ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Internet of Things με το πρωτόκολλο XMPP

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Ν. Πικριδάς

Επιβλέπων : Ευστάθιος Δ. Συκάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Internet of Things με το πρωτόκολλο XMPP

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Ν. Πικριδάς

Επιβλέπων : Ευστάθιος Δ. Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .

.....	.....	.....
Ευστάθιος Συκάς	Γεώργιος Στασινόπουλος	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρη Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2018

.....

Αναστάσιος Ν. Πικριδάς

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το Internet of Things αποτελεί ξεχωριστό επιστημονικό κλάδο με μεγάλο ενδιαφέρον τη σύγχρονη εποχή. Υπάρχει πληθώρα τεχνολογιών για την υποστήριξη του και έτσι έχουν ήδη εμφανιστεί οι πρώτες εμπορικές εφαρμογές. Ωστόσο, η πορεία ανάπτυξής του είναι σχετικά αργή. Με βάση τα παραπάνω, η εργασία αυτή προλογίζεται με την ανάπτυξη του Internet of Things ως όραμα αλλά και ως επιστημονικό κλάδο, καταλήγοντας στις σύγχρονες προκλήσεις που αντιμετωπίζει. Βασικές προκλήσεις αποτελούν η χρήση συσκευών περιορισμένων δυνατοτήτων ως Things καθώς και η έλλειψη αποδοτικών τεχνικών Provisioning. Σκοπός της παρούσας εργασίας είναι η ανάπτυξη προσέγγισης για τη δημιουργία δικτύων Internet of Things, τα οποία ανταποκρίνονται στις παραπάνω προκλήσεις, με χρήση του πρωτοκόλλου XMPP. Το XMPP αποτελώντας δικτυακό πρωτόκολλο του στρώματος εφαρμογής που αναπτύχθηκε για Instant Messaging, προσφέρει πληθώρα προτυποποιημένων επεκτάσεων που το καθιστούν ικανό για χρήση και σε άλλες εφαρμογές. Χρησιμοποιώντας τις επεκτάσεις αυτές ως δομικά στοιχεία αναπτύχθηκε η προσέγγιση, βασικά σημεία της οποίας είναι η ομαδοποίηση των Things σε Multi User Chat rooms και η εισαγωγή της έννοιας του Provisioning Server, ο οποίος αποτελεί κεντρική οντότητα που διαχειρίζεται τον έλεγχο πρόσβασης στα rooms. Με βάση την προσέγγιση αυτή, αναπτύχθηκαν τρεις υλοποιήσεις. Η πρώτη αφορά βιβλιοθήκη σε γλώσσα Java για τη δημιουργία δικτύων Internet of Things, με χρήση της βιβλιοθήκης Smack. Η δεύτερη αποτελεί διαδικτυακή πλατφόρμα/panel για τη διαχείριση του Provisioning Server. Η τρίτη υλοποίηση αφορά desktop εφαρμογή για τη δημιουργία δοκιμαστικών δικτύων, με χρήση της βιβλιοθήκης που αναπτύχθηκε.

Λέξεις Κλειδιά :

Internet of Things, Publish Subscribe, Έλεγχος Πρόσβασης, Provisioning, XMPP, Multi User Chat, Smack, Java, Δεδομένα Αισθητήρων, Εντολές Ελέγχου, PHP

Abstract

Internet of Things (IoT) is considered to be a unique scientific discipline with great interest these days. Plenty of existing technologies can be used, thus the first commercial applications have already appeared. However, its course of development is relatively slow. Based on the above, the prologue of this thesis focuses on the development of Internet of Things as a vision and as a scientific field, concluding with the modern challenges it faces. Key challenges include the use of limited-capability devices for Things, as well as the lack of efficient Provisioning techniques. The purpose of this thesis is to develop an approach for creating IoT networks that respond to the above challenges, using the XMPP protocol. XMPP, which is an application layer protocol, developed for Instant Messaging, offers a variety of standard extensions that make it capable of use in other applications as well. Using these extensions as building blocks, the approach has been developed, the key points of which are the grouping of Things into Multi User Chat rooms and the introduction of the Provisioning Server, which is a central entity that manages room access control. Based on this approach, three implementations have been developed. The first is a Java library for creating IoT networks using the Smack library. The second is an online panel for Provisioning Server management. The third implementation concerns a desktop application which can be used to build test networks, using the library that was developed.

Keywords :

Internet of Things, Publish Subscribe, Access Control, Provisioning, XMPP, Multi User Chat, Smack, Java, Sensor Data, Control Commands, PHP

Ευχαριστίες

Θα ήθελα να εκφράσω τις βαθύτατες ευχαριστίες μου στον επιβλέποντα καθηγητή μου κύριο Ευστάθιο Συκά, που μου προσέφερε την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, καθώς και στον καθοδηγητή μου ερευνητή κύριο Δημήτρη Καλογερά, ο οποίος με βοήθησε να διαμορφώσω την άποψη μου για τα σύγχρονα ζητήματα του IoT. Η αμέριστη βοήθεια και κατανόηση τους αποτέλεσε αρωγό τόσο στην εκπόνηση αυτής της εργασίας, όσο και στη μετέπειτα εξέλιξη μου. Επιπρόσθετα, θα ήθελα να ευχαριστήσω κάθε άτομο που στάθηκε δίπλα μου καθ' όλη τη διάρκεια των σπουδών μου και ιδιαίτερα τους γονείς μου χωρίς τον μόχθο των οποίων δεν θα ήταν δυνατή η ολοκλήρωση των σπουδών μου.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Εισαγωγή

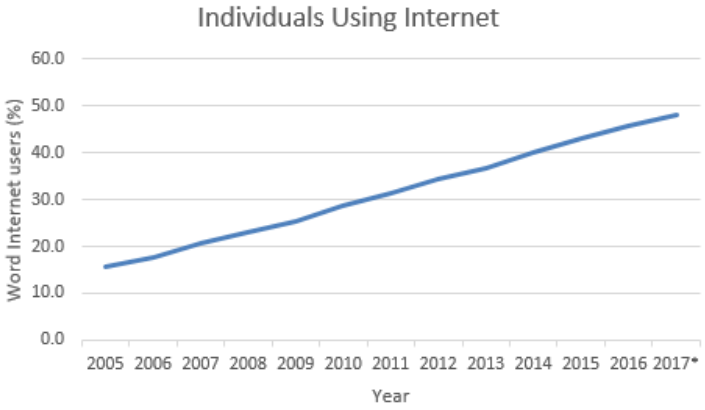
1.1 Οι διασυνδεδεμένες τεχνολογίες στην ιστορία

Ο όρος διασυνδεδεμένες τεχνολογίες, αναφέρεται στις τεχνολογίες που εμπεριέχουν την έννοια της επικοινωνίας, όχι απαραίτητα μεταξύ ανθρώπων αλλά και μηχανών. Η σύγχρονη εικόνα για αυτές τις τεχνολογίες αποτελείται από τη διασύνδεση υπολογιστικών μηχανών μέσω των τηλεπικοινωνιακών συστημάτων.

Η παραπάνω εικόνα δεν ήταν πάντα έτσι. Χαρακτηριστικό όλων των τεχνολογιών είναι η εξέλιξη. Έτσι και οι διασυνδεδεμένες τεχνολογίες, προέκυψαν ως αποτέλεσμα μιας εξελικτικής διαδικασίας, στην οποία συντέλεσε ένα δίκτυο διαφορετικών παραγόντων, με ξεχωριστούς σκοπούς ο καθένας. Τέτοιοι παράγοντες αποτελούν επιστημονικοί, εμπορικοί, πολιτικοί και κοινωνικοί κύκλοι.

1.2 Η ανάπτυξη του διαδικτύου

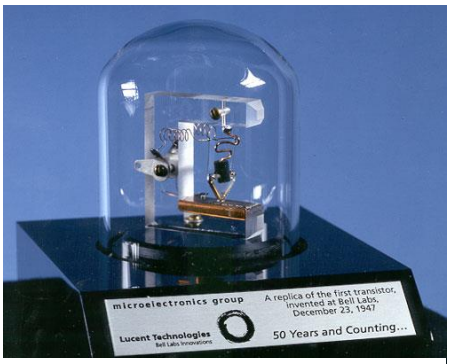
Το διαδίκτυο αποτελεί πλέον την κυρίαρχη διασυνδεδεμένη τεχνολογία. Υπολογίζεται πως το 2017 οι χρήστες του διαδικτύου έφτασαν τα 3,578 δισεκατομμύρια, δηλαδή το 48% του παγκόσμιου πληθυσμού [1].



Σχήμα 1, Παγκόσμιο ποσοστό πληθυσμού που χρησιμοποιεί το διαδύκτιο

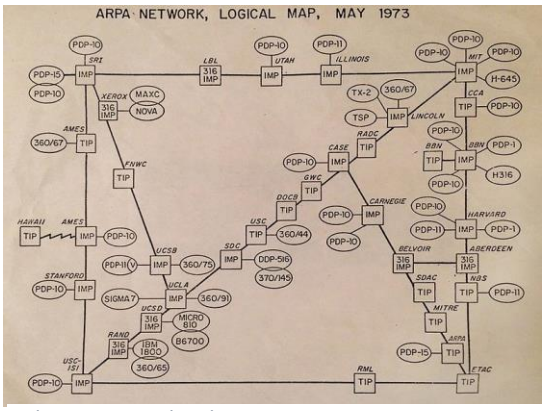
Η ανάπτυξη του διαδικτύου σχετίζεται αμιγώς με την εξέλιξη των τηλεπικοινωνιών και συγκεκριμένα της τηλεφωνίας. Η τηλεφωνία άρχισε να αναπτύσσεται από τα τέλη του 19ου αιώνα και άρχισε να διαδίδεται παγκοσμίως στις αρχές του 20ου [2].

Στα μέσα του 20ου αιώνα, οι αυξημένες ανάγκες της τηλεφωνίας, που είχε πλέον ραγδαία ζήτηση, οδηγούν στην ανακάλυψη και εξέλιξη του τρανζίστορ. Το τρανζίστορ ανοίγει τον δρόμο στην εποχή των ψηφιακών ηλεκτρονικών, τα οποία ενώ εξελίχθηκαν αρχικά μόνο για την υποστήριξη της τηλεφωνίας, οδήγησαν στην ανάπτυξη πολύ σημαντικότερων τεχνολογιών: των επεξεργαστών και κατ' επέκταση των ηλεκτρονικών υπολογιστών [3].



Σχήμα 2, Αντίγραφο του 1ου τρανζίστορ

Στο δεύτερο μισό του 20ου αιώνα, ο ηλεκτρονικός υπολογιστής καθιερώνεται, γίνεται προσωπικός και μπαίνει σε κάθε σπίτι. Παράλληλα, αναπτύσσεται και το διαδίκτυο. Οι πρώτες ιδέες για την επικοινωνία υπολογιστών μέσω ενός δικτύου, εμφανίστηκαν στις αρχές της δεκαετίας του ’60. Το 1967 δημιουργήθηκε η ιδέα του ARPANET, του πρώτου δικτύου υπολογιστών ευρείας έκτασης. Η ανάπτυξή του ξεκίνησε το 1968 και το 1969 στάλθηκε το πρώτο μήνυμα, μεταξύ των δύο πρώτων κόμβων, στο ARPANET: του πανεπιστημίου της Utah και του πανεπιστημίου της California στην Santa Barbara [4]. Στα χρόνια που ακολούθησαν, το ARPANET απέκτησε μεγάλο ενδιαφέρον στα αμερικάνικα πανεπιστήμια, που επιθυμούσαν να ενταχθούν, γέμισε με κόμβους και αποτέλεσε σημείο ανάπτυξης πολλών σημαντικών διαδικτυακών πρωτοκόλλων και τεχνολογιών που χρησιμοποιούμε και σήμερα (όπως το FTP, το Telnet και το e-mail) [5].



Σχήμα 3 – Λογικός Χάρτης του ARPANET, 1973

1.3 Το διαδίκτυο των πραγμάτων (Internet of Things)

1.3.1 Ορισμός

Ο όρος Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) αναφέρεται στο δίκτυο των φυσικών αντικειμένων στα οποία υπάρχει πρόσβαση μέσω του διαδικτύου, όπως ορίζεται από τους αναλυτές και τους οραματιστές της τεχνολογίας. Τα αντικείμενα αυτά περιέχουν ενσωματωμένη τεχνολογία έτσι ώστε να μπορούν να αλληλοεπιδρούν με εσωτερικές καταστάσεις ή το εξωτερικό περιβάλλον. Τέτοια αντικείμενα μπορούν να αποτελούν οικιακές συσκευές, οχήματα, κοινόχρηστες συσκευές/αισθητήρες μίας πόλης/κοινότητας, βιομηχανικές συσκευές, δρόμοι, ρούχα και γενικότερα οποιοδήποτε αντικείμενο του τεχνητού ανθρώπινου περιβάλλοντος.

Το IoT μπορεί να θεωρηθεί ως ιδέα προς την ύψιστη εκμετάλλευση και ανάπτυξη των διασυνδεδεμένων τεχνολογιών και ειδικότερα του διαδικτύου. Αποτελεί την όλο και μεγαλύτερη είσοδο και ύφανση των (μικρο)υπολογιστών και των τηλεπικοινωνιών, στο ανθρώπινο και φυσικό περιβάλλον, σε μεγαλύτερη πυκνότητα. Επίσης, το IoT αποτελεί περισσότερο ιδέα, επιστημονικό κλάδο ανάπτυξης και συνδυασμό τεχνολογιών, παρά αυτοτελή τεχνολογία.

1.3.2 Η ανάπτυξη της ιδέας

Αν και το ενδιαφέρον για στοχευμένη έρευνα πάνω στο Internet of Things προέκυψε μέσα στην τελευταία δεκαετία, μεμονωμένες ιδέες αλλά και γενικότερες βλέψεις, υπάρχουν από το παρελθόν, κατά την ανάπτυξη των τηλεπικοινωνιών και του διαδικτύου. Τέτοιες ιδέες περιέχονται σε φουτουριστικά σενάρια, θεωρητικές επιστημονικές δημοσιεύσεις, αλλά και μεμονωμένες πειραματικές εργασίες, σχετικές με τις δυνατότητες του διαδικτύου.

Το πέρασμα της μεταπολεμικής περιόδου, στις δεκαετίες του 60’ και 70’, συναντά τη ραγδαία ανάπτυξη ψηφιακών ηλεκτρονικών και επεξεργαστών, που οδηγείται από την ψηφιοποίηση της τηλεφωνίας. Επηρεασμένο από αυτές τις εξελίξεις, το φουτουριστικό πνεύμα της εποχής, εμπνέεται και προβλέπει τεχνολογίες του μέλλοντος, όπου πολλές από αυτές εμπίπτουν, σε αυτό που σήμερα αποκαλούμε Internet of Things. Παράδειγμα ενός τέτοιου σεναρίου αποτελεί η ταινία μικρού μήκους της Philco – Ford Corporation, του 1967, με τίτλο *The Home Of The Future : Year 1999 AD*. Στην ταινία παρουσιάζεται το σπίτι του μέλλοντος, ‘’μίας κοινωνίας πλούσιας σε άνεση και ευκολίες που θεωρούνται δεδομένες’’. Ιδιαίτερο χαρακτηριστικό αυτού του σπιτιού αποτελεί ο κεντρικός υπολογιστής, ο οποίος συνδέεται με όλες τις οικιακές συσκευές, αλλά υπάρχει και δυνατότητα απομακρυσμένης επικοινωνίας με αυτόν από άλλο υπολογιστή [6].

Σχήμα 4 – Στιγμιότυπο της ταινίας

Ένα από τα πρώτα πρακτικά πειραματικά παραδείγματα, δημιουργήθηκε την εποχή της ανάπτυξης του διαδικτύου και συγκεκριμένα το 1982, στο Carnegie Mellon University. Στο πανεπιστήμιο υπήρχε ένας αυτόματος πωλητής αναψυκτικού. Ωστόσο, η θέση του πωλητή απείχε αρκετά από πολλά γραφεία και εστίες. Οι φοιτητές του πανεπιστημίου, ήθελαν να γνωρίζουν αν στο μηχάνημα υπάρχουν φιάλες αναψυκτικού, αλλά και πόσο πρόσφατα έχουν τοποθετηθεί, για να γνωρίζουν αν έχουν ψυχθεί αρκετά, πριν μπουν στον κόπο να επισκεφτούν το μηχάνημα. Ιδανική λύση θα αποτελούσε, να έχουν τη δυνατότητα να πληροφορηθούν από τον υπολογιστή τους. Η τεχνολογία για μία τέτοια εφαρμογή υπήρχε, οπότε οι φοιτητές κατάφεραν να συνδέσουν το μηχάνημα στο διαδίκτυο (τότε ARPANET), δημιουργώντας το πρώτο ‘’Internet Coke Machine’’. Αν και η χρήση αυτής της υπηρεσίας αναπτύχθηκε για τα μέλη του πανεπιστημίου, στην πραγματικότητα οποιοσδήποτε υπολογιστής στο διαδίκτυο είχε πρόσβαση [7].

Σχήμα 5 – Το Internet Coke Machine στο Carnegie Mellon University

Αργότερα, στην αρχή της ακμής του διαδικτύου, τη δεκαετία του 90’ κυκλοφόρησαν πολλές επιστημονικές δημοσιεύσεις, σχετικά με τις προοπτικές ανάπτυξής του. Αν και την εποχή εκείνη ο όρος διαδίκτυο αποτελούσε κάτι συγκεκριμένο, δηλαδή την ευρεία σύνδεση υπολογιστών σε παγκόσμιο επίπεδο, πολλές από αυτές τις δημοσιεύεις, επαναπροσδιορίζουν και κάνουν πιο ρευστή αυτή την έννοια. Γενική ιδέα, αποτελεί πως η ανάπτυξη του διαδικτύου δεν έχει ολοκληρωθεί αλλά η σύγχρονη εποχή αποτελεί κομβικό σημείο της, καθώς δεν υπάρχει όριο στις ιδέες και τους στόχους που μπορούν να επιτευχθούν στο μέλλον. Συνεπώς, παρουσιάζονται ιδέες, οραματισμοί και σχεδιασμοί για την πορεία της, πολλές από τις οποίες, αν όχι οι περισσότερες, οδηγούν στην έννοια του Internet of Things και τη δημιουργούν.

Αξίζει να σημειωθεί πως ο όρος που χρησιμοποιούταν, εκείνη την εποχή ήταν διαφορετικός και στην πραγματικότητα όχι ένας, με πιο δημοφιλείς τους Πανταχού Παρούσα Υπολογιστική (Ubiquitous Computing) και Διάχυτη Υπολογιστική (Pervasive Computing). Τον πρώτο και πιο γνωστό από τους δύο όρους, εισήγαγε ο Mark Weiser, επικεφαλής τεχνολόγος του Xerox PARC, το 1988 [8]. Ο Weiser, επηρεασμένος από διάφορους ανθρωπιστικούς τομείς, εκτός της πληροφορικής, συνέθεσε τη δική του ξεχωριστή οπτική για το πώς πρέπει να είναι οι υπολογιστές και το διαδίκτυο, μέσα από τη δημοσίευσή του με τίτλο *The Computer for the 21st Century*, στο επιστημονικό περιοδικό *Scientific American*, το 1991. Η δημοσίευση ανοίγει με την πρόταση ‘’The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it’’ (μτφρ. Οι πιο βαθιές τεχνολογίες είναι αυτές που εξαφανίζονται. Υφαίνονται μόνες τους στο ύφασμα της καθημερινής ζωής έως ότου δε διακρίνονται από αυτή). Υποστηρίζει ότι η πληροφορική πρέπει να ενσωματωθεί στην καθημερινή ζωή και το ανθρώπινο περιβάλλον, αντί να δημιουργεί ένα νέο δικό της κόσμο. Ο υπολογιστής, ακόμα και όταν είναι φορητός ή έχει ήχο και γραφικά, δεν είναι παρά ένα κουτί, κάτι ξεχωριστό από το περιβάλλον. Σύμφωνα με την άποψη του Weiser, οι υπολογιστές είναι χρήσιμοι, μόνο όταν υπηρετούν το ήδη υπάρχον περιβάλλον και για να γίνει αυτό, το κέντρο της πληροφορικής δεν πρέπει να είναι ο υπολογιστής, αλλά ο άνθρωπος. Παρομοιώνει την κατάσταση με τους κινητήρες, όπου κάποτε υπήρχε ένας ανά εργοστάσιο, ενώ πλέον ένα απλό αμάξι, περιέχει πάνω από 20. Αναφέρει πως αν και φαντάζει απίθανο να υπάρχουν εκατοντάδες υπολογιστές σε ένα δωμάτιο, το ίδιο απίθανο κάποτε φάνταζε να διατρέχουν εκατοντάδες Volt σε καλώδια μέσα στους τοίχους ενός δωματίου, κάτι που σήμερα θεωρούμε δεδομένο και ‘’αόρατο’’ [9].

Μεγάλο ενδιαφέρον, επίσης, παρουσιάζει η ομιλία του Bill Joy στο World Economic Forum στο Davos, το 1999, στην οποία παρουσιάζει το όραμά του για το διαδίκτυο το οποίο αποκαλεί *Six Webs* (Έξι Δίκτυα). Ο Joy χωρίζει το διαδίκτυο και τις δυνατότητές του σε έξι κατηγορίες. Μερικές από αυτές έχουν υλοποιηθεί ήδη, άλλες εν μέρει, μόνο πειραματικά ή και καθόλου. Τα Six Webs είναι τα εξής:

- Το κοντινό διαδίκτυο (The Near Web):
  Το διαδίκτυο που βλέπει κανείς σε μία κοντινή οθόνη, όπως έναν επιτραπέζιο υπολογιστή
- Το παρόν διαδίκτυο (The Here Web):
  Το διαδίκτυο στο οποίο κάποιος έχει πρόσβαση συνεχώς, όπου κι αν είναι, καθώς έχει πρόσβαση σε αυτό μέσω κάποιας συσκευής που φέρει μαζί του
- Το μακρινό διαδίκτυο (The Far Web):
  Το διαδίκτυο που βλέπει κανείς από μακριά μέσω κάποιας μεγάλης οθόνης
- Το παράξενο διαδίκτυο (The Weird Web):
  Το διαδίκτυο στο οποίο κάποιος μπορεί να αλληλοεπιδρά μέσω ομιλίας
- Επιχείρηση σε Επιχείρηση (Business to Business – B2B):
  Το διαδίκτυο στο οποίο δεν έχει πρόσβαση ο καταναλωτής, αλλά χρησιμοποιείται αποκλειστικά για την επικοινωνία μεταξύ μηχανημάτων/συσκευών των επιχειρήσεων
- Συσκευή σε Συσκευή (Device to Device – D2D)
  Αποτελεί το διαδίκτυο των αισθητήρων σε πόλεις, με σκοπό τη βελτίωση των αστικών συστημάτων.

1.3.3 Το Internet of Things στη σύγχρονη εποχή

Η σύγχρονη εποχή θα μπορούσε να χαρακτηριστεί ως εποχή ανάπτυξης του Internet of Things. Οι απαραίτητες τεχνολογίες υπάρχουν και συνεχίζουν να βελτιώνονται. Το διαδίκτυο πλέον, όπως αναφέρθηκε προηγουμένως, αποτελεί κυρίαρχη και καθιερωμένη διασυνδεδεμένη τεχνολογία.

Νέες τεχνολογίες ασύρματης μετάδοσης δεδομένων έχουν αναπτυχθεί και εξελίσσονται συνεχώς. Μερικές από αυτές αναπτύχθηκαν για την κινητή τηλεφωνία. Κάποιες τεχνολογίες έχουν τη δυνατότητα κάλυψης μεγάλων αποστάσεων (π.χ. GPRS, LTE, LoRaWan), ενώ άλλες μικρότερων, για τη δημιουργία τοπικών ασύρματων δικτύων (π.χ. WiFi, Bluetooth, ZigBee, 6LoWPAN).

Στο κομμάτι του υλικού (hardware), οι επεξεργαστές εξακολουθούν να γίνονται ολοένα μικρότεροι και ισχυρότεροι. Τα κινητά τηλέφωνα τείνουν να αποτελούν μικρό αλλά ικανό υπολογιστή, με πρόσβαση στο διαδίκτυο. Τέτοιες δυνατότητες, αρχίζουν να διαθέτουν και συσκευές ακόμα μικρότερες από κινητά, όπως τα έξυπνα ρολόγια. Μικροί, χαμηλού κόστους, με πολλές δυνατότητες και εύκολοι στην χρήση μικροεπεξεργαστές, μερικοί με δυνατότητες σύνδεσης σε ασύρματα δίκτυα και το διαδίκτυο κυκλοφορούν στο εμπόριο (π.χ. *Atmel, Parallax, Arduino, Raspberry Pi, ARM*). Με αυτούς τους μικροεπεξεργαστές, ερευνητές, φοιτητές, ακόμα και μη ειδικοί ή παιδιά μπορούν να πειραματιστούν και να δημιουργήσουν με τις νέες τεχνολογίες, αναπτύσσοντας δικές τους ιδέες.

Όλες οι παραπάνω εξελίξεις ωθούν το όραμα του Internet of Things, περισσότερο από κάθε άλλη εποχή. Η τεχνολογία είναι πλέον ώριμη και προσιτή. Οι δημιουργοί έχουν στα χέρια τους, όλα τα εργαλεία που χρειάζονται, για να δημιουργήσουν δίκτυα από καθημερινά αντικείμενα, που συνδέονται μεταξύ τους και είναι προσβάσιμα εξ αποστάσεως. Με αυτό τον τρόπο η ιδέα του IoT γίνεται δημοφιλής στον μέσο κόσμο, κυριότερα από τις αρχές της δεκαετίας του 2010 [10]. Το IoT θεωρείται πλέον ξεχωριστός επιστημονικός κλάδος και όχι απλή εφαρμογή τεχνολογίας. Η επιστημονική κοινότητα εστιάζει καθαρά σε αυτό με νέες έρευνες. Εταιρίες δημιουργούν τα πρώτα σχετικά εμπορικά προϊόντα. Τέτοια προϊόντα αποτελούν έξυπνα ρολόγια/βηματοδότες, έξυπνοι θερμοστάτες, συστήματα ελεγχόμενου φωτισμού, γενικότερα έξυπνες οικιακές εγκαταστάσεις/συσκευές, βιομηχανικοί αυτοματισμοί κ.α.

Οι παραπάνω εξελίξεις θα μπορούσαν να θεωρηθούν αρκετές να υποστηρίξουν την ανάπτυξη του Internet of Things. Ωστόσο, η διείσδυσή του στον κόσμο γίνεται με σχετικά αργούς ρυθμούς. Πολλές από τις εμπορικές εφαρμογές έχουν επικριθεί για πλεονασμό, έλλειψη ασφάλειας και ελάχιστη αξία χρήσης [11]. Ποια είναι τα εμπόδια που καθυστερούν την ανάπτυξη και διάδοση του IoT; Η απάντηση δίνεται με νέα ερώτηση: Είναι αρκετή μόνο η ύπαρξη της τεχνολογίας; Η ερώτηση αυτή αποτελεί ‘’τροφή για σκέψη’’, ώστε να προκύψουν νέα ερωτήματα: Ποια η κατεύθυνση του IoT; Ποια ζητήματα προκύπτουν στον κόσμο από αυτό; Ποιος και πώς πρέπει να αντιμετωπίσει τα ζητήματα αυτά; Η σημασία των παραπάνω ερωτημάτων είναι πλέον αντιληπτή από την επιστημονική κοινότητα, η οποία προσπαθεί με

νέες έρευνες να εστιάσει στην επίλυσή τους. Τα παραπάνω θέματα θα αναλυθούν περισσότερο στο επόμενο κεφάλαιο.

2 Ανάλυση σύγχρονων θεμάτων και προσεγγίσεων στο Internet of Things

2.1 Απαιτήσεις και κοινωνικά ζητήματα

Το Internet of Things αποτελεί ξεχωριστό επιστημονικό κλάδο τη σύγχρονη εποχή. Ωστόσο, όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, προκύπτουν πολλαπλά ζητήματα, τα οποία πρέπει να επιλυθούν για την ανάπτυξή του. Τα ζητήματα αυτά εμπλέκουν την ακαδημαϊκή κοινότητα, τον εμπορικό κόσμο, την πολιτική και γενικότερα την κοινωνία. Κάθε ομάδα έχει τους δικούς της σκοπούς και στόχους, αλλά κοινή επιδίωξη είναι (ή τουλάχιστον θα έπρεπε να είναι) η ανάπτυξη του IoT για το συμφέρον του συνόλου της κοινωνίας.

Τα κυριότερα ζητήματα που θα αναλυθούν στη συνέχεια του κεφαλαίου, μπορούν να χωριστούν σε δύο κατηγορίες. Η πρώτη κατηγορία, αφορά την επιλογή και συμβατότητα τεχνολογιών και προτυποποιήσεων. Η δεύτερη εστιάζει στην ασφάλεια, τον έλεγχο και τη διαχείριση, μέσα στις τεχνολογίες, αλλά και των ίδιων των τεχνολογιών και όσων σχετίζονται με αυτές.

2.1.1 Συμβατότητα τεχνολογιών και προτυποποιήσεων

Οι τεχνολογίες για την υποστήριξη του IoT υπάρχουν και αναπτύσσονται συνεχώς. Ταυτόχρονα, δημιουργούνται νέες τεχνολογίες, μεταξύ άλλων κάποιες εστιασμένες σε αυτό. Συνεπώς, υπάρχουν πολλές και διαφορετικές προσεγγίσεις, η συμβατότητα των οποίων δεν είναι πάντα δεδομένη.

Η ανάπτυξη των τεχνολογιών γίνεται συνήθως από την ακαδημαϊκή κοινότητα, μέσω ανοιχτών μη κερδοσκοπικών οργανισμών. Η αρχική ιδέα μίας τεχνολογίας ή προτύπου μπορεί να προέρχεται από μεμονωμένα άτομα. Η δημιουργία μίας κοινότητας, εξυπηρετεί αφενός την προώθηση και αφετέρου την εξέλιξη της ιδέας αυτής. Ακαδημαϊκοί, ανεξάρτητοι ερευνητές, αλλά ακόμα και εταιρίες, συμμετέχουν σε μία κοινότητα υποστηρίζοντας, εξελίσσοντας και προωθώντας την τεχνολογία της. Με τη συμμετοχή του κάποιος, μπορεί να αποκτήσει εμπειρία και γνώσεις, να αποτελέσει αρωγός στην εξέλιξη και κατ' επέκταση να κερδίσει αναγνώριση, ειδικά στην περίπτωση επικράτησης της τεχνολογίας που υποστηρίζει.

Οι εταιρίες, από την άλλη πλευρά, έχουν τις δικές τους βλέψεις σχετικά με τις τεχνολογίες και τις κοινότητες που τις υποστηρίζουν. Σκοπός τους είναι να πρωταγωνιστήσουν στην αγορά του IoT, η οποία είναι ακόμη νέα και ρευστή. Για να δημιουργήσουν τα προϊόντα τους, επιλέγουν τις τεχνολογίες που κρίνουν καταλληλότερες. Στην επιλογή λαμβάνεται υπόψιν, όχι μόνο η ίδια η τεχνολογία με τις δυνατότητες που προσφέρει, αλλά και το κόστος χρήσης της. Το κόστος αυτό, πολλές φορές εξαρτάται από τις τρέχουσες καταστάσεις της εταιρίας (π.χ. εξοπλισμός), αλλά και του γειτονικού της περιβάλλοντος (π.χ. εθνικές εγκαταστάσεις, κόστος αδειών). Υιοθετώντας ή ερευνώντας την υιοθέτησή κάποιας τεχνολογίας, μία εταιρία μπορεί και επίσημα να δηλώσει την υποστήριξή στην κοινότητά της τεχνολογίας αυτής. Μία

τέτοια κίνηση, ενισχύει το εταιρικό της προφίλ, προβάλλοντας την ενεργή παρουσία της στην έρευνα και γενικότερα στον κλάδο του IoT.

2.1.2 Διαχείριση, έλεγχος και ασφάλεια

Η βασική λειτουργία του Internet of Things, δηλαδή η διασύνδεση πολλαπλών καθημερινών αντικειμένων μέσω του διαδικτύου, γεννά αρκετούς προβληματισμούς. Κάποιες συσκευές (Things) παράγουν δεδομένα και άλλες ελέγχονται (κάποιες μπορεί και τα δύο ταυτόχρονα). Αν οι συσκευές αυτές συνδέονται στο διαδίκτυο, τότε ποιος μπορεί να δει τα δεδομένα τους και ποιος να τις ελέγξει; Μπορεί οποιοσδήποτε να μάθει για την κατανάλωση ενέργειας ενός σπιτιού; Που καταλήγουν τα προσωπικά δεδομένα και ποιος εγγυάται για την ασφάλειά τους; Ποιος ελέγχει και διασφαλίζει την τήρηση όλων των παραπάνω και γενικά το IoT και τη βιομηχανία του; Τα ερωτήματα αυτά μπορούν να αναχθούν στα εξής τρία θέματα:

- έλεγχος πρόσβασης
- απόρρητο και ασφάλεια
- πολιτική και νομοθεσία

Το θέμα του ελέγχου πρόσβασης στο IoT είναι απαραίτητο για την ασφαλή ανάπτυξή του. Στην πραγματικότητα, δε μπορεί το IoT να καθιερωθεί στην καθημερινή ζωή χωρίς έλεγχο πρόσβασης. Για παράδειγμα, αν τα δεδομένα κατανάλωσης ενέργειας μίας οικίας είναι διαθέσιμα δημόσια, αυτά μπορούν να χρησιμοποιηθούν από διαρρήκτες, για να μάθουν πότε λείπουν όλοι από την οικία. Αν συνδέσει κάποιος τον κλιματισμό του γραφείου του, με το διαδίκτυο, μέσω μίας δημόσιας διεπαφής, τότε θα μπορεί ο οποιοσδήποτε να τον ελέγξει. Ο κόσμος χρειάζεται να ξέρει ποιοι μπορούν να δουν τα δεδομένα του και να χειριστούν τις συσκευές του. Στην ιδανικότερη περίπτωση, θα ήθελε να έχει ο ίδιος τον έλεγχο. Οι σχετικές τεχνολογίες και τα πρότυπα που αναπτύσσονται, πρέπει να λαμβάνουν σοβαρά υπόψιν τους τα ζητήματα αυτά. Στην ακαδημαϊκή κοινότητα, αν και στο παρελθόν δεν υπήρχε αρκετό ενδιαφέρον, οι έρευνες σχετικά με το IoT, εστιάζουν ολοένα και περισσότερο, στα ζητήματα ελέγχου πρόσβασης.

Τα θέματα απορρήτου και ασφάλειας είναι επίσης σημαντικά. Τα δύο βασικά ερωτήματα που προκύπτουν είναι: Που πηγαίνουν/ποιος μαζεύει τα δεδομένα; Ποιος εγγυάται την ασφάλειά τους; Η απάντηση και στα δύο ερωτήματα, εξαρτάται από την ίδια την εφαρμογή. Μικρότερες εφαρμογές που συλλέγουν λίγα σχετικά δεδομένα μπορούν να τρέχουν σε ιδιωτικό εξυπηρετητή (server). Σε αυτή την περίπτωση τα δεδομένα καταλήγουν εκεί, δηλαδή στον διαχειριστή της εφαρμογής, ο οποίος έχει και την ευθύνη της ασφάλειάς τους. Μεγαλύτερες εφαρμογές ωστόσο, που συλλέγουν πολλά δεδομένα και χειρίζονται πολλές συσκευές ταυτόχρονα, πιθανότατα να χρειαστούν υποστήριξη από υπολογιστικό νέφος (cloud). Η δημιουργία cloud απαιτεί μεγάλες και ακριβές υποδομές, καθώς και έξοδα συντήρησης. Συνεπώς, στις περισσότερες περιπτώσεις, το cloud εξασφαλίζεται ως ενοικιαζόμενη υπηρεσία, από μεγαλύτερες εταιρίες που ειδικεύονται σε αυτό. Τα δεδομένα, επομένως, σε αυτές τις περιπτώσεις, καταλήγουν στις εταιρίες που παρέχουν το cloud. Οι πάροχοι του cloud, έχουν την ευθύνη της διατήρησης του απορρήτου (που έχει συμφωνηθεί με τον διαχειριστή της εφαρμογής) και της ασφάλειας των δεδομένων. Η τήρηση αυτών των

ευθυνών, είναι αναγκαία ώστε ο πάροχος να εμπνέει εμπιστοσύνη και έτσι να προσελκύει περισσότερους πελάτες.

Σχετικά με τα θέματα της πολιτικής και της νομοθεσίας, το IoT είναι κάτι σχετικά καινούργιο και μη διαδεδομένο. Οι προοπτικές του IoT και τα ζητήματα που αναφέρθηκαν προηγουμένως στην ενότητα αυτή, επιβάλουν την ύπαρξη σχετικών νομικών διατάξεων [12]. Τέτοιες διατάξεις πρέπει να εξασφαλίζουν την ομαλή και υγιή ένταξη του IoT στην κοινωνία, διασφαλίζοντας τις ευθύνες, τις υποχρεώσεις και τα δικαιώματα κάθε ομάδας. Επιπρόσθετα, οι διατάξεις αυτές πρέπει να προωθούν τη σύγκλιση των τεχνολογιών και να ρυθμίζουν τον ‘’πόλεμο’’ μεταξύ των προτύπων και των εταιριών, με σκοπό τη γενικότερη ανάπτυξη του κλάδου [12]. Εν τούτοις, οι προσπάθειες που έχουν γίνει μέχρι σήμερα, είναι λίγες.

2.2 Τεχνολογικές προκλήσεις

Αν και τα προηγούμενα ζητήματα χρήζουν μεγάλου ενδιαφέροντος, βάση των ζητημάτων για την ανάπτυξη του Internet of Things αποτελούν οι τεχνολογικές προκλήσεις. Η φύση του IoT ωθεί τις τεχνολογίες στα όρια της εξέλιξής τους, στα θέματα της απόδοσης και κατανάλωσης ενέργειας. Στο όραμα του IoT τα Things αποτελούν όσο το δυνατόν μικρότερες συσκευές, σε όσο μεγαλύτερη πυκνότητα μέσα στο ανθρώπινο περιβάλλον. Αν και οι επεξεργαστές εξελίσσονται συνεχώς και γίνονται αποδοτικότεροι, επιτακτική είναι η χρήση συσκευών περιορισμένων δυνατοτήτων. Ο όρος αυτός αναφέρεται σε συσκευές με περιορισμένη επεξεργαστική ισχύ, μνήμη και κατανάλωση ενέργειας. Αυτά τα χαρακτηριστικά είναι απαραίτητα για δύο λόγους/περιπτώσεις. Αρχικά, όσο ανεπτυγμένοι και αποδοτικοί κι αν είναι οι επεξεργαστές, πάντα υπάρχει κάποιο όριο μεγέθους, κάτω από το οποίο οι δυνατότητες τους είναι περιορισμένες (σε σχέση με μεγαλύτερες συσκευές). Το IoT ωθεί τα Things να αποτελούν τις μικρότερες επεξεργαστικές συσκευές, της κάθε εποχής, που μπορούν να υπάρξουν. Ο δεύτερος λόγος εμπίπτει στην πυκνότητα των things μέσα στο ανθρώπινο περιβάλλον. Ο μεγάλος αριθμός των things γεννά ζητήματα κόστους και κατανάλωσης ενέργειας.

2.3 Ενδεικτικές τεχνολογίες και πρωτόκολλα

Η δημιουργία ενός δικτύου Internet of Things, απαιτεί τον συνδυασμό πολλών τεχνολογιών. Κάθε μία από αυτές τις τεχνολογίες επιλέγεται για να υποστηρίξει ένα επίπεδο επικοινωνίας. Τα επίπεδα αυτά μπορούν να συμπεριληφθούν στις εξής βασικές κατηγορίες:

- Φυσικό επίπεδο μεταφοράς δεδομένων
- Πρωτόκολλο μεταφοράς δεδομένων
- Πρωτόκολλο λειτουργίας (IoT) δικτύου

Η πρώτη κατηγορία ανήκει στο χαμηλότερο επίπεδο της στοίβας του δικτύου. Καθορίζει το φυσικό μέσο μετάδοσης (καλώδιο, αέρας κλπ) και τη διαμόρφωση της πληροφορίας πάνω σε αυτό. Δεδομένη αποτελεί η επιλογή της χρήσης καλωδίου, αλλά στον κόσμο του IoT, η πλειονότητα των εφαρμογών χρειάζεται συσκευές που επικοινωνούν ασύρματα και πολλές φορές είναι φορητές. Υπάρχουν αρκετές κατηγορίες ασύρματων τεχνολογιών. Κάθε κατηγορία εξυπηρετεί διαφορετικό σκοπό αλλά περιέχει τεχνολογίες που ανταγωνίζονται

μεταξύ τους. Στη συνέχεια θα παρουσιαστούν οι κυριότερες κατηγορίες, με ενδεικτικά παραδείγματα τεχνολογιών για κάθε μια:

- Δημοφιλέστερα είναι τα LPWANs (Low Power Wide Area Networks). Αυτά τα δίκτυα επιτρέπουν ασύρματη επικοινωνία σε ευρείες, ακόμα και εθνικές, εκτάσεις, με χρήση της ελάχιστης δυνατής ενέργειας από τις συσκευές [13]. Υπάρχουν δύο κατηγορίες τέτοιων δικτύων: τα Cellular και τα Non-Cellular. Με τα πρώτα αναφερόμαστε στις τεχνολογίες που έχουν προκύψει από την κινητή τηλεφωνία, ενώ με τα δεύτερα στις υπόλοιπες.
    - Non-Cellular
        - LoRa: υποστηρίζεται από το LoRa Alliance που είναι μη κερδοσκοπικός οργανισμός. Μέλη του LoRa Alliance αποτελούν μεγάλες εταιρίες όπως οι Cisco, IBM, Orange, ZTE, SK Telecom, Arduino, Bosch, Legrand [14]
        - SigFox: υποστηρίζεται από την ομώνυμη εταιρία, η οποία έχει συνεργαστεί με μεγάλες εταιρίες όπως οι Texas Instruments, Silicon Labs και On Semiconductor [15]
    - Cellular: αυτές οι τεχνολογίες αναπτύσσονται από τον 3GPP που αποτελεί όμιλος συνεργασίας μεταξύ τηλεπικοινωνιακών οργανισμών ανά τον κόσμο [16]
        - 4G (LTE κ.α.): οι τεχνολογίες αυτές, αν και διαθέσιμες, απαιτούν μεγάλη κατανάλωση ενέργειας από τις συσκευές και συνεπώς δεν έχουν κριθεί κατάλληλες για χρήση στο IoT [17]
        - LTE Cat-M1: αποτελεί παραλλαγή του LTE για εφαρμογές στο IoT, με χαμηλότερες απαιτήσεις κατανάλωσης ενέργειας. Εταιρίες όπως η AT&T και η Verizon, έχουν εγκαταστάσεις που είναι ήδη συμβατές (απαιτείται μόνο ανανέωση λογισμικού) με την τεχνολογία και συνεπώς έχουν κέρδος αν την επιλέξουν [17].
        - NB-IoT/Cat-M2 (NarrowBand IoT): η συγκεκριμένη τεχνολογία δεν λειτουργεί στο φάσμα του LTE, συνεπώς αυξάνει το κόστος των εγκαταστάσεων. Ωστόσο, δεν απαιτεί τη χρήση ενδιάμεσων κόμβων (gateways). Εταιρίες όπως οι Huawei, Ericsson, Qualcomm και Vodafone σχεδιάζουν να το χρησιμοποιήσουν εμπορικά [17].
        - EC-GSM: η τεχνολογία αυτή κάνει χρήση του υπάρχοντος GSM δικτύου, το οποίο υποστηρίζεται από το 80% των εγκαταστάσεων κινητής τηλεφωνίας παγκοσμίως. Έχει δοκιμαστεί από τις Ericsson, Intel και Orange. Ωστόσο, δεν υπάρχει ιδιαίτερο ενδιαφέρον συγκριτικά με τις υπόλοιπες τεχνολογίες.
        - 5G: το πρότυπο των δικτύων 5ης γενιάς (5G) βρίσκεται ακόμα στην φάση του σχεδιασμού. Αναμένεται να παρέχει ακόμα μεγαλύτερο bandwidth, ώστε να υποστηρίζει περισσότερες συσκευές και υψηλότερες ταχύτητες. Υπάρχουν πλάνα για μικρότερες καθυστερήσεις και λιγότερες απαιτήσεις κατανάλωσης ενέργειας, για την υποστήριξη του IoT [18]. Ένας απ'τους στόχους είναι να μπορεί να υποστηρίξει ταυτόχρονα εκατοντάδες χιλιάδες ασύρματους αισθητήρες [18]. Η FCC των Ηνωμένων Πολιτειών, έχει δώσει ήδη το πράσινο φως, για τη χρήση του απαραίτητου φάσματος, από το 2016 [18].
- Για ασύρματη επικοινωνία σε μικρότερους χώρους υπάρχουν τα WPANs και WLANS (Wireless Personal/Local Area Networks). Η μικρή τους εμβέλεια, τα καθιστά χρήσιμα κυρίως σε περιπτώσεις αυτοματισμών σπιτιών, γραφείων και γενικότερα κτηρίων. Αυτό φαίνεται και από τις εταιρίες, οι οποίες δηλώνουν ενδιαφέρον για τις τεχνολογίες αυτές.

- Wi-Fi
- ZigBee: υποστηρίζεται από το ZigBee Alliance και είναι τεχνολογία κατάλληλη για εφαρμογές που απαιτούν χαμηλή κατανάλωση ενέργειας και χαρακτηρίζονται από χαμηλό ρυθμό μετάδοσης δεδομένων [19]. Μέλη του ZigBee Alliance αποτελούν εταιρίες όπως οι Huawei, Legrand, Philips, Schneider Electric, Somfy, Amazon, Belkin, BlackBerry, Carrier, General Electric, IKEA, Osram, Bosch, Samsung και Siemens [20].
- Z-Wave: ασύρματη τεχνολογία που χρησιμοποιείται κυρίως για αυτοματισμούς σπιτιού [21]. Υποστηρίζεται από το Z-Wave Alliance, μέλη του οποίου αποτελούν εταιρίες όπως οι LG, Bosch, Deutsche Telekom, Huawei, Legrand, Logitech, Panasonic, Sharp, SK Telecom και Somfy [22].

Το πρωτόκολλο μεταφοράς δεδομένων αποτελεί τη δεύτερη βασική κατηγορία και ανήκει στο ενδιάμεσο επίπεδο της στοίβας ενός δικτύου IoT. Σε αυτό το επίπεδο, δεν γίνεται απλή μεταφορά δεδομένων μεταξύ κάποιων τερματικών συσκευών (όπως στο προηγούμενο επίπεδο), αλλά μεταφορά δεδομένων μεταξύ κόμβων ενός δικτύου. Το πρωτόκολλο/α μεταφοράς δεδομένων, ορίζει τους κανόνες, τις διαδικασίες και τους τρόπους, με τους οποίους ορίζεται και λειτουργεί αυτό το δίκτυο. Σε αυτό το επίπεδο θεωρείται δεδομένη η χρήση του πρωτοκόλλου IP (Internet Protocol). Το πρωτόκολλο αυτό αποτελεί την βάση του σημερινού διαδικτύου, το οποίο όπως έχει αναφερθεί, αποτελεί την κυρίαρχη διασυνδεδεμένη τεχνολογία. Η 6η έκδοσή του (IPv6), θα μπορεί να υποστηρίζει ταυτόχρονα, περίπου 3.4 x 10^{38} διευθύνσεις, δηλαδή συσκευές [23]. Το 2008 ιδρύθηκε η IPSO Alliance. Η IPSO Alliance, είναι μία μη-κερδοσκοπική οργάνωση, η οποία όπως εκφράζουν και τα αρχικά της (IP for Smart Objects), προωθεί τη χρήση του IP για το IoT [24]. Μέλη της οργάνωσης αποτελούν πολλές μεγάλες εταιρίες όπως οι Intel, ARM, Bosch, Ericsson, Sigma, Silicon Labs κ.α.

Στην τρίτη βασική κατηγορία τεχνολογιών ενός IoT δικτύου, βρίσκεται το πρωτόκολλο λειτουργίας. Η κατηγόρια αυτή ανήκει στο υψηλότερο επίπεδο της στοίβας του δικτύου. Η επικοινωνία αποτελεί πλέον μεταφορά δεδομένων μεταξύ IoT κόμβων/συσκευών (αντί για απλών κόμβων δικτύου, όπως στο προηγούμενο επίπεδο). Το πρωτόκολλο λειτουργίας, ορίζει τους κανόνες, τις διαδικασίες και τους τρόπους, με τους οποίους ορίζεται και λειτουργεί το IoT δίκτυο και κατ' επέκταση την αρχιτεκτονική του. Η κατηγορία αυτή είναι προφανές πως είναι και η πιο κοντινή στο Internet of Things. Υπάρχει πλέον αρκετή ποικιλία ανάμεσα σε πρωτόκολλα, αλλά συνεχώς προκύπτουν νέα και εξελίσσονται τα υπάρχοντα. Είναι σημαντικό να αναφερθεί πως τα πρωτόκολλα αυτά, εκτός από ζητήματα λειτουργικότητας και επίδοσης, καλούνται να επιφέρουν λύσεις σε ζητήματα και προκλήσεις που αναφέρθηκαν στην αρχή του κεφαλαίου, όπως η ασφάλεια και ο έλεγχος πρόσβασης. Παρακάτω παρουσιάζονται ενδεικτικά μερικά από αυτά τα πρωτόκολλα.

- HTTP (HyperText Transfer Protocol)
  Αποτελεί το κυρίαρχο πρωτόκολλο στρώματος εφαρμογής στο διαδίκτυο. Σχεδιάστηκε και χρησιμοποιείται κατά κύριο λόγο για την εξυπηρέτηση ιστοσελίδων και διαδικτυακών εφαρμογών. Χρησιμοποίει το μοντέλο επικοινωνίας request – response (αίτηση – απάντηση). Αυτό σημαίνει πως κάθε επικοινωνία αρχίζει με μία ερώτηση ενός κόμβου σε κάποιον άλλον με ένα αίτημα (request). Ο δεύτερος κόμβος αφού λάβει και επεξεργαστεί το αίτημα, αποκρίνεται με μία απάντηση (response). Κατά κύριο λόγο, το μοντέλο αυτό έχει οδηγήσει στην αρχιτεκτονική client – server (πελάτη – εξυπηρετητή). Στην αρχιτεκτονική αυτή, οι κόμβοι

διαχωρίζονται σε servers (εξυπηρετητές) και clients (πελάτες). Clients είναι αυτοί που στέλνουν requests στους servers, ενώ servers είναι αυτοί που αναλαμβάνουν να δέχονται τα requests και να αποκρίνονται με responses. Η αρχιτεκτονική αυτή δεν ευνοεί την ανάπτυξη IoT δικτύων, καθώς η αμφίδρομη επικοινωνία επιτυγχάνεται με δυσκολία. Ωστόσο, χρησιμοποιείται συχνά σε σχετικές εφαρμογές, καθώς αποτελεί σταθερή, καθιερωμένη και ευρέως υποστηρίξιμη τεχνολογία. Οι παραπάνω παράγοντες συνεπώς, καθιστούν το HTTP μία απλή και εύκολη στην υλοποίηση λύση, για σχετικά μικρές εφαρμογές IoT.

