο χρόνο παραλαβής του ARP πακέτου σύγκρουσης και αποστέλλοντας broadcast ένα μοναδικό σήμα ARP το οποίο θα περιλαμβάνει την διεύθυνση IP και τη διεύθυνση υλικού σαν διευθύνσεις προέλευσης του σήματος ελέγχου. Ωστόσο αν παραληφθεί ακόμη ένα πακέτο σύγκρουσης ARP σε σύντομο χρονικό διάστημα, η συσκευή ή το σημείο ελέγχου θα πρέπει να διαμορφώσει μια νέα διεύθυνση AUTO-IP σύμφωνα με την περιγραφή που έγινε παραπάνω.
Οι συσκευές ή τα σημεία ελέγχου θα πρέπει να αποκρίνονται σύμφωνα με τους παραπάνω τρόπους στα ARP πακέτα σύγκρουσης και να μην αγνοείται κανένα από αυτά. Επίσης αν μια νέα IP επιλεγεί, θα πρέπει να ανακληθούν όλες οι παλαιές

ανακοινώσεις και να γίνει γνωστή η νέα διεύθυνση στις υπόλοιπες συσκευές του δικτύου. Έπειτα από ένα επιτυχημένο καθορισμό μια AUTO-IP διεύθυνσης όλα τα επόμενα ARP πακέτα είτε είναι απαντήσεις είτε είναι αιτήματα θα πρέπει να μεταδίδονται μέσω broadcast ώστε να είναι δυνατή η έγκαιρη ανίχνευση συγκρούσεων διευθύνσεων.

1.2.4 Περιοδικός έλεγχος διαθέσιμης δυναμικής διεύθυνσης

Μια συσκευή ή ένα σημείο ελέγχου που διαμορφώνει αυτόματα μια διεύθυνση, θα πρέπει να έχει τη δυνατότητα να ελέγχει περιοδικά την ύπαρξη ενός DHCP server. Αυτό επιτυγχάνεται με την αποστολή DHCPDISCOVER μηνυμάτων. Η συχνότητα κατά την οποία διενεργείται ο έλεγχος αυτός είναι καθαρά εξαρτώμενη από την εκάστοτε εφαρμογή. Ωστόσο ένα περιοδικός έλεγχος κάθε πέντε λεπτά θα διατηρούσε μια ισορροπία μεταξύ του απαιτούμενου εύρους ζώνης για το δίκτυο και της διατήρησης της συνδεσιμότητας.

Μόλις ένα μήνυμα DHCPOFFER παραληφθεί, η συσκευή ή το σημείο ελέγχου θα πρέπει να δεσμεύσει μια δυναμική IP και να εγκαταλήψει την διαδικασία αυτόματης διαμόρφωσης IP AUTO-IP. Κατά την ανάκτηση μιας νέας IP η συσκευή θα πρέπει να ακυρώσει οποιαδήποτε ανακοίνωση προς το δίκτυο η οποία αφορά την παλαιά διεύθυνση και να ξεκινήσει ανακοινώσεις με τη νέα.

1.2.5 Ονοματοδοσία συσκευών και αλληλεπίδραση DNS

Όταν μια συσκευή έχει μια έγκυρη IP διεύθυνση, μπορεί να συνδεθεί και να λειτουργήσει στο δίκτυο μέσω αυτής. Για την αναγνώριση και τον εντοπισμό μιας συσκευής στο δίκτυο από τον τελικό χρήστη, είναι δυνατή η χρήση ενός περισσότερο φιλικού τρόπου αναπαράστασης της συσκευής απ' ότι μια IP διεύθυνση. Μια συσκευή μπορεί να παρέχει ένα όνομα host σε έναν DHCP server. Στην περίπτωση αυτή, θα πρέπει να εξασφαλίζει πως το όνομα host είναι μοναδικό είτε θα πρέπει να παρέχει τη δυνατότητα προς το χρήστη να αλλάξει το αλλάξει.

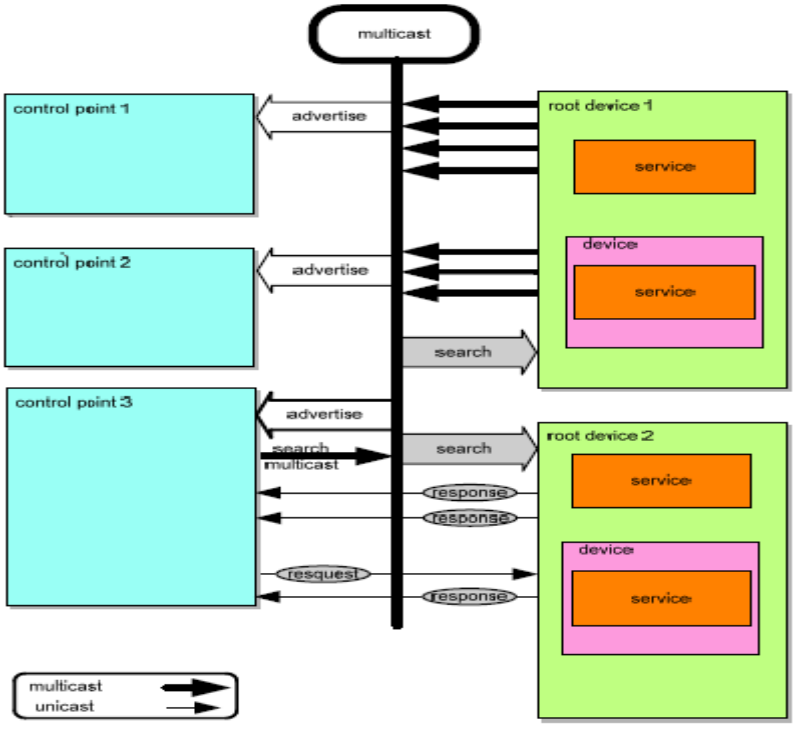
Επιπροσθέτως, τα ονόματα είναι περισσότερο στατικά σε σύγκριση με τις διευθύνσεις IP. Έτσι σε περίπτωση που η IP μιας συσκευής έχει αλλάξει, ένας client που αναφέρεται σε αυτή τη συσκευή μπορεί να την αναζητήσει και πάλι με το ίδιο όνομα. Η αντιστοίχηση των ονομάτων DNS των συσκευών με τις IP διευθύνσεις, μπορεί να εισαχθεί στη βάση δεδομένων του DNS είτε χειροκίνητα είτε δυναμικά σύμφωνα με το πρότυπο RFC 2136 (Dynamic Updates in the Domain Name System - DNS UPDATE). Ενώ οι συσκευές οι οποίες υποστηρίζουν τις δυναμικές DNS ανανεώσεις μπορούν να καταχωρούν τα DNS αρχεία τους άμεσα στους DNS servers,

είναι επίσης δυνατό να ενημερώνεται ένας κεντρικός υπολογιστής DHCP για να καταχωρεί τα αρχεία DNS εκ μέρους αυτών των DHCP clients.

1.3 Discovery

Η διαδικασία Discovery είναι το βήμα 1 στην τεχνολογία UPnP. Η διαδικασία αυτή ακολουθεί έπειτα από την διαδικασία addressing (βήμα 0) όπου οι συσκευές και τα σημεία ελέγχου έχουν αποκτήσει IP στο δίκτυο. Κατά τη φάση discovery του UPnP τα σημεία ελέγχου εντοπίζουν τις συσκευές ενδιαφέροντος ενώ επίσης ενεργοποιείται η διαδικασία description (βήμα 2) όπου τα σημεία ελέγχου συλλέγουν πληροφορίες για τις ιδιότητες και τα χαρακτηριστικά των συσκευών που είναι συνδεδεμένες στο δίκτυο.

Έτσι κατά τη διαδικασία discovery όταν μια συσκευή προστεθεί στο δίκτυο, η τεχνολογία UPnP της επιτρέπει να κάνει γνωστά τα χαρακτηριστικά των υπηρεσιών της προς τα σημεία ελέγχου. Ομοίως όταν ένα σημείο ελέγχου προστεθεί στο δίκτυο το UPnP του επιτρέπει να αναζητήσει συσκευές ενδιαφέροντος. Η ανταλλαγή της πληροφορίας και στις δύο περιπτώσεις γίνεται από κάποιο μήνυμα discovery το οποία θα περιλαμβάνει βασικά χαρακτηριστικά για τη συσκευή είτε για τις υπηρεσίες της όπως: τον τύπο, ένα μοναδικό αναγνωριστικό είτε ένα δείκτη σε πιο λεπτομερείς πληροφορία και παραμέτρους που προσδιορίζουν την τρέχουσα κατάσταση της συσκευής.



Σχήμα 4. Αρχιτεκτονική Discovery

Όταν μια συσκευή γνωρίζει πως συνδέθηκε πρόσφατα στο δίκτυο, θα πρέπει να μεταδώσει με μέθοδο multicast ένα αριθμό discovery μηνυμάτων ώστε να διαφημίσει την ίδια, τις ενσωματωμένες συσκευές της καθώς και τις υπηρεσίες της. Τα ενδιαφερόμενα σημεία ελέγχου θα ακούσουν στην multicast διεύθυνση για τις ειδοποιήσεις πως νέες ιδιότητες είναι διαθέσιμες. Μια πολυκατευθυνόμενη (multi-homed) συσκευή πρέπει να κάνει multicast τα discovery μηνύματα σε όλες τις UPnP διεπαφές. Ένα πολυκατευθυνόμενο σημείο ελέγχου μπορεί να ακούσει στη multicast διεύθυνση για μία , μερικές ή όλες τις ενεργές UPnP διεπαφές.
Όταν ένα νέο σημείο ελέγχου προστεθεί στο δίκτυο μπορεί να κάνει multicast ένα discovery μήνυμα αναζητώντας συσκευές και υπηρεσίες ενδιαφέροντος. Όλες οι συσκευές θα πρέπει να ακούσουν στη multicast διεύθυνση και να απαντήσουν σε περίπτωση που κάποια από τις root συσκευές, τις ενσωματωμένες είτε τις υπηρεσίες της ταιριάζουν στα κριτήρια αναζήτησης του discovery μηνύματος. Επιπροσθέτως ένα σημείο ελέγχου μπορεί να εκπέμψει unicast ένα discovery μήνυμα σε μια συγκεκριμένη IP διεύθυνση στο port 1900(συνήθως) είτε σε port που έχει καθοριστεί από την τεχνολογία UPnP αναζητώντας μια UPnP συσκευή ή υπηρεσία στη συγκεκριμένη IP διεύθυνση. Η ενέργεια αυτή δηλώνει πως το σημείο ελέγχου γνωρίζει πως η συσκευή σε αυτήν τη διεύθυνση IP είναι μια UPNP 1.1 συσκευή που ακούει στο κατάλληλο port. Ένα σημείο ελέγχου μπορεί να χρησιμοποιήσει unicast αναζήτηση για έναν αριθμό εφαρμογών. Μια unicast αναζήτηση μπορεί άμεσα να παρέχει πληροφορίες και χαρακτηριστικά μιας συγκεκριμένης συσκευής όπως το UUID και το URL. Μια συσκευή θα πρέπει να μπορεί να ακούσει στα unicast μηνύματα αναζήτησης στο καθορισμένο port και να απαντήσει στην περίπτωση που οποιαδήποτε root συσκευή είτε ενσωματωμένη συσκευή είτε υπηρεσία ταιριάζει στα κριτήρια του discovery μηνύματος.

Ένα multi-homed σημείο ελέγχου μπορεί να εκπέμψει multicast μηνύματα discovery σε μία, μερικές ή όλες τις UPnP ενεργές διεπαφές. Οι multi-homed συσκευές πρέπει να μπορούν ακούσουν στη multicast διεύθυνση σε όλες τις διεπαφές για τα μηνύματα αυτά. Οι multi-homed συσκευές επίσης θα πρέπει να είναι σε θέση να ακούσουν και τα unicast εισερχόμενα μηνύματα στο port που έχει οριστεί και να απαντούν σε αυτά εφόσον τις αφορούν.

Επομένως ένα σημείο ελέγχου μπορεί να πληροφορηθεί για μία συσκευή που παρουσιάζει ενδιαφέρον είτε επειδή αυτή έστειλε μηνύματα διαφημίζοντας αυτήν και τις υπηρεσίες , είτε διότι ανταποκρίθηκε θετικά σε ένα μήνυμα ανίχνευσης του σημείου ελέγχου. Και στις δύο περιπτώσεις, αν ένα σημείο ελέγχου ενδιαφέρεται να μάθει περισσότερα για μία συσκευή και τις υπηρεσίες της, μπορεί χρησιμοποιώντας την πληροφορία του μηνύματος ανίχνευσης, να στείλει ένα μήνυμα εξέτασης περιγραφής. Η λειτουργία της περιγραφής description (βήμα 2) εξηγεί τα μηνύματα description λεπτομερώς.

Όταν μια συσκευή αποσυνδεθεί από το δίκτυο θα πρέπει αν είναι δυνατό να κάνει multicast ένα αριθμό discovery μηνυμάτων ανακαλώντας τις προηγούμενες ανακοινώσεις δηλώνοντας πως οι root συσκευές, οι ενσωματωμένες συσκευές και οι υπηρεσίες δεν είναι πλέον διαθέσιμες. Όταν η διεύθυνση IP μιας συσκευής αλλάξει, η συσκευή θα πρέπει να ανακαλέσει οποιαδήποτε προηγούμενη ανακοίνωση και να διαφημίσει τη νέα IP διεύθυνση.

Όταν μια multi-homed συσκευή είναι πλέον μη διαθέσιμη σε κάποιες από τις UPnP διεπαφές της, θα πρέπει εφόσον είναι δυνατό να κάνει multicast ένα αριθμό discovery μηνυμάτων ανακαλώντας τις προηγούμενες ανακοινώσεις στις εξαρτώμενες διεπαφές, δηλώνοντας πως οι root συσκευές, οι ενσωματωμένες συσκευές και οι υπηρεσίες τους δεν είναι πια διαθέσιμες στις διεπαφές αυτές. Εάν οι συσκευή παραμένει διαθέσιμη σε οποιεσδήποτε άλλες διεπαφές της, δε θα πρέπει στις μη επηρεασμένες διεπαφές να κάνει μηνύματα discovery.

Όταν μια συσκευή είναι διαθέσιμη σε μια νέα επιπλέον διεπαφή θα πρέπει να αυξήσει την τιμή του πεδίου BOOTID.UPNP.ORG και να κάνει multicast ένα αριθμό discovery μηνυμάτων ανανέωσης στις υπάρχουσες διεπαφές ανακοινώνοντας τη νέα τιμή BOOTID.UPNP.ORG. Έπειτα από την αποστολή όλων των μηνυμάτων ανανέωσης θα πρέπει να κάνει multicast σε όλες τις διεπαφές (υπάρχουσες και νέες) με την νέα τιμή BOOTID.UPNP.ORG.

Ομοίως όταν μια από τις IP διευθύνσεις σε μια multi-homed συσκευή αλλάξει θα πρέπει να ανακαλέσει οποιεσδήποτε προηγούμενες ανακοινώσεις με την προηγούμενη IP διεύθυνση, θα πρέπει να αυξήσει την τιμή του BOOTID.UPNP.ORG πεδίου και να κάνει multicast ένα αριθμό μηνυμάτων αναέωσης στις υπάρχουσες UPnP διεπαφές ανακοινώνοντας τη νέα τιμή BOOTID. Έπειτα από την αποστολή των μυνημάτων ανανέωσης θα πρέπει να κάνει multicast ένα αριθμό discovery μηνυμάτων σε όλες τις διεπαφές παλαιές και νέες με τη νέα τιμή του πεδίου BOOTID.UPNP.ORG.

Τέλος όταν μια συσκευή χάσει τη σύνδεσή της με μία από τις UPnP διεπαφές της και στη συνέχεια την επανακτήσει, θα πρέπει να αυξήσει την τιμή του πεδίου BOOTID.UPNP.ORG και να κάνει multicast ένα αριθμό μηνυμάτων ανανέωσης στις μη επηρεασμένες διεπαφές, ανακοινώνοντας τη νέα τιμή BOOTID. Έπειτα από την αποστολή όλων των μηνυμάτων ανανέωσης θα πρέπει να κάνει multicast ένα αριθμό μηνυμάτων discovery σε όλες τις διεπαφές (επηρεασμένες και μη) με τη νέα τιμή πεδίου BOOTID.UPNP.ORG.

Για τον περιορισμό συμφόρησης του δικτύου ο χρόνος time-to-live (TTL) κάθε IP πακέτου για καθένα από τα multicast μηνύματα θα πρέπει να τεθεί στα 2

δευτερόλεπτα και να είναι διαμορφώσιμος. Όταν ο χρόνος TTL είναι μεγαλύτερος του 1, είναι πιθανό τα multicast μηνύματα να περάσουν σε πολλούς δρομολογητές. Ωστόσο τα σημεία ελέγχου και οι συσκευές που δε χρησιμοποιούν AUTO-IP διευθύνσεις θα πρέπει να στείλουν IGMP μηνύματα (IGMP είναι το πρωτόκολλο που χρησιμοποιείται από το πρωτόκολλο IPv4 για να εξασφαλιστεί ότι η εισερχόμενη multicast κυκλοφορία διαβιβάζεται από έναν δρομολογητή στο τμήμα δικτύων με το οποίο ο δρομολογητής είναι συνδεμένος) ώστε οι δρομολογητές να προωθήσουν τα μηνύματα πίσω σε αυτούς (αυτό δεν είναι απαραίτητο όταν χρησιμοποιούνται αυτόματες διευθύνσεις IP, διότι στην περίπτωση αυτή τα πακέτα δε θα προωθηθούν από τους δρομολογητές).

1.3.1 Εκδόσεις UPnP

Η διαδικασία Discovery παίζει σημαντικό ρόλο στη λειτουργικότητα των συσκευών και των σημείων ελέγχου που χρησιμοποιούν διαφορετικές εκδόσεις της UPnP τεχνολογίας. Η UPnP αρχιτεκτονική των συσκευών πλαισιώνεται από πρωτεύοντες και δευτερεύουσες οι οποίες ονομάζονται major και minor και χαρακτηρίζονται από έναν αριθμό (για παράδειγμα η έκδοση 2.10 είναι πιο πρόσφατη από την 2.2). Οι εξελιγμένες δευτερεύουσες (minor) εκδόσεις πρέπει να είναι συμβατές με τις προηγούμενες δευτερεύουσες εκδόσεις μιας πρωτεύουσας (major) έκδοσης. Οι εξελιγμένες πρωτεύουσες εκδόσεις δεν απαιτείται να είναι συμβατές με τις προηγούμενες. Οι πληροφορίες για την έκδοση μιας συσκευής ή σημείου ελέγχου, περιλαμβάνονται στα μηνύματα discovery και description. Τα μηνύματα discovery περιλαμβάνουν την έκδοση UPnP που οι συσκευές και τα σημεία ελέγχου υποστηρίζουν στα πεδία επικεφαλίδων (SERVER και USER-AGENT), οι υποστηριζόμενες εκδόσεις των τύπων συσκευών και υπηρεσιών περιλαμβάνονται επίσης σε σχετικά discovery μηνύματα. Επιπροσθέτως, τα description μηνύματα περιλαμβάνουν την ίδια πληροφορία. Τα πεδία επικεφαλίδων SERVER και USER-AGENT χρησιμοποιούνται επίσης στις λειτουργίες control και eventing ώστε να δοθεί η πληροφορία για το ποιες είναι οι υποστηριζόμενες εκδόσεις των σημείων ελέγχου και των συσκευών.

1.3.2 Μορφοποίηση Μηνύματος SSDP

Το πρωτόκολλο SSDP χρησιμοποιεί UDP αντί του TCP και έχει τους δικούς του κανόνες επεξεργασίας. Όλα τα μηνύματα SSDP θα πρέπει να έχουν μια γραμμή έναρξης και μία λίστα από πεδία επικεφαλίδας. Δεν θα πρέπει να περιλαμβάνουν σώμα μηνύματος, καθώς αν ληφθεί μήνυμα με τη μορφή αυτή πιθανό να αγνοηθεί.

1.3.2.1 SSDP Γραμμή Έναρξης (Start Line)

Η γραμμή έναρξης σε ένα μήνυμα SSDP θα πρέπει έχει την παρακάτω μορφή:

NOTIFY * HTTP/1.1\r\n
M-SEARCH * HTTP/1.1\r\n
HTTP/1.1 200 OK\r\n

1.3.2.2 Πεδία Επικεφαλίδας Μηνύματος SSDP

Κάθε πεδίο επικεφαλίδας μηνύματος αποτελείται από ένα πεδίο ονόματος ακολουθούμενο από (":"), ακολουθούμενο επίσης από ένα πεδίο τιμών. Ακολουθεί ένα παράδειγμα επικεφαλίδας SSDP:

HOST: 239.255.255.250:1900

1.3.2.3 Επεκτάσεις πεδίων επικεφαλίδας SSDP

Οι ερευνητικές ομάδες και οι προμηθευτές UPnP έχουν τη δυνατότητα να επεκτείνουν τα μηνύματα SSDP με επιπρόσθετα πεδία επικεφαλίδας. Το UPnP Forum επίσης μπορεί να καθορίσει τα επιπρόσθετα πεδία επικεφαλίδας. Για παράδειγμα παρακάτω στην παράγραφο Advertisement ορίζονται τα πεδία επικεφαλίδων (BOOTID.UPNP.ORG, CONFIGID.UPNP.ORG, NEXTBOOTID.UPNP.ORG, και SEARCHPORT.UPNP.ORG).
Η μορφοποίηση των καθορισμένων από τους προμηθευτές πεδίων επικεφαλίδων θα πρέπει να είναι η εξής;

field-name = token “.” domain-name

Όπου domain name θα πρέπει να είναι το όνομα του προμηθευτή (Vendor Domain Name). Κάποια παραδείγματα πεδίων επικεφαλίδων καθορισμένα από τους προμηθευτές είναι τα παρακάτω:

myheader.philips.com: “some value”
myheader.sony.com: “other value”

1.3.2.4 Μορφοποίηση UUID και προτεινόμενοι αλγόριθμοι παραγωγής

Οι συσκευές UPnP 1.1 θα πρέπει να ακολουθούν την παρακάτω μορφοποίηση UUIDs. Ωστόσο τα σημεία ελέγχου UPnP 1.1 θα πρέπει να είναι ικανά να δέχονται UUIDs που δεν ακολουθούν τη συγκεκριμένη μορφοποίηση.

Παράδειγμα UUID:
"2fac1234-31f8-11b4-a222-08002b34c003"

Τα UUID μπορούν να παραχθούν από οποιανδήποτε αλγόριθμο αρκεί αυτός να τηρεί τις παρακάτω προϋποθέσεις:
1. Πρέπει να είναι δύσκολο να παραχθεί αντίγραφο του UUID από άλλη πηγή.
2. Να δημιουργεί UUID αριθμούς σε μορφή των 128 bit.
3. Θα πρέπει να παραμένει σταθερό στην πάροδο του χρόνου.

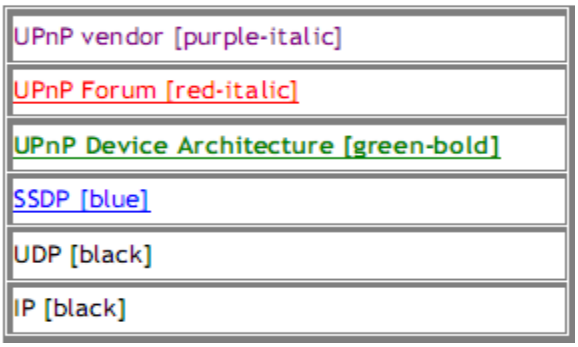
1.3.3 Advertisement (διαφήμιση συσκευής)

Όταν μια συσκευή προστεθεί στο δίκτυο, διαφημίζει τις υπηρεσίες της στα σημεία ελέγχου. Αυτό το επιτυγχάνει κάνοντας multicast μηνύματα discovery σε μια καθορισμένη διεύθυνση και Port (239.255.255.250:1900). Τα σημεία ελέγχου ακούν στο Port αυτό ώστε να εντοπίσουν πότε υπάρχουν νέες δυνατότητες διαθέσιμες στο δίκτυο. Για μια πλήρη γνωστοποίηση των δυνατοτήτων της η συσκευή, κάνει multicast αντίστοιχα μηνύματα discovery για τις root συσκευές τις ενσωματωμένες και τις υπηρεσίες τους. Τα μηνύματα θα πρέπει να έχουν διάρκεια όση είναι η διάρκεια της διαδικασίας της γνωστοποίησης (advertising). Αν η συσκευή παραμένει διαθέσιμη, τότε οι ανακοινώσεις θα πρέπει να αποσταλούν και πάλι με νέα διάρκεια. Αν η συσκευή τεθεί μη διαθέσιμη θα πρέπει να ακυρώσει την διαδικασία advertisement αλλά αν η συσκευή δεν είναι σε θέση να το κάνει, τότε οι ανακοινώσεις θα λήξουν από μόνες τους. Αν η συσκευή τεθεί μη διαθέσιμη σε κάποιες και όχι όλες τις UPnP διεπαφές της, τότε θα πρέπει να ακυρώσει τη διαδικασία advertisements στις εξαρτώμενες. Βέβαια σε περίπτωση που η συσκευή δεν είναι σε θέση να το κάνει αυτό θα γίνει με τη λήξη των ανακοινώσεων. Σημαντικό είναι να αναφερθεί πως τα μηνύματα που χρησιμοποιούν τα ακόλουθα πεδίαεπικεφαλίδων: BOOTID.UPNP.ORG, NEXTBOOTID.UPNP.ORG, CONFIGID.UPNP.ORG, SEARCHPORT.UPNP.ORG, ορίζονται παρακάτω. Το πεδίο τιμών της επικεφαλίδας BOOTID.UPNP.ORG θα πρέπει να αυξάνεται κάθε φορά που η συσκευή επανασυνδέεται στο δίκτυο και στέλνει ανακοίνωση αρχικοποίησης (reboot) ή προσθέτει μια νέα UPnP διεπαφή. Σε περίπτωση που η συσκευή ρητά ανακοινώσει μια αλλαγή στο πεδίο τιμών του BOOTID.UPNP.ORG χρησιμοποιώντας ένα SSDP μήνυμα, όσο διάστημα παραμένει διαθέσιμη στο δίκτυο, θα πρέπει να χρησιμοποιείται το ίδιο πεδίο τιμών του BOOTID.UPNP.ORG για όλες τις

ανακοινώσεις, τις αναζητήσεις ανταποκρίσεων, τα μηνύματα ανανέωσης και τέλος τα μηνύματα αποχαιρετισμού (bye-bye). Τα σημεία ελέγχου μπορούν να αποκωδικοποιήσουν αυτό το πεδίο επικεφαλίδας ώστε να εντοπίσουν πιθανές αλλαγές στην κατάσταση της συσκευής λόγω κάποιου reboot. Καθώς μια συσκευή δε μπορεί να αλλάξει IP διευθύνσεις χωρίς να αλλάξει το πεδίο τιμών BOOTID.UPN.ORG, το πεδίο τιμών BOOTID.UPN.ORG μπορεί επίσης να χρησιμοποιηθεί για τη διάκριση των multi-homed συσκευών (στην περίπτωση αυτή, ένα σημείο ελέγχου θα δει μηνύματα SSDP από διαφορετικές IP διευθύνσεις με το ίδιο UUID και BOOTID.UPN.ORG πεδίο τιμών) από τις συσκευές που αλλάζουν διευθύνσεις (στην περίπτωση αυτή, το BOOTID.UPN.ORG πεδίο τιμών θα είναι διαφορετικό.) Η τιμή του NEXTBOOTID.UPNP.ORG πεδίου επικεφαλίδας υποδεικνύει την τιμή του BOOTID.UPN.ORG πεδίου επικεφαλίδας το οποίο μια multi-homed συσκευή προτίθεται να χρησιμοποιήσει σε μελλοντικές ανακοινώσεις έπειτα από την πρόσθεση μιας νέας UPnP διεπαφής. Η τιμή του CONFIGID.UPNP.ORG πεδίου επικεφαλίδας είναι ένα αναγνωριστικό του τρέχοντος συνόλου περιγραφών συσκευών και υπηρεσιών. Τα σημεία ελέγχου μπορούν να αποκωδικοποιήσουν αυτό το πεδίο ώστε να εντοπίσουν εάν χρειάζεται να αποστείλουν νέα μηνύματα ερώτησης περιγραφής. Η τιμή του SEARCHPORT.UPNP.ORG πεδίου επικεφαλίδας καθορίζει το port στο οποίο η συσκευή ακούει τα unicast M-SEARCH μηνύματα. Τα σημεία ελέγχου μπορούν να αποκωδικοποιήσουν αυτό το πεδίο ώστε να γνωρίζουν σε ποιο port θα αποσταλούν τα μηνύματα M-SEARCH. Όλα τα παραπάνω πεδία θα εξηγηθούν και αναλυτικότερα παρακάτω.

1.3.3.1 Πρωτόκολλα και πρότυπα advertisement

Για να αποσταλούν (από τις συσκευές) και να παραληφθούν (από τα σημεία ελέγχου), οι διαφημίσεις (advertisements) ακολουθείται το παρακάτω υποσύνολο της στοίβας πρωτοκόλλου UPnP



Σχήμα 5. Στοίβα πρωτοκόλλου advertisement

Στο υψηλότερο επίπεδο της στοίβας, τα μηνύματα discovery περιέχουν συγκεκριμένες πληροφορίες για τους προμηθευτές για παράδειγμα URL για την περιγραφή της συσκευής και ένα αναγνωριστικό της συσκευής. Στο αμέσως χαμηλότερο επίπεδο, οι πληροφορίες για τους προμηθευτές συμπληρώνονται από πληροφορίες του UPnP Forum για παράδειγμα ο τύπος συσκευής. Τα μηνύματα των παραπάνω στρωμάτων φιλοξενούνται στα ειδικά πρωτόκολλα UPnP . Τα μηνύματα SSDP μεταφέρονται μέσω UDP πάνω από το πρωτόκολλο IP. Ως επισήμανση τα χρώματα στις τετράγωνες αγκύλες υποδεικνύουν σε ποιο πρωτόκολλο ανήκουν τα πεδία των επικεφαλίδων και οι τιμές στα μηνύματα discovery.

1.3.3.2 Διαθεσιμότητα συσκευής

Όταν μια συσκευή προστίθεται στο δίκτυο πρέπει να κάνει multicast μηνύματα discovery ώστε να διαφημίσει τη root συσκευή, οποιεσδήποτε εσωτερικές συσκευές και τις υπηρεσίες τους. Κάθε μήνυμα discovery θα πρέπει να περιλαμβάνει τέσσερα βασικά συστατικά.

1. Ένα τύπο notification (για παράδειγμα τύπος συσκευής) όπου θα αποστέλλεται με ένα πεδίο επικεφαλίδας NT (notification Type).
2. Ένα σύνθετο αναγνωριστικό για τη διαφήμιση όπου θα αποστέλλεται με ένα πεδίο επικεφαλίδας USN (Unique Service Name).
3. Ένα URL για περισσότερες πληροφορίες για τη συσκευή όπου θα αποστέλλεται με το πεδίο επικεφαλίδας LOCATION.
4. Μια χρονική διάρκεια για την οποία η διαφήμιση θα είναι έγκυρη και θα αποστέλλεται με το πεδίο επικεφαλίδας CACHE-CONTROL.

Ειδικότερα, μια root συσκευή θα πρέπει να κάνει multicast τρία discovery μηνύματα για τη root συσκευή, δύο μηνύματα discovery για κάθε ενσωματωμένη συσκευή και ένα για κάθε τύπο υπηρεσίας σε κάθε συσκευή.

Αν μια root συσκευή έχει d ενσωματωμένες συσκευές, s ενσωματωμένες υπηρεσίες και μόνο k ευδιάκριτους τύπους υπηρεσιών, αυτό συνεπάγεται ότι θα έχουμε 3+2d+k αιτήματα. Αν μια συγκεκριμένη συσκευή ή ενσωματωμένη συσκευή περιέχει πολλαπλές περιπτώσεις ενός συγκεκριμένου τύπου υπηρεσίας, είναι απαραίτητο απλά να διαφημίσει τον τύπο υπηρεσίας μονάχα μια φορά και όχι κάθε περίπτωση ξεχωριστά. Αν βέβαια δυο ενσωματωμένες συσκευές περιέχουν μια υπηρεσία του ίδιου τύπου τότε αυτές τις υπηρεσίες πρέπει να τις ανακοινώνουν ξεχωριστά. Αυτό γνωστοποιεί την πλήρη έκταση των δυνατοτήτων των συσκευών στα ενδιαφερόμενα σημεία ελέγχου. Τα μηνύματα αυτά θα πρέπει να αποστέλλονται ως μια σειρά με σχεδόν παρόμοιους χρόνους λήξης.

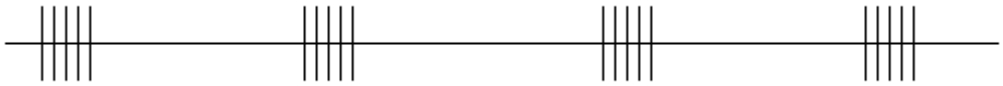
Οι πιο πρόσφατες UPnP συσκευές και υπηρεσίες θα πρέπει να είναι πλήρως συμβατές με τις προηγούμενες εκδόσεις του ίδιο τύπου. Οι συσκευές θα πρέπει να διαφημίζουν την υψηλότερη υποστηριζόμενη έκδοση από κάθε υποστηριζόμενο τύπο. Για παράδειγμα, αν μια συσκευή υποστηρίζει την έκδοση 2 μιας υπηρεσίας ‘’Audio ‘’, χρειάζεται να ανακοινώσει μόνο την έκδοση 2 παρόλο που υποστηρίζει και την έκδοση 1. Ένα σημείο ελέγχου που υποστηρίζει μια δεδομένη έκδοση μιας συσκευής ή υπηρεσίας, θα πρέπει να είναι σε θέση να αλληλεπιδρά και με υψηλότερες εκδόσεις λόγω απαίτησης συμβατότητας, αλλά κάνοντας χρήση της λειτουργικότητας που έχει οριστεί στην χαμηλότερη έκδοση. Για παράδειγμα ένα σημείο ελέγχου που υποστηρίζει μονάχα την έκδοση 1 μιας υπηρεσίας ‘’Audio’’, και η συσκευή ανακοινώνει πως υποστηρίζει την έκδοση 2 της υπηρεσίας ‘’Audio’’, το σημείο ελέγχου θα πρέπει να αναγνωρίσει τη συσκευή και να είναι σε θέση να τη χρησιμοποιήσει.

Η επιλογή της κατάλληλης διάρκειας για τις ανακοινώσεις (advertisements) είναι μια ισορροπία μεταξύ της ελαχιστοποίησης της κυκλοφορίας στο δίκτυο και της πιο άμεσης ενημέρωσης για την κατάσταση της συσκευής. Οι σχετικά σύντομες διάρκειες κοντά στο ελάχιστο των 1800 δευτερολέπτων εξασφαλίζει πως τα σημεία ελέγχου έχουν την τρέχουσα κατάσταση της συσκευής, εις βάρος όμως πρόσθετης κυκλοφορίας στο δίκτυο. Οι μεγαλύτερες διάρκειες των advertisements συμβιβάζονται ως αναφορά την ανανέωση της κατάστασης της συσκευής ωστόσο μπορούν να μειώσουν σημαντικά την κυκλοφορία στο δίκτυο. Γενικά, οι προμηθευτές συσκευών πρέπει να επιλέξουν μια τιμή η οποία θα ανταποκρίνεται στην αναμενόμενη χρήση της συσκευής. Σύντομες διάρκειες για συσκευές που πρόκειται να είναι μέρος του δικτύου για μικρά χρονικά διαστήματα και μακροχρόνιες διάρκειες γι’ αυτές που πρόκειται να παραμείνουν για μεγαλύτερο χρονικό διάστημα συνδεδεμένες. Έτσι για παράδειγμα συσκευές που συχνά συνδέονται και επανασυνδέονται στο δίκτυο όπως οι κινητές και ασύρματες συσκευές, θα πρέπει να χρησιμοποιούν μια πιο σύντομη διάρκεια ανακοινώσεων (advertisements) έτσι ώστε τα σημεία ελέγχου να έχουν καλύτερη εικόνα της διαθεσιμότητας τους. Οι ανακοινώσεις, αρχικές και μεταγενέστερες θα πρέπει να έχουν παρόμοιες χρονικές διάρκειες. Οι ανακοινώσεις στο αρχικό τους σύνολο θα πρέπει να σταλούν όσο το δυνατό συντομότερα, ενώ οι μεταγενέστερες ανανεώσεις των ανακοινώσεων μπορούν να μεταδοθούν τμηματικά καθ’ όλη τη χρονική διάρκεια σύνδεσης της συσκευής παρά να αποσταλούν ως ομάδα.

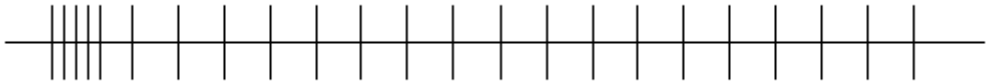
Η διάδοση των ανανεώσεων των ανακοινώσεων καθ’ όλη τη διάρκεια του χρόνου και όχι ως ομάδα, βελτιώνει την αξιοπιστία σε περίπτωση που υπάρχουν δυσλειτουργίες στο δίκτυο.

Τα σχήματα που ακολουθούν αναπαριστούν τη διάδοση των ανακοινώσεων από τη χρονική στιγμή που η συσκευή συνδέεται στο δίκτυο. Έτσι κατά τη χρονική περίοδο σύνδεσης που συμβολίζεται με την οριζόντια γραμμή, ξεκινά πρώτα η αποστολή των αρχικών ανακοινώσεων και εν συνεχεία ακολουθούν οι μεταγενέστερες, όπου στην πρώτη περίπτωση αναπαρίσταται η αποστολή τους κατά ομάδες ενώ στη δεύτερη τμηματικά καθ' όλη τη διάρκεια της χρονικής περιόδου.

Σχήμα 1.1



Σχήμα 1.2



Οι συσκευές θα πρέπει να περιμένουν ένα τυχαίο διάστημα πριν την αποστολή του συνόλου των αρχικών ανακοινώσεων μειώνοντας έτσι την πιθανότητα των δικτυακών θυελλών(network storms). Ομοίως τυχαίο διάστημα θα πρέπει να περιμένουν και στις περιπτώσεις ανάκτησης νέας IP διεύθυνσης είτε πρόσθεσης μιας νέας UPnP διεπαφής.

Λόγω της αναξιόπιστης φύσης του πρωτοκόλλου UDP, οι συσκευές θα πρέπει να αποστείλουν το σύνολο των μηνυμάτων discovery περισσότερο από μια φορά, με κάποια καθυστέρηση μεταξύ των συνόλων μερικές εκατοντάδες χιλιοστά του δευτερολέπτου. Για την αποφυγή της συμφόρησης του δικτύου τα μηνύματα discovery δε θα πρέπει να αποστέλλονται περισσότερες από τρείς φορές. Επιπροσθέτως η συσκευή θα πρέπει να αποστείλει εκ νέου τις ανακοινώσεις (advertisements) περιοδικά πριν τη λήξη της διάρκειας που ορίζεται στο πεδίο επικεφαλίδας CACHE-CONTROL. Συνίσταται η ανανέωση αυτή των ανακοινώσεων να γίνεται σε ένα τυχαία κατανεμημένο διάστημα πριν από το μισό του χρόνου λήξης των ανακοινώσεων, έτσι ώστε να παρέχεται η δυνατότητα ανάκτησης χαμένων ανακοινώσεων πριν από τη λήξη καθώς και να κατανέμεται στη διάρκεια του χρόνου η ανανέωση των ανακοινώσεων πολύπλοκων συσκευών προς αποφυγή συμφόρησης στην κυκλοφορία του δικτύου. Επίσης σημαντικό αναφοράς είναι το ότι τα πακέτα UDP είναι περιορισμένα σε μήκος και κάθε discovery μήνυμα θα πρέπει να ταιριάξει εξ ολοκλήρου σε ένα μόνο πακέτο UDP. Βάση αυτού δεν

υπάρχει καμία εγγύηση πως τα παραπάνω μηνύματα 3+2d+k θα φτάσουν σε μια συγκεκριμένη σειρά.

Μια multi-homed συσκευή θα πρέπει να εκτελέσει τις παραπάνω διαδικασίες ανακοινώσεων για κάθε UPnP διεπαφή. Οι ανακοινώσεις που αποστέλλονται στις πολλαπλές UPnP διεπαφές πρέπει να περιλαμβάνουν τις ίδιες τιμές εκτός των HOST, CACHE-CONTROL και LOCATION. Το πεδίο τιμής HOST μια ανακοίνωσης (advertisement) πρέπει να είναι η καθορισμένη multicast διεύθυνση που καθορίζεται για τα πρωτόκολλα IPv4 και IPv6 που χρησιμοποιούνται στη διεπαφή. Το URL που ορίζεται στο πεδίο επικεφαλίδας LOCATION θα πρέπει να είναι εφικτό για τη διεπαφή στην οποία θα σταλεί η ανακοίνωση. Τέλος, οι ανακοινώσεις που αποστέλλονται σε διαφορετικές διαπαφές θα πρέπει να έχουν διαφορετικές τιμές πεδίου CAHE-CONTROL και μπορεί να σταλούν με διαφορετικές συχνότητες.

Όταν μια συσκευή προστίθεται στο δίκτυο, πρέπει να αποστείλει ένα multicast μήνυμα με τη μέθοδο NOTIFY και ssdp:alive στο πεδίο επικεφαλίδας NTS με την ακόλουθη μορφοποίηση. Οι τιμές με γραμματοσειρά *italics* αντικαθιστώνται από πραγματικές τιμές.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: notification type
NTS: ssdp:alive
SERVER: OS/version UPnP/1.1 product/version
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update
message
CONFIGID.UPNP.ORG: number used for caching description information
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Ο χρόνος TTL θα πρέπει να τεθεί σε 2 δευτερόλεπτα και θα πρέπει να μπορεί να διαμορφωθεί. Στη συνέχεια αναλύονται λεπτομερώς τα πεδία που αναφέρθηκαν παραπάνω.

Request line

Θα πρέπει να είναι ‘’ NOTIFY * HTTP/1.1’’

Όπου NOTIFY → είναι μέθοδος για την αποστολή notifications και γεγονότων

→τα μηνύματα που αναφέρονται γενικά και όχι σε μια συγκεκριμένη πηγή πρέπει να δηλώνονται σαν ‘’’’

HTTP /1.1 → έκδοση http

Πεδία επικεφαλίδων

HOST → Το συγκεκριμένο πεδίο είναι απαραίτητο και περιλαμβάνει την multicast διεύθυνση και το port που είναι δεσμευμένο για το πρωτόκολλο SSDP. Θα πρέπει να έχει τη μορφή 239.255.255.250:1900. Σε περίπτωση που το port number δεν έχει δηλωθεί, ο δέκτης θα πρέπει να υποθέσει το προεπιλεγμένο SSDP port 1900.

CACHE-CONTROL → Το συγκεκριμένο πεδίο είναι επίσης απαραίτητο και αποτελείται από το ''max-age'' ακολουθούμενο από έναν αριθμό που δηλώνει τα δευτερόλεπτα για τα οποία η διαφήμιση (advertisement) είναι έγκυρη. Μετά από το χρόνο αυτό τα σημεία ελέγχου θεωρούν πως η συσκευή δεν είναι πλέον διαθέσιμη στο δίκτυο. Σε περίπτωση που ένα σημείο ελέγχου έχει λάβει τουλάχιστον μια διαφήμιση από τη root συσκευή, της ενσωματωμένες συσκευές είτε τις υπηρεσίες τους , η οποία είναι ενεργή ακόμα, τότε θα θεωρήσει πως τα παραπάνω είναι ακόμη διαθέσιμα. Η συγκεκριμένη διάρκεια θα πρέπει να είναι μεγαλύτερη ή ίση των 1800 δευτερολέπτων (30 λεπτών), ωστόσο υπάρχουν και εξαιρέσεις που καθορίζονται από ένα UPnP προμηθευτή.

LOCATION → Απαραίτητο πεδίο, όπου η τιμή του περιλαμβάνει ένα URL για την UPnP περιγραφή (description) της root συσκευής.

NT → Απαραίτητο πεδίο όπου η τιμή του περιλαμβάνει τύπους notification (γνωστοποίησης) οι οποίοι πρέπει να είναι κάποιοι από τους ακόλουθους:

upnp:rootdevice
Αποστέλλεται μία φορά για την root συσκευή.

uuid:*device-UUID*
Αποστέλλεται μια φορά για κάθε συσκευή, root ή ενσωματωμένη. Όπου *device-UUID* ορίζεται από τον UPnP προμηθευτή. Παραπάνω αναφέρθηκε ένα παράδειγμα UUID.

urn:**schemas-upnp-org**:**device**:*deviceType*:*ver*
Αποστέλλεται μια φορά για κάθε συσκευή, root ή ενσωματωμένη. Όπου *deviceType* και *ver* καθορίζονται από το UPnP Forum και το *ver* ορίζει την έκδοση του τύπου της συσκευής.

urn:**schemas-upnp-org**:**service**:*serviceType*:*ver*
Αποστέλλεται μια φορά για κάθε υπηρεσία. Όπου *serviceType* και *ver* καθορίζονται από το UPnP Forum και το *ver* ορίζει την έκδοση του τύπου υπηρεσίας.

urn:*domain-name*:**device**:*deviceType*:*ver*
Αποστέλλεται μια φορά για κάθε συσκευή root ή ενσωματωμένη. Όπου *domain-name* είναι Domain name του προμηθευτή, *deviceType* και *ver* καθορίζονται από τον UPnP προμηθευτή και *ver* καθορίζει την έκδοση του τύπου συσκευής.

urn:*domain-name*:**service**:*serviceType*:*ver*
Αποστέλλεται μια φορά για κάθε υπηρεσία. Όπου *domain-name* είναι Domain name του προμηθευτή, *serviceType* και *ver* καθορίζονται από τον UPnP προμηθευτή και *ver* καθορίζει την έκδοση του τύπου υπηρεσίας.

NTS → Είναι απαραίτητο πεδίο, η τιμή του περιέχει sub τύπους notifications (γνωστοποιήσεις) και θα πρέπει να είναι της μορφής ssdp:alive.

SERVER →Είναι απαραίτητο πεδίο τύπου string και καθορίζεται από τον UPnP προμηθευτή. Το πεδίο αυτό πρέπει να ξεκινά με τα παρακάτω χαρακτηριστικά που ορίζονται από το πρωτόκολλο HTTP/1.1 . Το πρώτο πεδίο πρέπει να περιγράφει το λειτουργικό σύστημα στον τύπο *OS name*/*OS version*, το δεύτερο πεδίο περιγράφει την έκδοση του UPnP UPnP/1.1 και το τρίτο πεδίο περιγράφει το προϊόν *product name*/*product version*. Για παράδειγμα, "SERVER *unix*/*5.1* **UPnP/1.1** *MyProduct*/*1.0*''. Τα σημεία ελέγχου πρέπει να είναι προετοιμασμένα να δεχτούν μια υψηλότερη δευτερεύουσα έκδοση από τη UPnP έκδοση που υποστηρίζουν.

USN → Είναι απαραίτητο πεδίο όπου η τιμή του περιλαμβάνει ένα μοναδικό όνομα υπηρεσίας (Unique Service Name) και θα πρέπει να είναι ένα από αυτά που ακολουθούν παρακάτω. Προσδιορίζει μια μοναδική περίπτωση συσκευής ή υπηρεσίας. Το πρόθεμα (πριν από τη διπλή άνω και κάτω τελεία) πρέπει να ταιριάξει με την τιμή του στοιχείου UDN στην περιγραφή της συσκευής.

uuid:*device-UUID*::**upnp:rootdevice**
Αποστέλλεται μια φορά για κάθε συσκευή. Όπου *device-UUID* καθορίζεται από τον UPnP προμηθευτή.

uuid:*device-UUID*
Αποστέλλεται μια φορά για κάθε συσκευή, root ή ενσωματωμένη. Όπου *device-UUID* ορίζεται από τον UPnP προμηθευτή.

uuid:*device-UUID*::urn:**schemas-upnp-org**:**device**:*deviceType*:*ver*
Αποστέλλεται μια φορά για κάθε συσκευή root ή ενσωματωμένη. Όπου το *device-UUID* ορίζεται από τον UPnP προμηθευτή, όπου *deviceType* και *ver* καθορίζονται από το UPnP Forum και το *ver* ορίζει την έκδοση του τύπου της συσκευής.

uuid:*device-UUID*::urn:**schemas-upnp-org**:**service**:*serviceType*:*ver*
Αποστέλλεται μια φορά για κάθε υπηρεσία. Όπου το *device-UUID* ορίζεται από τον UPnP προμηθευτή, όπου *serviceType* και *ver* καθορίζονται από το UPnP Forum και το *ver* ορίζει την έκδοση του τύπου της συσκευής.

uuid:*device-UUID*::urn:*domain-name*:**device**:*deviceType*:*ver*
Αποστέλλεται μια φορά για κάθε συσκευή root ή ενσωματωμένη. Όπου *device-UUID, domain-name* (Domain name του προμηθευτή), *deviceType* και *ver* καθορίζονται από τον UPnP προμηθευτή και *ver* καθορίζει την έκδοση του τύπου συσκευής.

uuid:*device-UUID*::urn:*domain-name*:**service**:*serviceType*:*ver*
Αποστέλλεται μια φορά για κάθε υπηρεσία. Όπου *device-UUID, domain-name* (Domain name του προμηθευτή), *serviceType* και *ver* καθορίζονται από τον UPnP προμηθευτή και *ver* καθορίζει την έκδοση του τύπου υπηρεσίας.

BOOTID.UPNP.ORG → Είναι επίσης απαραίτητο πεδίο. Το πεδίο επικεφαλίδας BOOTID.UPNP.ORG αντιπροσωπεύει την περίπτωση εκκίνησης (boot) της συσκευής και εκφράζεται με μια μονοτονικά αυξανόμενη τιμή. Θα πρέπει να είναι μη αρνητικός ακέραιος αριθμός 31bit, ASII κωδικοποίησης χωρίς δεκαδικά ψηφία να ακολουθούν (αν υπάρχουν ακολουθούμενα μηδενικά θα πρέπει να αγνοηθούν από τον παραλήπτη), όπου θα πρέπει να αυξάνεται κατά ένα για κάθε αρχική ανακοίνωση της συσκευής. Η τιμή του πεδίου θα πρέπει να παραμείνει η ίδια σε όλες τις περιοδικά επαναλαμβανόμενες ανακοινώσεις. Το πεδίο επικεφαλίδας BOOTID.UPNP.ORG θα πρέπει να συμπεριλαμβάνεται σε όλες τις ανακοινώσεις της root συσκευής, των ενσωματωμένων συσκευών και των υπηρεσιών τους. Στην περίπτωση που η συσκευή ανακοινώσει μια ανανέωση στην τιμή της αποστέλλοντας ένα SSDP μήνυμα ανανέωσης, για όσο θα παραμένει διαθέσιμη στο δίκτυο, σε όλες τις ανακοινώσεις, αναζητήσεις, ανταποκρίσεις μηνύματα ανανέωσης και αποχαιρετισμού θα πρέπει να χρησιμοποιείται η ίδια τιμή BOOTID.UPNP.ORG. Τα σημεία ελέγχου μπορούν να χρησιμοποιούν αυτό το πεδίο επικεφαλίδας ώστε να διαπιστώσουν τις περιπτώσεις που η συσκευή αποσυνδέεται και επανασυνδέεται στο δίκτυο και για διάφορους άλλους λόγους όπως η εγκαθίδρυση νέων συνδρομών ή ο έλεγχος για τις αλλαγές στην κατάσταση της συσκευής.

CONFIGID.UPNP.ORG → Απαραίτητο πεδίο όπου θα πρέπει να είναι ένας αριθμός μη αρνητικός, 31 bit ακέραιος, ASCII κωδικοποίησης, δεκαδικός, χωρίς ακολουθούμενα μηδενικά (σε περίπτωση που υπάρχουν αυτά θα πρέπει να αγνοηθούν από το παραλήπτη)και πρέπει να αντιπροσωπεύει τον αριθμό

διαμόρφωσης της root συσκευής. Οι UPnP 1.1 συσκευές μπορούν να ορίσουν configid αριθμούς από το 0 έως 16777215 (2^24-1). Οι ακόμη μεγαλύτεροι αριθμοί είναι δεσμευμένοι για μελλοντική χρήση. Η διαμόρφωση (configuration) μιας συσκευής αποτελείται από την ακόλουθη πληροφορία: το DDD της root συσκευής και όλων των ενσωματωμένων συσκευών της και τα SCPDs όλων των συμπεριλαμβανομένων υπηρεσιών. Εάν οποιοδήποτε μέρος της διαμόρφωσης αλλάξει, τότε την τιμή του πεδίου CONFIGID.UPNP.ORG πρέπει να αλλάξει και αυτή. Το πεδίο επικεφαλίδας CONFIGID.UPNP.ORG πρέπει να περιληφθεί σε όλες τις ανακοινώσεις της root συσκευής, των ενσωματωμένων συσκευών και των υπηρεσιών τους. Τέλος, λόγω του CONFIGID.UPNP.ORG μειώνονται τα μέγιστα (peaks) φορτία των δικτύων που μπορεί να προκληθούν στις UPnP συσκευές.

SEARCHPORT.UPNP.ORG→Αν μια συσκευή δεν αποστέλλει το πεδίο επικεφαλίδας SEARCHPORT.UPNP.ORG, θα πρέπει να ανταποκριθεί στα unicast μηνύματα M-SEARCH στο port 1900. Μόνο στην περίπτωση που το port 1900 είναι μη διαθέσιμο η συσκευή μπορεί να επιλέξει ένα διαφορετικό port ώστε να ανταποκριθεί στα μηνύματα αυτά. Στη περίπτωση που το πεδίο SEARCHPORT.UPNP.ORG αποστέλλεται, τότε θα πρέπει να είναι ένας ακέραιος, δεκαδικός αριθμός κωδικοποίησης ASCII χωρίς την ακολουθία μηδενικών(αν ακολουθούν θα αγνοηθούν από τον παραλήπτη), στο εύρος τιμών 49152-65535.