- CoAP (Constrained Application Protocol)
  Αποτελεί πρωτόκολλο εφαρμογής που σχεδιάστηκε ώστε να χρησιμοποιείται από κόμβους περιορισμένων δυνατοτήτων σε μικρά δίκτυα χαμηλής ενέργειας και μεγάλης απώλειας [25]. Τέτοιοι κόμβοι μπορούν να αποτελούν μικρές, φθηνές και χαμηλής κατανάλωσης ενέργειας συσκευές σε ένα IoT δίκτυο. Το πρωτόκολλο μπορεί να αλληλοεπιδράσει εύκολα με το πρωτόκολλο HTTP, ώστε να υπάρχει δυνατότητα διασύνδεσης των μικρών IoT δικτύων με το διαδίκτυο, μέσω διαμεσολαβητών (proxies). Το μοντέλο επικοινωνίας είναι ίδιο με το HTTP, δηλαδή ισχύει η αρχιτεκτονική client – server. Κάθε thing μπορεί να αποτελεί έναν μικρό server, όπου οι πόροι του είναι διαθέσιμοι στους clients μέσω URIs (Uniform Resource Identifiers).
- MQTT (Message Queue Telemetry Transport)
  Το πρωτόκολλο αυτό, όπως και το CoAP, είναι σχεδιασμένο για κόμβους/συσκευές περιορισμένων δυνατοτήτων. Η αρχιτεκτονική είναι και εδώ client – server, ωστόσο, χρησιμοποιείται το μοντέλο επικοινωνίας publish – subscribe (έκδοσης – συνδρομής). Server αποτελεί ένας διακομιστής διαμεσολάβησης (broker) με τον οποίο συνδέονται όλοι οι κόμβοι στο δίκτυο ως clients. Τα things του δικτύου αποτελούν publishers (εκδότες) και στέλνουν δεδομένα στον broker κάτω από μία κατηγορία (topic). Οι ενδιαφερόμενοι clients για τα δεδομένα των things, αποτελούν subscribers (συνδρομητές) και δηλώνουν στον broker την εγγραφή τους στα topics που επιθυμούν. Στην συνέχεια, ο broker αναλαμβάνει να ενημερώνει για νέα δεδομένα των topics, όλους τους subscribers αυτών, κάθε φορά που δέχεται δημοσίευση από κάποιων publisher. Αυτό το μοντέλο επικοινωνίας χρήζει μεγάλης σημασίας σε IoT δίκτυα. Ιδιαίτερα αν θεωρήσουμε ότι η πλειονότητα των things αποτελούν αισθητήρες περιορισμένων δυνατοτήτων και κατανάλωσης ενέργειας, ενώ οι ενδιαφερόμενοι κόμβοι για τα δεδομένα των things αποτελούν περισσότερο ικανές συσκευές, τότε είναι σημαντικό τα things να στέλνουν ένα μόνο μήνυμα για κάθε ενημέρωση των δεδομένων του αισθητήρα τους, αντί να ειδοποιούν κάθε ενδιαφερόμενο ξεχωριστά.

3 Μία νέα προσέγγιση με το πρωτόκολλο Extensible Messaging and Presence Protocol (XMPP)

3.1 Σκοπός

Στα προηγούμενα κεφάλαια παρουσιάστηκε συνοπτικά το Internet of Things, το όραμα καθώς και οι προκλήσεις σχετικά με την ανάπτυξή του. Σκοπός αυτής της εργασίας είναι η χρήση του πρωτοκόλλου XMPP, για την ανάπτυξη μίας λύσης που προσεγγίζει το όραμα αυτό και ανταποκρίνεται στις προκλήσεις.

Το πρωτόκολλο XMPP, το οποίο θα παρουσιαστεί και θα αναλυθεί στη συνέχεια, αποτελεί πρωτόκολλο λειτουργίας του IoT δικτύου. Το βασικότερό του πλεονέκτημα έγκειται στο γεγονός ότι οι προτυποποιήσεις του εξελίσσονται συνεχώς, παρέχοντας νέα δομικά στοιχεία, μέσα στο ίδιο το πρωτόκολλο αλλά και δίνοντας τη δυνατότητα ανάπτυξης και χρήσης νέων στοιχείων εκτός προτυποποίησης. Συνεπώς, με βάση αυτό το πλεονέκτημα, στόχος είναι η δημιουργία ενός περιβάλλοντος λειτουργίας IoT δικτύου, το οποίο ανταποκρίνεται στις σύγχρονες προκλήσεις και έχει την δυνατότητα να κλιμακωθεί, απαντώντας στο όραμα του IoT.

3.2 Το πρωτόκολλο XMPP

3.2.1 Ορισμός

Το πρωτόκολλο XMPP (eXtensible Messaging and Presence Protocol) αποτελεί πρωτόκολλο που υποστηρίζει την αμφίδρομη επικοινωνία μεταξύ κόμβων δικτύου σε πραγματικό χρόνο. Στη στοίβα δικτύου λειτουργεί στο Application Layer (Επίπεδο Εφαρμογής), πάνω από το πρωτόκολλο TCP (Transmission Control Protocol / Πρωτόκολλο Ελέγχου Μεταφοράς). Μπορεί επίσης να λειτουργήσει πάνω από το HTTP (που ανήκει στο Application Layer), μέσω του πρωτοκόλλου BOSH, που προσδίδει την ιδιότητα της αμφίδρομης επικοινωνίας. Τα μηνύματα που χρησιμοποιούνται είναι δομημένα με βάση το XML (eXtensible Markup Language). Προτυποποιείται στα RFC 6120 και RFC 6121 (αρχικά RFC 3920 και RFC 3921) [26].

Με αρχική ονομασία Jabber το 1999, το πρωτόκολλο XMPP στόχευε αρχικά σε υπηρεσίες Instant Messaging (άμεσων μηνυμάτων). Ωστόσο, οι δυνατότητες επέκτασής του, το έκαναν κατάλληλο για χρήση σε σχεδόν οποιαδήποτε εφαρμογή μεταφοράς δεδομένων είτε μεταξύ ανθρώπων (Streaming, Social Networks, κλπ), είτε μεταξύ μηχανών (M2M – Machine to Machine, Online Gaming, Internet of Things κλπ).

3.2.2 Τοπολογία δικτύου

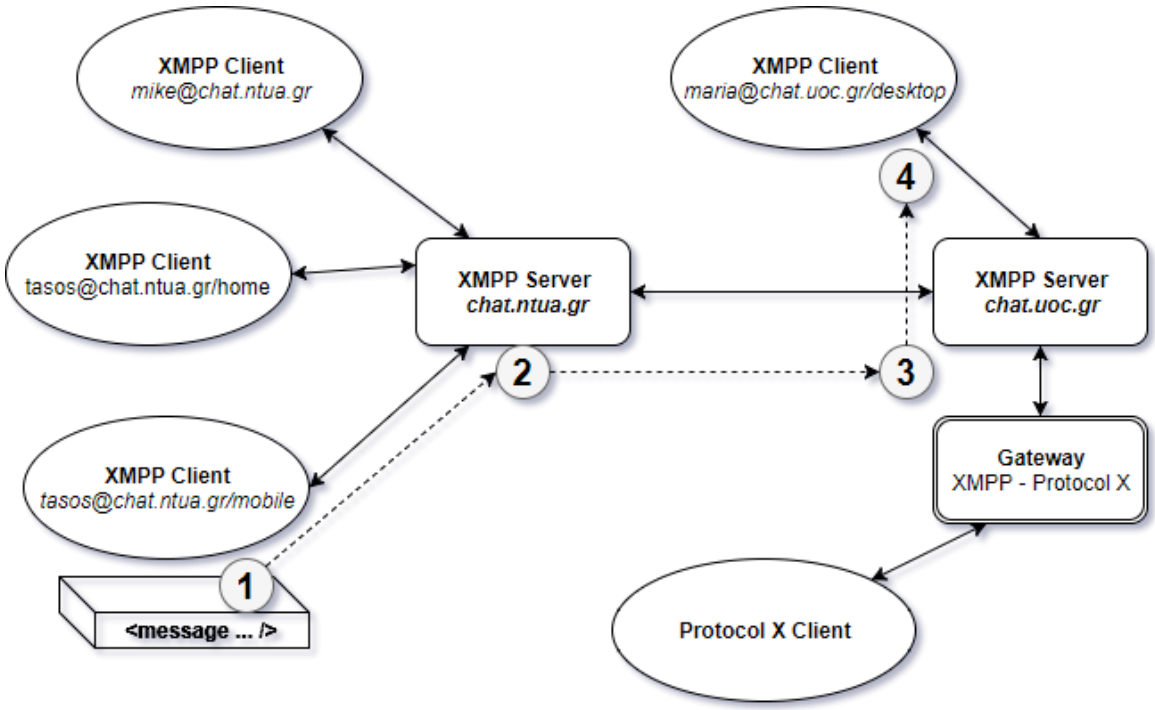
Αν και στο λογικό επίπεδο η επικοινωνία μεταξύ κόμβων του XMPP δικτύου είναι Peer to Peer (άμεση από κόμβο σε κόμβο), στο φυσικό επίπεδο παρεμβάλλεται ο server. Συνεπώς, η αρχιτεκτονική που χρησιμοποιείται είναι client – server. Οπότε, οι clients που επιθυμούν να επικοινωνήσουν μεταξύ τους, συνδέονται στον server και τα μηνύματα μεταξύ τους ακολουθούν την διαδρομή client – server – client.

Μεγάλο πλεονέκτημα της τοπολογίας του XMPP είναι η αποκέντρωση. Πάρα το γεγονός ότι η αρχιτεκτονική είναι client – server, δεν υπάρχει κάποιος μοναδικός κεντρικός server ο οποίος ελέγχει όλη την επικοινωνία μεταξύ των clients. Ο καθένας έχει την δυνατότητα να τρέξει τον δικό του server, δημιουργώντας και ελέγχοντας το δικό του μικρό ή μεγάλο δίκτυο. Επιπρόσθετα, μπορεί να γίνει διασύνδεση μεταξύ πολλαπλών XMPP servers, ώστε να υποστηρίζεται η επικοινωνία μεταξύ των clients τους. Σε αυτή την περίπτωση η διαδρομή ενός μηνύματος μεταξύ clients που ανήκουν σε διαφορετικό server είναι client – server – server – client. Ταυτόχρονα, υποστηρίζεται και η επικοινωνία με δίκτυα άλλων πρωτοκόλλων μέσω εφαρμογών διαμεσολάβησης που μπορεί να υλοποιεί ο server.

Ο server λειτουργεί κάτω από ένα domain name (όνομα τομέα π.χ. chat.ntua.gr) ή απλώς μία διεύθυνση IP. Κάθε client του δικτύου χαρακτηρίζεται από μία ταυτότητα που ονομάζεται JID (Jabber ID) και έχει δομή παρόμοια με αυτή της διεύθυνσης ενός email. Το JID περιέχει τρία μέρη και είναι της μορφής **user@domain/resource** :

- **user** – Αποτελεί το όνομα/ψευδώνυμο του χρήστη (client) μέσα στο δίκτυο (π.χ. tasos).
- **domain** – Αποτελεί το domain ή την IP του server στον οποίο είναι εγγεγραμμένος ο χρήστης (π.χ. chat.ntua.gr).
- **resource** (προαιρετικά) – Στην περίπτωση που ο ίδιος χρήστης επιθυμεί να συνδέεται από διαφορετικά σημεία/εφαρμογές/συσκευές, μπορεί να το δηλώσει ορίζοντας διαφορετικό resource σε καθένα από αυτά (π.χ. mobile, desktop, home, office).

Συνεπώς, σύμφωνα με τα παραδείγματα που δόθηκαν παραπάνω, ένα ενδεικτικό JID είναι το **tasos@chat.ntua.gr/mobile** . Στο παρακάτω σχήμα φαίνεται ενδεικτικά η τοπολογία ενός XMPP δικτύου και στις διακεκομμένες γραμμές η διαδρομή που θα ακολουθήσει ένα μήνυμα από τον *tasos@chat.ntua.gr/mobile* στη *maria@chat.uoc.gr/desktop* :



3.2.3 Μορφή και τύποι μηνυμάτων

Όπως αναφέρθηκε προηγουμένως, τα μηνύματα του XMPP βασίζονται στη γλώσσα XML. Η XML αποτελεί γλώσσα σήμανσης και ορίζει κανόνες για την κωδικοποίηση δομημένων μηνυμάτων/εγγράφων σε μορφή που είναι εύκολα αναγνώσιμη τόσο από μηχανές (υπολογιστές), όσο και από τον άνθρωπο. Στην εικόνα παρακάτω φαίνεται ένα παράδειγμα ενός εγγράφου XML.

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Σχήμα 6 – Παράδειγμα XML εγγράφου

Φαίνεται εύκολα ότι η σήμανση XML αποτελείται από αντικείμενα (π.χ. Books, Book, title, author) και καθένα από αυτά τα αντικείμενα εκτός από ιδιότητες (π.χ. ISBN=”….”), μπορεί να περιέχει άλλα αντικείμενα (π.χ. Books περιέχει Book, Book περιέχει title και author). Αυτό ορίζει επίπεδα βάθους (levels) μέσα στο έγγραφο. Στο παραπάνω έγγραφο το αντικείμενο Books βρίσκεται στο υψηλότερο επίπεδο ή επίπεδο 0. Στο αμέσως χαμηλότερο επίπεδο, δηλαδή το επίπεδο 1, βρίσκονται τα Book. Τέλος, στο επίπεδο 2 βρίσκονται τα αντικείμενα title και author.

Η σύνδεση ενός client με έναν XMPP Server αποτελεί ένα XML Stream (ροή XML). Αυτό σημαίνει πως υπάρχει ένα κοινό XML έγγραφο, μεταξύ client και server το οποίο συμπληρώνεται από κοινού όσο η σύνδεση είναι ανοιχτή και ολοκληρώνεται όταν αυτή κλείσει. Στο μηδενικό επίπεδο αυτού του εγγράφου υπάρχει το αντικείμενο stream το οποίο ορίζει τη ροή αυτή.

Στο επόμενο επίπεδο, το επίπεδο 1, βρίσκονται τα XMPP Stanzas (όχι ονομασία του πραγματικού XML αντικειμένου). Το Stanza αποτελεί το XML αντικείμενο της βασικής μονάδας μηνύματος του XMPP. Δηλαδή, το stanza χαρακτηρίζει ένα μήνυμα μεταξύ client – client (όπως έχει αναφερθεί η δρομολόγηση γίνεται μέσω του server), client – server ή και server – server. Υπάρχουν τρία είδη stanza, όπου το καθένα αποτελεί διαφορετικό αντικείμενο XML και υλοποιεί διαφορετικούς βασικούς σκοπούς.

Τα τρία είδη των Stanzas είναι τα εξής:

- **message** – αποτελεί τη βασική μονάδα ενός απλού μηνύματος μεταξύ clients. Βασικές του ιδιότητες είναι το from και το to (αποστολέας και παραλήπτης αντίστοιχα). Βασικό αντικείμενο που περιέχει είναι το body στο οποίο βρίσκεται το σώμα του μηνύματος. Φαίνεται συνεπώς, ότι αυτό το Stanza αποτελεί το βασικό για τη χρήση του XMPP σε εφαρμογές Instant Messaging.
- **iq** (Information Query) – είναι το stanza που ορίζει την διαδικασία ερωτο-απαντήσεων μεταξύ client – client ή client – server. Αυτή η διαδικασία απευθύνεται περισσότερο στην επικοινωνία μεταξύ μηχανών, παρά μεταξύ των ίδιων ανθρώπινων χρηστών. Βασική του ιδιότητα (εκτός από το to) αποτελεί το type (τύπος), που ορίζει τον τύπο του μηνύματος και μπορεί να έχει τις τιμές: get (για ερωτήσεις), set (για εντολές), result (για αποτελέσματα) και error (για σφάλματα).
- **presence** – αφορά τη δήλωση της παρουσίας και διαθεσιμότητας ενός client στο δίκτυο (ή την αίτηση συνδρομής στην παρουσία ενός άλλου client).

Ιδιαίτερο ενδιαφέρον παρουσιάζει το γεγονός πως το περιεχόμενο των μηνυμάτων αυτών, δεν είναι καθορισμένο και εκτός των προτυποποιήσεων που καθορίζουν ενδεικτικούς τρόπους χρήσης του για την υλοποίηση διαφόρων λειτουργιών, μπορεί να χρησιμοποιηθεί με οποιονδήποτε τρόπο, περιέχοντας οποιοδήποτε είδος πληροφορίας και ορίζοντας νέα μοντέλα επικοινωνίας και δυνατότητες. Στην παρακάτω εικόνα φαίνεται ένα ενδεικτικό XMPP Stream το οποίο περιέχει και τα τρία είδη των Stanzas.

```
<stream>
to='digital.output@example.org' id='1'>
     <message from='romeo@montague.lit' to='juliet@capulet.com' id='msg_1'>
          <body>Hellooo!!</body>
     </message>
     <presence to="user@otherhost.com" type="subscribe"/>
     <iq type='set' from='master@example.org/amr'
     to='digital.output@example.org' id='1'>
          <set xmlns='urn:xmpp:iot:control' xml:lang='en'>
               <boolean name='Output' value='true'/>
          </set>
     </iq>
</stream>
```

3.2.4 Οι επεκτάσεις (XEP) του XMPP

Μεγάλο πλεονέκτημα του XMPP, όπως έχει αναφερθεί, είναι οι επεκτάσεις που υποστηρίζει και εξελίσσονται συνεχώς. Συγκεκριμένα, έχει ιδρυθεί οργανισμός για ανάπτυξη προτυποποιημένων επεκτάσεων του πρωτοκόλλου. Ο οργανισμός αυτός ονομάζεται XSF (XMPP Standards Foundation). Οι επεκτάσεις που αναπτύσσονται και δημοσιεύονται ονομάζονται XEPs (XMPP Extension Protocols).

Κάθε XEP αποτελεί ξεχωριστό έγγραφο που ορίζει μία νέα προτυποποιημένη διαδικασία χρήσης του XMPP. Τέτοιες διαδικασίες χρήσης μπορούν να ορίσουν τη χρήση του XMPP σε διάφορους τομείς, όπως:

- κοινωνικά δίκτυα (π.χ. προφίλ χρηστών)
- online παιχνίδια
- μεταφορά και streaming αρχείων
- καθώς και στο Internet of Things

Αυτή την στιγμή υπάρχουν συνολικά 396 XEPs και συνεχώς εκδίδονται νέα. Χάρη πληρότητας, αναφέρεται πως υπάρχουν ακόμα και χιουμοριστικά XEP, ώστε να υπάρχει ευχάριστο και θετικό κλίμα μέσα στην κοινότητα.

Στη διαδικασία πρότασης XEP μπορεί να συμμετέχει οποιοσδήποτε επιθυμεί, απλά συντάσσοντας και υποβάλλοντας μία αρχική έκδοση του XEP. Τότε τα μέλη του οργανισμού το εξετάζουν και εφόσον το δεχτούν, κάνουν τις απαραίτητες διορθώσεις ώστε να δημοσιευθεί. Συγκεκριμένα, κάθε XEP περνάει μέσα από μία διαδικασία καταστάσεων, οι οποίες είναι οι εξής [27]:

- Experimental – πρώτο στάδιο, εφόσον το συμβούλιο το έχει αποδεχθεί και μπορεί να χρησιμοποιηθεί πειραματικά
- Proposed – πρώτο βήμα εφόσον η πειραματική χρήση του XEP αφήσει θετικές εντυπώσεις σε όλο το XSF και γίνονται συζητήσεις για τελικές αλλαγές
- Draft – το επόμενο βήμα μετά το Proposed, το οποίο αντιστοιχεί στην υποψηφιότητα για έκδοση
- Active – πλήρως αποδεκτό αλλά τροποποιήσεις μπορεί να γίνουν
- Final – πλήρως αποδεκτό και σταθερό χωρίς να αναμένονται τροποποιήσεις
- Deprecated – παλιό/ξεπερασμένο, νέες εκδόσεις του αποφεύγονται γιατί πιθανώς υπάρχει κάποιο νέο που το αντικαθιστά
- Deferred – Experimental το οποίο δεν αναβαθμίστηκε μέσα σε 1 χρόνο
- Obsolete – XEP το οποίο θεωρείται ότι δεν θα έπρεπε πλέον να υλοποιείται
- Retracted – XEP το οποίο έχει ζητήσει ο ίδιος ο συγγραφέας να αφαιρεθεί
- Rejected – XEP το οποίο έχει απορριφθεί από το συμβούλιο

Τα XEP που βρίσκονται σε σταθερό αλλά ακόμα και σε πειραματικό στάδιο, είναι κατάλληλα για υλοποίηση. Υπάρχουν ήδη αρκετές βιβλιοθήκες σε πολλές γλώσσες προγραμματισμού που υλοποιούν αρκετά από αυτά.

3.3 Παρουσίαση ενδεικτικότερων XEP

Σκοπός της εργασίας αυτής, όπως έχει γίνει ήδη γνωστό, είναι η εύρεση και υλοποίηση μίας λύσης για την ανάπτυξη ενός IoT δικτύου που ανταποκρίνεται στις σύγχρονες ανάγκες και προκλήσεις, με τη χρήση του πρωτοκόλλου XMPP. Εργαλείο για την επίτευξη του σκοπού αυτού, είναι η πληθώρα των XEP που υπάρχουν, αν και μπορούν να αναπτυχθούν νέες ειδικά προσαρμοσμένες διαδικασίες και είδη μηνυμάτων (stanzas), εφόσον αυτό κριθεί απαραίτητο. Προτιμότερη είναι ωστόσο η χρήση όσο το δυνατό περισσότερων έτοιμων δομικών στοιχείων, για λόγους συμμόρφωσης με την κοινότητα και τα ήδη σταθερά και ευρέως αποδεκτά πρότυπά της. Για τη σύγκριση και επιλογή των καταλληλότερων XEP σε μία ενιαία

υλοποίηση, χρειάζεται να τεθούν συγκεκριμένοι στόχοι, κριτήρια και απαιτήσεις. Εντούτοις, κρίθηκε σκόπιμο να παρουσιαστούν πρώτα αναλυτικά τα XEP που μελετήθηκαν. Με αυτόν τον τρόπο, στο επόμενο στάδιο, όπου γίνεται παρουσίαση στόχων/απαιτήσεων και επιλογή/σχεδιασμός της υλοποίησης, ο αναγνώστης έχει ήδη σαφή εικόνα σχετικά με τα XEP που ανταποκρίνονται σε κάθε μία πρόκληση.

Για την παρουσίαση αυτή, τα ενδεικτικότερα XEP θα χωριστούν σε δύο κατηγορίες:

- Non IoT Specific – XEP τα οποία σχεδιάστηκαν για γενική χρήση στο XMPP ή καθαρά για χρήσεις σε Instant Messaging, ωστόσο παρέχουν χρήσιμες δυνατότητες
- IoT Specific – XEP τα οποία σχεδιάστηκαν αποκλειστικά για τη χρήση του XMPP σε δίκτυα IoT

3.3.1 Non IoT Specific XEPs

Η επιλογή XEP τα οποία δε σχεδιάστηκαν για χρήση στο IoT, τη στιγμή που υπάρχουν άλλα σχεδιασμένα αποκλειστικά για αυτό, μπορεί να φαντάζει κακή. Ωστόσο, τα XEP αυτά παρέχουν καλά σχεδιασμένες λύσεις/δυνατότητες, που ανταποκρίνονται σε προκλήσεις που αφορούν γενικότερα τη διασυνδεσιμότητα μεταξύ όσο ανθρώπων, τόσο και μηχανών. Επιπρόσθετα, τα περισσότερα βρίσκονται σε σταθερή και ευρέως αποδεκτή έκδοση, σε αντίθεση με τα IoT Specific που βρίσκονται σε πειραματική κατάσταση και είναι υπο αμφισβήτηση για κάποιους.

XEP-0060: Publish – Subscribe

Έχει ήδη αναλυθεί περιληπτικά το μοντέλο επικοινωνίας Publish – Subscribe, όπως και το μοντέλο Request – Response. Στο μοντέλο Publish – Subscribe, οι Publishers παράγουν και δημοσιεύουν δεδομένα κάτω από μία θεματική ενότητα (topic). Οι Subscribers εγγράφονται σε αυτά τα topics και λαμβάνουν τα δεδομένα τους κάθε φορά που δημοσιεύονται νέα. Σε αντίθεση, στο μοντέλο Request – Response, οι παραγωγοί δεδομένων απαντούν σε αιτήματα (requests) των ενδιαφερόμενων για δεδομένα και συνεπώς πρέπει να στείλουν μήνυμα σε κάθε ενδιαφερόμενο αλλά και οι ενδιαφερόμενοι πρέπει να στέλνουν αιτήματα συνεχώς για να λαμβάνουν τα νέα δεδομένα. Οι αντιθέσεις αυτές και το πλεονέκτημα του μοντέλου Publish – Subscribe σχετικά με το IoT θα αναλυθούν στην επόμενη ενότητα.

Το XEP-0060: Publish – Subscribe περιγράφει μία συγκεκριμένη υλοποίηση αυτού του μοντέλου επικοινωνίας. Η υπηρεσία αυτή υλοποιείται στον XMPP Server, με τη χρήση κατάλληλου component (στοιχείου) και παρέχεται ως υπηρεσία κάτω από ένα subdomain (π.χ. pubsub.chat.ntua.gr). Τα topics του Publish – Subscribe εδώ χαρακτηρίζονται ως nodes (κόμβοι) που δημιουργούν δεντρική δομή. Συνεπώς κάθε node (δηλαδή topic), μπορεί να έχει sub-nodes (sub-topics), που στην ουσία αποτελούν διαφορετικά topic για το Publish – Subscribe. Υπάρχει ένας Owner (Ιδιοκτήτης) που αποτελεί τον XMPP Client και είναι αυτός που δημιουργεί το root (ριζικό) node στον Server. Ο Owner ελέγχει και διαχειρίζεται το node αυτό, καθώς και οποιαδήποτε nodes δημιουργηθούν κάτω του. Οι Publishers εφόσον είναι αποδεκτοί από τον Owner σε κάποιο node, δημοσιεύουν εκεί τα δεδομένα τους. Οι Subscribers μπορούν να ενημερωθούν για την λίστα των ριζικών κόμβων από τον Server, κάνουν αίτηση εγγραφής (subscribe) σε κόμβο και εφόσον γίνουν δεκτοί, σύμφωνα με

διαδικασία που ορίζει ο Owner (π.χ. όλοι δεκτοί, εξέταση κάθε μέλους, λίστα αποδοχής, λίστα απόρριψης), ενημερώνονται αυτόματα για κάθε δεδομένο που δημοσιεύεται σε αυτόν [28].

XEP-0045: Multi-User Chat

Το XMPP όπως είναι γνωστό σχεδιάστηκε αρχικά για τις ανάγκες του Instant Messaging. Σκοπός του ήταν να παρέχει τις περισσότερες δυνατότητες σε αυτό τον τομέα. Συνεπώς, θα αποτελούσε έλλειψη αν δεν υπήρχε επέκταση που ορίζει τη χρήση του για υλοποίηση υπηρεσίας Multi-User Chat. Η επικοινωνία γίνεται μέσα σε rooms (δωμάτια) συνομιλίας, μέσα στο πλαίσιο των οποίων, οι χρήστες στέλνουν και λαμβάνουν μηνύματα, δημόσια για όλους τους χρήστες. Εκτός από τις βασικές λειτουργίες ορίζεται και ένα ισχυρό μοντέλο ελέγχου πρόσβασης. [29]

Το Multi-User Chat, όπως και το Publish – Subscribe, παρέχεται από τον Server ως υπηρεσία κάτω από ένα subdomain (π.χ. rooms.chat.ntua.gr). Παρομοίως, ορίζεται και ο Owner (ιδιοκτήτης) που αποτελεί τον Client ο οποίος (ύστερα από κατάλληλη αίτηση στον server) δημιουργεί ένα δωμάτιο. Ο Owner επιλέγει τις βασικές ρυθμίσεις και αποτελεί τον αρχικό διαχειριστή του δωματίου. Τέτοιες ρυθμίσεις μπορεί να είναι για παράδειγμα το αν το δωμάτιο θα είναι μίας χρήσης ή όχι, το μέγιστο αριθμό ατόμων, αν θα έχουν δικαίωμα εισόδου μόνο συγκεκριμένα μέλη (clients), αν θα υπάρχουν συντονιστές και αν θα μπορούν τα μέλη να προσκαλέσουν άλλα μέλη. Επιπρόσθετα η διαχείριση έχει να κάνει με τα Affiliations (σχέσεις) και Roles (ρόλοι) που μπορούν να δοθούν σε άλλα μέλη του δωματίου. Οι έννοιες των Affiliations και Roles ορίζουν ένα ισχυρό μοντέλο ελέγχου πρόσβασης, το οποίο προσφέρει ένα ευρύ φάσμα τρόπων χρήσης. Τα Affiliations ορίζουν τη σχέση μέλους με την διαχείριση ελέγχου πρόσβασης στο δωμάτιο. Υπάρχουν τα εξής Affiliations (με ιεραρχική σειρά από την πιο σημαντική στην λιγότερο):

- Owner – Ο ιδιοκτήτης του δωματίου με όλες τις δυνατότητες διαχείρισης
- Admin – Διαχειριστής του δωματίου με δυνατότητες διαχείρισης που του έχουν δοθεί από τον Owner
- Member – Απλό μέλος του δωματίου / έχει απλά τη δυνατότητα να εισέλθει στην συζήτηση
- Outcast – Αποκλεισμένος από το δωμάτιο / δεν έχει τη δυνατότητα να εισέλθει
- None – Τίποτα από τα παραπάνω / δεν είναι αποκλεισμένος, ούτε αποτελεί μέλος

Τα Roles ορίζουν το επίπεδο πρόσβασης ενός μέλους στο δωμάτιο. Υπάρχουν τα εξής Roles (με ιεραρχική σειρά από την πιο σημαντική στη λιγότερο):

- Moderator – Συντονιστής / έχει τη δυνατότητα να ελέγχει τη συνομιλία και να απομακρύνει από το δωμάτιο (ban) οποιοδήποτε μέλος επιθυμεί
- Participant – Συμμετέχων / έχει τη δυνατότητα να λαμβάνει και να στείλει μηνύματα στη συνομιλία του δωματίου
- Visitor – Επισκέπτης / μέλος του δωματίου όπως οι προηγούμενοι αλλά μπορεί μόνο να λάβει μηνύματα της συνομιλίας, όχι να στείλει
- None – Δεν αποτελεί μέλος του δωματίου / σε περίπτωση που το δωμάτιο είναι members-only δε μπορεί να εισέλθει σε αυτό

Συνοψίζοντας, τα Affiliations ορίζουν το επίπεδο ‘’εξουσίας’’ των μελών στον έλεγχο πρόσβασης, ενώ τα Roles ορίζουν το ίδιο το επίπεδο πρόσβασης κάθε μέλους. Συνεπώς, τα μέλη με υψηλό Affiliation (Owner, Admins) μπορούν να ορίσουν ποια θα είναι τα μέλη του δωματίου και τι επίπεδο πρόσβασης θα έχει το καθένα.

Από την άλλη πλευρά, οι διαδικασίες για τα απλά μέλη είναι λιγότερες και πιο εύκολες. Κάθε client, μπορεί μέσω ενός απλού αιτήματος στην Multi-User Char (MUC) υπηρεσία του server, να λάβει τη λίστα των διαθέσιμων δωματίων. Έπειτα μπορεί να κάνει αίτηση εισόδου σε όποιο δωμάτιο επιθυμεί. Εφόσον γίνει δεκτός, αποτελεί μέλος του δωματίου και λαμβάνει αυτόματα όλα τα μηνύματα που δημοσιεύονται σε αυτό, μέσω του server. Αν του δοθεί το δικαίωμα μπορεί και αυτός αντίστοιχα να στείλει δικά του μηνύματα στα δωμάτιο. Όπως έχει διευκρινιστεί, υπάρχει και η δυνατότητα να του δοθεί ακόμα υψηλότερο Role ή Affiliation ώστε να αποκτήσει δυνατότητες διαχείρισης. Επιπρόσθετα, αξίζει να σημειωθεί ότι υπάρχει δυνατότητα προβολής της λίστας των μελών του δωματίου, με τα Jid τους, καθώς και το ποια από αυτά τα μέλη βρίσκονται αυτή την στιγμή σε αυτό (presence).

Είναι ξεκάθαρο πως το Multi-User Chat παρέχει μία ευρεία γκάμα δυνατοτήτων. Το μοντέλο επικοινωνίας είναι και εδώ Publish – Subscribe όπου τα δωμάτια έχουν τον ρόλο των topics. Συμπληρωματικά παρέχονται δυνατότητες ελέγχου πρόσβασης (access control) και ελέγχου παρουσίας (presence). Όλα αυτά είναι αρκετά και μεγάλα πλεονεκτήματα, για ένα δίκτυο IoT. Το γεγονός αυτό θα αναλυθεί εκτενέστερα στην επόμενη ενότητα.

3.3.2 IoT Specific XEPs

Τα IoT-Specific XEPs είναι μία σειρά από XEP που σχεδιάστηκαν μαζί για χρήση του XMPP στο Internet of Things. Αυτά τα XEP είναι σχετικά νέα σε σχέση με τα προηγούμενα που παρουσιάστηκαν και μη σταθερά, ωστόσο ο σχεδιασμός τους αφορούσε την κάλυψη συγκεκριμένων αναγκών. Κατά τη μελέτη για αυτή την εργασία, τα XEPs αυτά βρίσκονταν σε κατάσταση Experimental. Ωστόσο, μετά την αρχή της συγγραφής (συγκεκριμένα στις 20/05/2017) ο συγγραφέας τους τα ανακάλεσε και βρίσκονται πλέον σε κατάσταση Retracted. Είναι προφανές πως υπήρξε κριτική και αμφισβήτηση της αξίας τους. Ωστόσο σε αυτό το σημείο η ανάλυση περιορίζεται στην περιγραφή δυνατοτήτων καθενός από τα αξιοσημείωτα XEP. Στη συνέχεια, κατά τον σχεδιασμό της λύσης που θα υλοποιηθεί, γίνεται σαφής αξιολόγηση των XEP.

XEP-0323: Internet of Things - Sensor Data

Το XEP αυτό περιγράφει τις διαδικασίες και τη μορφή των μηνυμάτων για την παροχή δεδομένων από αισθητήρες. Οι αισθητήρες αποτελούν XMPP Clients με το δικό τους Jid. Οι ενδιαφερόμενοι για τα δεδομένα αποτελούν επίσης XMPP Clients. Συνεπώς καθορίζεται η διαδικασία του Read-Out Request. Ο ενδιαφερόμενος για δεδομένα στέλνει το Read-Out Request στον αισθητήρα. Σε αυτό το request μπορούν να ζητούνται μία ή περισσότερες τιμές δεδομένων. Ο αισθητήρας αφού λάβει το request μπορεί (εφόσον δεχτεί το αίτημα) να απαντήσει με το κατάλληλο response [30]. Το response μπορεί να καθορίζει πολλαπλές ιδιότητες κάθε τιμής. Τέτοιες ιδιότητες αποτελούν οι εξής:

- είδος – π.χ. στιγμιαία μέτρηση, υπολογισμένη τιμή, ακρότατο, αποθηκευμένη τιμή, χρονική τιμή
- τύπος – π.χ. ακέραιος, αριθμητικός, κείμενο, λογικός, χρονικός
- μονάδα μέτρησης – π.χ. Watt, ^{o}C, Sec

```
<iq type='get'
  from='client@example.org/amr'
  to='device@example.org'
  id='S0006'>
  <req xmlns='urn:xmpp:iot:sensordata' seqnr='6' momentary='true'>
    <node nodeId='Device04'/>
    <field name='Energy'/>
    <field name='Power'/>
  </req>
</iq>

<iq type='result'
  from='device@example.org'
  to='client@example.org/amr'
  id='S0006'>
  <accepted xmlns='urn:xmpp:iot:sensordata' seqnr='6'/>
</iq>

<message from='device@example.org'
  to='client@example.org/amr'>
  <fields xmlns='urn:xmpp:iot:sensordata' seqnr='6' done='true'>
    <node nodeId='Device04'>
      <timestamp value='2013-03-07T22:03:15'>
        <numeric name='Energy' momentary='true' automaticReadout='true'
value='12345.67' unit='MWh'/>
        <numeric name='Power' momentary='true' automaticReadout='true' value='239.4'
unit='W'/>
      </timestamp>
    </node>
  </fields>
</message>
```

Παραπάνω φαίνεται το παράδειγμα ενός Read-Out Request με τα αντίστοιχα Stanza που χρησιμοποιούνται. Παρατηρείται ότι το Request γίνεται με IQ Stanza, μία πρώτη απάντηση με το αν έγινε δεκτό το Request περιέχεται και αυτή σε IQ Stanza και ύστερα το Response με τις τιμές περιέχεται σε Message Stanza.

Σημαντικά χαρακτηριστικά του Sensor Data XEP είναι το μοντέλο επικοινωνίας Request – Response, καθώς και η δυνατότητα για σωστή και πλήρη κωδικοποίηση των δεδομένων αισθητήρων μαζί με τα χαρακτηριστικά τους. Τέτοια κωδικοποίηση εκτός από παροχή όλων των πληροφοριών συμβάλει και στη διαλειτουργικότητα μεταξύ διαφορετικών υλοποιήσεων/συσκευών.

XEP-0325: Internet of Things – Control

Σε ένα IoT δίκτυο, στο επίπεδο των δεδομένων, εκτός από τη συλλογή δεδομένων αισθητήρων εξίσου σημαντικός είναι και ο έλεγχος συσκευών. Δηλαδή, στο δίκτυο συνδέονται συσκευές που είτε παρέχουν δεδομένα, είτε τελούν υπό κάποια λειτουργία η οποία ελέγχεται, είτε και τα δύο. Συνεπώς, ως συμπλήρωμα στο Sensor Data XEP, υπάρχει το Control XEP. Αυτό το XEP προσδιορίζει τη διαδικασία με την οποία ένας XMPP Client (στο εξής αναφέρεται ως Client) μπορεί να στείλει εντολές ελέγχου σε έναν άλλο XMPP Client (στο εξής αναφέρεται ως Thing). Στην πραγματικότητα το Thing ορίζει μία σειρά από

παραμέτρους ελέγχου. Ο Client μπορεί να ζητήσει μία λίστα με αυτές τις παραμέτρους και εφόσον τις γνωρίζει μπορεί να στείλει request για να θέσει μία ή περισσότερες εξ αυτών [31]. Για την διαδικασία ζήτησης φόρμας, ο Client στέλνει ένα request στο Thing με την εντολή getForm και έπειτα εφόσον το Thing αποδεχτεί το αίτημα αποκρίνεται με τη φόρμα παραμέτρων, όπως φαίνεται ακολούθως:

```
<iq type='get'
  from='master@example.org/amr'
  to='dimmer@example.org'
  id='3'>
  <getForm xmlns='urn:xmpp:iot:control' xml:lang='en'/>
</iq>

<iq type='result'
  from='dimmer@example.org'
  to='master@example.org/amr'
  id='3'>
  <x type='form'
    xmlns='jabber:x:data'
    xmlns:xdv='http://jabber.org/protocol/xdata-validate'
    xmlns:xdl:='http://jabber.org/protocol/xdata-layout'
    xmlns:xdd:='urn:xmpp:xdata:dynamic'>
    <title>Dimmer</title>
    <xdl:page label='Output'>
      <xdl:fieldref var='FaceTimeMilliseconds'/>
      <xdl:fieldref var='OutputPercent'/>
      <xdl:fieldref var='MainSwitch'/>
    </xdl:page>
    <field var='xdd session' type='hidden'>
      <value>325ED0F3-9A9A-45A4-9634-4E0D41C5EA06</value>
    </field>
    <field var='FadeTimeMilliseconds' type='text-single' label='Fade Time (ms):'>
      <desc>Time in milliseconds used to fade the light to the desired level.</desc>
      <value>300</value>
      <xdv:validate datatype='xs:int'>
        <xdv:range min='0' max='4095'/>
      </xdv:validate>
      <xdd:notSame/>
    </field>
    <field var='OutputPercent' type='text-single' label='Output (%):'>
      <desc>Dimmer output, in percent.</desc>
      <value>100</value>
      <xdv:validate datatype='xs:int'>
        <xdv:range min='0' max='100'/>
      </xdv:validate>
      <xdd:notSame/>
    </field>
    <field var='MainSwitch' type='boolean' label='Main switch'>
      <desc>If the dimmer is turned on or off.</desc>
      <value>true</value>
      <xdd:notSame/>
    </field>
  </x>
</iq>
```

Το request όπως και το response περιέχονται σε IQ Stanzas (get και results) αντίστοιχα. Στο response παρατηρείται τόσο μία σύντομη αλλά και μία αναλυτική περιγραφή των παραμέτρων (σε αυτή την περίπτωση *FadeTimeMilliseconds*, *OutputPercent* και *MainSwitch*). Κάθε παράμετρος μπορεί να έχει τύπο (π.χ. απλό κείμενο ή λογική μεταβλητή), αναλυτική περιγραφή σε κείμενο (π.χ. *''Time in milliseconds used to fade the light to the desired level''*), τρέχουσα τιμή και εύρος τιμών. Συνεπώς, η φόρμα παραμέτρων παρέχει όλες τις πληροφορίες που χρειάζεται ο Client ώστε να μπορεί να ελέγξει το Thing.

Εφόσον ο Client γνωρίζει τις παραμέτρους του Thing (είτε μέσω της φόρμας, είτε εκ των προτέρων) μπορεί να στείλει Control Commands (εντολές ελέγχου) σε αυτό. Με αυτές τις εντολές μπορεί να αιτηθεί την αλλαγή μίας ή περισσοτέρων από αυτές τις παραμέτρους. Το Thing με τη σειρά του, εφόσον αποδεχτεί το αίτημα (χαρακτηριστικοί λόγοι απόρριψης μπορούν να είναι λάθος παράμετροι και αποκλεισμός πρόσβασης), θέτει ανάλογα τις τιμές του και αποκρίνεται με θετική επιβεβαίωση. Σημειώνεται ότι το σενάριο με επιβεβαίωση απαιτεί τη χρήση IQ Stanzas. Στο XEP ορίζεται και η περίπτωση αποστολής εντολών χωρίς επιβεβαίωση μέσω ενός Message Stanza. Παρακάτω απεικονίζεται ένα παράδειγμα του τελευταίου σεναρίου για την αλλαγή δύο παραμέτρων (*FadeTimeMilliseconds* και *OutputPercent* με βάση το προηγούμενο παράδειγμα):

```
<message from='master@example.org/amr'
  to='dimmer@example.org'>
  <set xmlns='urn:xmpp:iot:control'>
    <int name='FadeTimeMilliseconds' value='500'/>
    <int name='OutputPercent' value='10'/>
  </set>
</message>
```

XEP-0326: Internet of Things – Concentrators

Στο XEP αυτό ορίζεται η έννοια του (Thing) Concentrator. Ο Concentrator (συγκεντρωτής) είναι ένας XMPP Client ο οποίος αντιπροσωπεύει πολλά Things, καθένα από τα οποία αποτελεί node (κόμβο). Κάθε κόμβος χαρακτηρίζεται από μία ταυτότητα nodeId. Επίσης, κάθε node μπορεί να έχει child nodes (παιδιά), επιτρέποντας έτσι τη δεντρική δομή από nodes [32].

Ο Concentrator μπορεί να παρέχει μεθόδους προς άλλους XMPP Clients με τις οποίες:

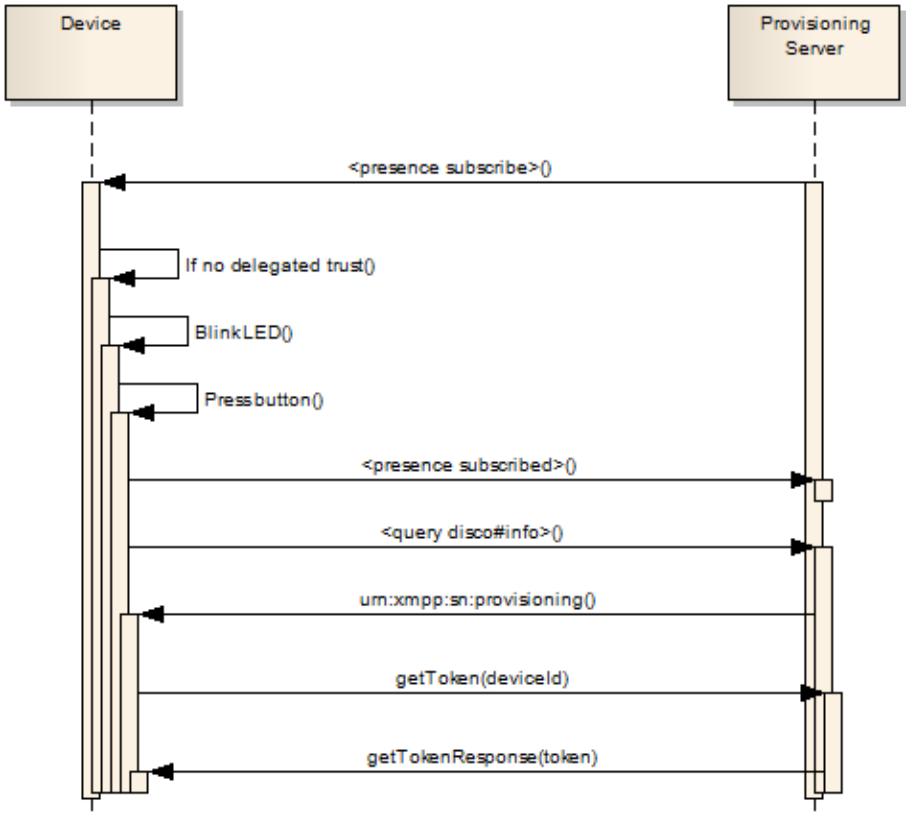
- γίνεται ανάκτηση των ονομάτων/ταυτοτήτων, της δομής και δυνατοτήτων (π.χ. isReadable, isControllable, hasCommands) των κόμβων του
- επιτρέπεται η μερική ή ολική διαχείριση των κόμβων του
- γίνεται αποστολή query ή command στους κόμβους, για συλλογή δεδομένων ή έλεγχο αντίστοιχα των Things του
- γνωστοποιούνται όλες οι παραπάνω μέθοδοι αναλυτικά με την μέθοδο getCapabilities (απαιτείται)

Είναι σημαντικό να σημειωθεί πως τα nodes εδώ δεν αποτελούν ξεχωριστούς XMPP Clients, ούτε γίνεται αναφορά για το πώς αυτά συνδέονται με τον Concentrator. Συνεπώς, τα nodes μπορούν να αποτελούν Things τα οποία συνδέονται με οποιονδήποτε τρόπο (φυσικό μέσο, πρωτόκολλο κλπ) με τον Concentrator, ώστε να συνδεθούν με το XMPP δίκτυο μέσω αυτού. Επίσης, άξιο αναφοράς είναι πως και εδώ όπως και στα προηγούμενα IoT XEPs το μοντέλο επικοινωνίας είναι Request – Response.

XEP-0324: Internet of Things – Provisioning

Το XEP αυτό περιγράφει διαδικασίες που επιτρέπουν τον καθολικό έλεγχο πρόσβασης (access control) σε ένα IoT δίκτυο. Για τον σκοπό αυτό ορίζεται η έννοια του Provisioning Server. Ο Provisioning Server αποτελεί Server Component ή XMPP Client ο οποίος ελέγχει την πρόσβαση που έχουν οι ενδιαφερόμενοι Clients στα Things [33].

Στο XEP αυτό αρχικά ορίζεται η διαδικασία με την οποία ένα Thing αποδέχεται και εγκαθίσταται στον Provisioning Server. Οι απαιτήσεις για τη συσκευή ενός τέτοιου Thing είναι πολύ μικρές. Αρκεί ένα push button (πιεστικό κουμπί) και ένα led ως διεπαφή χρήστη (interface) και η σύνδεση σε έναν XMPP Server κάτω από ένα Jid, προσβάσιμο από τον Provisioning Server. Κατά τη σύνδεση του Thing στον XMPP Server, ο Provisioning Server στέλνει αίτημα φιλίας σε αυτό. Μόλις φτάσει το αίτημα στο Thing, αυτό αναβοσβήνει το led για ένα χρονικό διάστημα (π.χ. 30’’), ειδοποιώντας το άτομο που εγκαθιστά το Thing. Αν πατηθεί το push button μέσα στο χρονικό διάστημα αυτό, το Thing αποδέχεται το αίτημα φιλίας. Το Thing τότε ζητάει την λίστα των υπηρεσιών που παρέχει ο νέος φίλος και αν μέσα σε αυτές περιέχεται το Provisioning Extension, τότε τον εμπιστεύεται ως υπεύθυνό του. Τέλος, ο Provisioning Server στέλνει στο Thing ένα token με το οποίο επιβεβαιώνονται τα access rights της συσκευής, για μετέπειτα χρήση. Παρακάτω φαίνεται ένα Sequence Diagram που περιγράφει την ανωτέρω διαδικασία.



Σημειώνεται ωστόσο πως το Jid του Provisioning Server μπορεί να εγγράφεται στη συσκευή κατά το στάδιο της παραγωγής, ώστε να μην χρειάζεται αυτή η διαδικασία.

Στην συνέχεια, το Thing μπορεί να χρησιμοποιεί τον Provisioning Server ώστε να γνωρίζει τα access rights (δικαιώματα πρόσβασης) που έχουν άλλοι Clients σε αυτό. Αυτό γίνεται μέσω αιτημάτων που στέλνει στον Provisioning Server, κάθε φορά που δέχεται αιτήματα πρόσβασης από κάποιον Client. Βάση της απάντησης που λάβει μπορεί να αποδεχτεί ή να απορρίψει τέτοια αιτήματα. Τα είδη αιτημάτων που περιγράφονται είναι τα εξής:

- isFriend – αίτημα με το οποίο ρωτάει το Thing αν κάποιος Client είναι φίλος και μπορεί να αποδεχτεί το αίτημα φιλίας του (Presence Subscribe), βάση του Jid. Επιπρόσθετα, ο Provisioning Server μπορεί κατά βούληση να επιβάλει την δημιουργία ή διαγραφή φιλιών
- canRead – αίτημα με το οποίο το Thing ρωτάει τον Provisioning Server αν κάποιος Client με συγκεκριμένο Jid, μπορεί να εκτελέσει Read-Out Request (XEP-0323: Sensor Data) σε αυτό. Η απάντηση μπορεί να μην είναι γενική για το Thing, αλλά να περιορίζεται σε nodes και σε fields (έλεγχος πρόσβασης συγκεκριμένα σε nodes ή fields)
- canControl – αίτημα με το οποίο το Thing μαθαίνει αν κάποιος Client με συγκεκριμένο Jid, μπορεί να εκτελέσει Control Request (XEP-0325: Control) σε αυτό. Όπως και στην προηγούμενη περίπτωση, έτσι και σε αυτή, η απάντηση μπορεί να περιορίζεται βάση των nodes ή των παραμέτρων ελέγχου.

Σημειώνεται επίσης, πως οι απαντήσεις του Provisioning Server μπορούν να γίνονται cached (αποθηκεύονται προσωρινά) στο Thing ώστε να μη χρειάζονται συνέχεια νέα Request.

Η έννοια του Provisioning Server είναι σημαντική και πρέπει να υπάρχει σε κάθε σύγχρονη υλοποίηση ενός δικτύου IoT που αποβλέπει σε καθολική και εκτεταμένη χρήση. Περισσότερα για αυτό θα αναλυθούν στην επόμενη ενότητα.

XEP-0347: Internet of Things - Discovery

Το XEP αυτό περιγράφει διαδικασίες με τις οποίες τα Things μπορούν να ανακαλυφθούν μέσα στο IoT δίκτυο. Οι Clients από μόνοι τους δε μπορούν να ξέρουν τα Jid των Things που υπάρχουν. Επίσης, νέα Things μπορούν να προστίθενται συνεχώς. Για τον σκοπό αυτό ορίζεται η έννοια του Thing Registry. Το Thing Registry μπορεί να αποτελεί Server Component ή XMPP Client, κομμάτι του Provisioning Server ή αυτοτελής και σκοπός του είναι η εγγραφή των Things του δικτύου σε αυτό, ώστε να μπορούν να ανακαλυφθούν από Clients [34].

Κάθε Thing μπορεί ιδανικότερα να λαμβάνει ένα Jid και το domain του XMPP Server κατά το στάδιο της παραγωγής. Ωστόσο, μπορεί να υπάρχει και η δυνατότητα ρύθμισης τους κατά τη χρήση. Το ίδιο ισχύει και για τις διευθύνσεις (domain ή Jid) του Thing Registry και του Provisioning Server.

Κατά τη σύνδεσή τους στο δίκτυο, τα Things εγγράφονται κάτω από ιδιοκτησία. Ως ιδιοκτήτης ενός Thing μπορεί να είναι είτε το ίδιο, είτε κάποιος άλλος Client. Στην πρώτη περίπτωση η δήλωση (αυτό-)ιδιοκτησίας γίνεται μαζί με την εγγραφή του Thing στο Registry. Στη δεύτερη περίπτωση, το Thing εγγράφεται στο Registry και η δήλωση ιδιοκτησίας γίνεται σε μετέπειτα στάδιο. Κατά την εγγραφή του στο Registry, το Thing στέλνει κάποια meta-tags (πληροφορίες ετικέτας). Μεταξύ αυτών βρίσκεται και το KEY που αποτελεί μοναδικό κωδικό

του Thing. Για τη δήλωση ιδιοκτησίας ενός Thing από έναν άλλον Client, απαιτείται η αποστολή αυτού του KEY. Για τον σκοπό αυτό προτείνεται κωδικοποίηση του KEY σε QR-Code που βρίσκεται σε αυτοκόλλητο πάνω στο ίδιο το Thing, ώστε μόνο ο πραγματικός του ιδιοκτήτης να μπορεί να δηλώσει την ιδιοκτησία του πάνω σε αυτό. Με παρόμοια διαδικασία, μπορούν να εγγραφούν στο Registry, Things που αποτελούν μέρος ενός Concentrator (και όχι ξεχωριστό XMPP Client).

Στη συνέχεια, εφόσον ένα Thing έχει εγγραφεί και έχει ιδιοκτησία είναι public (δημόσιο) για αναζήτηση στο Registry. Οποιοσδήποτε Client με πρόσβαση στο Registry μπορεί να αναζητήσει Things μέσω συγκεκριμένων Request. Στα Request αυτά καθορίζει ένα σύνολο κανόνων αναζήτησης, σχετικά με τα tags, των Things που επιθυμεί. Για παράδειγμα κάποιο από τα tags θα μπορούσε να έχει την τοποθεσία του Thing και κάποιο άλλο την ιδιότητα του. Με αυτά τα δεδομένα, ένας Client θα μπορούσε να αναζητήσει όλους του αισθητήρες θερμότητας που βρίσκονται κοντά σε κάποια συγκεκριμένη τοποθεσία.

3.4 Επιλογή, ορισμός, σκοπός και ανάλυση της προσέγγισης

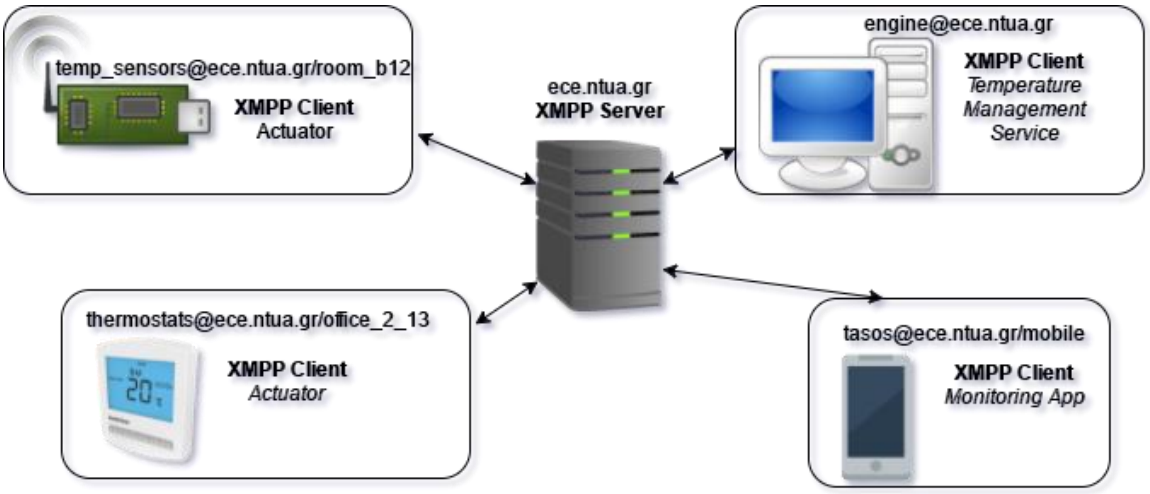
Πλέον, έχει δοθεί μία βασική γνώση γύρω από το πρωτόκολλο XMPP και τις δυνατότητές του, καθώς επίσης και μία αντίληψη για το Internet of Things και τις σύγχρονες προκλήσεις του. Σε αυτή την ενότητα θα αναλυθεί η επιλογή της λύσης, που θα υλοποιηθεί σε αυτή την εργασία. Αρχικά θα γίνει ανάλυση των απαιτήσεων. Στην συνέχεια, με βάση τις απαιτήσεις, θα ακολουθήσει μία σύγκριση των ενδεικτικών XEP της προηγούμενης ενότητας. Τέλος, η σύγκριση αυτή θα οδηγήσει στην επιλογή της τελικής λύσης προς υλοποίηση.

3.4.1 Γενική χρήση της τοπολογίας του XMPP σε IoT Networks

Η τοπολογία του XMPP βασίζεται στους XMPP Clients. Οι Clients συνδέονται στον XMPP Server δημιουργώντας έτσι ένα δίκτυο. Συνεπώς, σκόπιμο είναι να καθοριστεί το τι οντότητες, στα πλαίσια του IoT, μπορούν να αποτελέσουν XMPP Clients:

- Things (sensors, actuators κλπ)
- φυσικά άτομα / προσωπικοί λογαριασμοί
- γενικότερα συσκευές (smartphone, computer, concentrator)
- ΑΛΛΑ ΚΑΙ υπηρεσίες δικτύου που τρέχουν σε υπολογιστή ή cloud και μπορούν να προστίθενται δυναμικά

Διακρίνεται οπότε μία ελευθερία/πολυμορφία σχετικά με τις μορφές των οντοτήτων που μπορούν να συνυπάρξουν στο IoT δίκτυο ως XMPP Clients. Ένα παράδειγμα της τοπολογίας αυτής διακρίνεται στην παρακάτω εικόνα:



Ο XMPP Server από την μεριά του μπορεί να ορίσει ένα κλειστό, ως προς τον έξω κόσμο, IoT δίκτυο. Η οποιαδήποτε δυνατότητα ή περιορισμός εξωτερικής διασύνδεσης ορίζεται και ελέγχεται από αυτόν. Αυτό το στοιχείο προσδίδει ένα βασικό χαρακτηριστικό ασφάλειας και ελέγχου.

3.4.2 Καθορισμός απαιτήσεων

Στο 2ο Κεφάλαιο έχουν ήδη αναφερθεί οι σημαντικότερες σύγχρονες προκλήσεις στον κόσμο του Internet of Things. Σκοπός αυτής της εργασίας είναι η ανάπτυξη και υλοποίηση μίας λύσης, με χρήση του πρωτοκόλλου XMPP, που ανταποκρίνεται με το καλύτερο δυνατό τρόπο στα θέματα αυτά. Στην ενότητα αυτή, η ανασκόπηση αυτών των θεμάτων οδηγεί σε συμπεράσματα και προτάσεις κατεύθυνσης προς τη λύση. Ταυτόχρονα, κάθε θέμα και πρόταση μπορεί να συσχετιστεί με συγκεκριμένα XEP. Συνεπώς, ασκείται η κατάλληλη κριτική σε αυτά ώστε σε επόμενο στάδιο να αποτελεί βάσιμη η άμεση σύγκρισή τους.

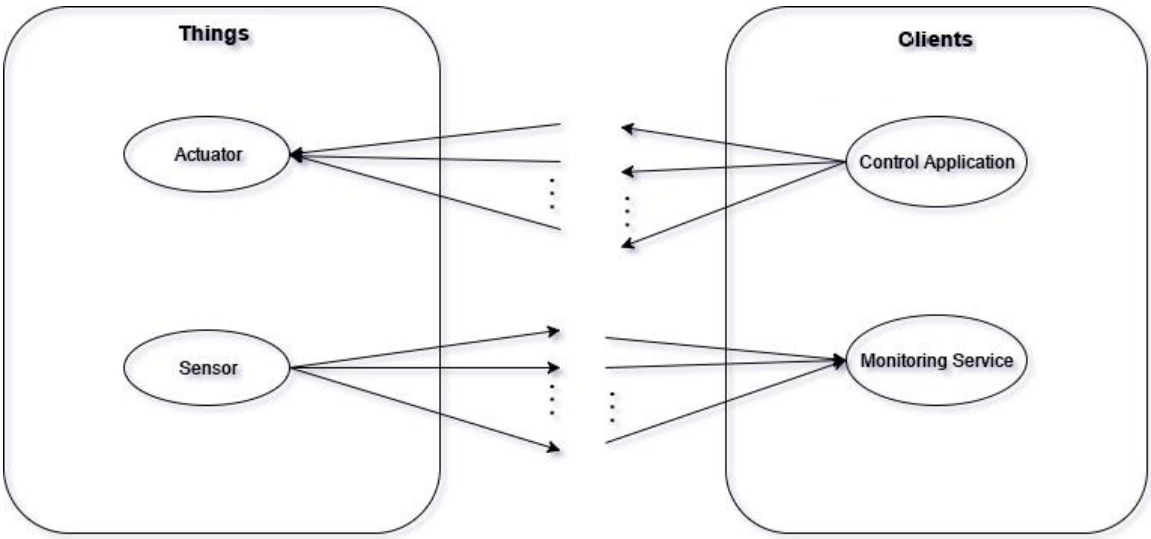
Για τον σωστό και κατανοητό ορισμό των απαιτήσεων, αυτές θα χωριστούν σε δύο γενικές κατηγορίες λειτουργιών: στο Επίπεδο Δεδομένων (Data Plain) και στο Επίπεδο Διαχείρισης (Control Plain). Στο Επίπεδο Δεδομένων ανήκουν οι λειτουργικές απαιτήσεις που αφορούν τη βασική μεταφορά δεδομένων μεταξύ κόμβων (Clients ή Things) μέσα στο IoT Δίκτυο. Τέτοιες λειτουργίες αποτελούν τις πιο βασικές και χαρακτηριστικότατες για την λειτουργία του δικτύου, όπως συλλογή δεδομένων από αισθητήρες, αποστολή εντολών σε Things κλπ. Από την άλλη πλευρά, στο Επίπεδο Διαχείρισης ανήκουν οι λειτουργικές απαιτήσεις που αφορούν τον έλεγχο και την διαχείριση του δικτύου. Στον σύγχρονο κόσμο του IoT, η ικανοποίηση απαιτήσεων καθενός εκ των δύο κατηγοριών είναι εξίσου σημαντική. Λύσεις που υστερούν σε μία από αυτές δυσκολεύονται να αποκτήσουν ευρεία αποδοχή. Αυτό συνέβη και με τα Internet of Things Specific XEPs του XMPP, τα οποία όπως έχει ήδη αναφερθεί, πρόσφατα ανακλήθηκαν από τον συγγραφέα τους.

3.4.2.1 Επίπεδο δεδομένων – Data Plain

Ένα IoT δίκτυο ορίζεται από κόμβους που επικοινωνούν, ανταλλάζοντας δεδομένα μεταξύ τους. Τα δεδομένα αυτά αποτελούν είτε δεδομένα αισθητήρων, είτε εντολές χειρισμού συσκευών. Χωρίς βλάβη της γενικότητας οι κόμβοι αυτοί μπορούν να χωριστούν σε δύο κατηγορίες οντοτήτων. Η πρώτη κατηγορία αποτελείται από μικρές συσκευές, διάσπαρτες στο φυσικό περιβάλλον που καταγράφουν δεδομένα ή/και εκτελούν λειτουργίες. Αυτές οι οντότητες αποτελούν τα Things. Στην δεύτερη κατηγορία, ανήκουν οποιοιδήποτε άλλοι κόμβοι/οντότητες. Σκοπός της ύπαρξης αυτών των οντοτήτων στο δίκτυο είναι η αλληλεπίδραση με τα Things. Αυτές οι οντότητες από εδώ και στο εξής θα αναφέρονται απλά ως Clients (οι clients του XMPP θα αναφέρονται ως XMPP Clients).

Οι απαιτήσεις ενός δικτύου IoT στο Επίπεδο Δεδομένων επηρεάζονται από τον ορισμό του Thing. Βασική είναι η υποστήριξη συσκευών περιορισμένων δυνατοτήτων σε μεγάλη πυκνότητα. Επιπρόσθετα, η σύνδεσή τους στο διαδίκτυο ενδέχεται να είναι και αυτή περιορισμένη.

Για την καλύτερη κατανόηση των απαιτήσεων ενός δικτύου IoT στο Επίπεδο Δεδομένων αρκεί η εξέταση της τοπολογίας του δικτύου στο επίπεδο αυτό. Τα μοτίβα ροής των δεδομένων ορίζουν τα ίδια τις ανάγκες.



Στο παραπάνω απλοποιημένο σχήμα φαίνεται αυτή η τοπολογία. Η γενική προφανής ροή δεδομένων είναι πως n (πολλά) Things επικοινωνούν αμφίδρομα με m (πολλούς) Clients.
Η πρώτη απαίτηση που δημιουργείται είναι αυτή της αμφίδρομης επικοινωνίας. Η ροή των δεδομένων μπορεί να είναι από τα Things προς τους Clients, στην περίπτωση των δεδομένων αισθητήρων, αλλά και από τους Clients προς τα Things στην περίπτωση εντολών ελέγχου. Είναι επίσης γνωστό πως κάποια Things μπορούν να κάνουν και τα δύο, δηλαδή και να παρέχουν δεδομένα και να δέχονται εντολές. Σημαντική είναι συνεπώς η διευκόλυνση της αμφίδρομης αυτής επικοινωνίας.

Για την ανάκτηση της επόμενης απαίτησης, χρειάζεται η εξέταση του παραπάνω σχήματος ξεχωριστά για κάθε μία πλευρά, αυτή του Client και αυτή του Thing. Στην περίπτωση ενός Thing, αυτό χρειάζεται να επικοινωνήσει με m (πολλούς) Clients. Η σύνδεση και επικοινωνία με κάθε Client ξεχωριστά απαιτεί υπολογιστικό, δικτυακό άρα συνεπώς και ενεργειακό κόστος. Το Thing οπότε επιθυμεί την μείωση αυτού του m (δηλαδή το πλήθος των Clients με τους οποίους επικοινωνεί). Ιδανικότερη αποτελεί η περίπτωση που το m γίνεται 1. Το αντίστοιχο ισχύει και για τον Client, ο οποίος έχει τον φόρτο να επικοινωνήσει με n Things και θα επιθυμούσε την μείωση του n. Πώς όμως γίνεται μία οντότητα που πρέπει να επικοινωνήσει με πολλές άλλες, στην πράξη να χρειαστεί να συνδεθεί με λιγότερες ή ακόμα και μία; Η απάντηση έχει δοθεί με το μοντέλο επικοινωνίας Publish – Subscribe. Η οντότητα μπορεί να ανοίξει και να διαχειριστεί έναν κόμβο Publish – Subscribe. Τα δεδομένα που θέλει να στείλει, τα στέλνει μόνο σε αυτόν τον κόμβο και από εκείνον ενημερώνονται αυτόματα οι ενδιαφερόμενοι Subscribers. Παρομοίως τα μηνύματα προς την οντότητα λαμβάνονται από τον κόμβο, ο οποίος ενημερώνει σε πραγματικό χρόνο την οντότητα. Επιπρόσθετα, η οντότητα δε χρειάζεται να λάβει αιτήσεις για την αποστολή δεδομένων, αλλά τα δημοσιεύει όποτε αυτή το επιθυμεί, σε αντίθεση με τον μοντέλο Request - Response.

Τα πλεονεκτήματα της χρήσης Publish – Subscribe, αντί του Request – Response είναι πλέον γνωστά. Ωστόσο ποιο από τα δύο είδη οντοτήτων στο IoT δίκτυο πρέπει να ωφεληθεί από αυτό; Είναι τα Things ή οι Clients που πρέπει να μπορούν να επικοινωνήσουν με όλους, χρησιμοποιώντας μία μόνο σύνδεση επικοινωνίας; Όπως έχει ήδη εξηγηθεί, η ανάγκη για λιγότερες συνδέσεις προκύπτει από τον φόρτο που δημιουργούν οι πολλές. Τα Things εξ ορισμού αποτελούνται ενδεχομένως από συσκευές περιορισμένων δυνατοτήτων και κατανάλωσης ενέργειας. Η αποστολή πολλαπλών μηνυμάτων απαιτεί μεγαλύτερη υπολογιστική ισχύ, χρήση δικτύου και συνεπώς κατανάλωση ενέργειας. Σε αντίθεση, οι Clients αποτελούν (ή υποστηρίζονται από) ικανότερες συσκευές καθώς σκοπός τους είναι η αλληλεπίδραση με πολλά Things. Δηλαδή, τα Things χαρακτηρίζονται ως πολυπληθέστερα και μικρότερα (σε μέγεθος άρα και ικανότητες/πόρους), ενώ οι Clients ως λιγότεροι και μεγαλύτεροι. Με αυτό τον ορισμό γίνεται προφανές ότι το όφελος του Publish – Subscribe πρέπει να υποστηρίξει τα Things. Συνεπώς, ένα Thing ορίζοντας έναν κόμβο Publish – Subscribe, αρκεί να επικοινωνεί μόνο με αυτόν, στέλνοντας δεδομένα ή λαμβάνοντας εντολές, ενώ οι ενδιαφερόμενοι για το Thing Clients αρκεί να κάνουν Subscribe στον κόμβο για την λήψη δεδομένων ή να στείλουν σε αυτόν τις εντολές αντίστοιχα.

Η τρίτη απαίτηση έγκειται στην μορφή με την οποία αποστέλλονται τα δεδομένα. Τα δεδομένα αυτά αποτελούν είτε δεδομένα αισθητήρων, είτε εντολές ελέγχου. Ωστόσο η δομή τους πρέπει να είναι αυστηρά καθορισμένη, ιδανικότερα προτυποποιημένη. Με αυτό τον τρόπο αποφεύγονται ασάφειες και υποστηρίζεται η συμβατότητα μεταξύ συσκευών. Παράλληλα, η δομή αυτή πρέπει να επιτρέπει τον πλήρη χαρακτηρισμό των δεδομένων και την επεξήγησή τους, υποστηρίζοντας έτσι κάθε περίπτωση επικοινωνίας.

3.4.2.2 Επίπεδο Διαχείρισης - Control Plain

Μία σύγχρονη προσέγγιση ενός δικτύου IoT δεν θα μπορούσε να σταθεί χωρίς πρόβλεψη για το Επίπεδο Διαχείρισης. Για καλύτερη κατανόηση της σημασίας του, αρκεί κανείς να φανταστεί ένα δίκτυο μόνο με το Επίπεδο Δεδομένων. Σε ένα τέτοιο δίκτυο μπορούν να συνδεθούν Clients και Things παρέχοντας ή ζητώντας δεδομένα. Οποιοσδήποτε Client μπορεί να έχει πρόσβαση σε οποιοδήποτε Thing. Ωστόσο αυτό όπως έχει ήδη αναφερθεί μπορεί να έχει καταστρεπτικές συνέπειες. Η απαίτηση στο Επίπεδο Διαχείρισης που δημιουργεί αυτός ο κίνδυνος είναι ο Έλεγχος Πρόσβασης (Access Control). Ο Έλεγχος Πρόσβασης αποτελεί την πρόβλεψη για το ποιοι Clients έχουν πρόσβαση σε ποια Things, δηλαδή πιο απλά ποιος μπορεί να μιλήσει σε ποιον. Όσο απλό κι αν φαίνεται, οι τρόποι και οι τεχνικές με τους οποίους μπορεί να σχεδιαστεί ένα τέτοιο σύστημα είναι πολλοί.

Βασικό κομμάτι του ελέγχου πρόσβασης είναι το που αυτός ορίζεται και υλοποιείται. Μία απλή και εύκολη λύση θα ήταν στα ίδια τα Things. Με αυτό τον τρόπο τα ίδια τα Things προγραμματίζονται να έχουν λίστες με Clients στους οποίους επιτρέπουν την πρόσβαση ή Clients στους οποίους την απορρίπτουν. Η λίστα αυτή μπορεί να επαναπροσδιορίζεται. Ωστόσο πως μπορούν να ξέρουν οι ιδιοκτήτες των Things ποιους Clients να εμπιστευτούν; Συνεπώς η τεχνική αυτή ωθεί το χωρισμό του δικτύου σε μικρά κλειστά κομμάτια, σε αντίθεση με το όραμα του IoT για μεγάλα συνδεδεμένα δίκτυα. Για καλύτερη κατανόηση της έννοιας αυτής, θα μπορούσε κανείς να φανταστεί ένα δίκτυο από Smart Homes (έξυπνα σπίτια). Σε ένα τέτοιο δίκτυο υπάρχουν Things σε μεγάλη πυκνότητα μέσα σε κάθε σπίτι μίας κοινότητας, ελέγχοντας συσκευές και παρέχοντας δεδομένα (π.χ. θερμοκρασία, κατανάλωση ενέργειας, φωτεινότητα κλπ). Σε ένα διαμέρισμα ο ένοικος μη γνωρίζοντας σε ποιους μπορεί να εμπιστευθεί τα δεδομένα του μπορεί να ορίσει να επιτρέπεται η πρόσβαση των Things του μόνο από τον ίδιο. Παρομοίως, σε μία πολυκατοικία η διαχείριση θα δυσκολευτεί να αποφασίσει σε ποιους Clients της κοινότητας μπορεί να επιτρέψει την πρόσβαση. Επιπρόσθετα, θα μπορούσε να προκληθεί κακόβουλη χρήση του δικτύου αν κάποιος προσποιούμενος Client της κοινότητας ή ένοικος της πολυκατοικίας καταφέρει να αποκτήσει πρόσβαση σε Things υποκλέβοντας δεδομένα. Η λύση σε αυτά τα θέματα είναι ο κεντρικός έλεγχος πρόσβασης (federated access control / provisioning). Στον κεντρικό έλεγχο πρόσβασης υπάρχει μία συγκεκριμένη αρχή η οποία καθορίζει την πρόσβαση των Clients στα Things. Σε αυτή την περίπτωση η αρχή αυτή φέρει πλήρη ευθύνη του ελέγχου που επιβάλλει. Οι ιδιοκτήτες των Things συμμετέχουν στο δίκτυο με προϋπόθεση την εμπιστοσύνη τους σε αυτή. Υποχρέωση της αρχής είναι να εμπνέει αρκετή εμπιστοσύνη με κίνητρο και σκοπό την επέκταση του δικτύου. Ακόμα και σε αυτή την περίπτωση θα μπορούσε κάποια αρχή, σκόπιμα ή μη, να επιτρέπει την κακόβουλη χρήση του δικτύου. Ωστόσο, σε αντίθεση με τον έλεγχο πρόσβασης στα ίδια τα Things, εδώ οι ιδιοκτήτες τους έχουν να κρίνουν μία μόνο αρχή ελέγχου πρόσβασης, αντί για πολλαπλούς Clients. Μία τέτοια αρχή (καθολικού ελέγχου πρόσβασης) είναι αυτή που περιγράφεται στο XEP-0324: Internet of Things – Provisioning, ως Provisioning Server.

Δύο ακόμα, όχι λιγότερο σημαντικές, απαιτήσεις είναι αυτές του ελέγχου της ανακάλυψης (discovery) και ελέγχου παρουσίας (presence). Σε ένα IoT δίκτυο μπορούν να υπάρχουν πολλά Things καθώς και να προστίθενται νέα συνεχώς. Οι Clients δε μπορούν μόνοι τους να γνωρίζουν ποια Things υπάρχουν στο δίκτυο. Συνεπώς πρέπει οι Clients να μπορούν με κάποιον τρόπο να τα ανακαλύπτουν (discovery). Για αυτό τον σκοπό χρήσιμη είναι η ύπαρξη υπηρεσίας παροχής πληροφοριών σχετικά με τα Things που υπάρχουν. Παρομοίως, οι Clients δεν μπορούν να γνωρίζουν ποια από τα Things είναι την τρέχουσα στιγμή συνδεδεμένα στο δίκτυο και διαθέσιμα. Συνεπώς σκόπιμη είναι η ύπαρξη υπηρεσίας/λειτουργίας όπου δηλώνεται η κατάσταση σύνδεσης των Things στο δίκτυο (π.χ. online/συνδεδεμένο , offline/εκτός σύνδεσης), από την οποία οι Clients μπορούν να ενημερωθούν. Οι δύο παραπάνω υπηρεσίες υλοποιούν διαδικασίες όμοιες μεταξύ τους καθώς και με τον καθολικό έλεγχο πρόσβασης. Η βασική αρχή είναι ότι όλες διατηρούν μία λίστα με τα Things που υπάρχουν στο δίκτυο. Στην περίπτωση του καθολικού ελέγχου πρόσβασης, για κάθε Thing υπάρχει η λίστα με τους επιτρεπόμενους για πρόσβαση Clients, στην περίπτωση του presence υπάρχει η διαθεσιμότητα (online – offline) για κάθε Thing και στην περίπτωση του discovery αρκεί μόνο η λίστα με τα Things. Αυτό το γεγονός οδηγεί στο συμπέρασμα πως η αρχή καθολικού ελέγχου πρόσβασης, μπορεί να προσφέρει και τις υπηρεσίες του discovery και του presence, καλύπτοντας καθολικά τις ανάγκες του δικτύου στο Επίπεδο Διαχείρισης.

3.4.3 Τελική σύγκριση των XEP

Σε αυτό το σημείο και εφόσον έχουν καθοριστεί οι απαιτήσεις υλοποίησης μπορεί να γίνει η τελική σύγκριση των ενδεικτικότερων XEP. Ωστόσο πριν γίνει αυτό σκόπιμη είναι μία ανασκόπηση των απαιτήσεων που ορίστηκαν παραπάνω:

- **Data Plain (Επίπεδο Δεδομένων)**
    - Αμφίδρομη επικοινωνία
    - Επικοινωνία 1 προς n (Publish – Subscribe)
    - Σταθερή και επαρκής προτυποποίηση της δόμησης των IoT δεδομένων
- **Control Plain (Επίπεδο Διαχείρισης)**
    - Access Control (Έλεγχος Πρόσβασης)
    - Provisioning (αρχή κεντρικού ελέγχου πρόσβασης)
    - Discovery (ανακάλυψη)
    - Presence (παρουσία)

Παρομοίως σκόπιμη είναι και η ανασκόπηση των ενδεικτικότερων XEP που παρουσιάστηκαν νωρίτερα στο κεφάλαιο:

- **Internet of Things Specific XEPs**
    - XEP-0323: Sensor Data
    - XEP-0325: Control
    - XEP-0326: Concentrators
    - XEP-0347: Discovery
    - XEP-0324: Provisioning
- **Non – Internet of Things Specific XEPs**
    - XEP-0060: Publish – Subscribe
    - XEP-0045: Multi User Chat

Είναι ήδη γνωστά τα χαρακτηριστικά των παραπάνω XEP, οπότε μπορεί να δημιουργηθεί ο παρακάτω πίνακας σύγκρισης. Σε αυτόν τον πίνακα κάθε γραμμή αντιστοιχεί σε κάθε ένα από τα XEP, ενώ κάθε στήλη αντιστοιχεί σε κάθε μία από τις απαιτήσεις. Μπλε (έντονο) χρώμα έχουν οι γραμμές των οποίων τα XEP εξυπηρετούν ανάγκες που σχετίζονται κυρίως με το επίπεδο δεδομένων, ενώ πράσινο (ανοιχτό) χρώμα έχουν οι γραμμές των οποίων τα XEP εξυπηρετούν ανάγκες που σχετίζονται κυρίως με το επίπεδο διαχείρισης. Στην περίπτωση που το XEP μίας γραμμής καλύπτει την απαίτηση μίας στήλης, στο αντίστοιχο κελί σημειώνεται η ένδειξη X, στην περίπτωση που καλύπτει περιορισμένα την απαίτηση σημειώνεται η ένδειξη O, στην περίπτωση που δεν την καλύπτει σημειώνεται η ένδειξη – και στην περίπτωση που το XEP δεν σχετίζεται με την απαίτηση το κελί είναι κενό.

XEP / Απαίτηση	**Δόμηση IoT Δεδομένων**	**1:n Επικ. (Pub-Sub)**	**Αμφίδρομη Επικοινωνία**	**Discovery**	**Presence**	**Access Control**	**Provisioning**
IoT Sensor Data	**X**	**-** (μόνο 1:1)	**-** (μόνο oneway)			**X**	
IoT Control	**X**	**-** (μόνο 1:1)	**-** (μόνο oneway)			**X**	
IoT Discovery				**X**	**X**	**X**	
IoT Provisioning					**X**		**X**
IoT Concentrators		**O**	**-**	**O**	**O**	**-**	
Publish – Subscribe	**-** (ελεύθερα δεδομένα)	**X**	**X**	**X**	**-**	**X**	
Multi User Chat	**-** (ελεύθερα δεδομένα)	**X**	**X**	**X**	**X**	**X**	

Στο παραπάνω σχήμα διακρίνεται η αδυναμία των Internet of Things Specific XEPs να καλύψουν τις απαιτήσεις που έχουν τεθεί. Συγκεκριμένα η αδυναμία έγκειται στο μοντέλο επικοινωνίας. Το μοντέλο επικοινωνίας σε αυτά είναι το Request – Response με επικοινωνία 1 προς 1 και όχι αμφίδρομη. Αποτελεί γνωστό πως αυτό το μοντέλο είναι ασύμφορο ιδιαίτερα ως προς τα Things και συνεπώς το ίδιο το όραμα του IoT.

Αντίθετα φαίνεται η υπεροχή των μη IoT Specific XEPs (Publish – Subscribe & Multi User Chat) τόσο στο μοντέλο επικοινωνίας, όσο και στις υπόλοιπες δυνατότητες που παρέχουν. Το μοντέλο επικοινωνίας σε αυτά είναι το Publish – Subscribe και η επικοινωνία είναι αμφίδρομη. Ταυτόχρονα δίνονται οι δυνατότητες:

- Access Control: μέσω αποδοχής ως Publisher ή Subscriber από τον Owner στο Publish – Subscribe και μέσω των Affiliation και των Roles στο Multi User Chat
- Discovery: μέσω της ενημέρωσης από τον Server για τα ριζικά nodes στο Publish – Subscribe και μέσω της ενημέρωσης με τη λίστα των rooms αλλά και τη λίστα με τους Participants στο room από τον Server στο Multi User Chat
- Presence: μόνο στο Multi User Chat μέσω της λίστας των Participant στο room που παρέχει πληροφορίες διαθεσιμότητας (online / offline)

Συνεπώς διακρίνεται ένα πλεονέκτημα του Multi User Chat έναντι του Publish – Subscribe όσον αφορά το Presence. Ταυτόχρονα το Multi User Chat επιτρέπει την ανάπτυξη μίας ευκόλως διαχειριζόμενης αρχιτεκτονικής επικοινωνίας χρησιμοποιώντας την έννοια του Room.

Ένα μη IoT Specific XEP προσφέρει το επιπλέον πλεονέκτημα της χρήσης μίας σταθερής και αποδεκτής από την κοινότητα τεχνολογίας. Ωστόσο μόνο με το Multi User Chat δεν έχουν καλυφθεί όλες οι απαιτήσεις. Υπάρχουν δύο βασικά κενά. Το πρώτο έγκειται στο Provisioning καθώς δεν περιγράφεται κάποια κεντρική αρχή που ορίζει τα Affiliations και τα Roles σε όλα τα Rooms, αλλά αυτό (Access Control) γίνεται ξεχωριστά σε κάθε Room από τον ιδιοκτήτη. Επίσης, η μορφή των μηνυμάτων που στέλνονται μέσα στο Room δεν είναι καθορισμένη/προτυποποιημένη. Αυτό σημαίνει πως τα μηνύματα μπορούν να πάρουν οποιαδήποτε μορφή όπως απλό κείμενο, XMPP Stanzas ή ακόμα και άλλες μορφές κωδικοποίησης.

Για την κάλυψη των παραπάνω απαιτήσεων μπορούν να χρησιμοποιηθούν στοιχεία των IoT Specific XEPs. Συγκεκριμένα στο IoT Provisioning περιγράφεται επαρκώς η έννοια του Provisioning Server και μερικά από τα μηνύματα/stanzas που μπορούν να χρησιμοποιηθούν. Μία τέτοια οντότητα θα μπορούσε να χρησιμοποιηθεί για να δημιουργεί και να διαχειρίζεται το Access Control στα Rooms του Multi User Chat. Παρομοίως, τα IoT Sensor Data και IoT Control παρέχουν μία ολοκληρωμένη πρόταση, από πλήρως προτυποποιημένους τύπους μηνυμάτων/stanzas, τα οποία καλύπτουν κάθε ανάγκη σήμανσης δεδομένων αισθητήρων και εντολών ελέγχου (από και προς τα Things). Τέτοια stanzas μπορούν να ενσωματωθούν ως περιεχόμενο στα μηνύματα που στέλνονται στο Multi User Chat, αξιοποιώντας τη δυνατότητα ενσωμάτωσης ειδικά διαμορφωμένων stanzas που δίνεται από το XMPP.

Οι συγκρίσεις αυτές οδηγούν και στα τελικά δομικά XEP της λύσης προς υλοποίηση. Χάριν πληρότητας, τα χαρακτηριστικά των τελευταίων συγκρίσεων παρουσιάζονται και στον πίνακα παρακάτω. Οι τελικές επιλογές είναι το Multi User Chat με χρήση της έννοιας του Provisioning Server από το IoT Provisioning και των Stanzas των IoT Sensor Data και IoT Control. Ο κορμός αυτής της λύσης, το Multi User Chat, αποτελεί σταθερό και αποδεκτό από

την κοινότητα πρότυπο μοντέλου επικοινωνίας, ενώ η χρήση στοιχείων από τα IoT Specific XEPs προσθέτει τα συγκεκριμένα στοιχεία που χρειάζονται για τη χρήση του μοντέλου αυτού στον κόσμο του IoT. Ο συνδυασμός αυτός καλύπτει το σύνολο των απαιτήσεων, που τέθηκαν σύμφωνα με τις σύγχρονες ανάγκες και προκλήσεις που αντιμετωπίζει το IoT, καθώς και με γνώμονα τη δημιουργία μίας λύσης ικανής για αποδεκτή και ευρεία ανάπτυξη.

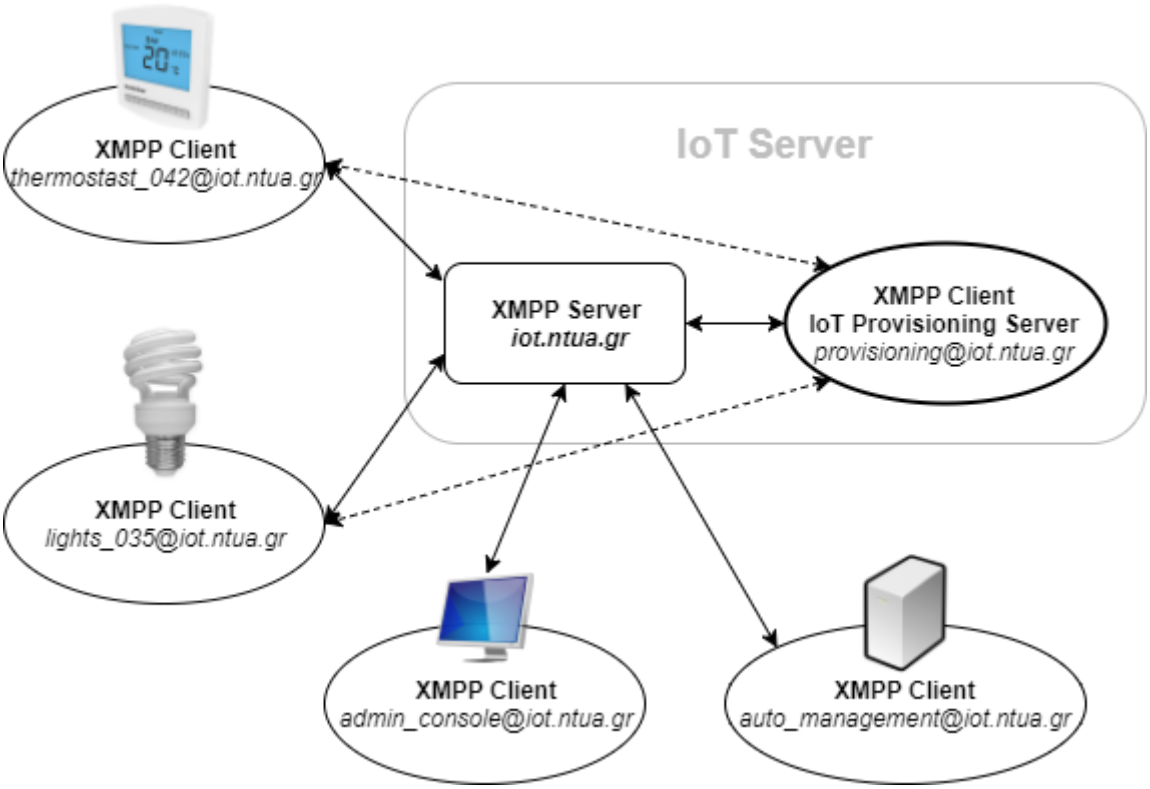
XEP / Απαίτηση	**Δόμηση IoT Δεδομένων**	**1:n Επικ. (Pub-Sub)**	**Αμφίδρομη Επικοινωνία**	**Discovery**	**Presence**	**Access Control**	**Provisioning**
IoT XEPs	**X**	-	-	**X**	**X**	**X**	**X**
Publish - Subscribe	-	**X**	**X**	**X**	-	**X**	
Multi User Chat	-	**X**	**X**	**X**	**X**	**X**	
Multi User Chat + IoT Provisioning + IoT Sensor & Control Stanzas	**X**	**X**	**X**	**X**	**X**	**X**	**X**

3.4.4 Παρουσίαση της λύσης

Στην προηγούμενη ενότητα προσδιορίστηκαν τα δομικά στοιχεία της λύσης προς υλοποίηση. Ωστόσο δεν έχει προσδιοριστεί η ακριβής χρήση τους σε συνδυασμό για τη λειτουργία ενός IoT δικτύου.

Στην υλοποίηση του δικτύου τα Things, οι Clients καθώς και ο Provisioning Server αποτελούν XMPP Clients, οι οποίοι συνδέονται στον XMPP Server. Ωστόσο, ο Provisioning Server μπορεί ‘’τρέχει’’ μαζί με τον XMPP Server καθώς αποτελεί κεντρικό παράγοντα του δικτύου. Αυτό βέβαια αποτελεί λεπτομέρεια της υλοποίησης και βασικό στοιχείο είναι πως ο Provisioning Server διαχειρίζεται από κάποια έμπιστη αρχή. Θα μπορούσαν στην πραγματικότητα να υπάρχουν περισσότεροι από έναν Provisioning Servers, όπου ο καθένας διαχειρίζεται το δικό του δίκτυο από Things, ενώ τα Things εγγράφονται σε αυτόν που εμπιστεύονται περισσότερο. Η τεχνική αυτή οδηγεί στην ιδέα των δικτύων εμπιστοσύνης που είχε αναφερθεί προηγουμένως. Η σύνδεση των Things, Clients και του Provisioning Server ως XMPP Clients με τον XMPP Server φαίνεται στο παρακάτω ενδεικτικό σχήμα. Με

συνεχείς γραμμές φαίνονται οι συνδέσεις στον XMPP Server ενώ με διακεκομμένες η επικοινωνία των Things με τον Provisioning Server.



Βασικά στοιχεία του δικτύου αποτελούν ο Provisioning Server και το Multi User Chat (MUC). Με το Multi User Chat δημιουργούνται rooms (δωμάτια) στα οποία συνδέονται Things. Κάθε Thing συνδέεται μόνο σε ένα δωμάτιο ώστε να επικοινωνεί με τον εξωτερικό κόσμο μόνο μέσω αυτού. Οι Clients μπορούν να συνδεθούν με ένα ή περισσότερα δωμάτια για να αποκτήσουν πρόσβαση στα Things που υπάρχουν σε αυτά.

Τα rooms παρέχουν Access Control (έλεγχο πρόσβασης) τον οποίο χειρίζεται ο Owner (ιδιοκτήτης) του κάθε room. Αρμόδιος για τη διαχείριση του Access Control στα rooms πρέπει να είναι μόνο ο Provisioning Server. Συνεπώς, ο Provisioning Server είναι αυτός που δημιουργεί, αποτελεί Owner και διαχειρίζεται το Access Control των rooms. Στη συνέχεια θα παρουσιαστούν συνοπτικά τα δύο βασικά σενάρια χρήσης: σύνδεση ενός thing στο IoT δίκτυο και σύνδεση ενός Client στο IoT δίκτυο.

Σύνδεση ενός Thing στο IoT δίκτυο:

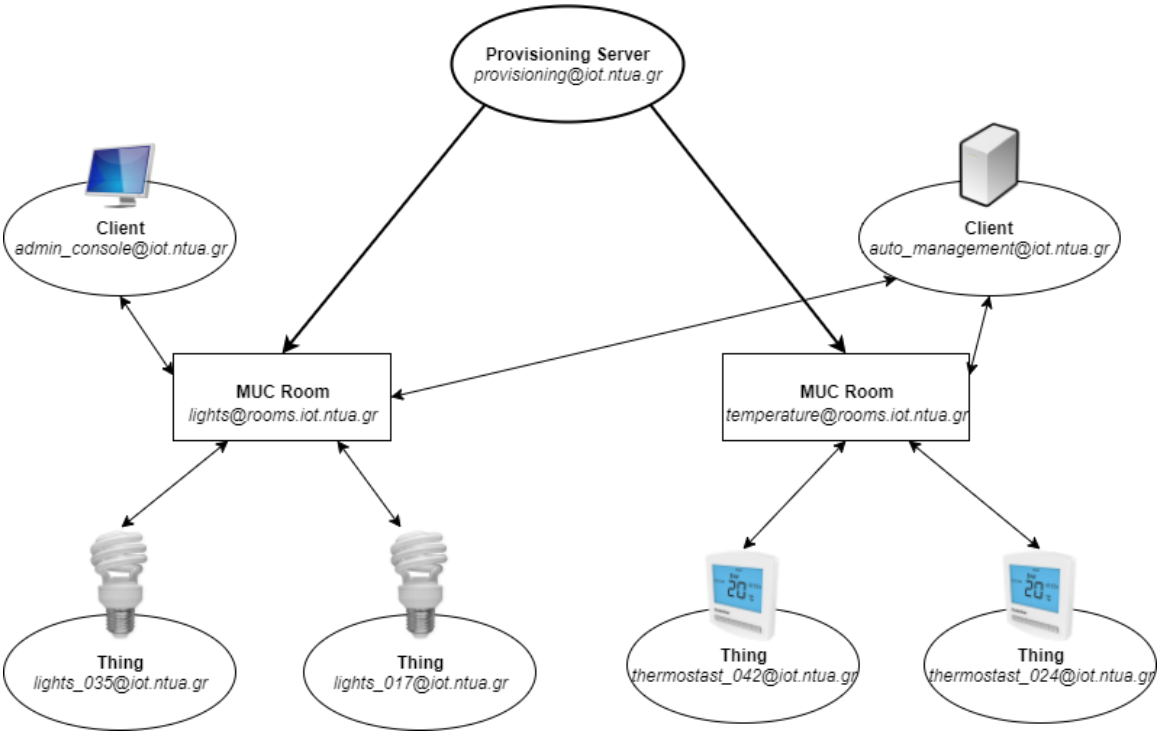
- το Thing συνδέεται ως XMPP Client στον XMPP Server με το δικό του JID
- το Thing ζητάει τη διεύθυνση του room που πρέπει να εισέλθει μέσω ειδικού αιτήματος στον Provisioning Server

- ο Provisioning Server διατηρεί μία λίστα με rooms και τα Things τους. Αν υπάρχει το Thing στη λίστα τότε ανακτά τη διεύθυνση του δωματίου. Αν δεν υπάρχει τότε δημιουργεί, μέσω της MUC υπηρεσίας του XMPP Server, νέο room του οποίου αποτελεί Owner και δίνει δικαίωμα πλήρης επικοινωνίας του Thing σε αυτό. Σε κάθε περίπτωση αποκρίνεται στο Thing με τη διεύθυνση ενός room στο οποίο πρέπει να εισέλθει
- το Thing λαμβάνοντας τη διεύθυνση του room του, εισέρχεται σε αυτό
- το Thing μπορεί πλέον να επικοινωνήσει με το IoT δίκτυο, στέλνοντας δεδομένα και λαμβάνοντας εντολές μόνο προς και από αυτό το room

Σύνδεση ενός Client στο IoT δίκτυο:

- ο Client συνδέεται ως XMPP Client στον XMPP Server με το δικό του JID
- ο Client ζητάει τη λίστα με τα διαθέσιμα rooms μέσω αιτήματος στη MUC υπηρεσία του XMPP Server
- η MUC υπηρεσία αποκρίνεται με τη λίστα όλων των διαθέσιμων rooms
- ο Client επιλέγει το room στο οποίο επιθυμεί να εισέλθει και στέλνει αίτημα εισόδου
- το room διαθέτει λίστα συμμετεχόντων. Τη λίστα αυτή διαχειρίζεται ο Provisioning Server. Ο Client μπορεί είτε να βρίσκεται στη λίστα με δικαίωμα πλήρης επικοινωνίας (λήψη και αποστολή μηνυμάτων), είτε με δικαίωμα μόνο λήψης μηνυμάτων (ως visitor), είτε να μη βρίσκεται καθόλου στη λίστα
- εφόσον ο Client βρίσκεται στη λίστα, του επιτρέπεται η είσοδος στο room και μπορεί πλέον να αλληλοεπιδράσει με τα Things που υπάρχουν σε αυτό, ανάλογα με τα δικαιώματά του (λήψη μόνο δεδομένων ή και αποστολή εντολών). Σε περίπτωση που δε βρίσκεται στη λίστα λαμβάνει ενδεικτική απόκριση από το room (τύπου Access Denied)
- ο Client μπορεί εφόσον το επιθυμεί να συνδεθεί και με άλλα rooms

Ένα ενδεικτικό παράδειγμα της λειτουργικής τοπολογίας του δικτύου με τα Things, τους Clients, τα rooms και τον Provisioning Server φαίνεται στο παρακάτω σχήμα. Διακρίνεται πως ο Provisioning Server έχει και διαχειρίζεται δύο rooms (σύνδεση με έντονη γραμμή): το *lights@rooms.iot.ntua.gr* και το *temperature@rooms.iot.ntua.gr*. Σε καθένα από τα δύο rooms συνδέονται και επικοινωνούν Things, στο πρώτο σχετικά με τον φωτισμό και στο δεύτερο σχετικά με τον κλιματισμό ενός κτηρίου. Στα rooms αυτά μπορούν να συνδεθούν και να επικοινωνήσουν Clients. Ο Client *auto_management@iot.ntua.gr*, που αποτελεί λογισμικό αυτόματης διαχείρισης, μπορεί να συνδεθεί και στα δύο rooms, έχοντας πρόσβαση και στον φωτισμό και στον κλιματισμό. Ωστόσο, ο *admin_console@iot.ntua.gr* μπορεί να συνδεθεί μόνο στο *lights@rooms.iot.ntua.gr*, έχοντας πρόσβαση μόνο στον φωτισμό.

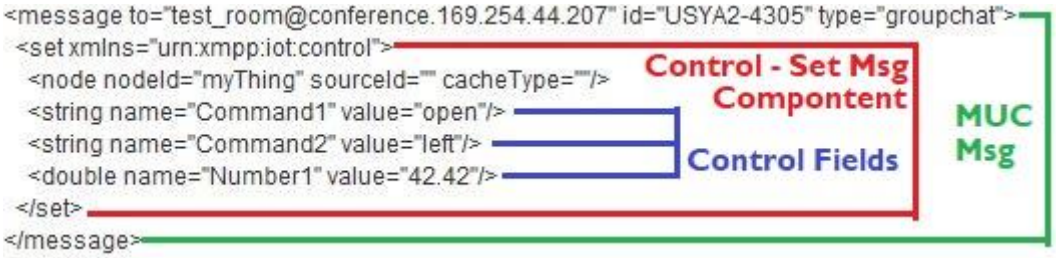


Μέχρι στιγμής έχει περιγραφεί πλήρως η σύνδεση των οντοτήτων (Things & Clients) στο IoT δίκτυο και στα rooms του ώστε να μπορούν να επικοινωνήσουν. Ωστόσο δεν έχει γίνει αναφορά για τα μηνύματα που θα στέλνονται, δηλαδή για τη ‘’γλώσσα επικοινωνίας’’. Το σώμα μηνυμάτων που στέλνονται μέσα στα MUC rooms μπορεί να είναι είτε απλό κείμενο, είτε κάποιο κομμάτι από XMPP Stanza. Σε αυτή την λύση τα μηνύματα που στέλνονται έχουν τη μορφή που περιγράφεται στα IoT Sensor Data & IoT Control XEPs. Τα Things στέλνουν τα δεδομένα τους με χρήση των μηνυμάτων που περιγράφονται στο Sensor Data XEP έχοντας τη δυνατότητα να ορίσουν χρονική στιγμή, είδος δεδομένων, μονάδα κλπ. Ένα παράδειγμα τέτοιου μηνύματος αποτελεί το παρακάτω.

```
<message to="test_room@conference.169.254.44.207" id="USYA2-143" type="groupchat">
 <fields xmlns="urn:xmpp:iot:sensordata" seqnr="0" done="true">
  <node nodeId="myThing" sourceId="" cacheType="">
   <timestamp value="2018-02-24T14:02:33.037+00:00">
    <numeric name="temprature" value="25.631682187226566"/>
    <boolean name="rains" value="false"/>
    <string name="msg" value="hello from thing"/>
   </timestamp>
  </node>
 </fields>
</message>
```


Διακρίνεται ότι μέσα στο σώμα του μηνύματος του MUC υπάρχει το component με όνομα Fields που ορίζεται στο Sensor Data XEP. Μέσα σε αυτό το component διακρίνονται τα πεδία δεδομένων στα οποία ορίζεται τύπος δεδομένων, όνομα και τιμή. Κάθε φορά που ένα Thing στέλνει δεδομένα, αυτά λαμβάνονται μέσω του MUC από όλους τους Clients του room.

Παρομοίως, οι Clients στέλνουν τις εντολές τους μέσω των μηνυμάτων που ορίζονται στο Control XEP. Παρακάτω φαίνεται ένα παράδειγμα τέτοιου μηνύματος.



Διακρίνεται πως στο σώμα του MUC μηνύματος, υπάρχει το Set component που ορίζεται στο Control XEP. Μέσα στο component αυτό υπάρχουν εντολές ελέγχου με τύπο δεδομένων, όνομα και τιμή.

Η χρήση αυτής της μορφής μηνυμάτων δεδομένων και ελέγχου προσφέρει ευελιξία καθώς υπάρχει πρόβλεψη για οποιαδήποτε παραμετροποίηση (τύπο δεμένων, μονάδα μέτρησης κλπ). Ωστόσο το πιο σημαντικό είναι πως αυτή η μορφή δεδομένων υπόκειται σε ανοιχτή προτυποποίηση συμβάλλοντας στη διαλειτουργικότητα.