1.3.3.3 Μη διαθέσιμες συσκευές

Όταν μια συσκευή και οι υπηρεσίες της πρόκειται να αποχωρήσουν από το δίκτυο, η συσκευή θα πρέπει να κάνει multicast ένα ssdp:bye bye μήνυμα για κάθε ένα από τα ssdp:alive multicast μηνύματα που δεν έχουν λήξει ακόμη. Αν η συσκευή αφαιρεθεί απότομα από το δίκτυο, πιθανό να μην προλάβει να αποστείλει τα μηνύματα αυτά. Έτσι σαν επιφύλαξη θα πρέπει τα discovery μηνύματα να έχουν μια τιμή χρόνου λήξης στο πεδίο επικεφαλίδας CACHE-CONTROL όπως αναφέρθηκε παραπάνω. Σημειωτέον πως όταν ένα σημείο ελέγχου αποχωρήσει από το δίκτυο δεν είναι απαραίτητο να γίνει κάποια ανάλογη ενέργεια. Κάθε multicast μήνυμα θα πρέπει να έχει μια μέθοδο NOTIFY και ssdp:bye bye στο πεδίο επικεφαλίδας NTS με την μορφοποίηση που ακολουθεί παρακάτω. Οι τιμές *italics* κι εδώ αντικαθιστώνται από πραγματικές τιμές.

Όταν μια multi-homed συσκευή πρόκειται να αποχωρήσει από το δίκτυο από μια ή περισσότερες UPnP διεπαφές της, θα πρέπει να ανακαλέσει τα discovery μηνύματα αποστέλλοντας ένα multicast μήνυμα για κάθε ssdp:alive μήνυμα που είχε αποσταλεί πριν σε αυτές τις διεπαφές και IPδιευθύνσεις. Βέβαια δε χρειάζεται να αποσταλούν τέτοιου είδους μηνύματα για τις διεπαφές οι οποίες πρόκειται να παραμείνουν στο δίκτυο.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
NT: notification type
NTS: ssdp:byebye
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update
message
CONFIGID.UPNP.ORG: number used for caching description information
```

Ο χρόνος TTL για κάθε IP πακέτο θα πρέπει να είναι 2 δευτερόλεπτα και να είναι διαμορφώσιμος.

1.3.3.4 Ανανέωση (Update) συσκευών

Όταν μια UPnP διεπαφή προστίθεται σε μια multi-homed συσκευή, η συσκευή θα πρέπει να αυξήσει την τιμή BOOTID.UPNP.ORG, να κάνει multicast ένα ssdp:update μήνυμα για κάθε root συσκευή, ενσωματωμένη συσκευή και τις υπηρεσίες για όλες τις υπάρχουσες ενεργές διεπαφές, να ανακοινώσει την αλλαγή στην τιμή BOOTID.UPNP.ORG και να διαφημίσει εκ νέου τον εαυτό της σε υπάρχουσες και νέες UPnP διεπαφές με τη νέα τιμή BOOTID.UPNP.ORG. Ομοίως αν κάποια multi-homed συσκευή χάσει τη σύνδεση με κάποια διεπαφή και την επανακτήσει, είτε αν αλλάξει η IP σε κάποια διεπαφή, θα πρέπει να αυξήσει την τιμή BOOTID.UPNP.ORG να κάνει multicast ssdp:update μηνύματα για τη root συσκευή, τις ενσωματωμένες και τις υπηρεσίες τους σε όλες τις μη επηρεαζόμενες διεπαφές ανακοινώνοντας την αλλαγή στο BOOTID.UPNP.ORG πεδίο, και να διαφημίσει τον εαυτό της σε υπάρχουσες και νέες διεπαφές με την νέα τιμή BOOTID.UPNP.ORG. Σε κάθε περίπτωση τα ssdp:update μηνύματα για τις root συσκευές θα πρέπει να αποσταλούν το συντομότερο. Τα υπόλοιπα θα πρέπει να αποσταλούν απλωμένα στη διάρκεια χρόνου. Βέβαια όλα τα μηνύματα ssdp:update πρέπει να αποστέλλονται πριν τα μηνύματα ανακοινώσεων της νέας τιμής BOOTID.UPNP.ORG

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
LOCATION: URL for UPnP description for root device
NT: notification type
NTS: ssdp:update
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: BOOTID value that the device has used in its previous announcements
CONFIGID.UPNP.ORG: number used for caching description information
NEXTBOOTID.UPNP.ORG: new BOOTID value that the device will use in subsequent announcements
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Ο χρόνος TTL πρέπει επίσης αρχικά να είναι 2 δευτερόλεπτα και αν χρειαστεί να είναι διαμορφώσιμος.

Πεδία επικεφαλίδας

NEXTBOOTID.UPNP.ORG → Πρόκειται επίσης για απαραίτητο πεδίο. Η τιμή του περιλαμβάνει το νέο πεδίο τιμής του BOOTID.UPNP.ORG που η συσκευή πρόκειται να χρησιμοποιήσει σε μελλοντικές ανακοινώσεις που αφορούν την ίδια τις ενσωματωμένες συσκευές ή τις υπηρεσίες τους. Θα πρέπει να είναι μη αρνητικός, ακέραιος, δεκαδικός, μη ακολουθούμενος από μηδενικά (αν ακολουθούν θα πρέπει να αγνοηθούν από το παραλήπτη. Τέλος θα πρέπει η τιμή αυτή να είναι μεγαλύτερη από την τιμή του BOOTID.UPNP.ORG.

LOCATION → Η τιμή του πεδίου θα πρέπει να είναι η ίδια με του πεδίου LOCATION που έχει σταλεί στα προηγούμενα SSDP μηνύματα.

NTS → Είναι απαραίτητο πεδίο. Η τιμή του περιέχει sub τύπους notifications (γνωστοποιήσεις) και θα πρέπει να είναι της μορφής ssdp:update.

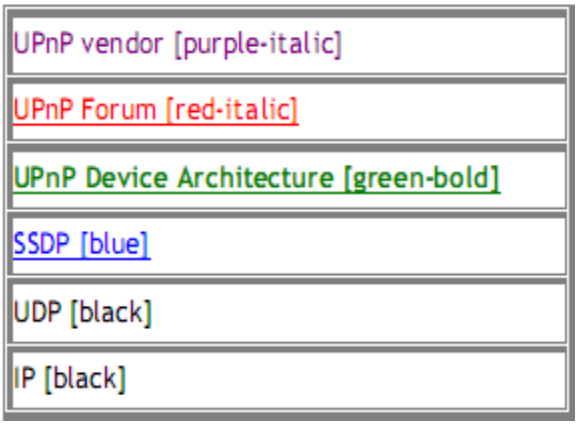
Όλα τα υπόλοιπα πεδία του μυνήματος έχουν περιγραφεί παραπάνω.

1.3.4 Αναζήτηση

Όταν ένα σημείο ελέγχου προστίθεται στο δίκτυο, το UPnP discovery πρωτόκολλο του επιτρέπει να αναζητήσει συσκευές ενδιαφέροντος στο δίκτυο. Αυτό επιτυγχάνεται εκπέμποντας στη multicast διεύθυνση και port (239.255.255.250:1900) ένα μήνυμα αναζήτησης με κατάλληλη μορφοποίηση προσδιορίζοντας την συσκευή ή την υπηρεσία στην οποία απευθύνεται . Επίσης τα σημεία ελέγχου , μπορούν να εκπέμψουν ένα unicast μήνυμα αναζήτησης σε μια γνωστή διεύθυνση IP και στο port 1900 ή στο καθορισμένο port από το SEARCHPORT.UPNP.ORG ώστε να επιβεβαιώσουν την ύπαρξη UPnP συσκευών και υπηρεσιών στην IP διεύθυνση. Για παράδειγμα, μια αναζήτηση unicast μπορεί να χρησιμοποιηθεί για ένα γρήγορο έλεγχο ώστε να διαπιστωθεί αν μια γνωστή UPnP συσκευή ή υπηρεσία είναι ακόμη διαθέσιμη στο δίκτυο. Τα multi-homed σημεία ελέγχου επίσης μπορεί να επιλέξουν να αποστείλουν μηνύματα discovery σε μερικές μία ή όλες τις UPnP διεπαφές.

1.3.4.1 Πρωτόκολλα αναζήτησης και πρότυπα

Τα σημεία ελέγχου για να αναζητήσουν συσκευές καθώς και οι συσκευές για να εντοπιστούν από τα σημεία ελέγχου χρησιμοποιούν την ακόλουθη στοίβα πρωτοκόλλου



Σχήμα 6. Στοίβα πρωτοκόλλου αναζήτησης

Στο υψηλότερο επίπεδο της στοίβας τα μηνύματα αναζήτησης περιέχουν πληροφορίες καθορισμένες από τους προμηθευτές για παράδειγμα τα αναγνωριστικά των σημείων ελέγχου των συσκευών και των υπηρεσιών. Μετακινούμενοι στα χαμηλότερα επίπεδα της στοίβας, η πληροφορία των προμηθευτών συμπληρώνεται από το UPnP Forum και περιλαμβάνει για παράδειγμα τύπους συσκευών ή υπηρεσιών. Στη συνέχεια, τα αιτήματα αναζήτησης μεταδίδονται μέσω multicast και unicast SSDP μηνυμάτων Οι απαντήσεις στην αναζήτηση μεταδίδονται μέσω unicast SSDP μηνυμάτων. Και στις δύο περιπτώσεις τα μηνύματα μεταδίδονται μέσω UDP πάνω από το πρωτόκολλο IP.

1.3.4.2 Αιτήματα αναζήτησης μέσω M-SEARCH

Όταν ένα σημείο ελέγχου σκοπεύει να αναζητήσει μια συσκευή στο δίκτυο , θα πρέπει να αποστείλει ένα multicast αίτημα με τη μέθοδο M-SEARCH με την μορφοποίηση που ακολουθεί. Όταν τα σημεία ελέγχου γνωρίζουν την IP διεύθυνση μιας συγκεκριμένης συσκευής μπορούν να αποστείλουν μηνύματα unicast παρόμοιας μορφοποίησης με τη μέθοδο M-SEARCH.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
USER-AGENT: OS/version UPnP/1.1 product/version
```

Ο χρόνος TTL θα πρέπει να είναι 2 δευτερόλεπτα και να είναι διαμορφώσιμος.

Request line

Θα πρέπει να είναι “M-SEARCH * HTTP/1.1”

M-SEARCH→ μέθοδος αιτημάτων αναζήτησης

→τα μηνύματα που αναφέρονται γενικά και όχι σε μια συγκεκριμένη πηγή πρέπει να δηλώνονται σαν ‘’’’

HTTP /1.1 → έκδοση HTTP

Πεδία επικεφαλίδων

MAN → Αυτό το πεδίο είναι απαραίτητο και πρέπει να έχει την τιμή "ssdp:discover". Το πεδίο καθεαυτού καθορίζει τον σκοπό της εφαρμογής της επέκτασης του.

MX→ Είναι απαραίτητο πεδίο. Η τιμή του περιλαμβάνει τον μέγιστο χρόνο σε δευτερόλεπτα ο οποίος πρέπει να είναι μεγαλύτερος από 1 και μικρότερος από 5. Είναι ο μέγιστος χρόνος αναμονής όπου η συσκευή θα πρέπει να περιμένει για να ανταποκριθεί έτσι ώστε να διατηρηθεί μια ισορροπία φόρτισης των σημείων ελέγχων κατά τη διαδικασία των ανταποκρίσεων. Η τιμή αυτή αυξάνεται αν υπάρχει ένας μεγάλος αριθμός συσκευών που αναμένουν να ανταποκριθούν.

ST → Απαραίτητο πεδίο όπου η τιμή του περιλαμβάνει τον στόχο της αναζήτησης (συσκευή ή υπηρεσία). Θα πρέπει να είναι ένα από τα παρακάτω:

ssdp:all

Αναζήτηση για όλες τις συσκευές και τις υπηρεσίες.

Upnp:rootdevice

Αναζήτηση root συσκευών.

Uuid:*device-UUID*

Αναζήτηση μια συγκεκριμένης συσκευής με το συγκεκριμένο uuid.

urn:**schemas-upnp-org:device:***deviceType:ver*

Αναζήτηση μιας συσκευής ενός συγκεκριμένου τύπου *deviceType* και έκδοσης *ver*.

urn: **schemas-upnp-org:service:***serviceType:ver*

Αναζήτηση υπηρεσίας συγκεκριμένου τύπου *deviceType* και έκδοσης *ver*.

urn:*domain-name*:**device**:*deviceType:ver*
Αναζήτηση μιας συσκευής με συγκεκριμένο domain name. Όπου *domain-name* είναι το domain name του προμηθευτή.

urn:*domain-name*:**service**:*deviceType:ver*
Αναζήτηση μιας υπηρεσίας με συγκεκριμένο domain name. Όπου *domain-name* είναι το domain name του προμηθευτή.

USER-AGENT→ Είναι προαιρετικό τύπου string και καθορίζεται από τους προμηθευτές UPnP. Το πεδίο αυτό πρέπει να ξεκινά με τα παρακάτω χαρακτηριστικά που ορίζονται από το πρωτόκολλο HTTP/1.1 . Το πρώτο πεδίο πρέπει να περιγράφει το λειτουργικό σύστημα στον τύπο *OS name*/*OS version*, το δεύτερο πεδίο περιγράφει την έκδοση του UPnP UPnP/1.1 και το τρίτο πεδίο περιγράφει το προϊόν *product name*/*product version*. Για παράδειγμα, "USER AGENT: *unix*/*5.1* **UPnP/1.1** *MyProduct*/*1.0*''. Τα σημεία ελέγχου πρέπει να είναι προετοιμασμένα να δεχτούν μια υψηλότερη δευτερεύουσα έκδοση από τη UPnP έκδοση που υποστηρίζουν.
Τα υπόλοιπα πεδία έχουν περιγραφεί παραπάνω.

Για unicast αναζήτηση , η μορφοποίηση του μηνύματος θα είναι η παρακάτω. Οι τιμές σε *italics* αντικαθιστώνται από πραγματικές τιμές.

```
M-SEARCH * HTTP/1.1
HOST: hostname:portNumber
MAN: "ssdp:discover"
ST: search target
USER-AGENT: OS/version UPnP/1.1 product/version
```

Λόγω της αναξιόπιστης φύσης της τεχνολογίας του UDP, τα σημεία ελέγχου πρέπει να στείλουν κάθε M—SEARCH μήνυμα περισσότερες από μια φορές.
Για ένα multicast αίτημα, το σημείο ελέγχου θα περιμένει για απαντήσεις που θα φτάσουν από τις συσκευές τουλάχιστον το χρονικό διάστημα που ορίζεται στο πεδίο επικεφαλίδας MX. Η τυχαία κατανομή των απαντήσεων κατά το χρονικό διάστημα MX σημαίνει πως ο ανταποκριτής μπορεί να στείλει μια απάντηση σε MX δευτερόλεπτα έπειτα από τη λήψη του M-SEARCH αιτήματος. Η τιμή του πεδίου MX μπορεί να διαμορφώνεται από παραμέτρους όπως ο αριθμός των ανταποκριτών.

1.3.5 Ανταπόκριση στη αναζήτηση συσκευών

Για να εντοπιστεί από μια αναζήτηση, μια συσκευή θα πρέπει εκπέμψει multicast μια ανταπόκριση στην IP διεύθυνση και στο port της πηγής που έστειλε το αίτημα στην multicast διεύθυνση. Οι συσκευές ανταποκρίνονται όταν το πεδίο

επικεφαλίδας ST του M-SEARCH αιτήματος είναι ssdp:all, upnp:rootdevice, ‘’uuid:’’ ακολουθούμενο από ένα UUID που ταιριάζει ακριβώς με αυτό που διαφημίζεται από τη συσκευή είτε αν το M-SEARCH μήνυμα ταιριάζει με ένα τύπο συσκευής η υπηρεσίας που υποστηρίζεται από τη συσκευή. Οι multi-homed συσκευές θα πρέπει να ανταποκριθούν χρησιμοποιώντας την ίδια UPnP διεπαφή στην οποία έφτασε το αίτημα. Το URL που καθορίζεται στο πεδίο επικεφαλίδας LOCATION θα πρέπει να ορίζει μια διεύθυνση η οποία είναι εφικτή στην διεπαφή αυτή.

Οι συσκευές που ανταποκρίνονται σε ένα μήνυμα M-SEARCH θα πρέπει να περιμένουν πριν την ανταπόκριση τους ένα τυχαίο διάστημα μεταξύ 0 και του αριθμού δευτερολέπτων που καθορίζονται στο πεδίο επικεφαλίδας MX του αιτήματος αναζήτησης, έτσι ώστε να αποφευχθεί η συμφόρηση των αιτούντων σημείων ελέγχου με απαντήσεις από πολλαπλές συσκευές. Σε περιπτώσεις που βάση της αναζήτησης απαιτείται να υπάρξει μια πολύπλοκη απάντηση από διάφορα μέρη της συσκευής, οι ανταποκρίσεις θα πρέπει να μοιραστούν στο χρόνο τυχαία από το 0 έως τα δευτερόλεπτα που καθορίζονται από τον πεδίο επικεφαλίδας MX. Οι συσκευές θα πρέπει να υποθέσουν μια τιμή πεδίου μικρότερη από αυτή που ορίζεται στο πεδίο επικεφαλίδας MX. Έτσι αν το πεδίο επικεφαλίδας MX ορίζει μια τιμή μεγαλύτερη του 5, η συσκευή θα πρέπει να υποθέσει πως περιλαμβάνεται η τιμή 5 ή κάποια μικρότερη της. Οι συσκευές επίσης δε θα πρέπει να σταματούν να απαντούν σε άλλα αιτήματα το διάστημα που περιμένουν να ανταποκριθούν σε ένα άλλο αίτημα.

Για τα multicast μηνύματα M-SEARCH, εάν το αίτημα αναζήτησης δε περιλαμβάνει το πεδίο επικεφαλίδας MX η συσκευή θα πρέπει να απορρίψει το αίτημα. Οποιαδήποτε συσκευή ανταποκρίνεται σε ένα unicast αίτημα θα πρέπει να ανταποκρίνεται μέσα στο χρόνο του 1 δευτερολέπτου. Το URL που καθορίζεται στο πεδίο επικεφαλίδας LOCATION της απάντησης στο M-SEARCH θα πρέπει να είναι εφικτό από το σημείο ελέγχου στο οποίο αποστέλλεται η απάντηση.

Τα μηνύματα ανταπόκρισης θα πρέπει να έχουν την παρακάτω μορφοποίηση:
Οι τιμές στα *italics* αντικαθιστώνται από πραγματικές τιμές.

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
DATE: when response was generated
EXT:
LOCATION: URL for UPnP description for root device
SERVER: OS/version UPnP/1.1 product/version
ST: search target
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update
message
CONFIGID.UPNP.ORG: number used for caching description information
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Δεν υπάρχει περιορισμός στο χρόνο TTL .

Response Line

Θα πρέπει να είναι “HTTP/1.1 200 OK”.

Πεδία επικεφαλίδας

DATE → Το πεδίο αυτό περιέχει πληροφορίες για το πότε έλαβε χώρα μια απάντηση (response) σε μήνυμα M-SEARCH.

EXT→ Το πεδίο αυτό είναι απαραίτητο για λόγους συμβατότητας με την έκδοση UPnP 1.0. Περιλαμβάνεται μόνο όνομα πεδίου και όχι πεδίο τιμής.

ST → Είναι απαιτούμενο πεδίο και η τιμή του περιλαμβάνει το στόχο αναζήτησης. Η απάντηση που αποστέλλεται από τη συσκευή εξαρτάται από την τιμή του πεδίου επικεφαλίδας ST που έχει σταλεί κατά το αίτημα. Σε κάποιες περιπτώσεις η συσκευή θα πρέπει να στείλει πολλαπλά μηνύματα απάντησης όπως ακολουθεί. Αν το λαμβανόμενο πεδίο ST ήταν:

ssdp:all

Απαντά 3+2d+k φορές για την root συσκευή με d ενσωματωμένες συσκευές και s ενσωματωμένες υπηρεσίες και μόνο k ευδιάκριτους τύπους υπηρεσιών. Η τιμή του πεδίου ST θα πρέπει να είναι η ίδια με το πεδίο επικεφαλίδας NT στα μηνύματα με μέθοδο NOTIFY και ssdp:alive.

upnp:rootdevice

Απαντά μια φορά για τη root συσκευή

uuid:*device-UUID*

Απαντά μια φορά για κάθε συσκευή που ταιριάζει στα κριτήρια είτε root είτε ενσωματωμένη.

urn:**schemas-upnp-org**:**device**:*deviceType*:*ver*

Απαντά μια φορά για κάθε συσκευή που ταιριάζει στα κριτήρια, root ή ενσωματωμένη.

urn:**schemas-upnp-org**:**service**:*serviceType*:*ver*

Απαντά μια φορά για κάθε τύπο συσκευής που ταιριάζει στα κριτήρια.

urn:*domain-name*:**device**:*deviceType*:*ver*

Απαντά μια φορά για κάθε συσκευή που ταιριάζει στα κριτήρια, root ή ενσωματωμένη.

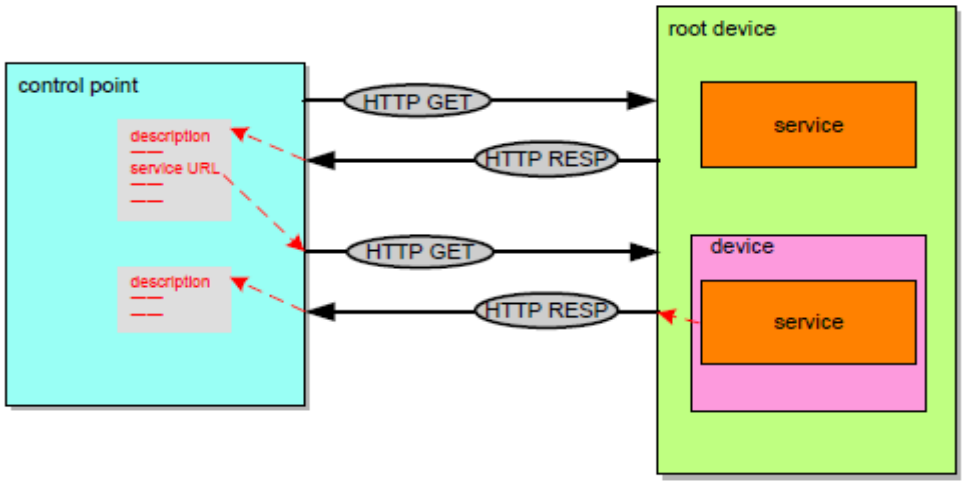
urn:*domain-name*:**service**:*serviceType*:*ver*
Απαντά μια φορά για κάθε τύπο συσκευής που ταιριάζει στα κριτήρια.

Αν υπάρχει κάποιο σφάλμα στο αίτημα αναζήτησης όπως κάποια μη έγκυρη τιμή στο πεδίο επικεφαλίδας MAN, είτε κάποιο παραλειπόμενο MX πεδίο επικεφαλίδας , η συσκευή θα πρέπει να αγνοήσει το αίτημα αυτό.

1.4 Description

Η διαδικασία της περιγραφής (description) είναι το βήμα 2 στην τεχνολογία UPnP. Η περιγραφή ακολουθεί έπειτα από τη διαδικασία addressing (βήμα 0) όπου οι συσκευές αποκτούν διεύθυνση στο δίκτυο και τη διαδικασία discovery (βήμα 1) όπου τα σημεία ελέγχου αναζητούν συσκευές ενδιαφέροντος. Η διαδικασία description ενεργοποιεί τη διαδικασία control (βήμα 3) όπου τα σημεία ελέγχου στέλνουν εντολές στις συσκευές, τη διαδικασία eventing (βήμα 4) όπου τα σημεία ακούν στις αλλαγές της κατάστασης των συσκευών και τέλος τη διαδικασία presentation (βήμα 5) όπου τα σημεία ελέγχου παρέχουν μια διεπαφή ή αλλιώς το γραφικό περιβάλλον της συσκευής προς το χρήστη.

Αφού ένα σημείο ελέγχου έχει ανακαλύψει μια συσκευή ενδιαφέροντος στο δίκτυο γνωρίζει πολύ λίγα πράγματα γι αυτήν. Μόνο πληροφορίες όπως τον τύπο της UPnP συσκευής ή υπηρεσίας, ένα μοναδικό αναγνωριστικό UUID και ένα πεδίο URL για την περιγραφή της συσκευής. Έτσι για να μάθει περισσότερα γι' αυτή και τις δυνατότητες της είτε να αλληλεπιδράσει μαζί της, θα πρέπει να ανακτήσει την περιγραφή της συσκευής και των δυνατοτήτων της μέσω του URL που παρέχεται στο μήνυμα discovery.



Σχήμα 7. Αρχιτεκτονική Description

Η περιγραφή μιας συσκευής χωρίζεται σε δύο λογικά μέρη. Την περιγραφή της συσκευής που εκτείνεται στην περιγραφή των φυσικών και λογικών στοιχείων της και την περιγραφή των υπηρεσιών της που ουσιαστικά αναδεικνύει τις δυνατότητες της. Η περιγραφή μιας UPnP συσκευής περιλαμβάνει πληροφορίες σχετικές με τον κατασκευαστή της όπως είναι το όνομα το μοντέλο, ο σειριακός αριθμός, η επωνυμία του κατασκευαστή, URL που δείχνει στον δικτυακό τόπο του κατασκευαστή κλπ. Για κάθε υπηρεσία που εμπεριέχεται στη συσκευή, η περιγραφή της συσκευής περιλαμβάνει τον τύπο υπηρεσίας, το όνομα της υπηρεσίας, ένα URL για την περιγραφή της υπηρεσίας, ένα URL για έλεγχο (control) και ένα URL για το eventing. Μια περιγραφή συσκευής επίσης περιλαμβάνει την περιγραφή για όλες τις ενσωματωμένες συσκευές της.

Πρέπει να σημειωθεί πως μια μοναδική φυσική συσκευή μπορεί να περιέχει περισσότερες από μία λογικές συσκευές. Αυτές μπορούν να μοντελοποιηθούν σαν μια μοναδική root συσκευή με ενσωματωμένες συσκευές και υπηρεσίες είτε σαν πολλαπλές root συσκευές χωρίς ενσωματωμένες συσκευές. Στην πρώτη περίπτωση, θα υπάρχει μια περιγραφή συσκευής για τη root συσκευή και η περιγραφή αυτή θα περιέχει μια περιγραφή για όλες τις ενσωματωμένες συσκευές. Στη δεύτερη περίπτωση, θα υπάρχουν περισσότερες UPnP περιγραφές συσκευών, μία για κάθε root συσκευή.

Η περιγραφή μιας συσκευής είναι υλοποιημένη από τον προμηθευτή, δίνεται σε μορφή XML κειμένων και βασίζεται σε μία πρότυπη φόρμα UPnP συσκευής, η οποία καθορίζεται από το UPnP Forum. Η περιγραφή μιας υπηρεσίας περιλαμβάνει μια λίστα από εντολές ή ενέργειες στις οποίες οι υπηρεσίες ανταποκρίνονται καθώς και παραμέτρους ή μεταβλητές για κάθε ενέργεια. Μια περιγραφή υπηρεσίας επίσης περιέχει μια λίστα μεταβλητών οι οποίες μοντελοποιούν την κατάσταση της υπηρεσίας την τρέχουσα στιγμή. Όπως η περιγραφή συσκευής, έτσι και η περιγραφή υπηρεσίας είναι υλοποιημένη από τον προμηθευτή δίνεται σε μορφή XML κειμένου και βασίζεται σε μια πρότυπη φόρμα UPnP υπηρεσίας που δίνεται από το UPnP Forum. Στο επόμενο κεφάλαιο WEB SERVICES ακολουθεί επίσης αναλυτική περιγραφή της γλώσσας XML.

Οι UPnP προμηθευτές μπορούν να διαφοροποιήσουν τα προϊόντα τους με περαιτέρω υπηρεσίες συμπεριλαμβάνοντας επιπλέον UPnP υπηρεσίες είτε ενσωματώνοντας επιπρόσθετες συσκευές. Όταν ένα σημείο ελέγχου ανακτά την περιγραφή μιας συγκεκριμένης συσκευής, αυτά τα επιπρόσθετα χαρακτηριστικά είναι επίσης στη διάθεση του control point για τις διαδικασίες ελέγχου (control) και ενημέρωσης συμβάντος (eventing).

Για την ανάκτηση της περιγραφής μιας UPnP συσκευής το σημείο ελέγχου χρησιμοποιώντας το URL που περιέχεται στο discovery μήνυμα εκπέμπει ένα HTTP GET αίτημα προς τη συσκευή και αυτή επιστρέφει το αντίστοιχο κείμενο περιγραφής. Η ανάκτηση μιας UPnP περιγραφής υπηρεσίας είναι παρόμοια διαδικασία όπου χρησιμοποιούνται ξεχωριστά URLs για τα κείμενα περιγραφής της κάθε υπηρεσίας. Εφόσον τουλάχιστον μια από τις discovery ανακοινώσεις (advertisements) δεν έχει λήξει και καμία δεν έχει ακυρωθεί, το σημείο ελέγχου θεωρεί πως η root συσκευή, οι ενσωματωμένες συσκευές και οι υπηρεσίες είναι ακόμη διαθέσιμες. Αν η συσκευή ακυρώσει τουλάχιστον μια από τις ανακοινώσεις ή αν όλες οι ανακοινώσεις λήξουν, το σημείο ελέγχου θα υποθέσει πως η συσκευή και οι υπηρεσίες της δεν είναι πλέον διαθέσιμες. Αν μια συσκευή χρειαστεί να αλλάξει μια από τις περιγραφές της, θα πρέπει να ακυρώσει όλες τις υπάρχουσες ανακοινώσεις και να αποστείλει νέες. Ως εκ τούτου τα σημεία ελέγχου δεν θα πρέπει να θεωρήσουν πως οι περιγραφές συσκευών και υπηρεσιών είναι αμετάβλητες αν μια συσκευή επανεμφανιστεί στο δίκτυο, αλλά θα πρέπει να διαπιστώσουν αν οι περιγραφές έχουν διαφοροποιηθεί εφόσον η τιμή του πεδίου CONFIGID.UPNP.ORG στις τρέχουσες ανακοινώσεις έχει αλλάξει. Όπως και η διαδικασία Discovery, έτσι και η Description παίζει σημαντικό ρόλο στη λειτουργικότητα των συσκευών και των σημείων ελέγχου που χρησιμοποιούν διαφορετικές εκδόσεις της UPnP τεχνολογίας.

1.4.1 Περιγραφή συσκευής

Η περιγραφή UPnP μιας συσκευής περιέχει διάφορες πληροφορίες που αφορούν τους προμηθευτές, ορισμούς για όλες τις ενσωματωμένες συσκευές, URL για την παρουσίαση της συσκευής και λίστες για όλες τις υπηρεσίες συμπεριλαμβανομένων URLs για έλεγχο (control) και ενημέρωση συμβάντος (eventing). Επίσης οι προμηθευτές όπως προαναφέρθηκε μπορούν να προσθέσουν ενσωματωμένες συσκευές και υπηρεσίες στις τυποποιημένες συσκευές.

Παρακάτω ακολουθεί ένα παράδειγμα από το μήνυμα που είναι υπεύθυνο για το description των συσκευών. Τα πεδία σε *italics* αντικαθιστώνται από πραγματικές τιμές. Κάποια από αυτά καθορίζονται από το UPnP Forum (κόκκινο χρώμα) είτε από τον UPnP προμηθευτή (μωβ). Για τις μη τυποποιημένες συσκευές όλα αυτά τα πεδία ορίζονται από τον UPnP προμηθευτή Τα στοιχεία που ορίζονται από την UPnP αρχιτεκτονική των συσκευών είναι με πράσινο χρώμα.

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0"
 configId="configuration number">
    <specVersion>
        <major>1</major>
        <minor>1</minor>
    </specVersion>
    <device>
        <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
        <friendlyName>short user-friendly title</friendlyName>
        <manufacturer>manufacturer name</manufacturer>
        <manufacturerURL>URL to manufacturer site</manufacturerURL>
        <modelDescription>long user-friendly title</modelDescription>
        <modelName>model name</modelName>
        <modelNumber>model number</modelNumber>
        <modelURL>URL to model site</modelURL>
        <serialNumber>manufacturer's serial number</serialNumber>
        <UDN>uuid:UUID</UDN>
        <UPC>Universal Product Code</UPC>
        <iconList>
            <icon>
                <mimetype>image/format</mimetype>
                <width>horizontal pixels</width>
                <height>vertical pixels</height>
                <depth>color depth</depth>
                <url>URL to icon</url>
            </icon>
            <!-- XML to declare other icons, if any, go here -->
        </iconList>
        <serviceList>
            <service>
                <serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
                <serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
                <SCPDURL>URL to service description</SCPDURL>
                <controlURL>URL for control</controlURL>
                <eventSubURL>URL for eventing</eventSubURL>
            </service>
            <!-- Declarations for other services defined by a UPnP Forum working committee
                 (if any) go here -->
            <!-- Declarations for other services added by UPnP vendor (if any) go here -->
        </serviceList>
        <deviceList>
        <!-- Description of embedded devices defined by a UPnP Forum working committee
             (if any) go here -->
        <!-- Description of embedded devices added by UPnP vendor (if any) go here -->
        </deviceList>
        <presentationURL>URL for presentation</presentationURL>
    </device>
```

Παράδειγμα μηνύματος description

Τα σημεία ελέγχου πρέπει να αναγνωρίσουν και να επικοινωνήσουν με τις υπηρεσίες χρησιμοποιώντας τις τιμές του πεδίου serviced πέραν της τιμής που ορίζεται από τον τύπο συσκευής. Αν υπάρχουν πολλές περιπτώσεις υπηρεσιών, το σημείο ελέγχου εξ ορισμού θα χρησιμοποιήσει την περίπτωση αυτή της υπηρεσίας που σχετίζεται με το πεδίο serviced που ορίζεται από τον τύπο της συσκευής.

1.4.2 Περιγραφή υπηρεσίας

Η UPnP περιγραφή υπηρεσίας περιέχει πληροφορίες για τις ενέργειες και τις μεταβλητές που αυτή χρησιμοποιεί, τον τύπο των δεδομένων των πεδίων τιμών και τα χαρακτηριστικά των ενεργειών.

Κάθε υπηρεσία πρέπει να έχει καμία ή περισσότερες ενέργειες ενώ με τη σειρά της κάθε ενέργεια πρέπει να έχει καμία ή περισσότερες μεταβλητές. Κάθε μεταβλητή είναι σχεδιασμένη σαν μεταβλητή εισόδου ή εξόδου. Οι μεταβλητές εισόδου θα

πρέπει να αναφέρονται πρώτες. Κάθε υπηρεσία επίσης έχει μια ή περισσότερες μεταβλητές κατάστασης. Επιπροσθέτως ως αναφορά τον ορισμό των μη τυποποιημένων συσκευών, οι UPnP προμηθευτές μπορούν να προσθέσουν ενέργειες και υπηρεσίες στις τυποποιημένες συσκευές καθώς επίσης να ενσωματώσουν τυποποιημένες συσκευές και υπηρεσίες σε μη τυποποιημένες συσκευές.

Παρακάτω παρουσιάζεται μια λίστα από ένα μήνυμα description των υπηρεσιών όπου με χαρακτήρες *italics* παρουσιάζονται οι θέσεις για τις πραγματικές τιμές . Για τις τυποποιημένες συσκευές ό,τι παρουσιάζεται με κόκκινα γράμματα συμπληρώνεται από το UPnP Forum ενώ οι θέσεις που παρουσιάζονται με μωβ χρώμα συμπληρώνονται από τους προμηθευτές της τεχνολογίας UPnP. Για τις μη τυποποιημένες συσκευές όλες οι θέσεις καθορίζονται από τους UPnP προμηθευτές. Τέλος τα στοιχεία που καθορίζονται από την αρχιτεκτονική των συσκευών UPnP παρουσιάζονται με πράσινο χρώμα.

```
<?xml version="1.0"?>
<scpd
 xmlns="urn:schemas-upnp-org:service-1-0"
 xmlns:dt1="urn:domain-name:more-datatypes"
 <!-- Declarations for other namespaces added by UPnP Forum working committee (if any) go
      here -->
 <!-- The value of the attribute must remain as defined by the UPnP Forum working committee.
      -->
 xmlns:dt2="urn:domain-name:vendor-datatypes"
 <!-- Declarations for other namespaces added by UPnP vendor (if any) go here -->
 <!-- Vendors must change the URN's domain-name to a Vendor Domain Name -->
 <!-- Vendors must change vendor-datatypes to reference a vendor-defined namespace -->
 configId="configuration number">
    <specVersion>
       <major>1</major>
       <minor>1</minor>
    </specVersion>
    <actionList>
       <action>
          <name>actionName</name>
          <argumentList>
             <argument>
                <name>argumentNameIn1</name>
                <direction>in</direction>
                <relatedStateVariable>stateVariableName</relatedStateVariable>
             </argument>
             <!-- Declarations for other IN arguments defined by UPnP Forum working
                  Committee (if any) go here -->
             <argument>
                <name>argumentNameOut1</name>
                <direction>out</direction>
                <retval/>
                <relatedStateVariable>stateVariableName</relatedStateVariable>
             </argument>
             <argument>
                <name>argumentNameOut2</name>
                <direction>out</direction>
                <relatedStateVariable>stateVariableName</relatedStateVariable>
             </argument>
             <!-- Declarations for other OUT arguments defined by UPnP Forum working
                  committee (if any) go here -->
          </argumentList>
       </action>
       <!-- Declarations for other actions defined by UPnP Forum working committee
            (if any)go here -->
```