Συνοψίζοντας τελικά τη βασική λειτουργία του δικτύου: τα Things και οι Clients εισέρχονται στο XMPP δίκτυο ως XMPP Clients. Ο Provisioning Server (μπορεί να είναι και πολλοί) εισέρχεται και αυτός ως XMPP Client, δημιουργεί και διαχειρίζεται το access control στα MUC rooms και παρέχει σε κάθε Thing τη διεύθυνση του room του. Κάθε Thing εισέρχεται σε ένα μόνο room και εκεί αποστέλλει τα δεδομένα του με τη μορφή των μηνυμάτων του XEP Sensor Data. Οι Clients μπορούν να ενημερωθούν για την λίστα με τα rooms από τη MUC υπηρεσία του XMPP Server και να εισέλθουν, εφόσον είναι αποδεκτοί, σε ένα ή περισσότερα rooms. Μέσα σε κάθε room, οι Clients λαμβάνουν τα δεδομένα των Things που υπάρχουν σε αυτό και εφόσον έχουν το δικαίωμα μπορούν να στείλουν εντολές ελέγχου προς Things, με τη μορφή των μηνυμάτων που ορίζεται στο XEP Control.

4 Ενδεικτικές υλοποιήσεις

4.1 Εισαγωγή - Επισκόπιση των στόχων της υλοποίησης

Στο προηγούμενο κεφάλαιο επιλέχθηκαν οι λεπτομέρειες μίας λύσης για ένα δίκτυο IoT με τη χρήση του πρωτοκόλλου XMPP, το οποίο ανταποκρίνεται στις σύγχρονες ανάγκες και προκλήσεις. Σκοπός πλέον είναι η ενδεικτική υλοποίηση ενός τέτοιου δικτύου.

Κύριος στόχος της υλοποίησης είναι η ανάπτυξη μίας πλατφόρμας λογισμικού που παρέχει όλες τις βασικές λειτουργίες του δικτύου. Μία τέτοια πλατφόρμα, πρέπει να καθιστά εύκολη και γρήγορη την ενσωμάτωσή της για τη δημιουργία Things, Clients καθώς και του Provisioning Server, προσφέροντας ένα απλό αλλά ευέλικτο Application Programming Interface (API – Διεπαφή Προγραμματισμού Εφαρμογών).

Δεύτερος στόχος της υλοποίησης είναι η ανάπτυξη δοκιμαστικών Things, Clients και ενός Provisioning Server για δοκιμή λειτουργίας της πλατφόρμας. Σε επόμενο στάδιο θα μπορούν να φτιαχτούν πιο εύχρηστοι Clients (π.χ. desktop / web / mobile εφαρμογές κλπ), να χρησιμοποιηθούν πραγματικά Things (π.χ. microcontrollers – αισθητήρες / έξυπνοι θερμοστάτες / φώτα / ρομπότ κλπ) καθώς και να αναπτυχθεί κάποιου είδους Graphical User Interface (GUI – γραφική διεπαφή χρήστη) για τον χειρισμό του Provisioning Server. Πλέον η πλατφόρμα θα είναι έτοιμη για χρήση σε πραγματικά σενάρια IoT δικτύων.

4.2 Βασικά δομικά στοιχεία της υλοποίησης

Για την ανάπτυξη της υλοποίησης απαραίτητα είναι τα κατάλληλα δομικά στοιχεία. Αυτά είναι τα στοιχεία που επιτρέπουν τη λειτουργία ενός XMPP δικτύου και την ανάπτυξη εφαρμογών σε αυτό. Τα δύο αυτά στοιχεία είναι ο XMPP Server και οι XMPP Clients. Στόχος είναι η χρήση στοιχείων που παρέχουν πληρέστερα βασικές λειτουργίες που χρειάζονται καθώς και επιτρέπουν την ανάπτυξη νέων. Στην συνέχεια θα παρουσιαστούν οι απαιτήσεις και η τελική επιλογή για καθένα από αυτά τα δύο.

4.2.1 XMPP Server

Όπως σχεδόν σε κάθε άλλη ανοιχτή διαδικτυακή τεχνολογία, έτσι και στο XMPP, υπάρχουν έτοιμοι Servers που υποστηρίζουν όλες τις βασικές λειτουργίες. Ένας XMPP Server παρέχει βασικές λειτουργίες ανταλλαγής μηνυμάτων, παρουσίας και δρομολόγησης XML και μπορεί να χρησιμοποιηθεί είτε για χρήση στο διαδίκτυο, είτε σε τοπικά / ιδιωτικά δίκτυα [35]. Εκτός από τις βασικές λειτουργίες, υπάρχουν και συμπληρωματικές. Για παράδειγμα υπάρχουν XEP που ορίζουν υπηρεσίες που λειτουργούν ως κομμάτι του Server (Server Component). Τέτοιες υπηρεσίες μπορεί είτε να παρέχονται, είτε όχι, είτε να υπάρχει δυνατότητα ανάπτυξης / εγκατάστασης νέων Server Components.

Στην περίπτωση της συγκεκριμένης υλοποίησης, συμπληρωματική λειτουργία αποτελεί το Multi User Chat. Αυτό απαιτεί την ύπαρξη ειδικού Server Component. Συνεπώς μία βασική απαίτηση για τον XMPP Server που θα χρησιμοποιηθεί είναι η υποστήριξη για το Multi User Chat. Η επίσημη λίστα από Servers που παρέχεται από το https://xmpp.org αν και όχι μεγάλη, παρέχει της πληρέστερες επιλογές, οι οποίες καλύπτουν κάθε γνωστό λειτουργικό σύστημα.

Ο Server που επιλέχθηκε είναι ο Openfire της Ignite Realtime (Open Source κοινότητα). Όπως και κάθε στοιχείο που θα χρησιμοποιηθεί σε αυτή την εργασία, είναι ανοιχτού κώδικα και συγκεκριμένα κάτω από την Open Source Apache License [36]. Ο Openfire είναι από τους πιο δημοφιλείς XMPP Servers καθώς παρέχει τα παρακάτω πλεονεκτήματα:

- είναι εύκολος στην εγκατάσταση και την διαχείριση του
- υποστηρίζει αρκετές συμπληρωματικές υπηρεσίες ή την δημιουργία και εγκατάσταση νέων
- μπορεί να εγκατασταθεί σε όλα τα δημοφιλή λειτουργικά συστήματα (Linux / macOS / Solaris / Windows) καθώς είναι γραμμένος σε Java

Βασικό πλεονέκτημα είναι πως υλοποιείται και υποστηρίζεται πλήρως το Multi User Chat. Επιπρόσθετα η κοινότητα Ignite Realtime, αποτελείται από σημαντικά μέλη της κοινότητας του XMPP και παρέχει πλήρη ποικιλία από λογισμικό και βιβλιοθήκες. Σε αυτή ανήκει και η βιβλιοθήκη Smack που θα χρησιμοποιηθεί για τη δημιουργία XMPP Clients (θα παρουσιαστεί στην συνέχεια).

4.2.2 XMPP Clients και βιβλιοθήκες κώδικα

Η ανάπτυξη της υλοποίησης αφορά καθολικά τη δημιουργία συγκεκριμένων XMPP Clients. Αυτό ισχύει καθώς κάθε μία από τις οντότητες προς ανάπτυξη (Things, Clients, Provisioning Server) αποτελεί XMPP Client που συνδέεται στο δίκτυο παρέχοντας συγκεκριμένες λειτουργίες. Στη λίστα του διαθέσιμου λογισμικού παρέχονται είτε έτοιμοι XMPP Clients, είτε βιβλιοθήκες για ενσωμάτωση και δημιουργία νέων.

Οι έτοιμοι XMPP Clients μπορούν να χρησιμοποιηθούν για τη βασική χρήση του δικτύου παρέχοντας τις απλές δυνατότητες του Instant Messaging και της παρουσίας/διαθεσιμότητας, συνήθως μέσα από ένα απλό γραφικό περιβάλλον. Αν και χρήσιμοι για δοκιμαστικούς σκοπούς, οι έτοιμοι XMPP Clients, δεν μπορούν να αποτελέσουν τις οντότητες που περιεγράφηκαν, πόσο μάλλον να ενσωματωθούν σε συσκευές όπως τα Things.

Από την άλλη μεριά, οι βιβλιοθήκες για τη δημιουργία XMPP Clients είναι αυτό που χρειάζεται για την ανάπτυξη των IoT οντοτήτων καθώς επιτρέπουν την ανάπτυξη προσαρμοσμένων λειτουργιών. Οι βασικές λειτουργίες της σύνδεσης στο δίκτυο, της αποστολής μηνυμάτων και της παρουσίας/διαθεσιμότητας παρέχονται συνήθως έτοιμες από την εκάστοτε βιβλιοθήκη. Ωστόσο, όπως και στην περίπτωση του XMPP Server, υπάρχουν απαιτήσεις για συμπληρωματικά στοιχεία που είτε είναι αναγκαία, είτε μπορούν να προσφέρουν ευκολίες.

Βασική απαίτηση είναι η υποστήριξη του Multi User Chat XEP καθώς το IoT δίκτυο προς υλοποίηση στηρίζεται σε αυτό. Ο Provisioning Server δημιουργεί και διαχειρίζεται rooms, ενώ οι Clients και τα Things εισέρχονται και ανταλλάσσουν δεδομένα σε αυτά. Στα rooms υπάρχει πλήρες access control μέσα από τα Affiliations και τα Roles, τα οποία διαχειρίζεται ο Provisioning Server. Όλες αυτές οι ενέργειες πρέπει να είναι διαθέσιμες από την βιβλιοθήκη

ή έστω να είναι διαθέσιμη η υλοποίηση των μηνυμάτων και διαδικασιών που ορίζονται από το XEP ώστε να δημιουργηθούν.

Μία δεύτερη απαίτηση είναι η ύπαρξη των βασικών μηνυμάτων των XEP IoT Sensor Data και IoT Control. Τα μηνύματα που αφορούν την μεταφορά δεδομένων θα χρειαστούν για τη μορφοποίηση των δεδομένων των Things και των εντολών ελέγχου των Clients. Σημειώνεται πως και σε αυτή την περίπτωση, η μη (ή ελλιπής) ύπαρξη τους δημιουργεί την απαίτηση έστω κάποιου τρόπου δημιουργίας τους. Παρόμοια είναι και η περίπτωση των μηνυμάτων μεταξύ των Things και του Provisioning Server, τα οποία αν και δεν ορίζονται από κάποιο XEP, υπάρχουν παρόμοια στο XEP IoT Provisioning.

Λιγότερο σημαντικές απαιτήσεις είναι η εύκολη χρήση της βιβλιοθήκης, η γλώσσα προγραμματισμού να είναι αντικειμενοστραφής και να υποστηρίζεται από πληθώρα συσκευών καθώς και η κοινότητα και η υποστήριξη που μπορεί να παρέχει.

Συνοψίζοντας λοιπόν τις παραπάνω απαιτήσεις για τη βιβλιοθήκη ανάπτυξης των XMPP Client:

- πλήρης υποστήριξη του Multi User Chat XEP
- υποστήριξη των μηνυμάτων των IoT Sensor Data και IoT Control XEPs
- δυνατότητα δημιουργίας και χρήσης νέων προσαρμοσμένων μηνυμάτων (XMPP Stanzas)
- αντικειμενοστραφής γλώσσα προγραμματισμού που υποστηρίζεται από πληθώρα συσκευών
- ευρεία κοινότητα και υποστήριξη

Αυτές οι απαιτήσεις ικανοποιούνται στο σύνολό τους από τη βιβλιοθήκη Smack της Ignite Realtime. Επιπλέον πλεονέκτημα αποτελεί πως η βιβλιοθήκη αυτή αναπτύχθηκε και υποστηρίζεται από την Open Source κοινότητα που ανέπτυξε τον Openfire Server, οπότε υπάρχει πλήρης συμβατότητα μεταξύ τους. Η Smack, ως βιβλιοθήκη δημιουργίας XMPP Clients, αποτελεί το βασικό κομμάτι γύρω από το οποίο θα αναπτυχθεί η υλοποίηση, συνεπώς στη συνέχεια θα γίνει πλήρης παρουσίασή της.

4.2.3 Εισαγωγή στη βιβλιοθήκη Smack

Η Smack αποτελεί ανοιχτού κώδικα βιβλιοθήκη για την δημιουργία XMPP Clients. Είναι γραμμένη σε γλώσσα Java και συνεπώς μπορεί να ενσωματωθεί σε cross-platform εφαρμογές καθώς και εφαρμογές Android δημιουργώντας από απλούς έως και σύνθετους XMPP Clients [37].

Βασικό της συστατικό είναι η καλώς δομημένη αντικειμενοστραφή οργάνωση του κώδικά της. Η βάση (core) του XMPP είναι πλήρως έτοιμη για χρήση μέσω απλών και εύχρηστων αντικειμένων και μεθόδων. Παράλληλα ωστόσο υποστηρίζονται πολλά XEP (επεκτάσεις) αλλά και η δημιουργία νέων. Τα XEP που υποστηρίζονται είναι φτιαγμένα ώστε να μπορούν να χρησιμοποιηθούν με ευκολία, ενώ για τη δημιουργία νέων προσφέρεται ένα καλώς δομημένο περιβάλλον μέσα από το οποίο είναι δυνατός ο ορισμός προσαρμοσμένων μηνυμάτων (Stanzas) καθώς και κομματιών μηνυμάτων (Stanza Components). Μέρος των

XEP που παρέχονται έτοιμα, είναι και το Multi User Chat για το οποίο υπάρχει πλήρης υποστήριξη όλων των δυνατοτήτων του. Τα IoT XEPs και συγκεκριμένα τα μηνύματα του Sensor Data και Control παρέχονται και αυτά, ωστόσο με αρκετούς περιορισμούς, ακόμα και λάθη τα οποία όμως θα διορθωθούν. Η διαδικασία δημιουργίας νέων μηνυμάτων θα χρησιμοποιηθεί έτσι κι αλλιώς εξάλλου για τα μηνύματα μεταξύ του Provisioning Server και των Things.

Επίσης όπως έχει ήδη αναφερθεί, η Ignite Realtime αποτελεί μεγάλη Open Source κοινότητα με πολλά μέλη από την ευρεία κοινότητα του XMPP και συνεπώς παρέχεται πλήρης υποστήριξη.

Συνεπώς με όλα τα παραπάνω η βιβλιοθήκη Smack καλύπτει όλες τις απαιτήσεις που παρουσιάστηκαν προηγουμένως.

4.3 Ανάπτυξη επέκτασης της βιβλιοθήκης Smack

Σε αυτό το μέρος θα παρουσιαστεί το τεχνικό κομμάτι της ανάπτυξης λογισμικού για την υλοποίηση του IoT δικτύου που σχεδιάστηκε. Βασικότερα κομμάτια αυτής της υλοποίησης είναι οι οντότητες των Things, Clients καθώς και του Provisioning Server που αποτελούν XMPP Clients. Βασικός σκοπός είναι η ανάπτυξη μίας πλατφόρμας λογισμικού η οποία μπορεί να ενσωματωθεί σε εφαρμογές παρέχοντας έτοιμες τις παραπάνω οντότητες. Ύστερα η χρήση αυτών των οντοτήτων μπορεί να γίνει κάτω από οποιονδήποτε έτοιμο XMPP Server (με την προϋπόθεση ότι αυτός υποστηρίζει την υπηρεσία Multi User Chat). Οπότε, η δημιουργία μίας τέτοιας πλατφόρμας αφορά την ανάπτυξη βιβλιοθήκης κώδικα που αποτελεί επέκταση της βιβλιοθήκης Smack και υλοποιεί αυτές τις οντότητες.

Για την πλήρη κατανόηση ολόκληρου του project τόσο σε επίπεδο αρχιτεκτονικής, όσο και σε επίπεδο πηγαίου κώδικα, αρχικά θα γίνει μία παρουσίαση χρήσης της βιβλιοθήκης Smack σε βασικά σενάρια που είναι απαραίτητα, όπως η απλή σύνδεση στον XMPP Server, η αποστολή μηνυμάτων, η δημιουργία custom τύπων μηνυμάτων και η δημιουργία, διαχείριση και χρήση των MUC rooms. Στη συνέχεια θα γίνει ανάλυση της αρχιτεκτονικής, των σεναρίων χρήσης και της ανάπτυξης καθεμίας εκ των τριών οντοτήτων (Things, Clients και Provisioning Server).

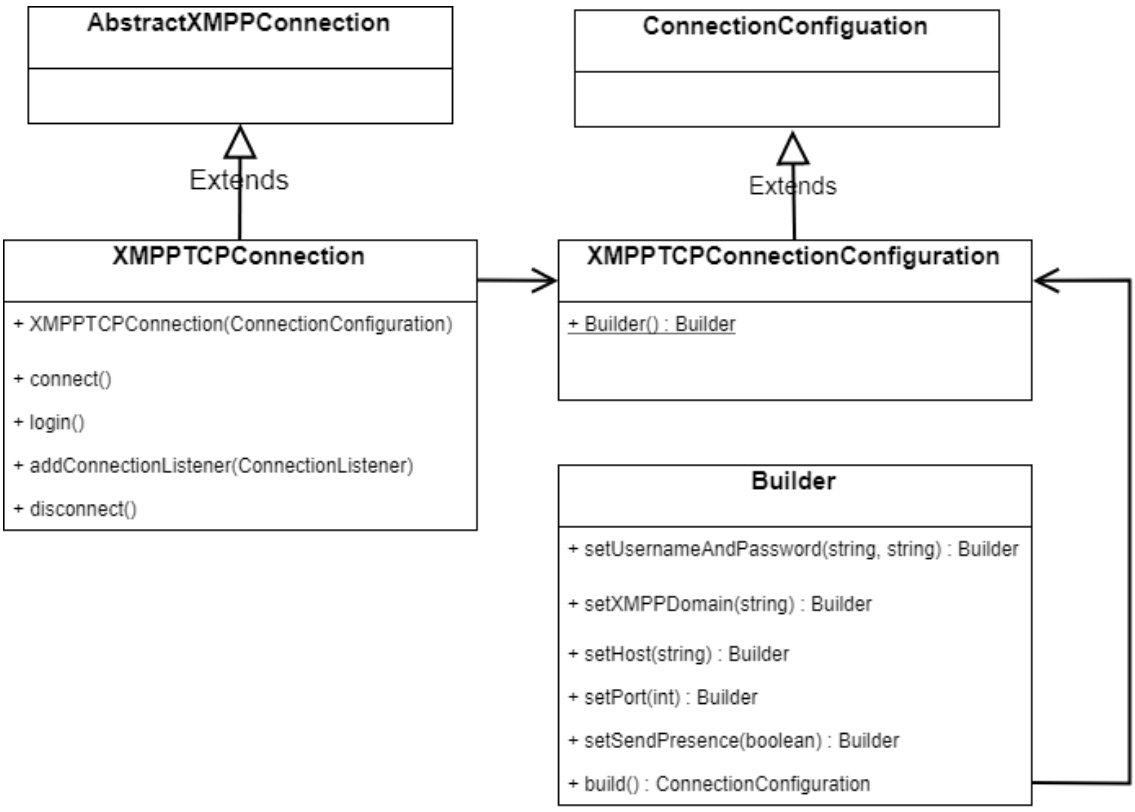
4.3.1 Βασικές περιπτώσεις χρήσης της βιβλιοθήκης Smack

Η βιβλιοθήκη Smack παρέχει ένα καλώς δομημένο και εύκολο στην χρήση API για τη διαχείριση τόσο των βασικών όσο και των πιο σύνθετων λειτουργιών ενός XMPP Client. Ωστόσο για την καλή κατανόηση απαραίτητη είναι η καλή γνώση των βασικών μοντέλων αρχιτεκτονικής λογισμικού και αντικειμενοστραφούς προγραμματισμού, καθώς επικρατεί μέσα στη σχεδίαση της βιβλιοθήκης. Τα βασικά σενάρια χρήσης που θα παρουσιαστούν σε αυτό το σημείο είναι απαραίτητα και θα χρησιμοποιηθούν ως δομικά στοιχεία στην τελική υλοποίηση.

4.3.1.1 Σύνδεση στον XMPP Server

Η πρώτη βασική λειτουργία ενός XMPP Client είναι η σύνδεση του σε έναν XMPP Server κάτω από μία διεύθυνση Jid. Ο XMPP Client χρησιμοποιεί ένα *username* και ένα *password* για να συνδεθεί στον XMPP Server ο οποίος έχει κάποιο *domain*. Εφόσον ο XMPP Client στείλει το *username* και το *password* και ο XMPP Server επαληθεύσει τα στοιχεία αυτά, ο XMPP Client είναι συνδεδεμένος στο δίκτυο με το Jid του να είναι *username@domain*.

Παρακάτω φαίνεται το Class Diagram (Διάγραμμα Κλάσεων) των κλάσεων που θα χρησιμοποιηθούν σε αυτό το σενάριο.



Μία σύνδεση XMPP Client χαρακτηρίζεται από ένα αντικείμενο AbstractXMPPConnection και συγκεκριμένα XMPPTCPConnection αν η σύνδεση είναι TCP. Η δημιουργία ενός τέτοιου αντικειμένου χρειάζεται ένα XMPPTCPConnectionConfiguration το οποίο παρέχει όλες τις παραμέτρους σύνδεσης. Το XMPPTCPConnectionConfiguration δημιουργείται μέσω του static του αντικειμένου, Builder, το οποίο παρέχει όλες τις απαραίτητες μεθόδους για τη ρύθμιση παραμέτρων σύνδεσης, κάποιες από τις οποίες είναι οι εξής :

- setUsernameAndPassword(string Username, string Password) : ορισμός του username και του Password
- setXMPPDomain(string Domain) και setHost(string Host) : ορισμός του domain και του host αντίστοιχα του XMPP Server
- setPort(int Port) : ορισμός της TCP θύρας σύνδεσης στον XMPP Server
- setSendPresence(Boolean sendPresence) : ορισμός (ή μη) κατάστασης διαθεσιμότητας με την είσοδο στο δίκτυο

- build() : αυτή η μέθοδος επιστρέφει το τελικό αντικείμενο XMPPTCPConnectionConfiguration

Εφόσον δημιουργηθεί ένα αντικείμενο XMPPTCPConnectionConfiguration, αυτό μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός αντικειμένου XMPPTCPConnection το οποίο ορίζει και τη σύνδεση με τον XMPP Server.

Το αντικείμενο XMPPTCPConnection παρέχει κάποιες βασικές μεθόδους για τη διαχείριση της σύνδεσης, κάποιες από τις οποίες είναι οι εξής :

- connect() : σύνδεση με τον XMPP Server. Η πρώτη μέθοδος που πρέπει να κληθεί για τη δημιουργία της σύνδεσης
- login() : δημιουργία και αποστολή αιτήματος σύνδεσης στο XMPP δίκτυο με τα στοιχεία (username, password) που έχουν δηλωθεί στο ConnectionConfiguration. Πρέπει να κληθεί αφού έχει γίνει σύνδεση με τη μέθοδο connect().
- addConnectionListener(ConnectionListener) : προσθήκη αντικειμένου που υλοποιεί το interface (διεπαφή) ConnectionListener, δηλαδή ενός αντικειμένου το οποίο καλείται/ειδοποιείται κάθε φορά που συμβαίνει κάποιο γεγονός σύνδεσης (έγινε σύνδεση, έγινε login, έκλεισε η σύνδεση, έκλεισε η σύνδεση με σφάλμα, γίνεται επανασύνδεση, επιτυχής επανασύνδεση, αποτυχημένη επανασύνδεση)
- disconnect() : αποσύνδεση από τον XMPP Server

Όλα τα παραπάνω συνοψίζονται στο παρακάτω απόσπασμα κώδικα Java :

```
protected AbstractXMPPConnection connection;
private XMPPTCPConnectionConfiguration conConfig;

//Create ConnectionConfiguration object using Builder
conConfig = XMPPTCPConnectionConfiguration.builder()
                                .setUsernameAndPassword("tasos","12")
                                .setXmppDomain("iot.ntua.gr")
                                .setHost("iot.ntua.gr")
                                .setPort(5222);
                                .setSendPresence(true)
                                .build();

/* Create XMPPTCPConnection object using the ConnectionConfiguration
object */
connection = new XMPPTCPConnection(conConfig);

//Add a ConnectionListener
connection.addConnectionListener(conListener);

//Connect and Login to the XMPP Server
connection.connect();
connection.login();

...

//Disconnect from the XMPP Server
connection.disconnect();
```

Στο απόσπασμα αυτό ο χρήστης με Jid tasos@iot.ntua.gr και password 12, πραγματοποιεί σύνδεση στον XMPP Server με domain iot.ntua.gr και στη συνέχεια αποσυνδέεται.

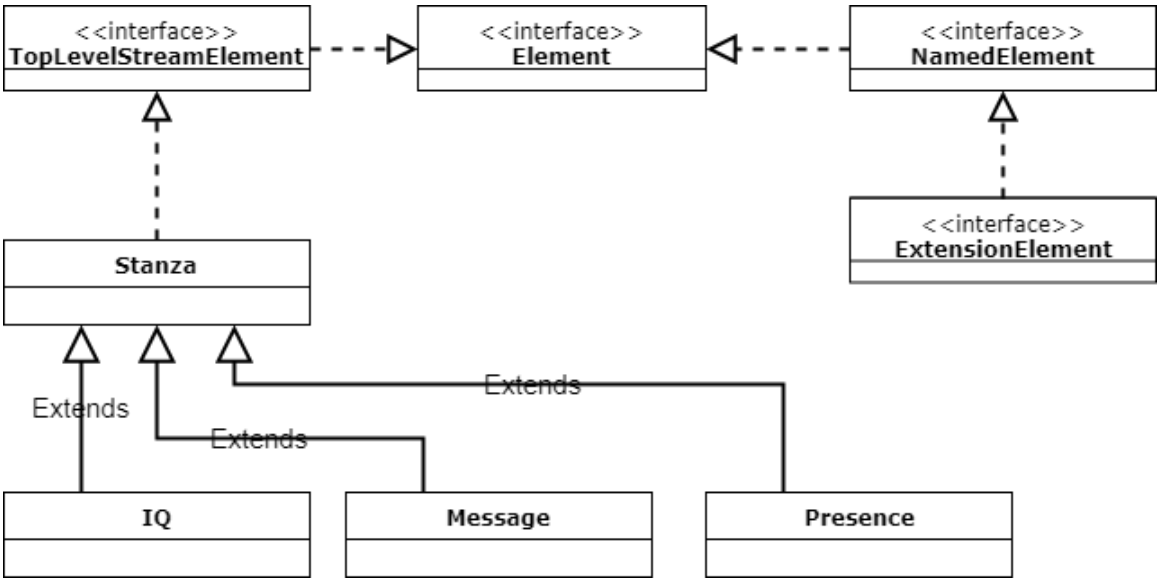
4.3.1.2 Δημιουργία custom μηνυμάτων - extensions

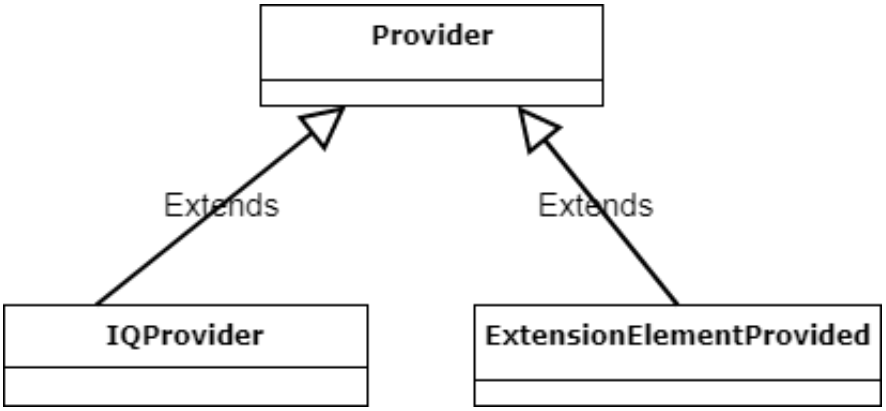
Βασικό στοιχείο της βιβλιοθήκης Smack είναι το περιβάλλον που παρέχει για την ανάπτυξη νέων προσαρμοσμένων Stanzas (μηνυμάτων) ή Stanza Components (κομματιών μηνυμάτων). Η δυνατότητα αυτή καθίσταται ιδιαίτερα χρήσιμη στην περίπτωση της συγκεκριμένης υλοποίησης καθώς τέτοια στοιχεία θα αναπτυχθούν τουλάχιστον για την επικοινωνία των Clients με τον Provisioning Server.

Τα γενικά αντικείμενα / κλάσεις που χρησιμοποιούνται για όλα τα μηνύματα και τα μέρη τους είναι τα εξής:

- **Element** : χαρακτηρίζει/αναπαριστά ένα XML element (στοιχείο XML). Συνήθως οι κλάσεις αυτές παρέχουν εύκολο τρόπο δημιουργίας ενός τέτοιου αντικειμένου μέσα από τον Constructor αλλά και μεθόδους που επιτρέπουν την εισαγωγή άλλων Elements σε αυτό. Πάντα πρέπει να παρέχεται η μέθοδος toXML() που παράγει το τελικό XML κείμενο. Συνεπώς οι κλάσεις αυτές επιτρέπουν τη δημιουργία και δόμηση δομών XML με χρήση μόνο των αντικειμένων τους. Στο τελικό στάδιο της αποστολής ενός μηνύματος αρκεί να χρησιμοποιηθεί το τελικό Element που αναπαριστά ολόκληρο το Stanza και η βιβλιοθήκη Smack (αναδρομικά μέσω των μεθόδων toXML() του κάθε Element) μεταφράζει το Element σε XML κείμενο το οποίο και στέλνεται.
- **Provider** : εκτελεί την αντίστροφη διαδικασία από το Element, δηλαδή αποτελεί το στοιχείο το οποίο κάνει parse (ανάλυση / αποκωδικοποίηση) ενός κειμένου XML σε ένα αντικείμενο Element. Κάθε Provider αντιστοιχίζεται σε κάποιο Element και πρέπει οπωσδήποτε να παρέχει τη μέθοδο parse η οποία δέχεται το XML και επιστρέφει το αντίστοιχο Element. Επιπρόσθετα, κάθε Provider πρέπει να δηλώνεται στην κλάση ProviderManager ώστε να μπορεί να χρησιμοποιηθεί αυτόματα από το Smack στο κομμάτι της λήψης μηνυμάτων.

Οι παραπάνω δύο κλάσεις είναι γενικές, δηλαδή χαρακτηρίζουν γενικά κάποιο στοιχείο XML. Ωστόσο, δε θα μπορούσε να μη χρησιμοποιηθεί το στοιχείο της επέκτασης και της κληρονομικότητας που παρέχει μία αντικειμενοστραφής γλώσσα προγραμματισμού όπως η Java. Συνεπώς, καθεμία από τις κλάσεις έχει αρκετές επεκτάσεις. Στην πραγματικότητα κάθε Element (και Provider) μπορεί να επεκταθεί, δημιουργώντας κλάση που χαρακτηρίζει μία πιο συγκεκριμένη περίπτωση στοιχείου XML και έτσι δημιουργείται ένα δέντρο επεκτάσεων. Στη συνέχεια φαίνεται το βασικό διάγραμμα κλάσεων του Element και ύστερα του Provider. Σημειώνεται πως αυτές δεν είναι όλες οι κλάσεις που παρέχονται από την βιβλιοθήκη αλλά υπάρχει πληθώρα στοιχείων που καλύπτουν τουλάχιστον τα βασικά μηνύματα για κάθε XEP που παρέχεται.





Διακρίνεται λοιπόν πως τα Elements διακρίνονται σε TopLevelStreamElements και σε NamedElements. Τα TopLevelStreamElements όπως δηλώνει και το όνομα επεκτείνονται στην κλάση Stanza, η οποία επεκτείνεται στις κλάσεις IQ, Message και Presence (τα 3 είδη Stanza του XMPP). Από την άλλη πλευρά το NamedElement αναπαριστά οποιοδήποτε κομμάτι XML που χρησιμοποιείται μέσα στο Stanza. Το ExtensionElement αποτελεί ένα NamedElement που έχει συγκεκριμένο όνομα και namespace. Τα ExtensionElements χρησιμοποιούνται κυρίως στις επεκτάσεις καθώς τα XEP δηλώνουν δικά τους κομμάτια μηνυμάτων με συγκεκριμένο όνομα και namespace που χαρακτηρίζει το ίδιο το XEP.

Η δημιουργία λοιπόν ενός προσαρμοσμένου Stanza ή Stanza Component απαιτεί την επέκταση της κατάλληλης κλάσης Element για τη δημιουργία νέας και ακολούθως του αντίστοιχου Provider (εφόσον αυτό χρειάζεται). Στη συνέχεια για τη χρήση αυτού του μηνύματος αρκεί να χρησιμοποιηθεί το αντίστοιχο νέο Element (ή να ενσωματωθεί σε κάποιο Stanza) και να γίνει αποστολή του Stanza Element. Αντίστοιχα για την υποστήριξη της λήψης του προσαρμοσμένου νέου μηνύματος, αρκεί να δηλωθεί ο Provider στον ProviderManager και πλέον τα αντίστοιχα μηνύματα ή στοιχεία θα μπορούν να αποκωδικοποιηθούν αυτόματα στα αντίστοιχα Element.

Στο παρακάτω απόσπασμα κώδικα φαίνεται ένα παράδειγμα επέκτασης της κλάσης IQ για τη δημιουργία είδους IQ Stanza με όνομα joinRequest:

```
public class IoTJoinRequest extends IQ {

    public static final String ELEMENT = "joinRequest";
    public static final String NAMESPACE =
Constants.IOT_PROVISIONING_NAMESPACE;

    private final NodeInfo nodeInfo;

    public IoTJoinRequest(NodeInfo node) {
        super(ELEMENT, NAMESPACE);
        nodeInfo = node;
    }

    public NodeInfo getNodeInfo(){
      return nodeInfo;
    }

    @Override
    protected IQChildElementXmlStringBuilder
getIQChildElementBuilder(IQChildElementXmlStringBuilder xml) {
        NodeElement nodeElement = new NodeElement(nodeInfo);
        xml.rightAngleBracket();
        xml.append(nodeElement.toXML());
        return xml;
    }

}
```

Στο παραπάνω παράδειγμα το Element IoTJoinRequest αναπαριστά ένα IQ Stanza με όνομα joinRequest και έχει το namespace που ορίζει το IoT Provisioning XEP. Το μοναδικό στοιχείο που έχει αυτό το IQ είναι ένα XML Element με το όνομα NodeInfo. Συνεπώς κατά τη δημιουργία ενός αντικειμένου της κλάσης αυτής πρέπει να δοθεί ένα αντικείμενο της κλάσης NodeInfo. Αντίστοιχα υπάρχει μέθοδος που επιστρέφει το αντικείμενο NodeInfo, για την περίπτωση που το joinRequest έχει ληφθεί και είναι επιθυμητή η ‘’ανάγνωσή’’ του. Τέλος γίνεται override (επικάλυψη) της μεθόδου IQChildElementXmlStringBuilder η οποία αποτελεί μέρος της διαδικασίας μετατροπής της κλάσης στο αντίστοιχο XML κείμενο. Στη μέθοδο αυτή γίνεται η σωστή εισαγωγή του NodeInfo στο σώμα του IQ Stanza.

Στη συνέχεια παρουσιάζεται απόσπασμα κώδικα για τη δημιουργία του αντίστοιχου Provider για το στοιχείο joinRequest.

```
public class IoTJoinRequestProvider extends IQProvider<IoTJoinRequest> {

	@Override
	public IoTJoinRequest parse(XmlPullParser parser, int initialDepth)
throws Exception {
		NodeInfo nodeInfo = null;
		outerloop: while (true) {
			final int eventType = parser.next();
			final String name = parser.getName();
			switch (eventType) {
			case XmlPullParser.START_TAG:
				if (name.equals("node")){
					String nodeId = parser.getAttributeValue(null,
"nodeId");
					String sourceId = parser.getAttributeValue(null,
"sourceId");
					nodeInfo = new NodeInfo(nodeId,sourceId,"");
				}
				break;
			case XmlPullParser.END_TAG:
				if (parser.getDepth() == initialDepth) {
					break outerloop;
				}
				break;
			}
		}
		return new IoTJoinRequest(nodeInfo);
	}

}
```

Σε αυτή την περίπτωση γίνεται ξεκάθαρο ότι γίνεται επέκταση της κλάσης IQProvider και συγκεκριμένα για την κλάση IoTJoinRequest. Εδώ γίνεται override της μεθόδου parse η οποία κάνει parse το XML κείμενο και επιστρέφει ένα αντικείμενο IoTJoinRequest. Στη μέθοδο αυτή γίνεται αναζήτηση για το στοιχείο NodeInfo στο XML και μόλις αυτό βρεθεί δημιουργείται το αντίστοιχο αντικείμενο, ώστε στο τέλος αυτό να χρησιμοποιηθεί για τη δημιουργία του αντικειμένου IoTJoinRequest το οποίο και τελικά θα επιστραφεί.

Για τη δήλωση του Provider στον ProviderManager υπάρχει αυτοματοποιημένο σύστημα (σε περίπτωση που δεν επιθυμείται η δυναμική του δήλωση μέσα στον κώδικα). Αρκεί η συμπλήρωση της αντίστοιχης εγγραφής μέσα στο αρχείο smack.providers που υπάρχει στον φάκελο META-INF του αρχείου JAR. Όλοι οι Providers που ορίζονται σε κάθε τέτοιο αρχείο δηλώνονται αυτόματα στον ProviderManager κατά την εκκίνηση της βιβλιοθήκης. Χάριν πληρότητας παρακάτω φαίνεται το απόσπασμα της δήλωσης του IoTJoinRequestProvider σε ένα τέτοιο αρχείο.

```
<iqProvider>
	<elementName>joinRequest</elementName>
	<namespace>urn:xmpp:iot:provisioning</namespace>
	<className>sm01.IoTProviders.IoTJoinRequestProvider</className>
</iqProvider>
```

Διακρίνεται ότι η δήλωση περιέχει το όνομα και το namespace του XML στοιχείου, καθώς και το όνομα της κλάσης του αντίστοιχου Provider.

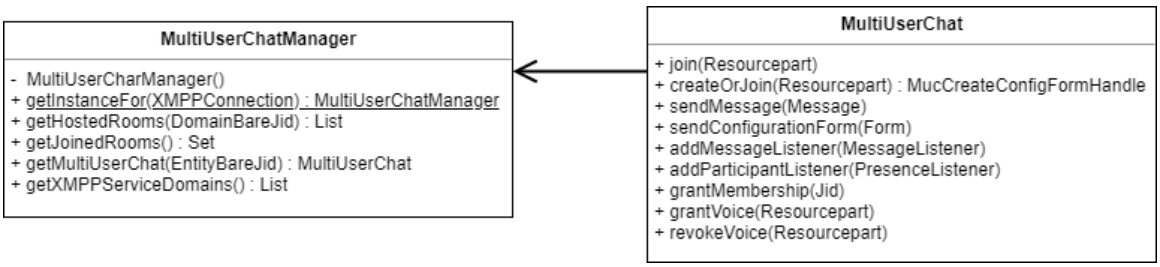
4.3.1.3 MUC rooms: Δημιουργία, σύνδεση και διαχείριση

Ένα από το πιο βασικά στοιχεία στο οποίο βασίζεται η λύση προς υλοποίηση είναι το Multi User Chat. Ο Provisioning Server δημιουργεί και διαχειρίζεται τα rooms, ενώ τα Things και οι Clients εισέρχονται σε αυτά και επικοινωνούν.

Στο Smack οι βασικές κλάσεις για τη χρήση του MUC είναι οι εξής:

- **MultiUserChatManager** : αφορά τη διαχείριση γενικών λειτουργιών του MUC όπως σύνδεση σε δωμάτιο
- **MultiUserChat** : αντιπροσωπεύει ένα δωμάτιο και παρέχει μεθόδους για όλες τις απαραίτητες λειτουργίες μέσα σε αυτό

Παρακάτω φαίνεται το αντίστοιχο διάγραμμα κλάσεων:



Όπως φαίνεται στο παραπάνω διάγραμμα η κλάση MultiUserChatManager είναι Singleton, δηλαδή για την ανάκτηση ενός αντικειμένου της δε μπορεί να χρησιμοποιηθεί ο constructor της, αλλά η static μέθοδός της getInstanceFor. Συνεπώς η static μέθοδος getInstanceFor με παράμετρο ένα αντικείμενο XMPPConnection που αντιστοιχεί στην σύνδεση με τον XMPP Server, επιστρέφει ένα αντικείμενο MultiUserChatManager προς χρήση.

Παρομοίως, η κλάση MultiUserChat δεν έχει προσβάσιμο constructor. Αντικείμενα MultiUserChat μπορούν να ανακτηθούν μόνο μέσω ενός MultiUserChatManager, μέσω της μεθόδου getMultiUserChat, η οποία δέχεται ως παράμετρο το Jid του room. Σημειώνεται ότι αυτή η μέθοδος πρέπει να κληθεί ακόμα και στην περίπτωση που δεν έχει δημιουργηθεί το room και στην συνέχεια να κληθεί η κατάλληλη μέθοδος του MultiUserChat για την δημιουργία του.

Εκτός από τις παραπάνω βασικές για τη χρήση μεθόδους και οι δύο κλάσεις παρέχουν διάφορες άλλες, μερικές από τις οποίες φαίνονται στο παραπάνω διάγραμμα και περιγράφονται παρακάτω:

- **MultiUserChatManager**
  - getXMPPServiceDomains() : επιστρέφει μία λίστα με τα Jid των υπηρεσιών MUC του XMPP Server
  - getHostedRooms(DomainBareJid) : επιστρέφει μία λίστα με τα rooms που υπάρχουν στην υπηρεσία MUC του XMPP Server, η οποία έχει το Jid που δίνεται ως παράμετρος

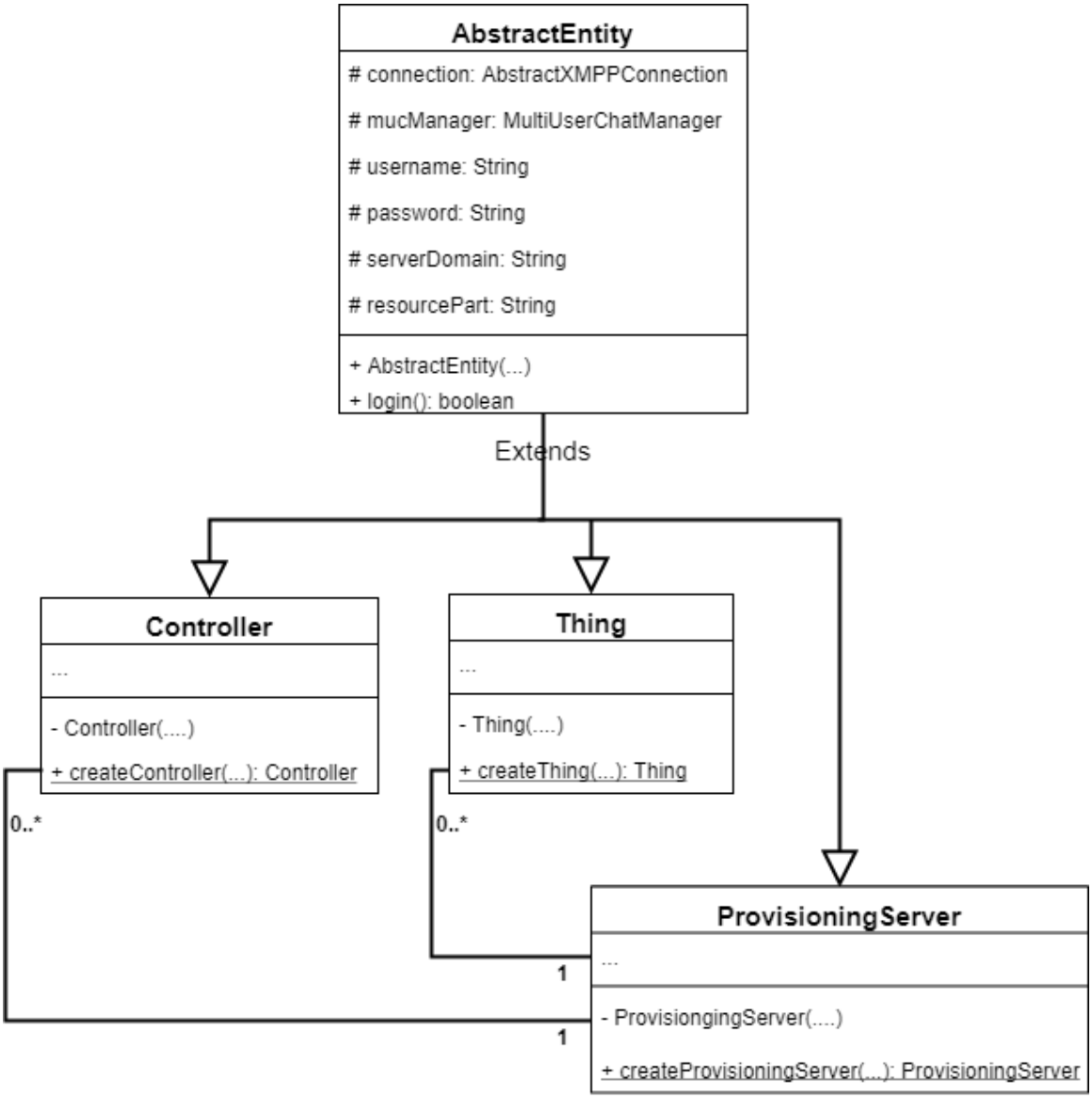
- getJoinedRooms() : επιστρέφει ένα σετ (λίστα) με τα rooms στα οποία έχει γίνει είσοδος

- **MultiUserChat**
  - addParticipantListener(PresenceListener) : προστίθεται ο PresenceListener που δίνεται ως παράμετρος. Έτσι ο listener αυτός καλείται κάθε φορά που κάποιος εισέρχεται ή εξέρχεται από το δωμάτιο. Μπορεί να προστεθεί πριν την είσοδο στο δωμάτιο ώστε κατά την είσοδο να υπάρξει αρχική ενημέρωση
  - addMessageListener(MessageListener) : προστίθεται ο MessageListener που δίνεται ως παράμετρος. Αυτός ο listener καλείται κάθε φορά που υπάρχει νέο μήνυμα στο δωμάτιο, με το μήνυμα αυτό
  - join(Resourcepart) : γίνεται είσοδος στο δωμάτιο με τη χρήση του Nickname που δίνεται ως παράμετρος. Σε περίπτωση που δεν επιτραπεί η είσοδος γίνεται throw συγκεκριμένο exception
  - sendMessage(Message) : στέλνεται στο δωμάτιο το μήνυμα (Message Stanza) που δίνεται ως παράμετρος. Υπάρχει και εκδοχή της ίδιας μεθόδου που παίρνει ως παράμετρο απλό κείμενο / string αντί για Message Stanza
  - createOrJoin(Resourcepart) : γίνεται είσοδος στο δωμάτιο στην περίπτωση που έχει ήδη δημιουργηθεί και επιστρέφεται null, ενώ σε διαφορετική περίπτωση δημιουργείται νέο και επιστρέφεται ένα αντικείμενο MucCreateConfigFormHandle που περιέχει τις επιλογές συμπλήρωσης φόρμας επιλογών δωματίου
  - sendConfigurationForm(Form) : αποστέλλει τη φόρμα με τις επιλογές / ρυθμίσεις του δωματίου. Μπορεί να εκτελεστεί μόνο από τον Owner ή τον Admin
  - grantMembership(Jid) : δίνει δικαίωμα εισόδου στον Client με Jid που δίνεται ως παράμετρος. Μπορεί να εκτελεστεί μόνο από τον Owner ή τον Admin
  - grantVoice(Resourcepart) : δίνει δικαίωμα αποστολής μηνυμάτων στο μέλος του δωματίου που έχει το Nickname που δίνεται ως παράμετρος. Μπορεί να εκτελεστεί μόνο από τους Owner / Admin ή Moderator
  - revokeVoice(Resourcepart) : αφαιρεί το δικαίωμα αποστολής μηνυμάτων στο μέλος του δωματίου που έχει το Nickname που δίνεται ως παράμετρος. Σημειώνεται ότι το μέλος μπορεί να βλέπει ακόμα τα μηνύματα του δωματίου. Μπορεί να εκτελεστεί μόνο από τους Owner / Admin ή Moderator

4.3.2 Γενική σχεδίαση και ανάπτυξη της κλάσης AbstractEntity

Η τρέχουσα υλοποίηση απαιτεί την ανάπτυξη των τριών βασικών οντοτήτων του IoT δικτύου. Αυτές οι οντότητες είναι τα Things, οι Clients και ο Provisioning Server. Κάθε μία από τις οντότητες μπορεί να αποτελέσει μία κλάση που προσφέρει τις απαραίτητες μεθόδους για τη χρήση τους σε εφαρμογές. Ωστόσο και οι τρεις αυτές οντότητες έχουν κάποια βασικά κοινά στοιχεία. Αρχικά αποτελούν XMPP Client με δικό τους Jid, εισέρχονται στον XMPP Server και διαχειρίζονται τη σύνδεσή τους. Έπειτα, όλες χρησιμοποιούν την υπηρεσία Multi User Chat του Server.

Αυτά τα κοινά στοιχεία οδηγούν στην ανάπτυξη μίας abstract (αφηρημένης) κλάσης, η οποία παρέχει αυτές τις βασικές κοινές λειτουργίες, αφήνει περιθώριο βελτίωσης τους και συμβάλει στην καλύτερη αρχιτεκτονική οργάνωση. Οι κλάσεις των τριών οντοτήτων μπορούν να αποτελούν επεκτάσεις της κλάσης αυτής, κληρονομώντας όλες τις βασικές μεθόδους και χαρακτηριστικά της. Η κλάση αυτή θα έχει όνομα AbstractEntity. Παρακάτω παρουσιάζεται το γενικό διάγραμμα κλάσης των IoT οντοτήτων.



Οι κλάσεις Thing και ProvisioningServer αντιστοιχούν στις ομώνυμες οντότητες, ενώ η κλάση Controller αντιστοιχεί στον Client. Όπως έχει περιγραφεί παραπάνω οι κλάσεις των τριών οντοτήτων επεκτείνουν την κλάση AbstractEntity η οποία παρέχει τις βασικές λειτουργίες σύνδεσης ενός XMPP Client. Επιπρόσθετα, διακρίνεται πως οι κλάσεις των τριών οντοτήτων έχουν ιδιωτικό constructor και αντικείμενά τους δημιουργούνται από static μεθόδους. Αυτό μπορεί να φανεί χρήσιμο ειδικά σε περιπτώσεις που επιθυμείται να υπάρχει καθολικός έλεγχος της βιβλιοθήκης στους XMPP Clients που δημιουργούνται (π.χ. καθορισμός ορίου των instances). Τέλος, παρατηρούνται οι σχέσεις των Things και Clients (κλάση Controller) με τον Provisioning Server οι οποίες είναι πολλά προς ένα.

Τώρα που είναι γνωστός ο γενικός αρχιτεκτονικός σχεδιασμός, μπορεί να αναλυθεί η κλάση AbstractEntity που υλοποιεί τις βασικές λειτουργίες σύνδεσης ενός XMPP Client. Για τον σκοπό αυτό θα παρουσιαστούν τα βασικά σενάρια χρήσης (use case scenarios).

Use Case Scenario 1: Δημιουργία αντικειμένου AbstractEntity

1. Κλήση του constructor με τη χρήση username, password, resource part και server domain
2. Αποθήκευση όλων των παραπάνω παραμέτρων σε μεταβλητές της κλάσης
3. Δημιουργία Jid της μορφής username@serverDomain
4. Αποθήκευση του Jid σε μεταβλητή της κλάσης

Το παραπάνω Use Case Scenario είναι αρκετό για τη δημιουργία του αντίστοιχου κώδικα για τη μέθοδο constructor της κλάσης AbstractEntity, ο οποίος φαίνεται παρακάτω.

```
private String jidStr;
protected String username;
protected String password;
protected String serverDomain;
protected String resourcePart;
protected AbstractXMPPConnection connection;
private XMPPTCPConnectionConfiguration conConfig;
protected MultiUserChatManager mucManager;
protected static Jid provisioningServer;

public AbstractEntity(String user, String pass, String domain, String
resource) throws Exception{
      username = user;
      password = pass;
      serverDomain = domain;
      resourcePart = resource;
      if((!JidUtil.isValidEntityBareJid(user+"@"+domain))||(pass.equals
("")||(domain.equals(""))) throw new Exception("Wrong credentials");
}
```

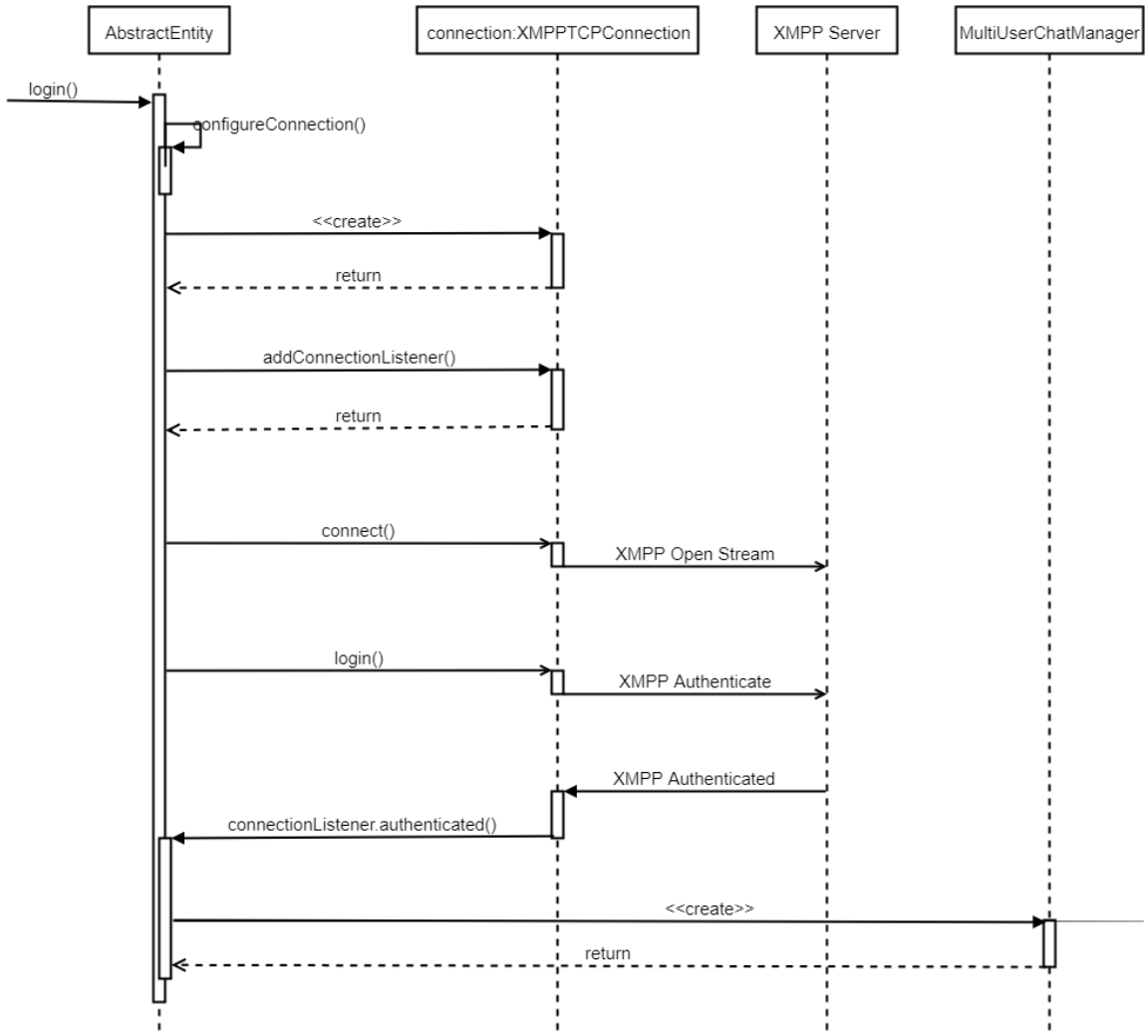
Στο παραπάνω κώδικα φαίνεται η προσθήκη ελέγχου για την εγκυρότητα των στοιχείων username και server domain, ώστε να δημιουργείτε έγκυρο Jid. Σε περίπτωση που δεν δημιουργείται έγκυρο Jid, γίνεται throw ένα γενικό exception. Σε περίπτωση που δεν υπήρχε αυτός ο έλεγχος το exception θα γινόταν throw από τη βιβλιοθήκη Smack αργότερα κατά την προσπάθεια σύνδεσης. Ωστόσο, είναι σημαντική η κατά το δυνατό καλύτερη διευθέτηση ελέγχων και σφαλμάτων σε κάθε περίπτωση.

Use Case Scenario 2: Σύνδεση με τον XMPP Server

1. Το αντικείμενο AbstractEntity δημιουργεί ένα αντικείμενο connectionConfiguration με τη χρήση των username, password, resource part και server domain που έχουν δοθεί κατά τη δημιουργία του
2. Το αντικείμενο connectionConfiguration χρησιμοποιείται για τη δημιουργία ενός αντικειμένου XMPPConnection
3. Στο αντικείμενο XMPPConnection προστείθεται ένα αντικείμενο connectionListener για την ειδοποίηση του AbstractEntity για συμβάντα της σύνδεσης
4. Καλείται η μέθοδος connect() του XMPPConnection
5. Το αντικείμενο XMPP Connection στέλνει αίτημα δημιουργίας XMPP Stream στον XMPP Server
6. Ο XMPP Server αποκρίνεται ανοίγοντας το XMPP Stream με τον XMPP Client (αντικείμενο XMPPConnection του AbstractEntity)
7. Καλείται η μέθοδος login() του XMPPConnection
8. Το αντικείμενο XMPP Connection στέλνει αίτημα εισόδου με ταυτοποίηση στοιχείων του στον XMPP Server
9. Ο XMPP Server αποκρίνεται θετικά αποστέλλοντας το αντίστοιχο μήνυμα στο XMPPConnection
10. Το XMPPConnection λαμβάνοντας τη θετική απάντηση του Server ενημερώνει το AbstractEntity, καλώντας τη μέθοδο authenticated() του connectionListener που έχει δοθεί
11. Το AbstractEntity γνωρίζοντας πως η σύνδεση έχει πραγματοποιηθεί δημιουργεί ένα αντικείμενο MultiUserChatManager χρησιμοποιώντας το XMPPConnection
12. Η σύνδεση πλέον έχει ολοκληρωθεί πλήρως και το αντικείμενο AbstractEntity διαθέτει ένα αντικείμενο MultiUserChatManager για τη χρήση της υπηρεσίας MUC

Σημειώνεται πως καθένα από τα βήματα αυτά έχουν παρουσιαστεί αναλυτικά προηγουμένως. Οι διαδικασίες που ακολουθούνται σε αυτή την περίπτωση είναι αυτές που απαιτούνται για τη βασική σύνδεση που θα χρειαστούν και οι τρεις οντότητες του IoT δικτύου (Things, Clients και Provisioning Server).

Η δημιουργία του αντικειμένου connectionConfiguration απαιτεί τη χρήση του static αντικειμένου Builder της ίδιας της κλάσης. Το αντικείμενο Builder προσφέρει πολλαπλές μεθόδους για την εισαγωγή παραμέτρων και αρκετές από αυτές είναι απαραίτητες. Συνεπώς για τη σωστή οργάνωση του κώδικα, η διαδικασία αυτή εκτελείται σε ξεχωριστή μέθοδο, την configureConnection() η οποία δημιουργεί το αντίστοιχο αντικείμενο connectionConfiguration και το αποθηκεύει σε μεταβλητή της κλάσης. Επιπρόσθετα το αντικείμενο connectionListener επιλέχθηκε να είναι ορισμένο μέσα στην κλάση AbstractEntity ώστε να υπάρχει η δυνατότητα άμεσης αναφοράς στις μεταβλητές της. Οι παρατηρήσεις αυτές αφορούν περισσότερο τον πηγαίο κώδικα που δίνεται πιο μετά. Παρακάτω φαίνεται το Sequence Diagram (ακολουθιακό διάγραμμα) όλου του σεναρίου.



Στο παραπάνω διάγραμμα παρατηρείται πως η επικοινωνία με τον XMPP Server, όπως αναμένεται, είναι ασύγχρονη. Οι μέθοδοι login() και connect() του XMPPConnection απλά στέλνουν τα αντίστοιχα μηνύματα στον XMPP Server. Η απόκριση έρχεται σε ύστερο χρόνο και θέτει την κλήση της αντίστοιχης μεθόδου του connectionListener που έχει δηλώσει το AbstractEntity.

Παράλληλα, διακρίνονται οι αντίστοιχες εξαρτήσεις μεταξύ των διαδικασιών που εκτελούνται. Για παράδειγμα το configureConnection() είναι απαραίτητο να προηγηθεί για τη δημιουργία του XMPPConnection.

Ενορχηστρωτής όλων των διαδικασιών είναι η public μέθοδος login() που θα δημιουργηθεί στην κλάση AbstractEntity, την οποία και θα καλούν οι κλάσεις που θα την επεκτείνουν (Thing, Controller, ProvisioningServer). Οποιεσδήποτε πιο συγκεκριμένες λειτουργίες θα υλοποιηθούν σε private (ιδιωτικές) μεθόδους (όπως η configureConnection()). Παρακάτω διακρίνεται το αντίστοιχο απόσπασμα κώδικα όλου του σεναρίου.

```
private boolean configureConnection(){
	if (conConfig!=null) return true;
	try {
		Builder conConfigBuilder =
MPPTCPConnectionConfiguration.builder()
				.setUsernameAndPassword(username, password)
				.setXmppDomain(serverDomain)
				.setHost(serverDomain)
				.setPort(5222);
		if (!resourcePart.equals(""))
			conConfigBuilder =
conConfigBuilder.setResource(resourcePart);
			conConfig = (XMPPTCPConnectionConfiguration)
conConfigBuilder.setSecurityMode(SecurityMode.disabled)
				.setSendPresence(true)
				.build();
			return true;
	} catch (XmppStringprepException e) {
		return false;
	}
}

public boolean login(){
	if (!configureConnection()) return false;
	connection = new XMPPTCPConnection(conConfig);
	connection.addConnectionListener(conListener);
	try {
		connection.connect();
		connection.login();
		return true;
	} catch (Exception e) {
		return false;
	}
}

private ConnectionListener conListener = new ConnectionListener(){

	public void connected(XMPPConnection connection) {}

	public void authenticated(XMPPConnection connection, boolean
resumed) {
		mucManager=MultiUserChatManager.getInstanceFor(connection);
	}

	public void connectionClosed() {}
	public void connectionClosedOnError(Exception e) {}
	public void reconnectionSuccessful() {}
	public void reconnectingIn(int seconds) {}
	public void reconnectionFailed(Exception e) {}

};
```

Με αυτό το απόσπασμα κώδικα ολοκληρώνεται και η περιγραφή της υλοποίησης της κλάσης AbstractEntity. Πλέον οι κλάσεις που θα την επεκτείνουν μπορούν να συντομεύσουν ολόκληρη τη διαδικασία σύνδεσης, χρησιμοποιώντας στον constructor τους, μόνο τις δύο παρακάτω γραμμές κώδικα.

```
super(username, password, server_domain, resource_part);
login();
```

4.3.3 Η ανάπτυξη του Thing Component - Κλάση Thing

Βασικότερη οντότητα της υλοποίησης είναι αυτή του Thing. Η υλοποίηση της οπότε θα παρουσιαστεί πρώτη. Όπως έχει ήδη αναφερθεί κάθε μία από αυτές τις τρεις οντότητες (Thing, Client, Provisioning Server) υλοποιείται ως κλάση η οποία επεκτείνει την κλάση AbstractEntity. Συνεπώς ισχύει κι εδώ το αντίστοιχο διάγραμμα κλάσεων που παρουσιάστηκε στην ενότητα 4.3.2.

Η κλάση Thing έχει private constructor και η δημιουργία ενός αντικειμένου της γίνεται μόνο από static μεθόδους. Στον constructor μπορούν να χρησιμοποιηθούν οι μέθοδοι που κληρονομούνται από την AbstractEntity ώστε να γίνεται άμεση η σύνδεση κατά τη δημιουργία. Η διαχείριση θεμάτων σύνδεσης γίνεται από την AbstractEntity η οποία και μπορεί να βελτιωθεί. Συνεπώς ο κώδικας σχετικά με τη δημιουργία αντικειμένου Thing μπορεί να είναι ο εξής:

```
private NodeInfo nodeInfo;
private MultiUserChat muc;
private ControlListener controlListener = null;

private Thing(String user, String pass, String domain, String resource,
String node) throws Exception {
	super(user, pass, domain, resource);
	nodeInfo = new NodeInfo(node,"","");
	login();
}

public static Thing createThing(String user, String pass, String
domain, String resource, String node){
	try {
		return new Thing(user,pass,domain,resource, node);
	} catch (Exception e) {
		return null;
	}
}
```

Παρατηρούνται επίσης κάποιες private μεταβλητές της κλάσης. Αυτές είναι οι εξής:

- nodeInfo: αποτελεί το αντικείμενο NodeInfo που περιέχει το χαρακτηριστικό όνομα του Thing που θα περιέχεται στα μηνύματα Sensor Data
- muc: αποτελεί το (μοναδικό) room στο οποίο θα αλληλοεπιδρά το Thing
- controlListener: η κλάση αυτή μπορεί να δεχτεί έναν listener τύπου ControlListener για την ενημέρωση μηνυμάτων εντολών από Clients (θα εξηγηθεί στη συνέχεια)

Για καλύτερη κατανόηση της υλοποίησης θα γίνει κι εδώ παρουσίαση των αντίστοιχων Use Case Scenarios, με τα αντίστοιχα διαγράμματα UML και αποσπάσματα κώδικα για το καθένα.

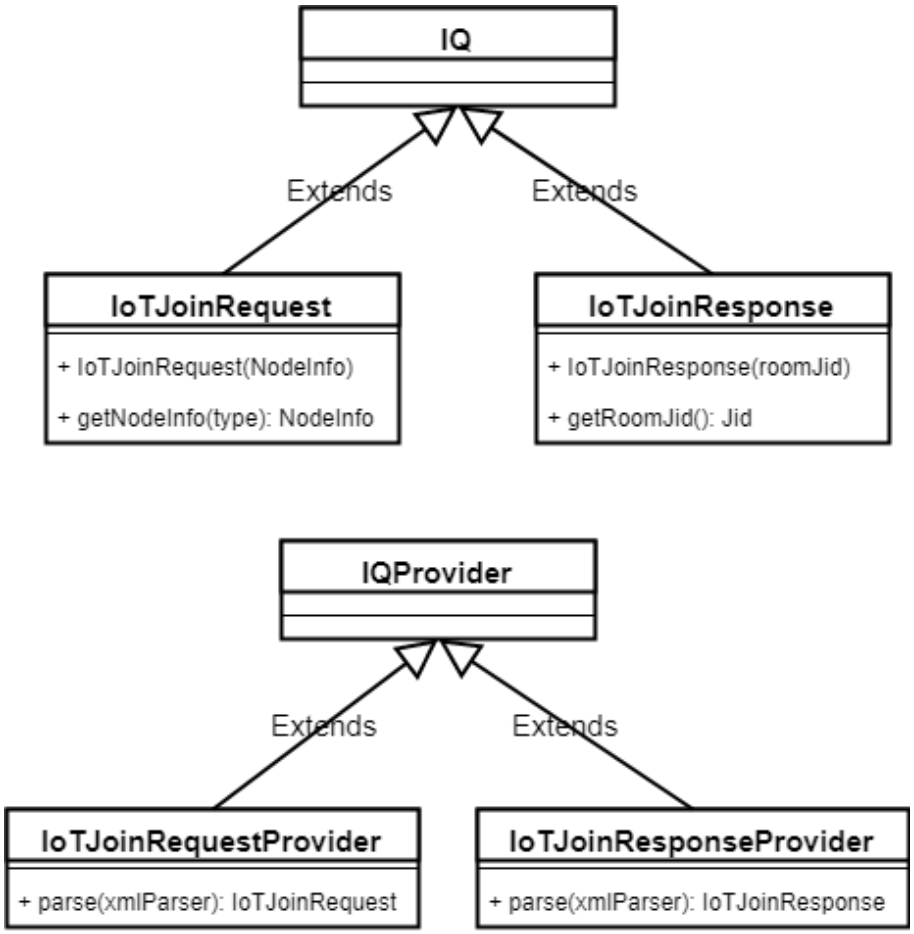
Use Case Scenario 1: Σύνδεση του Thing στο (μοναδικό του) room

1. Το Thing δημιουργεί ένα αίτημα τύπου JoinRequest με χρήση του στοιχείου NodeInfo που έχει
2. Το Thing στέλνει το αίτημα JoinRequest στον Provisioning Server και αναμένει για την απάντηση
3. Ο Provisioning Server δέχεται το αίτημα και ανταποκρίνεται με την απάντηση τύπου JoinResponse, η οποία περιέχει το Jid του room στο οποίο πρέπει να εισέλθει το Thing
4. Το Thing μόλις δεχθεί την απάντηση, δημιουργεί αντικείμενο τύπου MultiUserChat για το δωμάτιο με το Jid που έλαβε
5. Το Thing δηλώνει τον δικό του MessageListener στο αντικείμενο MultiUserChat
6. Το Thing στέλνει αίτημα εισόδου στο room
7. Το room αποδέχεται το αίτημα εισόδου και το Thing είναι πλέον συνδεδεμένο σε αυτό

Πριν γίνει περαιτέρω ανάπτυξη του σεναρίου, χρήσιμη είναι η παρουσίαση των μηνυμάτων JoinRequest και JoinResponse. Τα δύο αυτά μηνύματα αποτελούν IQ Stanzas, τύπου get και result αντίστοιχα. Αυτά αποτελούν τα μηνύματα που χρησιμοποιεί το Thing για να επικοινωνήσει με τον Provisioning Server, ώστε να ενημερωθεί για το room στο οποίο πρέπει να εισέλθει. Τα μηνύματα αυτά, καθώς δεν είναι μέρος κάποιου XEP και δεν περιέχονται στην βιβλιοθήκη Smack, αναπτύχθηκαν από την αρχή. Παρακάτω φαίνεται ένα παράδειγμα τέτοιων μηνυμάτων (joinRequest και joinResponse αντίστοιχα):

```
<iq to="provisioning@iot.ntua.gr/server" from="user1@iot.ntua.gr/myTh-
ing" id="xO3UZ-77" type="get">
  <joinRequest xmlns="urn:xmpp:iot:provisioning">
      <node nodeId="myThing" sourceId="" cacheType=""/>
  </joinRequest>
</iq>

<iq to="user1@iot.ntua.gr/myThing" from="provision-
ing@iot.ntua.gr/server" id="xO3UZ-77" type="result">
  <joinResponse xmlns="urn:xmpp:iot:provisioning" roomJid="mything@con-
ference.iot.ntua.gr"/>
</iq>
```

Παρατηρείται ότι τα μηνύματα αυτά είναι σχετικά απλά στη δομή τους οπότε και στην υλοποίηση. Η υλοποίηση των μηνυμάτων με χρήση της βιβλιοθήκης Smack θα γίνει με τον τρόπο που παρουσιάστηκε στην υποενότητα 4.3.1.2. Καθένα από τα δύο μηνύματα θα αποτελέσει μία κλάση τύπου Element που επεκτείνει την κλάση IQ. Ωστόσο, καθώς τα μηνύματα πρέπει να είναι αναγνώσιμα, τόσο από το Thing, όσο και από τον Provisioning Server, απαιτούνται και αντίστοιχες κλάσεις τύπου Provider. Αυτές οι κλάσεις θα επεκτείνουν την κλάση IQProvider. Παρακάτω φαίνεται το αντίστοιχο διάγραμμα κλάσεων.



Οι μέθοδοι των κλάσεων είναι οι βασικές. Συγκεκριμένα, η κλάση IoTJoinRequest παρέχει τον constructor που δέχεται το στοιχείο NodeInfo για την κατασκευή του Element και τη μέθοδο gedNodeInfo() για την ανάκτηση του NodeInfo κατά τη λήψη του. Παρομοίως η κλάση IoTJoinResponse παρέχει τον constructor ο οποίος δέχεται το Jid του room για τη δημιουργία του μηνύματος και τη μέθοδο getRoomJid() για την ανάκτηση αυτού του Jid κατά τη λήψη. Τέλος, οι Providers και για τα δύο μηνύματα παρέχουν την υποχρεωτική μέθοδο parse, για την ανάκτηση του στοιχείου NodeInfo και του attribute (ιδιότητας) roomJid αντίστοιχα, μέσα από το σώμα του XML μηνύματος. Παρακάτω δίνονται τα αντίστοιχα αποσπάσματα κώδικα των κλάσεων.

IoTJoinRequest

```
public class IoTJoinRequest extends IQ {

    public static final String ELEMENT = "joinRequest";
    public static final String NAMESPACE =
Constants.IOT_PROVISIONING_NAMESPACE;

    private final NodeInfo nodeInfo;

    public IoTJoinRequest(NodeInfo node) {
        super(ELEMENT, NAMESPACE);
        nodeInfo = node;
    }

    public NodeInfo getNodeInfo(){
       return nodeInfo;
    }

    @Override
    protected IQChildElementXmlStringBuilder
getIQChildElementBuilder(IQChildElementXmlStringBuilder xml) {
        NodeElement nodeElement = new NodeElement(nodeInfo);
        xml.rightAngleBracket();
        xml.append(nodeElement.toXML());
        return xml;
    }

}
```

IoTJoinResponse

```
public class IoTJoinResponse extends IQ {

    public static final String ELEMENT = "joinResponse";
    public static final String NAMESPACE = Constants.IOT_PROVISION-
ING_NAMESPACE;

    private final EntityBareJid roomJid;

    public IoTJoinResponse(EntityBareJid room) {
        super(ELEMENT, NAMESPACE);
        roomJid = room;
    }

    public EntityBareJid getRoomJid(){
       return roomJid;
    }

    @Override
    protected IQChildElementXmlStringBuilder getIQChildElement-
Builder(IQChildElementXmlStringBuilder xml) {
        xml.attribute("roomJid", roomJid.toString());
       xml.setEmptyElement();
        return xml;
    }

}
```

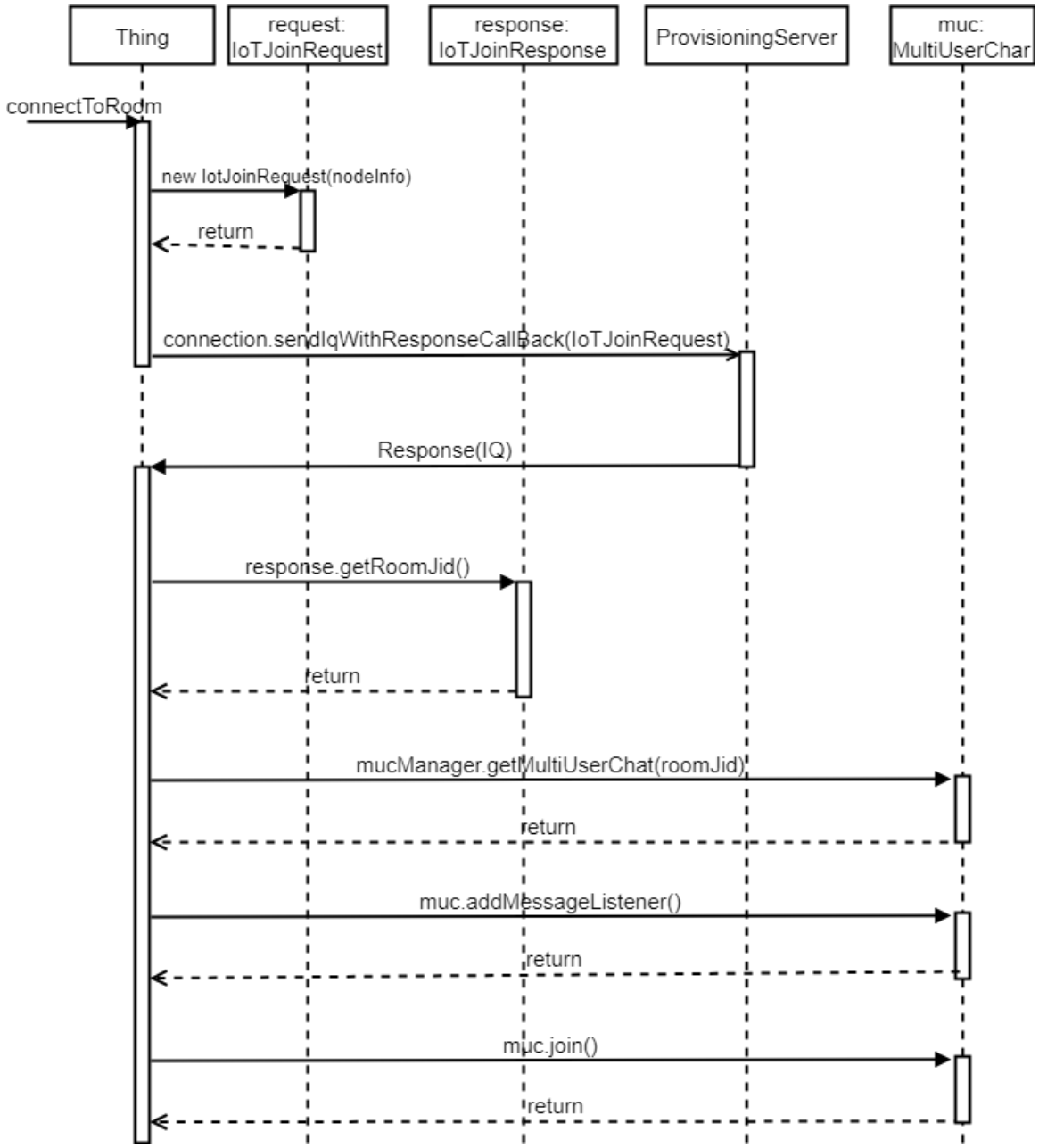
IoTJoinRequestProvider

```
public class IoTJoinRequestProvider extends IQProvider<IoTJoinRequest> {

	@Override
	public IoTJoinRequest parse(XmlPullParser parser, int initialDepth)
			throws Exception {
		NodeInfo nodeInfo = null;
		outerloop: while (true) {
			final int eventType = parser.next();
			final String name = parser.getName();
			switch (eventType) {
			case XmlPullParser.START_TAG:
				if (name.equals("node")){
					String nodeId = parser.getAttributeValue(null, "no-
deId");
					String sourceId = parser.getAttributeValue(null,
"sourceId");
					nodeInfo = new NodeInfo(nodeId,sourceId,"");
				}
				break;
			case XmlPullParser.END_TAG:
				if (parser.getDepth() == initialDepth) {
					break outerloop;
				}
				break;
			}
		}
			return new IoTJoinRequest(nodeInfo);
	}
}
```

IoTJoinResponseProvider

```
public class IoTJoinResponseProvider extends IQProvider<IoTJoinResponse> {

	@Override
	public IoTJoinResponse parse(XmlPullParser parser, int initialDepth)
			throws Exception {
		EntityBareJid roomJid = ParserUtils
					.getEntityJidAttribute(parser, "roomJid")
					.asEntityBareJid();
		return new IoTJoinResponse(roomJid);
	}

}
```

Τώρα που έγινε αναλυτική παρουσίαση της υλοποίησης των μηνυμάτων joinRequest και joinResponse, μπορεί να ακολουθήσει το sequence diagram (ακολουθιακό διάγραμμα) του σεναρίου.



Τα βήματα και οι ενέργειες που εκτελούνται στο σενάριο είναι πλέον αρκετά κατανοητά, ώστε να οδηγήσουν στο αντίστοιχο απόσπασμα κώδικα. Ο κώδικας για το σενάριο αυτό θα αποτελέσει τη μέθοδο connectToRoom() που θα παρέχει η κλάση Thing. Η μέθοδος αυτή δεν απαιτεί κανένα όρισμα καθώς το Jid του room δίνεται από τον ProvisioningServer.

```
public boolean connectToRoom(){
	IQ joinRequest = new IoTJoinRequest(nodeInfo);
	if (provisioningServer==null) return false;
	joinRequest.setTo(provisioningServer);
	try {
		connection.sendIqWithResponseCallback(joinRequest,
		new StanzaListener(){
			public void processStanza(Stanza packet)
			throws NotConnectedException, InterruptedException {
				IoTJoinResponse response = (IoTJoinResponse) packet;
				EntityBareJid roomJid = response.getRoomJid();
				muc = mucManager.getMultiUserChat(roomJid);
				muc.addMessageListener(roomMessageListener);
				try {

	muc.join(Resourcepart.from(nodeInfo.getNodeId()));
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		},
		new ExceptionCallback(){
			public void processException(Exception exception) {}
		});

		return true;
	} catch (Exception e) {
		return false;
	}
}
```

Σημειώνεται πως διαχειρίζονται όλες οι περιπτώσεις σφάλματος exception, έτσι ώστε η συνάρτηση να επιστρέφει true σε περίπτωση που ολοκληρώθηκε η σύνδεση και false σε περίπτωση που υπήρξε σφάλμα.

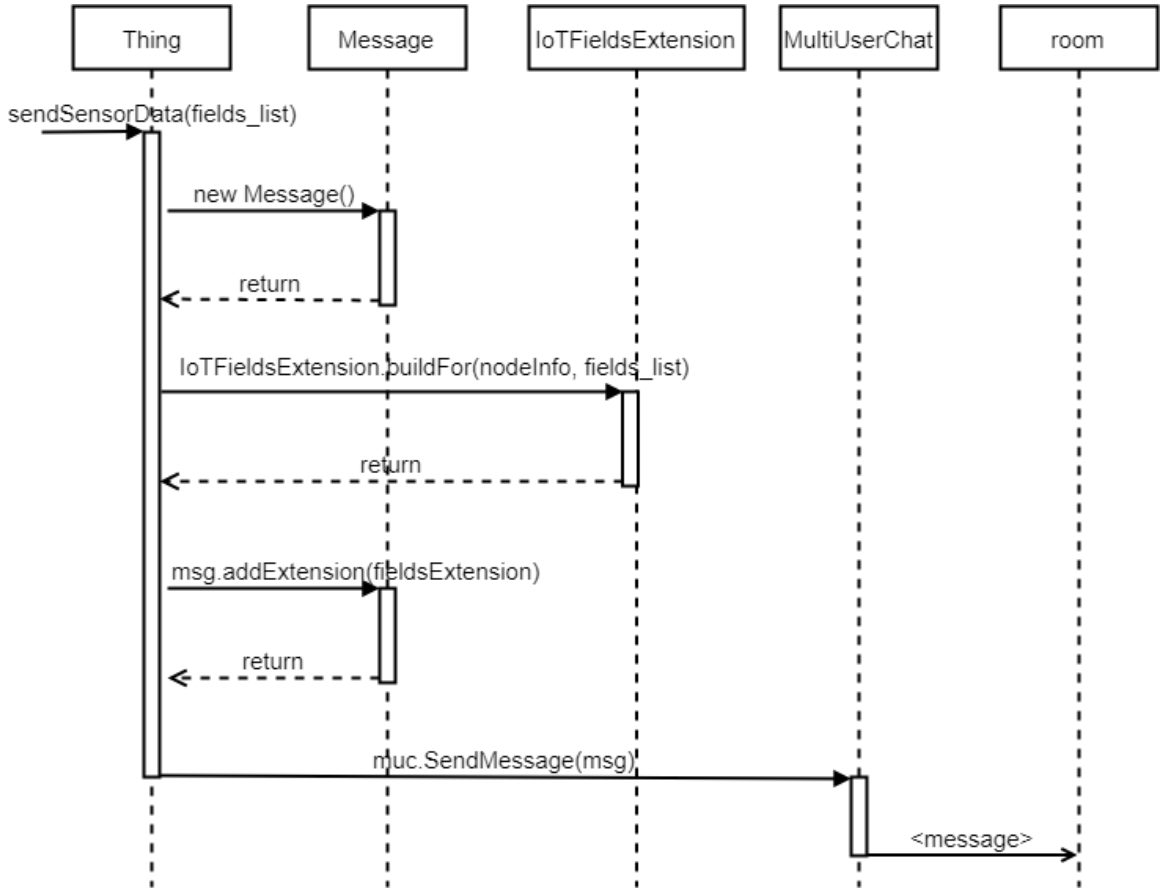
Use Case Scenario 2: Αποστολή δεδομένων Sensor Data στο room

1. Το Thing δέχεται μία λίστα από πεδία δεδομένων (IoTDataFields) προς αποστολή
2. Το Thing δημιουργεί ένα κενό σώμα μηνύματος Message Stanza
3. Το Thing δημιουργεί ένα στοιχείο FieldsExtension χρησιμοποιώντας το στοιχείο nodeInfo και την λίστα με τα IoTDataFields
4. Το Thing ενσωματώνει το στοιχείο FieldsExtension στο σώμα του Message Stanza
5. Με τη χρήση του αντικειμένου MultiUserChat, το Thing αποστέλλει το μήνυμα στο room

Σημειώνεται πως και σε αυτή την περίπτωση έγινε ανάπτυξη νέων ή override (επικάλυψη) υπάρχοντων XMPP Elements και Providers για την κάλυψη των όποιων ελλείψεων παρατηρήθηκαν από τη βιβλιοθήκη Smack. Ωστόσο εδώ, όπως και στη συνέχεια, χάριν απλότητας, παραλείπεται η ανάλυσή τους. Ένα παράδειγμα μηνύματος φαίνεται παρακάτω.

```
<message from ="user1@iot.ntua.gr/myThing"
to="test_room@conference.iot.ntua.gr" id="u0cht-137" type="groupchat">
  <fields xmlns="urn:xmpp:iot:sensordata" seqnr="0" done="true">
    <node nodeId="myThing" sourceId="" cacheType="">
      <timestamp value="2018-04-01T13:47:51.506+00:00">
        <numeric name="temprature" value="22.689414704011497"/>
        <boolean name="rains" value="true"/>
        <string name="msg" value="hello from thing"/>
      </timestamp>
    </node>
  </fields>
</message>
```

Διακρίνεται τόσο η επικεφαλίδα του μηνύματος Message που στέλνεται προς το room, όσο και το στοιχείο FieldsExtension (fields) με τα DataFields. Παρακάτω διακρίνεται το sequence diagram του σεναρίου.



Οι διαδικασίες και εδώ είναι αρκετά απλές και κατανοητές πλέον. Το σενάριο θα υλοποιηθεί ως μέθοδος που παρέχει η κλάση Thing με όνομα sendSensorData(). Αυτή η μέθοδος δέχεται ως παράμετρο μία λίστα από IoTDataFields, η οποία έχει πεδία δεδομένων με όνομα, περιγραφή και τιμή δεδομένων και στέλνει τα δεδομένα αυτά στο room. Σημειώνεται πως ως πάγια τεχνική και σε αυτή την περίπτωση τα exceptions χειρίζονται έτσι ώστε η μέθοδος να

επιστρέφει true αν η διαδικασία πέτυχε και false εάν απέτυχε. Παρακάτω διακρίνεται το αντίστοιχο απόσπασμα κώδικα της μεθόδου.

```
public boolean sendSensorData(List<IoTDataField> data){
	Message msg = new Message();
	IoTFieldsExtension fieldsExtension = IoTFieldsExtension
			.buildFor(0, true, nodeInfo, data);
	msg.addExtension(fieldsExtension);
	try {
		muc.sendMessage(msg);
		return true;
	} catch (Exception e) {
		return false;
	}
}
```

Use Case Scenario 3: Λήψη δεδομένων control από το room

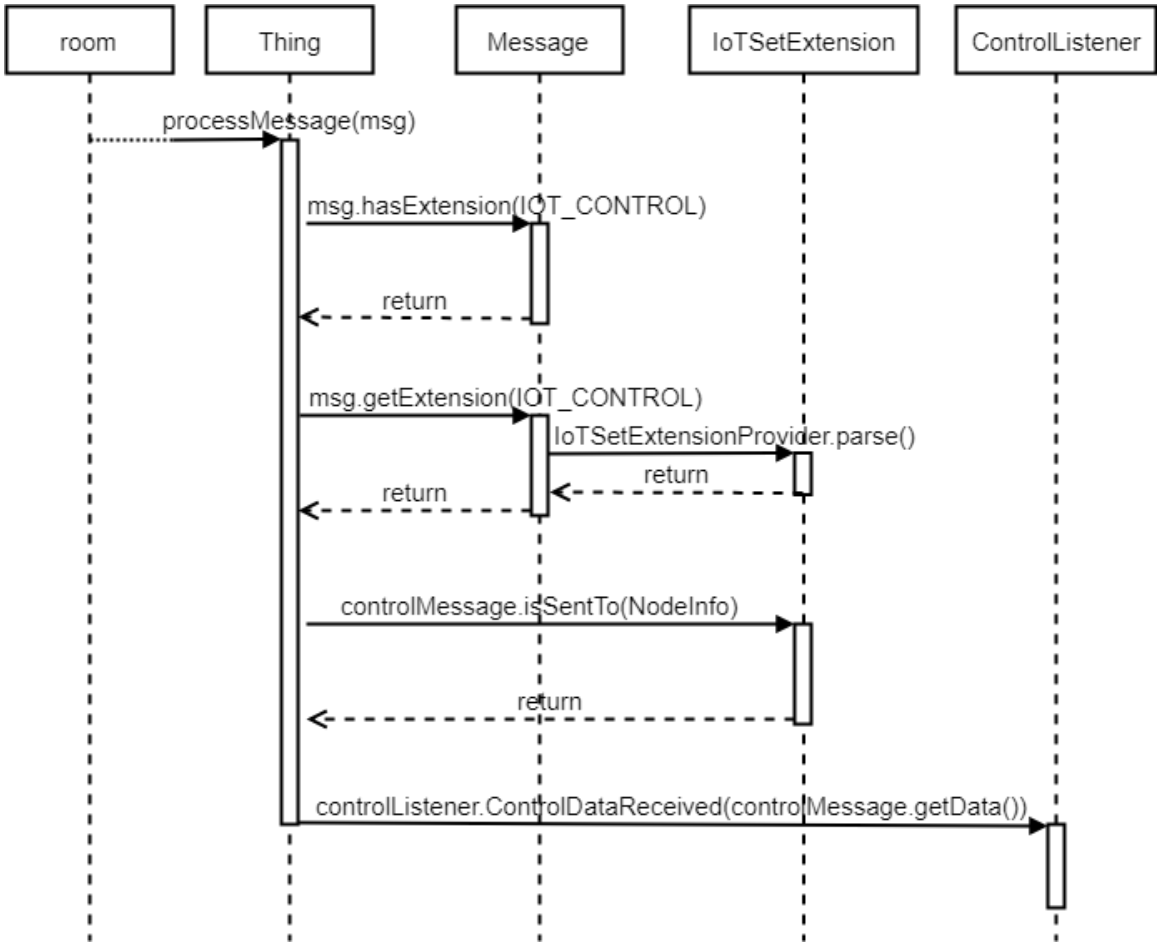
1. Ο MessageListener που έχει δηλώσει το Thing στο MultiUserChat ειδοποιείται από το room για νέο μήνυμα
2. Γίνεται έλεγχος ότι το μήνυμα περιέχει το στοιχείο SetExtension
3. Γίνεται parse του στοιχείου σε αντικείμενο της κλάσης IoTSetExtension
4. Γίνεται έλεγχος ότι το στοιχείο SetExtension αναφέρεται στο NodeInfo του Thing
5. Ανακτάται η λίστα με SetData, που αποτελούν πεδία εντολών ελέγχου με όνομα και τιμή, από το στοιχείο SetExtension
6. Ειδοποιείται ο controlListener που έχει δηλωθεί στο Thing για τη νέα εντολή ελέγχου, με τη λίστα των SetData

Ένα ενδεικτικό μήνυμα Control που μπορεί να λάβει το Thing από το room φαίνεται παρακάτω.

```
<message to="user1@iot.ntua.gr/myThing"
from="test_room@conference.iot.ntua.gr/myController" id="1r7cZ-379"
type="groupchat">
  <set xmlns="urn:xmpp:iot:control">
    <node nodeId="myThing" sourceId="" cacheType=""/>
    <string name="Command1" value="turn on"/>
    <string name="Command2" value="lights"/>
    <long name="Number1" value="5"/>
  </set>
</message>
```

Διακρίνεται τόσο η επικεφαλίδα του Message Stanza, όσο και το στοιχείο SetExtension (set) με τη λίστα από SetData. SetData μπορούν να αποτελούν στοιχεία τύπου bool, int, long, double και string. Ωστόσο με τη χρήση των διαδικασιών που έχουν αναφερθεί είναι δυνατή η ανάπτυξη νέων τύπων SetData. Η διαδικασία αυτή χρησιμοποιήθηκε για την ανάπτυξη του τύπου string που δεν παρείχε η βιβλιοθήκη Smack.

Στη συνέχεια διακρίνεται το sequence diagram του παραπάνω σεναρίου.



Σε αυτή την περίπτωση το σενάριο οδηγεί στην υλοποίηση του αντικειμένου MessageListener που έχει το Thing. Συγκεκριμένα ένα αντικείμενο MessageListener απαιτεί την υλοποίηση της μεθόδου processMessage(), η οποία δέχεται ως παράμετρο το αντικείμενο Message που αντιπροσωπεύει το Message Stanza που λαμβάνεται.
Σημειώνεται πως ο MessageListener τελικά ειδοποιεί τον ControlListener. Ο ControlListener αποτελεί Listener που δηλώνεται στο Thing και ειδοποιείται κατευθείαν για νέες εντολές ελέγχου, όποτε αυτές ληφθούν, κατευθείαν με τη λίστα των SetData. Συνεπώς το Thing πρέπει να παρέχει την αντίστοιχη μέθοδο δήλωσης του ControlListener. Παρακάτω φαίνεται το αντίστοιχο απόσπασμα κώδικα του παραπάνω σεναρίου, μαζί με τη μέθοδο αυτή.

```
private MessageListener roomMessageListener = new MessageListener(){
      public void processMessage(Message msg) {
            if (!msg
            .hasExtension(Constants.IOT_CONTROL_NAMESPACE)) return;
            if (controlListener==null) return;
            IoTSetExtension setExtension = (IoTSetExtension)
            msg.getExtension(Constants.IOT_CONTROL_NAMESPACE);
            ControlMessage controlMessage = ControlMessage
                        .parseSetExtension(setExtension);
            if (!controlMessage.isSentTo(nodeInfo.getNodeId())) return;
            controlListener
            .ControlDataReceived(controlMessage.getData());
      }
};

                                    .
                                    .
                                    .

public void setControlListener(ControlListener listener){
      controlListener = listener;
}
```

4.3.4 Η ανάπτυξη του Client Component - Κλάση Controller

Επόμενη οντότητα προς υλοποίηση είναι αυτή του Client. Ο Client αποτελεί την οντότητα που επιθυμεί να αλληλοεπιδράσει με τα Things, λαμβάνοντας τα δεδομένα τους και αποστέλλοντάς τους εντολές ελέγχου. Όπως είναι ήδη γνωστό η οντότητα αυτή θα αποτελέσει κλάση, η οποία επεκτείνει την AbstractEntity και παρέχει απλές μεθόδους για τη δημιουργία και χρήση της. Για αποφυγή συγχύσεων μέσα στον κώδικα, το όνομα της κλάσης είναι Controller αντί για Client, όπως φαίνεται και στο γενικό διάγραμμα κλάσεων παραπάνω. Ακολουθώντας τη λογική με την οποία δημιουργείται και το Thing, έτσι και η κλάση Controller έχει ιδιωτικό constructor και η δημιουργία αντικειμένου της γίνεται μέσω static μεθόδου. Παρακάτω διακρίνεται ο σχετικός κώδικας. Σε αυτόν επίσης διακρίνονται οι private μεταβλητές της κλάσης:

- rooms: αποτελεί τη λίστα με τα αντικείμενα MultiUserChat των rooms στα οποία έχει εισέλθει ο Client
- jidToRoom: αποτελεί τη λίστα που συνδέει κάθε Jid με το αντίστοιχο αντικείμενο MultiUserChat (από αυτά που υπάρχουν και στην λίστα rooms)
- onlineThings: αποτελεί τη λίστα που συνδέει κάθε MultiUserChat με μία λίστα από Jids που αποτελούν τα Things που είναι online στο αντίστοιχο room
- thingPresenceListener: αποτελεί τον προσαρμοσμένο listener που υλοποιήθηκε ώστε να ειδοποιείται για κάθε αλλαγή της λίστας των Things που είναι online σε κάποιο room
- sensorDataListener: αποτελεί τον προσαρμοσμένο listener που υλοποιήθηκε ώστε να ειδοποιείται για κάθε μήνυμα Sensor Data που λαμβάνεται από κάποιο Thing

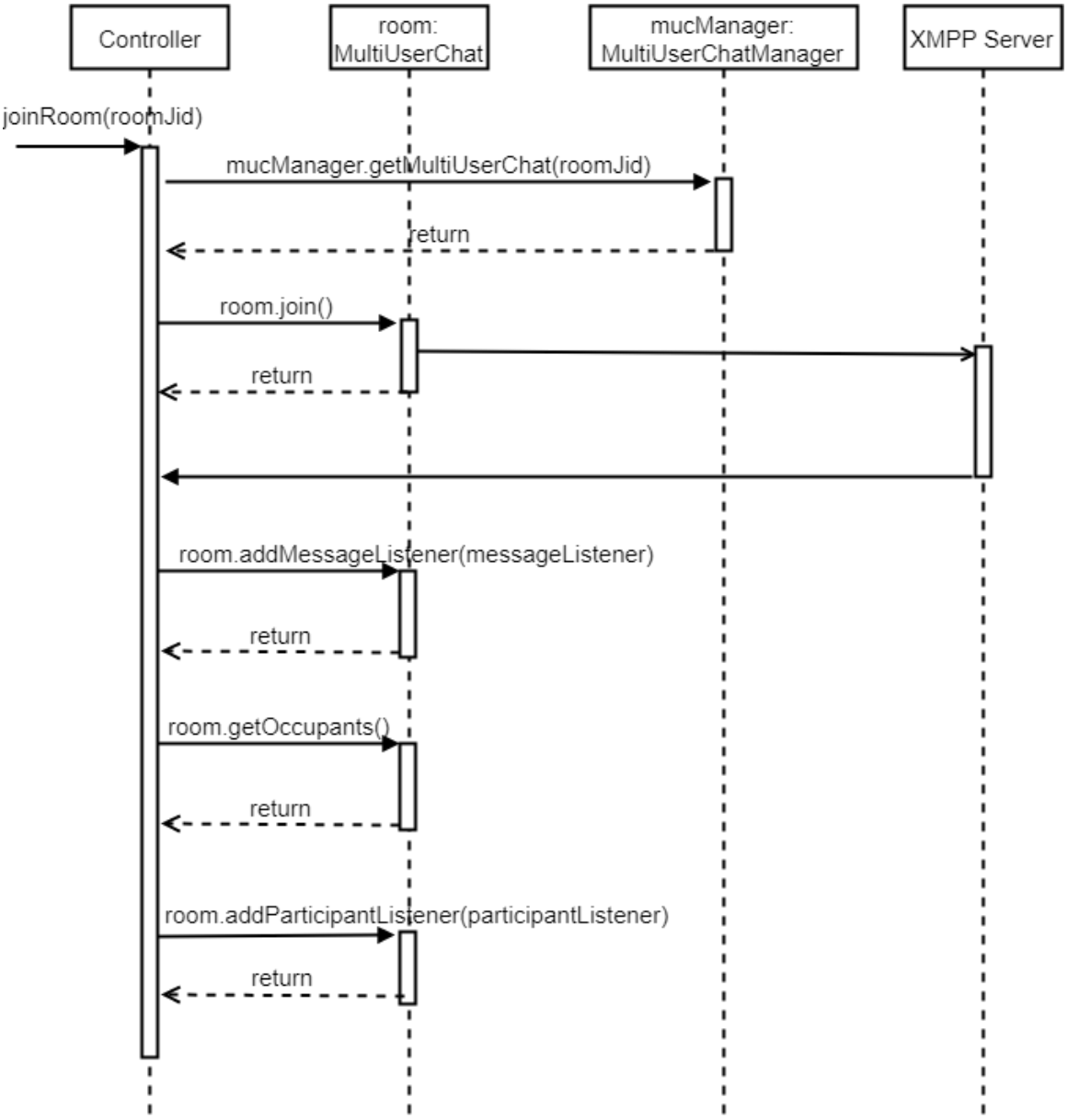
```
private List<MultiUserChat> rooms;
private Map<BareJid,MultiUserChat> jidToRoom;
private Map<MultiUserChat,List<EntityFullJid>> onlineThings;
private ThingPresenceListener thingPresenceListener = null;
private SensorDataListener sensorDataListener = null;

private Controller(String user, String pass, String domain, String
resource) throws Exception {
	super(user, pass, domain, resource);
	login();
	rooms = new ArrayList<MultiUserChat>();
	onlineThings = new HashMap<MultiUserChat,List<EntityFullJid>>();
	jidToRoom = new HashMap<BareJid,MultiUserChat>();
}

public static Controller createController(String user, String pass,
String domain, String resource){
	try{
		return new Controller(user, pass, domain, resource);
	}
	catch (Exception e){
		return null;
	}
}
```

Παρακάτω παρουσιάζονται τα βασικά σενάρια χρήσης με τα αντίστοιχα ακολουθιακά διαγράμματα και αποσπάσματα κώδικα για την κλάση.

Use Case Scenario 1: Σύνδεση σε δωμάτιο

1. Ο Controller λαμβάνει το Jid του room με το οποίο επιθυμείται να γίνει σύνδεση
2. Μέσω του αντικειμένου MultiUserChatManager ο Controller ανακτά το αντικείμενο MultiUserChat του room
3. Μέσω του αντικειμένου MultiUserChat γίνεται αποστολή μηνύματος εισόδου / join στο room
4. Η MUC υπηρεσία του server δέχεται το αίτημα και σε περίπτωση που ο Client έχει δικαίωμα πρόσβασης στο room αποκρίνεται θετικά
5. Ο Controller λαμβάνει τη θετική απάντηση και θέτει στο αντικείμενο MultiUserChat τον δικό του MessageListener για να ενημερώνεται για τα μηνύματα που στέλνονται στο room
6. Ο Controller τοποθετεί το αντικείμενο MultiUserChat στις λίστες με τα διαθέσιμα δωμάτιά του
7. Ο Controller λαμβάνει την αρχική λίστα των Things που είναι Online στο room και τη συνδέει με το αντικείμενο MultiUserChat
8. Δηλώνεται ο ParticipantListener του Controller στο αντικείμενο MultiUserChat για την ενημέρωση της λίστα των online Things του δωματίου

Το αντίστοιχο ακολουθιακό διάγραμμα του σεναρίου διακρίνεται παρακάτω.



Όπως φαίνεται η παραπάνω διαδικασία θα υλοποιηθεί ως μέθοδος που παρέχει η κλάση και δέχεται το Jid του room προς σύνδεση. Στη συνέχεια δίνεται το αντίστοιχο απόσπασμα κώδικα για τη μέθοδο αυτή.