```
        <!-- Declarations for other actions added by UPnP vendor (if any) go here -->
    </actionList>
    <serviceStateTable>
        <stateVariable sendEvents="yes"|"no" multicast="yes"|"no">
            <name>variableName</name>
            <dataType>basic data type</dataType>
            <defaultValue>default value</defaultValue>
            <allowedValueRange>
                <minimum>minimum value</minimum>
                <maximum>maximum value</maximum>
                <step>increment value</step>
            </allowedValueRange>
        </stateVariable>
        <stateVariable sendEvents="yes"|"no" multicast="yes"|"no">
            <name>variableName</name>
            <dataType type="dt1:variable data type">string</dataType>
            <defaultValue>default value</defaultValue>
            <allowedValueList>
                <allowedValue>enumerated value</allowedValue>
                <!-- Other allowed values defined by UPnP Forum working committee
                    (if any) go here -->
                <!-- Other allowed values defined by vendor (if any) go here -->
            </allowedValueList>
        </stateVariable>
        <stateVariable sendEvents="yes"|"no" multicast="yes"|"no">
            <name>variableName</name>
            <dataType type="dt2:vendor data type">string</dataType>
            <defaultValue>default value</defaultValue>
        </stateVariable>
        <!-- Declarations for other state variables defined by UPnP Forum working committee
            (if any) go here -->
        <!-- Declarations for other state variables added by UPnP vendor (if any) go here -->
    </serviceStateTable>
</scpd>
```

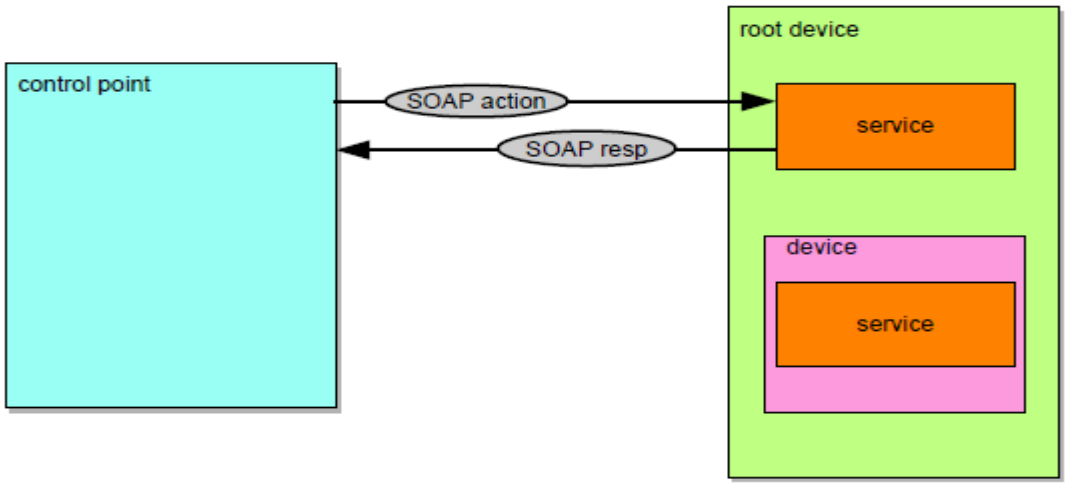
Παράδειγμα μηνύματος description υπηρεσιών

1.4.3 Πρότυπα υπηρεσιών UPnP

Όπως έχει αναφερθεί και πιο πάνω, η UPnP περιγραφή μιας υπηρεσίας υλοποιείται από τον UPnP προμηθευτή δίνεται σε XML και ακολουθεί μια πρότυπη φόρμα υπηρεσίας UPnP όπου δίνεται από το UPnP Forum. Ουσιαστικά, το πρότυπο UPnP υπηρεσίας καθορίζει τον τύπο της υπηρεσίας και κάθε περιγραφή UPnP υπηρεσίας δίνει την υπόσταση σε αυτό το πρότυπο με τους πρόσθετους ορισμούς των προμηθευτών. Το πρότυπο υπηρεσίας υλοποιείται από το UPnP Forum, ενώ η περιγραφή της από τον UPnP προμηθευτή.

1.5 Control

Η διαδικασία Control (έλεγχος) είναι το επόμενο βήμα στην τεχνολογία UPnP. Με δεδομένη τη γνώση για τις συσκευές και τις υπηρεσίες τους, ένα σημείο ελέγχου μπορεί να επικαλεστεί ενέργειες προς τις υπηρεσίες και εν συνεχεία να λαμβάνει απαντήσεις που υποδεικνύουν το αποτέλεσμα της κάθε ενέργειας. Η επίκληση ενεργειών είναι ένα είδος απομακρυσμένης κλήσης διαδικασιών όπου ένα σημείο ελέγχου αποστέλλει την ενέργεια στην υπηρεσία της συσκευής και μόλις αυτή ολοκληρωθεί ή αποτύχει, η υπηρεσία επιστρέφει οποιαδήποτε αποτελέσματα ή λάθη.

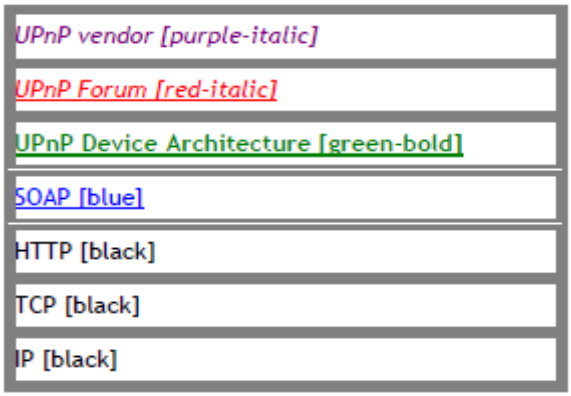


Σχήμα 8. Αρχιτεκτονική διαδικασίας control

Για τον έλεγχο μιας συσκευής, ένα σημείο ελέγχου επικαλείται μια ενέργεια προς την υπηρεσία της συσκευής. Για να το κάνει αυτό, αποστέλλει ένα κατάλληλο μήνυμα ελέγχου στο πεδίο URL για την υπηρεσία που παρέχεται στην περιγραφή της συσκευής. Τα αποτελέσματα της καλούμενης ενέργειας της υπηρεσίας μπορούν να μοντελοποιηθούν με βάση τις αλλαγές στις μεταβλητές που περιγράφουν την κατάσταση της υπηρεσίας κατά την εκτέλεσή τους. Όταν αυτές οι μεταβλητές κατάστασης αλλάζουν, τα σημεία ελέγχου μπορούν να ενημερώνονται για το συμβάν.

Τα μηνύματα ελέγχου καθώς και οι απαντήσεις εκφράζονται σε XML μορφή με χρήση του πρωτοκόλλου SOAP (Simple Object Access Protocol). Οι απαντήσεις στα μηνύματα SOAP κατά τη διαδικασία control θα πρέπει να αποστέλλονται στην ίδια IP διεύθυνση από την οποία λήφθηκε το αίτημα.

Για την επίκληση ενεργειών και επιστροφή τιμών, τα σημεία ελέγχου και οι συσκευές χρησιμοποιούν το ακόλουθο υποσύνολο της UPnP στοίβας πρωτοκόλλου.



Σχήμα 9. Στοίβα πρωτοκόλλου Control.

Στο υψηλότερο επίπεδο, τα μηνύματα ελέγχου περιέχουν πληροφορίες για τους UPnP προμηθευτές όπως τιμές μεταβλητών κ.α. Στα αμέσως χαμηλότερο επίπεδο της στοίβας, η πληροφορίες για τους προμηθευτές συμπληρώνονται από πληροφορίες του UPnP Forum όπως ονόματα ενεργειών , ονόματα μεταβλητών κ.α. Τα παραπάνω μηνύματα ακολουθούν τη μορφοποίηση του πρωτοκόλλου SOAP και μεταφέρονται μέσω HTTP και TCP πάνω από το IP πρωτόκολλο.

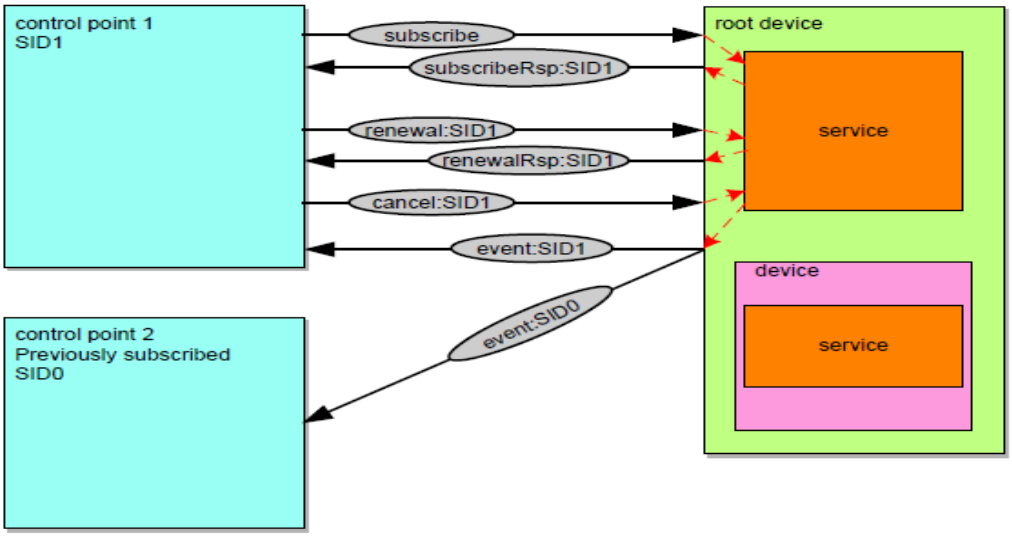
1.6 Eventing

Το βήμα 4 στην τεχνολογία UPnP είναι η διαδικασία eventing. Η διαδικασία αυτή είναι στενά συνδεδεμένη με τη διαδικασία control (βήμα 3) όπου τα σημεία ελέγχου αποστέλλουν ενέργειες στις συσκευές. Μέσω της διαδικασίας eventing τα σημεία ελέγχου ακούν στις αλλαγές κατάστασης των συσκευών. Οι διαδικασίες control και eventing είναι συμπληρωματικές της διαδικασίας presentation (βήμα 5) όπου τα σημεία ελέγχου παρέχουν γραφική διεπαφή για τις συσκευές.

Αφού τα σημεία ελέγχου έχουν ανακαλύψει μια συσκευή και έχουν ανακτήσει μια περιγραφή για αυτή και τις υπηρεσίες της, υπάρχει η ανάγκη της ενημέρωσης για οποιαδήποτε αλλαγή στην κατάσταση της. Όπως αναφέρθηκε παραπάνω στη φάση description, μια UPnP περιγραφή υπηρεσίας περιλαμβάνει μια λίστα ενεργειών, τις απαντήσεις των υπηρεσιών σε αυτές, και μια λίστα τιμών που μοντελοποιούν την κατάσταση της υπηρεσίας την τρέχουσα χρονική στιγμή.

Μόλις υπάρξει κάποια αλλαγή στην κατάσταση των μεταβλητών τότε η υπηρεσία εκπέμπει ενημερώσεις για την αλλαγή ενώ το σημείο ελέγχου θα πρέπει να εγκαθιδρύσει μια συνδρομή ώστε να λάβει την πληροφορία αυτή.

Υπάρχουν δύο τύποι eventing, το unicast όπου τα σημεία ελέγχου όπως προαναφέρθηκε θα πρέπει να εγκαθιδρύσουν μια συνδρομή ώστε να λάβουν τις ανανεώσεις των μεταβλητών και το multicast όπου οι τιμές των μεταβλητών θα πρέπει να καθοριστούν ως multicast γεγονότα και να αποσταλούν μέσω UDP σε οποιονδήποτε μπορεί να τα ακούσει στην multicast διεύθυνση. Αυτός ο τύπος eventing είναι χρήσιμος για την περίπτωση που γεγονότα τα οποία δεν είναι σχετικά με ήδη καθορισμένες UPnP αλληλεπιδράσεις πρέπει να μεταδοθούν προς το σημείο ελέγχου για την ενημέρωση του χρήστη, καθώς επίσης όταν πολλαπλές ελεγχόμενες συσκευές χρειάζεται να ενημερώσουν άλλες ελεγχόμενες συσκευές.

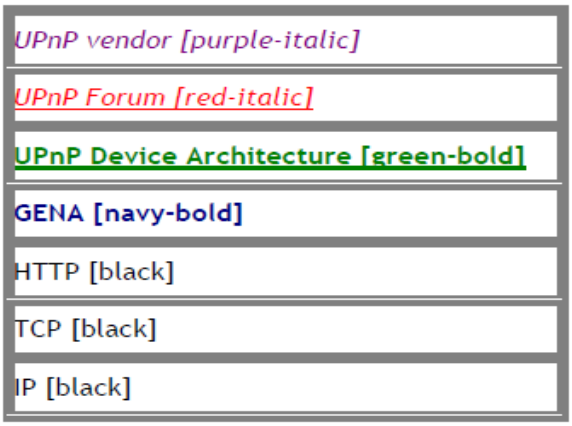


Σχήμα 10. Αρχιτεκτονική Unicast Eventing

Για τη συνδρομή στη διαδικασία eventing ο συνδρομητής (σημείο ελέγχου) αποστέλλει μηνύματα συνδρομής (subscription). Αν η συνδρομή γίνει αποδεκτή, η υπηρεσία της συσκευής (publisher) όπου είναι η πηγή των γεγονότων events, απαντά με μια διάρκεια για τη συνδρομή αυτή. Για να διατηρήσει τη συνδρομή ενεργή ένας συνδρομητής (σημείο ελέγχου) πρέπει να την ανανεώσει πριν αυτή λήξει. Όταν ο συνδρομητής πλέον δε χρειάζεται κάτι άλλο από τη συσκευή (publisher), θα πρέπει να ακυρώσει τη συνδρομή του.

Η υπηρεσία της συσκευής (publisher) ενημερώνει τις αλλαγές στις μεταβλητές κατάστασης αποστέλλοντας ανάλογα event μηνύματα. Τα μηνύματα αυτά περιλαμβάνουν τα ονόματα των μεταβλητών κατάστασης καθώς και τις τρέχουσες τιμές τους εκφρασμένα σε μορφή XML. Ένα επιπλέον αρχικό μήνυμα event αποστέλλεται όταν ο συνδρομητής ξεκινά μια συνδρομή. Για την υποστήριξη περιπτώσεων με πολλαπλά σημεία ελέγχου, η διαδικασία eventing μπορεί να χρησιμοποιηθεί ώστε να διατηρούνται ενημερωμένα τα ενδιαφερόμενα σημεία ελέγχου για τις επιδράσεις των ενεργειών που προκλήθηκαν από άλλα σημεία ελέγχου είτε με τη χρήση άλλων μηχανισμών για έλεγχο των συσκευών.

Τα μηνύματα ενημέρωσης αποστέλλονται σε όλους τους συνδρομητές, επίσης οι συνδρομητές λαμβάνουν μηνύματα event για όλες τις επηρεαζόμενες μεταβλητές για οποιοδήποτε λόγο και έχουν μεταβληθεί. Για την αποστολή και λήψη subscription και event μηνυμάτων τα σημεία ελέγχου και οι υπηρεσίες ακολουθούν το παρακάτω υποσύνολο της UPnP στοίβας πρωτοκόλλου.



Σχήμα 11. Στοίβα πρωτοκόλλου Unicast Eventing

Στο υψηλότερο επίπεδο της στοίβας, τα μηνύματα subscription και event περιέχουν πληροφορίες για τους προμηθευτές UPnP όπως URLs για subscription (συνδρομή), διάρκεια της συνδρομής είτε συγκεκριμένες τιμές μεταβλητών. Στο αμέσως χαμηλότερο επίπεδο της στοίβας, οι πληροφορίες για τους προμηθευτές συμπληρώνονται από πληροφορία η οποία δίνεται από το UPnP Forum, όπως αναγνωριστικά υπηρεσιών ή ονόματα μεταβλητών. Για τα μηνύματα των παραπάνω επιπέδων χρησιμοποιούνται συγκεκριμένα πρωτόκολλα UPnP. Τα μηνύματα παραδίδονται μέσω HTTP με χρήση επιπρόσθετων μεθόδων και πεδίων επικεφαλίδων. Τα HTTP μηνύματα παραδίδονται μέσω TCP πάνω από το πρωτόκολλο IP. Τα χρώματα στις αγκύλες υποδεικνύουν σε ποιο πρωτόκολλο ανήκουν τα πεδία των επικεφαλίδων στα μηνύματα subscription.

1.6.1 Subscription

Η κατάσταση μιας υπηρεσίας έχει επηρεαστεί μόνο εάν έχουν επηρεαστεί μία ή περισσότερες από τις μεταβλητές κατάστασης. Στην περίπτωση αυτή, η υπηρεσία εκπέμπει event μηνύματα σε συνδρομητές όπου ενδιαφέρονται. Η υπηρεσία (publisher) διατηρεί μια λίστα συνδρομητών (subscribers) η οποία έχει τις εξής πληροφορίες: Ένα μοναδικό αναγνωριστικό για κάθε συνδρομή, μια διεύθυνση URL για την παράδοση των event μηνυμάτων, ένα event key όπου είναι 0 για το αρχικό μήνυμα συμβάντος (event) και αυξάνεται κατά ένα για κάθε επόμενο μήνυμα μέσω του οποίου μπορεί να επιβεβαιωθεί πως δεν έχουν υπάρξει event μηνύματα που έχουν χαθεί, διάρκεια συνδρομής και τέλος η υποστηριζόμενη έκδοση http από τον συνδρομητή. Μια multi homed υπηρεσία επίσης θα πρέπει να διατηρεί πληροφορία στην UPnP διεπαφή στην οποία έχει παραληφθεί το κάθε μήνυμα subscription. Η ίδια διεπαφή θα πρέπει να χρησιμοποιηθεί για την εκπομπή των μηνυμάτων subscription στον αντίστοιχο συνδρομητή.

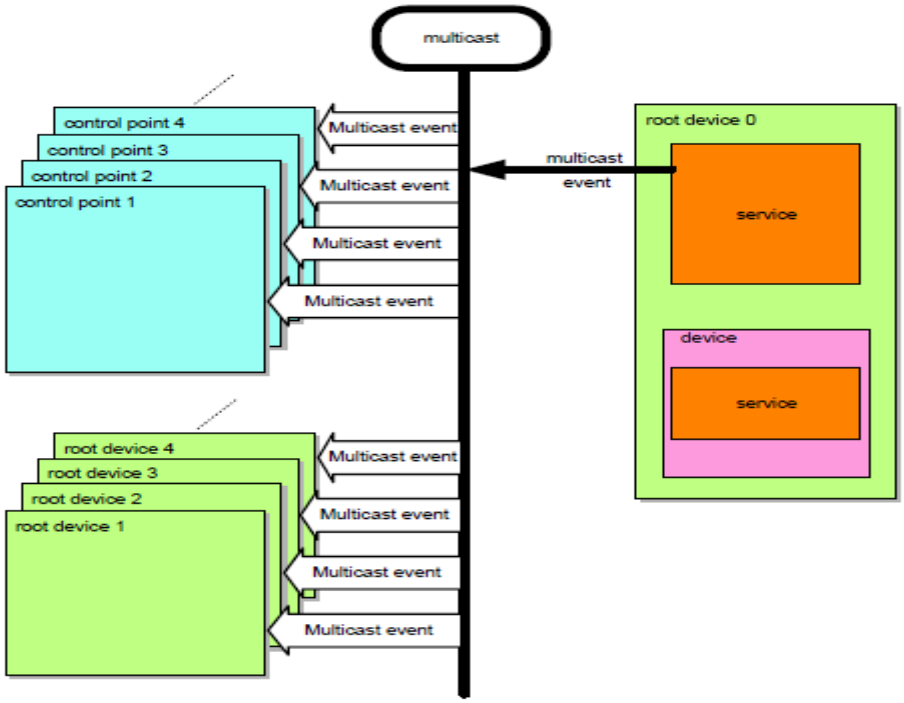
Η υπηρεσία (publisher) θα πρέπει να δέχεται όσο περισσότερες συνδρομές μπορεί, λαμβάνοντας υπόψη ότι ο αριθμός των event μηνυμάτων που χρειάζονται να μεταφερθούν ανά event αυξάνει γραμμικά με τον αριθμό συνδρομών. Η λίστα των συνδρομητών ανανεώνεται μέσω των μηνυμάτων subscription, renewal και cancelation. Για να εγκαθιδρύσει μια συνδρομή eventing για μια υπηρεσία, ο συνδρομητής (σημείο ελέγχου) αποστέλλει ένα μήνυμα subscription το οποίο περιλαμβάνει ένα URL για την υπηρεσία (publisher), ένα αναγνωριστικό για την υπηρεσία, και ένα URL παράδοσης για τα μηνύματα event. Το μήνυμα subscription επίσης μπορεί να περιλαμβάνει μια ζητούμενη διάρκεια για τη συνδρομή. Το URL και το αναγνωριστικό της υπηρεσίας περιέχονται στα μηνύματα περιγραφής (description).

Αν η συνδρομή είναι αποδεκτή τότε η υπηρεσία (publisher) απαντά με ένα μοναδικό αναγνωριστικό για τη συνδρομή και μια διάρκεια. Η διάρκεια θα πρέπει να επιλέγεται έτσι ώστε να ανταποκρίνεται στην υπόθεση για το πόσο συχνά ένα σημείο ελέγχου αποσυνδέεται από το δίκτυο. Αν το σημείο ελέγχου αποσυνδέεται κάθε λίγα λεπτά, η διάρκεια θα πρέπει να είναι εξίσου μικρή, επιτρέποντας στην υπηρεσία(publisher) να διαπιστώσει άμεσα κάθε λήξη συνδρομής. Αν το σημείο ελέγχου είναι σχεδόν μόνιμο στο δίκτυο, η διάρκεια θα πρέπει να είναι αρκετά μεγαλύτερη ελαχιστοποιώντας έτσι την επεξεργασία και την κυκλοφορία στο δίκτυο που σχετίζεται με τις ανανεώσεις των συνδρομών. Το συντομότερο δυνατό έπειτα από την αποδοχή η υπηρεσία(publisher) αποστέλλει επίσης το αρχικό μήνυμα event στον συνδρομητή. Αυτό το μήνυμα περιλαμβάνει ονόματα και τρέχουσες τιμές για όλες τις επηρεαζόμενες μεταβλητές. Αυτό το αρχικό μήνυμα αποστέλλεται ακόμη και αν το σημείο ελέγχου σταματήσει τη συνδρομή πριν αυτό παραδοθεί. Η συσκευή θα πρέπει να εξασφαλίσει πως το σημείο ελέγχου έλαβε την απάντηση στο αίτημα subscription πριν αποστείλει το αρχικό μήνυμα ώστε να εξασφαλίσει πως το σημείο ελέγχου έχει λάβει το SID (subscription id) και να μπορεί έτσι να συσχετίσει το event μήνυμα με το subscription.

Για να διατηρήσει ενεργή τη συνδρομή, ο συνδρομητής θα πρέπει να ανανεώσει την ανανεώσει πριν αυτή λήξει αποστέλλοντας ένα μήνυμα renewal. Τα μήνυμα renewal αποστέλλεται στο ίδιο URL όπως και το subscription μήνυμα, ωστόσο δεν περιλαμβάνει το URL παράδοσης των event μηνυμάτων περιλαμβάνει όμως το αναγνωριστικό της συνδρομής subscription.Η απάντηση για ένα renewal μήνυμα είναι η ίδια με αυτή του μηνύματος subscription. Αν η συνδρομή λήξει, το αναγνωριστικό της καθίσταται μη έγκυρο και η υπηρεσία (publisher) σταματά την αποστολή event μηνυμάτων προς το συνδρομητή ενώ επίσης κάνει εκκαθάριση στη λίστα των συνδρομητών. Αν ένας συνδρομητής προσπαθήσει να στείλει ένα οποιοδήποτε μήνυμα και όχι subscription, η υπηρεσία (publisher) πρέπει να απορρίψει το μήνυμα διότι το αναγνωριστικό της συνδρομής (subscription) είναι μη

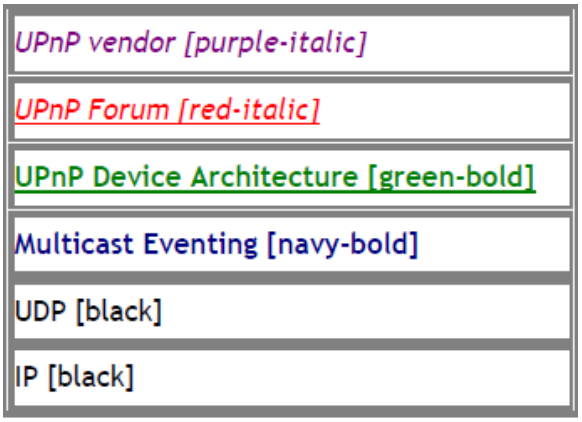
έγκυρο. Όταν ένας συνδρομητής δε χρειάζεται πλέον να αλληλεπιδράσει μέσω διαδικασίας eventing με κάποια υπηρεσία, θα πρέπει να ακυρώσει τη συνδρομή. Με την ακύρωση μιας συνδρομής, αυτομάτως μειώνεται και η φόρτιση υπηρεσιών, σημείων ελέγχου και του δικτύου γενικότερα. Σε περίπτωση που ένας συνδρομητής αφαιρεθεί από το δίκτυο απότομα, πιθανό να μην μπορεί να αποστείλει ακυρωτικό μήνυμα. Ως εναλλακτική σε αυτό, η συνδρομή θα λήξει από μόνη της εάν δεν ανανεωθεί.

Συνίσταται ιδιαιτέρως, οι συνδρομητές να παρακολουθούν τα discovery μηνύματα των υπηρεσιών (publishers). Εάν η υπηρεσία (publisher) ακυρώσει τις ανακοινώσεις της (advertisements) ή η τιμή του BOOTID.UPNP.ORG αυξηθεί χωρίς να έχει προηγηθεί ssdp:update μήνυμα το οποίο να ταιριάζει στο πεδίο επικεφαλίδας NEXTBOOTID.UPNP.ORG, οι συνδρομητές θα πρέπει να υποθέσουν πως η συνδρομή τους έχει λήξει.



Σχήμα 12. Αρχιτεκτονική Multicast Eventing

Η υπηρεσία (publisher) ενημερώνει τις αλλαγές των μεταβλητών κατάστασης αποστέλλοντας multicast event μηνύματα. Τα multicast μηνύματα event περιλαμβάνουν τα ονόματα μιας ή περισσοτέρων μεταβλητών και την τρέχουσα τιμή των μεταβλητών αυτών εκφραζόμενα σε μορφή XML. Για την αποστολή και λήψη event μηνυμάτων, τα σημεία ελέγχου και οι υπηρεσίες ακολουθούν το παρακάτω υποσύνολο UPnP στοίβας πρωτοκόλλου.



Σχήμα 13. Στοίβα πρωτοκόλλου Multicast Eventing

Στο υψηλότερο επίπεδο της στοίβας τα multicast event μηνύματα περιέχουν πληροφορία για τους προμηθευτές όπως καθορισμένες μεταβλητές κατάστασης από τους προμηθευτές και τιμές τους. Στο αμέσως χαμηλότερο επίπεδο η πληροφορία των προμηθευτών συμπληρώνεται από πληροφορία που δίδεται από το UPnP Forum αναγνωριστικά υπηρεσιών και ονόματα μεταβλητών. Για τα παραπάνω μηνύματα χρησιμοποιείται HTTP πρωτόκολλο διαμόρφωσης επικεφαλίδας και σώματος και η μετάδοσή τους καθορίζεται μέσω UDP sockets.
Το multicast eventing είναι εγγενώς αναξιόπιστο λόγω του ότι βασίζεται σε UDP. Επιπροσθέτως, υπάρχει μια μεγάλη πιθανότητα απώλειας μηνυμάτων με μεγάλο μέγεθος πακέτων. Για την αύξηση της πιθανότητας επιτυχούς μεταφοράς κάθε πακέτο μπορεί να μεταδοθεί παραπάνω από μια φορά.

1.6.2 Event μηνύματα

Μια υπηρεσία δηλώνει τις αλλαγές στις μεταβλητές κατάστασης αποστέλλοντας μηνύματα event. Αυτά τα μηνύματα περιέχουν ονόματα των μεταβλητών κατάστασης και την τρέχουσα τιμή τους. Τα μηνύματα event θα πρέπει να αποστέλλονται χρονικά με τέτοιο τρόπο ώστε να ενημερώνονται οι συνδρομητές με ακρίβεια για την κατάσταση της υπηρεσίας και να μπορούν να παρέχουν μια διεπαφή χρήστη η οποία να ανταποκρίνεται στην κατάστασή της. Αν η τιμή μιας η περισσοτέρων μεταβλητών αλλάξει την ίδια στιγμή, η υπηρεσία (publisher) θα πρέπει να συμπεριλάβει τις αλλαγές αυτές σε ένα μοναδικό συμβάν (event) μειώνοντας έτσι την επεξεργασία και την κίνηση στο δίκτυο.

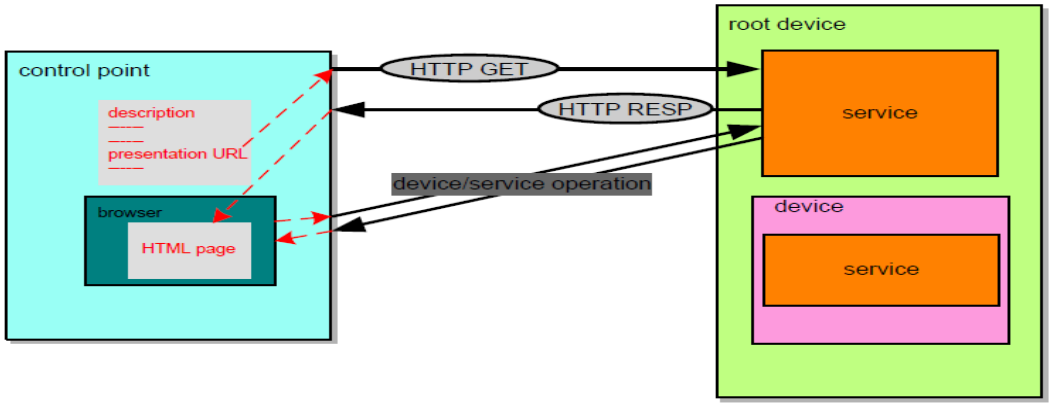
Όπως αναφέρθηκε και πιο πάνω κατά την έναρξη μια συνδρομής αποστέλλεται ένα αρχικό μήνυμα. Το μήνυμα αυτό περιέχει ονόματα και τιμές από όλες τις επηρεαζόμενες μεταβλητές και θα πρέπει να αποσταλεί το συντομότερο αφότου η υπηρεσία (publisher) αποδεχτεί μια συνδρομή. Επιπροσθέτως το μήνυμα θα πρέπει

να αποσταλεί ακόμη και αν το σημείο ελέγχου αποχωρήσει από τη συνδρομή πριν αυτό παραδοθεί. Τα unicast όπως και τα multicast μηνύματα είναι χαρακτηρισμένα με ένα event (key) κλειδί. Στο unicast eventing, θα πρέπει να διατηρείται ένα ξεχωριστό event key για κάθε συνδρομη (subscription) διευκολύνοντας έτσι την εύρεση λαθών. Το event key μια συνδρομή αρχικοποιείται στο 0 τη στιγμή που η υπηρεσία αποστέλλει το αρχικό μήνυμα. Για κάθε επαναλαμβανόμενο event μήνυμα η υπηρεσία αυξάνει κατά ένα το event key συμπεριλαμβάνοντας το στο event μήνυμα. Το ίδιο ακριβώς συμβαίνει και στην περίπτωση των multicast events. Επίσης σε περίπτωση που το κλειδί φτάσει στην τιμή 4294967295, η τιμή επιστρέφει στο 1 και όχι στο 0.

1.7 Presentation

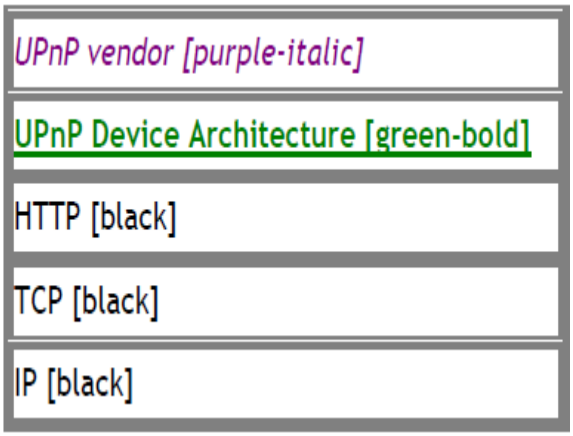
Η παρουσίαση (Presentation) είναι το βήμα 5 στην τεχνολογία UPnP. Η παρουσίαση παρέχει μια διεπαφή στο χρήστη βασισμένη σε HTML για τη διαχείριση είτε την προβολή της κατάστασης της συσκευής. Θα πρέπει να σημειωθεί επίσης πως ενώ όλες τις παραπάνω λειτουργίες θα πρέπει να τις υλοποιεί κάθε UPnP συσκευή, το στάδιο της παρουσίασης είναι προαιρετικό.

Αφού ένα σημείο ελέγχου έχει ανακαλύψει μια συσκευή (βήμα 1) και έχει ανακτήσει περιγραφή για αυτή (βήμα 2) έχει τη δυνατότητα να ξεκινήσει την λειτουργία της παρουσίασης. Εάν η συσκευή παρέχει ένα URL για παρουσίαση, το σημείο ελέγχου μπορεί να ανακτήσει μια σελίδα από το URL αυτό, να τη φορτώσει σε ένα browser και ανάλογα με τις δυνατότητες της σελίδας να παρέχει στο χρήστη τη δυνατότητα για προβολή ή και διαχείριση της κατάστασης της συσκευής.



Σχήμα 14. Αρχιτεκτονική Presentation

Η ανάκτηση μιας σελίδας παρουσίασης (presentation) είναι μια απλή διαδικασία βασισμένη στο HTTP κατά την οποία γίνεται χρήση του παρακάτω υποσυνόλου της UPnP στοίβας πρωτοκόλλου.



Σχήμα 15. Στοίβα πρωτοκόλλου Presentation

Στο υψηλότερο επίπεδο της στοίβας , η σελίδα της παρουσίασης (presentation) καθορίζεται από τον UPnP προμηθευτή. Στο αμέσως χαμηλότερο επίπεδο, η αρχιτεκτονική UPnP συσκευών καθορίζει πως η σελίδα αυτή είναι γραμμένη σε HTML. Η σελίδα παραδίδεται μέσω HTTP πάνω από τα πρωτόκολλα TCP και IP.

Για να ανακτήσει μια σελίδα παρουσίασης (presentation), το σημείο ελέγχου αποστέλλει ένα HTTP GET αίτημα στο presentation URL, και η συσκευή επιστρέφει μια σελίδα παρουσίασης (presentation). Οι απαντήσεις στα αιτήματα HTTP GET θα πρέπει να αποστέλλονται κάνοντας χρήση της ίδιας διεύθυνσης στην ίδια διεπαφή όπου λήφθηκε το αίτημα HTTP GET. Σε αντίθεση με τα UPnP πρότυπα συσκευών και υπηρεσιών καθώς και τους τύπους συσκευών και υπηρεσιών, οι δυνατότητες της presentation σελίδας καθορίζονται εξ ολοκλήρου από τον UPnP προμηθευτή.

2. WEB SERVICES

2.1 Εισαγωγή

Το Web ξεκίνησε αρχικά να υποστηρίζει την αλληλεπίδραση απλών κειμένων, γραφικών και όχι σε ικανοποιητικό βαθμό την αλληλεπίδραση λογισμικού και τη μεταφορά μεγάλου όγκου δεδομένων. Για το σκοπό αυτό ήταν απαραίτητη μια πιο αποδοτική μέθοδος για ιδιωτική είτε εταιρική χρήση με την οποία οι εφαρμογές να μπορούν να επικοινωνούν απευθείας μεταξύ τους και στην ουσία να συνενώνονται σαν να ήταν μέρος ενός πληροφοριακού συστήματος.

Τα Web Services είναι μια καινοτόμος αρχιτεκτονική όπου παρέχει τη δυνατότητα δημιουργίας και χρήσης ηλεκτρονικών υπηρεσιών στο διαδίκτυο με απλό και οικονομικό τρόπο. Έχουν επηρεάσει ριζικά τους κανόνες και την εξέλιξη του διαδικτύου. Διασυνδέουν προγράμματα που είναι κατανεμημένα παγκόσμια και δίνουν τη δυνατότητα μεταφοράς μεγάλου όγκου δεδομένων με τρόπο αποδοτικότερο από ότι στο παρελθόν. Αυτό έχει ως αποτέλεσμα την γρηγορότερη και πιο παραγωγική επικοινωνία τόσο για τις επιχειρήσεις όσο και για τους καταναλωτές.

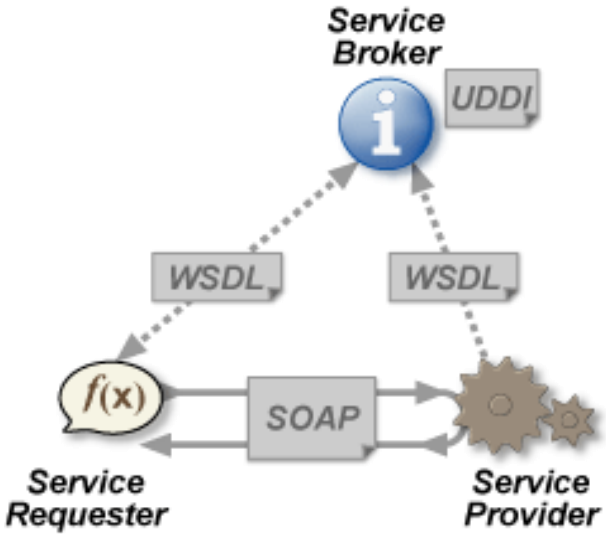
Υπάρχουν πολλοί ορισμοί για το τι είναι Web Services. Ένας πολύ καλός ορισμός που έρχεται από την IBM χαρακτηρίζει *τα* Web Services ως μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και προγραμματιστικής γλώσσας. Σύμφωνα με τον ορισμό αυτό ένα Web Service είναι μια διεπαφή λογισμικού που περιγράφει ένα σύνολο λειτουργιών οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για την περιγραφή μιας λειτουργίας προς εκτέλεση καθώς και των δεδομένων προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα Web Services της οποίας οι οντότητες αλληλεπιδρούν μεταξύ τους, καθορίζει μια εφαρμογή Web Service.

Σύμφωνα με τη Microsoft κάποια βασικά χαρακτηριστικά των Web Services είναι τα παρακάτω:

- Παρέχουν χρήσιμη λειτουργικότητα σε χρήστες του διαδικτύου μέσα από ένα πρότυπο δικτυακό πρωτόκολλο, όπου στις περισσότερες περιπτώσεις είναι το SOAP(Simple Object Access Protocol).
- Παρέχουν ένα τρόπο για λεπτομερή περιγραφή των διεπαφών τους έτσι ώστε ο χρήστης να μπορεί να δημιουργήσει μια εφαρμογή πελάτη η οποία να είναι σε θέση να επικοινωνήσει μαζί τους.
- Καταχωρούνται μέσω UDDI (Universal Discovery Description and Integration) ώστε οι χρήστες να μπορούν να τα εντοπίσουν εύκολα.

Ο οργανισμός World Wide Web Consortium (W3C) που ορίζει τα πρότυπα των Web Services, χαρακτηρίζει ένα Web Service ως ένα σύστημα λογισμικού που ταυτοποιείται από ένα URL όπου οι δημόσιες διεπαφές του (interfaces) και συνδέσεις του (bindings) ορίζονται και περιγράφονται μέσω XML. Το σύστημα αυτό μπορεί να ανακαλυφθεί από άλλα συστήματα λογισμικού και αυτά με τη σειρά τους είναι σε θέση να αλληλεπιδράσουν με το Web Service χρησιμοποιώντας XML μηνύματα που μεταφέρονται μέσω πρωτοκόλλων του διαδικτύου.

Ένας άλλος χρήσιμος ορισμός θα χαρακτήριζε ένα web service ως ένα λογισμικό προσβάσιμο από το web ή το intranet μιας επιχείρησης μέσω ενός URL, που προσπελαύνεται μέσω XML πρωτοκόλλων, όπως είναι το SOAP και αποστέλλεται μέσω πρωτοκόλλων HTTP. Ένας πελάτης μπορεί να προσπελάσει το web service μέσω των interface και bindings του, τα οποία ορίζονται μέσω XML αρχείων όπως είναι ένα αρχείο (WSDL) Web Services Definition Language.



Σχήμα 16. Η δομή των Web Services

Σύμφωνα με όλα τα παραπάνω λοιπόν τα Web Services ορίζουν μια αρχιτεκτονική κατανεμημένων συστημάτων αποτελούμενη από πολλά διαφορετικά υπολογιστικά συστήματα τα οποία επικοινωνούν μέσω του δικτύου ώστε να δημιουργήσουν ένα σύστημα. Αποτελούνται από ένα σύνολο προτύπων τα οποία επιτρέπουν την υλοποίηση κατανεμημένων εφαρμογών με τη χρήση διαφορετικών εργαλείων προερχόμενα από διάφορους προμηθευτές. Οι εφαρμογές χρησιμοποιούν ένα συνδυασμό από ενότητες λογισμικού (software modules) οι οποίες καλούνται από συστήματα που ανήκουν σε διαφορετικά τμήματα ενός οργανισμού είτε και σε άλλους οργανισμούς.

Τα Web Services αποτελούν μια εξέλιξη του Web και βασισμένα σε προϋπάρχοντα κατανεμημένα υπολογιστικά περιβάλλοντα με σκοπό την δυνατότητα της διαλειτουργικότητας μεταξύ των εφαρμογών παρέχουν ένα πρότυπο για το πώς οι

εφαρμογές θα επικοινωνήσουν με άλλες εφαρμογές στο δίκτυο, ανεξάρτητα από τη προγραμματιστική γλώσσα ή το λειτουργικό σύστημα. Οι εφαρμογές που βασίζονται στα Web Services μπορούν χειριστούν απλά αιτήματα για πληροφορία είτε να πραγματοποιούν πολύπλοκες επιχειρηματικές διαδικασίες. Έτσι οι χρήστες μπορούν όχι απλά να χρησιμοποιούν προγραμματιστικά το δίκτυο αλλά και να προσπελάσουν τις υπηρεσίες αυτές και τη λειτουργικότητά τους.

Αυτό συνεπάγεται πως η τεχνολογία Web Services σε περιπτώσεις εταιρικής χρήσης παρέχει τη δυνατότητα σε επιχειρήσεις να επεκτείνουν και να επαναχρησιμοποιήσουν την υπάρχουσα λειτουργικότητα των συστημάτων τους μειώνοντας κατά αυτόν τον τρόπο το λειτουργικό τους κόστος.

2.2 Δομή και αρχιτεκτονική των Web Services

2.2.1 Βασικοί ρόλοι στην αρχιτεκτονική Web Services

Η αρχιτεκτονική των Web Services είναι βασισμένη στην αλληλεπίδραση μεταξύ τριών οντοτήτων: Η οντότητα που ζητά την υπηρεσία (service requestor), η οντότητα που παρέχει την υπηρεσία (service provider) και η οντότητα του καταλόγου υπηρεσιών (service registry).

Πιο συγκεκριμένα η κάθε μία οντότητα έχει τους εξής ρόλους:

Service Requestor:
Είναι ο αιτούμενος (πελάτης) όπου εκκινεί την όλη διαδικασία. Ρόλος του αρχικά είναι η αναζήτηση της κατάλληλης περιγραφής μιας υπηρεσίας από της καταχωρήσεις της υπηρεσίας καταλόγου. Εν συνεχεία αφού εντοπίσει τον επιθυμητό πάροχο της υπηρεσίας εγκαθιδρύει μια σύνδεση μαζί του, καλεί την υπηρεσία και λαμβάνει αποτελέσματα. Οι ενέργειες του service requestor μπορούν να επιτευχθούν με τη χρήση ενός browser είτε με λογισμικό το οποίο μπορεί και να μην παρέχει γραφική διεπαφή χρήστη.

Service Provider:
Είναι η οντότητα που παρέχει την υπηρεσία στο διαδίκτυο δημοσιοποιώντας την περιγραφή της στην οντότητα του καταλόγου υπηρεσιών. Επιπρόσθετος ρόλος του παρόχου υπηρεσιών είναι η λήψη των μηνυμάτων κλήσης των αιτούμενων για την υπηρεσία και η παροχή των αποτελεσμάτων.

Service Registry:
Περιέχει καταχωρήσεις των περιγραφών των δημοσιευμένων υπηρεσιών διαδικτύου. Στις περιγραφές παρέχονται επίσης οι τρόποι αναζήτησης της κάθε υπηρεσίας.

Η περιγραφή μιας υπηρεσίας δημοσιοποιείται από τον πάροχο της υπηρεσίας στον κατάλογο υπηρεσιών και ουσιαστικά περιέχει την πληροφορία που ο αιτούμενος πρέπει να γνωρίζει προκειμένου να κάνει κλήση της υπηρεσίας.
Στα Web Services τα SOAP, UDDI και WSDL αντιπροσωπεύουν τους ρόλους που αναφέρθηκαν παραπάνω. Πιο αναλυτικά:

Μέσω του SOAP (Simple Object Access Protocol) επιτυγχάνεται η αποστολή μηνυμάτων σε στις διάφορες ενότητες λογισμικού. Το UDDI (Universal Discovery Description and Integration) είναι ο κατάλογος όπου βρίσκονται καταχωρημένες οι περιγραφές των υπηρεσιών, και αποτελεί ουσιαστικά τη βάση αναζήτησης για τον εντοπισμό της εκάστοτε υπηρεσίας. Μέσω WSDL (Web Service Definition Language) επιτυγχάνεται η περιγραφή των διαφορετικών υπηρεσιών στο UDDI.

2.2.2 Βασικές λειτουργίες στην αρχιτεκτονική Web Services

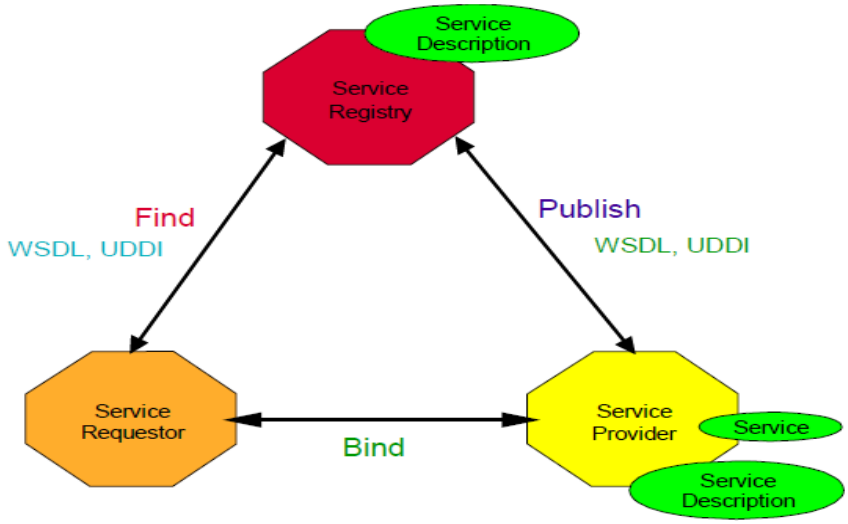
Για μια εφαρμογή που εκμεταλλεύεται την τεχνολογία Web Services, τρεις είναι οι βασικές λειτουργίες που λαμβάνουν χώρο: Η δημοσίευση της περιγραφής της εκάστοτε υπηρεσίας, η αναζήτηση για την εύρεση των περιγραφών της υπηρεσίας και η σύνδεση ή κλήση των υπηρεσιών με βάση την περιγραφή τους. Πιο αναλυτικά αυτές είναι:

- Publish: Για να είναι προσβάσιμη η περιγραφή μιας υπηρεσίας, θα πρέπει να δημοσιοποιηθεί έτσι ώστε ο αιτών της υπηρεσίας να μπορεί να τη βρει. Έτσι από τον πάροχο της υπηρεσίας γίνεται η καταχώρηση της περιγραφής για την υπηρεσία με χρήση της γλώσσας περιγραφής WSDL.

- Find: Αφορά την αναζήτηση και εύρεση της περιγραφής της υπηρεσίας στους καταλόγους UDDI. Κατά τη διαδικασία της εύρεσης, ο αιτών της υπηρεσίας ανακτά μια περιγραφή της υπηρεσίας απευθείας είτε ζητά στον κατάλογο των καταχωρήσεων τον τύπο της υπηρεσίας. Η λειτουργία find μπορεί να συμπεριληφθεί σε δύο διαφορετικές φάσεις για τον service requestor: Στο χρόνο σχεδίασης για την ανάκτηση της διεπαφής της περιγραφής της υπηρεσίας για την ανάπτυξη του λογισμικού, και το χρόνο εκτέλεσης για την ανάκτηση της σύνδεσης με την υπηρεσία και την τοποθεσία της περιγραφής για επίκληση.

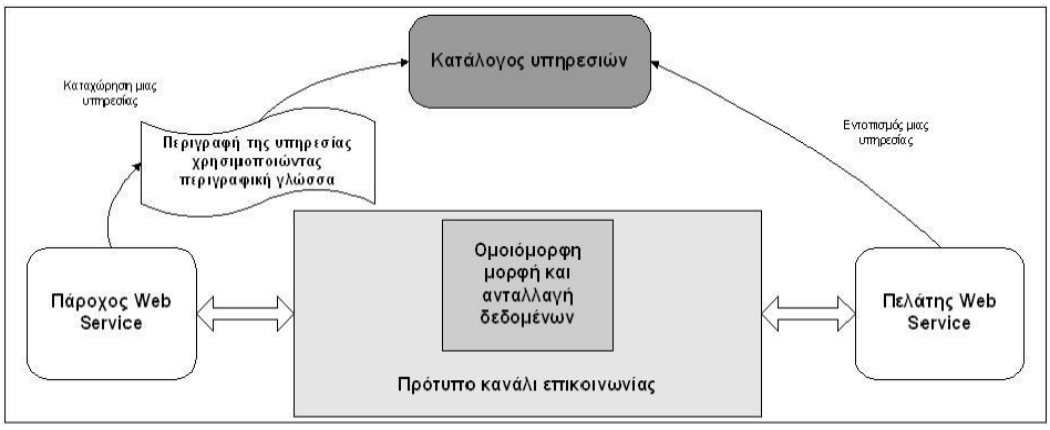
- Bind: Είναι η διαδικασία εγκαθίδρυσης σύνδεσης μεταξύ του αιτούντος (service requestor) και παρόχου της υπηρεσίας (service provider), και εν συνεχεία η κλήση της επιθυμητής υπηρεσίας και η αποστολή των ανάλογων αποτελεσμάτων. Μια υπηρεσία χρειάζεται να καλεστεί. Στη λειτουργία bind ο αιτών της υπηρεσίας καλέι ή ξεκινά μια αλληλεπίδραση με την υπηρεσία κατά το χρόνο εκτέλεσης κάνοντας χρήση των πληροφοριών binding που παρέχονται στην περιγραφή της υπηρεσίας ώστε να εντοπίσει να επικοινωνήσει και να καλέσει την υπηρεσία.

Η επίτευξη των παραπάνω γίνεται με την ανταλλαγή μηνυμάτων SOAP μεταξύ των service requestor, service registry και service provider η οποία είναι βασισμένη σε γλώσσα XML.



Σχήμα 17. Βασικοί ρόλοι της αρχιτεκτονικής Web Services και η μεταξύ τους αλληλεπίδραση

Βάση της παραπάνω αρχιτεκτονικής καθίσταται δυνατή η ανάπτυξη εφαρμογών που μπορούν να κατανεμηθούν και να προσπελαστούν μέσα σε ένα δίκτυο, καθώς επίσης υπάρχουσες εφαρμογές μπορούν να μετασχηματιστούν σε υπηρεσίες που εξυπηρετούν άλλες εφαρμογές. Ωστόσο για την υλοποίησή μιας αρχιτεκτονικής σαν αυτή, προκύπτουν κάποιες απαιτήσεις . Απαιτείται η ύπαρξη μηχανισμών μέσω των οποίων παρέχεται η δυνατότητα προσπέλασης των υπηρεσιών και των καταλόγων από τους πελάτες, η δυνατότητα καταχώρησης των υπηρεσιών και η αναζήτηση τους στους καταλόγους από τους πελάτες και τέλος η δυνατότητα των υπηρεσιών να παραθέτουν τις διεπαφές τους στο δίκτυο και οι πελάτες να είναι σε θέση να τις προσπελαύνουν.



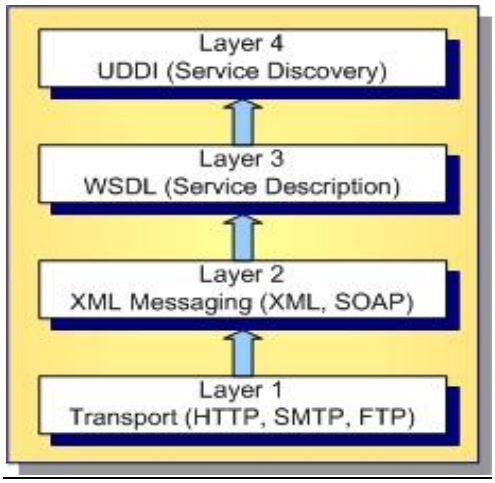
Σχήμα 18. Η αρχιτεκτονική των Web Services

2.2.3 Στοίβα πρωτοκόλλου των Web Services

Τα Web Services για να μεταφέρουν ή να μετατρέψουν την πληροφορία ανάμεσα σε διαφορετικά υπολογιστικά περιβάλλοντα απαιτούν διάφορες συσχετιζόμενες τεχνολογίες που τοποθετούνται σε διαφορετικό επίπεδα. Παρακάτω εξηγείται η λειτουργικότητα της στοίβας πρωτοκόλλων των Web Services.

Η στοίβα πρωτοκόλλου των Web Services περιλαμβάνει τα ακόλουθα βασικά επίπεδα:

- Επίπεδο μεταφοράς (transport layer) αφορά στη μεταφορά των μηνυμάτων μεταξύ των εφαρμογών
- Επίπεδο XML messaging αφορά στην κωδικοποίηση των μηνυμάτων σε XML ώστε να είναι κατανοητά για οποιονδήποτε πελάτη είτε πάροχο.
- Επίπεδο WSDL (service Description) στο οποίο παρέχεται η περιγραφή της παρεχόμενης υπηρεσίας
- UDDI επίπεδο (service Discovery) αφορά στη συγκέντρωση των υπηρεσιών σε ένα κοινό μητρώο καταχώρησης με σκοπό τον εντοπισμό τους



Σχήμα 19. Στοίβα πρωτοκόλλου των Web Services

Αρχικά και ως αναφορά το πρώτο επίπεδο (επίπεδο μεταφοράς) απαιτείται ένα πρωτόκολλο για τη μεταφορά των XML μηνυμάτων μεταξύ των εφαρμογών όπως το HTTP (Hypertext Transfer Protocol) το οποίο χρησιμοποιείται συνηθέστερα, το SMTP (Simple Mail Transport Protocol), και το FTP(File Transfer Protocol).

Για την οποιαδήποτε κλήση και απόκριση της υπηρεσίας θα πρέπει να υπάρχει μια κοινή γλώσσα ώστε να επιτυγχάνεται η επικοινωνία μεταξύ των παρόχων και αιτούντων των υπηρεσιών. Έτσι το δεύτερο επίπεδο της στοίβας πρωτοκόλλου των Web Services (XML messaging) βασίζεται σε ένα μοντέλο XML. Το επίπεδο αυτό

είναι υπεύθυνο για την κωδικοποίηση των μηνυμάτων σε μια κοινή XML μορφοποίηση ώστε να είναι κατανοητά. Το επίπεδο αυτό περιλαμβάνει τα XML-RPC και SOAP.

Το επίπεδο WSDL (service description) αντιπροσωπεύει ένα τρόπο καθορισμού μιας δημόσιας διεπαφής για την web υπηρεσία. Για οποιονδήποτε πελάτη επιθυμεί να γνωρίζει για την υπηρεσία, για παράδειγμα τα δεδομένα που αναμένει να λάβει, αν του αποφέρει κάποια αποτελέσματα ή όχι, τον υποστηριζόμενο τρόπο μεταφοράς κλπ. η περιγραφή αυτή παρέχεται μέσω WSDL (Web Services Description Language). Έτσι πέραν της κοινής μορφοποίησης που θα πρέπει να έχουν τα μηνύματα, θα πρέπει να υπάρχει και μια μορφοποίηση την οποία όλοι οι πάροχοι υπηρεσιών θα πρέπει να τη χρησιμοποιούν έτσι ώστε να προσδιορίζεται και η περιγραφή για τις λεπτομέρειες της υπηρεσίας.

Το επόμενο και τελευταίο επίπεδο της στοίβας πρωτοκόλλου UDDI (service discovery) αφορά στη δημοσιοποίηση και εντοπισμό των υπηρεσιών. Έτσι λοιπόν για τους πελάτες που αιτούνται μια υπηρεσία θα πρέπει να εξασφαλίζεται ένας κοινός τρόπος αναζήτησης και ανάκτησης της. Αυτό μπορεί να επιτευχθεί με την καταχώρηση των υπηρεσιών σε ένα κοινό μητρώο από το οποίο μπορεί κανείς εύκολα να τις εντοπίσει και να τις ανακτήσει. Η δυνατότητα εντοπισμού των Web Services παρέχεται μέσω UDDI.

2.3 Πρότυπα των Web Services

2.3.1 XML

Η XML (eXtensible Markup Language) η οποία χρησιμοποιείται για τον ορισμό και την υλοποίηση των Web Service, είναι μια γλώσσα ανεξάρτητη από σύστημα και υλικό και αφορά στην αναπαράσταση δεδομένων και της μορφή τους σε ένα έγγραφο XML (XML Document). Ένα XML έγγραφο στην απλούστερη μορφή του, ουσιαστικά είναι ένα αρχείο το οποίο περιέχει δεδομένα καθώς και τη σήμανση η οποία καθορίζει την μορφή των δεδομένων σε αντίθεση με την HTML την πρώτη markup γλώσσα που απλά απεικονίζει τα δεδομένα και δεν αναφέρει το τι ακριβώς είναι τα δεδομένα αυτά. Έτσι μέσω της XML είναι δυνατή η ανταλλαγή δεδομένων διαφορετικής μορφής μεταξύ διαφορετικών υπολογιστικών συστημάτων δηλαδή ανεξαρτήτως πλατφόρμας, λογισμικού και λειτουργικού συστήματος. Αποτελεί δηλαδή ένα εργαλείο για την δημιουργία, αποθήκευση και μετάδοση δεδομένων.
Παρακάτω δίνεται ένα παράδειγμα αναπαράστασης δεδομένων σε ένα XML έγγραφο.

```
<?xml version= “1.0”>
<Employee>
```

```
  <Name> Peter Henderson</Name>
  <Address>
     <Street>4142 Nordic Street </Street>
     <City>Stockholm</City>
     <Country>Sweden</Country>
  </Address>
 <Phone>987-654-3210</Phone>
 <Email>kakalos@gmail.com</Email>
</ Employee>
```

2.3.1.1 DTD και XML Schema

Το DTD (Document Type Definition) είναι ένα μοντέλο το οποίο χρησιμοποιείται από την XML για την καθορισμό της δομής ενός XML εγγράφου. Παρέχει ένα αυστηρό πλαίσιο και κανόνες που πρέπει να ακολουθηθούν κατά τη δημιουργία του εγγράφου. Μπορεί να χρησιμοποιηθεί για τον έλεγχο της εγκυρότητας και ακεραιότητας των δεδομένων που περιέχονται στο XML έγγραφο. Αποτελεί ένα σύνολο κανόνων που αφορούν τις ετικέτες που διαθέτει ένα XML αρχείο.

Παρακάτω ακολουθεί το αντίστοιχο DTD αρχείο του XML εγγράφου που προαναφέρθηκε.

```
<!ELEMENT Employee (Name, Address, Phone, Email)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Address (Street, City, Country)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
```

Ένα DTD αρχείο δεν επαρκεί για τον καθορισμό της μορφής ενός XML εγγράφου καθώς για παράδειγμα δεν διευκρινίζεται ο τύπος δεδομένων περιέχονται στο XML έγγραφο. Έτσι αυτού του είδους τις πληροφορίες που δεν συμπεριλαμβάνονται στο DTD αρχείο έρχεται να συμπληρώσει το XML Schema. Παρακάτω ακολουθεί ένα παράδειγμα XML εγγράφου και εν συνεχεία παράδειγμα του XML Schema του.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<Employee
        xmlns="http://simple.example.com/EmpeeXmlDoc"
        xmlns:xsi="http://www.w3.org/2011/XMLSchema-instance"
        xsi:schemaLocation=
```

```
                    "http://simple.example.com/EmpeeXmlDoc
                     file:./EmpeeXmlDoc.xsd">
  <Name>Peter Henderson</Name>
  <Address>
    <Street>4142 Nordic Street </Street>
    <City>Stockholm</City>
    <Country>Sweden</Country>
  </Address>
  <Phone>987-654-3210</Phone>
  <Email>kakalos@gmail.com</Email>
</Employee>
```

Το παράδειγμα του XML schema θα έχει ως εξής:

```
<?xml version"1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://simple.example.com/EmpeeXmlDoc"
  xmlns="http://simple.example.com/EmpeeXmlDoc"
  elementFormDefault="qualified">
  <xsd:element name="Employee">
   <xsd:complexType>
    <xsd:sequence>
        <xsd:element name="Name" type="xsd:string" />
        <xsd:element name="Address">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="Street" type="xsd:string" />
          <xsd:element name="City" type="xsd:string" />
          <xsd:element name="Country" type="xsd:string" />
          </xsd:sequence>
          </xsd:complexType>
          </xsd:element>
          <xsd:element name="Phone" type="xsd:string" />
          <xsd:element name="Email" type="xsd:string" />
          </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
   </xsd:schema>
```

Ένα στοιχείο αναπαρίσταται με την ετικέτα element ενώ η ετικέτα complexType δίνει τη δυνατότητα για τον ορισμό σύνθετων τύπων δεδομένων τα οποία επίσης μπορεί να περιέχουν μια ακολουθία (sequence) στοιχείων αυτού του τύπου. Μέσω των XML Namespaces παρέχεται η δυνατότητα προσδιορισμού ενός στοιχείου ως μοναδικό δίνοντας λύση στο ζήτημα του ότι κάποια από τα ονόματα των στοιχείων ίσως να επαναλαμβάνονται και από άλλα στοιχεία. Κάθε αναφορά σε namespace γίνεται μέσω της τιμής xsd ως πρόθεμα. Η ιδιότητα targetNamespace καθορίζει το namespace στο οποίο ανήκουν οι νέοι καθορισμένοι τύποι. Η ιδιότητα xmlns ορίζει το επιλεγμένο namespace, έτσι αν βρεθεί κάποιο στοιχείο στο οποίο δεν ορίζεται κάποιο πρόθεμα, στην περίπτωση αυτή το στοιχείο ανήκει στο προκαθορισμένο namespace.
Η προδιαγραφή του XML Schema παίζει σημαντικό ρόλο για την υλοποίηση και σχεδιασμό των Web Services. Τα WSDL αρχεία επίσης είναι κατασκευασμένα με χρήση της σύνταξης του XML Schema.
Οι βασικές διαφορές μεταξύ ενός DTD και XML Schema είναι πως, το δεύτερο ουσιαστικά είναι επέκταση του DTD. Το XML Schema υποστηρίζει Namespaces ενώ το DTD όχι. Επίσης το XML Schema σε αντίθεση με το DTD χρησιμοποιεί σύνταξη XML.

2.3.1.2 XML Parsers

Η διαδικασία ανάκτησης δεδομένων από τα XML έγγραφα μέσω τεχνικών σειριακής προσπέλασης αρχείων δεν θα μπορούσε να είναι μια πρακτική και αποτελεσματική λύση, ιδιαίτερα σε περιπτώσεις που τα δεδομένα πρέπει να προστίθενται είτε να αφαιρούνται δυναμικά. Το κενό αυτό έρχεται να καλύψει η διαδικασία parsing (λεκτικής ανάλυσης) κατά την οποία το κείμενο αναλύεται στα επιμέρους στοιχεία του.

Στη διαδικασία parsing λαμβάνει μέρος ένας λεκτικός αναλυτής μέσω του οποίου επιτυγχάνεται η ανάγνωση και η ενημέρωση ενός XML εγγράφου, έτσι ώστε η πληροφορία που περιέχεται σε αυτό να είναι κατανοητή και διαχειρίσιμη. Ουσιαστικά ένας XML parser διαβάζει ένα XML αρχείο και το μετατρέπει σε ένα έγγραφο το οποίο μπορεί να διαχειριστεί η εφαρμογή. Οι XML parsers χωρίζονται σε δύο βασικές κατηγορίες, τους event based parsers και tree based parsers.

Το πιο καθιερωμένο API για τους tree based parsers είναι το DOM (Document Object Model), το οποίο όταν χρησιμοποιείται ο parser επεξεργάζεται όλες τις εγγραφές και δημιουργεί μια συνδεδεμένη λίστα από δομές δεδομένων για να τις αναπαραστήσει. Αυτό δίνει τη δυνατότητα στο χρήστη να θεωρήσει τη δενδρική μορφή του XML εγγράφου με αντικειμενοστραφή τρόπο και να επιδράσει σε αυτό. Η εφαρμογή για να δημιουργήσει μια εγγραφή δεδομένων χρησιμοποιεί το DOM

API ώστε να δημιουργήσει ένα DOM δενδρικής μορφής το οποίο εν συνεχεία ο parser κωδικοποιεί.

Για τους event based parsers το πιο καθιερωμένο API είναι το SAX (Simple API for XML), το οποίο ορίζει ένα σύνολο από λειτουργίες οι οποίες πρέπει να υλοποιηθούν μέσα στην εφαρμογή. Όταν ένα συγκεκριμένο event εμφανιστεί οι λειτουργίες αυτές καλούνται από τον parser. Ως events θεωρούνται τα δεδομένα που περιέχονται στα πεδία καθώς και η αρχή και το τέλος των εγγραφών.

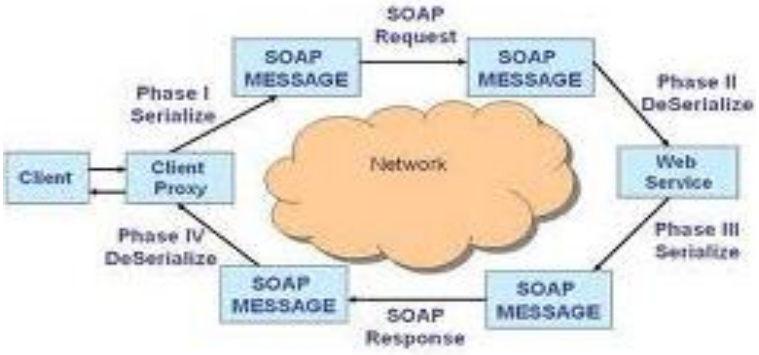
2.3.2 Πρωτόκολλο SOAP

Η εισαγωγή του διαδικτυακού πρωτοκόλλου SOAP είναι ένα πολύ σημαντικό βήμα για την ανάπτυξη των εφαρμογών καθώς δίνεται πλέον η δυνατότητα για διασύνδεση των εφαρμογών με παγκόσμια κατανεμημένους εξυπηρετητές μέσω της χρήσης προτύπων του internet HTTP και XML.

2.3.2.1 Σκοπός του πρωτοκόλλου SOAP

Σκοπός του SOAP είναι η επικοινωνία μεταξύ των εφαρμογών μέσω απομακρυσμένων κλήσεων διαδικασιών (RPC) με χρήση πρωτοκόλλων διαδικτύου όπως το HTTP. Το SOAP είναι ουσιαστικά ένας RPC μηχανισμός όπου λόγω του ότι είναι βασισμένος σε XML επιτρέπει την επικοινωνία εφαρμογών που τρέχουν σε διαφορετικά λειτουργικά συστήματα με διαφορετικές τεχνολογίες και γλώσσες προγραμματισμού. Όπως με την XML δίνεται η λύση για μια κοινή γλώσσα επικοινωνίας, το SOAP είναι αυτό το οποίο παρέχει μια κοινή μορφοποίηση των μηνυμάτων έτσι ώστε τα αντικείμενα να έχουν τη δυνατότητα παρόλο που μπορεί να είναι άγνωστα μεταξύ τους, να ανταλλάσουν μηνύματα. Η σύνταξη XML είναι αυτή που χρησιμοποιείται από το SOAP ως αναφορά την κωδικοποίηση των μηνυμάτων ενώ σε ότι έχει να κάνει με την μετάδοσή τους χρησιμοποιούνται πρωτόκολλα όπως το HTTP ή το SMTP.

Το SOAP λοιπόν αποτελεί ένα μηχανισμό μέσω του οποίου τα web services μπορούν να διασυνδεθούν. Παρέχει ένα πρωτόκολλο επικοινωνίας με το οποίο μπορούν να διασυνδεθούν διαφορετικοί διαδικτυακοί τόποι οι οποίο πιθανό να χρησιμοποιούν διαφορετικές τεχνολογίες για την πρόσβαση στο διαδίκτυο. Το μοντέλο επικοινωνίας του SOAP διασφαλίζει πως ένα μήνυμα XML παρέχει συμβάσεις για το πώς θα παραδοθεί από τον αποστολέα στον παραλήπτη.



Σχήμα 20. Αναπαράσταση εφαρμογής του πρωτοκόλλου SOAP

2.3.2.2 Συντακτική δομή του πρωτοκόλλου SOAP

Ένα μήνυμα SOAP μήνυμα συντακτικά αποτελείται από το envelope element, το header element που είναι προαιρετικό, το body element και το fault element που επίσης είναι προαιρετικό. Μέσω του envelope element καθορίζεται πως το XML έγγραφο είναι ένα SOAP μήνυμα καθώς επίσης ορίζεται η αρχή και το τέλος του μηνύματος. Στο header element το οποίο όπως προαναφέρθηκε είναι προαιρετικό περιλαμβάνονται χαρακτηριστικά του μηνύματος SOAP τα οποία θα χρησιμοποιηθούν από τον εκάστοτε παραλήπτη κατά την επεξεργασία του. Τα δεδομένα του μηνύματος περιέχονται στο body element ενώ στο fault element που είναι επίσης προαιρετικό περιλαμβάνονται πληροφορίες για σφάλματα που μπορεί να προκληθούν κατά την επεξεργασία του μηνύματος.