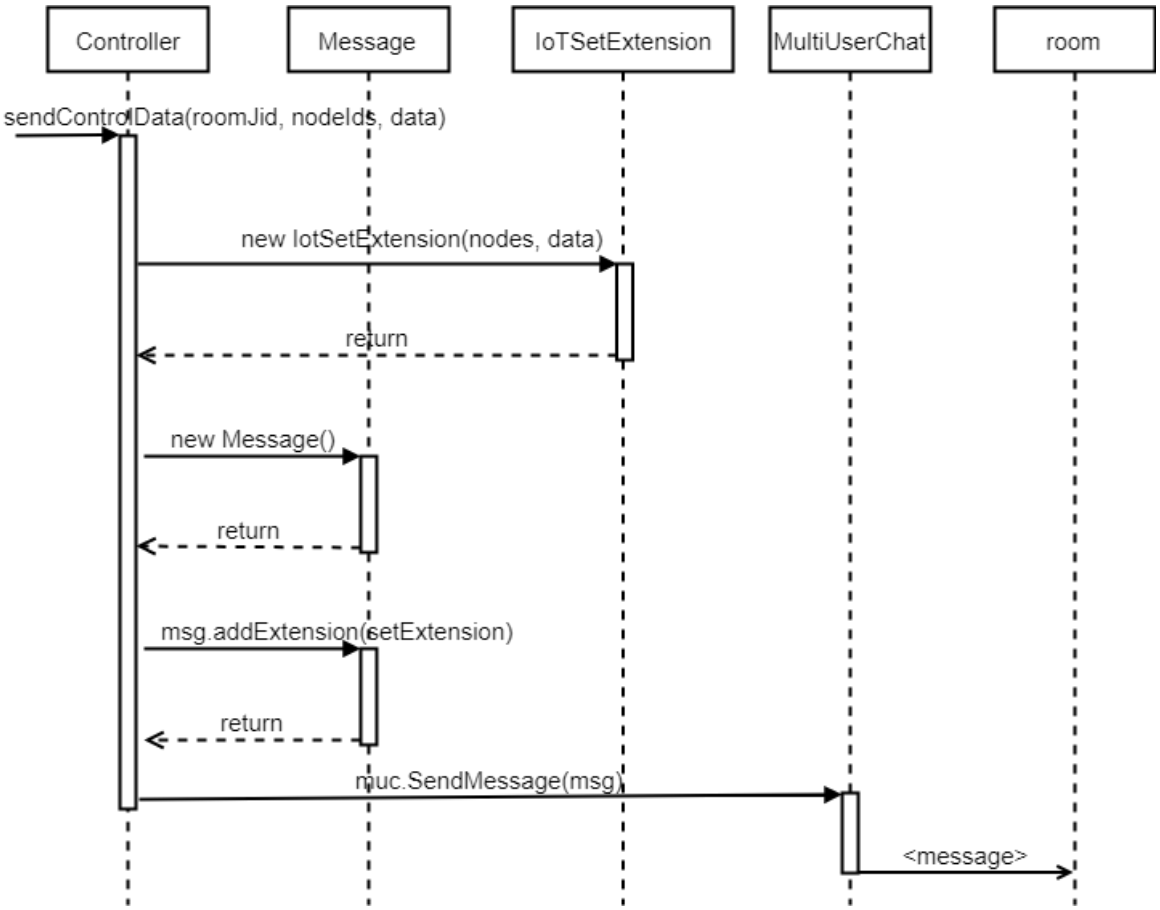
```
public boolean joinRoom(EntityBareJid roomJid){
	if (mucManager.getJoinedRooms().contains(roomJid)) return true;
	MultiUserChat room;
	try{
		room = mucManager.getMultiUserChat(roomJid);
		room.join(Resourcepart.from(this.resourcePart));
		room.addMessageListener(roomMessageListener);
		rooms.add(room);
		jidToRoom.put(room.getRoom(), room);
		List<EntityFullJid> online = room.getOccupants();
		for (EntityFullJid occupant : online)
			if (room.getOccupantPresence(occupant)==null)
				online.remove(occupant);
		onlineThings.put(room, online);
		if (thingPresenceListener!=null)
thingPresenceListener.thingPresenceChanged(roomJid,
onlineThings.get(room));
		room.addParticipantListener(participantListener);
		return true;
	}
	catch (Exception e){
		return false;
	}
}
```

Σημειώνεται ότι η παραπάνω μέθοδος σε περίπτωση που η είσοδος απορριφθεί επιστρέφει FALSE. Αυτό γίνεται μέσω του χειρισμού του Exception που επιστρέφεται από τη μέθοδο join της κλάσης MultiUserChat σε αυτή την περίπτωση.

Use Case Scenario 2: Αποστολή εντολών ελέγχων σε Thing(s)

1. Δίνεται στον Controller το Jid του room, μία λίστα από ονόματα nodeId των Things και μία λίστα από στοιχεία IotSetData με τις εντολές ελέγχου
2. Γίνεται ανάκτηση του αντικειμένου MultiUserChat μέσω του Jid του room από την αντίστοιχη λίστα (jidToRoom)
3. Δημιουργείται μία λίστα από στοιχεία NodeElements με χρήση της λίστας των nodeId
4. Δημιουργείται ένα στοιχείο IoTSetExtension μέσω της λίστας των NodeElements και της λίστας των IotSetData
5. Δημιουργείται ένα νέο Message Stanza
6. Στο Message Stanza προστίθεται το στοιχείο IoTSetExtension
7. Μέσω του αντικειμένου MultiUserChat του room γίνεται αποστολή του μηνύματος που δημιουργήθηκε

Και σε αυτή την περίπτωση το παραπάνω σενάριο θα υλοποιηθεί ως μέθοδος που παρέχει η κλάση Controller. Η μέθοδος αυτή δέχεται το Jid του room, τη λίστα με τα nodeId των Things και τη λίστα από τα στοιχεία IoTSetData. Το ακολουθιακό διάγραμμα του σεναρίου διακρίνεται παρακάτω.



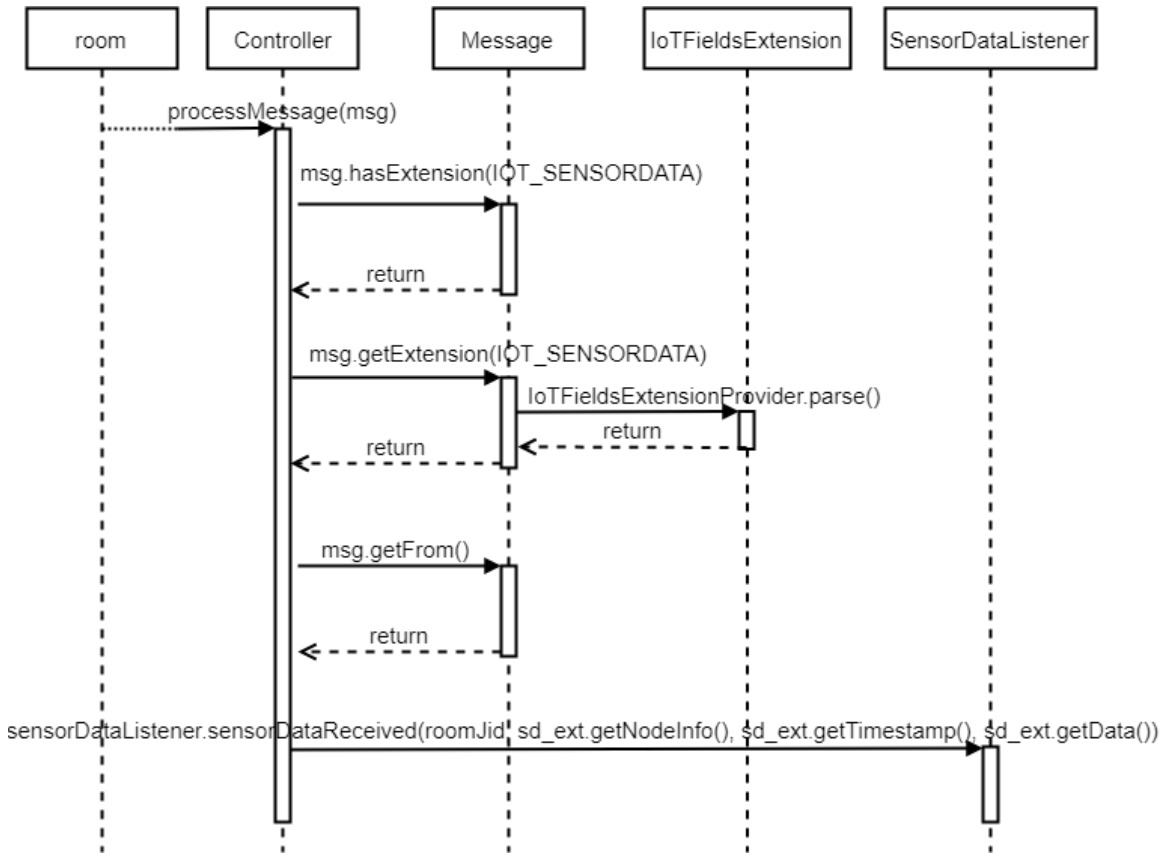
Τέλος, δίνεται το αντίστοιχο απόσπασμα κώδικα της μεθόδου. Παρατηρείται ότι και εδώ γίνεται ίδιος χειρισμός του Exception που μπορεί να εγείρει η αποστολή του μηνύματος και σε αυτή την περίπτωση η μέθοδος επιστρέφει FALSE.

```
public boolean sendControlData(BareJid roomJid, List<String> nodeIds,
List<SetData> data){
	MultiUserChat muc = jidToRoom.get(roomJid);
	List<NodeElement> nodes = new ArrayList<NodeElement>();
	for (String nodeId : nodeIds)
		nodes.add(new NodeElement(new NodeInfo(nodeId,"","")));
	IoTSetExtension setExtension = new IoTSetExtension(nodes, data);
	Message msg = new Message();
	msg.addExtension(setExtension);
	try{
		muc.sendMessage(msg);
		return true;
	}
	catch (Exception e){
		return false;
	}
}
```

Use Case Scenario 3: Λήψη δεδομένων Sensor Data από Thing

1. Ειδοποιείται ο MessageListener του Controller για νέο μήνημα από κάποιο room με νέο Message Stanza
2. Γίνεται έλεγχος ότι το μήνυμα περιέχει το στοιχείο επέκτασης IoTSensorData
3. Γίνεται ανάκτηση του στοιχείου IoTFieldsExtension που περιέχει τα Sensor Data από το μήνυμα
4. Γίνεται ανάκτηση του Jid του room από το οποίο στάλθηκε το μήνυμα
5. Γίνεται ανάκτηση του nodeId που αντιστοιχεί στο Thing, του timestamp και της λίστας με τα IotDataField από το στοιχείο IoTFieldsExtension
6. Ειδοποιείται ο SensorDataListener που έχει δηλωθεί για νέο Sensor Data από το room με το Jid του room, το nodeId του Thing, το timestamp και τη λίστα με τα IotDataFields

Σε αυτή την περίπτωση η υλοποίηση αφορά τον MessageListener που χρησιμοποιεί η κλάση Controller και ειδοποιεί την αντίστοιχο Listener (SensorDataListener) που δηλώνεται σε αυτόν. Όπως έχει ήδη αναφερθεί η δημιουργία ενός MessageListener απαιτεί την υλοποίηση της συνάρτησης processMessage η οποία καλείται κάθε φορά που λαμβάνεται ένα Message Stanza, με όρισμα το ίδιο το στοιχείο Message. Στη συνέχεια δίνεται το ακολουθιακό διάγραμμα του σεναρίου.



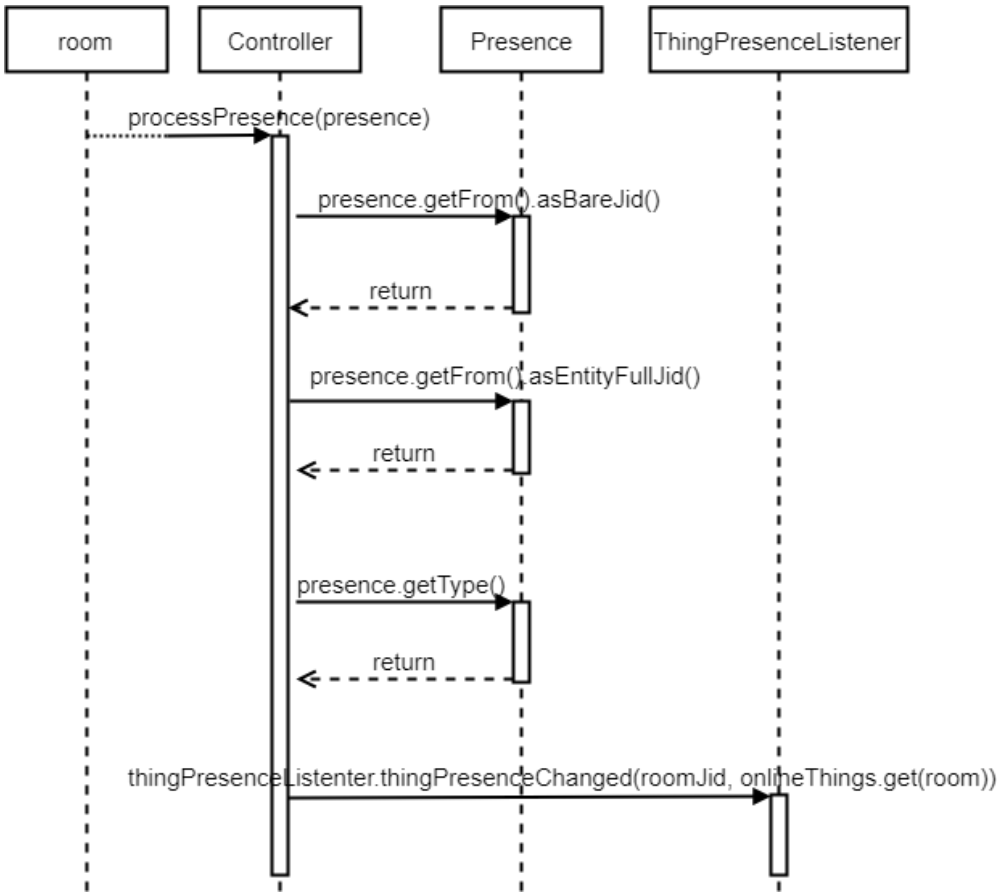
Τέλος δίνεται το αντίστοιχο απόσπασμα κώδικα του MessageListener που χρησιμοποιεί η κλάση Controller.

```
private MessageListener roomMessageListener = new MessageListener(){

	public void processMessage(Message msg) {
		if (!msg.hasExtension(Constants.IOT_SENSORDATA_NAMESPACE))
		return;
		if (sensorDataListener==null) return;

		IoTFieldsExtension fieldsExtension = (IoTFieldsExtension)
			msg
			.getExtension(Constants.IOT_SENSORDATA_NAMESPACE);

		BareJid roomJid = msg.getFrom().asBareJid();
		SensorDataMessage parsed = SensorDataMessage
				.parseFieldsExtension(fieldsExtension);

		sensorDataListener.sensorDataReceived(roomJid,
					parsed.getNodeInfo(),
					parsed.getTimestamp(),
					parsed.getData());
	}

};
```

Use Case Scenario 4: Ενημέρωση για αλλαγές των online Things του δωματίου

1. Ειδοποιείται ο PresenceListener (participantListener) του Controller για αλλαγή της κατάστασης συμμετεχόντων σε κάποιο room με νέο Presence Stanza
2. Γίνεται ανάκτηση του Jid του room από το οποίο στάλθηκε το Stanza
3. Μέσω της λίστας jidToRoom γίνεται ανάκτηση του αντικειμένου MultiUserChat που αντιστοιχεί στο δωμάτιο
4. Γίνεται ανάκτηση του Jid του Thing, στο οποίο αντιστοιχεί το Presence, από το Stanza
5. Γίνεται ανάκτηση του είδους του Presence Stanza και συγκεκριμένα το αν αυτό αναφέρεται σε κατάσταση unavailable ή όχι
6. Στην περίπτωση που το Presence αντιστοιχεί σε κατάσταση unavailable, αφαιρείται το Jid του Thing από την αντίστοιχη λίστα των διαθέσιμων Things του room, ενώ σε αντίθετη περίπτωση προστίθεται
7. Ειδοποιείται ο thingPresenceListener που έχει δηλωθεί, με το Jid του room και τη νέα λίστα των online Things που υπάρχουν σε αυτό

Όπως και το παραπάνω σενάριο, έτσι και αυτό αντιστοιχεί στην υλοποίηση ενός Stanza Listener. Ωστόσο, σε αυτή την περίπτωση ο Listener δέχεται Presence Stanza. Η αντίστοιχη μέθοδος προς υλοποίηση είναι η processPresence η οποία καλείται με όρισμα το ίδιο το στοιχείο Presence, κάθε φορά που λαμβάνεται τέτοιο μήνυμα. Στη συνέχεια δίνεται το αντίστοιχο ακολουθιακό διάγραμμα του σεναρίου.



Τέλος, δίνεται το αντίστοιχο απόσπασμα κώδικα για την υλοποίηση του PresenceListener της κλάσης Controller.

```
private PresenceListener participantListener = new PresenceListener(){

	public void processPresence(Presence presence) {
		BareJid roomJid = presence.getFrom().asBareJid();
		MultiUserChat room = jidToRoom.get(roomJid);

		EntityFullJid thingJid = presence
					.getFrom()
					.asEntityFullJidIfPossible();
		Boolean unavailable = presence
					.getType()
					.equals(Presence.Type.unavailable);

		if (unavailable) onlineThings.get(room).remove(thingJid);
		else onlineThings.get(room).add(thingJid);

		if (thingPresenceListener!=null) thingPresenceListener
					.thingPresenceChanged(roomJid,
						onlineThings.get(room));
	}
};
```

Παραπάνω έγινε παρουσίαση των βασικών σεναρίων χρήσης, για την κατανόηση των σημαντικότερων μηχανισμών της κλάσης. Ωστόσο, υπάρχουν μερικές επιπλέον μέθοδοι που υλοποιήθηκαν, ώστε να παρέχεται πλήρης διευκόλυνση στον χειρισμό της για τη δημιουργία ενός Client. Αυτές οι μέθοδοι παρουσιάζονται συνοπτικά στη συνέχεια.

- getJoinedRooms(): επιστρέφει τη λίστα με τα Jid των δωματίων στα οποία έχει γίνει είσοδος
- getOnlineThingsJids(Jid room): δέχεται το Jid ενός δωματίου που έχει γίνει είσοδος και επιστρέφει τη λίστα με τα Jid των Thing που είναι διαθέσιμα σε αυτό
- getOnlineThingsNodeIds(Jid room): ίδια με την παραπάνω αλλά σε αυτή την περίπτωση επιστρέφει τη λίστα με τα nodeIds των Thing
- setThingPresenceListener(ThingPreseceListener l): δέχεται και θέτει τον ThingPresenceListener, ο οποίος ενημερώνεται για την αλλαγή των διαθέσιμων Thing που υπάρχουν στα δωμάτια
- setSensorDataListener(SensorDataListener l): δέχεται και θέτει τον SensorDataListener, ο οποίος ενημερώνεται κάθε φορά που λαμβάνεται κάποιο Sensor Data από κάποιο Thing

Χάριν πληρότητας δίνονται και τα αντίστοιχα αποσπάσματα κώδικα των παραπάνω μεθόδων.

```
public Set<BareJid> getJoinedRooms(){
	return jidToRoom.keySet();
}

public List<EntityFullJid> getOnlineThingsJids(EntityBareJid room){
	return onlineThings.get(room);
}

public List<String> getOnlineThingsNodeIds(EntityBareJid room){
	List<EntityFullJid> jids = onlineThings.get(room);
	List<String> nodeIds = new ArrayList<String>();
	for (EntityFullJid jid : jids){
		nodeIds.add(jid.getResourceOrEmpty().toString());
	}
	return nodeIds;
}

public void setThingPresenceListener(ThingPresenceListener listener){
	thingPresenceListener = listener;
}

public void setSensorDataListener(SensorDataListener listener){
	sensorDataListener = listener;
}
```

4.3.5 Η ανάπτυξη του Provisioning Server/Component - Κλάση ProvisioningServer

Χαρακτηριστικό κομμάτι της υλοποίησης αποτελεί ο Provisioning Server. Βασική του λειτουργία είναι η διαχείριση του access control στην επικοινωνία των Clients με τα Things, με χρήση των MUC rooms. Ο Provisioning Server δημιουργεί rooms στα οποία θα εισέλθουν Things και χρησιμοποιεί το δικαίωμα διαχείρισής τους, ως ιδιοκτήτης, περιορίζοντας την πρόσβαση στους επιθυμητούς Clients. Οι βασικότερες λειτουργίες του είναι οι εξής:

- Δημιουργία και διαχείριση των δικαιωμάτων πρόσβασης των MUC rooms
- Απόκριση σε αιτήματα (joinRequest) των Things σχετικά με το room που ανήκουν

Οι Clients όπως είναι ήδη γνωστό, δεν επικοινωνούν άμεσα με τον Provisioning Server, αλλά μπορούν να κάνουν κατευθείαν αιτήσεις εισόδου στα rooms. Η αποδοχή ή μη των αιτημάτων αυτών εξαρτάται από τη λίστα συμμετεχόντων που έχει θέσει ο Provisioning Server.

Όσον αφορά τη διαχείριση δικαιωμάτων πρόσβασης των rooms, υπάρχουν αρκετές διαφορετικές επιλογές χειρισμού. Οι κυριότερες επιλογές είναι δύο:

- Blacklist: στο room μπορεί να εισέλθει οποιοσδήποτε XMPP Client, εκτός από αυτούς που ανήκουν στη λίστα αποκλεισμού (banned)
- Whitelist: στο room μπορεί να εισέλθει κάποιος XMPP Client μόνο αν ανήκει στη λίστα συμμετεχόντων (member)

Η πρώτη επιλογή είναι η προεπιλεγμένη κατά τη δημιουργία ενός δωματίου. Η δεύτερη απαιτεί τη δημιουργία ενός δωματίου με την επιλογή membersonly. Στην υλοποίηση αυτή προτιμήθηκε η χρήση του whitelist καθώς επιτρέπει τον πληρέστερο έλεγχο. Ο Provisioning Server καθορίζει αυστηρά τους συμμετέχοντες κάθε δωματίου (Things και Clients). Συνεπώς, απαραίτητη είναι η ανάγκη δημιουργίας και διαχείρισης λιστών με:

- τα rooms
- τα Things του κάθε room (ένα room ανά Thing)
- τους Clients του κάθε room (πολλά room ανά Client)

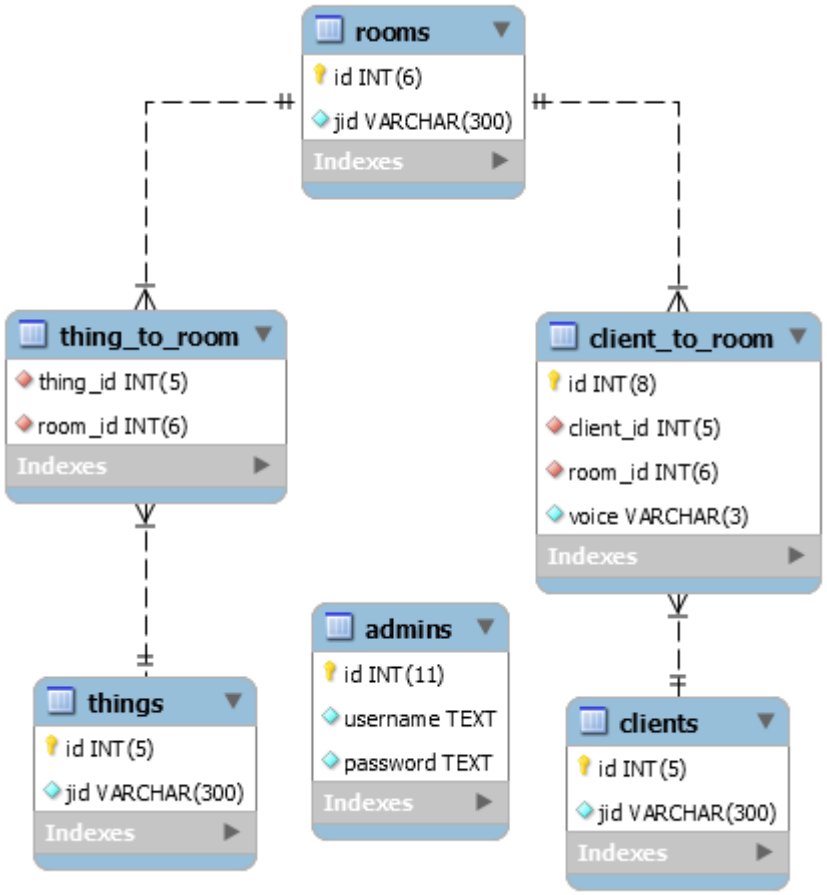
Επιπρόσθετα στη λίστα με τους Clients του κάθε room, πρέπει να είναι σαφές για το αν ο Client έχει και δικαίωμα αποστολής εντολών ελέγχου ή μόνο λήψης sensor data (voice).

Η λογική με την οποία γίνεται η διαχείριση των δεδομένων αυτών αφέθηκε ελεύθερη καθώς δεν αποτελεί αντικείμενο μελέτης αυτής της εργασίας. Ο Provisioning Server αντιπροσωπεύει αρχή διασφάλισης του access control στο IoT δίκτυο, εμπνέοντας εμπιστοσύνη στους κατόχους των Things. Συνεπώς, βασικός τρόπος διαχείρισης μπορεί να αποτελέσει κάποιο γραφικό Administration Panel (πίνακας διαχείρισης), το οποίο χειρίζονται άνθρωποι, αποφασίζοντας χειροκίνητα για κάθε Thing και Client. Ωστόσο, μπορεί να αναπτυχθεί οποιοδήποτε αλγοριθμικό σύστημα το οποίο παίρνει τις αποφάσεις, με λιγότερη έως καθόλου ανθρώπινη παρέμβαση.

Σε κάθε περίπτωση, τα παραπάνω δεδομένα με τις λίστες πρέπει να διατηρούνται σε βάση δεδομένων και όχι σε προσωρινή μνήμη. Αυτό είναι σημαντικό τόσο για την ασφάλειά τους (για παράδειγμα σε περίπτωση αποτυχίας στην εκτέλεση του Provisioning Server), όσο και για τη μελλοντική επέκταση και συντήρηση του δικτύου (προσθήκη λογισμικού, μεταφορά μεταξύ server κλπ). Επιπρόσθετα, η βάση δεδομένων αυτή θα αποτελέσει το access point μεταξύ της κλάσης Provisioning Server που θα υλοποιηθεί και της λογικής διαχείρισής του. Το περιβάλλον ανάπτυξης βάσης δεδομένων που προτιμήθηκε είναι η MySQL καθώς είναι ανοιχτού κώδικα και εύκολη στη χρήση και διαχείριση της. Η σχεδίαση της βάσης είναι σχετικά απλή και περιέχει τους εξής πίνακες:

- **rooms**: λίστα με τα Jid των rooms
- **things**: λίστα με τα Jid των Things
- **clients**: λίστα με τα Jid των Clients
- **thing_to_room**: πίνακας σύνδεσης (ενός) room με (πολλά) Things
- **client_to_room**: πίνακας σύνδεσης (ενός) room με (πολλούς) Clients
- **admins**: πίνακας για τη μελλοντική ανάπτυξη ενός Administration Panel με τα username και password των χρηστών του panel

Παρακάτω διακρίνεται το ER diagram (διάγραμμα οντοτήτων-συσχετίσεων) της βάσης δεδομένων.



Παρατηρείται το πεδίο voice του πίνακα client_to_room που αντιστοιχεί στο δικαίωμα αποστολής ή μη εντολών ελέγχου, του Client μέσα στο room.

Παρακάτω περιγράφονται τα βασικά σενάρια χρήσης της κλάσης Provisioning Server.

Use Case Scenario 1: Δημιουργία αντικειμένου της κλάσης και αρχικοποίηση

1. Ο Provisioning Server δέχεται τα στοιχεία εισόδου του ως XMPP Client
2. Με την χρήση των παραπάνω στοιχείων γίνεται είσοδος στο XMPP δίκτυο
3. Δημιουργία αντικειμένων / αρχικοποίηση των λιστών της κλάσης
4. Δήλωση του αντικειμένου joinRequestHandler της κλάσης ως IQ Request Listener για τη λήψη και απόκριση σε μηνύματα joinRequest των Things
5. Ανανέωση των λιστών της κλάσης από τη βάση δεδομένων (Use Case Scenario 2)
6. Δημιουργία των room με χρήση των λιστών της κλάσης (Use Case Scenario 3)

Το παραπάνω σενάριο αντιστοιχεί στην υλοποίηση του constructor της κλάσης ο οποίος είναι όπως και στις κλάσης των άλλων οντοτήτων private (ιδιωτικός). Η δημιουργία αντικειμένου της κλάσης γίνεται από static μέθοδο της κλάσης. Πριν το αντίστοιχο απόσπασμα κώδικα θα παρουσιαστούν οι λίστες της κλάσης.

- thingToRoom: αποτελεί λίστα που αντιστοιχεί το Jid κάθε Thing με μία λίστα η οποία αντιστοιχεί το κάθε NodeId του Thing με το Jid του αντίστοιχου room
- roomThings: αποτελεί λίστα που αντιστοιχεί το Jid κάθε room με μία λίστα με τα Jid των Things του room
- roomSubscribers: αποτελεί λίστα που αντιστοιχεί το Jid κάθε room με μία λίστα η οποία αντιστοιχεί το κάθε Jid των Clients του room με τον τύπο συμμετοχής στο room (Voice / NoVoice)
- jidToRoom: αποτελεί λίστα που αντιστοιχεί το Jid κάθε room με το αντίστοιχο αντικείμενο MultiUserChat

Σημειώνεται ότι αυτές οι τέσσερις λίστες αρκούν για την ανάκτηση όλων των δεδομένων από τη βάση. Για παράδειγμα δε χρειάζεται λίστα για όλα τα Things καθώς η πληροφορία αυτή υπάρχει στη λίστα thingToRoom.

Ακολουθούν τα αντίστοιχα αποσπάσματα κώδικα της δήλωσης των λιστών και του constructor της κλάσης Provisioning Server βάση του παραπάνω σεναρίου χρήσης.

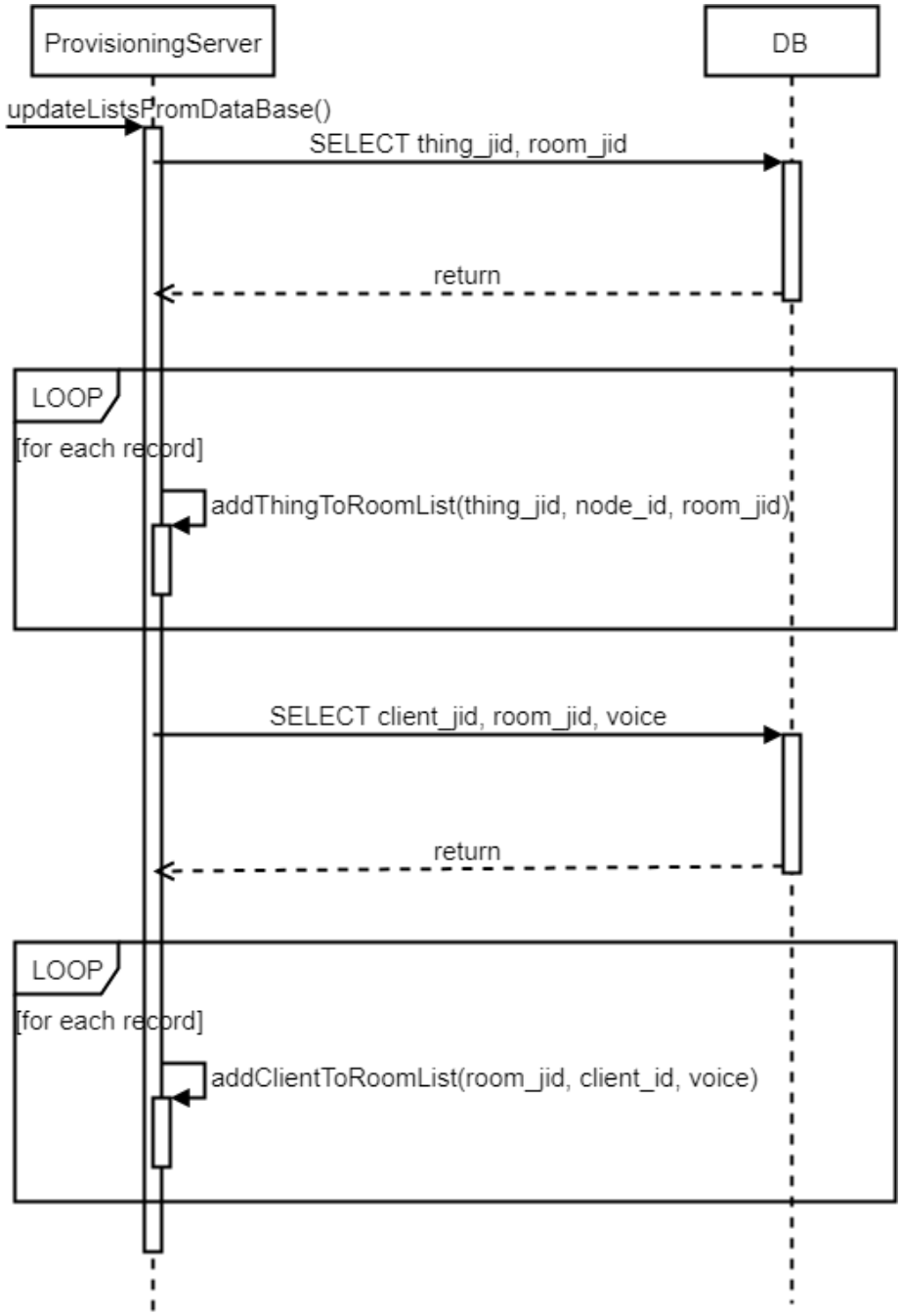
```
private Map<Jid,Map<String,EntityBareJid>> thingToRoom;
private Map<EntityBareJid,List<Jid>> roomThings;
private Map<EntityBareJid,Map<Jid,RoomSubscriberType>> roomSubscribers;
private Map<EntityBareJid,MultiUserChat> jidToRoom;
```

```
private ProvisioningServer(String user, String pass, String domain,
String resource) throws Exception{

	super(user,pass,domain,resource);

	thingToRoom = new HashMap<Jid,Map<String,EntityBareJid>>();
	roomThings = new HashMap<EntityBareJid,List<Jid>>();
	roomSubscribers = new
			HashMap<EntityBareJid,Map<Jid,RoomSubscriberType>>();
	jidToRoom = new HashMap<EntityBareJid,MultiUserChat>();

	login();
	connection.registerIQRequestHandler(joinRequestHandler);
	mucService = mucManager.getXMPPServiceDomains().get(0);

	updateListsFromDataBase(true);
}


public static ProvisioningServer createProvisioningServer(String user,
String pass, String domain, String resource){
	try{
		return new ProvisioningServer(user,pass,domain,resource);
	}
	catch(Exception e){
		return null;
	}
}


public static ProvisioningServer createProvisioningServer(String user,
String pass, String domain){
	try{
		return new ProvisioningServer(user,pass,domain,"");
	}
	catch(Exception e){
		return null;
	}
}
```

Use Case Scenario 2: Ανανέωση των λιστών από τη βάση δεδομένων

1. Ο Provisioning Server συνδέεται με τη βάση δεδομένων
2. Μέσω ερωτήματος στη βάση λαμβάνει τα Jid των Things με το Jid του room για καθένα
3. Για κάθε εγγραφή της απάντησης προσθέτει τα απαραίτητα στοιχεία στις λίστες των rooms (jidToRoom, roomThings) και των Things (thingToRoom)
4. Μέσω ερωτήματος στη βάση λαμβάνει συνδέσεις client με room στη μορφή Jid του Client - Jid του room - voice
5. Για κάθε εγγραφή της απάντησης προσθέτει τα απαραίτητα στοιχεία στις λίστες των room (roomSubscribers)

Το ακολουθιακό διάγραμμα του σεναρίου διακρίνεται παρακάτω.



Διακρίνεται ότι το σενάριο θα υλοποιηθεί ως μέθοδος της κλάσης με όνομα updateListsFromDatabase().

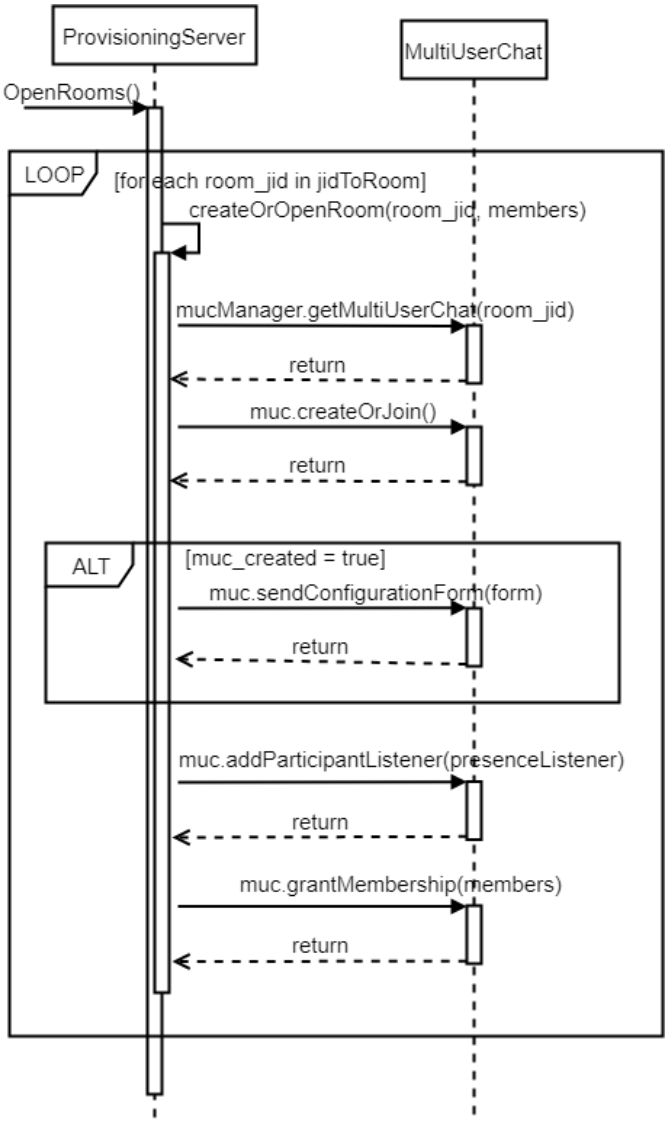
Η διαδικασία προσθήκης Thing και Client σχεδιάστηκε ώστε να υλοποιηθεί σε ξεχωριστές συναρτήσεις, τις addThingToRoomList() και addClientToRoomList() αντίστοιχα. Στη συνέχεια παρουσιάζονται τα αντίστοιχα αποσπάσματα κώδικα.

```
private void updateListsFromDataBase(boolean init){

	String Q1 = "SELECT `thing`.`jid` AS `thing`, `room`.`jid` AS `room`"+
			"FROM `thing_to_room` AS `bind`, `things` AS `thing`,
	`rooms` AS `room`"+
			"WHERE `thing`.`id` = `bind`.`thing_id` AND `room`.`id` =
	`bind`.`room_id`",

	   Q2 = "SELECT `client`.`jid` AS `client`, `room`.`jid` AS `room`,
	`bind`.`voice` "+
	    "FROM `client_to_room` AS `bind`, `clients` AS `client`, `rooms` AS
	`room` "+
		"WHERE `client`.`id`= `bind`.`client_id` AND `room`.`id` =
	`bind`.`room_id`";

	try {
		Class.forName("com.mysql.jdbc.Driver");
		Connection con = (Connection)
		DriverManager.getConnection(DBURL, DBUSER, DBPASS);

		Statement st = (Statement) con.createStatement();
		ResultSet rs = st.executeQuery(Q1);
		Jid thing_jid; EntityBareJid room_jid; String node_id;
		while (rs.next()){
			room_jid  =
			JidCreate.entityBareFrom(rs.getString("room"));
			thing_jid = JidCreate.from(rs.getString("thing"));
			node_id = thing_jid.getResourceOrEmpty().toString();

			addThingToRoomList(thing_jid, node_id, room_jid);
			jidToRoom.put(room_jid, null);
		}
		rs.close(); st.close();


		st = (Statement) con.createStatement();
		rs = st.executeQuery(Q2);
		Jid client_jid; boolean voice;
		while (rs.next()){
			room_jid  =
			JidCreate.entityBareFrom(rs.getString("room"));
			client_jid = JidCreate.from(rs.getString("client"));
			voice = rs.getString("voice").equals("yes");

			addClientToRoomList(room_jid, client_jid, voice);
		}
		rs.close(); st.close();

		OpenRooms();

	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

Use Case Scenario 3: Δημιουργία των room με χρήση των λιστών της κλάσης

- Για κάθε Jid δωματίου στη λίστα jidToRoom
    1. Δημιουργία της λίστας με τα μέλη του room με τη χρήση της λίστας roomThings και roomSubscribers
    2. Δημιουργία του αντικειμένου MultiUserChat για το δωμάτιο
    3. Εάν δεν υπάρχει το δωμάτιο αποστολή φόρμας με τις ρυθμίσεις του δωματίου
    4. Προσθήκη του participantListener της κλάσης ProvisioningServer για την ενημέρωση εισόδου συμμετεχόντων στο room
    5. Προσθήκη των συμμετεχόντων του δωματίου μέσω της λίστας με τα μέλη του
    6. Σύνδεση του jid του δωματίου με το αντικείμενο MultiUserChat στη λίστα jidToRoom

Παρακάτω παρουσιάζεται το ακολουθιακό διάγραμμα του σεναρίου.



Όπως και το προηγούμενο σενάριο, έτσι και αυτό αντιστοιχεί στην υλοποίηση μίας μεθόδου της κλάσης. Η μέθοδος σε αυτή την περίπτωση θα είναι η OpenRooms(). Οι διαδικασίες εισόδου ή δημιουργίας και εισόδου ενός room θα υλοποιηθούν ως ξεχωριστή μέθοδος με

όνομα createOrOpenRoom(). Η μέθοδος αυτή καλείται από την OpenRooms() για κάθε δωμάτιο της λίστας jidToRoom.

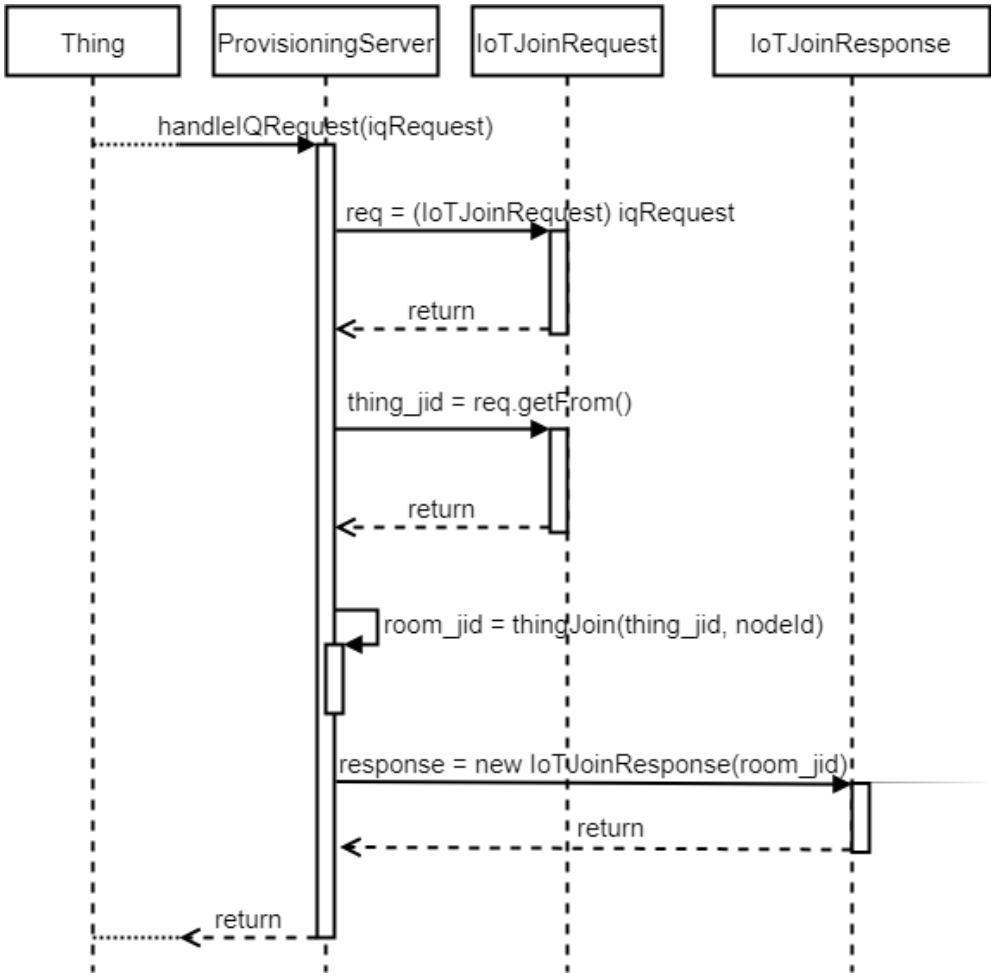
Ακολουθεί το απόσπασμα κώδικα για τις δύο παραπάνω μεθόδους.

```
private void OpenRooms() {
	Set<EntityBareJid> keySet = jidToRoom.keySet();
	for (EntityBareJid room : keySet){
		List<Jid> members = new ArrayList<Jid>();
		if (roomThings.get(room)!=null)
			members.addAll(roomThings.get(room));
		if (roomSubscribers.get(room)!=null)
			members.addAll(roomSubscribers.get(room).keySet());
		createOrOpenRoom(room.asEntityBareJidString(), members);
	}
}

private MultiUserChat createOrOpenRoom(String roomJidStr,List<Jid>
members){
	try {
		MultiUserChat muc = mucManager
		.getMultiUserChat(JidCreate.entityBareFrom(roomJidStr));
		MucCreateConfigFormHandle form = muc
		.createOrJoin(Resourcepart.from("Provisioning"));
		if (form!=null){
			Form ansForm = muc
				.getConfigurationForm()
				.createAnswerForm();
			ansForm
			.setAnswer("muc#roomconfig_persistentroom",true);
			ansForm
			.setAnswer("muc#roomconfig_membersonly",true);
			ansForm
			.setAnswer("muc#roomconfig_moderatedroom",true);

			muc.sendConfigurationForm(ansForm);
			muc.join(Resourcepart.from("Provisioning"));
		}
		else for (Affiliate member : muc.getMembers())
			muc.revokeMembership(member.getJid());
		muc.addParticipantListener(presenceListener);
		for (Jid member : members) muc.grantMembership(member);
		jidToRoom.put(muc.getRoom(), muc);
		return muc;
	} catch (Exception e) {
		e.printStackTrace();
		return null;
	}
}
```

Use Case Scenario 4: Απόκριση σε αίτημα joinRequest ενός Thing

1. Ο Provisioning Server ενημερώνεται για τη λήψη του IQ Request μέσω του joinRequestHandler
2. Μετατροπή του IQ Stanza στο αντικείμενο IoTJoinRequest
3. Ανάκτηση του Jid και του NodeInfo του Thing που έστειλε το request μέσω του στοιχείου IoTJoinRequest
4. Στην περίπτωση που υπάρχει ήδη το room του Thing στη λίστα τότε ανάκτηση του Jid αυτού, διαφορετικά δημιουργία νέου room με χρήση του NodeId του Thing και ανάκτηση του Jid του
5. Δημιουργία νέου στοιχείου IQ Response τύπου IoTJoinResponse με χρήση του Jid του δωματίου
6. Αποστολή του IQ Response πίσω στο Thing ως απάντηση

Παρακάτω φαίνεται το ακολουθιακό διάγραμμα του σεναρίου.



Παρατηρείται ότι το σενάριο αυτό αντιστοιχεί στην υλοποίηση της συνάρτησης handleIQRequest() του IQRequestHandler της κλάσης ProvisioningServer. Ο handler (χειριστής) αυτός δηλώνεται κατά τη δημιουργία της κλάσης και ειδοποιείται για κάθε εισερχόμενο IQ Stanza τύπου IoTJoinRequest, επιστρέφοντας το αντίστοιχο response. Επιπρόσθετα σχεδιάστηκε η υλοποίηση της μεθόδου thingJoin() η οποία δέχεται το Jid και το nodeId ενός Thing και επιστρέφει το Jid του room στο οποίο θα εισέλθει. Στην περίπτωση που δεν υπάρχει room για το Thing στη λίστα, η μέθοδος δημιουργεί νέο.

Στη συνέχεια παρουσιάζονται τα αποσπάσματα κώδικα για τον IQRequestHandler και τη μέθοδο thingJoin().

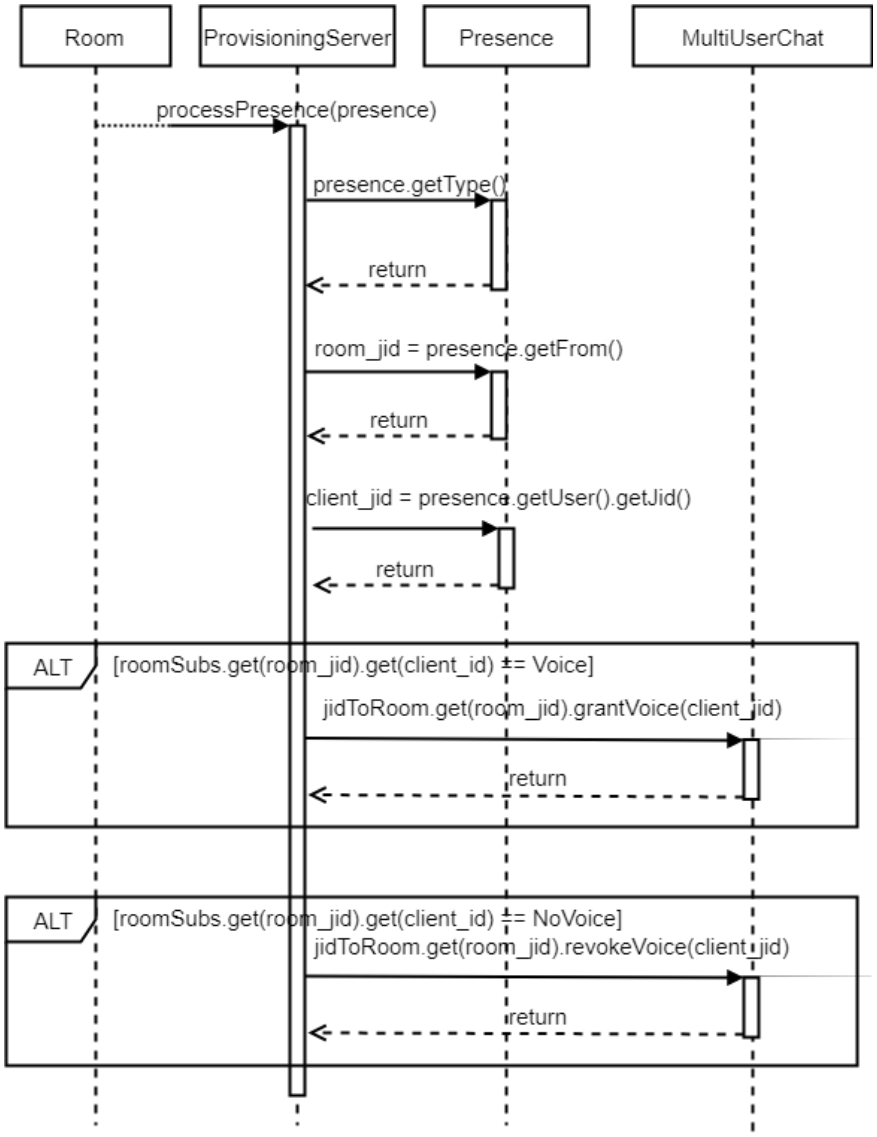
```
private IQRequestHandler joinRequestHandler = new IQRequestHandler(){
	public IQ handleIQRequest(IQ iqRequest) {
		IQ res = null;
		IoTJoinRequest req = (IoTJoinRequest) iqRequest;
		NodeInfo nodeInfo = req.getNodeInfo();
		Jid from = req.getFrom();
		EntityBareJid roomJid = thingJoin(from,nodeInfo);
		res = new IoTJoinResponse(roomJid);
		res.setStanzaId(iqRequest.getStanzaId());
		res.setTo(iqRequest.getFrom());
		res.setType(Type.result);
		return res;
	}

	public Mode getMode() {return Mode.sync;}
	public Type getType() {return Type.get;}
	public String getElement() {return "joinRequest";}

	public String getNamespace() {
		return Constants.IOT_PROVISIONING_NAMESPACE;
	}
};

private EntityBareJid thingJoin(Jid from, NodeInfo node){
	Map<String,EntityBareJid> userMap = thingToRoom.get(from);
	if (userMap==null) addThingToRoom(from,node.getNodeId());
	else {
		EntityBareJid roomJid;
		roomJid = userMap.get(node.getNodeId());
		if (roomJid==null) addThingToRoom(from,node.getNodeId());
		else return roomJid;
	}
	return thingToRoom.get(from).get(node.getNodeId());
}
```

Use Case Scenario 5: Διαχείριση του voice κατά την είσοδο Client σε room

1. Ο Provisioning Server ενημερώνεται για το Presence συμμετέχοντα στο room
2. Ελέγχεται ότι το Presence Stanza αφορά την είσοδο στο δωμάτιο
3. Γίνεται ανάκτηση του Jid του room από το Presence Stanza
4. Γίνεται ανάκτηση του Jid του XMPP Client από το Presence Stanza

5. Ελέγχεται ότι ο XMPP Client που αφορά το Presence αποτελεί Client μέσω της λίστας roomSubscribers
6. Γίνεται ανάκτηση του αντικειμένου MultiUserChat του room μέσω της λίστας JidToRoom
7. Στην περίπτωση που η εγγραφή του Client ως μέλος στο room είναι τύπου Voice, δίνεται το δικαίωμα Voice (αποστολής μηνυμάτων) στον αντίστοιχο συμμετέχοντα στο room, ενώ στην περίπτωση που είναι τύπου NoVoice αφαιρείται το δικαίωμα Voice από τον συμμετέχοντα

Στη συνέχεια διακρίνεται το ακολουθιακό διάγραμμα του σεναρίου.



Διακρίνεται ότι το σενάριο αυτό αντιστοιχεί στην υλοποίηση του PresenceListener της κλάσης που δηλώνεται ως ParticipantListener κατά τη δημιουργία και είσοδο του Provisioning Server σε κάθε room.

Τέλος, ακολουθεί το απόσπασμα κώδικα για τον PresenceListener.