Όλα τα παραπάνω στοιχεία δηλώνονται στο προεπιλεγμένο namespace για το soap envelope:
http://www.w3.org/2003/05/soap-envelope

Το προεπιλεγμένο namespace για την κωδικοποίηση του SOAP και τους τύπους δεδομένων είναι:
http://www.w3.org/2001/12/soap-encoding

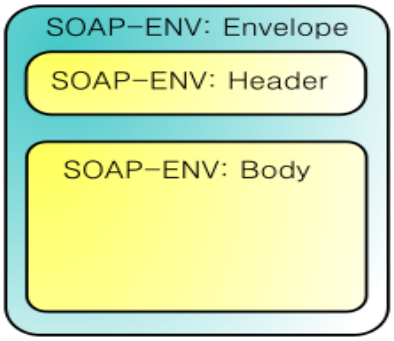
Οι σημαντικότεροι κανόνες οι οποίοι θα πρέπει να διέπουν τα SOAP μηνύματα ως αναφορά το συντακτικό τους είναι οι εξής:

- Ένα μήνυμα SOAP πρέπει να είναι κωδικοποιημένο με χρήση XML
- Ένα μήνυμα SOAP πρέπει να χρησιμοποιεί το SOAP Envelope namespace καθώς και το SOAP Encoding namespace

- Το SOAP μήνυμα δεν πρέπει να περιέχει αναφορά σε DTD καθώς επίσης και πληροφορία για την επεξεργασία XML .

Παρακάτω ακολουθεί η δομή ενός SOAP μηνύματος:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
...
...
</soap:Header>
<soap:Body>
...
...
</soap:Body>
</soap:Envelope>
```



Σχήμα 21. Βασική δομή μηνύματος SOAP

Η ύπαρξη ενός envelope element είναι απαραίτητη. Το SOAP envelope περικλείει όλα τα υπόλοιπα στοιχεία στο μήνυμα SOAP, καθώς επίσης η ύπαρξη του ως root element όπως αναφέρθηκε και παραπάνω καθορίζει πως το XML document είναι ένα SOAP μήνυμα. Τα xmlns:soap namespace θα πρέπει να είναι πάντα το http://www.w3.org/2003/05/soap-envelope. Σε περίπτωση που το namespace είναι διαφορετικό το μήνυμα θα απορριφθεί. Η ιδιότητα encodingStyle ορίζει τους κανόνες κωδικοποίησης που χρησιμοποιεί το μήνυμα. Μπορεί να βρίσκεται σε οποιοδήποτε στοιχείο του μηνύματος και να επηρεάζει το στοιχείο αλλά και τα child elements στοιχείου αυτού.

Το header element (Soap Header) στην περίπτωση που περιλαμβάνεται στο μήνυμα θα πρέπει να βρίσκεται αμέσως μετά το envelope element. Το όνομα του στοιχείου αυτού θα πρέπει να είναι “Header” ενώ το όνομα του namespace του θα πρέπει να

είναι το το http://www.w3.org/2003/05/soap-envelope. Το ίδιο μπορεί να περιέχει ιδιότητες και child elements ορισμένα με namespaces. Κάθε child element του Soap Header ονομάζεται SOAP header block.

Η ύπαρξη του body element (Soap Body) είναι απαραίτητη σε όλα τα μηνύματα SOAP και θα πρέπει να ακολουθεί αμέσως μετά το header element στην περίπτωση που αυτό υπάρχει, διαφορετικά αμέσως μετά τον ορισμό του envelope element. Το SOAP Body περιέχει ένα μηχανισμό για τη μετάδοση πληροφοριών στον αποδέκτη ενός μηνύματος SOAP. Περιλαμβάνει όλες τις προδιαγραφές τις αίτησης για το μήνυμα όπως είναι οι κλήσεις μεθόδων. Το όνομα του στοιχείου πρέπει να είναι "Body" ενώ το namespace του http://www.w3.org/2003/05/soap-envelope. Μπορεί και αυτό να περιέχει ιδιότητες και child elements ορισμένα με namespaces. Το fault element (SOAP Fault) χρησιμοποιείται ώστε να μεταφέρει τις πληροφορίες των πιθανών σφαλμάτων σε ένα SOAP μήνυμα. Το fault element το οποίο είναι προαιρετικό, στην περίπτωση που υπάρχει περιλαμβάνεται στο body element και δεν εμφανίζεται παραπάνω από μια φορά σε ένα SOAP μήνυμα. Το όνομα του στοιχείου θα πρέπει να είναι "Fault" και το namespace του http://www.w3.org/2003/05/soap-envelope. Θα πρέπει να περιέχει δύο η περισσότερα child elements με την ακόλουθη σειρά:

- Ένα υποχρεωτικό Code element
- Ένα υποχρεωτικό Reason element
- Ένα προαιρετικό Node element
- Ένα προαιρετικό Role element
- Ένα προαιρετικό Detail element

2.3.2.3 Ανταλλαγή μηνυμάτων στο πρωτόκολλο SOAP

Η απλότητα της χρήσης του HTTP είναι και ο λόγος που το καθιστά το πλέον καθιερωμένο πρωτόκολλο για την ανταλλαγή μηνυμάτων SOAP. Η επικοινωνία HTTP επιτυγχάνεται μέσω του TCP/IP. Ο client αρχικά συνδέεται μέσω χρήσης τουTCP/IP με τον server και εκτελεί μια αίτηση HTTP:
POST/request.HTTP/1.1
Host: www.example.org
Content-Type: application/soap
Content-Length: 200

Η αίτηση επεξεργάζεται από το server και εν συνεχεία ο client λαμβάνει μια απάντηση HTTP πάνω από την ίδια TCP σύνδεση:
200 OK
Content-Type: application/soap
Content-Length: 200

Αν η σύνδεση (binding) είναι επιτυχής ο κωδικός κατάστασης είναι 200 ενώ σε περίπτωση που ο server δεν ήταν σε θέση να αποκωδικοποιήσει την αίτηση θα επέστρεφε μια απάντηση παρόμοια της ακόλουθης:
400 Bad Request
Content-Length: 0

Ένα αίτημα SOAP μπορεί να μεταδίδεται είτε με τις HTTP μεθόδους POST ή GET. Η πρώτη εκ των δύο ορίζει τουλάχιστον δύο HTTP headers: Content-Type και Content-Length. Το HTTP header Content-Type καθορίζει τον MIME τύπο καθώς και την κωδικοποίηση για τους χαρακτήρες που θα έχει το body ενός μηνύματος SOAP ενώ το HTTP header Content-Length καθορίζει το μέγεθος σε Bytes που περιλαμβάνει το body ενός μηνύματος SOAP. Εν συνεχεία ακολουθούν παραδείγματα ενός SOAP Request και ενός SOAP Response με HTTP Binding.

Αποστολή SOAP αιτήματος GetStockPrice το οποίο περιλαμβάνει την παράμετρο StockName:

```
Post /InStock HTTP/1.1
Host: www.webserv.org
Content-Type: text/soap+xml; charset=utf-8
Content-Length: length

<?xml version=”1.0”?>
<soap:Envelope
xmlns:soap=”http://www.w3.org/2003/05/soap-envelope”
soap:encodingStyle=”http://www.w3.org/2001/12/soap-encoding”>
 <soap:Body xmlns:m=”http://www.webserv.org/stock”>
   <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
   </m:GetStockPrice>
 </soap:Body>
</soap:Envelope>
```

2.3.3 Περιγραφή των υπηρεσιών διαδικτύου - WSDL

Όπως αναφέρθηκε παραπάνω μέσω του πρωτοκόλλου SOAP γίνονται οι απομακρυσμένες κλήσεις διαδικασιών από κάποιον πελάτη προς κάποιον εξυπηρετητή. Ωστόσο από πλευράς πελάτη προκύπτει το ζήτημα της σύνταξης του μηνύματος SOAP που θα αποσταλεί. Από το πρωτόκολλο SOAP όπως έχει αναφερθεί καθορίζονται κάποιοι κανόνες και η φόρμα που θα πρέπει να ακολουθούν τα μηνύματα, ο πελάτης όμως θα πρέπει να έχει στη διάθεσή του

λεπτομέρειες για το μήνυμα που θα αποσταλεί, όπως το που ακριβώς θα αποσταλεί, ποιες μεθόδους θέλει να καλέσει από την υπηρεσία είτε ποια είναι τα πρωτόκολλα επικοινωνίας τα οποία υποστηρίζονται από τον εξυπηρετητή. Έτσι εφόσον το SOAP αποτελεί ένα πρότυπο επικοινωνίας δεδομένων, χρειάζεται μια γλώσσα περιγραφής των δεδομένων αυτών. Για παράδειγμα ως αναφορά την κλήση μιας κατάλληλης μεθόδου από την υπηρεσία, είναι απαραίτητο ο πελάτης να έχει γνώση του ονόματός της, των παραμέτρων που δέχεται καθώς και του τύπου της.

2.3.3.1 Χρησιμότητα του WSDL

Προκειμένου λοιπόν να είναι γνωστές όλες οι απαραίτητες πληροφορίες για την κλήση μιας υπηρεσίας διαδικτύου, θα πρέπει ο πελάτης να έχει στη διάθεσή του μια περιγραφή τους. Για την ανάκτηση της περιγραφής με τρόπο εύκολο και τυποποιημένο, χρειάζεται να υπάρχει ένας μηχανισμός ο οποίος θα παρέχει ακριβώς αυτά που είναι απαραίτητα για την κλήση της υπηρεσίας. Αυτή ακριβώς την ανάγκη έρχεται να καλύψει η WSDL (Web Services Description Language). Πιο συγκεκριμένα η WSDL παρέχει πληροφορία για το interface της υπηρεσίας δίνοντας ουσιαστικά μια περιγραφή για όλες τις δημόσιες συναρτήσεις της κάθε διαθέσιμης υπηρεσίας. Περιγράφει τους τύπους δεδομένων που συμπεριλαμβάνονται σε οποιοδήποτε μήνυμα αφορά αίτηση ή απόκριση. Δίνει πληροφορίες σχετικά με το πρωτόκολλο μεταφοράς που πρόκειται να χρησιμοποιηθεί για το binding και τέλος παρέχει την πληροφορία σχετικά με τη διεύθυνση της υπηρεσία δίνοντας έτσι την δυνατότητα στον πελάτη να γνωρίζει το που θα πρέπει να αποστείλει το μήνυμα. Σε ένα αρχείο WSDL η περιγραφή της μορφής των μηνυμάτων βασίζεται στο πρότυπο XML Schema, πράγμα που σημαίνει πως η WSDL είναι ανεξάρτητη από γλώσσα προγραμματισμού και βασισμένη σε πρότυπα. Έτσι αναφορικά με τη δυνατότητα περιγραφής Web Services διεπαφών αυτές μπορεί να είναι προσβάσιμες από πολλές διαφορετικές πλατφόρμες και γλώσσες προγραμματισμού.
Σύμφωνα λοιπόν με τα παραπάνω, η WSDL χρησιμοποιείται για να παρέχει την πληροφορία για το τι μπορεί να κάνει ένα web service, που βρίσκεται και πως μπορεί να το καλέσει κανείς.

2.3.3.2 Συντακτική δομή του WSDL

Παρακάτω ακολουθεί ένα παράδειγμα της δομής και του συντακτικού ενός WSDL εγγράφου.

```
<wsdl: definitions name-“nmtoken”?  targetNamespace=“uri”>
  <import  namespace=“uri” location=“uri”/> *
  <wsdl:documentation …. /> ?
  <wsdl:types> ?
```

```
      <wsdl:documentation … /> ?
      <xsd:schema …. /> *
 </wsdl:types>
 <wsdl:message name=“ncname”> *
      <wsdl:documentation … /> ?
      <part name=“ncname” element=“qname”?  type=“qname”?/> *
 </wsdl:message>
 <wsdl:portType name=“ncname”> *
    <wsdl:documentation … /> ?
    <wsdl:operation name=“ncname”> *
        <wsdl:documentation …. /> ?
        <wsdl:input message=“qname”> ?
              <wsdl:documentation  …. /> ?
        </wsdl:input>
        <wsdl:output message=“qname”> ?
              <wsdl:documentation …. /> ?
        </wsdl:output>
        <wsdl:fault name=“ncname” message=“qname”> *
              <wsdl:documentation …. /> ?
        </wsdl:fault>
    </wsdl:operation>
 </wsdl:portType>
 <wsdl:serviceType name=”ncname”> *
    <wsdl:portType name=”qname”/> +
</wsdl:serviceType>
<wsdl:binding name=”ncname” type=”qname”> *
     <wsdl:documentation …. /> ?
     <- - binding details - -> *
     <wsdl:operation name=”ncname”> *
          <wsdl:documentation … /> ?
          <- - binding details - -> *
          <wsdl:input> ?
                <wsdl:documentation …. /> ?
                <- - binding details - - >
          </wsdl:input>
          <wsdl:output> ?
                <wsdl:documentation …. /> ?
               <- - binding details - -> *
          </wsdl:output>
          <wsdl:fault name=”ncname”> *
               <wsdl:documentation … /> ?
```

```
                <- - binding details - -> *
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name=”ncname” serviceType=”qname”> *
        <wsdl:documentation …. /> ?
        <wsdl:port name=”ncname” binding=”qname”> *
            <wsdl:documentation …. /> ?
            <- - address details - ->
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Όπως φαίνεται και στο πιο πάνω έγγραφο ένα αρχείο WSDL χαρακτηρίζεται από τα παρακάτω βασικά στοιχεία:

definitions
Είναι το root element στο WSDL αρχείο στο οποίο δηλώνεται το όνομα του web service καθώς και τα namespaces τα όπου χρησιμοποιούνται σε όλο το αρχείο. Επίσης όπως καθορίζεται και από το συντακτικό της XML το στοιχείο Definitions περιλαμβάνει και όλα τα υπόλοιπα στοιχεία που ακολουθούν.

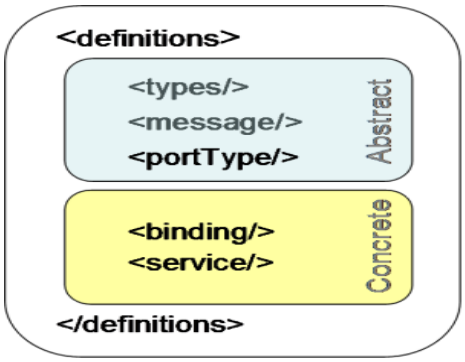
types
Σε αυτό το element σε ένα WSDL αρχείο περιλαμβάνεται η περιγραφή των τύπων όλων των δεδομένων τα οποία μεταφέρονται μεταξύ client και server.

message
Είναι το element στο οποίο περιγράφονται τα μηνύματα αιτήσεων και αποκρίσεων. Δηλώνεται το όνομα του μηνύματος, οι παράμετροι τις οποίες δέχεται το μήνυμα καθώς και οι τιμές που επιστρέφει.

portType
Το portType element είναι το στοιχείο μέσω του οποίου μπορούν να επιτευχθούν συνδυαστικές λειτουργίες. Το portType μπορεί να συνδυάσει άλλα υπάρχοντα στοιχεία message, όπως για παράδειγμα μπορεί να συνδυάσει ένα μήνυμα αίτησης και απόκρισης σε μια λειτουργία.

binding
Μέσω του συγκεκριμένου element περιγράφεται ο τρόπος με τον οποίο υλοποιείται μια υπηρεσία (service). Στο binding element παρατίθεται οποιαδήποτε πληροφορία αφορά στο SOAP.
service

Στο service element δηλώνεται η διεύθυνση της υπηρεσίας (service) όπου ο αποστολέας του αιτήματος μπορεί να εντοπίσει την υπηρεσία για να την καλέσει.



Σχήμα 22. Δομή ενός WSDL αρχείου

Βάση των πιο πάνω elements ως αναφορά τη δομή του WSDL αρχείου ορίζονται επίσης τα παρακάτω στοιχεία:

documentation
Στο element αυτό συμπεριλαμβάνονται πληροφορίες και οδηγίες γενικής φύσεως. Το συγκεκριμένο element μπορεί να εμπεριέχεται σε οποιοδήποτε άλλο.

Import
Μέσω του import element μπορούν να εισαχθούν άλλα έγγραφα WSDL και XML Schemas. Πιο συγκεκριμένα δύνεται η δυνατότητα σε δύο έγγραφα WSDL να εισάγουν τα βασικά τους στοιχεία που είναι τα ίδια καθώς και να έχουν τα δικά τους service elements. Έτσι ουσιαστικά η ίδια υπηρεσία μπορεί να βρίσκεται διαθέσιμη ώστε να κληθεί από δύο φυσικές διευθύνσεις.

Όπως φαίνεται και στο παραπάνω σχεδιάγραμμα της δομής του WSDL, η περιγραφή σε ένα WSDL αρχείο μπορεί να διαχωριστεί σε δύο βασικές κατηγορίες ως αναφορά τους ορισμούς που λαμβάνουν χώρο στην καθεμία. Έτσι στην πρώτη κατηγορία Abstract definitions λαμβάνουν χώρο περισσότερο γενικευμένοι ορισμοί για την υπηρεσία ενώ στη δεύτερη Concrete definitions οι ορισμοί που γίνονται είναι πιο συγκεκριμένοι. Πιο συγκεκριμένα, στο μέρος του WSDL αρχείου το οποίο απαρτίζεται από τα στοιχεία types, message και portTypes, η περιγραφή ενός web service interface δεν είναι λεπτομερής σε ότι αφορά την υλοποίηση του service. Το μέρος αυτό ονομάζεται Abstract definitions. Αντίθετα, το μέρος των στοιχείων binding και service του WSDL αρχείου όπου η περιγραφή αφορά στην υλοποίηση του service ονομάζεται Concrete definitions.Το σύνολο των δύο κατηγοριών

Abstract και Concrete Definitions όπου περιλαμβάνουν την περιγραφή του interface (abstract) και της υλοποίησης του service (concrete) ονομάζεται Service Definition.

2.3.4 Universal Description, Discovery and Integration (UDDI)

2.3.4.1 Ο ρόλος του UDDI και δυνατότητες που παρέχει

Το Universal Description, Discovery and Integration (UDDI), εστιάζει στον καθορισμό ενός συνόλου υπηρεσιών οι οποίες υποστηρίζουν την ανακάλυψη και την περιγραφή των διαθέσιμων web services που είναι διαθέσιμα, των παρόχων τους αλλά και των τεχνικών διεπαφών που θα χρησιμοποιηθούν για την προσπέλαση τους από τον εκάστοτε χρήστη. Το UDDI βασίζεται σε ένα κοινό σύνολο από βιομηχανικά πρότυπα όπως HTTP, XML, XML Schema και SOAP. Παρέχει μια θεμελιώδη υποδομή για ένα περιβάλλον λογισμικού το οποίο είναι προσανατολισμένο σε υπηρεσίες που μπορεί να είναι διαθέσιμες δημόσια είτε στον ιδιωτικό χώρο ενός οργανισμού.

Είναι ουσιαστικά ο κατάλογος των υπηρεσιών Web Services όπου λειτουργεί με τον ίδιο τρόπο όπου θα μπορούσαμε σε ένα χρυσό οδηγό να αναζητήσουμε μια εταιρεία να πληροφορηθούμε για τις υπηρεσίες που παρέχει και να ενημερωθούμε για κάποια συγκεκριμένη υπηρεσία. Αποτελεί ένα μηχανισμό που δίνει την δυνατότητα στους χρήστες να εντοπίζουν άλλα web services. Με τη χρήση μιας διεπαφής UDDI μια επιχείρηση μπορεί να συνδεθεί δυναμικά με υπηρεσίες που μπορεί να παρέχονται από άλλους εξωτερικούς συνεργάτες. Η ύπαρξη του UDDI φυσικά δε σηματοδοτεί πως μια υπηρεσία δεν μπορεί να διατίθεται χωρίς την απαραίτητη καταχώρησή της σε αυτό. Ωστόσο το UDDI είναι χρήσιμο στην περίπτωση που οποιαδήποτε υπηρεσία θα χρειαστεί να είναι διαθέσιμη για ένα ευρύτερο κοινό, έτσι ώστε να μπορεί να ανακαλυφθεί από τον εκάστοτε δυνητικό χρήστη.

Οι πάροχοι των web services μπορούν να κάνουν την καταχώρησή τους σε οποιοδήποτε διαχειριστή UDDI και τα δεδομένα τους να αντιγραφούν σε όλα τα αντίγραφα του καταλόγου άλλων διαχειριστών. Έτσι λοιπόν η αναζήτηση ενός παρόχου από τον χρήστη μπορεί να γίνει σε κατάλογο οποιουδήποτε διαχειριστή ανεξάρτητα από το σε ποιον διαχειριστή είχε κάνει την εγγραφή ο πάροχος. Σημαντικό είναι να αναφερθεί πως η οποιαδήποτε τροποποίηση των καταχωρημένων στοιχείων μπορεί να γίνει από τον ίδιο διαχειριστή στον οποίο έγινε η εγγραφή και μόνο. Αυτό συμβαίνει λόγω του ότι ο κάθε διαχειριστής διατηρεί διαφορετική πολιτική πιστοποίησης και ασφάλειας της πληροφορίας η

οποία δεν μεταφέρεται στα αντίγραφα καταλόγων των υπολοίπων διαχειριστών. Οι προδιαγραφές API του UDDI παρέχουν ένα σύνολο διεπαφών για την καταχώρηση των υπηρεσιών καθώς και αντίστοιχες διεπαφές αναζήτησης που εξυπηρετούν στην εύρεση των υπηρεσιών.

Επιπροσθέτως μέσω των διεπαφών, από τον διαχειριστή του UDDI παρέχεται η δυνατότητα αλληλεπίδρασης με το χρήστη για καταχώρηση διαχείριση και ανεύρεση παρόχων και υπηρεσιών στον κατάλογο με χρήση του διαδικτύου. Ένας κατάλογος UDDI μπορεί να περιλαμβάνει πελάτες δύο κατηγοριών. Η πρώτη κατηγορία αφορά παρόχους που επιθυμούν να κοινοποιήσουν μια υπηρεσία καθώς και τις διεπαφές αυτής, και η δεύτερη κατηγορία αφορά πελάτες που επιθυμούν να χρησιμοποιήσουν συγκεκριμένες υπηρεσίες και συνδέονται προγραμματιστικά με αυτές. Για να γίνει κανείς διαχειριστής ενός καταλόγου UDDI σε κάθε περίπτωση θα πρέπει να ακολουθεί αυστηρά την πολιτική διασφάλισης των πρωτότυπων δεδομένων και των αντιγράφων τους.

Υπάρχουν δύο ειδών κατάλογοι. Ο δημόσιος και ο ιδιωτικός. Ο δημόσιος ο οποίος ονομάζεται και UDDI Business Registry (UBR) είναι κατάλογος ο οποίος φιλοξενείται από ένα μικρό σύνολο εταιρειών. Συνηθέστερα χρησιμοποιούνται οι ιδιωτικοί κατάλογοι που μπορούν να φιλοξενηθούν στο διαδίκτυο είτε στο τοπικό δίκτυο ενός οργανισμού εξυπηρετώντας για παράδειγμα την καταχώρηση και αναζήτηση υπηρεσιών που περιορίζονται για ενδοεταιρική χρήση. Σημαντικό επίσης να αναφερθεί πως ένας δημόσιος κατάλογος (UBR) δεν εξασφαλίζει στον καλύτερο βαθμό το επίπεδο ασφαλείας που προαπαιτείται από κάθε επίδοξο πελάτη που προτίθεται να κάνει χρήση των υπηρεσιών του καταλόγου που προέρχονται από οποιοδήποτε πάροχο.

Ο εκάστοτε πάροχος αρχικά εκδίδει ένα WSDL αρχείο για την περιγραφή των υπηρεσιών που υποστηρίζει μέσω ενός επεξεργαστή SOAP ενώ για την καταχώρηση τους τον κατάλογο UDDI Registry κάνει χρήση των UDDI APIs. Η περιγραφή αυτή θα πρέπει να περιέχει όλα τα απαραίτητα στοιχεία που θα πρέπει να έχει στη διάθεσή του κάποιος δυνητικός χρήστης για να χρησιμοποιήσει την υπηρεσία. Ο κατάλογος UDDI Registry θα παρέχει ένα URL το οποίο θα δείχνει στο WSDL αρχείο ή οποιοδήποτε άλλο αρχείο υφίσταται για την περιγραφή των web services. Ο χρήστης που επιθυμεί να καλέσει την υπηρεσία κάνει αναζήτηση στον κατάλογο μέσω του SOAP επεξεργαστή του με σκοπό την ανάκτηση του WSDL αρχείου.

2.3.4.2 Διαχωρισμός της πληροφορίας

Βάση της πληροφορίας που περιέχουν τα έγγραφα που χρησιμοποιούνται για την καταχώρηση των web services στον κατάλογο UDDI, η πληροφορία που αφορά την

ύπαρξη ενός παρόχου διαχωρίζεται σε τρία μέρη τα οποία είναι γνωστά ως White pages, Yellow Pages και Green Pages.

Τα White Pages περιέχουν γενικές πληροφορίας επικοινωνίας με τον πάροχο μιας υπηρεσίας όπως την επωνυμία και το τηλέφωνο.

Τα Yellow Pages παρέχουν λεπτομερέστερη περιγραφή για τον πάροχο με χρήση πληροφοριών ταξινόμησης που μπορεί να αφορούν για παράδειγμα τους τύπους των υπηρεσιών και την τοποθεσία τους. Μπορεί να γίνεται δηλαδή χρήση κωδικών προϊόντων είτε κωδικών ταυτοποίησης του παρόχου τα οποία κάλλιστα θα μπορούσαν να τεθούν και ως κριτήρια αναζήτησης των υπηρεσιών που παρέχονται από τον πάροχο αυτό.

Τα Green Pages περιέχουν τεχνικές λεπτομέρειες μιας υπηρεσίας. Στο τμήμα αυτό δίνεται η πληροφορία σχετικά με τον τρόπο μέσω του οποίου είναι προσβάσιμες οι υπηρεσίες.

Σχήμα 23. Ταξινόμηση της παρεχόμενης πληροφορίας σε τρία διαφορετικά επίπεδα

2.3.4.3 Αρχιτεκτονική δομή του UDDI

Το UDDI ως αναφορά την δομή της αρχιτεκτονικής του χωρίζεται σε τρία βασικά μέρη. Αυτά είναι τα UDDI API, UDDI cloud services και UDDI data model.

Το UDDI API είναι ένα API βασισμένο στο SOAP όπου διαχωρίζεται σε ότι αφορά στην προσπέλαση των εφαρμογών για την αναζήτηση της πληροφορίας καθώς και την δυνατότητα των εφαρμογών για καταχώρηση της πληροφορίας στον κατάλογο. Βάση του διαχωρισμού αυτού προκύπτουν δύο ειδών SOAP-based APIs, τα Inquiry APIs και Publisher APIs. Τα Inquiry APIs αφορούν την αναζήτηση πληροφορίας από τις εφαρμογές ενώ η δυνατότητα στη εκάστοτε εφαρμογή για την καταχώρηση της πληροφορίας στον κατάλογο παρέχεται από τα Publisher APIs.

Το UDDI cloud services αφορά στο σύνολο όλων των διαδικτυακών τόπων που παρέχουν υλοποίηση του πρωτοκόλλου UDDI και συγχρονίζουν όλα τα δεδομένα που υπάρχουν σε κάθε UDDI κόμβο. Μέσω των cloud services δίνεται η δυνατότητα ύπαρξης ενός κατάλογου ο οποίος είναι λογικά κεντρικοποιημένος αλλά φυσικά κατανεμημένος, δηλαδή οποιαδήποτε δεδομένα καταχωρηθούν σε ένα μόνο κατάλογο μπορούν αυτομάτως να αντιγραφούν σε όλους τους υπάρχοντες.

Το UDDI data model αφορά σε ένα XML Schema το οποίο προορίζεται για την περιγραφή των υπηρεσιών και των παρόχων τους και αποτελείται από τέσσερις βασικούς τύπους δόμησης δεδομένων:

α) businessEntity το οποίο μέσω του businessKey που περιέχει ξεχωρίζει και καθίσταται ως μοναδικό, περιλαμβάνει πληροφορία που αφορά την επιχείρηση τον πάροχο δηλαδή της υπηρεσία Το businessEntity απαρτίζεται από πολλά διάφορα elements εκ των οποίων υποχρεωτικό element είναι το στοιχείο name. Το συγκεκριμένο element περιέχει το όνομα του παρόχου ενώ σε ένα businessEntity είναι δυνατό να υπάρχουν περισσότερα από ένα διαφορετικά name elements όπου αφορούν στον ίδιο πάροχο υπηρεσίας. Υπόλοιπα στοιχεία που μπορεί να περιλαμβάνονται σε ένα businesEntity και τα οποία δεν είναι υποχρεωτικά είναι:
Το discoveryURLs element όπου περιλαμβάνει διευθύνσεις URLs του παρόχου. Το description element που περιλαμβάνει μια περιγραφή του παρόχου της υπηρεσίας. Το contancts element όπου αναφέρεται σε πρόσωπα της επιχείρησης που παρέχει την υπηρεσία περιλαμβάνοντας επαφές τους όπως διευθύνσεις τηλέφωνα και άλλες σχετικές πληροφορίες. Τα elements identifierBag και categoryBag τα οποία καθιστούν προσβάσιμη τη δομή δεδομένων businesEntity μέσω αναζήτησης που βασίζεται σε διεθνή τυποποιημένα αναγνωριστικά και το Signature element το οποίο δίνει τη δυνατότητα στον πάροχο να εξασφαλίζει την αυθεντικότητα των βασικών τύπων δόμησης δεδομένων.

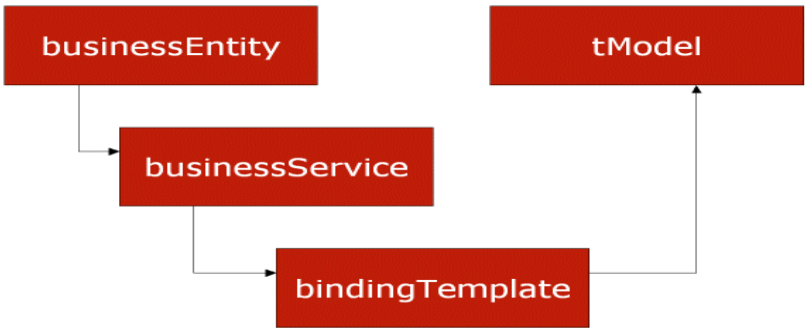
β)businessService: το οποίο μέσω του ServiceKey που περιέχει ξεχωρίζει και καθίσταται ως μοναδικό, περιλαμβάνει πληροφορία που αφορά την παρεχόμενη υπηρεσία. Το bussinesService μπορεί να αποτελείται από διάφορα μη υποχρεωτικά elements. Το στοιχείο name αφορά στο όνομα της υπηρεσίας και σε ένα businessService είναι δυνατό να περιλαμβάνονται και άλλα name elements που να αφορούν στην ίδια επιχείρηση πάροχο της υπηρεσίας. Τα στοιχεία που περιλαμβάνονται είναι κατά κύριο λόγο αντίστοιχα των στοιχείων του τύπου businesEntity ωστόσο εδώ αναφέρονται στην υπηρεσία. Έτσι μπορεί να υπάρχει το description element όπου περιέχει μια σύντομη περιγραφή για την υπηρεσία. Το categoryBag element με βάση το οποίο μπορούν να βρεθεί μια

δομή δεδομένων businessService από μια αναζήτηση μέσω τυποποιημένων αναγνωριστικών μιας κατηγορίας επιχειρήσεων στην οποία ανήκει η υπηρεσία. Τέλος το στοιχείο Signature όπου δίνει τη δυνατότητα στον πάροχο να εξασφαλίζει την αυθεντικότητα των βασικών τύπων δόμησης δεδομένων.

γ) bindingTemplate το οποίο ξεχωρίζει και καθίσταται μοναδικό μέσω του bindingKey που περιέχει, περιλαμβάνει πληροφορίες σχετικά με τον τρόπο και το σημείο όπου υπάρχει η δυνατότητα πρόσβασης σε μια υπηρεσία. Ο τύπος δόμησης δεδομένων bindingTemplate μπορεί να περιλαμβάνει κάποια elements εκ των οποίων υποχρεωτικό είναι το στοιχείο accessPoint. Με το συγκεκριμένο element δίνεται η δυνατότητα εύρεσης της διεύθυνσης της υπηρεσίας. Υπόλοιπα στοιχεία που υπάρχουν είναι: Το description element όπου περιέχει μια σύντομη περιγραφή του binding Template η οποία παρέχεται σε μορφή κειμένου. Το tModelInstanceDetails element όπου περιλαμβάνει έναν σύνολο από tModelKey τα οποία χρησιμεύουν ως μοναδικά αναγνωριστικά για διάφορες δομές tModel και όλα στο σύνολό τους ονομάζονται τεχνικό αποτύπωμα της υπηρεσίας. Το categoryBag element όπου περιλαμβάνει πληροφορία σχετικά με την κατηγορία στην οποία ανήκει μια δομή δεδομένων bindingTemplate. Τέλος το Signature element μέσω του οποίου δίνεται η δυνατότητα της ψηφιακής υπογραφής για τον τύπο δεδομένων με σκοπό την πιστοποίηση της αυθεντικότητας τους.

δ)Το tModel το οποίο καθορίζεται ως μοναδικό μέσω του αναγνωριστικού tModelKey, περιλαμβάνει την πληροφορία σχετικά με το που μπορεί να είναι προσβάσιμες εξωτερικές τεχνικές προδιαγραφές της υπηρεσίας που πιθανό να υπάρχουν.
Υποχρεωτικό element στο tModel είναι το στοιχείο name το οποίο μεταφράζεται σε ένα URI. Υπόλοιπα στοιχεία που μπορεί να περιλαμβάνει μια δομή δεδομένων tModel είναι: Το description element το οποίο παρέχει μια σύντομη περιγραφή του tModel όπου και στην περίπτωση αυτή δίνεται σε μορφή κειμένου. Το overviewDoc element το οποίο περιέχει το URL της εξωτερικής τεχνικής περιγραφής καθώς και πληροφορίες για τον τρόπο χρήσης της. Το identifierBag element το οποίο περιλαμβάνει αναγνωριστικά μέσω των οποίων το tModel είναι προσβάσιμο κατά την αναζήτηση σε διάφορα συστήματα ταξινόμησης του tModel. Το categoryBag element το οποίο περιλαμβάνει πληροφορίες που προσδιορίζουν την κατηγορία στην οποία μπορεί να ανήκει το tModel Το Signature element το οποίο επιτρέπει στον πάροχο της υπηρεσίας μέσω ψηφιακής υπογραφής την ταυτοποίηση της αυθεντικότητας του τύπου δόμησης δεδομένων.

Τέλος πέραν των προαναφερθέντων βασικών τύπων δόμησης δεδομένων, υπάρχουν άλλοι δύο τύποι, ο operationalInfo και ο publisherAssertion. Ο πρώτος τύπος εκ των δύο περιλαμβάνει πληροφορίες που προσδιορίζουν την προέλευση καθώς και την τροποποίηση οποιουδήποτε τύπου δόμησης δεδομένων όπως το σε ποιον κόμβο UDDI μπορεί να ανήκει είτε το πότε εκδόθηκε ενώ ο δεύτερος τύπος περιγράφει τον πιθανό συσχετισμό μεταξύ δύο δομών bussinessEntity όπως παραδείγματος χάρη όταν ο ένας πάροχος είναι θυγατρικός του άλλου.



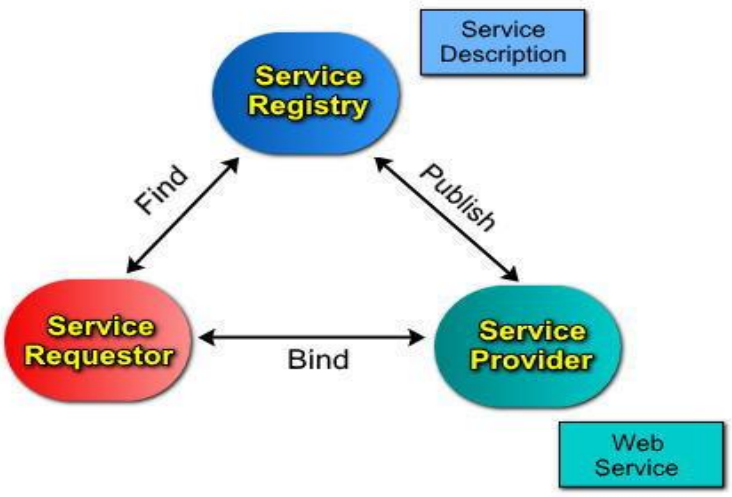
Σχήμα 24. Βασικοί τύποι δόμησης δεδομένων του UDDI Data Model

2.3.4.4 Διαδικασία αναζήτησης και καταχώρησης της πληροφορίας

Η διαδικασία της αναζήτησης και καταχώρησης της πληροφορίας στον κατάλογο UDDI υλοποιείται όπως έχει προαναφερθεί μέσω SOAP μηνυμάτων τα οποία βασίζονται στην XML γλώσσα κάτι που τα καθιστά κατάλληλα και εύχρηστα για έναν μεγάλο αριθμό εφαρμογών ενώ το πρωτόκολλο το οποίο χρησιμοποιείται στο επίπεδο μεταφοράς είναι το HTTP.

Η αναζήτηση των υπηρεσιών λοιπόν στον κατάλογο UDDI γίνεται με την αποστολή μηνύματος SOAP από τον δυνητικό χρήστη (Service Requestor) ενώ ο κατάλογος UDDI απαντά επίσης με ένα μήνυμα SOAP. Το μήνυμα του δυνητικού χρήστη ανάλογα με την πληροφορία που ζητά μπορεί να περιέχει κάποια από τα στοιχεία find_binding, find_business, find_relatedBusinesses, find_service και find_tModel που χρησιμοποιούνται κατά την αναζήτηση πληροφορίας η οποία βρίσκεται στους τύπους δόμησης δεδομένων bindingTemplate, businessEntity, publisherAssertion, businessService και tModel αντίστοιχα. Επίσης χρησιμοποιούνται: το στοιχείο get_bindingDetail για την αναζήτηση βάσει ενός bindingKey που μπορεί να βρίσκεται σε τύπο δόμηση δεδομένων bindingTemplate, το στοιχείο get_businessDetail για την αναζήτηση βάσει ενός bussinessKey που μπορεί να βρίσκεται σε τύπο δόμησης δεδομένων businessEntity, το στοιχείο getoperationalInfo για την αναζήτηση βάσει ενός entityKey που μπορεί να είναι ένα

bindingKey, businessKey, serviceKey ή tModelKey το οποίο βρίσκεται σε ένα τύπο δόμησης δεδομένων entityKey, το στοιχείο get_serviceDetail για την αναζήτηση βάσει ενός serviceKey που βρίσκεται σε ένα τύπο δόμησης δεδομένων businessService και τέλος το στοιχείο get_tModelDetail για την αναζήτηση βάσει ενός tModelKey που βρίσκεται σε ένα τύπο δόμησης δεδομένων tModel.



Σχήμα 25. Αναζήτηση και καταχώρηση πληροφορίας στο UDDI

Αντίστοιχα η καταχώρηση και η τροποποίηση των πληροφοριών που αφορούν τις υπηρεσίες στον κατάλογο UDDI επιτυγχάνεται μέσω ενός μηνύματος SOAP όπου στην περίπτωση αυτή αποστέλλεται από τον πάροχο της υπηρεσίας (Service Provider) προς τον κατάλογο UDDI, ενώ μια επιτυχής καταχώρηση επιβεβαιώνεται με την αποστολή ενός μηνύματος SOAP από πλευράς του καταλόγου UDDI το οποίο είναι κενό. Το μήνυμα του παρόχου ανάλογα με τις πληροφορίες για τις υπηρεσίες που επιθυμεί να καταχωρηθούν μπορεί να περιέχει κάποια από τα ακόλουθα στοιχεία: το στοιχείο add_publisherAssertions όπου χρησιμοποιείται για την προσθήκη ενός publisherAssertion σε κάποιο υπάρχον σύνολο συσχετίσεων μιας δομής δεδομένων businessEntity. Το στοιχείο delete_binding μέσω του οποίου μπορεί να γίνει η διαγραφή ενός η περισσοτέρων bindingTemplate με βάση το bindingKey. Το στοιχείο delete_Business το οποίο χρησιμοποιείται για την διαγραφή ενός η περισσοτέρων businessEntity από τον κατάλογο UDDI με βάση το businessKey. Το στοιχείο delete_publisherAssertions το οποίο χρησιμοποιείται για την διαγραφή ενός η περισσοτέρων publisherAssertion από κάποιο σύνολο συσχετίσεων μιας δομής δεδομένων businessEntity. Το στοιχείο delete_service το οποίο χρησιμοποιείται για την διαγραφή ενός η περισσοτέρων businessService από τον κατάλογο UDDI και από κάποια δομή δεδομένων businessEntity. Το στοιχείο delete_tModel όπου χρησιμοποιείται για την διαγραφή ενός η περισσοτέρων tModel. Το στοιχείο get_assertionStatusReport το οποίο χρησιμοποιείται για την

ανάκτηση μιας περιγραφής της κατάστασης πιθανών συσχετίσεων του businessEntity ενός παρόχου υπηρεσίας. Το στοιχείο get_publisherAssertions το οποίο χρησιμοποιείται για την εύρεση πιθανών συσχετίσεων του παρόχου της υπηρεσίας. Το στοιχείο get_registeredInfo για την εύρεση όλων των υπαρχόντων businessEntity και tModel τα οποία ελέγχονται από κάποιον πάροχο υπηρεσίας. Το στοιχείο save_binding όπου χρησιμοποιείται για την προσθήκη είτε την τροποποίηση ενός ή περισσοτέρων bindingTemplate ενώ το νέο bindingTemplate θα πρέπει να υπάρχει μέσα στο save_binding. Το στοιχείο save_service όπου χρησιμοποιείται για την προσθήκη είτε την τροποποίηση ενός ή περισσοτέρων businessService ενώ το νέο businessService θα πρέπει να υπάρχει μέσα στο save_service. Το στοιχείο save_tModel όπου χρησιμοποιείται για την προσθήκη είτε την τροποποίηση ενός η περισσοτέρων tModel ενώ το νέο tModel θα πρέπει να υπάρχει μέσα στο save_tModel και τέλος το στοιχείο set_publisherAssertions το οποίο χρησιμοποιείται για την διαχείριση του συνόλου των publisherAssertion που αφορά έναν πάροχο υπηρεσίας.

Σημαντικό είναι επίσης να αναφερθεί πως μεταφορά πληροφορίας σε ένα δίκτυο δεν υφίσταται μόνο κατά την αναζήτηση ή την καταχώρηση της πληροφορίας που αφορά την υπηρεσία αλλά και κατά την διάρκεια εκτέλεσης και λειτουργίας της υπηρεσίας. Κατά την λειτουργία της υπηρεσίας μπορούν να χρησιμοποιηθούν επίσης πρωτόκολλα όπως το SMTP για υπηρεσίες όπου μεσολαβεί η αποστολή email, το FTP για υπηρεσίες όπου γίνεται ανταλλαγή αρχείων είτε και πρωτόκολλα ασφαλείας όπως είναι το HTTPS.

2.4 Έλεγχος ποιότητας υπηρεσιών (QoS) στα Web Services

Η αναζήτηση ενός Web Service από κάποιον δυνητικό χρήστη γίνεται βάση των ποιοτικών χαρακτηριστικών του. Έτσι ένα διαθέσιμο Web Service σε κάθε περίπτωση θα πρέπει να παρέχει απαντήσεις σε ερωτήματα που αφορούν στα ποιοτικά χαρακτηριστικά του και κατ επέκταση στις απαιτήσεις του χρήστη που πρόκειται να καλέσει την υπηρεσία. Έτσι για παράδειγμα ένας δυνητικός χρήστης θα επιλέξει μια υπηρεσία κατά την αναζήτηση στον κατάλογο UDDI μεταξύ άλλων παρόμοιων σε λειτουργικότητα υπηρεσιών βάσει των ποιοτικών χαρακτηριστικών όπως ο χρόνος απόκρισης είτε η αξιοπιστία του web service για την εκάστοτε υλοποίηση εφαρμογής σε ένα σύστημα. Ένας ορισμός που αφορά την ποιότητα των υπηρεσιών των Web Services, την περιγράφει ως το σύνολο των χαρακτηριστικών της υπηρεσίας που της παρέχουν τη δυνατότητα να ικανοποιεί τις εκάστοτε απαιτήσεις.

Βάσει των παραπάνω λοιπόν τα ποιοτικά χαρακτηριστικά θα πρέπει να καθορίζονται με τέτοιο τρόπο ώστε μια επιχείρηση που παρέχει την υπηρεσία να

μπορεί να γνωρίζει τι μπορεί να υποστηρίξει η υπηρεσία αυτή καθώς και ο χρήστης να έχει τη δυνατότητα να κάνει την αναζήτηση της βάσει προκαθορισμένων ποιοτικών κριτηρίων. Έτσι τα ποιοτικά χαρακτηριστικά (QoS) θα μπορούσαν να ταξινομηθούν σε κάποιες κατηγορίες μέσα από τις οποίες να είναι δυνατός ο ποσοτικός προσδιορισμός των εκάστοτε παραμέτρων. Ο διαχωρισμός αυτός των παραμέτρων θα μπορούσε να γίνει με βάση α) τον χρόνο εκτέλεσης, β) τις προδιαγραφές και το κόστος, γ)τα χαρακτηριστικά που αφορούν στην εκτέλεση μιας υπηρεσίας, και δ) την ασφάλεια που παρέχει η υπηρεσία.

Πιο αναλυτικά για την καθεμία:

Ποιοτικά χαρακτηριστικά (QoS) ως προς το χρόνο εκτέλεσης: Αυτά μπορούν να αφορούν τις παρακάτω παραμέτρους

Scalability (επεκτασιμότητα): όπου έχει να κάνει με το κατά πόσο από την πλευρά του παρόχου είναι ευέλικτο το σύστημα για την επεξεργασία ενός αριθμού διεργασιών σε συγκεκριμένο χρόνο.

Capacity (χωρητικότητα): το πόσες ταυτόχρονες συνδέσεις μπορεί να εξυπηρετήσει ταυτόχρονα μια υπηρεσία.

Performance (απόδοση) η οποία εξαρτάται από το response time (χρόνος απόκρισης), το latency (λανθάνων χρόνος), το throughput και το execution time (χρόνος εκτέλεσης). Ως performance ορίζεται η παράμετρος όπου εκφράζει τον χρόνο κατά τον οποίο μπορεί να ολοκληρωθεί μια αίτηση για μια υπηρεσία. Το response time όπου θα πρέπει να είναι όσο το δυνατό μικρό αφορά τον μέγιστο χρόνο που απαιτείται για την ολοκλήρωση ενός αιτήματος για μια υπηρεσία, το latency που επίσης θα πρέπει να είναι σύντομο αφορά το χρονικό διάστημα που θα παρέλθει για την εξυπηρέτηση του αιτήματος για μια υπηρεσία από τη στιγμή που αυτό θα ληφθεί, το throughput το οποίο θα πρέπει να είναι υψηλό, εξαρτάται άμεσα από την χωρητικότητα και τον λανθάνων χρόνο είναι ο αριθμός των αιτημάτων που εξυπηρετούνται σε μια συγκεκριμένη χρονική στιγμή ενώ το execution time που το ιδανικό είναι να είναι αρκετά μικρό αφορά στο χρονικό διάστημα που απαιτείται για την εκτέλεση μιας σειράς λειτουργιών.

Reliability (αξιοπιστία): είναι η παράμετρος όπου εκφράζει το κατά πόσο ένα web service μπορεί να ανταπεξέλθει στα ποιοτικά χαρακτηριστικά του. Μπορεί να συσχετιστεί με το πλήθος των αποτυχιών σε ποσοστό συγκεκριμένου χρόνου όπως μια εβδομάδα ή ένα μήνα και γενικότερα με το κατά πόσο κάποια υπηρεσία μπορεί να ανταπεξέλθει στην υλοποίηση αιτημάτων υπό καθορισμένους όρους κατά χρονική αναλογία.

Accuracy (ακρίβεια): το ποσοστό των λαθών που δημιουργούνται από ένα service σε ένα συγκεκριμένο χρονικό διάστημα.

Availability (διαθεσιμότητα): ορίζεται από το λόγο του συνολικού χρόνου όπου το σύστημα λειτουργεί (upTime) προς το άθροισμα του χρόνου αυτού με τον συνολικό χρόνο όπου το σύστημα είναι εκτός λειτουργίας (uptime + downtime = totalTime) και εκφράζεται ως η πιθανότητα το σύστημα να είναι διαθέσιμο τη στιγμή που θα αποσταλεί κάποιο αίτημα για την υπηρεσία.

Robustness (ευρωστία): είναι η παράμετρος όπου εκφράζει το κατά πόσο η υπηρεσία είναι σε θέση να φέρει εις πέρας ένα αίτημα ακόμη και αν το λάβει μη ολοκληρωμένο είτε με εσφαλμένα δεδομένα.

Exception handling (αντιμετώπιση εξαιρετικών περιπτώσεων): αφορά στον τρόπο αντιμετώπισης μιας ενδεχόμενης μη συνηθισμένης περίπτωσης από την υπηρεσία. Όπως για παράδειγμα το πώς θα πρέπει να συνεχίσει στην διεκπεραίωση ενός αιτήματος από τη στιγμή που μπορεί να προκληθεί κάποιο σφάλμα.

Ποιοτικά χαρακτηριστικά (QoS) ως προς την πληρότητα των προδιαγραφών και το κόστος: Αυτά μπορεί να αφορούν παραμέτρους όπως

Supported Standard (συμβατότητα προτύπων) είναι παράμετρος όπου έχει να κάνει με το κατά πόσο συμβατή είναι η υπηρεσία με τα διάφορα βιομηχανικά πρότυπα και κατά συνέπεια πως ανταποκρίνεται σε συνεργασία με άλλες υπηρεσίες.

Completeness (πληρότητα): το κατά πόσο έχουν υλοποιηθεί τα καθορισμένα χαρακτηριστικά μιας υπηρεσίας.

Cost (κόστος): Παράμετρος που επικεντρώνει στην εκτίμηση του κόστους μιας υπηρεσίας. Αυτή μπορεί να εξαρτάται για παράδειγμα από τον όγκο των δεδομένων που απαιτούνται για την εκτέλεσή της.

Ποιοτικά χαρακτηριστικά (QoS) με βάση την διεξαγωγή εκτέλεσης μιας υπηρεσίας:

Αυτά αφορούν κατά κύριο λόγο στην παράμετρο Integrity (ακεραιότητα): Η εκτέλεση μιας υπηρεσίας θεωρείται επιτυχημένη στην περίπτωση που οποιαδήποτε διαδικασία συναλλαγής την απαρτίζει μπορεί να επανέλθει στην αρχική της κατάσταση εάν χρειαστεί λόγω αποτυχίας.

Ποιοτικά χαρακτηριστικά (QoS) με βάση την ασφάλεια μιας υπηρεσίας:

Ως αναφορά το επίπεδο ασφάλειας που θα πρέπει ο κάθε πάροχος να εξασφαλίζει αυτό μπορεί να πλαισιώνεται σε παραμέτρους όπως

Authentication (ταυτοποίηση): αφορά την ταυτοποίηση που παρέχει η υπηρεσία προς τους χρήστες που πρόκειται να έχουν πρόσβαση σε αυτή

Authorization (εξουσιοδότηση): αφορά την εξουσιοδότηση από την υπηρεσία προς συγκεκριμένους χρήστες καθιστώντας τους ως τους μόνους που μπορούν να έχουν πρόσβαση στην συγκεκριμένη υπηρεσία

Confidentiality (εμπιστευτικότητα): σε ποιο βαθμό μπορούν οι εξουσιοδοτούμενοι χρήστες να προσπελάσουν να παρέμβουν και να τροποποιήσουν τα δεδομένα της υπηρεσίας

Traceability (ανιχνευσιμότητα): η δυνατότητα ανίχνευσης του ιστορικού μιας υπηρεσίας αφότου έχει υπάρξει και έχει ολοκληρωθεί η υλοποίηση ενός αιτήματος

Data encryption (κρυπτογράφηση): η δυνατότητα της υπηρεσίας να κρυπτογραφεί τα δεδομένα της

Non-Repudiation (μη αποκήρυξη αιτήματος): η δυνατότητα της υπηρεσίας να παρέχει μια δέσμευση προς τον χρήστη ο οποίος έχει κάνει το αίτημα προς αυτή ώστε να μην μπορεί να αρνηθεί πως το έχει πραγματοποιήσει εφόσον αυτό έχει γίνει.

3. R-OSGi

3.1 Εισαγωγή στο OSGi

Πριν αναλυθεί η τεχνολογία R-OSGi είναι αναγκαίο να γίνει αναφορά σε όλες τις συσχετιζόμενες πτυχές του μοντέλου OSGi. Το πρότυπο OSGi ορίζεται από την OSGi Alliance. Χρησιμοποιείται σε ένα αριθμό συστημάτων όπως για παράδειγμα το Eclipse, καθώς επίσης σε διάφορες υπάρχουσες open-source εφαρμογές όπως Apache Felix, Knopflerfish και Concierge.

Η ιδέα του OSGi (Open Services Gateway Initiative) ξεκίνησε τον Μάρτιο του 1999 όπου προμηθευτές ενσωματωμένων συστημάτων και πάροχοι δικτύων συναίνεσαν στη δημιουργία ενός συνόλου προτύπων για ένα πλαίσιο υπηρεσιών βασισμένο στην Java το οποίο θα παρέχει την δυνατότητα για απομακρυσμένη διαχείριση.

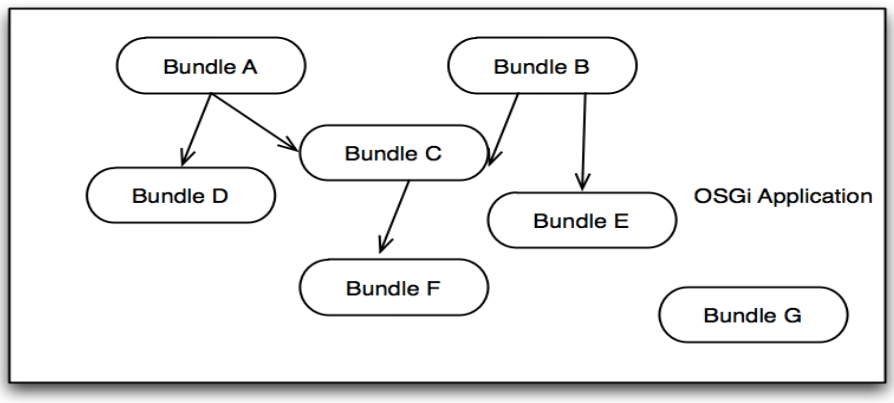
Σκοπός του η δημιουργία ανοιχτών προδιαγραφών για την παράδοση διαχειριζόμενων υπηρεσιών μέσω του δικτύου σε τοπικά δίκτυα και συσκευές μέσω ενός τυποποιημένου τρόπου σύνδεσης των συσκευών με το διαδίκτυο.

Η δυνατότητα λειτουργίας του OSGi σε ενσωματωμένα συστήματα και η ευελιξία του στην αναβάθμιση και προσθήκη νέων υπηρεσιών το καθιστούν ιδανικό για χρήση σε ηλεκτρονικά συστήματα αυτοκινήτων όπως για παράδειγμα στον έλεγχο διαφόρων υποσυστημάτων. Επίσης εφαρμογές για τις οποίες προορίζεται μπορει να αφορούν την σύνδεση έξυπνων Jini συσκευών, ασύρματων Bluetooth συνόλων συσκευών, δρομολογητών είτε αποκωδικοποιητών για υπηρεσία τηλεόρασης. Κάποιες από τις πιο καθιερωμένες εφαρμογές διασύνδεσης και διαχείρισης συσκευών μέσω του διαδικτύου είναι εφαρμογές που σχετίζονται με την παροχή και μέτρηση ενέργειας για το σπίτι, εφαρμογές συστημάτων ασφαλείας μέσω των οποίων ένας ιδιοκτήτης μπορεί να έχει την επίβλεψη και τον έλεγχο απομακρυσμένα από το σπίτι, εφαρμογές συνεχούς επίβλεψης και εντατικής φροντίδας ασθενών, και προγνωστικό έλεγχο μέσω αναφορών για τυχόν δυσλειτουργία οικιακών συσκευών.

Το OSGi πλαίσιο, είναι σήμερα μια πολύ καλά οργανωμένη προσέγγιση για την επίλυση του ζητήματος της δομοστοιχειοποίησης (modularization) των Java εφαρμογών. Αποτελεί ένα μοντέλο για την ανάπτυξη Java εφαρμογών από στοιχεία που ονομάζονται bundles που αλληλεπιδρούν μέσω υπηρεσιών, καθώς επίσης ένα πλαίσιο διαχείρισης του κύκλου ζωής των bundles.

Όπως φαίνεται λοιπόν και στο παρακάτω σχήμα μια εφαρμογή OSGi είναι ουσιαστικά μια συλλογή από στοιχεία τα bundles τα οποία είναι ανεξάρτητα μεταξύ τους και μπορούν να συνεργάζονται. Κάθε bundle παρέχει κάποιες υπηρεσίες και χρησιμοποιεί υπηρεσίες που παρέχονται από άλλα bundles, κάτι το οποίο συνεπάγεται τη μείωση του μεγέθους των εφαρμογών λόγω του ότι μοιράζονται μεγάλο μέρος κώδικα. Κάθε ένα bundle επίσης θα πρέπει να περιγράφει τον εαυτό του και την εξάρτησή του σε σχέση με τα υπόλοιπα bundles.

Ένα από τα χαρακτηριστικά του OSGi είναι η δυνατότητα που παρέχει για δυναμική διαχείριση των bundles. Έτσι νέα bundles μπορούν να προστεθούν και ήδη υπάρχοντα bundles μπορούν να αναβαθμιστούν ή να αφαιρεθούν καθ όλη τη διάρκεια λειτουργίας του συστήματος χωρίς να είναι αναγκαία η επανεκκίνηση της πλατφόρμας, κάτι που καθιστά το OSGi κατάλληλο για περιβάλλοντα υψηλής αξιοπιστίας. Το χαρακτηριστικό αυτό είναι επίσης ιδιαίτερα λειτουργικό για συσκευές όπου είναι αναγκαίο να αναβαθμίζονται αυτομάτως ενώ η λειτουργία τους στο σύστημα συνεχίζεται αδιάκοπα. Σημαντικό είναι επίσης να αναφερθεί πως τα συστήματα OSGi με βάση τα παραπάνω χαρακτηριστικά είναι κατάλληλα για περιπτώσεις όπου συνεισφέρουν περισσότερα από ένα άτομα όπως γίνεται σε εφαρμογές ανοιχτού κώδικα. Η διαχείριση του κύκλου ζωής των εφαρμογών γίνεται μέσω APIs που επιτρέπουν την λήψη των πολιτικών διαχείρισης. Το μητρώο υπηρεσιών επιτρέπει στα bundles να εντοπίσουν την προσθήκη νέων υπηρεσιών ή την κατάργησή τους. Το OSGi διατηρεί την συνοχή σε όλες τις μονάδες (modules) μέσω την παρακολούθησης των εξαρτήσεων μεταξύ των μονάδων αυτών.



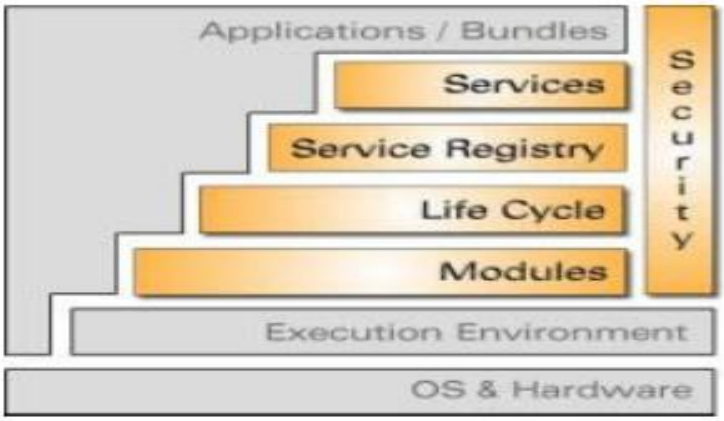
Σχήμα 26. Εφαρμογή OSGi

3.2 Αρχιτεκτονική OSGi

Όλες οι OSGi εφαρμογές είναι γραμμένες σε Java κάτι που συνεπάγεται πως είναι συμβατές σε οποιοδήποτε περιβάλλον υποστηρίζει την προγραμματιστική γλώσσα Java.

Κάθε πλαίσιο που υλοποιεί το πρότυπο OSGi, όπως έχει αναφερθεί παρέχει ένα περιβάλλον για την δομοστοιχειοποίηση (modularization) των εφαρμογών με μικρότερα bundles όπου το καθένα παρέχει κάποιες υπηρεσίες και μπορεί να χρησιμοποιεί υπηρεσίες από άλλα bundles.

Το πλαίσιο αυτό εννοιολογικά χωρίζεται στις ακόλουθες περιοχές όπως αυτές απεικονίζονται και στο σχήμα που ακολουθεί. Σημαντικό να αναφερθεί πως το πλαίσιο αυτό αναφέρεται σε επίπεδο της τοπικής αρχιτεκτονικής του OSGi όπου και η αναζήτηση υπηρεσιών πραγματοποιείται τοπικά και όχι απομακρυσμένα.



Σχήμα 27. Δομή της αρχιτεκτονικής του OSGi

3.2.1 Bundles

Τα bundles προσφέρουν μια λειτουργικότητα στο συνολικό πλαίσιο και υλοποιούνται σαν αρχεία jar. Είναι μια συλλογή από Java κλάσεις, με επιπρόσθετους πόρους εφοδιασμένα με ένα λεπτομερές αρχείο MANIFEST.MF. Στο αρχείο MANIFEST.MF βρίσκονται ο ορισμός των περιεχομένων και των απαιτήσεων του bundle, καθώς επίσης επιπρόσθετες υπηρεσίες που απαιτούνται για να δώσουν στο συμπεριλαμβανόμενο αυτό σύνολο των κλάσεων πιο εξελιγμένες συμπεριφορές στο βαθμό της θεώρησης του συνόλου αυτού ως ένα στοιχείο.

Τα bundles τρέχουν στο ίδιο Virtual Machine και ως εκ τούτου μπορούν να μοιράζονται κώδικα. Για να είναι διαθέσιμο ένα bundle προς άλλα bundles θα πρέπει να εξαχθεί (export) ενώ για να χρησιμοποιηθεί από άλλα bundles θα πρέπει να εισαχθεί (export). Η εξαγωγή στοιχείων σημαίνει πως packages (ένα σύνολο από κλάσεις) γίνονται διαθέσιμα προς άλλα bundles ενώ αντιστοίχως η εισαγωγή στοιχείων σημαίνει πως packages που χρειάζονται να είναι διαθέσιμα εξάγονται από άλλα bundles. Σε περίπτωση που περισσότερα από ένα bundles εξάγουν το ίδιο package (με διαφορετική έκδοση για παράδειγμα), το πλαίσιο επιλέγει μια κατάλληλη έκδοση για κάθε bundle όπου εισάγει αυτό το package.

Παρακάτω ακολουθεί ένα τυπικό αρχείο MANIFEST.MF:

```
Bundle-Name: Hello World
Bundle-SymbolicName: org.example.helloworld
Bundle-Description: A Hello World bundle
Bundle-ManifestVersion: 2
Bundle-Version: 1.0.0
Bundle-Activator: org.example.Activator
Export-Package: org.example.helloworld;version="1.0.0"
Import-Package: org.osgi.framework;version="1.3.0"
```

Η σημασία των περιεχομένων στο παράδειγμα αυτό ακολουθεί παρακάτω:

Bundle Name: Καθορίζει ένα αναγνώσιμο όνομα για το συγκεκριμένο bundle αναθέτοντας του απλά ένα σύντομο όνομα.

Bundle-SymbolicName: Είναι η μόνη απαιτούμενη επικεφαλίδα και δίνει ένα μοναδικό αναγνωριστικό για ένα bundle.

Bundle-Description: Είναι μια περιγραφή για την λειτουργικότητα του bundle.

Bundle-ManifestVersion: Προσδιορίζει την OSGi προδιαγραφή που θα χρησιμοποιηθεί για την ανάγνωση του bundle.

Bundle-Version: Προσδιορίζει τον αριθμό έκδοσης του bundle.

Bundle-Activator: Είναι η κλάση η οποία καλείται κατά την εκκίνηση ή παύση του bundle και αφορά αντίστοιχα στις μεθόδους start και stop. Μια πολύ βασική λειτουργία του BundleActivator είναι το ότι αποθηκεύει το αντικείμενο BundleContext όταν καλείται η μέθοδος start. Το αντικείμενο αυτό είναι πολύ βασικό λόγω του ότι αποτελεί δίοδο για τις λειτουργίες του πλαισίου. Παρακάτω

ακολουθεί ένα απλό παράδειγμα μιας τυπικής κλάσης Java η οποία υλοποιεί την διεπαφή του BundleActivator:

```
package org.exam;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;

public class Activator implements BundleActivator {
        private BundleContext context;

        public void start (BundleContext context) throws Exception {
                System.out.println (“Starting: Hello World”);
                this.context = context;
        }
        public void stop (BundleContext context) throws Exception {
              System.out.println (“Stopping: Goodbye Cruel World”};
              this.context = null;
       }
}
```

Export-Package: Εκφράζει το ποια Java packages που περιέχονται σε ένα bundle θα είναι διαθέσιμα για τον έξω κόσμο.

Import-Package: Προσδιορίζει το ποια Java Packages απαιτούνται από τον έξω κόσμο ώστε να συμπληρώνουν τις εξαρτήσεις που απατούνται σε ένα bundle.

3.2.2 Περιβάλλον εκτέλεσης (Execution environment)

Το επίπεδο του περιβάλλοντος εκτέλεσης προσδιορίζει το περιβάλλον Java υπό το οποίο τρέχει ένα bundle καθώς επίσης τις ελάχιστες απαιτήσεις που υπάρχουν στο περιβάλλον αυτό ως αναφορά την λειτουργία των bundles.

Το περιβάλλον εκτέλεσης είναι αυτό που καθορίζει ποιες είναι οι μέθοδοι και οι κλάσεις οι οποίες είναι διαθέσιμες σε μια συγκεκριμένη πλατφόρμα και οι υλοποιήσεις των bundles μπορεί εξαρτώνται από ένα συγκεκριμένο περιβάλλον εκτέλεσης.

Στην OSGi πλατφόρμα το περιβάλλον εκτέλεσης πληρεί υψηλές απαιτήσεις σε θέματα ασφαλείας βάση της OSGi αρχιτεκτονικής αλλά και των ασφαλών

χαρακτηριστικών της γλώσσας Java. Υποστηριζόμενα περιβάλλοντα εκτέλεσης μπορεί να είναι τα J2SE, CDC, CLDC, MIDP και διάφορα άλλα.

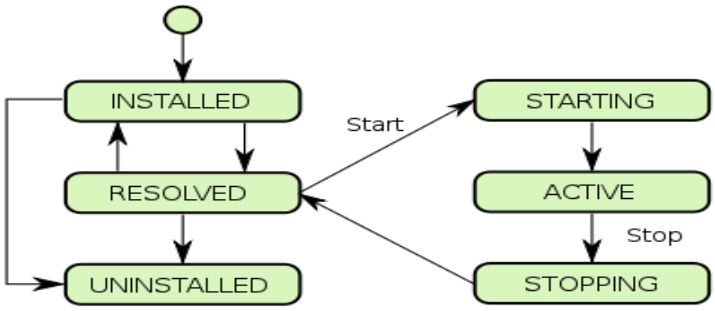
3.2.3 Modules

Είναι το επίπεδο όπου το πλαίσιο OSGi επεξεργάζεται τις πτυχές της δομοστοιχειοποίησης ενός bundle. Πιο συγκεκριμένα αφορά το επίπεδο το οποίο καθορίζει την ενθυλάκωση και την δήλωση των εξαρτήσεων, δηλαδή το πώς ένα bundle μπορεί να εισάγει και να εξάγει κώδικα. Τα μεταδεδομένα τα οποία δίνουν τη δυνατότητα στο πλαίσιο να το επιτύχει αυτό βρίσκονται στο manifest αρχείο του bundle.

Η δομοστοιχειοποίηση (modularity) βρίσκεται ουσιαστικά στον πυρήνα των OSGi προδιαγραφών και ενσωματώνεται στην έννοια των bundles. Όπως αναφέρθηκε ένα bundle υλοποιείται ως ένα JAR αρχείο. Ενώ στην Java οτιδήποτε μέσα σε ένα JAR είναι ορατό προς όλα τα υπόλοιπα JARs, το OSGi κρύβει οτιδήποτε μέσα σε αυτό το JAR παρά μόνο όταν αυτό εξαχθεί.

3.2.4 Κύκλος ζωής (Life Cycle)

Είναι το API για την διαχείριση του κύκλου ζωής δηλαδή για την εγκατάσταση, απεγκατάσταση, έναρξη, παύση και αναβάθμιση των bundles. Ο κύκλος ζωής ενός bundle ξεκινά από την κατάσταση installed και εφόσον όλες οι εξαρτήσεις που ορίζει ικανοποιούνται τότε μεταβαίνει στην κατάσταση resolved. Αφότου γίνει resolved τότε αφού φορτωθούν οι κλάσεις του και είναι διαθέσιμες μπορεί να αρχίσει να τρέχει. Η κατάσταση resolved σηματοδοτεί πως το bundle είτε είναι έτοιμο να ξεκινήσει είτε ότι έχει σταματήσει. Εν συνεχεία το bundle μεταβαίνει στην κατάσταση starting όπου τίθεται σε λειτουργία και θα καλεστεί η μέθοδος BundleActivator.start. Το bundle θα παραμείνει στην κατάσταση αυτή έως ότου ενεργοποιηθεί σύμφωνα με την πολιτική ενεργοποίησης που ακολουθεί. Η κατάσταση active επέρχεται αφού αρχίσει να τρέχει ενώ αν για κάποιο λόγο σταματήσει μεταβαίνει στην κατάσταση stopping όπου καλείται η μέθοδος BundleActivator.stop και έπειτα επιστρέφει και πάλι στην κατάσταση resolved. Τέλος αν το bundle μεταβεί στην κατάσταση uninstalled, σημαίνει πως το bundle έχει απενεργοποιηθεί και έκτοτε δεν μπορεί να μεταβεί σε κάποια άλλη κατάσταση. Η οποιαδήποτε αλλαγή συμβαίνει σε ένα bundle συμβαίνει δυναμικά και τα bundles τις παρακολουθούν και ανάλογα με αυτές αντιδρούν. Έτσι για παράδειγμα σε περίπτωση που κάποιο bundle επέλθει σε κατάσταση installed τα υπόλοιπα bundles έχουν τη δυνατότητα να δουν εάν ενδιαφέρονται για τις υπηρεσίες του.



Σχήμα 28. Κύκλος ζωής των bundles

3.2.5 Μητρώο Υπηρεσιών (Service Registry)

Είναι ένα κεντρικό σημείο καταχώρησης των υπηρεσιών το οποίο αποτελεί το API για την διαχείριση τους.

Το μητρώο υπηρεσιών στο OSGi υποστηρίζει εγγενώς την προσανατολισμένη στις υπηρεσίες αρχιτεκτονική SOA (Service Oriented Architecture). Τα bundles δημοσιεύουν υπηρεσίες στο μητρώο υπηρεσιών και άλλα bundles μπορούν να τις ανακαλύψουν από αυτό.

Οι υπηρεσίες αυτές καταχωρούνται στο μητρώο υπηρεσιών υπό ένα ή περισσότερα ονόματα διεπαφής με προεραιτικά μεταδεδομένα αποθηκευμένα ως προσαρμοσμένες ιδιότητες. Ένα bundle το οποίο ενδιαφέρεται για υπηρεσίες τις αναζητά στο μητρώο υπηρεσιών μέσω του ονόματος της διεπαφής και εν συνεχεία μπορεί να φιλτράρει τις υπηρεσίες με βάση τις προσαρμοσμένες ιδιότητες.

Το μητρώο υπηρεσιών αποτελεί ένα από τα βασικά χαρακτηριστικά του OSGi καθώς προωθεί την έννοια της δυναμικής συνεργασίας μεταξύ των bundles η οποία βασίζεται σε ένα μοντέλο δημοσίευσης/εύρεσης/σύνδεσης (publish/find/bind).

3.2.6 Ασφάλεια (Security)

Είναι το επίπεδο το οποίο αφορά στη διαχείριση των θεμάτων ασφαλείας με τον περιορισμό της λειτουργικότητας του bundle σε προκαθορισμένες δυνατότητες.

Ένας από τους βασικούς στόχους του OSGi είναι να τρέχει εφαρμογές που προέρχονται από μια ποικιλία πόρων υπό τον αυστηρό έλεγχο του συστήματος διαχείρισης. Έτσι ένα ολοκληρωμένο μοντέλο ασφάλειας το οποίο θα αφορά σε όλα

τα μέρη του συστήματος αποτελεί αναγκαία προϋπόθεση. Το OSGi χρησιμοποιεί διάφορους μηχανισμούς για τον σκοπό αυτό.

Το πρώτο σημείο άμυνας αφορά στην εικονική μηχανή (JVM) η οποία είναι σχεδιασμένη να περιορίζει τις δυνατότητες μιας Java εφαρμογής κατά τη διάρκεια του χρόνου εκτέλεσης.

Το δεύτερο σημείο άμυνας αφορά στα ασφαλή χαρακτηριστικά της γλώσσας Java. Ο διαμορφωτές πρόσβασης Java δηλώνουν τους επιτρεπόμενους καλούντες του κώδικα. Αυτός ο μηχανισμός ελέγχου επιτρέπει σε πολλαπλές Java εφαρμογές να βρίσκονται στην ίδια εικονική μηχανή (JVM) και να συνεργάζονται ενώ παράλληλα παρέχεται μια ασπίδα προστασίας για τον κώδικα η οποία και θα έπρεπε να υπάρχει.

Το τρίτο σημείο άμυνας είναι η λεγόμενη Java 2 code based ασφάλεια κατά την οποία κάθε bundle μπορεί να περιοριστεί στις δυνατότητές του έχοντας δικαιώματα (permissions) μόνο στους πόρους για τους οποίους θα έπρεπε να έχει πρόσβαση.

Το τελευταίο σημείο άμυνας αφορά στον αυστηρό διαχωρισμό που παρέχει το OSGi πλαίσιο μεταξύ των bundles. Τα bundles χρειάζονται τα κατάλληλα δικαιώματα ώστε να αλληλεπιδράσουν με άλλα bundles της OSGi πλατφόρμας υπηρεσιών.

Η ανάθεση της πολιτικής διαχείρισης στον διαχειριστή της πλατφόρμας επιτρέπει στην OSGi πλατφόρμα υπηρεσιών να χρησιμοποιείται σε πολλά σενάρια ασφαλείας καθώς επίσης σε σενάρια όπου η ασφάλεια διαχειρίζεται εκτός της OSGi πλατφόρμας υπηρεσιών. Οι OSGi και Java αρχιτεκτονικές ασφαλείας αποτελούν ένα ολοκληρωμένο μοντέλο το οποίο παρέχει στον διαχειριστή της πλατφόρμας να δημιουργήσει ένα από τα πιο ασφαλή περιβάλλοντα στην αγορά.

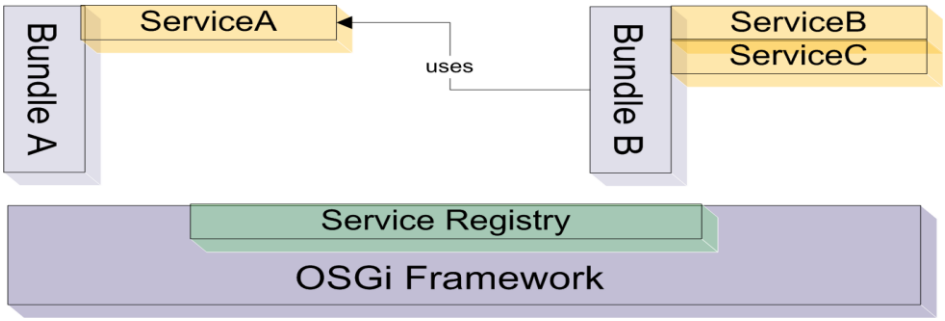
3.2.7 Υπηρεσίες

Το επίπεδο των υπηρεσιών συνδέει με ένα δυναμικό τρόπο τα bundles παρέχοντας ένα μοντέλο δημοσίευσης-εύρεσης-σύνδεσης (publish-find-bind) για τις Java διεπαφές και τα αντικείμενα.

Το OSGi υλοποιεί μια κεντρικοποιημένη υπηρεσιοστρεφής αρχιτεκτονική με χαλαρά συνδεδεμένες υπηρεσίες ενώ στο σχήμα που ακολουθεί απεικονίζεται η συνεργασία μεταξύ των bundles μέσω των υπηρεσιών. Οι υπηρεσίες είναι πλήρως δυναμικές και έχουν τον ίδιο κύκλο ζωής με το bundle που τις παρέχει.

Αποτελούν υπόσταση Java αντικειμένων και επιτρέπουν στα bundles να έρχονται σε συνεργασία, χωρίς ωστόσο να εξαρτώνται από κάποιο πακέτο είτε κάποια υλοποίηση ενός άλλου bundle παρά μόνο από την διεπαφή της υπηρεσίας εφόσον

μια υπηρεσία μπορεί να αναφερθεί μονάχα από την διεπαφή της. Στο μοντέλο OSGi λοιπόν οποιαδήποτε Java κλάση μπορεί να δημοσιευθεί ως μια υπηρεσία που μπορεί να χρησιμοποιείται από άλλα bundles στο σύστημα. Τυπικά, μια υπηρεσία περιλαμβάνει μια υλοποίηση (μια υπόσταση μιας κλάσης), μια ή περισσότερες διεπαφές υπηρεσίας (service interfaces) μέσω των οποίων η υπηρεσία δημοσιεύεται καθώς επίσης ένα σύνολο από ιδιότητες υπηρεσιών (service properties). Οι ιδιότητες επιτρέπουν σε διαφορετικούς παρόχους υπηρεσιών όπου μπορεί να παρέχουν υπηρεσίες με την ίδια διεπαφή (interface), να διαφοροποιούνται. Το πλαίσιο OSGi διατηρεί ένα μητρώο καταχώρησης (service registry) για όλες τις υπηρεσίες που δημοσιεύονται στο σύστημα. Τα bundles μπορούν να ανακτήσουν τις υπηρεσίες μέσω του ονόματος των διεπαφών τους και προαιρετικά χρησιμοποιούν το LDAP-style φίλτρο σχετικά με τις ιδιότητες της υπηρεσίας για υψηλότερη επιλεκτικότητα.Το LDAP (Lightweight Directory Access Protocol)είναι ένα πρωτόκολλο εφαρμογής για την πρόσβαση και την διατήρηση κατανεμημένων υπηρεσιών πληροφοριών καταλόγου

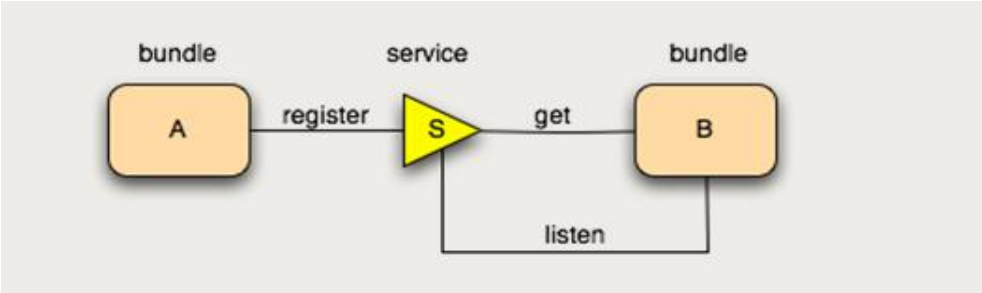


Σχήμα 29. OSGi πλαίσιο με bundles και υπηρεσίες

Ένα client bundle μπορεί να συνδεθεί στο αντικείμενο της υπηρεσίας και να επικαλεστεί λειτουργίες. Το OSGi αποστέλλει events κάθε φορά που θα αλλάξει η κατάσταση μιας υπηρεσίας ώστε η πρόσβαση στα αντικείμενα να είναι ελεγχόμενη. Έτσι με τα γεγονότα τα οποία αφορούν στην διαθεσιμότητα της υπηρεσίας όπως για παράδειγμα την εμφάνιση ή την εξαφάνιση της, το OSGi προβλέπει την ειδοποίηση των ενδιαφερόμενων.

Ένα βασικό χαρακτηριστικό του OSGi συστήματος είναι ότι το bundle που θα χρησιμοποιήσει την υπηρεσία δεν γνωρίζει το bundle το οποίο παράγει την υπηρεσία αυτή ή τον τύπο της υλοποίησης έτσι το σύστημα ενώ αποτελεί ένα σύστημα συνεργασίας παραμένει χαλαρά συνδεδεμένο. Βάσει αυτού το OSGi σύστημα αποτελεί ουσιαστικά μια επέκταση της Java στο οποίο ωστόσο οι εξαρτήσεις και η ορατότητα των πακέτων είναι δυναμικά κάτι που καθιστά το σύστημα πιο ευέλικτο και βελτιστοποιεί την απόδοση των εφαρμογών. Η

καταχώρηση, η χρήση και η ενημέρωση για την εμφάνιση ή την εξαφάνιση της ίδιας υπηρεσίας μπορεί να υφίσταται για οποιονδήποτε αριθμό από bundles.



Σχήμα 30. Καταχώρηση χρήση και ενημέρωση διαθεσιμότητας υπηρεσιών

Η OSGi πλατφόρμα υπηρεσιών είναι ένα πολύ δυναμικό περιβάλλον και μάλιστα πολύ περισσότερο από τα παραδοσιακά μοντέλα των Java εφαρμογών λόγω του ότι τα bundles μπορούν να εγκατασταθούν ή να απεγκατασταθούν οποιαδήποτε στιγμή. Αυτό ωστόσο δημιουργεί μια επιπρόσθετη πολυπλοκότητα για τον προγραμματιστή. Το OSGi διευθετεί το πρόβλημα αυτό με τις παρακάτω βοηθητικές λειτουργίες:

Service Tracker: Απλοποιεί σημαντικά την διαχείριση των υπηρεσιών παρέχοντας μια κλάση η οποία ιχνηλατεί τις υπηρεσίες για την εφαρμογή. Η εφαρμογή παρέχει τρείς μεθόδους οι οποίες καλούνται: α) για υπάρχουσες ή νέες υπηρεσίες, β)όταν διαμορφώνονται οι ιδιότητες μιας υπηρεσίας και γ) όταν οι υπηρεσίες είναι μη καταχωρημένες ή ο tracker είναι κλειστός.

Declarative Services: Τα Declarative Services (DS) αποτελούνται από συνθετικά μέρη (components) τα οποία περιέχουν ένα XML αρχείο (component.xml) καθώς και μια κλάση η οποία υλοποιεί τις υπηρεσίες. Μέσω των declarative services παρέχεται η δυνατότητα για καθορισμό υπηρεσιών μέσω μεταδεδομένων XML. Το Service Component Runtime (SCR) είναι ένα υποσύστημα το οποίο μπορεί να διαβάσει μια XML δήλωση από ένα bundle με καταχώρηση υπηρεσίας και εξαρτήσεις υπηρεσίας. και είναι θέση να καταχωρήσει τις δηλωμένες υπηρεσίες εκ μέρους του bundle. Μέσω των declarative Services μειώνεται το αποτύπωμα της συσκευής λόγω του ότι οι πόροι καταναλώνονται μόνο από τα bundles που χρειάζεται.

3.2.7.1 Βασικές υπηρεσίες OSGi

Το πρότυπο OSGi καθορίζει έναν αριθμό από προτυποποιημένες υπηρεσίες που θα πρέπει να περιέχονται σε μια συμβατή υλοποίηση και οι οποίες επιτρέπουν την

εύκολη ανάπτυξη των εφαρμογών. Εν συνεχεία παρατίθενται οι σημαντικότερες από αυτές ενώ είναι σημαντικό να αναφερθεί πως καθορίζονται αφηρημένα και υλοποιούνται ανεξάρτητα από τον εκάστοτε πάροχο.

Υπηρεσίες πλαισίου (Framework Services):

Το πλαίσιο OSGi μπορεί να παρέχει τις υπηρεσίες Permission Admin, Conditional Permission Admin, Package Admin, URL Handler και Start level. Οι υπηρεσίες αυτές είναι υπηρεσίες που αφορούν άμεσα στη διαχείριση της λειτουργίας του πλαισίου.

Permission Admin: Τα δικαιώματα των τρεχόντων ή μελλοντικών bundles μπορούν να διαχειριστούν μέσω αυτής της υπηρεσίας. Τα δικαιώματα αυτά ενεργοποιούνται άμεσα μόλις οριστούν.

Conditional Permission Admin: Η συγκεκριμένη υπηρεσία επεκτείνει την Permission Admin με δικαιώματα τα οποία μπορούν να ισχύουν όταν ορισμένες συνθήκες όπως η υπογραφή και η τοποθεσία του bundle είναι αληθείς ή ψευδείς κατά την στιγμή όπου τα δικαιώματα ελέγχονται. Ο αριθμός των συνθηκών μπορεί να επεκταθεί με προσαρμοσμένο κώδικα.

Package Admin: Τα bundles μοιράζονται packages με κλάσεις και πόρους. Η ενημέρωση των bundles πιθανό να απαιτεί από το σύστημα να υπολογιστούν εκ νέου οι εξαρτήσεις. Η υπηρεσία αυτή παρέχει πληροφορία σχετικά με την πραγματική κατάσταση διαμοιρασμού των packages του συστήματος καθώς επίσης μπορεί να ανανεώσει τα μοιραζόμενα packages σπάζοντας έτσι τις εξαρτήσεις και υπολογίζοντάς τις εκ νέου.

Start Level: Η υπηρεσία Start Level ορίζει το τρέχων επίπεδο εκκίνησης, αναθέτει ένα bundle στο επίπεδο εκκίνησης και εξετάζει τις τρέχουσες ρυθμίσεις.

URL Hanlders: Η υπηρεσία αυτή παρέχει στα bundles τη δυνατότητα να συνεισφέρουν δυναμικά στην διαχείριση περιεχομένων στην κλάση URL. Το bundle που υλοποιεί την υπηρεσία URL Handlers ακούει στην καταχώρηση stream και content handler υπηρεσιών και μόλις καταχωρηθούν τότε τις προσθέτει στη λίστα των διαχειριστών τους όπου θα χρησιμοποιηθούν για αιτήματα υπηρεσιών URL.

Υπηρεσίες συστήματος (System Services):

Οι υπηρεσίες συστήματος παρέχουν απαραίτητες λειτουργίες για κάθε σύστημα και είναι υπηρεσίες όπως οι Log Service, Configuration Admin service, Device Access

service, User Admin service, IO Connector service, Event Admin service και Preferences service.

Log Service: Η συγκεκριμένη υπηρεσία χρησιμοποιείται από τα bundles για την καταγραφή γεγονότων που θεωρούνται σημαντικά. Όπως πληροφορίες προειδοποίησης ή πληροφορίες εντοπισμού σφαλμάτων. Λαμβάνει καταχωρήσεις γεγονότων και στην συνέχεια τις αποστέλλει σε άλλα bundles που έχουν εγγραφεί στην πληροφορία αυτή.

Configuration Admin: Η υπηρεσία αυτή παρέχει ένα ευέλικτο και δυναμικό μοντέλο για καθορισμό και την λήψη πληροφορίας configuration.

Event Admin: Παρέχει ένα γενικό και ευέλικτο μηχανισμό δημοσίευσης και συνδρομής γεγονότων. Η υπηρεσία υποστηρίζει τόσο την σύγχρονη όσο και την ασύγχρονη παράδοση των γεγονότων. Είναι μηχανισμός δηλαδή που αφορά στην επικοινωνία μεταξύ bundles βασισμένος στο publish-subscribe μοντέλο.

Device Access: Αποτελεί ένα μηχανισμό για την αντιστοίχηση ενός driver σε μια συσκευή και την αυτόματη φόρτωση ενός bundle το οποίο υλοποιεί αυτό τον οδηγό. Αφορά δηλαδή στον συντονισμό της αυτόματης ανίχνευσης και σύνδεσης των υφιστάμενων συσκευών και χρησιμοποιείται για σενάρια Plug n Play.

User Admin: Η υπηρεσία αυτή χρησιμοποιεί μια βάση δεδομένων με πληροφορίες χρηστών δημόσια και ιδιωτική για σκοπούς ταυτοποίησης και αδειοδότητης.

IO Connector: Η υπηρεσία IO Connector επιτρέπει στα bundles να παρέχουν νέα και εναλλακτικά σχήματα πρωτοκόλλων.

Preferences Service: Είναι μια υπηρεσία που παρέχει πρόσβαση σε μια ιεραρχική βάση δεδομένων από ιδιότητες.

Device Management Tree: Παρέχει ένα δένδρο όπου περιέχει πληροφορία διαχείρισης συσκευών, οι υπηρεσίες χρησιμοποιούνται για να παρέχουν το περιεχόμενο για το δέντρο αυτό.

Deployment Admin: Τυποποιεί την πρόσβαση σε ορισμένες από τις αρμοδιότητες του agent διαχείρισης.

Application Admin: Η υπηρεσία αυτή παρέχει ένα μοντέλο όπου οι εφαρμογές μπορούν να εγγραφούν και να ενεργοποιηθούν κατ απαίτηση. Απλοποιεί δηλαδή την διαχείριση ενός περιβάλλοντος με πολλούς διαφορετικούς τύπους εφαρμογών όπου είναι ταυτόχρονα διαθέσιμες.

Monitoring: Παρέχει ένα πρότυπο τρόπο για τα bundles ώστε να παρέχουν τα δεδομένα των επιδόσεών τους.

Foreign Applications: Το πλαίσιο OSGi είναι αρκετά κατάλληλο για την ανάπτυξη μη OSGi εφαρμογών. Μέσω της υπηρεσίας αυτής δίνεται η δυνατότητα σε άλλα μοντέλα εφαρμογών για τη χρήση ενός API ώστε να υπάρχει πρόσβαση σε λειτουργίες του OSGi.

Υπηρεσίες Πρωτοκόλλου (Protocol Services)

Η OSGi Aliance έχει καθορίσει κάποιες υπηρεσίες οι οποίες χαρτογραφούν ένα εξωτερικό πρωτόκολλο σε μια OSGi υπηρεσία.

HTTP Service: Επιτρέπει την αποστολή και λήψη της πληροφορίας από το OSGi με χρήση του HTTP. Μέσω της υπηρεσίας αυτής τα bundles μπορούν να παρέχουν servlets τα οποία είναι διαθέσιμα μέσω http. Η ευχέρεια αυτή στην δυναμική ενημέρωση της OSGi πλατφόρμας υπηρεσιών καθιστά την υπηρεσία HTTP ένα αρκετά ελκυστικό web εξυπηρετητή ο οποίος μπορεί να ενημερωθεί με νέα servlets, αν είναι αναγκαίο και απομακρυσμένα χωρίς να είναι απαραίτητη η επανεκκίνηση.

UPnP: Καθορίζει το πώς τα OSGi bundles μπορούν να αναπτυχθούν για διαλειτουργικότητα με συσκευές UPnP. Η υπηρεσία UPnP του OSGi χαρτογραφεί συσκευές σε ένα UPnP δίκτυο στο μητρώο καταχώρησης υπηρεσιών.

Άλλες υπηρεσίες:

Wire Admin: Επιτρέπει την σύνδεση μεταξύ μιας υπηρεσίας παραγωγού (Producer service) και μιας υπηρεσίας καταναλωτή (Consumer service). Κανονικά τα bundles εγκαθιδρύουν τους κανόνες για να βρουν υπηρεσίες για τις οποίες ενδιαφέρονται. Ωστόσο σε πολλές περιπτώσεις αυτό θα μπορούσε να αποτελεί μια απόφαση ανάπτυξης. Η υπηρεσία Wire Admin Service ως εκ τούτου συνδέει διαφορετικές υπηρεσίες μαζί σαν να έτσι ήταν καθορισμένο από κάποιο configuration. Η υπηρεσία Wire Admin χρησιμοποιεί την έννοια της υπηρεσίας ενός παραγωγού και ενός καταναλωτή οι οποίοι ανταλλάσσουν αντικείμενα καλωδιακά.

XML Parser: Η υπηρεσία XML Parser επιτρέπει σε ένα bundle να εντοπίσει ένα parser με επιθυμητές ιδιότητες και συμβατότητα με JAXP (Java API for Processing). Το JAXP είναι ένα από τα Java XML APIs και παρέχει τη δυνατότητα της επικύρωσης και ανάλυσης XML εγγράφων.

3.3 Το πρότυπο OSGi Whiteboard

Τυπικά, το πρότυπο publish/subscribe χρησιμοποιείται για τον ακόλουθο σκοπό: Κάθε πηγή γεγονότων διατηρεί το δικό της μητρώο από συνδρομητές ακροατές και

διανέμει γεγονότα σε όλους τους συνδρομητές όπως αυτά λαμβάνουν χώρο. Το πρότυπο Whiteboard όπου χρησιμοποιείται στο OSGi, απλοποιεί αυτή τη διαδικασία. Αντί να απαιτείται κάθε ακροατής να κάνει συνδρομή σε μεμονωμένα γεγονότα και η πηγή να κρατά αυτές τις συνδρομές, χρησιμοποιείται το μητρώο υπηρεσιών (service registry) του OSGi. Οι ακροατές καταχωρούν τους εαυτούς τους υπό μια ειδική διεπαφή υπηρεσίας ακροατή. Εφόσον γίνει αυτό, ο ακροατής δεν απαιτείται να ανιχνεύει δυναμικά όλες τις πηγές των γεγονότων, αντιθέτως έχει αφανώς αποκτήσει μια καθολική συνδρομή σε όλες τις ήδη υπάρχουσες και μελλοντικές πηγές γεγονότων. Έτσι το μητρώο OSGi είναι ο πίνακας (whiteboard) στον οποίο μπορούν να κάνουν συνδρομή όλοι οι ακροατές. Οι πηγές γεγονότων μπορούν να ανακτήσουν όλους τους εγγεγραμμένους ακροατές οποτεδήποτε εμφανιστεί ένα γεγονός. Με μια τέτοια προσέγγιση, ο ακροατής μπορεί να θέσει τη συνδρομή ακόμη και αν η πηγή δεν είναι παρούσα την τρέχουσα στιγμή. Έχει αποδειχθεί πως το πρότυπο whiteboard είναι συχνά πιο αποτελεσματικό από το παραδοσιακό πρότυπο publish/subscribe σε συνάρτηση με το μέγεθος του κώδικα και τον συνολικό αριθμό κλάσεων που προκύπτει.

3.4 Η R-OSGi προσέγγιση

Το R-OSGi (Remoting-OSGi) δίνει τη δυνατότητα σε μια κεντρικοποιημένη OSGi εφαρμογή να είναι διαφανώς κατανεμημένη μέσω της χρήσης διαμεσολαβητών (proxies). Στην παρακάτω εικόνα φαίνεται ένα απλοποιημένο παράδειγμα με έναν πάροχο υπηρεσίας (I) και έναν καταναλωτή (J). Το πρωτόκολλο R- OSGi στον proxy χρησιμοποιείται για να κάνει απομακρυσμένες κλήσεις στην υπηρεσία, η οποία βρίσκεται στον peer I, ενώ γεγονότα από τον I προωθούνται διαφανώς στον J και προκύπτουν σαν να είχαν εκδοθεί από ένα τοπικό bundle. Η μόνη διαφορά μεταξύ τοπικών και απομακρυσμένων υπηρεσιών είναι επιπρόσθετες ιδιότητες που δίνουν τη δυνατότητα στις υπηρεσίες έχοντας επίγνωση της κατανομής να εκτελούν εξειδικευμένες λειτουργίες όπως λειτουργίες που αφορούν σε διαχείριση του συστήματος.

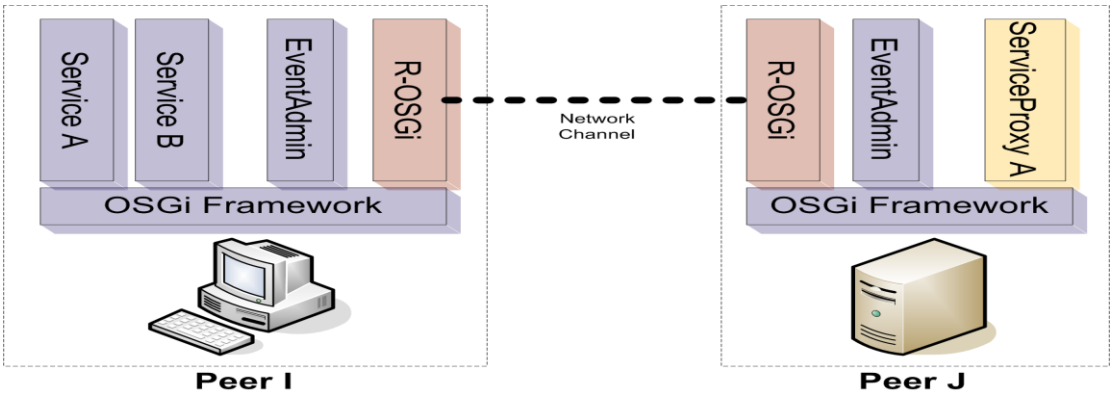
Το R-OSGi χρησιμοποιεί τέσσερις κύριες τεχνικές για να επιτύχει το στόχο της διαφάνειας:

1. Δυναμική παραγωγή proxy (dynamic proxy generation) κατά το χρόνο σύνδεσης για την επίκληση υπηρεσιών.
2. Ένα κατανεμημένο μητρώο υπηρεσιών βασισμένο σε SLP (Service Location Protocol).
3. Χαρτογράφηση (mapping) των αποτυχιών του δικτύου και απομακρυσμένων αποτυχιών σε hotplug events τοπικής μονάδας (γεγονότα που συμβαίνουν

κατά τη διάρκεια λειτουργίας του συστήματος όπου το σύστημα θα πρέπει να είναι σε θέση να αντιληφθεί άμεσα την αλλαγή).

4. Type injection για την επίλυση των κατανεμημένων εξαρτήσεων του συστήματος.

Θα υπάρξει εκτενέστερη περιγραφή παρακάτω.



Σχήμα 31. Επισκόπηση της αρχιτεκτονικής R-OSGi

3.4.1 Δυναμικοί διαμεσολαβητές υπηρεσίας

Το R-OSGi δημιουργεί διαφανείς πελάτες διαμεσολαβητές για απομακρυσμένες υπηρεσίες. Στον πελάτη, οι διαμεσολαβητές αυτοί συμπεριφέρονται ως μια τοπική υπηρεσία και επίσης παρέχονται από τοπικής υπόστασης bundles. Ένα proxy bundle ανακατευθύνει όλες τις κλήσεις μεθόδων στην υπηρεσία που βρίσκεται στην απομακρυσμένη μηχανή και η οποία μεταδίδει το αποτέλεσμα από την ανάκληση της μεθόδου στον τοπικό πελάτη.

Η προσέγγιση της δυναμικής παραγωγής του κωδικού διαμεσολάβησης στον πελάτη, διευκολύνει την αυθόρμητη αλληλεπίδραση μεταξύ υπηρεσιών, επίσης μειώνει στο ελάχιστο τα δεδομένα (με τη μορφή Java bytecode) τα οποία πρέπει να αποθηκευθούν στον εξυπηρετητή ή να μεταφερθούν μέσω του δικτύου όταν ένας client συνδέεται ή ζητά μια υπηρεσία.

Η τυπική πληροφορία που απαιτείται για την δημιουργία proxy για μια συγκεκριμένη διεπαφή υπηρεσίας καθορίζεται από την ανάλυση bytecode της υπηρεσίας όταν αυτή καταχωρείται. Όταν ένας client καλεί την διεπαφή της υπηρεσίας, ο πάροχος της υπηρεσίας ανταποκρίνεται με το αντίστοιχο Java

bytecode για την διεπαφή μαζί με οποιεσδήποτε σειριακές ιδιότητες της υπηρεσίας.

Από το bytecode της διεπαφής, ο client μπορεί να παράγει μια πλήρης διαμεσολάβηση για την υπηρεσία ενώ δε χρειάζεται να μεταφερθεί κώδικας διαμεσολάβησης. Αυτό είναι ιδιαίτερα χρήσιμο στην περίπτωση όπου servers τρέχουν σε συσκευές περιορισμένων πόρων, δεδομένου ότι το bundle του παρόχου υπηρεσίας δε χρειάζεται να διατηρήσει οποιοδήποτε κωδικό για τον πελάτη διαμεσολαβητή.

Ο Proxy δημιουργείται στην πλευρά του πελάτη μέσω ενός Proxy Generator. Η υλοποίηση του Proxy Generation γίνεται με χρήση διαχείρισης bytecode. Κάθε υλοποίηση μεθόδου αναθέτει την κλήση μεθόδου στο κανάλι δικτύου που παρέχεται από το R-OSGi και επικαλείται την ακόλουθη μέθοδο:

Object invokeMethod(final String serviceURL,
final String methodSignature, final Object[] args)
throws RemoteOSGiException;

Το URL υπηρεσίας είναι γνωστό κατά το χρόνο του proxy generation δεδομένου το ότι η κάθε υπηρεσία έχει τον δικό της proxy. Η υπογραφή μεθόδου (method signature) είναι επίσης μια σταθερά κάθε υλοποίησης μεθόδου. Ο πίνακας ορισμάτων (args array) είναι φτιαγμένος κατά το χρόνο εκτέλεσης με την συνάθροιση των πραγματικών ορισμάτων.

Η υλοποίηση proxy της διεπαφής της υπηρεσίας είναι πακεταρισμένη σε ένα JAR αρχείο μαζί με τη διεπαφή της υπηρεσίας.

Η διεπαφή της υπηρεσίας λαμβάνεται και δημιουργείται μια κλάση η οποία υλοποιεί κάθε μέθοδο που περιγράφεται στην διεπαφή σαν μια απομακρυσμένη επίκληση μεθόδου. Επιπρόσθετα η κλάση υλοποιεί τον BundleActivator για την ενσωμάτωση στην διαχείριση του OSGi κύκλου ζωής.

Κατά την καταχώρηση της απομακρυσμένης υπηρεσίας τα type injections θα πρέπει να έχουν προσδιοριστεί. Αυτά περιγράφονται από το ελάχιστο σύνολο τύπων που θα πρέπει να εισαχθούν ώστε να γίνει μια υπηρεσία διαχειρίσιμη από τον proxy, υπό την προϋπόθεση ότι ο proxy μπορεί να εισάγει οποιοδήποτε package που έχει επίσης εισαχθεί από την υπηρεσία. Τα injections εκπέμπονται ως μέρος της ζήτησης της υπηρεσίας και υλοποιούνται μαζί με την proxy κλάση σε ένα proxy bundle. Το proxy bundle αυτό παρέχει την απομακρυσμένη υπηρεσία και ανακατευθύνει κάθε κλήση μεθόδου στην αρχική υπηρεσία.

3.4.2 Καταχώρηση και τοποθεσία υπηρεσίας

Το OSGi είναι φτιαγμένο γύρω από τη φιλοσοφία μιας κεντρικοποιημένης υπηρεσίας μητρώου καταχώρησης. Για την διαφανή κατανομή των εφαρμογών OSGi, απαιτείται μια κατανεμημένη υλοποίηση μητρώου καταχώρησης. Δεν είναι δυνατή η δημιουργία κατανεμημένου μητρώου υπηρεσίας η οποία θα μοιάζει με ένα τοπικό μητρώο χωρίς την μεταβολή της υλοποίησης του OSGi πλαισίου. Έτσι το R-OSGi λειτουργεί με μια συμπληρωματική υπηρεσία πρωτοκόλλου ανακάλυψης και δημιουργεί διαμεσολαβητές για απομακρυσμένες υπηρεσίες οι οποίοι καταχωρούν τις υπηρεσίες τους με τη συμβατική OSGi υπηρεσία μητρώου. Ως εκ τούτου, τα συμβατικά OSGi bundles μπορούν να χρησιμοποιηθούν και να κατανεμηθούν στο OSGi χωρίς τροποποίηση.

Το OSGi χρησιμοποιεί ένα σαφές μοντέλο διασύνδεσης σύμφωνα με το οποίο το clIent bundle επικαλείται την υπηρεσία μητρώου, η οποία παραδίδει ένα σύνολο αναφορών της υπηρεσίας ως απάντηση. Το αίτημα περιλαμβάνει δύο ορίσματα: το όνομα κλάσης της ζητούμενης υπηρεσίας και ένα filter expression το οποίο μπορεί να χρησιμοποιηθεί για παράδειγμα, για τη διάκριση μεταξύ ταυτόσημων υλοποιήσεων του ίδιου τύπου υπηρεσίας. Τα φίλτρα βασίζονται στο LDAP φίλτρο (RFC 1960). Ένας πελάτης όπου έχει στην κατοχή του μια έγκυρη αναφορά υπηρεσίας μπορεί τότε να επιχειρήσει να δημιουργήσει μια σύνδεση με την υπηρεσία και στη συνέχεια να επικαλεστεί λειτουργίες της.

Καθώς το σαφές μοντέλο διασύνδεσης απλοποιεί την διαχείριση του δικτύου και τις αποτυχίες απομακρυσμένων κόμβων στο R-OSGi, η προσέγγιση της δημιουργίας διαμεσολαβητών για υπηρεσίες εισάγει ένα ενδεχόμενο πρόβλημα επεκτασιμότητας δεδομένου ότι σε ένα μεγάλο κατανεμημένο σύστημα μπορεί να υπάρχει μεγάλος αριθμός κόμβων και μεγάλος αριθμός υπηρεσιών. Η προληπτική ανακοίνωση κάθε υπηρεσίας για την διαθεσιμότητά της και η παραγωγή διαμεσολαβητών από το σύστημα για κάθε διαθέσιμη υπηρεσία μπορεί να αυξήσει την κίνηση στο δίκτυο και να σταματήσει την διεργασία των πόρων στους κόμβους.

Το μητρώο κατανεμημένων υπηρεσιών του R-OSGi δίνει μια λύση σε αυτό πρόβλημα, καθιστώντας την ανίχνευση υπηρεσιών και κατά συνέπεια και την παραγωγή διαμεσολαβητή άμεσα αντιδραστικά σε αυτό. Τα bundles μπορούν να εγγραφούν σε υπηρεσίες του τύπου DiscoveryListener και να θέσουν ιδιότητες για να μεταφέρουν πληροφορία σχετικά με τις διεπαφές των υπηρεσιών για τις οποίες ενδιαφέρονται, περιλαμβάνοντας προαιρετικά ένα φίλτρο χαρακτήρων. Ακολουθώντας το whiteboard pattern το R-OSGi κρατά ένα ίχνος όλων των εγγεγραμμένων υπηρεσιών μέσω της παρακολούθησης των γεγονότων

καταχώρησης υπηρεσιών από την τοπική υπηρεσία μητρώου. Εκκινεί την ανίχνευση των απομακρυσμένων υπηρεσιών οποτεδήποτε υπάρξει μια οντότητα στο σύστημα όπου ανακοινώνει τη ζήτηση για μια υπηρεσία.

Ομοίως, οι peers ανακοινώνουν στο δίκτυο τις παροχές τους σε υπηρεσίες και επιτρέπουν την απομακρυσμένη πρόσβαση σε αυτές σύμφωνα με την τοπικά προκαθορισμένη πολιτική. Κάθε φορά που μια υπηρεσία εγγράφεται με το τοπικό πλαίσιο με ιδιότητες που υποδηλώνουν πως θα έπρεπε να προσφέρονται απομακρυσμένα, το R-OSGi ενεργοποιεί την καταχώρηση της υπηρεσίας αυτής με το discovery layer απομακρυσμένων υπηρεσιών.

Σαφές καθορισμός του ποιες υπηρεσίες προσφέρονται για απομακρυσμένη πρόσβαση με αυτόν τον τρόπο παρέχεται από την εφαρμογή. Εναλλακτικά ένα υποκατάστατο bundle που είναι ξεχωριστά από την εφαρμογή και υπάρχει στον ίδιο κόμβο μπορεί να ακούσει σε τοπικές καταχωρήσεις υπηρεσιών, και επιλεκτικά να επανεξάγει κάποιες υπηρεσίες απομακρυσμένα χωρίς να απαιτεί από την ίδια την εφαρμογή να είναι να λειτουργεί κατανεμημένα.