```
private PresenceListener presenceListener = new PresenceListener(){

      public void processPresence(Presence presence) {
            if (!presence.getType().equals(Presence.Type.available))
                  return;

            EntityBareJid roomJid = presence
                                    .getFrom()
                                    .asEntityBareJidIfPossible();

            MUCUser mucUserExt = presence
            .getExtension("x","http://jabber.org/protocol/muc#user");

            Jid user = mucUserExt.getItem().getJid();

            RoomSubscriberType subType;

            if (roomSubscribers.get(roomJid)==null) subType = null;
            else subType = roomSubscribers.get(roomJid).get(user);

            if (subType==null) return;
            else if (subType.equals(RoomSubscriberType.NoVoice)){
                  try {
                        jidToRoom
                        .get(roomJid)
                        .revokeVoice(user.getResourceOrEmpty());
                  } catch (Exception e) {e.printStackTrace();}
            }
            else if (subType.equals(RoomSubscriberType.Voice)){
                  try {
                        jidToRoom
                        .get(roomJid)
                        .grantVoice(user.getResourceOrEmpty());
                  } catch (Exception e) {e.printStackTrace();}
            }
      }
};
```

4.4 Ενδεικτικές χρήσεις της βιβλιοθήκης

Η ανάπτυξη της βιβλιοθήκης επέκτασης της Smack ολοκληρώθηκε. Παρέχονται οι τρεις οντότητες Thing, Client και Provisioning Server ως κλάσεις με high level (υψηλού επιπέδου) μεθόδους για την εύκολη και γρήγορη ανάπτυξη εφαρμογών Internet of Things. Η βιβλιοθήκη αυτή μπορεί να ενσωματωθεί σε πληθώρα συσκευών που μπορούν να τρέξουν Java, όπως Smartphones (π.χ. Android), μικροϋπολογιστές (π.χ. Raspberry Pi), υπολογιστές κλπ.

Η ολοκλήρωση μίας βιβλιοθήκης ωστόσο αποτελεί ελλιπής χωρίς καμία ενδεικτική/πειραματική εφαρμογή. Για τον σκοπό αυτό αρχικά θα αναπτυχθεί ένας γραφικός πίνακας διαχείρισης (Administration Panel) για τον Provisioning Server και ύστερα θα αναπτυχθούν πειραματικά Things, Clients και Provisioning Server ώστε να διαπιστωθεί η σωστή λειτουργία της βιβλιοθήκης

4.4.1 Η ανάπτυξη ενός Administrator Panel για τον Provisioning Server/Component (Ιστοσελίδα)

Η βιβλιοθήκη έχει αναπτυχθεί ώστε τα δεδομένα του Access Control του Provisioning Server να παρέχονται σε βάση δεδομένων (συγκεκριμένα MySQL). Οποιαδήποτε υλοποίηση με χρήση της βιβλιοθήκης, επικοινωνεί με τη βιβλιοθήκη μέσω της βάσης αυτής, δηλαδή σκοπός της είναι η διαχείριση της βάσης. Συνεπώς η βάση αυτή αποτελεί proxy επικοινωνίας της λογικής της εφαρμογής με τη βιβλιοθήκη.

Ενδεικτικός τρόπος διαχείρισης της βάσης δεδομένων αποτελεί ένα γραφικό περιβάλλον διαχείρισης. Αυτό το περιβάλλον μπορεί να είναι διαχειρίσιμο από ανθρώπους, υπεύθυνους για το Provisioning του IoT δικτύου και συνεπώς γενικότερα για την ασφαλή λειτουργία του δικτύου. Σκοπός του Provisioning, όπως έχει αναλυθεί, είναι η παροχή εμπιστοσύνης στους κατόχους των Things σχετικά με την πρόσβαση στα δεδομένα τους, όπου η χρήση του ανθρώπινου παράγοντα υπερτερεί κάποιου γενικευμένου αλγορίθμου. Αυτό δε σημαίνει πως δε μπορούν να αναπτυχθούν αλγόριθμοι που εμπνέουν την απαραίτητη εμπιστοσύνη. Το κομμάτι αυτό ωστόσο ξεπερνά το εύρος μελέτης της τρέχουσας εργασίας.

Το γραφικό περιβάλλον διαχείρισης του Provisioning Server επιλέχθηκε να γίνει σε μορφή ιστοσελίδας. Η πρόσβαση στην ιστοσελίδα αυτή πρέπει να περιορίζεται στους ανθρώπους που πραγματικά χειρίζονται τον Provisioning Server, συνεπώς πρέπει να υπάρχει διαπίστευση στοιχείων (username & password). Το γραφικό περιβάλλον και οι λειτουργίες που παρέχονται πρέπει να εξασφαλίζουν την εύκολη και γρήγορη διαχείριση της βάσης δεδομένων. Συνοπτικά λοιπόν οι σελίδες που πρέπει να αναπτυχθούν είναι οι εξής:

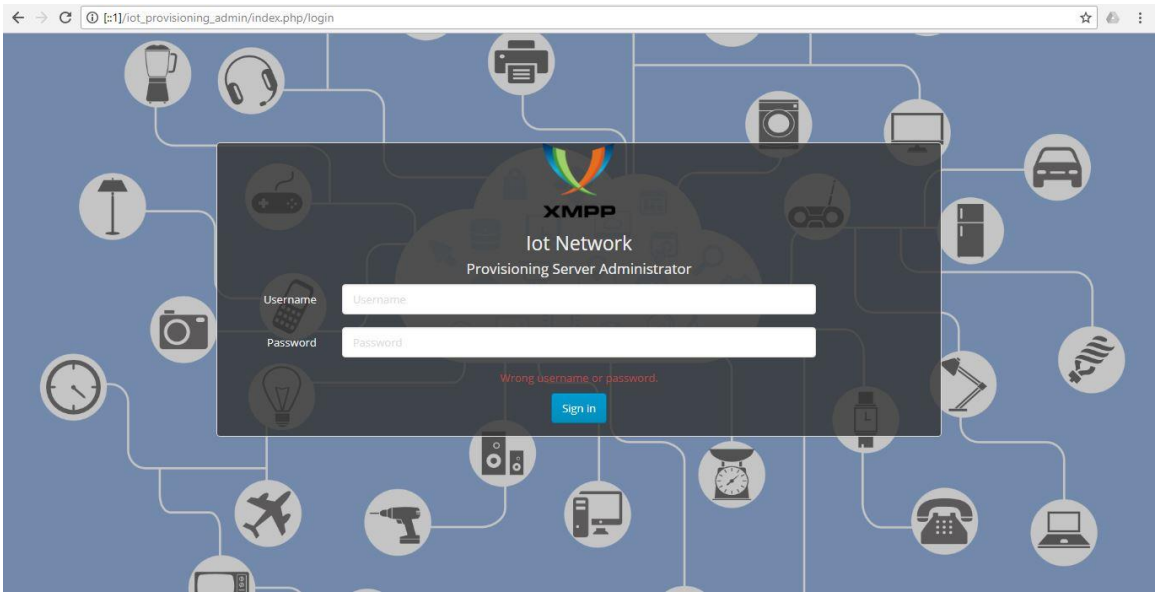
- **Σελίδα εισόδου με χρήση διαπιστευτηρίων**: σε αυτή τη σελίδα ο χρήστης πρέπει εισάγει username & password ώστε να εισέλθει στις υπόλοιπες. Η είσοδος από αυτή τη σελίδα πρέπει να αποθηκεύεται στο session με χρήση cookies ώστε οι επόμενες σελίδες να είναι προσβάσιμες μόνο όταν έχει γίνει προηγουμένως εισαγωγή των διαπιστευτηρίων
- **Σελίδα με τα rooms του δικτύου**: σε αυτή τη σελίδα υπάρχει λίστα με όλα τα rooms του IoT δικτύου και να υπάρχει δυνατότητα ταχείας αναζήτησής τους. Επίσης πρέπει να υπάρχει δυνατότητα προσθήκης νέου room.
- **Σελίδα ενός room**: η προηγούμενη σελίδα πρέπει να δίνει την επιλογή εισόδου στη διαχείριση ενός room. Στη σελίδα αυτή πρέπει να υπάρχουν όλα τα διαθέσιμα Things και οι Clients που έχουν πρόσβαση στο δωμάτιο με δυνατότητα ταχείας αναζήτησης. Επιπρόσθετα στους Clients πρέπει να είναι διαχειρίσιμο το δικαίωμα voice, η προσθήκη νέων και η διαγραφή τους από το room.
- **Σελίδα με τα Things του δικτύου**: η σελίδα αυτή περιέχει λίστα με τα Things του δικτύου και γρήγορη αναζήτηση τους. Για κάθε Thing πρέπει να υπάρχει εύκολος τρόπος προσθήκης του σε room ή αφαίρεση του από το υπάρχων. Επιπρόσθετα πρέπει να παρέχεται η δυνατότητα προσθήκης και διαγραφής Things.
- **Σελίδα με τους Clients του δικτύου**: η σελίδα αυτή, παρομοίως με την προηγούμενη περιέχει λίστα με τους Clients και γρήγορη αναζήτησή τους. Επίσης παρέχεται η δυνατότητα προσθήκης και διαγραφής τους.

Πριν γίνει συνοπτική παρουσίαση της ιστοσελίδας (αναλυτικά δίνεται ο κώδικας στο σχετικό παράρτημα), παρακάτω δίνονται οι τεχνολογίες που χρησιμοποιήθηκαν.

- Front End
    - HTML, CSS, JS: αποτελούν τα γνωστά open standard για τη δημιουργία εγγράφων ιστοσελίδων.
    - Bootstrap 3: αποτελεί Front-End Framework ανοιχτού κώδικα που προσφέρει έτοιμα HTML στοιχεία, CSS κώδικα και JS μεθόδους για τη δημιουργία responsive ιστοσελίδων με πλήρως σχεδιασμένα στοιχεία διεπαφής (panel, κουμπιά, φόρμες, λίστες, μενού πλοήγησης κλπ). Με κατάλληλη χρήση του Framework οι ιστοσελίδες που δημιουργούνται έχουν διαμόρφωση που ανταποκρίνεται στο μέγεθος της οθόνης, ώστε να εμφανίζονται ικανοποιητικά και σε SmartPhones.
    - jQuery: αποτελεί JavaScript βιβλιοθήκη επέκτασης που παρέχει μεθόδους για την επιλογή και διαχείριση HTML στοιχείων, χρήση animations, χειρισμό συμβάντων και επικοινωνίας μέσω AJAX (ασύγχρονα HTTP requests).
    - jQuery UI: αποτελεί επέκταση της βιβλιοθήκης jQuery που προσφέρει δυνατότητες για πιο προχωρημένα στοιχεία διεπαφής χρήστη ως μικρό Front End framework. Οι δυνατότητες που προσφέρει περιλαμβάνουν προχωρημένα animations καθώς και η δημιουργία πεδίων αυτόματης συμπλήρωσης autocomplete.
- Back End
    - PHP: αποτελεί βασική και την πιο δημοφιλή γλώσσα για Server-Side scripting. Είναι εύκολη στη χρήση και παρέχει όλες τις χρήσιμες δυνατότητες μίας γλώσσας καθώς μπορεί να χρησιμοποιηθεί και με αντικειμενοστραφή τρόπο.
    - CodeIgniter: αποτελεί MVC Framework ανοιχτού κώδικα για την ταχεία δημιουργία δυναμικών ιστοσελίδων σε PHP. Τα MVC Frameworks χαρακτηρίζονται από την αρχειοθέτηση του κώδικα σε τρεις κατηγορίες αρχείων: Controllers (περιέχουν όλη τη γενική λογική), Models (περιέχουν τη λογική αλληλοεπίδρασης με κάποιο μοντέλο δεδομένων) και τα Views (περιέχουν κάποιο κομμάτι εμφάνισης). Η τεχνική αυτή βοηθάει στην ταχύτερη ανάπτυξη, οργάνωση και συνεπώς ευκολότερη συντήρηση και επέκταση των project.

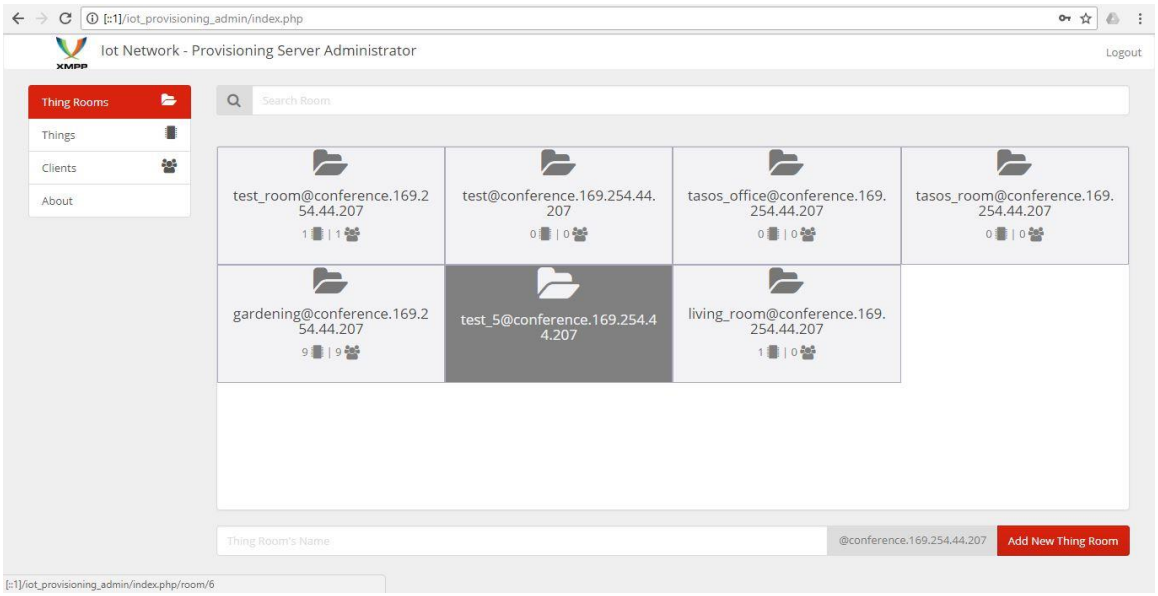
Παρακάτω γίνεται συνοπτική παρουσίαση της ιστοσελίδας Provisioning Server Administrator.

Σελίδα εισόδου με χρήση διαπιστευτηρίων



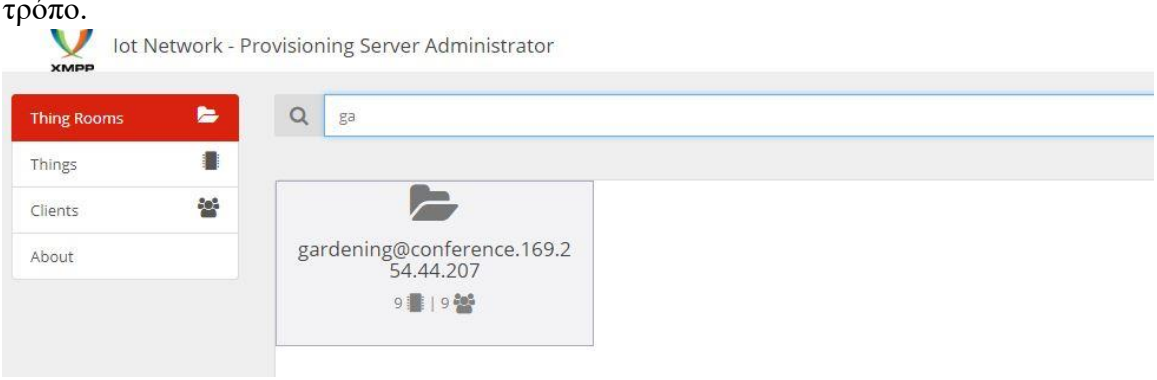
Διακρίνεται το μήνυμα σφάλματος σε περίπτωση εισαγωγής λανθασμένων διαπιστευτηρίων.

Σελίδα με τα rooms του δικτύου



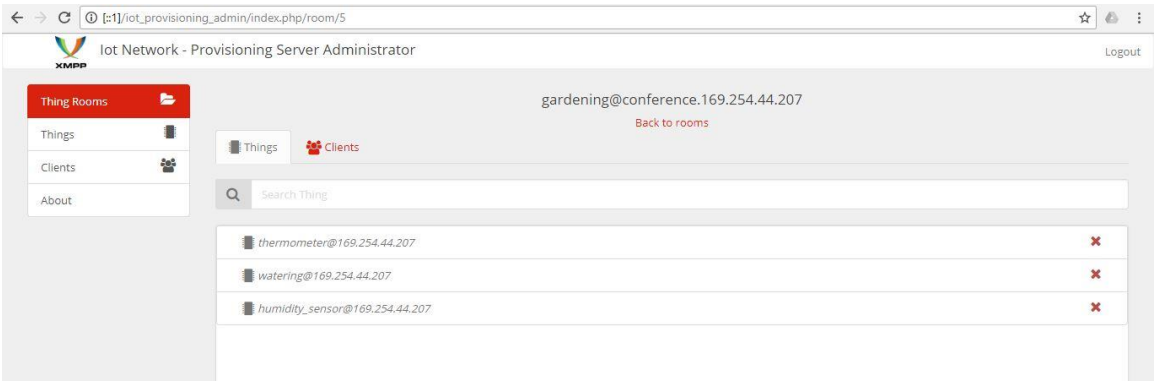
Στην παραπάνω εικόνα διακρίνεται το εφέ κατά την απόθεση (mouseover) του pointer σε κάποιο room. Στην παρακάτω εικόνα φαίνεται η χρήση της γρήγορης αναζήτησης.

Σημειώνεται πως σε όποιο σημείο υπάρχει αναζήτηση, αυτή έχει υλοποιηθεί με αυτό τον τρόπο.

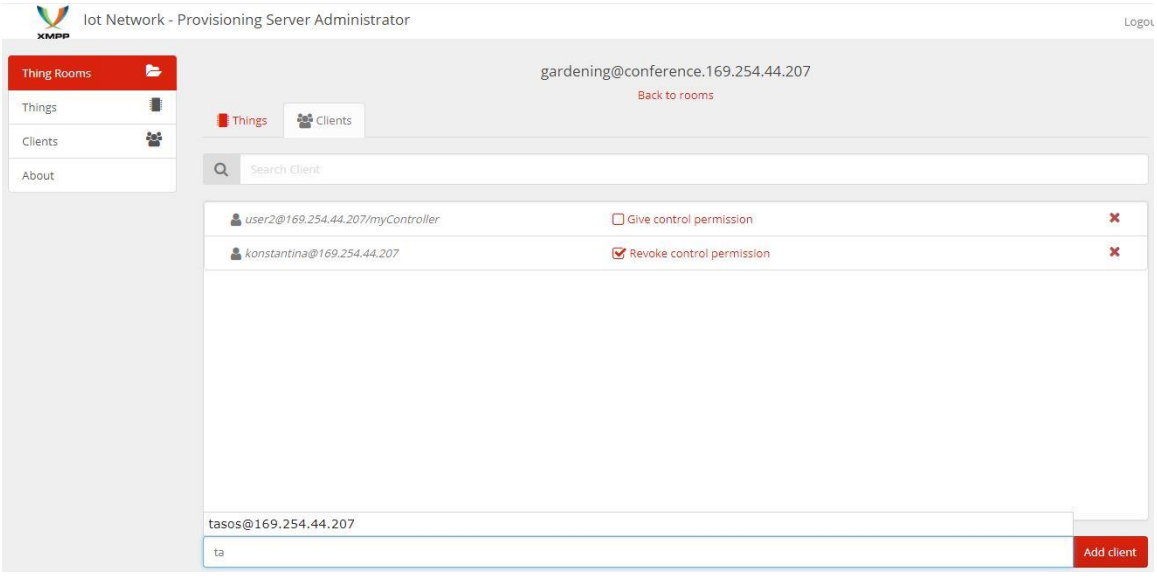


Σελίδα ενός room

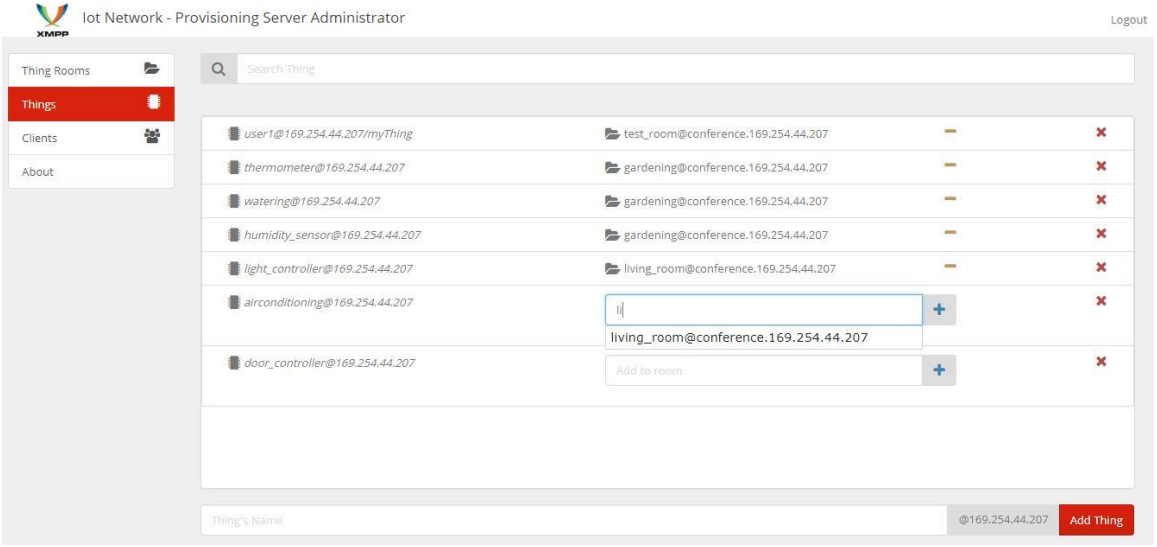
Η σελίδα αυτή έχει δύο tabs (καρτέλες). Η πρώτη αντιστοιχεί στα Thing του room.



Η δεύτερη έχει τους Client του room με τη δυνατότητα προσθήκης νέων. Παρατηρείται η χρήση αυτόματης συμπλήρωσης για την επιλογή προσθήκης νέου Client.

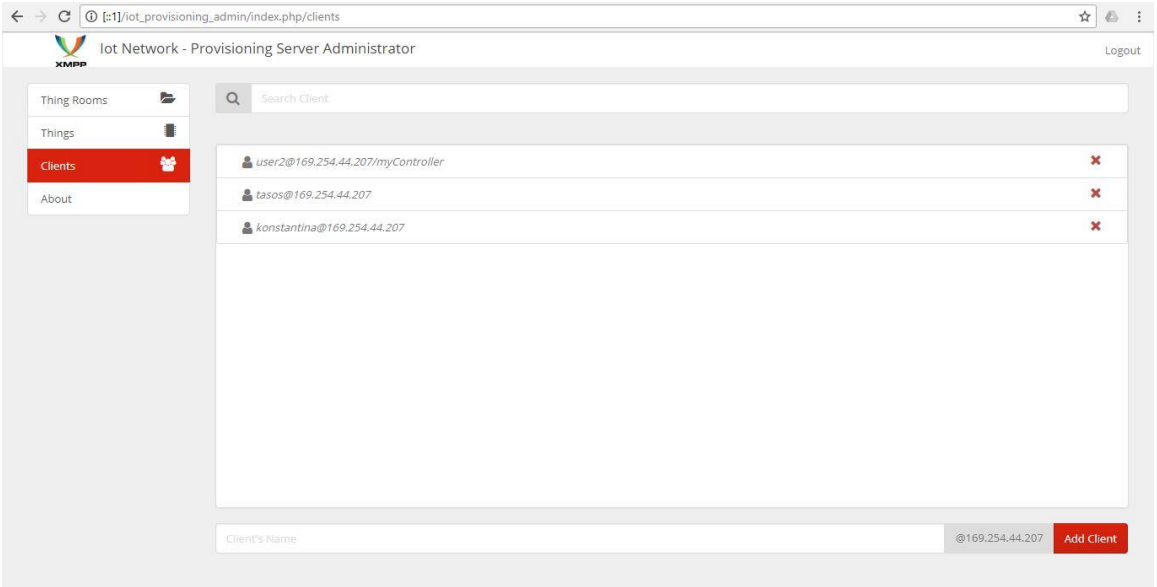


Σελίδα με τα Things του δικτύου



Διακρίνεται ο τρόπος με τον οποίο κάθε Thing μπορεί γρήγορα να προστεθεί η να διαγραφεί από κάποιο room. Κατά την προσθήκη Thing σε room η εισαγωγή γίνεται με χρήση αυτόματης συμπλήρωσης. Επιπρόσθετα διακρίνονται οι δυνατότητες εισαγωγής και διαγραφής ενός Thing από τη βάση.

Σελίδα με τους Clients του δικτύου



Όπως και στην προηγούμενη σελίδα, έτσι και σε αυτή διακρίνεται η δυνατότητα προσθήκης και αφαίρεσης Client από τη βάση.

4.4.2 Βασικές δοκιμές λειτουργίας

Με τη βιβλιοθήκη και το Administrator Panel ως ενδεικτική υλοποίηση χειρισμού του Provisioning Server, το μόνο που μένει είναι η υλοποίηση δοκιμαστικής εφαρμογής χρήσης της βιβλιοθήκης. Για τον σκοπό αυτό θα υλοποιηθούν δοκιμαστικά Things και Clients με τη χρήση της βιβλιοθήκης καθώς και ένα instance του Provisioning Server. Τα παραπάνω θα υλοποιηθούν με τη μορφή απλής δοκιμαστικής desktop εφαρμογής ώστε για την πραγματοποίηση της δοκιμής να αρκεί ένας μόνο υπολογιστής. Χάριν απλότητας η ανάλυση υλοποίησης θα παραλειφθεί. Ωστόσο, ο κώδικας περιλαμβάνεται στο αντίστοιχο παράρτημα.

4.4.2.1 Περιγραφή δοκιμαστικής εφαρμογής

Ακολουθεί συνοπτική περιγραφή της εφαρμογής:

- Η εφαρμογή υλοποιεί αυτόματα ένα instance του Provisioning Server κατά την εκκίνησή της
- Εμφανίζεται παράθυρο το οποίο δίνει την επιλογή δημιουργίας νέων δοκιμαστικών Thing και Clients απλά με την εισαγωγή του username (από το Jid) και του password ενός XMPP Client
- Κάθε Thing και Client ανοίγει σε ξεχωριστό παράθυρο

Περιγραφή δοκιμαστικού Thing:

- Κάθε Thing παράγει μία σειρά από δοκιμαστικά πεδία δεδομένων με τυχαίες τιμές ανά σταθερό χρονικό διάστημα (π.χ. 3sec). Τα πεδία αυτά είναι ίδια για κάθε Thing.
- Στο παράθυρο απεικονίζεται συγκριμένος αριθμός πεδίων κειμένου που αντιστοιχούν σε πεδία εντολών ελέγχου. Κάθε φορά που το Thing δέχεται εντολές ελέγχου, εμφανίζονται σε αυτά, τα ονάματα των πεδίων με τις αντίστοιχες τιμές τους.

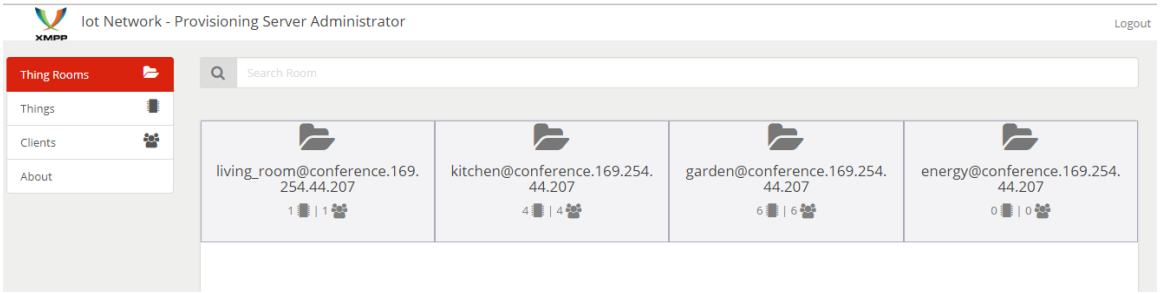
Περιγραφή δοκιμαστικού Client:

- Κατά την εκκίνηση του, ο Client προσπαθεί αυτόματα να εισέλθει σε όλα τα διαθέσιμα δωμάτια, ώστε τελικά να εισέλθει σε όλα τα δωμάτια στα οποία έχει πρόσβαση.
- Για κάθε δωμάτιο που έχει γίνει είσοδος, δημιουργείται συγκεκριμένη περιοχή στο παράθυρο με το Jid του δωματίου.
- Κάθε περιοχή δωματίου στο παράθυρο, διαθέτει υπο-περιοχές, μια για κάθε online Thing που βρίσκεται στο δωμάτιο. Τέτοιες υπο-περιοχές προστίθενται και αφαιρούνται δυναμικά σε πραγματικό χρόνο καθώς δημιουργούνται και καταστρέφονται Things.
- Κάθε περιοχή ενός Thing διαθέτει το όνομα (NodeId) του Thing. Κάθε φορά που λαμβάνονται νέα δεδομένα από το Thing η περιοχή αυτή ανανεώνεται με τον πίνακα πεδίων δεδομένων. Με αυτό τον τρόπο ο Client παρέχει προβολή δεδομένων σε πραγματικό χρόνο, όλων των Things στα οποία έχει πρόσβαση.
- Υπάρχουν συγκεκριμένα πεδία για συμπλήρωση εντολών ελέγχου. Τα πεδία αυτά είναι ίδια για κάθε Client. Στη συνέχεια ακολουθεί επιλογή του δωματίου και του Thing στο οποίο θα αποσταλούν οι εντολές καθώς και κουμπί υποβολής. Με αυτό τον τρόπο ο Client μπορεί να αποστείλει εντολές ελέγχου σε κάθε προσβάσιμο Thing. Σημειώνεται πως στην περίπτωση

που δεν υπάρχει δικαίωμα Voice για κάποιο δωμάτιο, οποιεσδήποτε εντολές σταλούν θα απορριφθούν.

4.4.2.2 Διαμόρφωση του IoT δικτύου

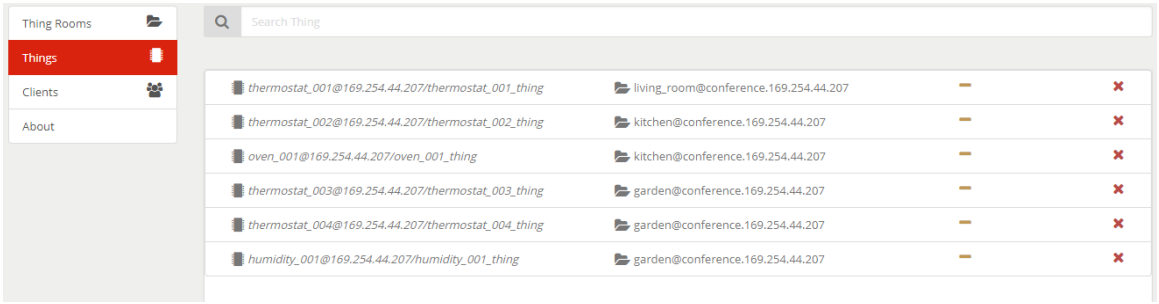
Για την πραγματοποίηση των δοκιμών, αρχικά θα γίνει διαμόρφωση ενός σεναρίου δικτύου στη βάση. Αυτό μπορεί να γίνει μέσω του Administrator Panel.



Τα δωμάτια που θα χρησιμοποιηθούν είναι τα εξής:

- living_room@conference.169.254.44.207
- kitchen@conference.169.254.44.207
- garden@conference.169.254.44.207

Τα Things που θα χρησιμοποιηθούν με τα δωμάτιά τους φαίνεται στην συνέχεια.



Συγκεκριμένα και ανά δωμάτιο υπάρχουν τα εξής Things:

- Δωμάτιο living_room@conference.169.254.44.207
  - thermostat_001@169.254.44.207
- Δωμάτιο kitchen@conference.169.254.44.207
  - thermostat_002@169.254.44.207
  - oven_001@169.254.44.207
- Δωμάτιο garden@conference.169.254.44.207
  - thermostat_003@169.254.44.207
  - thermostat_004@169.254.44.207
  - humidity_001@169.254.44.207

Τέλος οι Clients που θα χρησιμοποιηθούν είναι οι εξής δύο:

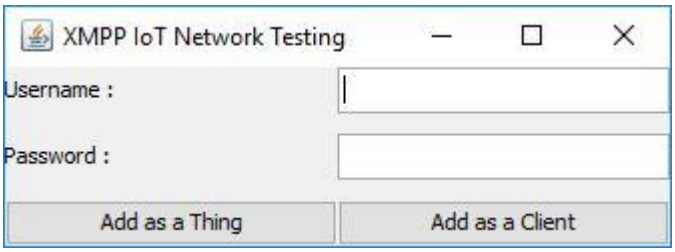
Αναλυτικότερα για τους Clients και την πρόσβασή τους στα δωμάτια ισχύει ο παρακάτω πίνακας.

Δωμάτιο \ Client	tasos@169.254.44.207	home_panel@169.254.44.207
living_room@conference.169.254.44.207	Access (Voice)	Access (Voice)
kitchen@conference.169.254.44.207	Access (Voice)	Access (No Voice)
garden@conference.169.254.44.207	Access (Voice)	No Access

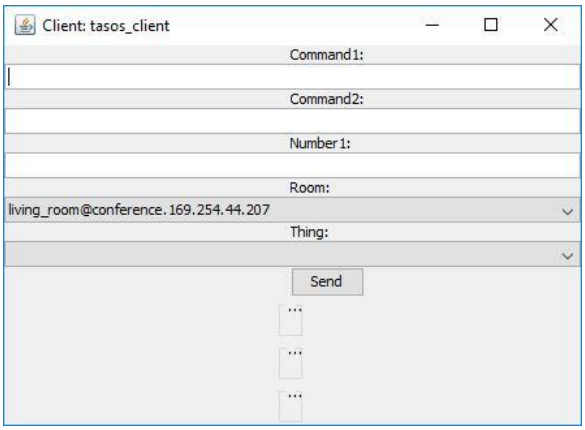
4.4.2.3 Παρουσίαση της εφαρμογής και εκτέλεση δοκιμών

Πλέον όλα είναι έτοιμα για την παρουσίαση της εφαρμογής. Η εφαρμογή χρησιμοποιεί τη βιβλιοθήκη που υλοποιήθηκε και συνεπώς οι δοκιμές θα απεικονίσουν τη λειτουργία της για την ανάπτυξη ενός IoT δικτύου στην πράξη.

Η εκκίνηση της εφαρμογής οδηγεί στο παράθυρο δημιουργίας Thing ή Client.

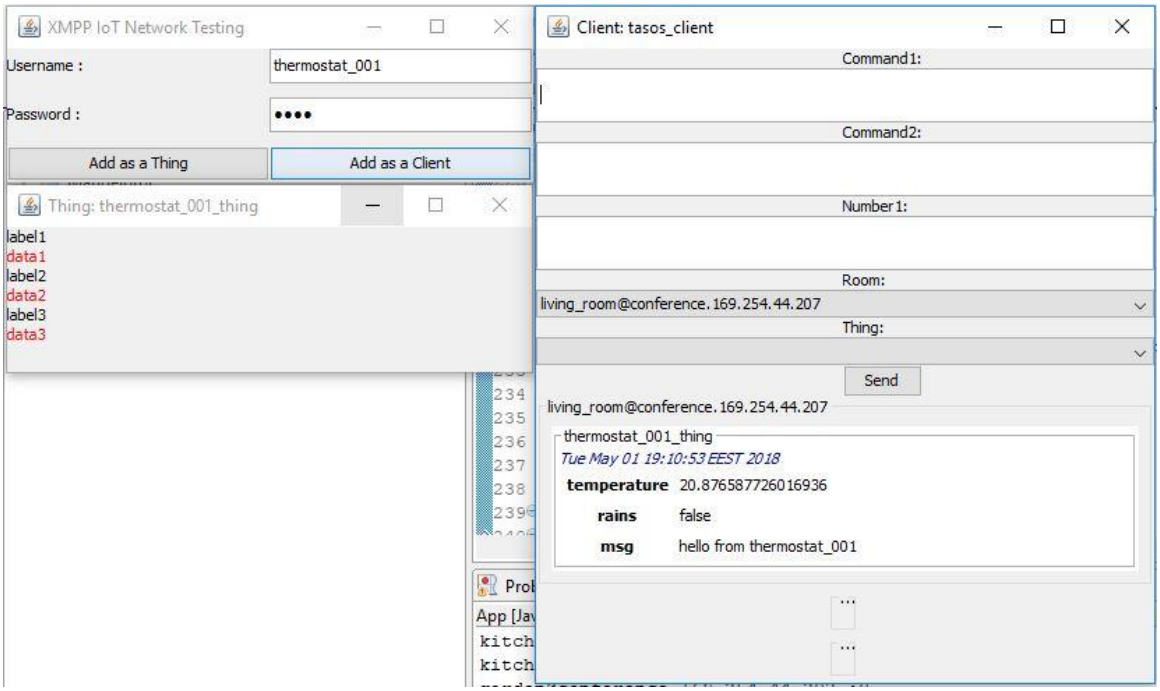


Μέσω του παραθύρου αυτού εισέρχεται στο IoT ο Client tasos (χάριν απλότητας από εδώ και στο εξής αντί για το πλήρες Jid θα αναφέρεται μόνο το username).

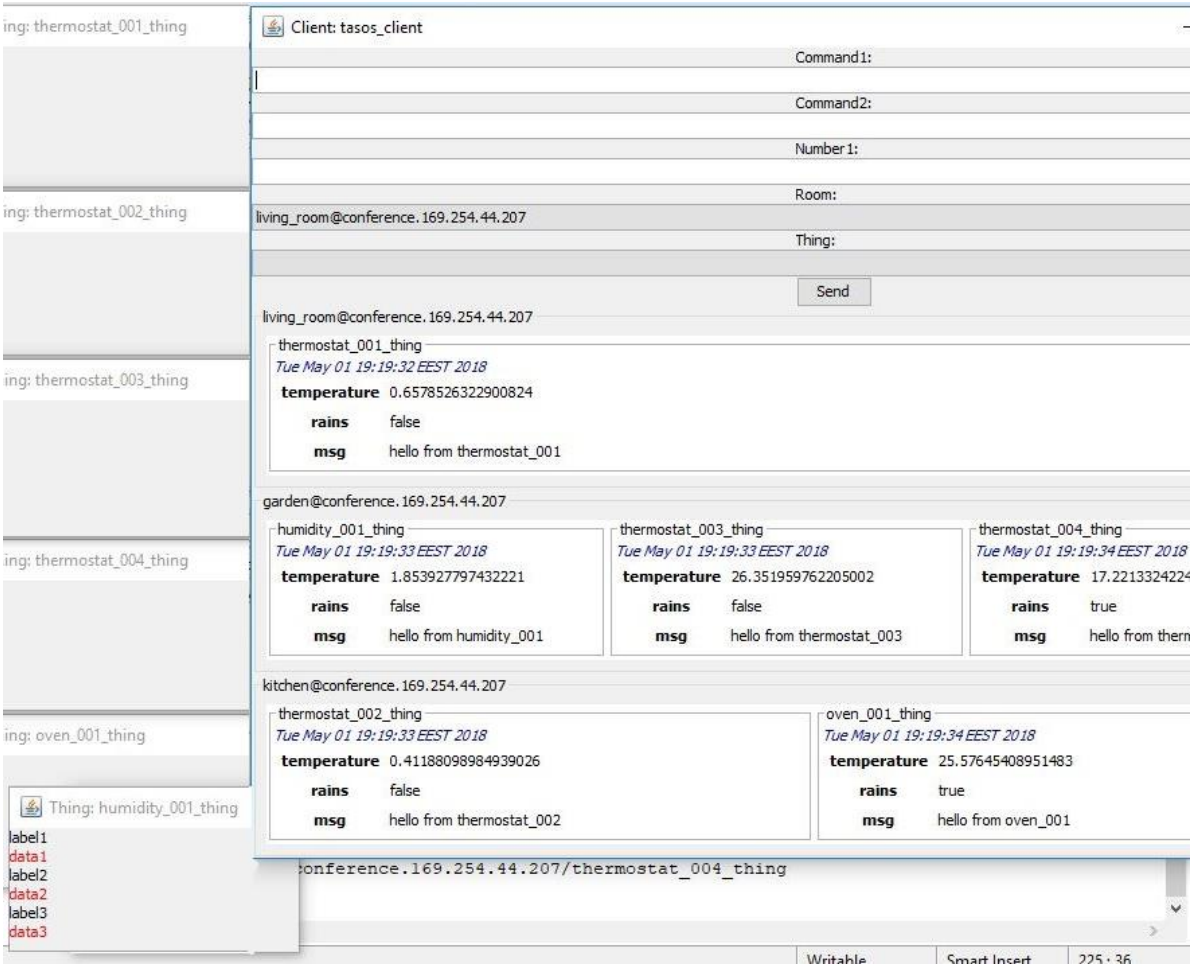


Σημειώνεται ότι οι περιοχές των δωματίων είναι σε σμίκρυνση (με σήμανση …) καθώς δεν υπάρχει κανένα Thing εντός τους.

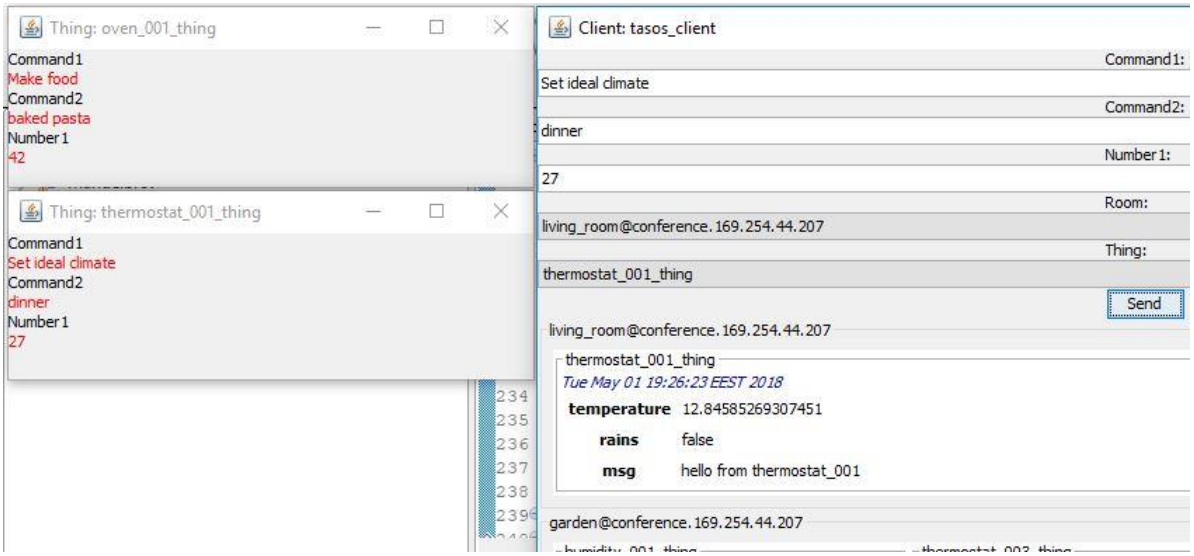
Στη συνέχεια εισέρχεται στο δίκτυο το πρώτο Thing που είναι ο thermostat_001 του δωματίου living_room. Παρατηρείται τόσο το παράθυρο του Thing όσο και η εμφάνιση του, με τα δεδομένα σε πραγματικό χρόνο, στο παράθυρο του Client.



Με τον ίδιο τρόπο εισέρχονται στο δίκτυο και όλα τα υπόλοιπα Things. Το παράθυρο του Client γεμίζει από Things καθώς έχει πρόσβαση σε όλα τα δωμάτια. Πλέον ο Client tasos δέχεται σε πραγματικό χρόνο τα δεδομένα όλων των Things του IoT δικτύου.

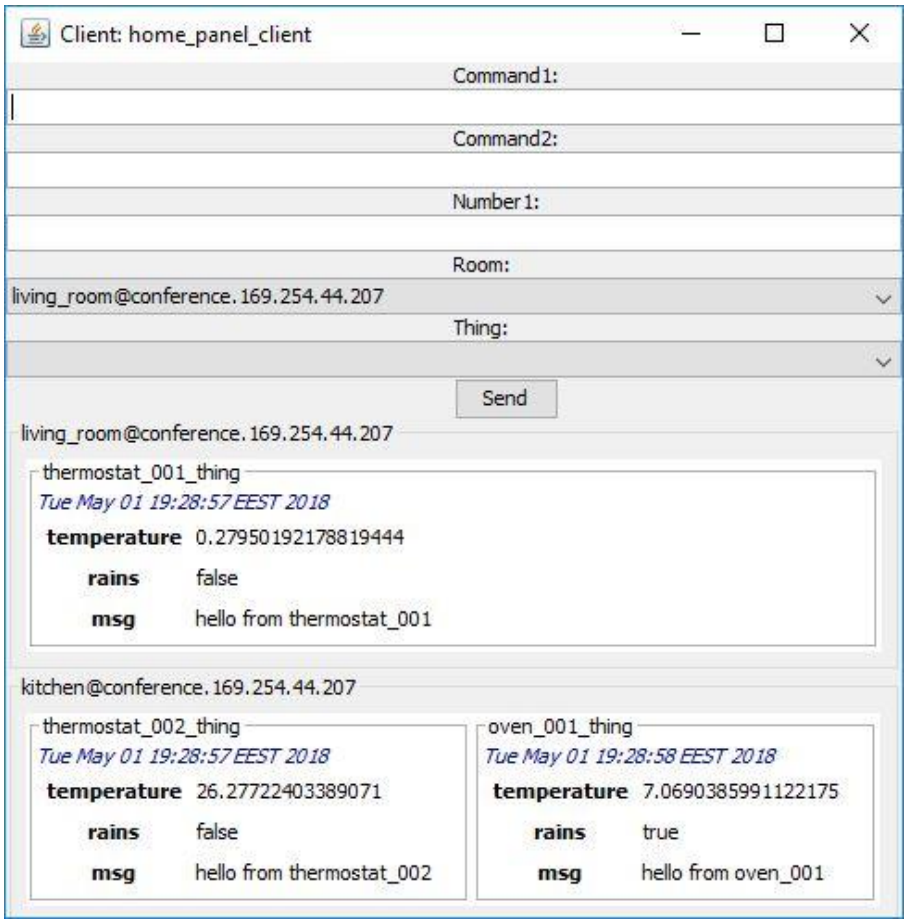


Ο Client tasos έχει δικαίωμα voice σε όλα τα δωμάτια. Συνεπώς μπορεί να στείλει εντολές ελέγχου σε όλα τα Things. Σχετικό παράδειγμα αποστολής εντολών ελέγχου φαίνεται παρακάτω.

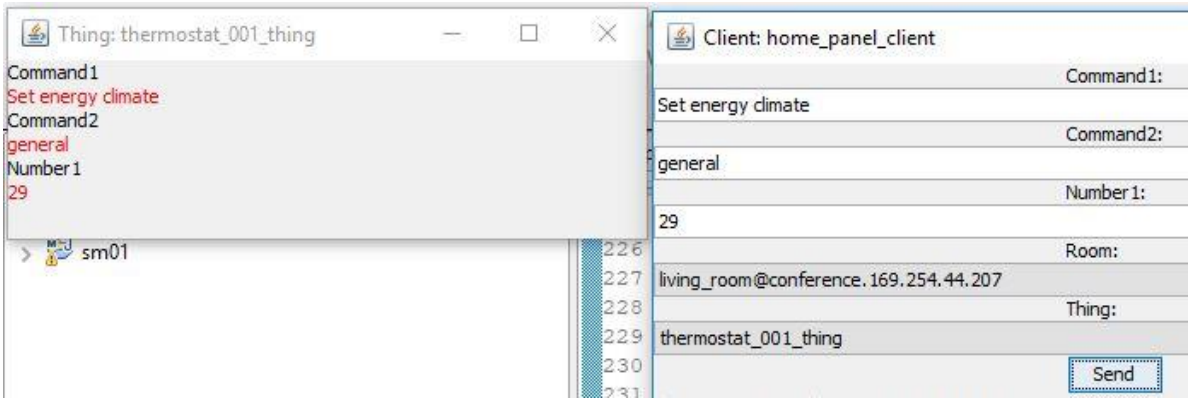


Παρατηρείται η εμφάνιση των πεδίων των εντολών στα αντίστοιχα παράθυρα των Things.

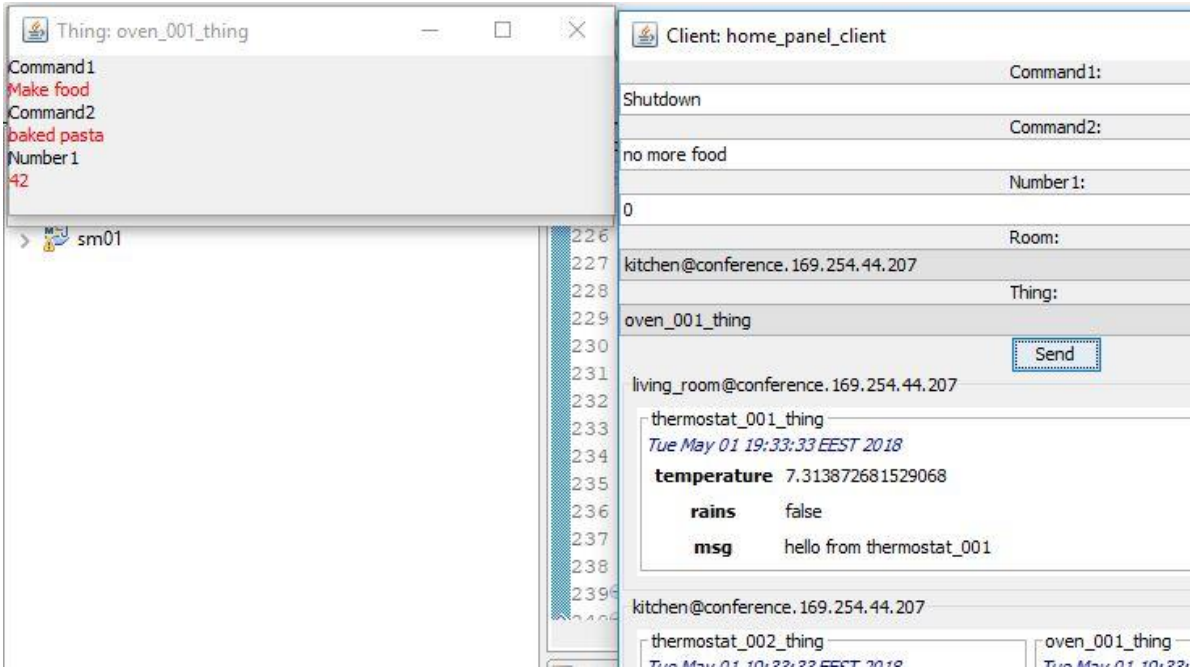
Στη συνέχεια και χωρίς να χρειαστεί η αποσύνδεση του tasos, εισέρχεται στο δίκτυο ο Client home_panel. Ο Client αυτός έχει πρόσβαση μόνο στα δωμάτια living_room και kitchen.



Ο Client home_panel έχει δικαίωμα voice στο δωμάτιο living_room και συνεπώς μπορεί να στείλει εντολές ελέγχου στα Things που υπάρχουν σε αυτό, όπως φαίνεται παρακάτω.



Ωστόσο, ο Client home_panel δεν έχει δικαίωμα voice στο δωμάτιο kitchen, οπότε οποιεσδήποτε προσπάθειες αποστολής εντολών ελέγχου σε Thing του δωματίου αυτού, αποτυγχάνουν.



5 Ανασκόπηση και ενδεικτικές βελτιώσεις - επεκτάσεις

5.1 Ανασκόπηση

Η εργασία αυτή προλογήθηκε με μία σύντομη ιστορική αναδρομή, από την ανάπτυξη των τεχνολογιών επικοινωνίας έως τη δημιουργία του διαδικτύου και από το διαδίκτυο έως το όραμα του Internet of Things. Το IoT ανερχόμενο από όραμα φουτουριστών της δεκαετίας του 60'-70', μετά την ανάπτυξη και εξέλιξη του διαδικτύου στις δεκαετίες 80'-00', αποτελεί πλέον ξεχωριστό επιστημονικό κλάδο.

Στη συνέχεια έγινε ανάλυση των σύγχρονων προκλήσεων που αντιμετωπίζει ο κλάδος, καθώς και σύντομη παρουσίαση των ενδεικτικών τεχνολογιών και προσεγγίσεων. Η ανάπτυξη του IoT εκτός από τις σχετικές τεχνολογικές προκλήσεις, εγείρει πληθώρα κοινωνικών ζητημάτων. Τέτοια ζητήματα αφορούν κυρίως τη διαχείριση πρόσβασης στα δεδομένα και τον έλεγχο των Things. Όλες οι παραπάνω παράμετροι πρέπει να ληφθούν υπόψιν στη διαμόρφωση των τεχνολογιών του κλάδου.

Στο 3^{ο} κεφάλαιο παρουσιάστηκε το πρωτόκολλο XMPP με σκοπό την χρήση του για την ανάπτυξη IoT δικτύων που ανταποκρίνονται στις προκλήσεις που εξετάστηκαν προηγουμένως. Αφού αναλύθηκαν τα χαρακτηριστικά του και μερικά ενδεικτικά δομικά στοιχεία που παρέχει, έγινε επιλογή της τελικής προσέγγισης. Η προσέγγιση αυτή μελετήθηκε ώστε να εκμεταλλεύεται στο έπακρο τα ήδη υπάρχοντα και περισσότερο αποδεκτά δομικά στοιχεία και δυνατότητες που προσφέρονται από το πρωτόκολλο, με την εισαγωγή ελάχιστων νέων. Βασικές έννοιες της προσέγγισης αποτελούν η ομαδοποίηση των Things μέσα σε Publish-Subscribe δωμάτια (Multi User Chat) και ο Provisioning Server που διαχειρίζεται τον έλεγχο πρόσβασης σε αυτά.

Στο 4^{ο} κεφάλαιο εκτελέστηκαν σχετικές υλοποιήσεις για την ανάπτυξη IoT δικτύων, βασισμένων στην προσέγγιση που αναπτύχθηκε προηγουμένως. Βασικό κομμάτι αποτέλεσε η υλοποίηση βιβλιοθήκης λογισμικού για την ανάπτυξη IoT εφαρμογών. Για τον σκοπό αυτό χρησιμοποιήθηκε η βιβλιοθήκη Smack που επιτρέπει την ανάπτυξη XMPP Client. Η βιβλιοθήκη που αναπτύχθηκε αποτελεί πλατφόρμα για την άμεση δημιουργία IoT οντοτήτων. Στη συνέχεια αναπτύχθηκε γραφικό περιβάλλον διαχείρισης ενός Provisioning Server καθώς και εφαρμογή που χρησιμοποιεί τη βιβλιοθήκη για τη δημιουργία δοκιμαστικών Things και Clients. Τέλος, και με χρήση όλων των υλοποιήσεων, δημιουργήθηκε IoT δίκτυο για την πραγματοποίηση βασικών δοκιμών λειτουργίας σε πραγματικές συνθήκες.

Οι παραπάνω δοκιμές αναδεικνύουν την επιτυχή λειτουργία της βιβλιοθήκης, για την ανάπτυξη ενός IoT δικτύου, στην πράξη. Οι δυνατότητες που παρέχει η βιβλιοθήκη ανταποκρίνονται στις ανάγκες που μελετήθηκαν και τις απαιτήσεις που τέθηκαν στα προηγούμενα κεφάλαια της εργασίας, με βάση τις σύγχρονες προκλήσεις που αντιμετωπίζει το Internet of Things και στόχο την τεχνολογικά και κοινωνικά βιώσιμη ανάπτυξή του.

5.2 Ενδεικτικές βελτιώσεις - επεκτάσεις

Όπως σε κάθε έργο, είτε αυτό αποτελεί μελέτη, είτε υλοποίηση, έτσι και σε αυτό υπάρχουν περιθώρια βελτιώσεων και επεκτάσεων. Αυτό ισχύει καθώς υφίστανται όρια τόσο σε επίπεδο όγκου και πεδίου εστίασης, όσο και σε επίπεδο χρόνου. Παρακάτω αναφέρονται μερικές ενδεικτικές επεκτάσεις. Κάποιες από αυτές θα μπορούσαν να περιληφθούν σε μελλοντικές εργασίες.

- Βελτιστοποίηση του κώδικα της βιβλιοθήκης που αναπτύχθηκε. Μία τέτοια βελτιστοποίηση μπορεί εστιάσει τόσο σε καλύτερη απόδοση, όσο και σε μεγαλύτερα επίπεδα ασφάλειας. Επιπρόσθετα, υπάρχουν περιθώρια για μεγαλύτερη οργάνωση του κώδικα και ανάπτυξη νέων δυνατοτήτων.
- Πραγματοποίηση δοκιμών scale up (κλιμάκωσης). Αυτές οι δοκιμές αφορούν τη δοκιμαστική χρήση της βιβλιοθήκης για την ανάπτυξη IoT δικτύου με όλο και περισσότερους κόμβους και μέτρηση της επίδοσής του σε συνάρτηση με τον αριθμό των Things, Clients και δωματίων. Η επίτευξη καλού αποτελέσματος σε τέτοιες δοκιμές καθιστά δυνατή τη χρήση της βιβλιοθήκης και γενικότερα της προσέγγισης στον πραγματικό κόσμο.
- Ενσωμάτωση της βιβλιοθήκης για τη δημιουργία πραγματικών Things και Clients. Η βιβλιοθήκη μπορεί να χρησιμοποιηθεί για τη δημιουργία IoT εφαρμογών οποιουδήποτε σεναρίου. Η ενσωμάτωση μπορεί να πραγματοποιηθεί σε συσκευές οι οποίες διαθέτουν επεξεργαστή που μπορεί να τρέξει Java και διαθέτουν σύνδεση στο διαδίκτυο.
- Ιεραρχική επέκταση της έννοιας του Provisioning Server. Στην προσέγγιση που μελετήθηκε στην τρέχουσα εργασία θεωρήθηκε πως υπάρχει ένας Provisioning Server στο IoT δίκτυο. Ωστόσο, το δίκτυο θα μπορούσε να έχει ιεραρχική/δεντρική μορφή. Κόμβος της τοπολογίας αυτής θα μπορούσε να αποτελεί είτε κάποιο Thing, είτε κάποιο υπο-δίκτυο με δικό του Provisioning Server. Ο Provisioning Server κάθε υπο-δικτύου εκτός από τη διαχείριση του ελέγχου πρόσβασης μέσα στο υπο-δίκτυο, θα μπορούσε να αποτελέσει πύλη σύνδεσης του υπο-δικτύου με IoT δίκτυα ανώτερου επιπέδου. Συνεπώς, κάθε υπο-δίκτυο αποτελεί πλέον ένα Thing για το ανώτερο στρώμα δικτύου και ο Provisioning Server του υπο-δικτύου έχει τον έλεγχο των δεδομένων που λαμβάνονται και προωθούνται προς το ανώτερο στρώμα.

Βιβλιογραφία

[1] Global and Regional ICT Data for the world by region (2005 - 2017). International Telecommunication Union (ITU). Retrieved from http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2017/ITU_Key_2005-2017_ICT_data.xls

[2] When old technologies were new: Implementing the future. Marvin, C. (1999). In H. Mackay & T. O'Sullivan (Eds.), The media reader: Continuity and transformation (pp. 58-72). New York, NY: Sage Publications. Retrieved from http://repository.upenn.edu/asc_papers/197

[3] 100 Years of Telephone Switching, Part 2: Electronics, Computer and Telephone Switching. Robert J. Chapuis, Amos E. Joel, Jr (2003). Ios Press. Retrieved from https://books.google.gr/books?id=07NmhqkOqwsC&lpg=PR4&ots=O0ys7NIAQG&dq=digitization%20of%20telephony%20and%20transistor%20history&lr&hl=el&pg=PR4#v=onepage&q&f=false

[4] Brief History of the Internet. Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff (1997). Retrieved from https://www.internetsociety.org/wp-content/uploads/2017/09/ISOC-History-of-the-Internet_1997.pdf

[5] Encyclopaedia Britannica:ARPANET. Kevin Featherly. Retrieved from https://www.britannica.com/topic/ARPANET

[6] The Home of The Future : Year 1999 AD (video). Philco - Ford Corporation (1967). Retrieved from https://www.youtube.com/watch?v=HI0to23KbKs

[7] The "Only" Coke Machine on the Internet. Computer Science Department - Carnegie Mellon University (1998). Retrieved from https://www.cs.cmu.edu/~coke/history_long.txt

[8] Ubiquitous Computing. Mark Weiser (1996). Retrieved from http://www.ubiq.com/hypertext/weiser/UbiHome.html

[9] The Computer for the 21st Century. Scientific American Special Issue on Communications, Computers, and Networks. Mark Weiser (1991). Retrieved from https://www.lri.fr/~mbl/Stanford/CS477/papers/Weiser-SciAm.pdf

[10] Google Trends: internet of things. Retrieved from https://trends.google.com/trends/explore?date=all&q=%22internet%20of%20things%22

[11] When the Internet of Things Starts to Feel Like the Internet of Shit. Motherboard. Vice Media Inc. Franceschi-Bicchierai, Lorenzo (2015). Retrieved from https://motherboard.vice.com/en_us/article/pgkdm7/when-the-internet-of-things-starts-to-feel-like-the-internet-of-shit

[12] IoT policy (Internet of Things policy). Margaret Rouse. Retrieved from http://internetofthingsagenda.techtarget.com/definition/IoT-policy-Internet-of-Things-policy

[13] Wikipedia.org: LPWAN. Retrieved from https://en.wikipedia.org/wiki/LPWAN

[14] Lora-Alliance.org: Member List. Retrieved from https://www.lora-alliance.org/The-Alliance/Member-List

[15] Wikipedia.org: SigFox. Retrieved from https://en.wikipedia.org/wiki/Sigfox

[16] Wikipedia.org: 3GPP. Retrieved from https://en.wikipedia.org/wiki/3GPP

[17] Retrieved from https://iot-for-all.com/cellular-iot-explained-nb-iot-vs-lte-m/

[18] Wikipedia.org: 5G. Retrieved from https://en.wikipedia.org/wiki/5G

[19] Wikipedia.org: ZigBee. Retrieved from https://en.wikipedia.org/wiki/Zigbee

[20] ZigBee.org: Our Members. Retrieved from http://www.zigbee.org/zigbeealliance/our-members/
[21] Wikipedia.org: Z-Wave. Retrieved from https://en.wikipedia.org/wiki/Z-Wave
[22] Z-WaveAlliance.org: Alliance Member Companies. Retrieved from http://z-wavealliance.org/z-wave_alliance_member_companies/
[23] RFC2460: Internet Protocol, Version 6 (IPv6), Specification. Internet Engineering Task Force (IETF). S. Deering, R. Hinden (1998). Retrieved from https://tools.ietf.org/html/rfc2460
[24] IpSO-Alliance.org: About Us. Retrieved from http://www.ipso-alliance.org/about-us/
[25] RFC7252: The Constrained Application Protocol (CoAP). Internet Engineering Task Force (IETF). Z. Shelby, K. Hartke, C. Bormann (2014). Retrieved from https://tools.ietf.org/search/rfc7252
[26] Wikipedia.org: XMPP. Retrieved from https://en.wikipedia.org/wiki/XMPP
[27] XMPP.org: XEP-0001: XMPP Extension Protocols. Retrieved from https://xmpp.org/extensions/xep-0001.html#states
[28] XMPP.org: XEP-0060: Publish-Subscribe. Retrieved from https://xmpp.org/extensions/xep-0060.html
[29] XMPP.org: XEP-0045: Multi-User Chat. Retrieved from https://xmpp.org/extensions/xep-0045.html
[30] XMPP.org: XEP-0323: Internet of Things - Sensor Data. Retrieved from https://xmpp.org/extensions/xep-0323.html
[31] XMPP.org: XEP-0325: Internet of Things - Control. Retrieved from https://xmpp.org/extensions/xep-0325.html
[32] XMPP.org: XEP-0326: Internet of Things - Concentrators. Retrieved from https://xmpp.org/extensions/xep-0326.html
[33] XMPP.org: XEP-0324: Internet of Things - Provisioning. Retrieved from https://xmpp.org/extensions/xep-0324.html
[34] XMPP.org: XEP-0347: Internet of Things - Discovery. Retrieved from https://xmpp.org/extensions/xep-0347.html
[35] XMPP.org: XMPP Servers. Retrieved from https://xmpp.org/software/servers.html
[36] Ignite Realtime - Openfire. Retrieved from https://www.igniterealtime.org/projects/openfire/index.jsp
[37] Ignite Realtime - Smack. Retrieved from https://www.igniterealtime.org/projects/smack/

Παράρτημα Α – Πηγές Εικόνων

[1] Global and Regional ICT Data for the world by region (2005 - 2017). International Telecommunication Union (ITU). http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2017/ITU_Key_2005-2017_ICT_data.xls

[2] A stylized replica of the first transistor (to commemorate the 50th anniversary of the invention of the point-contact transistor at Bell Labs in December 1947). Lucent Technologies (1997). https://upload.wikimedia.org/wikipedia/commons/b/bf/Replica-of-first-transistor.jpg

[3] ARPA Network, Logical Map, May 1973. US Army (1973)

[4] The Home of The Future : Year 1999 AD. Philco - Ford Corporation (1967). Retrieved from https://www.youtube.com/watch?v=HI0to23KbKs

[5] Internet Coke Machine. Computer Science Department - Carnegie Mellon University (1998)

[6] Writing/Saving XML Files (XML File Example) - Retrieved from https://www.kirupa.com

Παράρτημα Β – Πηγαίος Κώδικας

Provisioning Server Administrator Panel (Ιστοσελίδα)

controllers/Login.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Login extends CI_Controller
{

    public function __construct()
    {
        parent::__construct();
        $this->load->library('session');
        $this->load->helper('url');
        $this->load->database();


        if (!$this->session->has_userdata('user_status')) {
            $this->session->user_status = 'not_logged_in';
        }

        if ($this->session->user_status == 'logged_in')
            redirect(site_url('rooms'));
    }


    public function index()
    {
        $this->load->view('login');
    }

    public function do_login()
    {
        $usr  = $this->input->post('username');
        $pass = $this->input->post('password');

        $usrs = array();
        $usrs = $this->db->where(array(
            'username' => $usr,
            'password' => md5($pass)
        ))->get('admins')->result_array();

        if (count($usrs) > 0) {
            $this->session->user_status = 'logged_in';
            redirect(site_url(''));
        } else {
            $this->session->set_flashdata('login_error', 'yes');
            redirect(site_url('login'));
        }
    }
}
```