Το R-OSGi υλοποιεί το κατανεμημένο μητρώο καταχώρησης υπηρεσίας χρησιμοποιώντας το Service Location Protocol (SLP) ως βασικό μηχανισμό. Το SLP έχει διάφορα δεσμευτικά χαρακτηριστικά για το R-OSGi όπως την προσαρμοστικότητά του, την ενδογενώς κατανεμημένη διαδικασία αναζήτησης και την ομοιότητα με το OSGi στην ονομασία των υπηρεσιών. Για την χρήση του SLP ως ένα πλήρως αποκεντρωμένο μητρώο καταχώρησης υπηρεσίας, γίνεται εκμετάλλευση της προσαρμοστικής συμπεριφοράς του SLP πρωτοκόλλου. Στο SLP όταν υπάρχει ένας αφοσιωμένος Directory Agent, οι clients επικοινωνούν αποκλειστικά με αυτόν τον κεντρικό server καταχώρησης. Αν δεν υπάρχει Directory Agent οι πελάτες χρησιμοποιούν multicast. Μέσω αυτού του χαρακτηριστικού το R-OSGi υλοποιεί ένα κατανεμημένο SLP layer το οποίο μπορεί να χρησιμοποιηθεί σε ένα μεγάλο εύρος περιπτώσεων.

Όσον αφορά την ονομασία, τόσο το OSGi όσο και το SLP αναγνωρίζουν μια υπηρεσία από ένα μοναδικό string. Στο OSGi αυτό είναι το όνομα της διεπαφής υπό το οποίο έχει καταχωρηθεί η υπηρεσία. Στο SLP το όνομα είναι ένα URL υπηρεσίας της μορφής service:serviceType://URL όπου ο τύπος υπηρεσίας είναι της μορφής abstractType:concreteType. Περιγράφοντας όλες τις OSGi υπηρεσίες με τον ίδιο abstract type service:osgi και χρησιμοποιώντας το όνομα της διεπαφής ως concrete type, προκύπτει μια αμφίδρομη χαρτογράφηση μεταξύ OSGi και SLP υπηρεσιών. Το OSGi υποστηρίζει LDAPv2 φίλτρο για τις ιδιότητες των υπηρεσιών ώστε να καταστεί δυνατή η περισσότερο δηλωτική και λεπτομερής η αντιστοίχηση των υπηρεσιών. Αυτό το χαρακτηριστικό γίνεται ιδιαίτερα χρήσιμο όταν το μητρώο καταχώρησης υπηρεσιών δεν είναι πλέον κεντρικό αλλά κατανεμημένο.

Αφότου η υπηρεσία ανακαλυφθεί, το R-OSGi εισάγει ένα ενδιάμεσο βήμα πριν η υπηρεσία παραδοθεί. Αυτό είναι σημαντικό για λόγους ασφαλείας καθώς επιτρέπει σε χρήστες για παράδειγμα να δουν τις διαθέσιμες απομακρυσμένες σε έναGUI (Graphical User Interface) πριν συνδεθούν σε αυτές. Με ένα τέτοιο βήμα, το R-OSGi αντιστοιχίζει την συμπεριφορά του OSGi, η οποία επίσης χρησιμοποιεί μια εμμεσότητα για τις αναφορές των υπηρεσιών. Το R-OSGi επίσης υποστηρίζει τη ρητή σύνδεση σε ένα απομακρυσμένο peer εάν η εφαρμογή έχει εκ των προτέρων τη γνώση της κατανομής των υπηρεσιών στο σύστημα.

3.4.3 Διαφανής κατανομή

Ένα κεντρικοποιημένο πρόγραμμα είναι απίθανο να αποδώσει με την αποδεκτή απόδοση και πόσο μάλλον όταν συνυπολογίζονται μέσα σε κατανεμημένα στοιχεία. Τα βασικά προβλήματα εδώ είναι ο λανθάνων χρόνος στην επικοινωνία και η αναξιοπιστία είτε λόγω απώλειας μηνυμάτων είτε λόγω μερικής αποτυχίας από τους κόμβους του δικτύου.

Το R-OSGi παρακάμπτει αυτά τα προβλήματα με την έξυπνη αξιοποίηση του τρόπου με τον οποίο είναι γραμμένα τα προγράμματα OSGI. Επιπροσθέτως, αντί να συγκαλύπτει κατανεμημένες αποτυχίες, εκθέτει τα γεγονότα αυτά στα bundles εφαρμογής αλλά σε μια μορφή όπου το bundle είναι σχεδιασμένο να διαχειρίζεται. Το R-OSGi χαρτογραφεί τις αποτυχίες που προκύπτουν από την κατανομή των στοιχείων σε τοπικά hot-plug γεγονότα. Από το OSGi μοντέλο, οι developers συνηθίζουν να ασφαλίζουν τον κώδικα ενάντια στην περίπτωση μέρη του συστήματος να μην είναι διαθέσιμα. Συνήθως αυτό γίνεται ακούγοντας σε γεγονότα υπηρεσιών είτε χρησιμοποιώντας το OSGi Service-Tracker.

Η κλάση ServiceTracker όπως αναφέρθηκε και παραπάνω απλοποιεί την χρήση των υπηρεσιών από το μητρώο υπηρεσιών του πλαισίου. Ένα αντικείμενο ServiceTracker είναι κατασκευασμένο βάση κριτηρίων αναζήτησης καθώς και ενός αντικειμένου ServiceTracketCostumizer. Ένα ServiceTracker μπορεί να χρησιμοποιήσει ένα ServiceTrackerCostumizer ώστε να προσαρμόσει τα αντικείμενα των υπηρεσιών που πρέπει να ανιχνευθούν και τα οποία ταιριάζουν στα συγκεκριμένα κριτήρια αναζήτησης. Το ServiceTracker διαχειρίζεται λεπτομερώς την ακρόαση γεγονότων ώστε να επιλέξει αν θα χρησιμοποιηθούν υπηρεσίες. Η μέθοδος getServiceReferences μπορεί να κληθεί ώστε να ληφθούν αναφορές των υπηρεσιών όπου έχουν ανιχνευθεί. Οι μέθοδοι getService και getServices μπορούν να κληθούν ώστε να ληφθούν τα αντικείμενα υπηρεσιών για την υπηρεσία που έχει ανιχνευθεί.

Σε ένα καθαρά τοπικό configuration, οι υπηρεσίες μπορούν να τεθούν μη διαθέσιμες όταν κάποια οντότητα στο σύστημα, αποφασίζει να σταματήσει ή να απεγκαταστήσει το bundle όπου παρείχε την υπηρεσία. Με την χαρτογράφηση των δυσλειτουργιών του δικτύου εισάγονται σχέδια αποφυγής της αποτυχίας όπου δεν είναι πιθανά σε καθαρά κεντρικοποιημένες περιπτώσεις. Για παράδειγμα, αν μια υπηρεσία παρέχει peer αποτυχίες, θα ανιχνευθεί η βλάβη του καναλιού του δικτύου και η αποτυχία έτσι ώστε να επανασυνδεθεί. Έχοντας προβλέψει αυτό, το R-OSGi αμέσως απεγκαθιστά το proxy bundle. Ακόμη και αν το δίκτυο λειτουργεί χωρίς αποτυχίες, η υπηρεσία μπορεί να δώσει εξαιρέσεις. Οι εξαιρέσεις αυτές σειριακοποιούνται και επανεισάγονται στο proxy bundle για τον προσδιορισμό της ακριβούς συμπεριφορά της υπηρεσίας.

Οι υπηρεσίες OSGi δεν εγγυώνται για το χρόνο εκτέλεσης είτε είναι τοπικές είτε απομακρυσμένες. Επιπροσθέτως οι υπηρεσίες είναι συχνά event-driven και δεδομένου ότι τα γεγονότα στο OSGi είναι τυπικά απεσταλμένα ασύγχρονα, δε μπορούν να γίνουν υποθέσεις σχετικά με το χρόνο. Αυτή είναι μια υπολογίσιμη διαφορά σε απλά αντικείμενα, τα οποία είναι συχνότερα αναμενόμενο να εκτελέσουν μεθόδους σε πολύ σύντομο χρονικό διάστημα. Μια βασική διαφορά μεταξύ του R-OSGi σε σύγκριση με συστήματα όπως το COBRA είναι πως η διακριτότητα των κατανεμημένων οντοτήτων είναι πολύ μεγαλύτερη. Στο OSGi, οι υπηρεσίες ενθυλακώνουν ολόκληρες λειτουργικές μονάδες και οι εξαρτήσεις μεταξύ των υπηρεσιών είναι αρκετά περιορισμένες στο επίπεδο εφαρμογής. Τα αντικείμενα αντιθέτως τείνουν να έχουν ένα μεγαλύτερο αριθμό και συχνά εμφωλευμένες διασυνδέσεις που κάνουν τις επιπτώσεις του δικτύου πιο σοβαρές.

3.4.4 Type Injection

Στο OSGi όλος ο κώδικας είναι δομοστοιχειωτοποιημένος (modularized) σε bundles και τα εισαχθέντα στοιχεία κώδικα από άλλα bundles πρέπει να είναι ρητά δηλωμένα στο δηλωτικό JAR του bundle. Πολλές είναι επιπτώσεις που μπορεί να προκύπτουν μέσα στο πλαίσιο των proxy bundles. Η διεπαφή της υπηρεσίας μπορεί να χρησιμοποιεί τύπους σε παραμέτρους μεθόδων ή να επιστρέφει τιμές που δεν ανήκουν στις πρότυπες κλάσεις της Java και δε μπορούν να θεωρηθούν ότι υπάρχουν στον πελάτη.

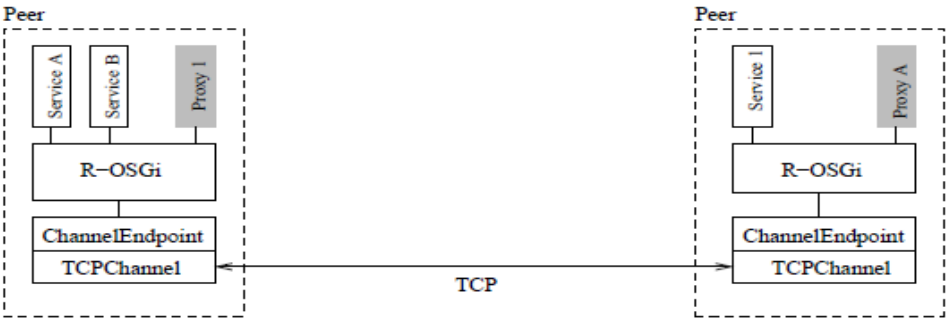
Θα πρέπει να εξασφαλίζεται πως ο παραγόμενος διαμεσολαβητής περιλαμβάνει όλους τους τύπους οι οποίοι χρησιμοποιούνται από μεθόδους της υπηρεσίας. Έτσι το R-OSGi έχει μια ειδική στρατηγική για να εξασφαλιστεί η συνοχή του τύπου για την διεπαφή της υπηρεσίας. Το type injection χρησιμοποιείται για να καθιστά τους διαμεσολαβητές υπηρεσιών αυτάρκεις.

Όταν η απομακρυσμένα προσβάσιμη υπηρεσία έχει καταχωρηθεί, κάθε τύπος που προκύπτει στη διεπαφή της υπηρεσίας παρακολουθείται από μια στατική ανάλυση κώδικα. Αν ο τύπος περιέχεται στο bundle υπηρεσίας και το package είναι δηλωμένο για να εξαχθεί από το bundle υπηρεσίας, η αντίστοιχη κλάση προστίθεται στο λεγόμενο Injection list. Αναφερόμενοι τύποι που δεν περιλαμβάνονται στο bundle υπηρεσίας μένουν εκτός. Τα Injections αποθηκεύονται με την καταχώρηση της υπηρεσίας. Κάθε φορά που ο client καλεί την υπηρεσία, τα Injections εκπέμπονται επιπρόσθετα της διεπαφής της υπηρεσίας και των ιδιοτήτων της υπηρεσίας. Κατά τη διάρκεια του Proxy generation τα Injections υλοποιούνται και αποθηκεύονται στο proxy bundle. Τα Packages από όλες τις injected κλάσεις είναι δηλωμένα σαν εξαχθέντα στοιχεία για την επιβεβαίωση της συνοχής του τύπου μέσα στο πλαίσιο. Κλάσεις των packages java.* και org.ogi.* εξαιρούνται από την όλη διαδικασία δεδομένου ότι θεωρείται πως ανήκουν στο περιβάλλον εκτέλεσης. Το αποτέλεσμα της injection στρατηγικής είναι ένα ελάχιστο σύνολο κλάσεων και εισαχθέντων στοιχείων packages όπου κάνουν την υπηρεσία proxy αυτάρκη και επιλύσιμη. Πέρα από την ανάλυση κώδικα για τον καθορισμό του ελάχιστου συνόλου Injections, οι καταχωρήσεις υπηρεσιών μπορούν να παρέχονται αυτοματοποιημένα με τις κλάσεις που πρόκειται να εισαχθούν στο bundle. Αυτό μπορεί να είναι χρήσιμο σε συγκεκριμένες περιπτώσεις όπως για παράδειγμα αν ένα όρισμα μιας μεθόδου υπηρεσίας είναι μια διεπαφή και ο πάροχος της υπηρεσίας θέλει να προσθέσει μια υπόσταση υλοποίησης αυτής της διεπαφής.

3.4.5 Κανάλια Δικτύου και μεταφορά μηνυμάτων

Η δομή της επικοινωνίας του R-OSGi είναι καθαρά βασισμένη σε μηνύματα. Για αποδοτικότητα στην ανάλυση και τη διαχείριση, όλα τα μηνύματα είναι δυαδικά. Τα μηνύματα αποτελούνται από μια επικεφαλίδα η οποία υποδηλώνει τον τύπο μηνύματος και κάποια κοινά χαρακτηριστικά, καθώς επίσης ένα σώμα με μέρη που αφορούν ειδικά τον τύπο του μηνύματος.

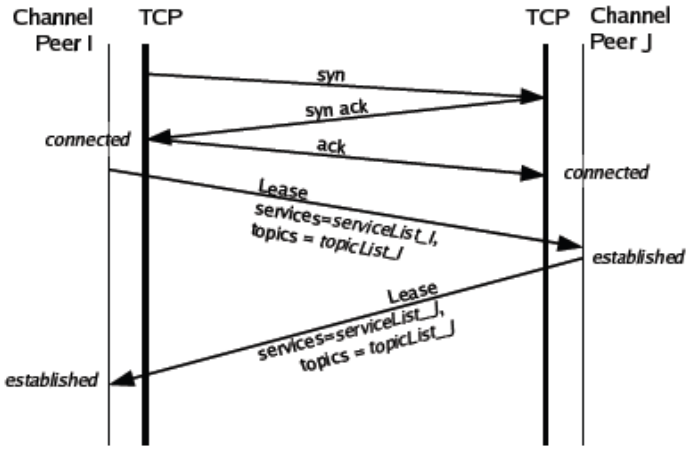
Τα κανάλια δικτύου στο R-OSGi είναι εξ ορισμού TCP συνδέσεις που χρησιμοποιούν την δυνατότητα TCP keep-alive. Όσο δηλαδή υπάρχει κίνηση εντός του χρονικού ορίου η σύνδεση διατηρείται ανοιχτή. Αυτό μειώνει την επιβάρυνση της χειραψίας (handshake) TCP όπου σε διαφορετική περίπτωση θα προηγούνταν ως διαδικασία σε κάθε κλήση υπηρεσίας. Κάθε κανάλι είναι μια σύνδεση ένας προς έναν μεταξύ δύο peers, όπως φαίνεται και στο παρακάτω σχήμα.



Σχήμα 32. Κανάλια δικτύου R-OSGi

Ο επεκτάσιμος σχεδιασμός του R-OSGi επιτρέπει τη σύνδεση και σε εναλλακτικά πρωτόκολλα και δίκτυα μεταφοράς. Μπορεί για παράδειγμα να υπάρχουν υλοποιημένα tunneling R-OSGi μηνυμάτων μέσω http για την υποστήριξη της επικοινωνίας μεταξύ τειχών προστασίας .

Όταν μια σύνδεση μέσω ενός δικτυακού καναλιού ξεκινήσει, οι δύο peers ανταλλάσσουν συμμετρικές συμβάσεις (leases) όπως φαίνεται στην παρακάτω εικόνα. Μια σύμβαση περιλαμβάνει τα ονόματα των υπηρεσιών που προσφέρει ο peer καθώς επίσης τα θέματα γεγονότων για τα οποία ο peer ενδιαφέρεται. Στο R-OSGi μια σύμβαση είναι περισσότερο μια συμφωνία μεταξύ των δύο peers παρά ένας χρονικός περιορισμός, έτσι σε αντίθεση με συστήματα όπως το Jini στο R-OSGi οι συμβάσεις δεν είναι βασισμένες σε κάποια λήξη χρονικού ορίου. Κάθε φορά που ανακοινώνονται αλλαγές στις υπηρεσίες ή στις συνδρομές μέσω των συμβάσεων, ο peer που έχει εκδώσει το lease είναι υποχρεωμένος να ακυρώσει το υπάρχων lease.



Σχήμα 33. Εγκαθίδρυση καναλιού R-OSGi και συμμετρικές συμβάσεις (leases)

3.4.6 Επίκληση μεθόδου

Η επίκληση μεθόδου στο R-OSGi χρησιμοποιεί το δικό της πρωτόκολλο βασισμένο σε μηνύματα. Κάθε επίκληση μεθόδου που αντιστοιχεί σε μια απομακρυσμένη υπηρεσία, μετατρέπεται στην κλήση invokeMethod και αποστέλλεται μέσω του υποκείμενου R-OSGi καναλιού. Ουσιαστικά η υπηρεσία R-OSGi περιλαμβάνει μια γενική μέθοδο η οποία καλείται από κάθε proxy μέθοδο με μια ανακλαστική έννοια. Αυτό οδηγεί επιτυχώς στην εκπομπή ενός συγκεκριμένου μηνύματος το οποίο μεταφέρει την περιγραφή μιας καλούμενης μεθόδου (το service URL και την method signature) μαζί με τα ορίσματα της κλήσης στον απομακρυσμένο peer. Από την πλευρά του παρόχου , το πρώτο βήμα είναι η αναζήτηση της αντίστοιχης υπηρεσίας και η κλήση της μεθόδου της υπηρεσίας. Αυτό είτε επιτυγχάνει και φέρει ως αποτέλεσμα την επιστροφή τιμής, είτε θα δώσει μια εξαίρεση. Σε κάθε περίπτωση παράγεται ένα μήνυμα απόκρισης και αποστέλλεται πίσω στον καλών peer. Η proxy μέθοδος στο σημείο αυτό επιστρέφει είτε το αποτέλεσμα είτε δίνει το αποσειριακοποιημένο αντικείμενο εξαίρεσης. Αυτό μοντελοποιεί την ακριβή συμπεριφορά όπου μπορεί να έχει μια μέθοδος τοπικής υπηρεσίας.

Ένας εμφανής περιορισμός είναι πως η τυπική παράμετρος των απομακρυσμένων υπηρεσιών πρέπει να σειριακοποιείται. Αυτό είναι γνωστό και από πολλούς άλλους μηχανισμούς απομακρυσμένης επίκλησης και προς το παρόν δεν μπορεί να αποφευχθεί και στο R-OSGi.

Δεδομένου το ότι η java περιέχει ήδη RMI (Remote Method Invocation) για απομακρυσμένη επίκληση, θα πρέπει να είναι δικαιολογημένη η μη χρήση του στο R-OSGi. Διάφοροι είναι λόγοι που έχουν οδηγήσει σε μια τέτοια τακτική. Το RMI είναι ένα java API που δίνει τη δυνατότητα για αντικειμενοστραφή προγραμματισμό μέσω της java όπου αντικείμενα σε διαφορετικά απομακρυσμένα hosts μπορούν να αλληλεπιδράσουν σε ένα κατανεμημένο δίκτυο. Δεν όμως είναι διαθέσιμο σε κάθε πλατφόρμα CDC (Connection Device Configuration). Το CDC είναι μια προδιαγραφή που περιγράφει το βασικό σύνολο των βιβλιοθηκών και των χαρακτηριστικών των virtual machine που θα πρέπει να υπάρχουν σε μια υλοποίηση.

Ακόμη και αν το RMI είναι διαθέσιμο, η απόδοση του σε τέτοιες υλοποιήσεις είναι συχνά μη ικανοποιητική. Σε αντίθεση με τα συνήθη απομακρυσμένα αντικείμενα, τα αντικείμενα υπηρεσιών στο OSGi έχουν ένα πολύ καλά καθορισμένο κύκλο ζωής και ως εκ τούτου δεν απαιτούν την επιβάρυνση από μηχανισμούς όπως κατανεμημένους μηχανισμούς συλλογής άχρηστων πληροφοριών.

Επιπλέον, ο καλά καθορισμένος κύκλος ζωής επιτρέπει την διατήρηση των εμμενουσών συνδέσεων ανοιχτών για μεγαλύτερο διάστημα μειώνοντας έτσι την επιβάρυνση των επιπρόσθετων TCP “χειραψιών” .

3.4.7 Απομακρυσμένα γεγονότα

Όπως το UPnP, το R-OSGi υλοποιεί την απομακρυσμένη επίκληση υπηρεσίας καθώς επίσης και μια αρχιτεκτονική βασισμένη σε events. Το R-OSGi χρησιμοποιεί την OSGi αντίληψη των γεγονότων και κάνει χρήση της EventAdmin υπηρεσίας. Η υπηρεσία EventAdmin παρέχει ένα μηχανισμό για την κατανεμημένη δημοσίευση των γεγονότων στους ακροατές και βασίζεται στο μοντέλο publish-subscribe. Το μοντέλο publish-subscribe είναι ένα πρότυπο ανταλλαγής μηνυμάτων όπου οι αποστολείς (publishers) δεν προγραμματίζουν την αποστολή μηνυμάτων απευθείας σε συγκεκριμένους αποδέκτες (subscribers). Αντί αυτού δημοσιεύουν μηνύματα που χαρακτηρίζονται σε κλάσεις χωρίς τη γνώση των subscribers που μπορεί να υπάρχουν και ταυτόχρονα οι subscribers εκδηλώνουν ενδιαφέρον σε μια ή περισσότερες κλάσεις και λαμβάνουν μόνο μηνύματα για τα οποία ενδιαφέρονται χωρίς να έχουν γνώση των publishers που μπορεί να υπάρχουν.

Στο R-OSGi, η υπηρεσία EventAdmin υλοποιείται σαν ένα whiteboard pattern με βάση το κατανεμημένο μητρώο καταχώρησης υπηρεσιών. Ένα bundle καταχωρεί ένα event μέσω της καταχώρησης μιας EventHandler υπηρεσίας μαζί με την ιδιότητα bundle.topics και την προαιρετική ιδιότητα event.filter δηλώνοντας ένα φίλτρο το οποίο αντιστοιχίζεται έναντι στο σύνολο ιδιοτήτων των events που προκύπτουν. Ένα EventHandler ουσιαστικά επεξεργάζεται τα inputs σε ένα πρόγραμμα και στη συγκεκριμένη περίπτωση αφορούν στη διαχείριση των events. Τα θεματικά strings των γεγονότων ακολουθούν μια ιεραρχική δομή. Τα bundles αρχικά καταχωρούν το EventHandler στο τοπικό μητρώο καταχώρησης υπηρεσίας. Η συνδρομή είναι ανακοινωμένη στους peers μέσω μιας συμμετρικής σύμβασης που εκπέμπεται κατά τη φάσης της σύνδεσης. Στην άλλη πλευρά του καναλιού ένα EventHandler είναι καταχωρημένο τοπικά για τα δηλωμένα topics και τυχόν events αντιστοίχησης που μπορεί να εκκρεμούν, αποστέλλονται πίσω μέσω του καναλιού. Για την δημοσίευση ενός event, τα bundles το στέλνουν στην τοπική EventAdmin υπηρεσία που στη συνέχεια τα στέλνει σε όλους τους καταχωρημένους ακροατές listeners.

3.4.8 Παρουσιάσεις (Presentations)

Το γεγονός ότι τα R-OSGi modules αντιμετωπίζονται ως μονάδες για κατανομή, προσφέρει μοναδικές ευκαιρίες για την εξειδίκευση κάποιων από αυτά τα modules. Μια τέτοια εξειδίκευση στο R-OSGi είναι και η ιδέα της παρουσίασης (presentation). Μια παρουσίαση είναι μια κλάση με ένα σχετικό user interface όπου μπορεί να κάνει download ο client για την απλή απομακρυσμένη χρήση. Δηλωμένες παρουσιάσεις εισάγονται αυτόματα στο proxy bundle και καταχωρούνται σαν υπηρεσίες με ένα whiteboard τρόπο. Στην πλευρά του client είναι πιθανό να τρέχει

το προαιρετικό bundle ServiceUI. Το bundle αυτό απεικονίζει την πληροφορία σχετικά με ανακαλυφθέντες υπηρεσίες και επιτρέπει στο χρήστη να ζητήσει τις υπηρεσίες αυτές. Αν η υπηρεσία έχει επισυναπτόμενη μια παρουσίαση, το γραφικό στοιχείο παρέχει ένα Java AWT (Abstract Windowing Toolkit) panel. Το panel αυτό απεικονίζεται σε ένα περιβάλλον καρτελών ώστε να επιτρέπει στο χρήστη να αλληλεπιδρά με πολλές απομακρυσμένες υπηρεσίες. Υπάρχουν για παράδειγμα παρουσιάσεις για τον έλεγχο έξυπνων συσκευών. Στην παρακάτω εικόνα φαίνεται η οθόνη ενός PDA που συνδέεται σε ένα Lego Mindstorms robot μέσω του R-OSGi. Το λογισμικό που ελέγχει το robot είναι σχεδιασμένο με χρήση R-OSGi και περιλαμβάνει μια παρουσίαση με το user interface για τον έλεγχο του robot. Το PDA αρχικά πραγματοποιεί τη σύνδεση με την υπηρεσία robot σαν μια R-OSGi υπηρεσία (εικόνα α). Στη συνέχεια ανακτά την παρουσίαση με το robot controller interface που τώρα τρέχει τοπικά και επιτρέπει στο PDA να είναι ο διαχειριστής του robot controller.

Το user interface για μια συσκευή έρχεται απευθείας από τη συσκευή και έτσι ο χρήστης μπορεί να συνδεθεί σε άγνωστες από το παρελθόν συσκευές χωρίς την ανάγκη κάποιου configuration ή την εγκατάσταση οδηγών.

Σχήμα 34 α. Σύνδεση στην υπηρεσία

Σχήμα 34 β. Απεικόνιση μιας υπηρεσίας robot

4. Συμπεράσματα και σύγκριση των τριών τεχνολογιών

4.1 Σύγκριση της αρχιτεκτονικής δομής

Η αρχιτεκτονική των τριών τεχνολογιών εξυπηρετεί και στις τρεις περιπτώσεις μια υπηρεσιοκεντρική φιλοσοφία. Αυτό έχει σαν αποτέλεσμα να παρουσιάζονται κοινά σημεία τα οποία ωστόσο μπορεί να διαφέρουν από πλευράς υλοποίησης.

4.1.1 Βασικές οντότητες στην αρχιτεκτονική των τριών τεχνολογιών

Στο UPnP βασικές οντότητες του δικτύου αποτελούν τα σημεία ελέγχου και οι συσκευές όπου επικοινωνούν μεταξύ τους για να χρησιμοποιηθούν οι διαθέσιμες συσκευές και κατ επέκταση οι υπηρεσίες τους. Ουσιαστικά η διαχείριση των συσκευών γίνεται μέσω των σημείων ελέγχων και καμία από τις δύο κατηγορίες δεν διαδραματίζει το ρόλο του εξυπηρετητή ή του πελάτη στο κλασσικό πλέον μοντέλο client – server. Αντίθετα, οι δύο συσκευές είναι εντελώς ομότιμες, ενώ με κατάλληλες τεχνικές είναι πολύ εύκολο να γίνει αντιστροφή των ρόλων τους.

Στα Web Services η αντίστοιχη έννοια συμπληρώνεται από τις οντότητες Service Requestor, Service Provider και Service Registry όπου και αποτελούν τη βάση της δομής της αρχιτεκτονικής τους. Service Requestor εδώ είναι ο αιτούμενος πελάτης που εκκινεί όλη τη διαδικασία με την αναζήτηση της κατάλληλης περιγραφής μιας υπηρεσίας από τις καταχωρήσεις της υπηρεσίας καταλόγου και εν συνεχεία αφού εντοπίσει τον επιθυμητό πάροχο υπηρεσίας εγκαθιδρύει μια σύνδεση μαζί του. Service Provider είναι η οντότητα που λαμβάνει τα μηνύματα κλήσης των αιτούμενων και παρέχει την υπηρεσία στο διαδίκτυο δημοσιοποιώντας την περιγραφή της στην οντότητα του καταλόγου υπηρεσιών. Τέλος Service Registry είναι η οντότητα που περιέχει καταχωρήσεις των περιγραφών των δημοσιευμένων υπηρεσιών διαδικτύου και τρόποι αναζήτησης της κάθε υπηρεσίας.

Βασικές οντότητες της αρχιτεκτονικής OSGi είναι τα bundles καθώς οι εφαρμογές OSGi όπου είναι γραμμένες σε Java είναι ουσιαστικά μια συλλογή από στοιχεία τα bundles τα οποία είναι ανεξάρτητα μεταξύ τους και μπορούν να συνεργάζονται ενώ το OSGi έχει τη δυνατότητα να τα διαχειρίζεται δυναμικά μέσω APIs. Το κάθε bundle παρέχει κάποιες υπηρεσίες και χρησιμοποιεί υπηρεσίες που παρέχονται από άλλα bundles, κάτι το οποίο συνεπάγεται τη μείωση του μεγέθους των εφαρμογών λόγω του ότι μοιράζονται μεγάλο μέρος κώδικα. Έτσι κάθε πλαίσιο που υλοποιεί το πρότυπο OSGi, παρέχει ένα περιβάλλον για την δομοστοιχειοποίηση των

εφαρμογών η οποία βρίσκεται ουσιαστικά στον πυρήνα των OSGi προδιαγραφών και ενσωματώνεται στην έννοια των bundles.

4.1.2 Μοντέλα υλοποίησης της αρχιτεκτονικής, πρότυπα και πρωτόκολλα που χρησιμοποιούνται

Για το UPnP η δημοσίευση των υπηρεσιών των συσκευών και αντίστοιχα η ανίχνευση συσκευών ενδιαφέροντος από τα σημεία ελέγχου επιτυγχάνεται μέσω του πρωτοκόλλου Simple Service Discovery Protocol (SSDP) στο οποίο η μεταφορά των μηνυμάτων γίνεται μέσω UDP. Η περιγραφή της υπηρεσίας των συσκευών εκφράζεται σε XML και το σημείο ελέγχου χρησιμοποιώντας το URL που περιέχεται στο discovery μήνυμα εκπέμπει ένα HTTP GET αίτημα προς τη συσκευή και αυτή επιστρέφει το αντίστοιχο κείμενο περιγραφής. Τα μηνύματα ελέγχου με τις ενέργειες που στέλνουν τα σημεία ελέγχου αφότου έχουν ανακτήσει την περιγραφή της συσκευής εκφράζονται επίσης σε XML με χρήση του πρωτοκόλλου SOAP ενώ για τη μεταφορά τους μεσολαβούν τα πρωτόκολλα HTTP και TCP. Οι δημοσιεύσεις των ενημερώσεων που αποστέλλονται κατά τη φάση eventing ως event μηνύματα πραγματοποιούνται με την αρχιτεκτονική Event Notification Architecture (GENA). Τα μηνύματα αυτά όπου για τη μεταφορά τους μεσολαβούν τα πρωτόκολλα HTTP καθώς επίσης TCP για το Unicast eventing και UDP για το multicast eventing αντίστοιχα εκφράζονται επίσης σε XML.

Στα Web Services αντίστοιχα το βασικό μοντέλο λειτουργιών δημοσίευσης-εύρεσης-σύνδεσης επιτυγχάνεται με την ανταλλαγή μηνυμάτων SOAP. Έτσι η επικοινωνία μεταξύ των εφαρμογών γίνεται με το SOAP μέσω RPC και με χρήση πρωτοκόλλων διαδικτύου όπως το HTTP κυρίως. Το SOAP ως πρότυπο επικοινωνίας όπου καθορίζει τη φόρμα και κανόνες που θα πρέπει να ακολουθούν τα μηνύματα. Είναι βασισμένο σε XML και επιτρέπει την επικοινωνία με διαφορετικές τεχνολογίες και γλώσσες προγραμματισμού. Σημαντικό είναι επίσης να αναφερθεί πως μεταφορά πληροφορίας σε ένα δίκτυο δεν υφίσταται μόνο κατά την αναζήτηση ή την καταχώρηση της πληροφορίας που αφορά την υπηρεσία αλλά και κατά την διάρκεια εκτέλεσης και λειτουργίας της υπηρεσίας. Έτσι κατά την λειτουργία της υπηρεσίας μπορούν να χρησιμοποιηθούν επίσης πρωτόκολλα όπως το SMTP για υπηρεσίες όπου μεσολαβεί η αποστολή email, το FTP για υπηρεσίες όπου γίνεται ανταλλαγή αρχείων είτε και πρωτόκολλα ασφαλείας όπως είναι το HTTPS. Η γνωστοποίηση για το που βρίσκεται η υπηρεσία, τι παρέχει και πως μπορεί να καλεστεί στα Web Services επιτυγχάνεται μέσω της γλώσσας περιγραφής WSDL η οποία παρέχει πληροφορία για το interface της υπηρεσίας δίνοντας ουσιαστικά μια περιγραφή για όλες τις δημόσιες συναρτήσεις της διαθέσιμης υπηρεσίας. Η καταχώρηση και η τροποποίηση των πληροφοριών που αφορούν τις υπηρεσίες γίνεται στον κατάλογο υπηρεσιών UDDI και επιτυγχάνεται μέσω ενός μηνύματος

SOAP όπου αποστέλλεται από τον Service Provider προς τον κατάλογο UDDI. Μια επιτυχής καταχώρηση επιβεβαιώνεται με την αποστολή ενός μηνύματος SOAP από πλευράς του καταλόγου UDDI. Για την ανάκτηση της υπηρεσίας από τον δυνητικό χρήστη ο κατάλογος UDDI Registry θα παρέχει ένα URL το οποίο θα δείχνει στο WSDL αρχείο όπου υφίσταται για την περιγραφή των web services. Η αναζήτηση των υπηρεσιών στον κατάλογο UDDI γίνεται με την αποστολή μηνύματος SOAP από τον Service Requestor ενώ ο κατάλογος UDDI απαντά επίσης με ένα μήνυμα SOAP. Κατά τη διαδικασία της αναζήτησης και καταχώρησης το πρωτόκολλο το οποίο χρησιμοποιείται στο επίπεδο μεταφοράς είναι το http ενώ παρομοίως και κατά την εγκαθίδρυση μιας σύνδεσης μεταξύ service requestor και service provider η επικοινωνία γίνεται με την ανταλλαγή μηνυμάτων SOAP.

Στο R-OSGi αντίστοιχα το οποίο όπως αναφέρθηκε αποτελείται από ένα σύνολο στοιχείων που ονομάζονται bundles, όλες οι εφαρμογές είναι γραμμένες σε Java κάτι που συνεπάγεται πως είναι συμβατές σε οποιοδήποτε περιβάλλον υποστηρίζει την προγραμματιστική γλώσσα Java. Το μοντέλο δημοσίευσης-εύρεσης-σύνδεσης κι εδώ υλοποιείται με παρόμοιο τρόπο με τις υπηρεσίες να δημοσιεύονται μέσω διεπαφών στο αντίστοιχο μητρώο υπηρεσιών από όπου τα υπόλοιπα bundles μπορούν να τις εντοπίζουν, να πληροφορούνται για την προσθήκη τους ή την κατάργησή τους και να ενημερώνονται αν πληρούν τις απαιτήσεις τους ώστε να συνδεθούν και να τις επικαλεστούν. Επίσης, και στο R-OSGi υφίσταται η αποστολή events κάθε φορά που αλλάζει η κατάσταση μιας υπηρεσίας έτσι ώστε να προβλέπεται η ενημέρωση των ενδιαφερόμενων σχετικά με την τρέχουσα κατάσταση και τη διαθεσιμότητα των υπηρεσιών. Επιπρόσθετα εδώ χρησιμοποιείται το πρότυπο whiteboard όπου μέσω του μητρώου OSGi οι συνδρομητές είναι εγγεγραμμένοι υπό μια ειδική διεπαφή ακροατή και έτσι οι πηγές γεγονότων μπορούν να ανακτήσουν όλους τους εγγεγραμμένους ακροατές οποιαδήποτε στιγμή εμφανιστεί ένα γεγονός, κάτι το οποίο είναι πιο αποτελεσματικό από το παραδοσιακό πρότυπο puplish-subscribe. Για την υλοποίηση του κατανεμημένου μητρώου καταχώρησης υπηρεσίας ως βασικός μηχανισμός από το R-OSGi χρησιμοποιείται το Service Location Protocol (SLP). Στο SLP υπάρχει ένας αφοσιωμένος Directory Agent και οι clients επικοινωνούν αποκλειστικά με αυτόν τον κεντρικό server καταχώρησης. Αν δεν υπάρχει Directory Agent οι πελάτες χρησιμοποιούν multicast. Τέλος, η δομή επικοινωνίας στο R-OSGi όπως και στις υπόλοιπες τεχνολογίες βασίζεται σε μηνύματα. Τα κανάλια του δικτύου είναι TCP συνδέσεις όπου χρησιμοποιούν τη δυνατότητα keep-alive και αποτελούν μια σύνδεση ένας προς έναν μεταξύ δύο peers ενώ ο επεκτάσιμος σχεδιασμός του R-OSGi δίνει επιπρόσθετα την δυνατότητα σύνδεσης και με εναλλακτικά πρωτόκολλα και δίκτυα μεταφοράς.

4.2 Πλεονεκτήματα των τριών τεχνολογιών

Παρακάτω αναφέρονται τα πλεονεκτήματα που προκύπτουν από την χρήση της κάθε τεχνολογίας.

4.2.1 Πλεονεκτήματα UPnP

1.Αυτόματη/δυναμική αναγνώριση για μεγάλο εύρος συσκευών. Μια συσκευή μπορεί να συνδεθεί δυναμικά στο δίκτυο να αποκτήσει IP να ανακοινώσει δυνατότητες και να πληροφορηθεί για δυνατότητες άλλων. Το ίδιο ισχύει και για την αποσύνδεση από το δίκτυο όπου γίνεται επίσης δυναμικά χωρίς να δημιουργούνται προβλήματα.

2.Το UPnP παρέχει ανεξαρτησία μέσου και συσκευής καθώς μπορεί να τρέξει σε οποιαδήποτε δικτυακή τεχνολογία μετάδοσης όπως για παράδειγμα WiFi, ομοαξονικά καλώδια, τηλεφωνικές γραμμές, γραμμές ενέργειας και Ethernet. Επίσης με τη χρήση πρωτοκόλλων διαδικτύου και με την εφαρμογή της τεχνολογίας σε διαφορετικά φυσικά μέσα (ενσύρματα και ασύρματα), δίνεται η δυνατότητα διαλειτουργικότητας μεταξύ διαφόρων τύπων συσκευών προερχόμενες από διαφορετικούς προμηθευτές καθώς και η επίτευξη συνεργασίας με το διαδίκτυο και το intranet κάποιου ιδιωτικού χώρου. Είναι μάλιστα σχεδιασμένο για περιβάλλοντα με αυτές τις απαιτήσεις. Επιπροσθέτως η εφαρμογή UPnP συσκευών είναι ανεξάρτητη από τη γλώσσα προγραμματισμού και από το λειτουργικό σύστημα που χρησιμοποιείται. Οι κατασκευαστές UPnP συσκευών μπορούν να χρησιμοποιήσουν οποιαδήποτε γλώσσα προγραμματισμού και οποιοδήποτε λειτουργικό σύστημα προκειμένου να σχεδιάσουν την εφαρμογή τους. Αυτό είναι ιδιαίτερα σημαντικό καθώς σε ένα δικτυωμένο περιβάλλον είναι αναμενόμενη η ύπαρξη διαφορετικών λειτουργικών συστημάτων αλλά και ξεχωριστών αναγκών από συγκεκριμένα συστήματα.

3.Δυνατότητα ελέγχου μέσω διεπαφών χρήστη. Η αρχιτεκτονική UPnP επιτρέπει στους χρήστες να ελέγχουν τις συσκευές που επιθυμούν, απομακρυσμένα και εύκολα μέσω απλής χρήσης ενός οποιουδήποτε internet browser.

4.Επεκτασιμότητα των υπηρεσιών. Κάθε UPnP enabled συσκευή μπορεί να εμπλουτιστεί με υπηρεσίες προστιθέμενης αξίας, που τοποθετούνται ιεραρχικά πάνω στα ήδη υπάρχοντα πρότυπα, και προσφέρουν επιπλέον δυνατότητες διαχείρισης, χωρίς να υπάρχει ανάγκη πλήρους επαναπροσδιορισμού του ήδη υπάρχοντος κώδικα.

5.Άλλο ένα σημαντικό χαρακτηριστικό του πρωτοκόλλου UPnP είναι το γεγονός ότι βασίζεται σε τεχνολογίες ομότιμων δικτύων (peer to peer). Η διαχείριση των

συσκευών γίνεται μέσω σημείων ελέγχου (control points). Καμία από τις συσκευές παρόλα αυτά δεν διαδραματίζει το ρόλο του εξυπηρετητή ή του πελάτη στο κλασικό πλέον μοντέλο client – server. Αντίθετα, οι δύο συσκευές είναι εντελώς ομότιμες, ενώ είναι πολύ εύκολη με κατάλληλες τεχνικές η αντιστροφή των ρόλων τους. Κάτι το οποίο είναι σημαντικό σε αρχιτεκτονικές όπου η πιθανή δυσλειτουργία ενός σημείου ελέγχου μπορεί να σημάνει την απώλεια ελέγχου μιας ολόκληρης περιοχής χρηστών

4.2.2 Πλεονεκτήματα Web Services

1.Διαλειτουργικότητα υπηρεσιών διαφορετικής πλατφόρμας και λειτουργικού συστήματος. Λόγω της χρήσης XML είναι δυνατή η αλληλεπίδραση μεταξύ υπηρεσιών σε οποιαδήποτε πλατφόρμα, γραμμένες σε οποιαδήποτε προγραμματιστική γλώσσα. Με την XML ως το μόνο πρότυπο στα Web Services, συστήματα φτιαγμένα από διαφορετικές τεχνολογίες όπως Java και .NET μπορούν να επικοινωνήσουν μεταξύ τους.

2.Δυνατότητα ανάκτησης δεδομένων ακόμα κι αν προστίθενται η αφαιρούνται δυναμικά μέσω parsing.

3.Λόγω χρήσης του πρωτοκόλλου SOAP παρέχεται η δυνατότητα διασύνδεσης των εφαρμογών με παγκόσμια κατανεμημένους εξυπηρετητές και διασύνδεση διαφορετικών ιστοτόπων μέσω της χρήσης προτύπων του internet HTTP και XML.

4.Δυνατότητα επεκτασιμότητας και ενοποίησης σε υπάρχουσες εφαρμογές. Τα Web Services μπορούν να προσαρμόσουν ήδη υπάρχουσες εφαρμογές σε μεταβαλλόμενες επιχειρησιακές συνθήκες και ανάγκες πελατών με την εισαγωγή άλλων λειτουργιών διαχείρισης διαδικασιών όπως η αξιοπιστία και η ασφάλεια, ανεξάρτητα της αρχικής λειτουργίας μιας εφαρμογής, αυξάνοντας τη μεταβλητότητα και τη χρησιμότητά της στο επιχειρησιακό περιβάλλον. Είναι δυνατή η ενοποίηση με υπάρχοντα συστήματα και η επαναχρησιμοποίηση της υπάρχουσας τεχνολογίας κάτι το οποίο εξυπηρετεί σε μεγάλο βαθμό εταιρικές ανάγκες αφού πολλές επιχειρήσεις έχουν ήδη αποθηκευμένο μεγάλο όγκο πληροφορίας σε κάποιο πληροφοριακό σύστημα όπου το κόστος αντικατάστασής του είναι απαγορευτικό. Επίσης ένα έτοιμο Web Service είναι πολύ εύκολο να επεκταθεί παρέχοντας επιπρόσθετες λειτουργίες στον χρήστη.

5.Το κόστος ενσωμάτωσης ενός web service σε κάποιο web site ή διαδικτυακή εφαρμογή είναι μικρό. Τα web services λειτουργούν με πρότυπες γλώσσες και πρωτόκολλα όπως η XML, το HTTP και το TCP/IP. Η πλειονότητα των εταιριών έχουν ήδη την δικτυακή υποδομή και τους ανθρώπους με γνώσεις και εμπειρία που τη

συντηρούν. Έτσι το κόστος για την εφαρμογή των web services είναι σημαντικά μικρό.

6.Η τεχνολογία Web Services χρησιμοποιεί ως πρωτόκολλο επικοινωνίας το SOAP και πρωτόκολλα όπως το HTTP και το SMTP ως μέσα μεταφοράς οπότε μπορεί να χρησιμοποιηθεί στο διαδίκτυο και να διαπερνά τύχη προστασίας χωρίς συμβιβασμούς στην ασφάλεια της υποδομής μιας επιχείρησης. Αυτό αυτομάτως μειώνει και σε ορισμένες περιπτώσεις εξαλείφει το κόστος υποδομής αφού οι περισσότερες επιχειρήσεις σήμερα έχουν και τον εξοπλισμό και την τεχνογνωσία για τη χρήση του διαδικτύου. Τα Web Services μπορούν να χρησιμοποιήσουν μεταξύ άλλων το HTTP ως πρωτόκολλο μεταφοράς και τα περισσότερα τείχη προστασίας επιτρέπουν την πρόσβαση μέσω της θύρας 80 (πρότυπη θύρα για το HTTP). Με αυτόν τον τρόπο οδηγούμαστε σε ευκολότερες και δυναμικές συνεργασίες μεταξύ των συστημάτων των διαφόρων οργανισμών.

4.2.3 Πλεονεκτήματα R-OSGi

1.Ένα από τα σημαντικά πλεονεκτήματα του OSGi και κατ επέκταση του R-OSGi είναι η μεγάλη ευελιξία στην δυναμική προσθήκη και αναβάθμιση υπηρεσιών καθώς ένα από τα χαρακτηριστικά είναι η δυνατότητα που παρέχει για δυναμική διαχείριση των bundles. Έτσι νέα bundles μπορούν να προστεθούν και ήδη υπάρχοντα bundles μπορούν να αναβαθμιστούν ή να αφαιρεθούν καθ όλη τη διάρκεια λειτουργίας του συστήματος χωρίς να είναι αναγκαία η επανεκκίνηση της πλατφόρμας και χωρίς να επηρεάζεται όλο το σύστημα, κάτι που καθιστά το OSGi κατάλληλο για περιβάλλοντα υψηλής αξιοπιστίας. Το χαρακτηριστικό αυτό είναι επίσης ιδιαίτερα λειτουργικό για συσκευές όπου είναι αναγκαίο να αναβαθμίζονται αυτομάτως ενώ η λειτουργία τους στο σύστημα συνεχίζεται αδιάκοπα.

2.Δυναμικές ενημερώσεις. Η οποιαδήποτε αλλαγή συμβαίνει σε ένα bundle συμβαίνει δυναμικά και τα bundles τις παρακολουθούν και ανάλογα με αυτές αντιδρούν. Έτσι για παράδειγμα, σε περίπτωση που κάποιο bundle επέλθει σε κατάσταση installed τα υπόλοιπα bundles έχουν τη δυνατότητα να δουν αν ενδιαφέρονται για τις υπηρεσίες του. Το δυναμικό αυτό μοντέλο υπηρεσιών επιτρέπει στα bundles να ενημερώνονται για το ποιες δυνατότητες είναι διαθέσιμες στο σύστημα και να προσαρμόζουν την λειτουργικότητα που μπορούν να παρέχουν. Αυτό καθιστά τον κώδικα πιο ευέλικτο και ελαστικό σε αλλαγές.

3.Ως αναφορά το ζήτημα της διαλειτουργικότητας η πλατφόρμα R-OSGi κι όλες οι εφαρμογές που τρέχουν πάνω από αυτή γράφονται στη γλώσσα java. Αυτό έχει το επιθυμητό αποτέλεσμα ότι οι ίδιες οι εφαρμογές μπορούν να τρέξουν χωρίς αλλαγές σε οποιοδήποτε περιβάλλον υποστηρίζει τη γλώσσα Java. Ένας από τους

βασικούς λόγους που το OSGi είναι τόσο επιτυχές είναι ότι προσφέρει ένα αρκετά ώριμο σύστημα στοιχείων που λειτουργεί πραγματικά σε έναν εκπληκτικό αριθμό από περιβάλλοντα. Παρόλο που το R-OSGi είναι ένα εξελιγμένο middleware για OSGi frameworks, χρησιμοποιεί ένα πολύ αποδοτικό δικτυακό πρωτόκολλο. Επίσης αποτελεί ένα ισχυρό μοντέλο για τη συνύπαρξη διαφορετικών στοιχείων/εφαρμογών στο ίδιο JVM και ως εκ τούτου μπορούν να μοιράζονται κώδικα, γεγονός που μειώνει την απαιτούμενη μνήμη. Έτσι καθίσταται ιδανικό για μικρές και ενσωματωμένες συσκευές με περιορισμένη μνήμη και εύρος ενώ παράλληλα αποτελεί ένα σύστημα στοιχείων που χρησιμοποιείται για να χτίσει υψηλής πολυπλοκότητας εφαρμογές όπως application servers, βιομηχανικούς αυτοματισμούς κλπ.

4.Δυνατότητα διαφανούς διάγνωσης και επιδιόρθωσης. Μέσω API διαχείρισης παρέχεται πρόσβαση στην εσωτερική κατάσταση του bundle καθώς επίσης και στο πως είναι συνδεδεμένο με άλλα bundles. Για παράδειγμα τα περισσότερα πλαίσια παρέχουν ένα κελί εντολών το οποίο δείχνει αυτή την εσωτερική κατάσταση. Μέρη των εφαρμογών μπορεί να σταματήσουν με τον εντοπισμό ενός πιθανού σφάλματος, ή επίσης μπορεί υπάρξουν διαγνωστικά bundles. Αντί των πολυάριθμων γραμμών απεγγραφών ή επανεκκινήσεων, οι εφαρμογές R-OSGi μπορούν συχνά να διορθωθούν με ένα υπάρχον κελί εντολών σε πραγματικό χρόνο.

5.Οι υπηρεσίες στο R-OSGi επιτρέπουν στα bundles να έρχονται σε συνεργασία, χωρίς ωστόσο να εξαρτώνται από κάποιο πακέτο είτε κάποια υλοποίηση ενός άλλου bundle παρά μόνο από την διεπαφή της υπηρεσίας. Επίσης το bundle που θα χρησιμοποιήσει την υπηρεσία δεν γνωρίζει το bundle το οποίο παράγει την υπηρεσία αυτή έτσι το σύστημα ενώ αποτελεί ένα σύστημα συνεργασίας παραμένει χαλαρά συνδεδεμένο. Βάσει αυτού το OSGi σύστημα αποτελεί ουσιαστικά μια επέκταση της Java στο οποίο ωστόσο οι εξαρτήσεις και η ορατότητα των πακέτων είναι δυναμικά κάτι που καθιστά το σύστημα πιο ευέλικτο και βελτιστοποιεί την απόδοση των εφαρμογών.

6.Είναι αρκετά γρήγορο καθώς προδεσμεύει bundles και γνωρίζει για κάθε bundle ακριβώς ποιο παρέχει την κλάση που πρέπει να κληθεί. Στην παραδοσιακή Java τα Jars είναι πλήρως ορατά και τοποθετούνται σε μια λίστα. Η αναζήτηση μια κλάσης απαιτεί αναζήτηση μεταξύ αυτής συχνά σε ένα πολύ μεγάλο αριθμό, (150 δεν είναι ασυνήθιστο) της λίστας. Αυτή η έλλειψη αναζήτησης που παρέχει το R-OSGi είναι ένας σημαντικός παράγοντας επιτάχυνσης κατά την εκκίνηση.

7.Επιλεκτική αδράνεια. Η επιλεκτική αδράνεια στο λογισμικό αποτελεί προτέρημα και η τεχνολογία R-OSGi έχει πολλούς μηχανισμούς σε θέση να κάνουν πράγματα μόνο όταν είναι πραγματικά απαραίτητο. Για παράδειγμα, τα bundles μπορούν να ξεκινήσουν πρόθυμα αλλά μπορούν επίσης να διαμορφωθούν ώστε να ξεκινούν μόνο όταν ένα άλλο bundle τα χρησιμοποιεί. Οι υπηρεσίες μπορούν να εγγραφούν

αλλά να δημιουργούνται μόνο όταν χρειάζονται. Οι προδιαγραφές έχουν βελτιστοποιηθεί για να επιτρέπουν αυτό το είδος σεναρίου αδράνειας που μπορεί να εξοικονομήσει τεράστια κατανάλωση εκτέλεσης.

8.Χαμηλό κόστος ανάπτυξης. Οι εφαρμογές στην R-OSGi πλατφόρμα καθίστανται πιο εύκολες στην ανάπτυξη λόγω της αρχιτεκτονικής η οποία απαρτίζεται από bundles. Επίσης τα bundles που αναπτύσσονται για ένα συγκεκριμένο προϊόν είναι πιθανό να είναι κατάλληλα και για υπόλοιπα παρόμοια προϊόντα.

9.Ενοποίηση μεταξύ διαφορετικών περιβαλλόντων. Η αυξανόμενη δημοτικότητα των προδιαγραφών του R-OSGi δίνει την δυνατότητα για την ενοποίηση εντελώς διαφορετικών περιβαλλόντων. Έτσι δίνεται η προοπτική σε εφαρμογές γραμμένες για οικιακή πύλη να μπορούν να τρέξουν για παράδειγμα σε ένα PDA. Αυτό είναι κάτι που δίνει μεγάλες προοπτικές έκτασης της αγοράς του R-OSGi.

4.3 Ασφάλεια

4.3.1 Ασφάλεια στο UPnP

Το πρωτόκολλο ασφάλειας του UPnP παρέχει αναγκαίες υπηρεσίες ασφάλειας σε μηνύματα SOAP όπου περιλαμβάνουν, την δυνατή ταυτοποίηση, ακεραιότητα, εμπιστευτικότητα και εξουσιοδότηση του μηνύματος. Έτσι βοηθά στην προστασία SOAP επικοινωνίας μεταξύ της ελεγχόμενης συσκευής και του σημείου ελέγχου από επιθέσεις. Σε αυτή την αρχιτεκτονική ασφάλειας μια εφαρμογή διαχείρισης η Security Console (SC), διατηρεί την αλληλεπίδραση μεταξύ χρήστη και συσκευής καθώς επίσης εγκαθιδρύει και διαχειρίζεται την πολιτική ελέγχου πρόσβασης για τις συσκευές. Μια ασφαλής συσκευή UPnP θα πρέπει να υλοποιεί κάποιους αλγόριθμους κρυπτογράφησης για την εγκαθίδρυση ασφαλών συνεδριών.

Η διαδικασία ανίχνευσης συσκευής περιλαμβάνει δύο φάσεις. Στην πρώτη φάση όταν μια ασφαλής συσκευή εισέλθει στο δίκτυο μεταδίδει την παρουσία της με χρήση SSDP περιγράφοντας τον εαυτό της και τις δυνατότητες ασφάλειας, ενώ για τον έλεγχο της χρειάζεται η παροχή αντίστοιχων κωδικών από την SC.

Στη δεύτερη φάση η SC πρέπει να εκτελέσει διάφορες εργασίες όπως η ανάκτηση ενός συνόλου κλειδιών συνεδρίας, τον έλεγχο του τύπου χορήγησης αδείας και τον καθορισμό των ιδιοτήτων χορήγησης. Τη στιγμή αυτή η SC έχει τον έλεγχο της συσκευής και επιπλέον μπορεί να επεξεργαστεί την Access Control List (ACL) της συσκευής για την παραχώρηση της κυριότητας σε άλλους πελάτες. Έτσι το CP (Control Point) μπορεί να εγκαθιδρύσει μια ασφαλή σύνδεση με την συσκευή

έπειτα από την επιβεβαίωση διαλειτουργικών αλγόριθμων ασφαλείας και την ανάκτηση κλειδιών συνεδρίας. Έπειτα από την εγκαθίδρυση συνεδρίας με την ελεγχόμενη συσκευή το CP μπορεί να στείλει μηνύματα ενεργειών. Είναι επίσης δυνατή η επίτευξη εμπιστευτικότητας για μηνύματα ενεργειών με την κρυπτογράφηση των μηνυμάτων. Όταν το CP επιθυμεί να τερματίσει μια συνεδρία καλεί τα ExpireSessionKeys τα οποία πληροφορούν την ελεγχόμενη συσκευή πως το CP τερματίζει την συνεδρία. Η ελεγχόμενη συσκευή τότε αφαιρεί την συσχετιζόμενη με το κλειδί πληροφορία και αποδεσμεύει κάθε πόρο που σχετίζεται με το CP αυτό.

Ωστόσο, στο UPnP έχουν ανακαλυφθεί τρωτά σημεία στην ασφάλεια όπου σε πολλές περιπτώσεις μάλιστα καθίσταται δυνατό να ελεγχθεί εξ ολοκλήρου το λειτουργικό ενός CP από τον εισβολέα. Για παράδειγμα ο εισβολέας μπορεί να εκμεταλλευτεί το ότι μόλις η συσκευή συνδεθεί στο δίκτυο στέλνει NOTIFY μηνύματα σε όλα τα σημεία ελέγχου. Έτσι αν αποστείλει κατ επανάληψη με φυσιολογικά διαστήματα κάποιο κακόβουλο μήνυμα NOTIFY, τότε αρχίζει η παραβίαση της υπηρεσίας SSDP του CP και σταδιακά μετατρέπεται σε υπερχείλιση του buffer. Επιπρόσθετα, ως γνωστόν το CP χρησιμοποιεί HTTP GET μέθοδο για την ανάκτηση της περιγραφής της συσκευής, έτσι αν ο εισβολέας θέσει ένα μεγάλο αρχείο σαν περιγραφή συσκευής τότε αυτό επίσης μπορεί να οδηγήσει σε buffer overflow. Ο εισβολέας επιπλέον, μπορεί να καταναλώσει την ενέργεια του CPU του CP και την μνήμη και αυτό τερματίζεται μόνο έπειτα από reboot του CP.

Δεδομένου ότι οι UPnP συσκευές δεν δεσμεύονται να υλοποιούν κάποιο απαιτούμενο πρότυπο ασφάλειας, είναι γεγονός πως το δίκτυο που είναι βασισμένο στο UPnP είναι εγγενώς μη ασφαλές. Ως εκ τούτου είναι δυνατό να γίνει επίθεση σε ένα δίκτυο όπου υπάρχουν UPnP συσκευές ακόμη και με περιορισμένα δικαιώματα στο δίκτυο. Επίσης σχετικά με το SSDP το οποίο είναι υπεύθυνο για την ανακάλυψη των συσκευών, δεν γίνεται να περιοριστεί η ανακοίνωση και η ανταπόκριση των συσκευών μόνο σε έμπιστα CP δεδομένου ότι πολλές συσκευές πιθανό να μην πληρούν πρότυπα ασφαλείας. Συνεπώς είναι ζήτημα των προμηθευτών να αναπτύξουν και να παρέχουν UPnP συσκευές με ενσωματωμένους μηχανισμούς ασφάλειας πριν τις καταστήσουν ευρέως διαδεδομένες προς χρήση.

Συμπερασματικά η αρχιτεκτονική UPnP είναι κατάλληλη για οικιακά δίκτυα αλλά όχι τόσο η ανάπτυξή της σε ομάδες δικτύων. Τα μηνύματα εκπομπής για λόγους ανίχνευσης και ενημέρωσης , η χρήση του αναξιόπιστου UDP πρωτοκόλλου αλλά και η έλλειψη μηχανισμών ασφαλείας είναι ορισμένοι από τους βασικότερους λόγους που καθιστούν το UPnP ακατάλληλο να υποστηρίξει τη διαχείριση συσκευών και υπηρεσιών από το δημόσιο δίκτυο.

4.3.2 Ασφάλεια στα Web Services

Στα Web Services η ασφάλεια αποτελεί ένα τομέα συνεχούς και ταχείας ανάπτυξης. Οι παραδοσιακές μέθοδοι που αφορούν στην εγκαθίδρυση και την εμπιστευτικότητα μεταξύ των συμβαλλόμενων μερών δεν είναι οι πλέον κατάλληλες για το διαδίκτυο είτε για μεγάλα τοπικά δίκτυα LAN ή WAN. Για την διασφάλιση της ακεραιότητας των δεδομένων και τον περιορισμό της προσπέλασης στις υπηρεσίες ιστού μόνο σε εξουσιοδοτημένους χρήστες χρησιμοποιούνται κάποια πρότυπα και προδιαγραφές όπως τα παρακάτω.

α.XML Ψηφιακές Υπογραφές. Οι XML ψηφιακές υπογραφές (XML Digital Signatures) είναι ένα πρότυπο για την ασφαλή επικύρωση της προέλευσης των μηνυμάτων. Η προδιαγραφή της XML υπογραφής επιτρέπει στα έγγραφα XML να υπογραφούν με ένα τυποποιημένο τρόπο, χρησιμοποιώντας διαφορετικούς αλγόριθμους ψηφιακής υπογραφής.

β. XML Κρυπτογράφηση. Βασικοί της στόχοι είναι η υποστήριξη κρυπτογράφησης των ψηφιακών περιεχομένων, η διασφάλιση πως η προσπέλαση των κρυπτογραφημένων δεδομένων περιοορίζεται μόνο σε εξουσιοδοτημένους χρήστες καθώς επίσης η δυνατότητα παρουσίασής τους σε μορφή XML.

γ. Το XKMS(XML Key Management Specification). Το οποίο παρέχει ασφαλέστερη επικοινωνίας μεταξύ των εφαρμογών με την χρήση της υποδομής δημοσίων κλειδιών. Είναι ένα πρωτόκολλο το οποίο περιγράφει την κατανομή και την καταχώρηση δημοσίων κλειδιών.

δ. Το πρότυπο SAML (Security Assertion Markup Language). Είναι ένα ανοιχτό πρότυπο βασισμένο στην XML για την ανταλλαγή δεδομένων πιστοποίησης και εξουσιοδότησης μεταξύ μερών σχετικά με το πότε μια οντότητα χρήστη επιτρέπεται να έχει πρόσβαση σε κάποιους πόρους. Ορίζει ένα πρωτόκολλο με το οποίο οι πελάτες μπορούν να αιτηθούν δηλώσεις από τις SAML αρχές και να πάρουν μια απάντηση από αυτές. Το πρωτόκολλο αυτό αποτελείται από μορφές αίτησης και ανταπόκρισης όπου είναι βασισμένες σε XML.

ε. Το πρότυπο XACML (eXtensible Access Control Markup Language) το οποίο ορίζει μια γλώσσα πολιτικής ελέγχου πρόσβασης υλοποιημένη σε XML καθώς και ένα μοντέλο επεξεργασίας που περιγράφει τον τρόπο αξιολόγησης των αιτήσεων εξουσιοδότησης σύμφωνα με τους κανόνες της πολιτικής. Ένας από τους βασικούς στόχους των προδιαγραφών του XACML είναι η προώθηση κοινής ορολογίας και διαλειτουργικότητας μεταξύ υλοποιήσεων εξουσιοδότησης από διαφορετικούς παρόχους.

Επίσης στο μοντέλο ασφάλειας των Web Services παρέχονται μια σειρά από προδιαγραφές οι οποίες απευθύνονται στο πως παρέχεται η προστασία στα

ανταλλασσόμενα μηνύματα σε ένα περιβάλλον υπηρεσίας ιστού και καλύπτουν ζητήματα ακεραιότητας, εμπιστευτικότητας, αυθεντικοποίησης, εξουσιοδότησης, ασφαλών διαδρομών επικοινωνίας και πολιτικής ασφάλειας. Οι συνεχώς εξελισσόμενες προδιαγραφές αυτές εκμεταλλεύονται στο έπακρον την επεκτασιμότητα του πρωτοκόλλου SOAP και οι σημαντικότερες είναι:

α.WS-Security όπου καθορίζει τον τρόπο με τον οποίο επισυνάπτονται οι υπογραφές, οι κρυπτογραφημένες κεφαλίδες και άλλες πληροφορίες ασφαλείας στα μηνύματα.

β.WS-Trust όπου καθορίζει ένα μοντέλο έμπιστης σχέσης για ασφαλείς συναλλαγές.

γ.WS-Policy όπου δίνει τη δυνατότητα στα Web Services να διαφημίσουν τις πολιτικές τους για ζητήματα όπως η ασφάλεια και η ποιότητα, και στους χρήστες να καθορίσουν τις απαιτήσεις της πολιτικής τους.

δ.WS-Privacy όπου καθορίζει το πως τηρείται η ιδιωτικότητα σε κάθε πληροφορία.

ε.WS-SecureConversation όπου ορίζει μηχανισμούς σχετικά με την εγκατάσταση ασφαλών περιβαλλόντων και παραγωγής κλειδιών για την επίτευξη μιας ασφαλούς επικοινωνίας.

στ.WS-Federation όπου καθορίζει τους κανόνες σχετικά με την ταυτότητα για ένα κατανεμημένο περιβάλλον.

4.3.3 Ασφάλεια στο R-OSGi

Ένας από τους βασικούς στόχους της αρχιτεκτονικής R-OSGi είναι να τρέχει εφαρμογές που προέρχονται από μια ποικιλία πόρων υπό τον αυστηρό έλεγχο του συστήματος διαχείρισης. Έτσι ένα ολοκληρωμένο μοντέλο ασφάλειας το οποίο θα αφορά σε όλα τα μέρη του συστήματος αποτελεί αναγκαία προϋπόθεση. Αρχικά λοιπόν θα πρέπει να αναφερθεί πως στην R-OSGi πλατφόρμα υπηρεσιών το περιβάλλον εκτέλεσης πληρεί υψηλές απαιτήσεις σε θέματα ασφαλείας λόγω της κάθε αυτού R-OSGi αρχιτεκτονικής αλλά και των ασφαλών χαρακτηριστικών της γλώσσας Java.

Το πρώτο σημείο άμυνας αφορά στην εικονική μηχανή (JVM) η οποία είναι σχεδιασμένη να περιορίζει τις δυνατότητες μιας Java εφαρμογής κατά τη διάρκεια του χρόνου εκτέλεσης. Το δεύτερο σημείο άμυνας αφορά στα ασφαλή χαρακτηριστικά της γλώσσας Java. Η πρόσβαση στην Java πραγματοποιείται με δήλωση των επιτρεπόμενων καλούντων στον κώδικα. Αυτός ο μηχανισμός ελέγχου επιτρέπει σε πολλαπλές Java εφαρμογές να βρίσκονται στην ίδια εικονική μηχανή (JVM) και να συνεργάζονται ενώ παράλληλα παρέχεται μια ασπίδα προστασίας για

τον κώδικα. Το τελευταίο σημείο άμυνας αφορά στον αυστηρό διαχωρισμό που παρέχει το R-OSGi μεταξύ των bundles. Τα bundles χρειάζονται τα κατάλληλα δικαιώματα ώστε να αλληλεπιδράσουν με άλλα bundles της R-OSGi πλατφόρμας υπηρεσιών ενώ παρέχεται η δυνατότητα υπογραφής των bundles έτσι ώστε να μπορεί να αξιολογηθεί η αξιοπιστία τους. Το καλά δομημένο μοντέλο αυτό ασφάλειας βελτιώνει την χρηστικότητα έχοντας ο σχεδιαστής του bundle καθορίσει τις απαιτούμενες λεπτομέρειες ασφαλείας σε μια μορφή εύκολη να ελεγχθεί.

Η ανάθεση της πολιτικής διαχείρισης στον διαχειριστή της πλατφόρμας επιτρέπει στην OSGi πλατφόρμα υπηρεσιών να χρησιμοποιείται σε πολλά σενάρια ασφαλείας καθώς επίσης σε σενάρια όπου η ασφάλεια διαχειρίζεται εκτός της OSGi πλατφόρμας υπηρεσιών. Οι OSGi και Java αρχιτεκτονικές ασφαλείας αποτελούν ένα ολοκληρωμένο μοντέλο το οποίο παρέχει στον διαχειριστή της πλατφόρμας τη δυνατότητα να δημιουργήσει ένα από τα πιο ασφαλή περιβάλλοντα στην αγορά.

4.4 Καταλληλότητα στο πεδίο εφαρμογής

4.4.1 Καταλληλότητα του UPnP στο πεδίο εφαρμογής

Η χρήση του αναξιόπιστου UDP πρωτοκόλλου αλλά και η έλλειψη μηχανισμών ασφαλείας είναι ορισμένοι από τους βασικότερους λόγους όπου το UPnP δεν είναι κατάλληλο για την υποστήριξη διαχείρισης και ελέγχου συσκευών από το δημόσιο δίκτυο.

Ωστόσο με τη χρήση πρωτοκόλλων διαδικτύου και με την εφαρμογή της τεχνολογίας σε διαφορετικά φυσικά μέσα (ενσύρματα και ασύρματα), δίνεται η δυνατότητα διαλειτουργικότητας διαφόρων τύπων συσκευών από διαφορετικούς προμηθευτές σε συνεργασία με το διαδίκτυο είτε το Intranet κάποιου ιδιωτικού δικτύου. Το UPnP έχει σχεδιαστεί κυρίως για να παρέχει εύχρηστη, ευέλικτη και βασισμένη στα πρότυπα διασύνδεση, σε ad-hoc και μη διαχειριζόμενα δίκτυα σε οικιακό περιβάλλον μικρές επιχειρήσεις, εμπορικά κτίρια και τοπικά δίκτυα σε συνεργασία με το διαδίκτυο όπως διευκρινίστηκε παραπάνω.

Με τη χρήση του UPnP μπορούμε για παράδειγμα να συνδέσουμε έναν εκτυπωτή χωρίς να χρειάζεται η παραμικρή παρέμβαση από το χρήστη. Ο εκτυπωτής ανακοινώνει την παρουσία του στο δίκτυο έτσι ώστε ένας αν ένας υπολογιστής επιθυμεί να εκτυπώσει, απλά θα αναζητήσει διαθέσιμους εκτυπωτές και θα προχωρήσει. Ο εκτυπωτής μπορεί επίσης να χρησιμοποιηθεί και από άλλες συσκευές UPnP όπως για παράδειγμα μια ψηφιακή φωτογραφική η οποία μόλις

έχει πάρει εντολή από το χρήστη για εκτύπωση φωτογραφιών και εφόσον έχει συνδεθεί με το δίκτυο, θα ψάξει για εκτυπωτή χωρίς να χρειάζεται τη μεσολάβηση κάποιου υπολογιστή ή εφαρμογής. Επιπλέον, οι συσκευές UPnP που διαθέτουν ρολόι, μπορούν να συντονιστούν αυτόματα μετά από διακοπή ρεύματος, μέσω του υπολογιστή που παίρνει την ώρα από το διαδίκτυο.

4.4.2 Καταλληλότητα των Web Services στο πεδίο εφαρμογής

Τα Web Services είναι σχεδιασμένα για να διασυνδέουν εφαρμογές που είναι παγκόσμια κατανεμημένες δίνοντας τη δυνατότητα μεταφοράς μεγάλου όγκου δεδομένων με αποτέλεσμα την γρηγορότερη και πιο παραγωγική επικοινωνία τόσο στον επιχειρηματικό τομέα όσο και για τους καταναλωτές.

Οι εφαρμογές που βασίζονται στα Web Services μπορούν να χειριστούν απλά αιτήματα για πληροφορία είτε να πραγματοποιούν πολύπλοκες επιχειρηματικές διαδικασίες. Έτσι οι χρήστες μπορούν όχι απλά να χρησιμοποιούν προγραμματιστικά το δίκτυο αλλά και να προσπελάσουν τις υπηρεσίες αυτές και τη λειτουργικότητά τους δίνοντας τη δυνατότητα επέκτασης και επαναχρησιμοποίησης της ήδη υπάρχουσας λειτουργικότητας των συστημάτων με αποτέλεσμα την μείωση του λειτουργικού κόστους ως αναφορά τις επιχειρηματικές απαιτήσεις.

Η χρήση των Web Services είναι ιδανική για την ανάλυση, τον συνδυασμό και την παρουσίαση της πληροφορίας. Για παράδειγμα σε ένα λογιστικό φύλλο το οποίο συνοψίζει μια ολοκληρωμένη οικονομική εικόνα όπως μετοχές, τραπεζικούς λογαριασμούς και δάνεια η πληροφορία αν είναι διαθέσιμη μέσω Web Services το λογιστικό φύλλο θα μπορούσε να ενημερώνεται συνεχώς και η προγραμματιστική πρόσβαση θα μπορούσε να είναι πιο εύκολη και αξιόπιστη.

Επίσης εκθέτοντας ήδη υπάρχουσες εφαρμογές σαν Web Services δίνεται η δυνατότητα στους χρήστες να δημιουργήσουν νέες πιο ισχυρές εφαρμογές οι οποίες θα χρησιμοποιούν τα Web Services σαν δομικά στοιχεία. Έτσι για παράδειγμα ο χρήστης θα μπορούσε να αναπτύξει μια εφαρμογή προμηθειών η οποία παίρνει αυτόματα τιμές από προμηθευτές, επιτρέπει στο χρήστη να επιλέξει προμηθευτή, να υποβάλει την παραγγελία και να παρακολουθεί την αποστολή έως ότου να γίνει η παραλαβή της. Η εφαρμογή του προμηθευτή εκτός από το να εκθέτει τις υπηρεσίες της στον ιστό, θα μπορούσε να χρησιμοποιήσει άλλα Web Services για να ελέγξει την πιστοληπτική ικανότητα του πελάτη, να χρεώσει τον τραπεζικό του λογαριασμό του και να καθορίσει την αποστολή με μια εταιρεία μεταφορών.

Βάση των παραπάνω λοιπόν είναι εμφανές πως τα Web Services μπορούν άρτια να προσφέρουν ένα απλούστερο τρόπο για την επίτευξη κατανεμημένων εφαρμογών,

διευκολύνοντας τα διασυνδεδεμένα συστήματα να ανταλλάξουν πληροφορίες και να συνδιαλλαγούν μεταξύ τους.

4.4.3 Καταλληλότητα του R-OSGi στο πεδίο εφαρμογής

Σκοπός του R-OSGi είναι η δημιουργία ανοιχτών προδιαγραφών για την παράδοση διαχειριζόμενων υπηρεσιών μέσω του δικτύου σε τοπικά δίκτυα και συσκευές μέσω ενός τυποποιημένου τρόπου σύνδεσης των συσκευών με το διαδίκτυο.

Λόγω της μεγάλης ευελιξίας του στην αναβάθμιση και προσθήκη νέων υπηρεσιών το R-OSGi είναι ιδανικό για εφαρμογές που αφορούν στη σύνδεση και απομακρυσμένη διαχείριση έξυπνων συσκευών όπως έξυπνο σπίτι, ασύρματων Bluetooth συσκευών, δρομολογητών, αποκωδικοποιητών τηλεόρασης, παροχή και μέτρηση ενέργειας για το σπίτι, εφαρμογές συστημάτων ασφαλείας με απομακρυσμένο έλεγχο, εφαρμογές συνεχούς φροντίδας ασθενών και προγνωστικό έλεγχο μέσω αναφορών για δυσλειτουργία οικιακών συσκευών.

Οι προδιαγραφές R-OSGi ξεκίνησαν στην αγορά των ενσωματωμένων οικιακών αυτοματισμών αλλά από το 1998 χρησιμοποιούνται ευρέως σε πολλές βιομηχανίες, αυτοκινητοβιομηχανίες, κινητή τηλεφωνία, αυτοματισμούς βιομηχανιών, πύλες και routers, ιδιωτικές συναλλαγές καταστημάτων, σταθερή τηλεφωνία και άλλες.

Είναι ιδανικό για μικρές και ενσωματωμένες συσκευές με περιορισμένη μνήμη και εύρος λόγω της χρήσης αποδοτικού δικτυακού πρωτοκόλλου και της συνύπαρξης διαφορετικών στοιχείων/εφαρμογών στο ίδιο JVM όπου ως εκ τούτου μπορούν να μοιράζονται κώδικα, γεγονός που μειώνει την απαιτούμενη μνήμη.

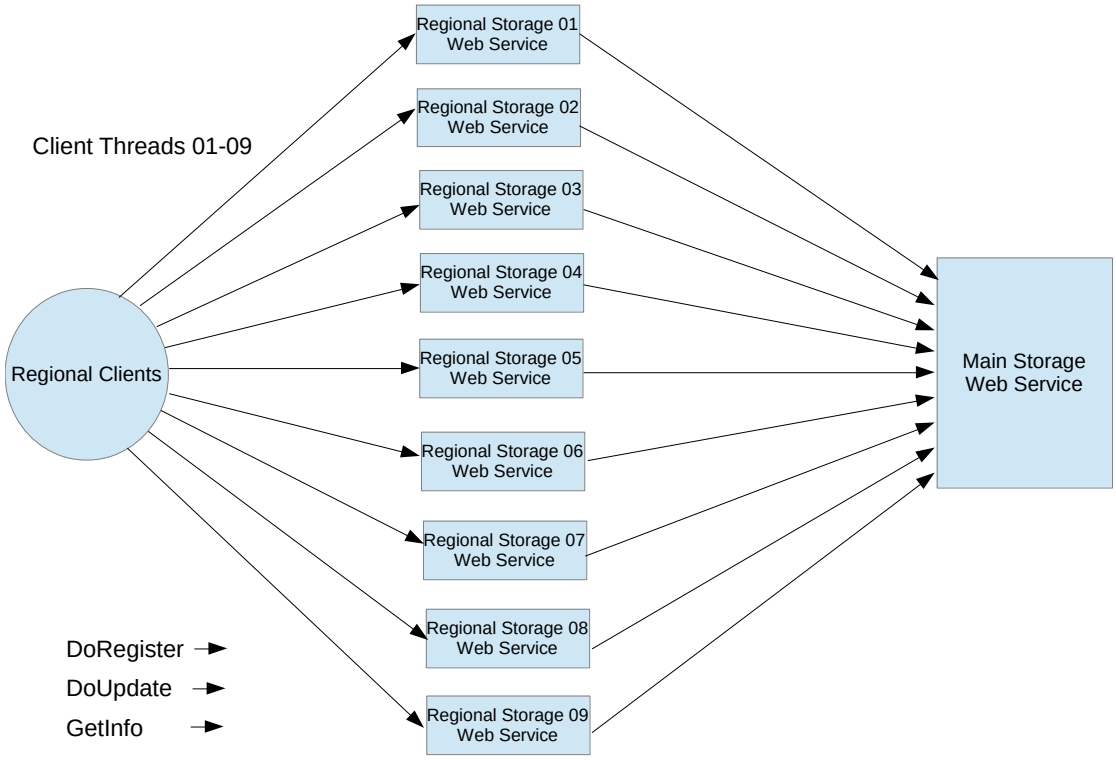
Είναι ιδανικό επίσης για συστήματα σε περιπτώσεις που υπάρχουν περισσότερα από ένα άτομα που συνεισφέρουν πχ εφαρμογές ανοιχτού κώδικα. Η χρήση του σε περιβάλλοντα που απαιτείται υψηλή αξιοπιστία είναι επίσης ιδανική, αφού επιτρέπει την προσθήκη η αφαίρεση συστατικών χωρίς την ανάγκη επανεκκίνησης της πλατφόρμας.

Ο βασικός λόγος που το R-OSGi είναι τόσο επιτυχές είναι ότι προσφέρει ένα αρκετά ώριμο σύστημα στοιχείων που λειτουργεί πραγματικά σε έναν εκπληκτικό αριθμό από περιβάλλοντα. Λόγω δυναμικής διαχείρισης του κύκλου ζωής των επιμέρους στοιχείων είναι κατάλληλο για συσκευές που πρέπει να αναβαθμίζονται αυτόματα ενώ δουλεύουν αδιάκοπα καθώς επίσης αποτελεί ένα ολοκληρωμένο μοντέλο ασφάλειας για όλα τα μέρη του συστήματος.

5. Αξιολόγηση Κλιμακωσιμότητας Υποδομής Web Services για την Ανάπτυξη Κατανεμημένων Εφαρμογών

5.1 Εισαγωγή

Η υλοποίηση έχει πραγματοποιηθεί σε περιβάλλον Eclipse και περιλαμβάνει την προσομοίωση ενός δικτύου μιας εμπορικής εφαρμογής αποτελούμενη από 9 Regional Web Services αποθήκες και μια κεντρική όπου είναι το Main Web Service και έχει το ρόλο του να ενημερώνεται για την πληροφορία όλων των υπολοίπων. Οι Regional αποθήκες βρίσκονται στα projects RegionalStorageWebService 01-09 και μαζί με την κεντρική η οποία βρίσκεται στο project MainStorageWebService είναι ανεβασμένες σε ένα Tomcat server και περιμένουν από τους πελάτες να επικοινωνήσουν μαζί τους.



Οι Clients βρίσκονται στο project ClientForRegionalWebServices και πιο συγκεκριμένα ξεκινάνε από την main της κλάσης RegionalClients. Αυτή αναλαμβάνει να δημιουργήσει τόσα threads πελατών όσα έχουμε καθορίσει στη μεταβλητή number Of Clients, τα οποία θα στέλνουν αιτήματα στις αποθήκες ταυτόχρονα. Τα αιτήματα αυτά αφορούν κάποια από τις ενέργειες καταχώρησης, ανανέωσης και ερώτησης ποσότητας προϊόντος. Ο αριθμός των αιτημάτων καθορίζεται από τη μεταβλητή NUMBER_OF_ACTIONS της κλάσης RegionalClientThreadFather. Παράλληλα, μέσω της βιβλιοθήκης log4j καταγράφονται κάποια logs, όπου

αφορούν στον υπολογισμό του χρόνου αποστολής και λήψης των αιτημάτων, στους χρόνους εκτέλεσης των threads και των συναρτήσεων της προσομοίωσης και στον υπολογισμό της μνήμης του συστήματος που χρησιμοποιείται.

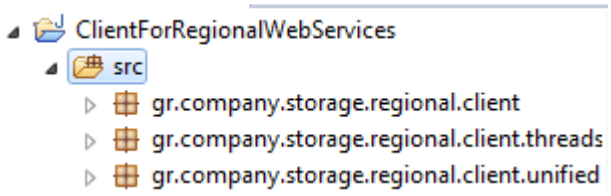
Τέλος, η κλάση CountMessageTime του project "StorageUtilities" διαβάζει το αρχείο με τα logs και αναλαμβάνει τον υπολογισμό των παραμέτρων για την αξιολόγηση της συμπεριφοράς του συστήματος και της υποδομής Web Services κατά την κλιμάκωση της απαίτησης που προκύπτει.

5.2Αρχιτεκτονική του συστήματος

Ακολουθεί πιο αναλυτικά η δομή της αρχιτεκτονικής του εν λόγω συστήματος.

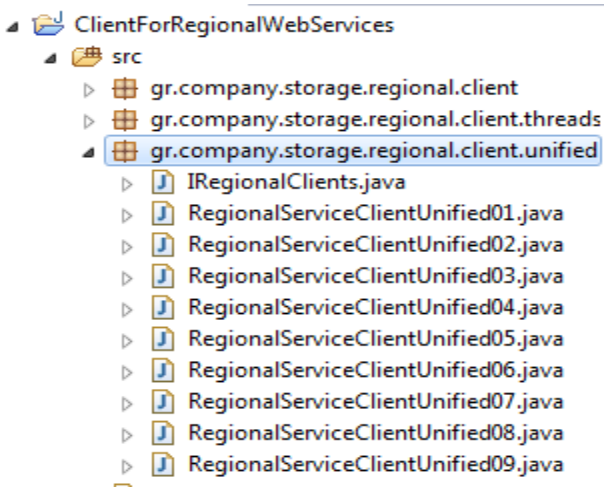
5.2.1Clients

Οι Clients είναι οι οντότητες όπου επικοινωνούν με τις περιφερειακές αποθήκες με τη δημιουργία των αντίστοιχων threads και τους αποστέλλουν αιτήματα ταυτόχρονα. Εδώ θα πρέπει να διευκρινιστεί πως ο κάθε Client στέλνει αιτήματα σε μία αντίστοιχή Regional αποθήκη. Τα αιτήματα αυτά είναι συγκεκριμένα η καταχώρηση εμπορεύματος, η ανανέωση εμπορεύματος και η ερώτηση ποσότητας εμπορεύματος. Οι υλοποίηση των Clients περιλαμβάνεται στο project ClientForRegionalWebServices



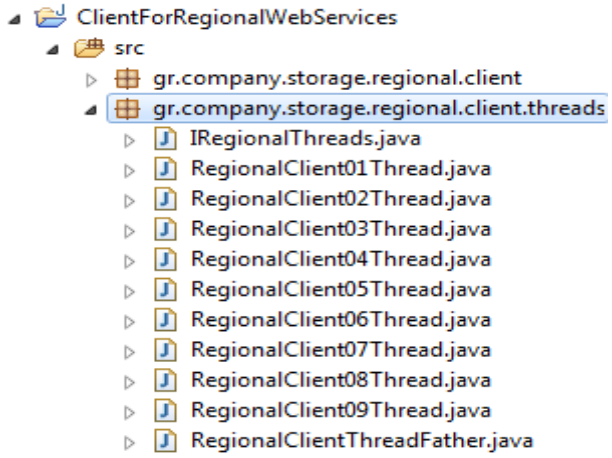
Το project ClientForRegionalWebServices αποτελείται από 3 packages όπως φαίνεται και στην παραπάνω εικόνα. Αυτά είναι:

Το unified package, μέσω του οποίου πραγματοποιείται η ενοποίηση όλων των clients σε ένα interface ώστε αντί να κληθεί κάποιος συγκεκριμένος Client καλείται το interface μέσω του οποίου προκύπτει κάποιος client.



Οι κλάσεις Unified 01-09 κάνουν implement το interface IRegionalClients όπου καθορίζονται οι αντίστοιχες συναρτήσεις.

Το δεύτερο package του project ClientForRregionalWebServices είναι το threads package, όπου περιλαμβάνονται τα threads τα οποία τρέχουν παράλληλα και καλούν τα Web Services.



Η κάθε κλάση RegionalClient01Thread έως RegionalClient09Thread κάνει implement το interface IRegionalThreads και επεκτείνει το RegionalClientThreadFather. Η κλάση RegionalClientThreadFather είναι το κυρίως Thread από το οποίο κληρονομούν όλα τα υπόλοιπα και εδώ απεικονίζεται το τι κάνει το κάθε thread κάθε φορά που τρέχει. Θα πρέπει να αναφερθεί επίσης πως το κάθε thread που υλοποιείται θέτει την αντίστοιχη Regional αποθήκη.

```
16 public abstract class RegionalClientThreadFather extends Thread {
17
18     protected IRegionalClients theClient = null; // for the children to fill in
19
20     protected String warehouseID = ""; // for the children to fill i
21
22     protected final int NUMBER_OF_ACTIONS = 100;
23
24     protected final int NUMBER_OF_PRODUCTS = 4;
25
26     protected final int TOP_QUANTITY = 100;
27
28     protected final int NUMBER_OF_POSSIBLE_ACTIONS = 3; // register, update,
29                                                         // getInfo
30
31     protected Logger logger = Logger.getLogger("client.log");
32
33     // και με enum γίνεται
34     protected final int DO_REGISTER = 0;
35     protected final int DO_UPDATE = 1;
36     protected final int DO_GETINFO = 2;
```

Όπως απεικονίζεται παραπάνω, αρχικά ορίζεται το theClient όπου είναι το Interface με το οποίο είναι δυνατή η προσθήκη οποιουδήποτε Client. Στο αντικείμενο αυτό μπορεί να ανατεθεί δηλαδή οποιαδήποτε κλάση υλοποιεί το interface IRegionalClients. Εν συνεχεία ορίζονται το warehouseID όπου είναι το ID της αποθήκης που θα κληθεί, και το NUMBER_OF_ACTIONS όπου εδώ είναι 100 και αντιστοιχεί στο πόσες φορές το thread που τρέχει θα καλέσει τη regional αποθήκη. Το NUMBER_OF_PRODUCTS είναι ο αποδεκτός αριθμός προϊόντων, TOP_QUANTITY η μέγιστη ποσότητα των προϊόντων και NUMBER_OF_POSSIBLE_ACTIONS ο αριθμός που αντιπροσωπεύει το πόσες είναι οι πιθανές ενέργειες που θα αποστέλλονται προς τις Regional αποθήκες. Στην συγκεκριμένη περίπτωση έχουμε τρεις και είναι οι register, update και getinfo οι οποίες αφορούν στην καταχώρηση, ανανέωση και ερώτηση ποσότητας προϊόντων αντίστοιχα. Επίσης ορίζεται ο logger μέσω του οποίου γίνονται οι καταγραφές των logs.

Το που βρίσκεται το log file ώστε να γίνει η καταγραφή των μηνυμάτων προσδιορίζεται από την βιβλιοθήκη log4j. Έτσι στη μέθοδο run με την οποία τρέχει το κάθε thread, περιλαμβάνεται η εντολή for που εκτελείται τόσες φορές όσες έχουμε προκαθορίσει στο NUMBER_OF_ACTIONS δηλαδή 100, ενώ κάθε φορά ακολουθείται κάποιο από τα cases που αντιστοιχούν στις πιθανές ενέργειες register, update και getinfo. Επιπλέον απεικονίζεται η αρχικοποίηση του αντικειμένου thMxB της κλάσης ThreadMXBean η οποία δίνει λειτουργίες για την παρακολούθηση του χρόνου κατά τον οποίο τα threads καταλαμβάνουν CPU για την εκτέλεσή τους. Ορίζεται επίσης το αντικείμενο osMxB της κλάσης OperatingSystemMXBean η οποία δίνει πληροφορίες για το λειτουργικό σύστημα, όπως η ελεύθερη μνήμη του συστήματος ή το μέσο όρο CPU που χρησιμοποιείται.

```
46         super.run();
47         // the child will have set this correctly (non-null)
48         ThreadMXBean thMxB = ManagementFactory.getThreadMXBean();
49         OperatingSystemMXBean osMxB = ManagementFactory
50                 .getOperatingSystemMXBean();
```

```
72          for (int i = 0; i < NUMBER_OF_ACTIONS; i++) {
73              try {
74                  Long mem =  (Long) freeMem.invoke(osMxB);
75                  double avrg = osMxB.getSystemLoadAverage();
76                  logger.info(String.format("%d:CLI:%s:%d",
77                          mem, "FREE-MEMORY", System.nanoTime()));
78                  logger.info(String.format("%f:CLI:%s:%d",
79                          avrg, "AVRG-CPU", System.nanoTime()));
80
```

Έτσι στην εντολή for βλέπουμε την καταγραφή logs για την ελεύθερη μνήμη του συστήματος, το μέσο όρο CPU (επιπλέον προδιαγραφή της υλοποίησης υποστηριζόμενη κυρίως από Linux), και εν συνεχεία ακολουθεί η αποστολή των αιτημάτων των πελατών προς τις regional αποθήκες και η καταγραφή των logs για την αποστολή τους. Πιο κάτω ακόμη απεικονίζεται και η καταγραφή των logs που αφορούν στο χρόνο εκτέλεσης του thread.

DO_REGISTER:

```
93              case DO_REGISTER:
94                  // register product
95                  nextProduct = randomGenerator.nextInt(NUMBER_OF_PRODUCTS);
96                  productsNumber = randomGenerator.nextInt(TOP_QUANTITY);
97                  try {
98                      String messageUID = UUID.randomUUID().toString();
99                      logger.info(String.format("%s:%d:CLI", messageUID,
00                              System.nanoTime()));
01                      // System.out.println(String.format("%s:%d:CLI", messageUID,
02                      // Calendar.getInstance().getTime().getTime()));
03                      theClient.doRegisterCargo(
04                              String.format("%s-%d", warehouseID, nextProduct),
05                              productsNumber, messageUID);
```

DO_UPDATE:

```
118             case DO_UPDATE:
119                 // Update Product
120                 nextProduct = randomGenerator.nextInt(NUMBER_OF_PRODUCTS);
121                 productsNumber = randomGenerator.nextInt(TOP_QUANTITY);
122                 try {
123                     String messageUID = UUID.randomUUID().toString();
124                     logger.info(String.format("%s:%d:CLI", messageUID,
125                             System.nanoTime()));
126                     // System.out.println(String.format("%s:%d:CLI", messageUID,
127                     // Calendar.getInstance().getTime().getTime()));
128                     theClient.doUpdateCargo(
129                             String.format("%s-%d", warehouseID, nextProduct),
130                             productsNumber, messageUID);
```

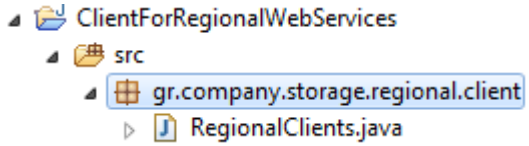
DO_GETINFO

```
142         case DO_GETINFO:
143             // Get Info
144             nextProduct = randomGenerator.nextInt(NUMBER_OF_PRODUCTS);
145             try {
146                 String messageUID = UUID.randomUUID().toString();
147                 logger.info(String.format("%s:%d:CLI", messageUID,
148                         System.nanoTime()));
149                 // System.out.println(String.format("%s:%d:CLI", messageUID,
150                 // Calendar.getInstance().getTime().getTime()));
151                 productsNumber = theClient.findCargoQuantity(
152                         String.format("%s-%d", warehouseID, nextProduct),
153                         messageUID);
```

Καταγραφή χρόνου εκτέλεσης thread.

```
173         logger.info(String.format("%d:CLI:%s:%d",
174                 thMxB.getCurrentThreadCpuTime(), "TOTAL-TIME",
175                 System.nanoTime()));
```

Τέλος το τρίτο package του project ClientForRregionalWebServices είναι το package client στο οποίο βρίσκεται η κλάση RegionalClients από την main της οποίας ξεκινάνε οι clients.



Συγκεκριμένα η κλάση RegionalClients αναλαμβάνει να δημιουργήσει threads πελατών τα οποία στέλνουν ταυτόχρονα αιτήματα στις regional αποθήκες. Ο αριθμός των threads που θα δημιουργηθούν εξαρτάται από τη μεταβλητή numberofClients όπου παρακάτω αντιστοιχεί σε 10 και μπορεί να μεταβληθεί για τις ανάγκες των μετρήσεων. Ως παράδειγμα επίσης απεικονίζονται οι συναρτήσεις δημιουργίας και επιστροφής threads των δύο πρώτων πελατών.

```
23 public class RegionalClients {
24
25     private final int numberOfClients = 10;
26     private Random myRandom = new Random();
27
28     /**
29      * @return regional client 01 thread
30      */
31     protected RegionalClient01Thread createRegionalClient01Thread() {
32         RegionalClient01Thread regionalClient01Thread = new RegionalClient01Thread();
33         return regionalClient01Thread;
34     }
35
36     /**
37      * @return regional client 02 thread
38      */
39     protected RegionalClient02Thread createRegionalClient02Thread() {
40         RegionalClient02Thread regionalClient02Thread = new RegionalClient02Thread();
41         return regionalClient02Thread;
42     }
```

Στη βασική συνάρτηση runclients περιλαμβάνεται η εντολή for η οποία εκτελείται τόσες φορές όσες προκαθορίσουμε με την μεταβλητή numberofclients, και ακολουθεί κάθε φορά κάποιο τυχαίο case για τις τιμές 1 έως 9 όπου το καθένα αφορά στη δημιουργία του αντίστοιχου thread.

```
109         for (int i=0; i<numberOfClients; i++) {
110             IRegionalThreads myThread;
111 
112             int nextThread = myRandom.nextInt(9) + 1;
113             switch (nextThread) {
114             case 1:
115                 myThread = createRegionalClient01Thread();
116                 break;
117             case 2:
118                 myThread = createRegionalClient02Thread();
119                 break;
120             case 3:
121                 myThread = createRegionalClient03Thread();
122                 break;
123             case 4:
124                 myThread = createRegionalClient04Thread();
125                 break;
```

Έτσι δημιουργούνται τα threads πελατών 01 – 09 (παραπάνω φαίνονται ως παράδειγμα τα τέσσερα πρώτα) και εν συνεχεία προστίθενται σε μια λίστα και ξεκινάνε.

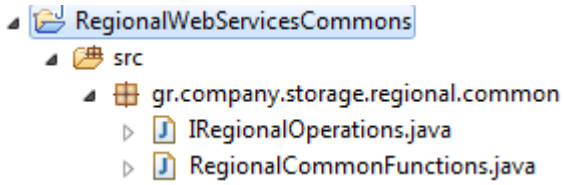
```
141             default:
142                 throw new Exception("Invalid Value");
143             }
144             threadList.add(myThread);
145         }
146 
147         for (int i=0; i<numberOfClients; i++) {
148             threadList.get(i).start();
149         }
150     }
151 
152     /**
153      * @param args
154      * @throws Exception
155      */
156     public static void main(String[] args) throws Exception {
157         RegionalClients regC = new RegionalClients();
158         regC.runClients();
159     }
160 
161 }
```

5.2.2 Regional Αποθήκες

Είναι τα Web Services τα οποία δέχονται τα αιτήματα των πελατών και εφόσον τα εκτελέσουν εν συνεχεία αποστέλλουν αίτημα για την ενημέρωση του Main Web Service που είναι η κεντρική αποθήκη.

Η κοινή υλοποίηση των 9 regional αποθηκών περιλαμβάνεται στο project RegionalWebServicesCommons και συγκεκριμένα στην κλάση

RegionalCommonFunctions η οποία κάνει implement το IRegionalOperations όπου βρίσκονται οι κοινές συναρτήσεις που πρέπει όλα τα Web Services να υλοποιούν.



Στην κλάση RegionalCommonFunctions όπως απεικονίζεται παρακάτω ορίζονται το mainService όπου λειτουργεί ως διαμεσολαβητής για την επικοινωνία με την κεντρική αποθήκη καθώς επίσης το hashmap στο οποίο καταχωρούνται τα δεδομένα και έχει ως κλειδί (string) το όνομα του προϊόντος και τιμή (integer) την ποσότητά του. Εδώ θα πρέπει να αναφερθεί πως οι Regional αποθήκες έχουν δικό τους hasmap ενώ η Main το δικό της. Ορίζονται επίσης το ID της αποθήκης storageID, το όνομα της αποθήκης storageName και ο logger για την καταχώρηση των logs που αφορούν στην λήψη αιτημάτων από τους Clients, την αποστολή αιτημάτων προς την Main και τον χρόνο εκτέλεσης των συναρτήσεων. Το που βρίσκεται το log file ώστε να γίνει η καταγραφή των logs όπως αναφέρθηκε και πριν, προσδιορίζεται από την βιβλιοθήκη log4j.

```
16 public abstract class RegionalCommonFunctions implements IRegionalOperations {
17
18     protected MainStorageWebServiceProxy mainService = new MainStorageWebServiceProxy();
19     /**
20      * Static Hashmap με κλειδί όνομα προϊόντος και τιμή την ποσότητά του.
21      */
22     protected static HashMap<String, Integer> cargoQuantities = new HashMap<String, Integer>();
23     protected static String storageID = null;
24     protected String storageName = ""; // αυτό αλλάζει η κάθε υλοποίηση για να
25                                        // διαφοροποιηθεί.
26     protected Logger logger = Logger.getLogger("storage.log");
```

Αφότου μια Regional αποθήκη αρχικά καταχωρηθεί στην Main αποθήκη με τυχαίο storageID, εν συνεχεία όπως απεικονίζεται κάνει register το εμπόρευμα όπου εστάλει από τον αντίστοιχο Client ενώ καταγράφονται στο log file η λήψη του αντίστοιχου αιτήματος, η αποστολή του αιτήματος στη Main εφόσον εκτελεστεί καθώς επίσης ο χρόνος εκτέλεσης την αντίστοιχης συνάρτησης όπου δίνεται μέσω της μεταβλητής timeDifference. Επίσης σύμφωνα με το exception που δίνεται, η καταχώρηση του εμπορεύματος και η αποστολή αιτήματος για καταχώρηση στην Main αποθήκη θα πραγματοποιηθεί αν το εμπόρευμα δεν είναι ήδη καταχωρημένο.

```
63      public void doRegisterCargo(String cargoID, Integer quantity,
64              String messageUID) throws Exception {
65          long startTime = System.nanoTime();
66          logger.info(String.format("%s:%d:%s", messageUID, System.nanoTime(), storageName));
67  //      System.out.println(String.format("%s:%d:%s", messageUID, Calendar
68  //              .getInstance().getTime().getTime(), storageName));
69
70          if (checkCargoExistence(cargoID)) {
71              throw new Exception("ERR-R01 : Cargo Already Registered!");
72          }
73
74          String uuid = UUID.randomUUID().toString();
75          logger.info(String.format("%s:%d:%s", uuid, System.nanoTime(), storageName));
76  //      System.out.println(String.format("%s:%d:%s", uuid, Calendar
77  //              .getInstance().getTime().getTime(), storageName));
78          long medianBefore = System.nanoTime();
79          mainService.doRegisterCargoOfStorage(storageID.toString(), cargoID,
80                  quantity, uuid);
81          long medianAfter = System.nanoTime();
82          cargoQuantities.put(cargoID, quantity);
83          long endTime = System.nanoTime();
84          long timeDifference = endTime - medianAfter + medianBefore - startTime;
85
86          //System.out.println("doRegisterCargo : " +timeDifference);
87          logger.info(String.format("%d:%s:%s:%s", timeDifference, "RE", "REGISTER-CARGO", storageName));
88
```

Με τον ίδιο τρόπο γίνεται και η ανανέωση της ποσότητας εμπορεύματος. Η Regional αποθήκη καταγράφει στο log file τη λήψη του αιτήματος από τον αντίστοιχο Client, τον χρόνο που καταλαμβάνεται στη CPU για την εκτέλεση της αντίστοιχης συνάρτησης καθώς επίσης την αποστολή του αιτήματος ανανέωσης ποσότητας στη Main αποθήκη. Σύμφωνα με τα exceptions θα πρέπει να υπάρχει καταχωρημένο εμπόρευμα και να υφίσταται αλλαγή στην ποσότητα του προϊόντος. Η Regional αποθήκη προσθέτει στο hashmap της την ανανέωση της ποσότητας σε περίπτωση που η ποσότητα αυτή δεν είναι μικρότερη του μηδενός.

```
92      public void doUpdateCargo(String cargoID, Integer updateQuantity,
93              String messageUID) throws Exception {
94          long startTime = System.nanoTime();
95          logger.info(String.format("%s:%d:%s", messageUID, System.nanoTime(), storageName));
96  //      System.out.println(String.format("%s:%d:%s", messageUID, Calendar
97  //              .getInstance().getTime().getTime(), storageName));
98
99          if (!checkCargoExistence(cargoID)) {
100             throw new Exception("ERR-R02 : Cargo Not Registered!");
101         }
102
103         if (cargoQuantities.get(cargoID) == updateQuantity) {
104             throw new Exception("ERR-R03 : No change in Cargo Quantity");
105         }
106
107         String uuid = UUID.randomUUID().toString();
108         logger.info(String.format("%s:%d:%s", uuid, System.nanoTime(), storageName));
109 //      System.out.println(String.format("%s:%d:%s", uuid, Calendar
110 //              .getInstance().getTime().getTime(), storageName));
111         long medianBefore = System.nanoTime();
112         mainService.doUpdateCargo(storageID.toString(), cargoID,
113                 updateQuantity, uuid);
114         long medianAfter = System.nanoTime();
115         if (updateQuantity <= 0) {
116             cargoQuantities.remove(cargoID);
117         } else {
118             cargoQuantities.put(cargoID, updateQuantity);
119         }
120         long endTime = System.nanoTime();
121         long timeDifference = endTime - medianAfter + medianBefore - startTime;
122         logger.info(String.format("%d:%s:%s:%s", timeDifference, "RE", "UPDATE-CARGO", storageName));
123     }
```

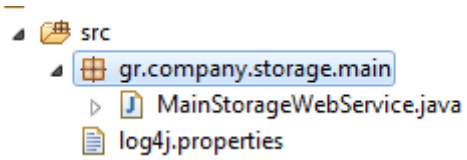
Τέλος, όπως φαίνεται στον κώδικα που ακολουθεί, εκτελείται η εύρεση της ποσότητας προϊόντος, καταγράφονται τα αντίστοιχα logs στο log file και επιστρέφεται ο αριθμός στην μεταβλητή totalnumber.

```
126     public Integer findCargoQuantity(String cargoID, String messageUID) {
127         long startTime = System.nanoTime();
128         Integer totalNumber = 0;
129
130         logger.info(String.format("%s:%d:%s", messageUID, System.nanoTime(), storageName));
131 //      System.out.println(String.format("%s:%d:%s", messageUID, Calendar
132 //              .getInstance().getTime().getTime(), storageName));
133
134         if (checkCargoExistence(cargoID)) {
135             totalNumber = cargoQuantities.get(cargoID);
136         }
137         long endTime = System.nanoTime();
138         long timeDifference = endTime - startTime;
139         logger.info(String.format("%d:%s:%s:%s", timeDifference, "RE", "CARGO-QUANTITY", storageName));
140         return totalNumber;
141     }
```

5.2.3 Main Αποθήκη

Η Main αποθήκη είναι το κεντρικό Web Service όπου κρατάει την πληροφορία για όλα τα υπόλοιπα Regional Web Services. Συγκεκριμένα η Main αποθήκη δεν

αποστέλλει κάποιο αίτημα προς άλλη οντότητα αλλά λαμβάνει αιτήματα από τις Regional αποθήκες και βάσει αυτών διαμορφώνει το hashmap της. Η υλοποίηση της κεντρικής αποθήκης περιλαμβάνεται στην κλάση MainStorageWebService η οποία βρίσκεται στο package gr.company.storage.main του project MainStorageWebService.



Αφού έχουν οριστεί το hashmap, ο logger και έχει καταχωρηθεί κάποια Regional αποθήκη στη main, όπως απεικονίζεται παρακάτω η κεντρική αποθήκη καταγράφει ένα εμπόρευμα για μια Regional αποθήκη, με την ποσότητα που δίνεται, ενώ στο log file καταγράφονται τα logs που αφορούν στη λήψη του αντίστοιχου αιτήματος της Regional αποθήκης και στον χρόνο εκτέλεσης της εν λόγω συνάρτησης. Σύμφωνα με τα exception που δίνονται αν υπάρχει ήδη το εμπόρευμα ή δεν υπάρχει η αποθήκη, το εμπόρευμα δεν μπορεί να καταχωρηθεί.

```
98    public void doRegisterCargoOfStorage(String storageID, String cargoID,
99            Integer quantity, String messageUID) throws Exception {
100       long startTime = System.nanoTime();
101       logger.info(String.format("%s:%d:MAIN", messageUID,System.nanoTime()));
102       //System.out.println(String.format("%s:%d:MAIN", messageUID,Calendar.getInstance().getTime().getTime()));
103
104       if (checkCargoOfStorageExistence(storageID, cargoID)) {
105           throw new Exception("ERR-002 : Cargo Already Registered!");
106       }
107
108       if (!checkStorageExistence(storageID)) {
109           throw new Exception("ERR-003 : Storage Not Registered!");
110       }
111
112       storagesAndQuantities.get(storageID).put(cargoID, quantity);
113       long endTime = System.nanoTime();
114       long timeDifference = endTime - startTime;
115       logger.info(String.format("%d:%s:%s:%s", timeDifference, "MAIN", "REGISTER-MAIN", "MAIN"));
116   }
```

Έπειτα, η Main ανανεώνει την ποσότητα ενός εμπορεύματος μια καταχωρημένης Regional αποθήκης, καταγράφονται τα αντίστοιχα logs στο log file, ενώ σύμφωνα με τα exceptions που δίνονται η ανανέωση δεν εκτελείται αν δεν υπάρχει η αποθήκη ή το εμπόρευμα ή αν η νέα ποσότητα είναι ίδια με την υπάρχουσα. Επίσης αν η ποσότητα είναι μικρότερη του 0, το εμπόρευμα διαγράφεται από το hashmap.

```
136    public void doUpdateCargo(String storageID, String cargoID,
137            Integer updateQuantity, String messageUID) throws Exception {
138        long startTime = System.nanoTime();
139        logger.info(String.format("%s:%d:MAIN", messageUID,System.nanoTime()));
140        //System.out.println(String.format("%s:%d:MAIN", messageUID,Calendar.getInstance().getTime().getTime()));
141
142        if (!checkCargoOfStorageExistence(storageID, cargoID)) {
143            throw new Exception("ERR-004 : Cargo or Storage Not Registered!");
144        }
145
146        if (storagesAndQuantities.get(storageID).get(cargoID) == updateQuantity) {
147            throw new Exception("ERR-005 : No change in Cargo Quantity");
148        }
149
150        if (updateQuantity <= 0) {
151            storagesAndQuantities.get(storageID).remove(cargoID);
152        } else {
153            storagesAndQuantities.get(storageID).put(cargoID, updateQuantity);
154        }
155        long endTime = System.nanoTime();
156        long timeDifference = endTime - startTime;
157        logger.info(String.format("%d:%s:%s:%s", timeDifference, "MAIN", "UPDATE-MAIN", "MAIN"));
158    }
```

Παρακάτω η Main επιστρέφει ένα πίνακα με τα ID των αποθηκών στις οποίες υπάρχει το εμπόρευμα.

```
171    public String[] findWhereCargoExists(String cargoID, String messageUID) {
172        logger.info(String.format("%s:%d:MAIN", messageUID,System.nanoTime()));
173        //System.out.println(String.format("%s:%d:MAIN", messageUID,Calendar.getInstance().getTime().getTime()));
174
175        ArrayList<String> storages = new ArrayList<String>();
176        String[] storagesArray = {};
177        String[] keys = {};
178        for (String storageID : storagesAndQuantities.keySet().toArray(keys)) {
179            if (checkCargoOfStorageExistence(storageID, cargoID)) {
180                storages.add(storageID);
181            }
182        }
183        return storages.toArray(storagesArray);
```

Τέλος επιστρέφει στην μεταβλητή totalnumber τον συνολικό αριθμό εμπορεύματος που υπάρχει σε όλες τις αποθήκες .

```
195    public Integer findTotalCargoNumber(String cargoID, String messageUID) {
196        logger.info(String.format("%s:%d:MAIN", messageUID,System.nanoTime()));
197        //System.out.println(String.format("%s:%d:MAIN", messageUID,Calendar.getInstance().getTime().getTime()));
198
199        Integer totalNumber = 0;
200        String[] keys = {};
201        for (String storageID : storagesAndQuantities.keySet().toArray(keys)) {
202            if (checkCargoOfStorageExistence(storageID, cargoID)) {
203                totalNumber += storagesAndQuantities.get(storageID)
204                        .get(cargoID);
205            }
206        }
207
208        return totalNumber;
209    }
```

5.3 Καταγραφή logs

Για την επίκληση κάθε υποστηριζόμενης ενέργειας αποστέλλονται αιτήματα, για τα οποία από την αντίστοιχη οντότητα Client, Regional Web Service ή Main Web Service καταγράφονται κάποια logs που αντιπροσωπεύουν τον χρόνο αποστολής και λήψης κάθε αιτήματος. Επιπρόσθετα, καταγραφή logs πραγματοποιείται για το χρόνο κατά τον οποίο καταλαμβάνεται χρήση CPU για την εκτέλεση των threads πελατών και των συναρτήσεων καθώς επίσης για την καταγραφή της ελεύθερης μνήμης που υπάρχει στο σύστημα τη δεδομένη στιγμή. Μέσω της βιβλιοθήκης log4j προσδιορίζεται το που βρίσκεται το log file στο οποίο θα γίνουν οι καταγραφές.

Πιο συγκεκριμένα οι Clients καταγράφουν στο log file την αποστολή κάθε αιτήματος register, update και get info προς την αντίστοιχη Regional αποθήκη, καταγράφουν επίσης logs που αφορούν στην ελεύθερη μνήμη του συστήματος, στο μέσο όρο CPU (επιπλέον προδιαγραφή της υλοποίησης υποστηριζόμενη κυρίως από Linux), ενώ επίσης καταγράφεται ο χρόνος κατά τον οποίο τα αντίστοιχα threads πελατών καταλαμβάνουν χρήση υπολογιστική ισχύος για την εκτέλεσή τους. Ως παράδειγμα απεικονίζεται η καταγραφή της αποστολής αιτήματος προς μια Regional αποθήκη για καταχώρηση προϊόντος.

```
60              case DO_REGISTER:
61                  // register product
62                  nextProduct = randomGenerator.nextInt(NUMBER_OF_PRODUCTS);
63                  productsNumber = randomGenerator.nextInt(TOP_QUANTITY);
64                  try {
65                      String messageUID = UUID.randomUUID().toString();
66                      logger.info(String.format("%s:%d:CLI", messageUID, System.nanoTime()));
67                      // System.out.println(String.format("%s:%d:CLI", messageUID,
68                      // Calendar.getInstance().getTime().getTime()));
69                      theClient.doRegisterCargo(
70                              String.format("%s-%d", warehouseID, nextProduct),
71                              productsNumber, messageUID);
```

Τα Regional Web Services με τη σειρά τους καταγράφουν logs που αφορούν στη λήψη των αιτημάτων των Clients, στην αποστολή αιτήματος καταχώρησης τους στο Main Web Service και στην αποστολή αιτημάτων για την ενημέρωση της Main αποθήκης σχετικά με την εκτέλεση των αιτημάτων που έλαβαν από τους πελάτες. Επίσης καταγράφονται logs που αφορούν στο χρόνο κατά τον οποίο καταλαμβάνεται χρήση υπολογιστικής ισχύος για την εκτέλεση των αντίστοιχων συναρτήσεων καταχώρησης, ανανέωσης και ερώτησης ποσότητας εμπορεύματος. Παρακάτω ως παράδειγμα απεικονίζεται η καταγραφή logs για την λήψη αιτήματος ανανέωσης της ποσότητας εμπορεύματος από τον αντίστοιχο Client, για την αποστολή του αιτήματος προς ενημέρωση της Main έπειτα από την εκτέλεση του αντίστοιχου αιτήματος πελάτη, καθώς επίσης για τον υπολογισμό του χρόνου κατά τον οποίο καταλαμβάνεται υπολογιστική ισχύς για την εκτέλεση της συνάρτησης ανανέωσης ποσότητας προϊόντος.

```
92      public void doUpdateCargo(String cargoID, Integer updateQuantity,
93              String messageUID) throws Exception {
94          long startTime = System.nanoTime();
95          logger.info(String.format("%s:%d:%s", messageUID, System.nanoTime(), storageName));
96  //      System.out.println(String.format("%s:%d:%s", messageUID, Calendar
97  //              .getInstance().getTime().getTime(), storageName));
98
99          if (!checkCargoExistence(cargoID)) {
100             throw new Exception("ERR-R02 : Cargo Not Registered!");
101         }
102
103         if (cargoQuantities.get(cargoID) == updateQuantity) {
104             throw new Exception("ERR-R03 : No change in Cargo Quantity");
105         }
106
107         String uuid = UUID.randomUUID().toString();
108         logger.info(String.format("%s:%d:%s", uuid, System.nanoTime(), storageName));
109 //      System.out.println(String.format("%s:%d:%s", uuid, Calendar
110 //              .getInstance().getTime().getTime(), storageName));
111         long medianBefore = System.nanoTime();
112         mainService.doUpdateCargo(storageID.toString(), cargoID,
113                 updateQuantity, uuid);
114         long medianAfter = System.nanoTime();
115         if (updateQuantity <= 0) {
116             cargoQuantities.remove(cargoID);
117         } else {
118             cargoQuantities.put(cargoID, updateQuantity);
119         }
120         long endTime = System.nanoTime();
121         long timeDifference = endTime - medianAfter + medianBefore - startTime;
122         logger.info(String.format("%d:%s:%s:%s", timeDifference, "RE", "UPDATE-CARGO", storageName));
123     }
```

Η Main αποθήκη αντίστοιχα καταγράφει logs για κάθε αίτημα που λαμβάνει από τις Regional αποθήκες. Επίσης καταγράφονται logs όπου αφορούν στο χρόνο κατά τον

οποίο καταλαμβάνεται υπολογιστική ισχύς για την εκτέλεση των συναρτήσεων. Ως παράδειγμα απεικονίζεται η καταγραφή λήψης του αιτήματος που αφορά την ανανέωση ποσότητας του εμπορεύματος μιας καταχωρημένης regional αποθήκης στην κεντρική καθώς επίσης η καταγραφή που αφορά στο χρόνο που χρειάστηκε για την εκτέλεση της συνάρτησης.

```
136    public void doUpdateCargo(String storageID, String cargoID,
137            Integer updateQuantity, String messageUID) throws Exception {
138        long startTime = System.nanoTime();
139        logger.info(String.format("%s:%d:MAIN", messageUID,System.nanoTime()));
140        //System.out.println(String.format("%s:%d:MAIN", messageUID,Calendar.getInstance().getTime().getTime()));
141
142        if (!checkCargoOfStorageExistence(storageID, cargoID)) {
143            throw new Exception("ERR-004 : Cargo or Storage Not Registered!");
144        }
145
146        if (storagesAndQuantities.get(storageID).get(cargoID) == updateQuantity) {
147            throw new Exception("ERR-005 : No change in Cargo Quantity");
148        }
149
150        if (updateQuantity <= 0) {
151            storagesAndQuantities.get(storageID).remove(cargoID);
152        } else {
153            storagesAndQuantities.get(storageID).put(cargoID, updateQuantity);
154        }
155        long endTime = System.nanoTime();
156        long timeDifference = endTime - startTime;
157        logger.info(String.format("%d:%s:%s:%s", timeDifference, "MAIN", "UPDATE-MAIN", "MAIN"));
158    }
```

Ακολουθεί η μορφοποίηση που ακολουθούν οι καταγραφές των αιτημάτων επίκλησης ενεργειών όπως αυτές απεικονίζονται στο log file. Τα συγκεκριμένα logs απαρτίζονται από τρία μέρη, όπως διακρίνονται παρακάτω. Το αναγνωριστικό του μηνύματος message UID, ο χρόνος καταγραφής του μηνύματος εκφρασμένος σε nanoseconds και το όνομα της οντότητας που κατέγραψε το μήνυμα.

```
8970b89b-96bc-4dfe-ae1b-30c026ef4cc2:1648943713540:CLI
605726c0-493c-4988-b86b-414226983b3f:1648943962798:RE07
6867c267-60e2-449d-85a7-8300ae934d85:1648946852355:MAIN
```

Τα logs που αφορούν στους χρόνους εκτέλεσης των threads πελατών και των συναρτήσεων, ακολουθούν την παρακάτω μορφοποίηση. Όπως διακρίνεται απαρτίζονται από τέσσερα μέρη. Το πρώτο μέρος αφορά στο χρόνο (nanoseconds) κατά τον οποίο έτρεχε ο αντίστοιχος πελάτης ή η συνάρτηση στη CPU. Το δέυτερο μέρος αφορά στο τι είδους οντότητα έκανε την καταγραφή. Το τρίτο μερος περιλαμβάνει το αναγνωριστικό από το οποίο καθορίζεται αν πρόκειται για χρόνο εκτέλεσης πελάτη ή συνάρτησης καθώς επίσης ποιας συνάρτησης. Το τελευταίο μέρος αφορά στο ποια ήταν η συγκεκριμένη οντότητα η οποία κατέγραψε το log.

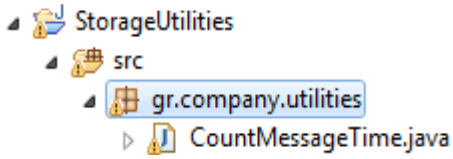
```
308925:RE:UPDATE-CARGO:RE09
280801800:CLI:TOTAL-TIME:Thread-22
```

Η μορφοποίηση των logs που σχετίζονται με την καταγραφή με σκοπό τον υπολογισμό της μνήμης του συστήματος, αποτελούνται ομοίως από τέσσερα μέρη. Τα μέρη αυτά αφορούν τα Bytes ελεύθερης μνήμης του συστήματος τη δεδομένη στιγμή, το είδος οντότητας που έκανε την καταγραφή, το αναγνωριστικό της καταγραφής μέσω του οποίου θα αποδοθεί η τιμή στην αντίστοιχη μεταβλητή για τον υπολογισμό της χρησιμοποιούμενης μνήμης και τέλος ο χρόνος κατά τον οποίο έγινε η καταγραφή σε nanoseconds. Η ίδια μορφοποίηση ακολουθείται και για την υποστηριζόμενη προδιαγραφή απόδοσης του μέσου όρου χρησιμοποιούμενης CPU.

```
5485723648:CLI:FREE-MEMORY:384054305907683
5488627712:CLI:FREE-MEMORY:384054307829757
5485678592:CLI:FREE-MEMORY:384054304630416
5487017984:CLI:FREE-MEMORY:384054306488040
```

5.4 Υπολογισμός παραμέτρων αξιολόγησης του συστήματος – CountMessageTime

Ο υπολογισμός των τιμών που προκύπτουν για την αξιολόγηση της συμπεριφοράς του συστήματος πραγματοποιείται με την κλάση CountMessageTime η οποία βρίσκεται στο package gr.company.utilities του project StorageUtilities και διαβάζει το αρχείο καταγραφών.



Στην κλάση CountMessageTime ορίζονται αρχικά το αρχείο από το οποίο η κλάση θα διαβάσει τα logs, καθώς επίσης το hashmap με κλειδί (string) το αναγνωριστικό του κάθε μηνύματος (message UID) και Long τον χρόνο που διαβάστηκε το κάθε μήνυμα.

```
14 public class CountMessageTime {
15
16     private static final String filename = "c:/props/service-client.log";
17     private static Map<String, Long> tokensTimes = new HashMap<String, Long>();
```

Εν συνεχεία αρχικοποιείται η μεταβλητή totalTimes όπου αντιστοιχεί στον συνολικό αριθμό των αιτημάτων που θα διαβαστούν και η μεταβλητή totalTime στην οποία προστίθεται κάθε φορά η διαφορά χρόνου μεταξύ αποστολής και λήψης κάθε αιτήματος. Επίσης αρχικοποιούνται μεταβλητές στις οποιες αποδίδονται οι τιμές για τον υπολογισμό του χρόνου εκτέλεσης των threads πελατών, των συναρτήσεων, του απαιτούμενου χρόνου για την εκτέλεση της όλης προσομοίωσης καθώς και για τον υπολογισμό της χρησιμοποιούμενης μνήμης του συστήματος.

Όπως αναφέρθηκε κατά την περιγραφή των logs στην προηγούμενη παράγραφο, κάθε καταγραφή περιλαμβάνει κάποια αναγνωριστικά. Έτσι στα logs που προκύπτουν για την καταγραφή αιτημάτων καθορίζεται ένα μοναδικό αναγνωριστικό του αιτήματος ενώ στα logs που αφορούν στους χρόνους εκτέλεσης threads πελατών και συναρτήσεων τα αναγνωριστικά όπως UPDATE-CARGO ή TOTAL-TIME καθορίζουν αν πρόκειται για υπολογισμό χρόνου εκτέλεσης πελάτη ή εκτέλεσης συνάρτησης καθώς επίσης ποιας συνάρτησης. Βάσει των αναγνωριστικών κάθε καταγραφής, οι χρόνοι των logs ανατίθονται στις αντίστοιχες μεταβλητές μέσω των οποίων υπολογίζονται τα δεδομένα όπου προκύπτουν για τον χαρακτηρισμό της συμπεριφοράς της προσομοίωσης.

Έτσι η μεταβλητή funcTimeSec δίνει τον χρόνο εκφρασμένο σε seconds κατά τον οποίο η εκτέλεση των συναρτήσεων καταλαμβάνει υπολογιστική ισχύ στο σύστημα στο οποίο εκτελείται η προσομοίωση. Προκύπτει από το άθροισμα των καταγεγραμμένων χρόνων όπου ανατίθενται στις μεταβλητές sumRegister, sumUpdate και sumInfo για τις συναρτήσεις καταγραφής, ανανέωσης και ερώτησης ποσότητας προϊόντων αντίστοιχα ενώ λαμβάνεται επίσης υπόψη η παραλληλία για επεξεργασία που προσφέρει το σύστημα στο οποίο τρέχει η προσομοίωση βάσει της τεχνολογίας επεξεργαστή που ακολουθεί .

```
if (totalTimes != 0) {
    int processors = Runtime.getRuntime()
            .availableProcessors();
    funcTime = (sumRegister + sumUpdate + sumInfo);
    double funcTimeSec = funcTime/processors * 1.0e-9;
```

Η μεταβλητή threadTimeSec δίνει τον χρόνο εκφρασμένο σε seconds κατά τον οποίο η εκτέλεση των threads πελατών καταλαμβάνει υπολογιστική ισχύ στο σύστημα όπου εκτελείται η προσομοίωση. Προκύπτει από το άθροισμα των καταγεγραμμένων χρόνων εκτέλεσης των threads πελατών όπου ανατίθεται στην μεταβλητή sumThreadTime ενώ επίσης και εδώ λαμβάνεται υπόψη η παραλληλία για επεξεργασία που προσφέρει το σύστημα. Η μεταβλητή simulationTime, αποδίδει τον συνολικό χρόνο εκφρασμένο σε seconds κατά τον οποίο διήρκησε η εκτέλεση όλης της προσομοίωσης. Προκύπτει από την διαφορά των μεταβλητών startTime και endTime από τις οποίες αντιπροσωπεύεται ο χρόνος έναρξης

εκτέλεσης της προσομοίωσης έως και την εκτέλεση και καταγραφή του τελευταίου αιτήματος.

```
double threadTimeSec = (double) (sumThrdTime / processors) * 1.0e-9;
double simulationTime = (endTime - startTime)* 1.0e-9;
```

Στη μεταβλητή totaltime προστίθεται κάθε φορά η διαφορά του χρόνου για την αποστολή και λήψη του κάθε αιτήματος, ενώ στη μεταβλητή totalTimes προστίθεται κάθε φορά ο αριθμός αιτημάτων που διαβιβάστηκαν μεταξύ των οντοτήτων Clients, Regional Web Services και Main Web Service και διαβάστηκαν από την κλάση CountMessageTime. Το πηλίκο που προκύπτει από την διαίρεση των totalTime και totalTimes, αποδίδει τον μέσο χρόνο μεταξύ αποστολής και λήψης των αιτημάτων της προσομοίωσης ο οποίος εκφράζεται σε msec.

```
System.out.println(String.format("%7d times with median of : %fmsec",
        totalTimes, ((float) totalTime*1.0e-6 / (float) totalTimes)));
```

Στην μεταβλητή sumFreeMemMB προστίθενται κάθε φορά οι τιμές ελεύθερης μνήμης που προκύπτουν από τα logs με αναγνωριστικό FREE-MEMORY όπου και μετατρέπονται σε MB. Η διαίρεση της με την τιμή memTimes όπου αφορά στο πόσα logs τιμών μνήμης διαβάστηκαν, μας δίνει το μέσο όρο ελεύθερης μνήμης του συστήματος κάθε φορά. Το υπόλοιπο που προκύπτει από το μέσο όρο ελεύθερης μνήμης και τη συνολική μνήμη του συστήματος (μεταβλητή mem), αφορά στη μέση χρησιμοποιούμενη μνήμη για την εκτέλεση της προσομοίωσης. Τέλος το υπόλοιπο που πρκύπτει από την ελάχιστη ελεύθερη μνήμη που διαβάστηκε και τη συνολική μνήμη του συστήματος, αφορά στο μέγιστο pick μνήμης που χρησιμοποιήθηκε στιγμιαία κατά την εκτέλεση της προσομοίωσης.

```
123             }
124             else if(tokens[2].equals("FREE-MEMORY"))
125             {
126                 try {
127                     memTimes++;
128                     long token = Long.valueOf(tokens[0]);
129                     sumFreeMemMB += token/MB;
130                     //for minimum free memory
131                     if(token < minFreeMem && Math.abs(token - minFreeMemPrev) < 1000000)
132                     {
133                         minFreeMem = token;
134                     }
135                     minFreeMemPrev = token;
```

```
System.out.println(String.format("%7d times with average of : %fmsec",
        totalTimes, ((float) totalTime*1.0e-6 / (float) totalTimes)));


System.out.println(String.format("client time :                  %fsec",
        threadTimeSec));
System.out.println(String.format("function time :                %fsec",
        (double)funcTimeSec));
System.out.println(String.format("The whole process lasts :      %fsec",
        simulationTime));


 System.out.println(String.format("-----------------------------------------"));
 System.out.println(String.format("                CPU Uses"));
 double cpu = (threadTimeSec + funcTimeSec)/simulationTime;
 double clientCpu = threadTimeSec/simulationTime;
 double funcCpu = funcTimeSec/simulationTime;

 System.out.println("Total cpu uses     -> " + cpu*100 + "%");
 System.out.println("Client cpu uses    -> " + clientCpu*100 + "%");
 System.out.println("Functions cpu uses -> " + funcCpu*100 + "%" );
 System.out.println(String.format("-----------------------------------------"));

System.out.println(String.format("                MEMORY Uses"));
System.out.println("Total System Memory-> " + mem + "MB");
System.out.println("Average used Memory-> " + (mem - sumFreeMemMB/memTimes) + "MB

 System.out.println("Maximum used Memory-> " + (mem-minFreeMem/MB) + "MB

System.out.println(String.format("-----------------------------------------"));
System.out.println(String.format("          Server-side analysis"));
System.out.println("From all functions:");
System.out.println("Registration takes  " + (double)sumRegister/(double)funcTime*100 + "% of time");
System.out.println("Update takes        " + (double)sumUpdate/(double)funcTime*100 + "% of time");
System.out.println("cargoQuantity takes " + (double)sumInfo/(double)funcTime*100 + "% of time");
```

Παραπάνω απεικονίζονται συνοπτικά τα αποτελέσματα που προκύπτουν από την εκτέλεση της προσομοίωσης :

“times with average of ” : ο αριθμός των αιτημάτων που διαβιβάστηκαν μεταξύ των οντοτήτων και ο μέσος χρόνος που χρειάστηκε

“client time”: ο χρόνος κατά τον οποίο καταλαμβάνεται χρήση CPU για την εκτέλεση των threads πελατών

“function time”: ο χρόνος κατά τον οποίο καταλαμβάνεται χρήση CPU για την εκτέλεση των συναρτήσεων καταχώρησης, ανανέωσης και ερώτησης ποσότητας προϊόντος

“the whole process lasts”: ο χρόνος που διήρκησε η εκτέλεσης όλης της προσομοίωσης

“Total cpu uses” : το ποσοστό της CPU που χρησιμοποιείται για την εκτέλεσή της προσομοίωσης. Προκύπτει από το άθροισμα των χρόνων χρήσης CPU για την

εκτέλεση πελατών και συναρτήσεων δια τον συνολικό χρόνο που χρειάστηκε για να τρέξει όλη η προσομοίωση

“Client cpu uses”: το ποσοστό χρήσης CPU για την εκτέλεση των threads πελατών. Προκύπτει από την διαίρεση του χρόνου χρήσης CPU για την εκτέλεση πελατών δια το συνολικό χρόνο εκτέλεσης της προσομοίωσης

“Function cpu uses”: το ποσοστό χρήσης CPU για την εκτέλεση των συναρτήσεων. Προκύπτει από την διαίρεση του χρόνου χρήσης CPU για την εκτέλεση των συναρτήσεων δια το συνολικό χρόνο εκτέλεσης της προσομοίωσης

“Total System Memory”: Η συνολική μνήμη του συστήματος στο οποίο εκτλείται η προσομοίωση. Προκύπτει από την τιμή που ανατίθεται στην μεταβλητή mem.

“Average used Memory”: Η μέση τιμή της μνήμης που χρησιμοποιήθηκε για την εκτέλεση της προσομοίωσης

“Maximum used Memory”: Η μέγιστη τιμή μνήμης που χρησιμοποιήθηκε για την εκτέλεση της προσομοίωσης

“Registration takes”: το ποσοστό του χρόνου που χρειάζεται για registration σε σχέση με τον συνολικό χρόνο κατά τον οποίο καταλαμβάνεται CPU για την εκτέλεση των συναρτήσεων. Προκύπτει από τη διαίρεση του αθροίσματος των καταγεγραμμένων χρόνων για την εκτέλεση των συναρτήσεων register δια το χρόνο function time

“Update takes”: το ποσοστό του χρόνου που χρειάζεται για update σε σχέση με τον συνολικό χρόνο κατά τον οποίο καταλαμβάνεται CPU για την εκτέλεση των συναρτήσεων. Προκύπτει από τη διαίρεση του αθροίσματος των καταγεγραμμένων χρόνων για την εκτέλεση των συναρτήσεων update δια το χρόνο function time

“CargoQuantity takes”: το ποσοστό του χρόνου που χρειάζεται για την ερώτηση ποσότητας προϊόντος σε σχέση με τον συνολικό χρόνο κατά τον οποίο καταλαμβάνεται CPU για την εκτέλεση των συναρτήσεων. Προκύπτει από τη διαίρεση του αθροίσματος των καταγεγραμμένων χρόνων για την εκτέλεση των συναρτήσεων ερώτησης ποσότητας προϊόντος δια το χρόνο function time

5.5 Μετρήσεις

Όπως αναφέρθηκε παραπάνω, η κλάση CountMessageTime διαβάζει το αρχείο των logs και αναλαμβάνει να υπολογίσει τον αριθμό των αιτημάτων που διάβασε καθώς επίσης τον μέσο χρόνο της διαφοράς μεταξύ αποστολής και λήψης τους. Η διαφορά αυτή, αφορά πιο συγκεκριμένα στο χρόνο όπου χρειάζεται το μήνυμα να δημιουργηθεί και να κωδικοποιηθεί, να περάσει στο δίκτυο, να φτάσει στο αντίστοιχο Web Service και εκεί να παραδοθεί στην κλάση ώστε να ξεκινήσει η επεξεργασία.

Στην κλάση CountMessageTime επιπρόσθετα υπολογίζεται ο χρόνος κατά τον οποίο καταλαμβάνεται χρήση υπολογιστικής ισχύος για την εκτέλεση των threads πελατών και για την εκτέλεση των συναρτήσεων καταχώρησης, ανανέωσης και ερώτησης ποσότητας προϊόντος. Τέλος υπολογίζεται η συνολική διάρκεια εκτέλεσης της προσομοίωσης, το ποσοστό χρήσης CPU που καταλαμβάνεται για την εκτέλεση της προσομοίωσης, το ποσοστό χρήσης CPU για την εκτέλεση threads πελατών και function time ξεχωριστά καθώς επίσης η μέση και μέγιστη τιμή της χρησιμοποιούμενης μνήμης του συστήματος στο οποίο η προσομοίωση εκτελείται.

Η παρούσα υλοποίηση έχει ως σκοπό την αξιολόγηση της συμπεριφοράς της υποδομής των Web Services ως αναφορά την κλιμακωσιμότητα (scalability) για την ανάπτυξη κατανεμημένων εφαρμογών. Έτσι, για την αξιολόγηση της επίδοσης του συστήματος κατά την κλιμάκωση των απαιτήσεων που υπάρχουν, παρέχεται η δυνατότητα μεταβολής του αριθμού των threads πελατών στον οποίο αντιστοιχεί η μεταβλητή number of clients της κλάσης RegionalClients. Θα πρέπει να αναφερθεί επίσης πως η αύξηση κατά ένα client αντιστοιχεί σε αύξηση των αιτημάτων ενεργειών κατά 100, κάτι το οποίο καθορίζεται από την μεταβλητή NUMBER OF ACTIONS της κλάσης RegionalClientThreadFather όπου στην συγκεκριμένη περίπτωση έχει τεθεί σε 100 και είναι επίσης παραμετροποιήσιμη.

Με τον τρόπο αυτό έχουμε αρχικά τη δυνατότητα να αξιολογήσουμε το πόσο γρήγορα μεταδίδονται τα αιτήματα για επεξεργασία όταν ο αριθμός των threads πελατών αυξάνει και κατ επέκταση αυξάνει και η απαίτηση στο σύστημα. Παρακάτω ακολουθεί ένας πίνακας βάσει των μετρήσεων όπου έχουν ληφθεί για διαφορετικό αριθμό threads πελατών κάθε φορά και αφορούν στον αριθμό των αιτημάτων, τον μέσο χρόνο αποστολής και λήψης τους ενώ επίσης παρατίθεται και το ποσοστό χρησιμοποιούμενης υπολογιστικής ισχύος και μνήμης του συστήματος για κάθε μέτρηση. Συγκεκριμένα λαμβάνονται μετρήσεις για 1000 έως και 15000 threads πελατών με ένα αυξητικό ρυθμό του πλήθους τους κατά βήμα 1000. Οι μετρήσεις που αφορούν σε όλες τις παραμέτρους αξιολογησης της προσομοίωσης έχουν πραγματοποιηθεί σε σύστημα χαρακτηριστικών INTEL CORE I7-3770 4.1GHZ

(8 πυρήνες, Overclocked από 3,9 στα 4,1GHZ) και μνήμη RAM 8GB DDR3 (Overclocked στα 1866MHZ), Λειτουργικό σύστημα: 64-bit Windows 7 professional.

Number Of Clients	Αριθμός αιτημάτων	Μέσος χρόνος αιτημάτων (ms)	CPU USES %	Average Memory Used %
1000	65634	350,277972	50,0385387	44,74786860
2000	102230	435,957159	64,0366282	49,13782404
3000	134724	512,08587	76,6852096	63,07124473
4000	167098	586,069718	89,0524749	66,27977307
5000	198034	678,786223	98,2603257	73,79510706
6000	225425	772,776015	99,7600095	78,75645937
7000	262135	868,377693	99,0915012	81,37118184
8000	301704	1177,235684	88,1136832	83,57925147
9000	340498	1515,751587	77,4499156	84,11523288
10000	386392	1931,997262	68,6068480	84,82709004
11000	432252	2390,795009	57,9643480	85,74646129
12000	476384	3187,548457	38,8459026	87,55769724
13000	512684	5779,804282	18,9964276	88,62512374
14000	547180	7411,109142	13,8014792	90,69248711
15000	598167	9862,551873	8,793690347	91,00691352

Ο πίνακας και τα αντίστοιχα γραφήματα προσδίδουν την αύξηση της χρονικής καθυστέρησης κατά τη διαδικασία αποστολής και λήψης των μηνυμάτων για μεγάλο πλήθος πελατών και ιδιαίτερα στις περιπτώσεις όπου η μνήμη τείνει στον κορεσμό της. Έτσι, σε αντιστοιχία με την γραμμική αύξηση των αιτημάτων των πελατών, ο μέσος χρόνος για την αποστολή και παραλαβή τους, αρχικά εμφανίζει ένα ρυθμό αύξησης με γραμμική συμπεριφορά, έως το πλήθος των 4000 Number Of Clients. Έως το σημείο αυτό, η τιμή του ποσοστού χρήσης της CPU τείνει να είναι ανοδική όπως και η χρησιμοποιούμενη μνήμη του συστήματος όπου έχει επέλθει σε ποσοστό 73%.

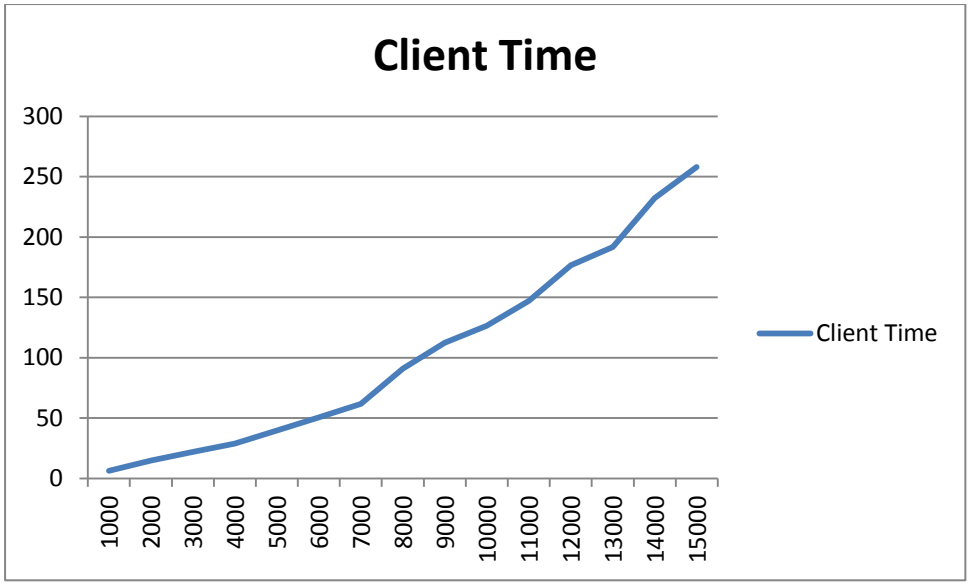
Εν συνέχεια, για πλήθος 5000 Number Of Clients ο χρόνος αποστολής – λήψης παρατηρείται να αποκτά ένα μεγαλύτερο ρυθμό αύξησης όπου ωστόσο δείχνει να διατηρεί τη γραμμική του συμπεριφορά. Το ποσοστό χρησιμοποιούμενης μνήμης επίσης διατηρεί μια ανοδική πορεία ενώ η υπολογιστική ισχύς του συστήματος πλέον χρησιμοποιείται εξ ολοκλήρου. Η συγκεκριμένη αύξηση καταδεικνύει την συμπεριφορά του συστήματος σε συνθήκες πίεσης, όπου αποτελεί και μια τυπική περίπτωση πραγματικής λειτουργίας των υπηρεσιοστρεφών κατανεμημένων συστημάτων.

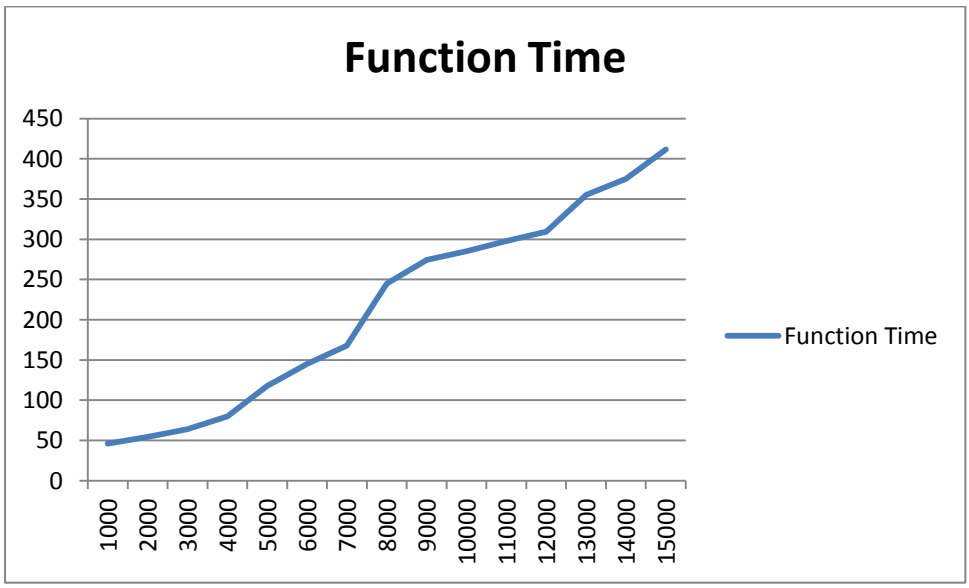
Ιδαίτερο ενδιαφέρον παρουσιάζουν οι μετρήσεις που αφορούν στο πλήθος 8000 έως και 15000 πελατών. Η χρήση υπολογιστικής ισχύος δείχνει να ακολουθεί μια καθοδική πορεία οδηγούμενη σε πολύ μικρά ποσοστά χρήσης της, η χρησιμοποιούμενη μνήμη όπου έχει επέλθει σε αρκετά υψηλό ποσοστό τείνει σε σημείο κορεσμού, ενώ ο μέσος χρόνος για τη διαδικασία αποστολής – λήψης των αιτημάτων αυξάνεται πλέον με πολύ μεγαλύτερο ρυθμό και με εμφανής την εξάλειψη της έως τώρα γραμμικής του συμπεριφοράς. Η συγκεκριμένη κατάσταση, απεικονίζει τις περιπτώσεις όπου ένα σύστημα καθίσταται πλέον μη λειτουργικό και

σε συνθήκες thrashing. Η αύξηση του πλήθους πελατών και κατ΄ επέκταση του φόρτου στα εν λόγω επίπεδα, έχει ως αποτέλεσμα την χρήση πολύ μεγάλου ποσοστού μνήμης έως και κορεσμού της. Έτσι το σύστημα οδηγείται σε μεγαλύτερη συχνότητα page faults και το ποσοστό CPU που διατίθεται για την εκτέλεση της προσομοίωσης είναι πολύ μικρό καθότι κατά το μεγαλύτερο μεγαλύτερο μέρος του αναλώνεται στην εξυπηρέτηση των page faults και την εναλλαγή σελίδων μεταξύ φυσικής και εικονικής μνήμης.

Εν συνεχεία ακολουθούν ο πίνακας και η γραφική παράσταση όπου αφορούν στις μετρήσεις που έχουν ληφθεί ως προς τον χρόνο όπου καταλαμβάνουν χρήση CPU, η εκτέλεση των threads πελατών και των συναρτήσεων καταχώρησης, ανανέωσης και ερώτησης ποσότητας προϊόντος. Οι χρόνοι είναι εκφρασμένοι σε seconds.

Number Of Clients	Client Time (sec)	Function Time (sec)	CPU USES %	Average Memory Used %
1000	6,175809	45,917187	50,0385387	44,74786860
2000	14,851045	54,367043	64,0366282	49,13782404
3000	21,975622	63,960512	76,6852096	63,07124473
4000	28,838933	79,980982	89,0524749	66,27977307
5000	39,822036	117,769312	98,2603257	73,79510706
6000	50,655304	144,948165	99,7600095	78,75645937
7000	61,58779	167,718341	99,0915012	81,37118184
8000	90,931999	244,830524	88,1136832	83,57925147
9000	112,505018	274,476429	77,4499156	84,11523288
10000	126,35582	285,282909	68,6068480	84,82709004
11000	147,064718	298,071062	57,9643480	85,74646129
12000	176,692648	309,526425	38,8459026	87,55769724
13000	191,805545	355,13082	18,9964276	88,62512374
14000	232,295062	375,018083	13,8014792	90,69248711
15000	258,030775	411,480148	8,793690347	91,00691352



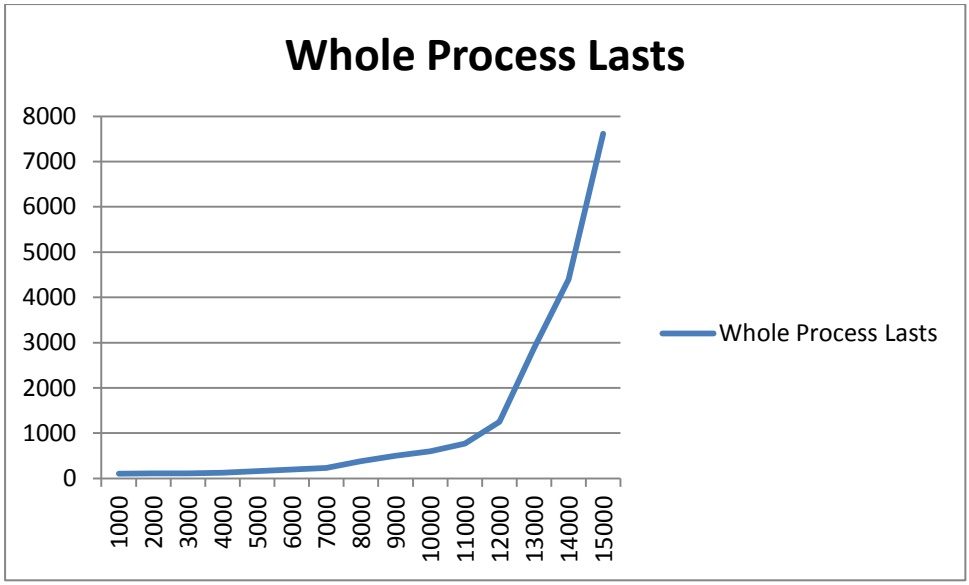


Τόσο για τον χρόνο εκτέλεσης των threads πελατών όσο και των συναρτήσεων θα πρέπει να διευκρινιστεί πως αφορούν στον χρόνο κατά τον οποίο καταλαμβάνεται χρήση CPU αποκλειστικά για την εκτέλεση των εν λόγω διεργασιών. Ο χρόνος εκτέλεσης πελατών για πλήθος πελατών 1000 – 4000 και όσο το ποσοστό χρησιμοποιούμενης μνήμης και υπολογιστικής ισχύς του συστήματος διατηρούν μια ανοδική πορεία, παρουσιάζει μια γραμμική συμπεριφορά με ένα σταθερό ρυθμό αύξησης. Σε συνθήκες φόρτου και πλήθος πελατών 5000 - 7000 όπου η υπολογιστική ισχύς χρησιμοποιείται εξ' ολοκλήρου, παρατηρείται πως η γραμμική συμπεριφορά διατηρείται, ωστόσο ο ρυθμός αύξησης μετατίθεται σε μεγαλύτερα επίπεδα. Η γραμμικότητα αυτή εξαλείφεται για πλήθος πελατών 8000 – 15000 όπου το σύστημα καθίσταται μη λειτουργικό και ο χρόνος κατά τον οποίο

καταλαμβάνεται χρήση υπολογιστικής ισχύος για την εκτέλεση threads πελατών, αυξάνει σε μεγαλύτερα ποσοστά λόγω του μικρού διαθέσιμου ποσοστού CPU για την εξυπηρέτηση της συγκεκριμένων διεργασιών. Η γραφική παράσταση του χρόνου εκτέλεσης των συναρτήσεων, προσδίδει επίσης μια αυξητική συμπεριφορά για όλη τη διάρκεια εκτέλεσης της προσομοίωσης η οποία ωστόσο δεν ακολουθεί απαραίτητα μια γραμμικότητα κάτι το οποίο οφείλεται κατά κύριο λόγο στην πολυπλοκότητα των συναρτήσεων που εκτελούνται. Αυτό σημαίνει πως οι συναρτήσεις που εκτελούνται, εκτελούν άλλες συναρτήσεις που δεν παρουσιάζουν γραμμικό χρόνο αύξησης, όπως για παράδειγμα η συνάρτηση get() του Hashmap.

Παρακάτω παρατίθενται οι μετρήσεις που αφορούν στον συνολικό χρόνο που απαιτείται για την εκτέλεση της προσομοίωσης, από την έναρξή της έως και την λήξη της διεργασίας και του τελευταίου αιτήματος. Ο χρόνος αυτός εκφρασμένος σε seconds

Number Of Clients	Whole Process Lasts (sec)	CPU USES %	Average Memory Used %
1000	104,105750	50,0385387	44,74786860
2000	108,091401	64,0366282	49,13782404
3000	112,063502	76,6852096	63,07124473
4000	122,197519	89,0524749	66,27977307
5000	160,381463	98,2603257	73,79510706
6000	196,0740280	99,7600095	78,75645937
7000	231,4084741	99,0915012	81,37118184
8000	381,0560523	88,1136832	83,57925147
9000	499,6538006	77,4499156	84,11523288
10000	599,9965611	68,6068480	84,82709004
11000	767,9475312	57,9643480	85,74646129
12000	1251,661154	38,8459026	87,55769724
13000	2879,153777	18,9964276	88,62512374
14000	4400,348196	13,8014792	90,69248711
15000	7613,537623	8,7936903	91,00691352

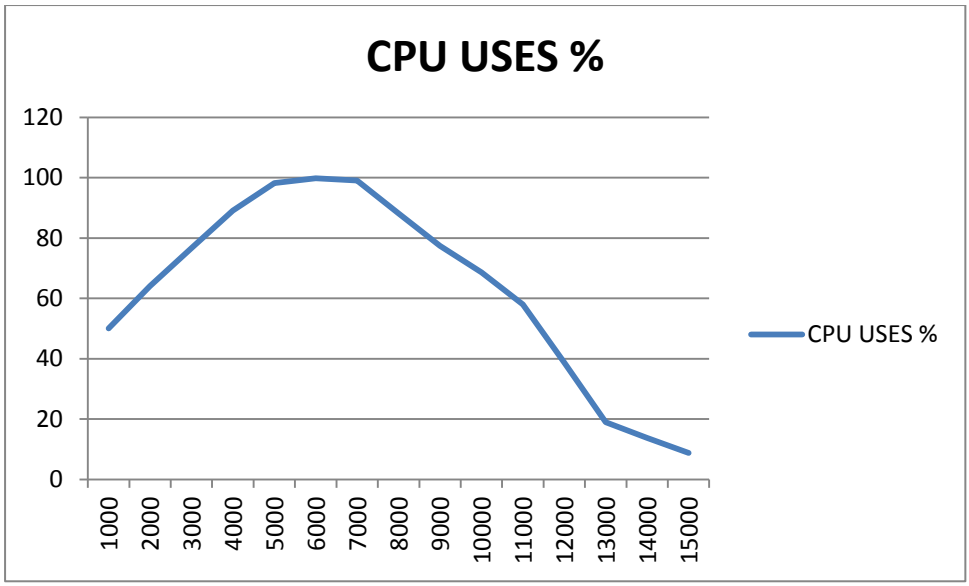


Ακολουθείται και εδώ μια αυξητική συμπεριφορά η οποία δεν θα μπορούσε να χαρακτηριστεί ως γραμμική για κάποιο από τα στάδια εκτέλεσης της προσομοίωσης. Ο λόγος είναι πως συμπεριλαμβάνει παραμέτρους όπως το function time καθώς και άλλες διεργασίες του συστήματος στο οποίο εκτελέστηκε η προσομοίωση όπου πιθανό να μεσολαβούσαν σταδιακά. Πιο συγκεκριμένα, για πλήθος πελατών 1000 έως 4000 το ποσοστό αύξησης είναι μικρότερο σε σύγκριση με τις συνθήκες φόρτου του συστήματος όπου η υπολογιστική ισχύς του συστήματος χρησιμοποιείται εξ ολοκλήρου.

Ιδιαίτερο ενδιαφέρον επίσης παρουσιάζει η απότομη αύξηση του χρόνου για τις περιπτώσεις όπου η φυσική μνήμη τείνει προς τον κορεσμό της και το ποσοστό υπολογιστικής ισχύος που χρησιμοποιείται για τις διεργασίες της προσομοίωσης είναι πλεόν πολύ μικρό. Με την μνήμη να τείνει στον κορεσμό της όπως αναφέρθηκε και παραπάνω, το σύστημα οδηγείται σε μεγαλύτερη συχνότητα page faults και το ποσοστό CPU κατά το μεγαλύτερο μέρος του αναλώνεται στην εξυπηρέτηση των page faults και την εναλαγγή σελίδων μεταξύ φυσικής και εικονικής μνήμης.

Εδώ θα πρέπει να αναφερθεί ως παράδειγμα πως σε μέτρηση που πραγματοποιήθηκε σε σύστημα 8GB RAM, για αριθμό 15000 Number Of Clients, η φυσική μνήμη βρισκόταν σχεδόν εξολοκλήρου σε χρήση ενώ επίσης το ποσοστό χρήσης της CPU ήταν πολύ χαμηλό και η συνολική διάρκεια εκτέλεσης της προσομοίωσης κράτησε περίπου 120 λεπτά.

Τέλος παρατίθεται το γράφημα όπου απεικονίζει τη χρήση CPU κατά την κλιμάκωση της απαίτησης σε ένα κατανεμημένο περιβάλλον υποδομής Web Services. Το ποσοστό χρήσης CPU, προκύπτει κάθε φορά από το άθροισμα των χρόνων χρήσης CPU για την εκτέλεση των threads πελάτων και των συναρτήσεων, δια τον συνολικό χρόνο όπου χρειάστηκε για την εκτέλεση της προσομοίωσης. Διακρίνονται εμφανώς, η αύξηση του ποσοστού χρήσης CPU για πλήθος πελατών 1000 έως 4000, η εξ΄ολοκλήρου χρήση της υπολογιστικής ισχύος κατά τις συνθήκες φόρτου για πλήθος πελατών 5000 έως 7000 και τέλος η συνεχής μείωση της χρήσης CPU για τις συνθήκες thrashing του συστήματος από 8000 έως και 15000 πελάτες.



5.6 Αξιολόγηση

Σε μια σύνοψη όλων των παραπάνω και ως αναφορά τα αποτελέσματα που έχουν προκύψει σχετικά με την αξιολόγηση της κλιμακωσιμότητας της υποδομής Web Services για την ανάπτυξη κατανεμημένων εφαρμογών, παρατηρήθηκαν τρεις βασικές περιπτώσεις από άποψη λειτουργικότητας και επίδοσης ενός συστήματος.

Στην πρώτη περίπτωση κατά την οποία η υπολογιστική ισχύς αλλά και η μνήμη του συστήματος δεν έχουν φτάσει σε κορεσμό, ένα σύστημα αποδίδει στο μέγιστο δυνατό, με τα ποσοστά των χρόνων εκτέλεσης των επιμέρους διεργασιών να διατηρούνται σε χαμηλά επίπεδα ενώ επίσης αποδίδεται μια γραμμική συμπεριφορά για παραμέτρους όπου δεν παρεμβάλλονται μη γραμμικές συνιστώσες.

Η δεύτερη περίπτωση αφορά στην εξολοκλήρου χρήση της υπολογιστικής ισχύος και είναι αυτή όπου καταδεικνύει την συμπεριφορά του συστήματος σε συνθήκες πίεσης, καθώς επίσης αποτελεί μια τυπική περίσταση πραγματικής λειτουργίας των υπηρεσιοστρεφών κατανεμημένων συστημάτων. Όπως παρατηρήθηκε στην περίπτωση αυτή, σε ένα σύστημα υποδομής Web Services οι χρόνοι εκτέλεσης των επιμέρους διεργασιών αυξάνονται κατά ένα μεγαλύτερο ποσοστό σε σχέση με τις συνθήκες όπου δεν υφίστανται υψηλά επίπεδα φόρτου, ωστόσο διατηρείται μια γραμμική συμπεριφορά σε παραμέτρους όπου δεν παρεμβάλονται μη γραμμικές συνιστώσες.

Η τρίτη περίπτωση αφορά στην κατάσταση όπου ένα σύστημα καθίσταται πλέον μη λειτουργικό και σε συνθήκες thrashing. Η χρησιμοποιούμενη μνήμη έχει επέλθει σε αρκετά υψηλό ποσοστό και τείνει σε σημείο κορεσμού λόγω της μεγάλης αύξηση του πλήθους πελατών και κατ΄ επέκταση του φόρτου. Αυτό έχει ως αποτέλεσμα το σύστημα να οδηγείται σε μεγαλύτερη συχνότητα page faults και το ποσοστό CPU που διατίθεται για την εκτέλεση της προσομοίωσης είναι πολύ μικρό καθότι κατά το μεγαλύτερο μέρος του αναλώνεται στην εξυπηρέτηση των page faults και την εναλλαγή σελίδων μεταξύ φυσικής και εικονικής μνήμης. Έτσι οι χρόνοι εκτέλεσης των επιμέρους διεργασιών αυξάνονται κατά πολύ περισσότερο καθώς επίσης εξαλείφεται η γραμμική συμπεριφορά του συστήματος.

Η διαπίστωση αυτή, επισημαίνει πως πέραν της μείωσης των χρόνων εκτέλεσης και του ποσοστού χρήσης CPU σύμφωνα με τις δυνατότητες της υπολογιστικής ισχύος του εκάστοτε συστήματος, η αύξησης της μνήμης σε ένα κατανεμημένο σύστημα προσδίδει και την αύξηση του πλήθους πελατών όπου είναι δυνατό να εξυπηρετηθούν.

Παράρτημα

Βιβλιογραφία

1. UPnP Forum, “UPnP device architecture 1.0”, October 2008, http://www.upnp.org
2. Brent Miller, Toby Nixon, Charlie Tai, Mark Wood. Home Networking with Universal Plug and Play, IEEE Communications Magazine, December 2001
3. http://en.wikipedia.org/wiki/Universal_Plug_and_Play
4. Introduction to Web Services Technologies: SOA, SOAP, WSDL and UDDI by Thomas Erl, year 2004
5. http://www.osgi.org/Technology/WhatIsOSGi
6. Understanding to Web Services. XML, WSDL, SOAP, and UDDI, by Eric Newcomer, Addisonn Wesley, 2002
7. http://wiki.oxygenelanguage.com/en/Web_Services_Protocol_Stack
8. Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, 1st edition, book by Ethan Cerami, publisher O' Reilly 2002
9. UDDI Version 3.0.2, UDDI Spec Technical Committee Draft, Dated 20041019 http://uddi.org/pubs/uddi-v3.0.2-20041019.htm
10. R-OSGi: Distributed Applications through Software Modularization, Jan S. Rellermeyer, Gustavo Alonso and Timothy Roscoe, Department of Computer Science, ETH Zurich
11. http://searchnetworking.techtarget.com/definition/OSGi
12. Design of Service Discovery & Provisioning Architecture based on theOpen Services Gateway Initiative Vincenzo Suraci, Massimiliano Taglien, University of Rome “La Sapienza”, D.I.S.
13. http://en.wikipedia.org/wiki/OSGi

14. http://r-osgi.sourceforge.net/mindstorms.html

15. http://wiki.eclipse.org/EIG:OSGi_Remote_Services

16. Services Everywhere: OSGi in Distributed Enviroments, Jan S.Rellemeyer, Gustavo Alonso, Department of Computer Science ETH Zurich

17. http://www.knopflerfish.org/osgi_service_tutorial.html

18. http://docs.jboss.org/osgi/jboss-osgi-1.0.0.Beta9/userguide/html/ChapIntroduction.html

19. About the OSGi Service Platform Technical Whitepaper, 7 June 2007 http://www.osgi.org/wiki/uploads/Links/OSGiTechnicalWhitePaper.pdf

20. OSGi Service Platform, Service Compendium Release 4, Version 4.2 March 10, 2009 http://www.osgi.org/download/r4-v4.2-cmpn-draft-20090310.pdf

21. http://www.vogella.com/articles/OSGiServices/article.html

22. http://gravity.sourceforge.net/servicebinder/osginutshell.html