controllers/Main.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Main extends CI_Controller
{

    public $room_server_name = '@conference.169.254.44.207';
    public $server_name = '@169.254.44.207';

    public function __construct()
    {
        parent::__construct();
        $this->load->library('session');
        $this->load->helper('url');
        $this->load->database();


        if (!$this->session->has_userdata('user_status'))
        {
            $this->session->user_status = 'not_logged_in';
        }

        if ($this->session->user_status != 'logged_in')
            redirect(site_url('login'));
    }


    public function index()
    {
        $this->rooms();
    }

    public function rooms()
    {
        $data['rooms']   = $this->db-
>select('rooms.id, rooms.jid, COUNT(client_to_room.client_id) AS client
s, COUNT(thing_to_room.thing_id) AS things')-
>join('client_to_room', 'client_to_room.room_id = rooms.id', 'left')-
>join('thing_to_room', 'thing_to_room.room_id = rooms.id', 'left')-
>group_by('rooms.id')->get('rooms')->result_array();
        $data['content'] = $this->load->view('rooms', $data, TRUE);
        $data['page']    = 'rooms';
        $this->load->view('main', $data);
    }

    public function add_room()
    {
        $name = $this->input->post('name');
        $sn   = $this->room_server_name;
        $jid  = $name . $sn;

        $insert = array(
            'jid' => $jid
```

```
        );
        $this->db->insert('rooms', $insert);
        redirect('rooms');
    }

    public function things()
    {
        $data['things']  = $this->db-
>select('things.id, things.jid, IFNULL(thing_to_room.room_id, \'no\') A
S room_id, IFNULL(rooms.jid, \'no\') AS room')-
>join('thing_to_room', 'thing_to_room.thing_id = things.id', 'left')-
>join('rooms', 'thing_to_room.room_id = rooms.id', 'left')-
>get('things')->result_array();
        $data['rooms']   = $this->db->get('rooms')->result_array();
        $data['content'] = $this->load->view('things', $data, TRUE);
        $data['page']    = 'things';
        $this->load->view('main', $data);
    }

    public function add_thing()
    {
        $name = $this->input->post('name');
        $sn   = $this->server_name;
        $jid  = $name . $sn . '/' . $name . '_thing';

        $insert = array(
            'jid' => $jid
        );
        $this->db->insert('things', $insert);
        redirect('things');
    }

    public function delete_thing($id)
    {
        $where = array(
            'id' => $id
        );
        $this->db->where($where)->delete('things');
        redirect('things');
    }

    public function add_thing_to_room($thing_id)
    {
        $room_jid = $this->input->post('room');

        $room = $this->db->where('jid', $room_jid)->get('rooms')-
>row_array();
        if ($room !== NULL)
        {
            $insert = array(
                'room_id' => $room['id'],
                'thing_id' => $thing_id
```

```
            );
            $this->db->insert('thing_to_room', $insert);
        }

        redirect('things');
    }

    public function remove_thing_from_room($id, $page)
    {
        $where = array(
            'thing_id' => $id
        );
        $this->db->where($where)->delete('thing_to_room');
        $page = str_ireplace('_', '/', $page);
        redirect(site_url($page));
    }

    public function clients()
    {
        $data['clients'] = $this->db->get('clients')->result_array();
        $data['content'] = $this->load->view('clients', $data, TRUE);
        $data['page']    = 'clients';
        $this->load->view('main', $data);
    }

    public function add_client()
    {
        $name = $this->input->post('name');
        $sn   = $this->server_name;
        $jid  = $name . $sn . '/' . $name . '_client';

        $insert = array(
            'jid' => $jid
        );
        $this->db->insert('clients', $insert);
        redirect('clients');
    }

    public function delete_client($id)
    {
        $where = array(
            'id' => $id
        );
        $this->db->where($where)->delete('clients');
        redirect('clients');
    }

    public function room($id, $tab = 'things')
    {
        $data['room']    = $this->db->where('id', $id)->get('rooms')-
>row_array();
        $data['clients'] = $this->db-
>select('clients.*, client_to_room.voice')-
```

```
from('clients, client_to_room')-
>where('client_to_room.room_id = ' . $id . ' AND client_to_room.client_
id = clients.id')->get()->result_array();
        $data['things']  = $this->db->select('things.*')-
>from('things, thing_to_room')-
>where('thing_to_room.room_id = ' . $id . ' AND thing_to_room.thing_id
= things.id')->get()->result_array();

        $data['all_clients'] = $this->db-
>where('clients.id NOT IN (SELECT client_to_room.client_id FROM client_
to_room WHERE client_to_room.room_id = ' . $id . ')')->get('clients')-
>result_array();

        $data['tab']     = $tab;
        $data['content'] = $this->load->view('room', $data, TRUE);
        $data['page']    = 'rooms';
        $this->load->view('main', $data);
    }

    public function client_voice($room_id, $client_id, $action)
    {
        $this->db->where(array(
            'room_id' => $room_id,
            'client_id' => $client_id
        ))->set('voice', $action)->update('client_to_room');
        redirect(site_url('room/' . $room_id . '/clients'));
    }

    public function add_client_to_room($room_id)
    {
        $client_jid = $this->input->post('client');

        $client = $this->db->where('jid', $client_jid)->get('clients')-
>row_array();
        if ($client !== NULL)
        {
            $insert = array(
                'room_id' => $room_id,
                'client_id' => $client['id'],
                'voice' => 'no'
            );
            $this->db->insert('client_to_room', $insert);
        }

        redirect('room/' . $room_id . '/clients');
    }

    public function remove_client_from_room($room_id, $client_id)
    {
        $where = array(
            'client_id' => $client_id,
            'room_id' => $room_id
        );
```

```
            $this->db->where($where)->delete('client_to_room');
            redirect('room/' . $room_id . '/clients');
        }

        public function logout()
        {
            $this->session->user_status = 'not_logged_in';
            redirect(site_url(''));
        }
}
```

views/login.php

```
<html>
  <head>
    <title></title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
      <link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-
ui.css">
        <link rel="stylesheet" href="
          <?=base_url('assets/css/simplex.min.css'); ?>">
          <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
            <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
            <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-
ui.min.js"></script>
            <!-- Latest compiled and minified CSS -->
            <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>
            <style>
      body{
        background: url('
              <?=base_url('assets/img/back_1.png'); ?>') fixed center no-repeat;
        background-size: cover;
      }

            </style>
          </head>
          <body>
            <div class="container-fluid">
              <div style="position:fixed; width:100%; height:100%; z-index:5; top:0;
left:0; background-color: rgba(150, 150, 150, 0.55);"></div>
              <div class="col-sm-8 col-sm-offset-2" style="z-index:10;">
                <div class="panel panel-default text-center" style="margin-top:15%;
background-color: rgba(50, 50, 50, 0.8); color:#eee;">
                  <img src="
                    <?=base_url('assets/img/xmpp_logo.png'); ?>" style="max-width:10%;">
                    <h3 style="color:#eee;">Iot Network
          </h3>
                    <h4 style="color:#eee;">Provisioning Server Administrator
          </h4>
                    <form class="form-horizontal" method="POST" action="
                      <?=site url('login/do login'); ?>">
                      <div class="form-group">
                        <label class="col-sm-2 control-label">Username
              </label>
                          <div class="col-sm-8">
                            <input type="text" class="form-control" placeholder="Username"
name="username">
                          </div>
                        </div>
                        <div class="form-group">
                          <label class="col-sm-2 control-label">Password
              </label>
                          <div class="col-sm-8">
                            <input type="password" class="form-control"
placeholder="Password" name="password">
                          </div>
                        </div>
                        <?php if ($this->session->flashdata('login_error') !== NULL) {
?>
                        <p class="text-danger">Wrong username or password.
            </p>
                        <?php
} ?>
                        <div class="form-group">
                          <div class="col-sm-offset-2 col-sm-8">
```

```
                    <button type="submit" class="btn btn-info">Sign in
    </button>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>
    </body>
</html>
```

views/main.php

```
<html>
        <head>
                <title></title>

                <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">

                <link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/j
query-ui.css">

                <link rel="stylesheet"
href="<?=base_url('assets/css/simplex.min.css');?>">
                <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css">

                <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script
>
                <script
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-
ui.min.js"></script>
                <!-- Latest compiled and minified CSS -->
                <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>

                <style>
                        .room-element{
                                transition: all 0.3s ease;
                        }

                        .room-element:hover{
                                text-decoration:none;
                                font-size: 120%;
                                color: #f3f3f5 !important;
                                background-color:808080 !important;
                        }

                        .room-element:hover h4{
                                color: #f3f3f5 !important;
                        }

                        .room-element a, .room-element a:visited, .room-
element a:active{
                                color:unset;
                                text-decoration:none;
                        }

                        .ui-menu.ui-autocomplete{
                                /*display:none !important;*/
                        }

                </style>
        </head>

        <body style="background-color:#efefee;">
```

```
                <nav class="navbar navbar-default">
                        <div class="container-fluid">
                                <div class="navbar-header">
                                        <a class="navbar-brand" href="#"
style="padding:0; margin-left:10%;">
                                                <img alt="Brand"
src="<?=base_url('assets/img/xmpp_logo.png');?>" style="max-height:95%; margin-
top:5%;">
                                        </a>
                                        <h4 style="margin-left:100px;">Iot
Network - Provisioning Server Administrator</h4>
                                </div>
                                <ul class="nav navbar-nav navbar-right">
                                        <li><a
href="<?=site_url('main/logout');?>">Logout</a></li>
                                </ul>
                        </div>
                </nav>

                <div class="container-fluid">

                        <div class="col-sm-2">
                                <div class="list-group table-of-contents">
                                        <a class="list-group-item
<?=($page=='rooms') ? 'active' : '';?>" href="<?=site_url('rooms');?>">
                                                <i class="fa fa-folder-open
fa-lg pull-right" aria-hidden="true"></i>
                                                Thing Rooms
                                        </a>
                                        <a class="list-group-item
<?=($page=='things') ? 'active' : '';?>" href="<?=site_url('things');?>">
                                                <i class="fa fa-microchip fa-
lg pull-right" aria-hidden="true"></i>
                                                Things
                                        </a>
                                        <a class="list-group-item
<?=($page=='clients') ? 'active' : '';?>" href="<?=site_url('clients');?>">
                                                <i class="fa fa-users fa-lg
pull-right" aria-hidden="true"></i>
                                                Clients
                                        </a>
                                        <a class="list-group-item
<?=($page=='about') ? 'active' : '';?>"
href="<?=site_url('about');?>">About</a>
                                </div>
                        </div>

                        <div class="col-sm-10">

                                <?=$content;?>

                        </div>

                        </div>
                </div>

        </body>

</html>
```

views/rooms.php

```
<div class="input-group">
                                                <span class="input-group-addon">
                                                        <i class="fa fa-search fa-lg"
aria-hidden="true"></i>
                                                </span>
                                                <input type="text" class="form-control"
placeholder="Search Room" id="room-search">
                                        </div>
                                        <br/> <br/>

                                        <div class="panel panel-default">
                                                <div class="container-fluid">
                                                        <div class="row"
style="height:65%; overflow-y:auto;" id="room-list">


                                                        </div>
                                                </div>

                                        </div>

                                        <form method="POST"
action="<?=site_url('main/add_room');?>">
                                                <div class="input-group">
                                                        <input type="text" class="form-
control" placeholder="Thing Room's Name" name="name" required>
                                                        <span class="input-group-
addon"><?=$this->room_server_name;?></span>
                                                        <div class="input-group-btn">
                                                                <button class="btn btn-
primary">Add New Thing Room</button>
                                                        </div>
                                                </div>
                                        </form>

                                        <script>
                                                var rooms = [];

                                                var room_server_name = '<?=$this-
>room_server_name;?>';

                                                <?php foreach ($rooms as $room) { ?>

        add_room_to_list(<?=$room['id'];?>, '<?=$room['jid'];?>', <?=$room['things'];?>,
<?=$room['clients'];?>);
                                                <?php } ?>

                                                function add_room_to_list(id, jid, things,
clients){
                                                        rooms.push({
                                                                'id': id,
                                                                'jid': jid,
                                                                'things': things,
                                                                'clients': clients
                                                        });
                                                }

                                                function room_display(){
                                                        var content = '';
                                                        var auto_list = [];
                                                        var bind_list = [];
                                                        for (var i=0; i<rooms.length;
i++){
                                                                content += '<div data-
id="'+rooms[i]['id']+'" class="col-sm-3 text-center room-element" style="min-
height:140px; border:1px solid #aab; border-radius:0%; background-color:#f3f3f5;">';
                                                                content += '       <a
href="<?=site_url('room/');?>'+rooms[i]['id']+'">';
                                                                content +=
'                  <i class="fa fa-folder-open fa-3x" aria-hidden="true"></i>';
```

```
                                        });
                                }

                                function room_display(){
                                        var content = '';
                                        var auto_list = [];
                                        var bind_list = [];
                                        for (var i=0; i<rooms.length;
i++){
                                                content += '<div data-
id="'+rooms[i]['id']+'" class="col-sm-3 text-center room-element" style="min-
height:140px; border:1px solid #aab; border-radius:0%; background-color:#f3f3f5;">';
                                                content += '       <a
href="<?=site_url('room/');?>'+rooms[i]['id']+'">';
                                                content +=
'                   <i class="fa fa-folder-open fa-3x" aria-hidden="true"></i>';
                                                content +=
'                   <h4 style="word-wrap:break-word;">'+rooms[i]['jid']+'</h4>';
                                                content +=
'                   <span class="text-muted">';
                                                content +=
'                           '+rooms[i]['things']+' <i class="fa fa-microchip fa-lg" aria-
hidden="true"></i>';
                                                content +=
'                           |';
                                                content +=
'                           '+rooms[i]['clients']+' <i class="fa fa-users fa-lg" aria-
hidden="true"></i>';
                                                content +=
'                   </span>';
                                                content += '
          </a>';
                                                content += '</div>';


          auto_list.push(rooms[i]['jid']);

          bind_list[rooms[i]['jid']] = rooms[i]['id'];
                                        }
                                        $('#room-list').html(content);

                                        $('#room-search').autocomplete({
                                                source:auto_list,
                                                minLength:0,


          response:function(event, ui){
                                                        var arr =
ui['content'];
                                                        if
(arr.length>0){
          $('.room-element').removeClass('toshow').addClass('tohide');

          //alert(JSON.stringify(arr));
                                                                for
(var i=0; i<arr.length; i++){

          $('.room-element[data-
id='+bind_list[arr[i]['label']]+']').removeClass('tohide').addClass('toshow');
                                                                }
                                                        }
                                                        else
$('.room-element').removeClass('tohide').addClass('toshow');

                                                        $('.room-
element.tohide').fadeOut(400);
                                                        $('.room-
element.toshow').fadeIn(400);
                                                },
                                                open: function( event,
ui ) {
```

```
            $('.room-element').removeClass('toshow').addClass('tohide');

            //alert(JSON.stringify(arr));
                                                                                    for
(var i=0; i<arr.length; i++){

            $('.room-element[data-
id='+bind_list[arr[i]['label']]+']').removeClass('tohide').addClass('toshow');
                                                                                    }
                                                                                }
                                                                                else
$('.room-element').removeClass('tohide').addClass('toshow');

                                                                                $('.room-
element.tohide').fadeOut(400);
                                                                                $('.room-
element.toshow').fadeIn(400);
                                                                            },
                                                                            open: function( event,
ui ) {
                                                                                $(".ui-
autocomplete").hide();
                                                                            }
                                                                        });
                                                                }

                                                                $(document).ready(room_display);
                                                            </script>
```

views/room.php

```
<div class="text-center">
        <h4><?=$room['jid'];?></h4>
        <a href="<?=site_url('rooms');?>">Back to rooms</a>
</div>

<ul class="nav nav-tabs">
        <li role="presentation" class="<?=($tab== 'things') ? 'active' : '' ;?>"><a
role="tab" data-toggle="tab" href="#things">
                <i class="fa fa-microchip fa-lg" aria-hidden="true"></i> Things
        </a></li>

        <li role="presentation" class="<?=($tab== 'clients') ? 'active' : '' ;?>"><a
role="tab" data-toggle="tab" href="#clients">
                <i class="fa fa-users fa-lg" aria-hidden="true"></i> Clients
        </a></li>
</ul>
<br/>

<div class="tab-content">

  <div role="tabpanel" class="tab-pane fade <?=($tab== 'things') ? 'in active' : '' ;?>"
id="things">

        <div class="input-group">
                                        <span class="input-group-addon">
                                                <i class="fa fa-search fa-lg"
aria-hidden="true"></i>
                                        </span>
                                        <input type="text" class="form-control"
placeholder="Search Thing" id="thing-search">
                                </div>
                                <br/>

                                <div class="panel panel-default">
                                        <div class="container-fluid">
                                                <div class="row"
style="height:60%; overflow-y:auto;">
                                                        <ul class="list-group">
                                                                <?php foreach
($things as $thing) { ?>
                                                                        <li
class="list-group-item thing" data-id="<?=$thing['jid'];?>">

        <div class="container-fluid">

                <div class="row">


                        <div class="col-sm-5 col-xs-12">

                                <i class="fa fa-microchip fa-lg" aria-
hidden="true"></i>

                                <em><?=$thing['jid'];?></em>

                        </div>


                        <div class="col-sm-5 col-xs-12">


                        </div>


                        <div class="col-sm-2 col-xs-12">

                                <a
```

```
href="<?=site_url('main/remove_thing_from_room/'.$thing['id'].'/room_'.$room['id']);?>"
class="pull-right" onclick="return confirm('Are you sure you want to remove this thing
from this room?');">

                                                    <span class="text-danger">

                                                            <i class="fa fa-times fa-lg"
aria-hidden="true"></i>

                                                    </span>

                                            </a>

                                    </div>




                            </div>

                    </div>

                    </li>

                                                                                    <?php } ?>
                                                                            </ul>

                                                            </div>
                                                    </div>

                                            </div>

                                            <script>

                                                    function init(){

                                                            var things = [];
                                                            $('li.thing').each(function(){

                    things.push($(this).attr('data-id'));
                                                            });

                                                            //alert(JSON.stringify(things));

                                                            $('#thing-
search').autocomplete({
                                                                    source:things,
                                                                    minLength:0,

                    response:function(event, ui){
                                                                            var arr =
ui['content'];
                                                                            if
(arr.length>0){

                    $('li.thing').removeClass('toshow').addClass('tohide');

                    //alert(JSON.stringify(arr));
                                                                                    for
(var i=0; i<arr.length; i++){

                    $('li.thing[data-
id="'+arr[i]['label']+'"]').removeClass('tohide').addClass('toshow');
                                                                                    }
                                                                            }
                                                                            else
$('li.thing').removeClass('tohide').addClass('toshow');


                    $('li.thing.tohide').fadeOut(400);
```

```
        $('li.thing.toshow').fadeIn(400);
                                                                },
                                                                open: function( event,
ui ) {
                                                                        $(".ui-
autocomplete").hide();
                                                                }
                                                        });
                                                }

                                                $(document).ready(init);

                                        </script>


  </div>

  <div role="tabpanel" class="tab-pane fade <?=($tab== 'clients') ? 'in active' : '' ;?>"
id="clients">

        <div class="input-group">
                                                <span class="input-group-addon">
                                                        <i class="fa fa-search fa-lg"
aria-hidden="true"></i>
                                                </span>
                                                <input type="text" class="form-control"
placeholder="Search Client" id="client-search">
                                        </div>
                                        <br/>

                                        <div class="panel panel-default">
                                                <div class="container-fluid">
                                                        <div class="row"
style="height:55%; overflow-y:auto;">
                                                                <ul class="list-group">
                                                                        <?php foreach
($clients as $client) { ?>
                                                                                <li
class="list-group-item client" data-id="<?=$client['jid'];?>">

        <div class="container-fluid">

                <div class="row">


                        <div class="col-sm-5 col-xs-12">

                                <i class="fa fa-user fa-lg" aria-hidden="true"></i>

                                <em><?=$client['jid'];?></em>

                        </div>


                        <div class="col-sm-5 col-xs-12">

                                <?php

                                        $caption = 'Give control permission';

                                        $icon = 'square-o';

                                        $action = 'yes';


                                        if ($client['voice']=='yes') {
```

```
                                                    $caption = 'Revoke control
permission';
                                                    $icon = 'check-square-o';
                                                    $action = 'no';
                                                }
                                            ?>
                                            <a
href="<?=site_url('main/client_voice/'.$room['id'].'/'.$client['id'].'/'.$action);?>">
                                                <i class="fa fa-<?=$icon;?> fa-lg" aria-
hidden="true"></i>
                                                <?=$caption;?>
                                            </a>
                                        </div>

                                        <div class="col-sm-2 col-xs-12">
                                            <a
href="<?=site_url('main/remove_client_from_room/'.$room['id'].'/'.$client['id']);?>"
class="pull-right" onclick="return confirm('Are you sure you want to remove this client
from this room?');">
                                                <span class="text-danger">
                                                    <i class="fa fa-times fa-lg"
aria-hidden="true"></i>
                                                </span>
                                            </a>
                                        </div>


                    </div>
                </div>
                </li>
                                                                <?php } ?>
                                                        </ul>
                                                    </div>
                                                </div>

                                            </div>
                                            <form method="POST"
action="<?=site_url('main/add_client_to_room/'.$room['id']);?>"/>
                                                <div class="input-group">
                                                    <input type="text" class="form-
control client-list" placeholder="Add client" name="client" required>
                                                    <div class="input-group-btn">
                                                        <button class="btn btn-
primary" onclick="return function(){$('form').submit();};">
                                                            Add client
                                                        </button>
```

```
                                        </div>
                                    </div>
                                </form>

                                <script>

                                    var all_clients = [];

                                    var room_server_name = '<?=$this->room_server_name;?>';

                                    <?php foreach ($all_clients as $client) { ?>

        add_client_to_list(<?=$client['id'];?>, '<?=$client['jid'];?>');
                                    <?php } ?>

                                    function add_client_to_list(id, jid){
                                        all_clients.push(jid);
                                    }

                                    function init_2(){

                                        $('.client-list').autocomplete({
                                            source:all_clients,
                                            position:{ my: "left bottom", at: "left top", collision: "none" }
                                        });

                                        var clients = [];
                                        $('li.client').each(function(){

        clients.push($(this).attr('data-id'));
                                        });

                                        //alert(JSON.stringify(things));

                                        $('#client-search').autocomplete({
                                            source:clients,
                                            minLength:0,

        response:function(event, ui){
                                                var arr = ui['content'];

                                                if (arr.length>0){

        $('li.client').removeClass('toshow').addClass('tohide');

        //alert(JSON.stringify(arr));
                                                    for (var i=0; i<arr.length; i++){

        $('li.client[data-id="'+arr[i]['label']+'"]').removeClass('tohide').addClass('toshow');
                                                    }
                                                }
                                                else $('li.client').removeClass('tohide').addClass('toshow');


        $('li.client.tohide').fadeOut(400);

        $('li.client.toshow').fadeIn(400);
                                            },
                                            open: function( event, ui ) {
                                                $(".ui-autocomplete").hide();
                                            }});}
```

```
$(document).ready(init_2);
                                        </script>


  </div>
</div>
```

views/things.php

```
<div class="input-group">
          <span class="input-group-addon">
                    <i class="fa fa-search fa-lg" aria-hidden="true"></i>
          </span>
          <input type="text" class="form-control" placeholder="Search Thing" id="thing-
search">
</div>
<br/> <br/>

<div class="panel panel-default">
          <div class="container-fluid">
                    <div class="row" style="height:65%; overflow-y:auto;">
                              <ul class="list-group">
                                        <?php foreach ($things as $thing) { ?>
                                                  <li class="list-group-item thing" data-
id="<?=$thing['jid'];?>">
                                                            <div class="container-fluid">
                                                                      <div class="row">

                                                                                <div
class="col-sm-5 col-xs-12">
                                                                                          <i
class="fa fa-microchip fa-lg" aria-hidden="true"></i>

          <em><?=$thing['jid'];?></em>
                                                                                </div>

                                                                                <div
class="col-sm-5 col-xs-12">

          <?php if ($thing['room']!='no') { ?>

          <i class="fa fa-folder-open fa-lg" aria-hidden="true"></i>

          <?=$thing['room'];?>

          <a href="<?=site_url('main/remove_thing_from_room/'.$thing['id'].'/things');?>"
class="pull-right">

                    <span class="text-warning">

                              <i class="fa fa-minus fa-lg" aria-hidden="true"></i>

                    </span>

          </a>

          <?php } ?>


          <?php if ($thing['room']=='no') { ?>

          <form method="POST"
action="<?=site_url('main/add_thing_to_room/'.$thing['id']);?>"/>

                    <div class="input-group">

                              <input type="text" class="form-control room-list"
placeholder="Add to room" name="room">

                              <span class="input-group-addon">

                                        <a href="#" class="pull-right add_to_room">

                                                  <span class="text-info">

                                                            <i class="fa fa-plus fa-lg"
aria-hidden="true"></i>

                                                  </span>
```

```
                                        </a>

                                </span>

                        </div>

                </form>

                <?php } ?>
                                                                                        </div>

                                                                                        <div
class="col-sm-2 col-xs-12">
                                                                                                <a
href="<?=site_url('main/delete_thing/'.$thing['id']);?>" class="pull-right"
onclick="return confirm('Are you sure you want to delete this thing from the list?');">

                <span class="text-danger">

                        <i class="fa fa-times fa-lg" aria-hidden="true"></i>

                </span>
                                                                                                </a>
                                                                                        </div>



                                                                        </div>
                                                                </div>
                                                        </li>
                                                <?php } ?>
                                </ul>

                        </div>
                </div>

</div>

<form method="POST" action="<?=site_url('main/add_thing');?>">
        <div class="input-group">
                <input type="text" class="form-control" placeholder="Thing's Name"
name="name" required>
                <span class="input-group-addon"><?=$this->server_name;?></span>
                <div class="input-group-btn">
                        <button class="btn btn-primary">Add Thing</button>
                </div>
        </div>
</form>

<script>
        var rooms = [];

        var room_server_name = '<?=$this->room_server_name;?>';

        <?php foreach ($rooms as $room) { ?>
                add_room_to_list(<?=$room['id'];?>, '<?=$room['jid'];?>');
        <?php } ?>

        function add_room_to_list(id, jid){
                rooms.push(jid);
        }

        function init(){
                $('.room-list').autocomplete({
                        source:rooms
                });

                var things = [];
                $('li.thing').each(function(){
                        things.push($(this).attr('data-id'));
```

```
                    });

                    //alert(JSON.stringify(things));

                    $('#thing-search').autocomplete({
                              source:things,
                              minLength:0,
                              response:function(event, ui){
                                        var arr = ui['content'];
                                        if (arr.length>0){

          $('li.thing').removeClass('toshow').addClass('tohide');
                                                  //alert(JSON.stringify(arr));
                                                  for (var i=0; i<arr.length; i++){
                                                            $('li.thing[data-
id="'+arr[i]['label']+'"]').removeClass('tohide').addClass('toshow');
                                                  }
                                        }
                                        else
$('li.thing').removeClass('tohide').addClass('toshow');

                                        $('li.thing.tohide').fadeOut(400);
                                        $('li.thing.toshow').fadeIn(400);
                              },
                              open: function( event, ui ) {
                                        $(".ui-autocomplete").hide();
                              }
                    });
          }

          $(document).ready(init);

          $('a.add_to_room').click(function(){
                    $(this).closest('form').submit();
          });
</script>
```

views/clients.php

```
<div class="input-group">
          <span class="input-group-addon">
                    <i class="fa fa-search fa-lg" aria-hidden="true"></i>
          </span>
          <input type="text" class="form-control" placeholder="Search Client" id="client-
search">
</div>
<br/> <br/>

<div class="panel panel-default">
          <div class="container-fluid">
                    <div class="row" style="height:65%; overflow-y:auto;">
                              <ul class="list-group">
                                        <?php foreach ($clients as $client) { ?>
                                                  <li class="list-group-item client" data-
id="<?=$client['jid'];?>">
                                                            <div class="container-fluid">
                                                                      <div class="row">

                                                                                <div
class="col-sm-5 col-xs-12">
                                                                                          <i
class="fa fa-user fa-lg" aria-hidden="true"></i>

          <em><?=$client['jid'];?></em>
                                                                                </div>

                                                                                <div
class="col-sm-5 col-xs-12">

                                                                                </div>

                                                                                <div
class="col-sm-2 col-xs-12">
                                                                                          <a
href="<?=site_url('main/delete_client/'.$client['id']);?>" class="pull-right" on-
click="return confirm('Are you sure you want to delete this client from the list?');">

          <span class="text-danger">

                    <i class="fa fa-times fa-lg" aria-hidden="true"></i>

          </span>
                                                                                          </a>
                                                                                </div>


                                                                      </div>
                                                            </div>
                                                  </li>
                                        <?php } ?>
                              </ul>

                    </div>
          </div>

</div>

<form method="POST" action="<?=site_url('main/add_client');?>">
          <div class="input-group">
                    <input type="text" class="form-control" placeholder="Client's Name"
name="name" required>
                    <span class="input-group-addon"><?=$this->server_name;?></span>
                    <div class="input-group-btn">
                              <button class="btn btn-primary">Add Client</button>
                    </div>
          </div>
</form>

<script>
```

```
<script>

	function init(){

		var clients = [];
		$('li.client').each(function(){
			clients.push($(this).attr('data-id'));
		});

		//alert(JSON.stringify(things));

		$('#client-search').autocomplete({
			source:clients,
			minLength:0,
			response:function(event, ui){
				var arr = ui['content'];
				if (arr.length>0){
					$('li.client').re-
moveClass('toshow').addClass('tohide');
					//alert(JSON.stringify(arr));
					for (var i=0; i<arr.length; i++){
						$('li.client[data-
id="'+arr[i]['label']+'"]').removeClass('tohide').addClass('toshow');
					}
				}
				else $('li.client').re-
moveClass('tohide').addClass('toshow');

				$('li.client.tohide').fadeOut(400);
				$('li.client.toshow').fadeIn(400);
			},
			open: function( event, ui ) {
				$(".ui-autocomplete").hide();
			}
		});
	}

	$(document).ready(init);

</script>
```

App.java

```
package sm01.sm01;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

import org.jivesoftware.smack.AbstractXMPPConnection;
import org.jivesoftware.smack.ConnectionConfiguration.SecurityMode;
import org.jivesoftware.smack.SmackConfiguration;
import org.jivesoftware.smack.provider.ProviderManager;
import org.jivesoftware.smack.tcp.XMPPTCPConnection;
import org.jivesoftware.smack.tcp.XMPPTCPConnectionConfiguration;
import org.jivesoftware.smackx.iot.provisioning.provider.IoTIsFriendProvider;

import sm01.Entities.AbstractEntity;
import sm01.Entities.ProvisioningServer;


public class App
{
        private static JButton thingBtn;
        private static JButton ctrlBtn;
        private static JTextField usrField;
        private static JPasswordField passField;
        public static void main( String[] args )
    {
                try {

        UIManager.setLookAndFeel(UIManager.getInstalledLookAndFeels()[3].getCla
ssName());
                } catch (Exception e) {
                        e.printStackTrace();
                }
                SmackConfiguration.DEBUG = true;
                StartProvisioningServer();
                JFrame window = new JFrame("XMPP IoT Network Testing");
                thingBtn = new JButton("Add as a Thing");
                ctrlBtn = new JButton("Add as a Client");
                thingBtn.addMouseListener(click);
                ctrlBtn.addMouseListener(click);

                usrField = new JTextField(20);
                passField = new JPasswordField(20);
                window.getContentPane().setLayout(new GridLayout(3, 2, 0, 10));

                window.add(new JLabel("Username :"));
                window.add(usrField);
                window.add(new JLabel("Password :"));
                window.add(passField);
                window.getContentPane().add(thingBtn);
                window.getContentPane().add(ctrlBtn);
        window.pack();
        window.setLocationByPlatform(true);
```

```
        window.setVisible(true);
    }

        private static void StartProvisioningServer() {
                String user = "provisioning", pass = "pass",
                            localhost = "169.254.44.207", domain =
"83.212.97.56";

        AbstractEntity.setProvisioningServer(user+"@"+localhost+"/"+"provisioni
ng");
                ProvisioningServer provServer =

        ProvisioningServer.createProvisioningServer(user, pass,
domain,"provisioning");
        }

        private static MouseListener click = new MouseListener(){
                public void mouseClicked(MouseEvent e) {
                        if (e.getComponent().equals(thingBtn)) new
TestThing2(usrField.getText(), passField.getText());
                        else if (e.getComponent().equals(ctrlBtn)) new
TestController2(usrField.getText(), passField.getText());
                }
                public void mouseEntered(MouseEvent e) {}
                public void mouseExited(MouseEvent e) {}
                public void mousePressed(MouseEvent e) {}
                public void mouseReleased(MouseEvent e) {}
        };
}
```

TestThing2.java

```
package sm01.sm01;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.util.List;
import java.util.Random;

import javax.swing.BoxLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;

import sm01.Entities.Thing;
import sm01.IoTElements.SetData;
import sm01.Listeners.ControlListener;
import sm01.Messages.DataFieldListBuilder;

public class TestThing2 {
	final static String host = "83.212.97.56";
	static String user = "user1";
	static String pass = "pass";
	static String nodeId = "myThing";
	private Thing thing;
	private JFrame window;
	private JLabel label1, data1, label2, data2, label3, data3;

	public TestThing2(final String username, String password){
		user = username; pass = password; nodeId = username+" thing";
		thing = Thing.createThing(user, pass, host, nodeId, nodeId);
		if (!thing.connectToRoom()) return;
		createWindow();
		thing.setControlListener(new ControlListener(){
			public void ControlDataReceived(List<SetData> data) {
				label1.setText(data.get(0).getName());
				data1.setText(data.get(0).getValue());
				label2.setText(data.get(1).getName());
				data2.setText(data.get(1).getValue());
				label3.setText(data.get(2).getName());
				data3.setText(data.get(2).getValue());
			}
		});
		(new Thread(new Runnable(){
			public void run() {
				while(true){
					try {

	Thread.currentThread().sleep(3000);
							thing.sendSensorData(

	DataFieldListBuilder.createDataFieldListBuilder()

	.addField("temperature", (new Random()).nextDouble()*30)

	.addField("rains", (new Random()).nextBoolean())

	.addField("msg", "hello from "+username)
	.getList()
	);

					} catch (InterruptedException e) {
						e.printStackTrace();
					}
				}
			}
		})).start();

		window.addWindowListener(new WindowListener(){

			public void windowActivated(WindowEvent arg0) {}
```

```
                        public void windowClosed(WindowEvent arg0) {}

                        public void windowClosing(WindowEvent arg0) {}

                        public void windowDeactivated(WindowEvent arg0) {}

                        public void windowDeiconified(WindowEvent arg0) {}

                        public void windowIconified(WindowEvent arg0) {}

                        public void windowOpened(WindowEvent arg0) {}

                });
        }

        private void createWindow() {
                window = new JFrame("Thing: "+nodeId);
                window.setLayout(new
BoxLayout(window.getContentPane(),BoxLayout.PAGE_AXIS));
                label1 = new JLabel("label1");
                window.add(label1);
                data1 = new JLabel("data1");
                window.add(data1);
                label2 = new JLabel("label2");
                window.add(label2);
                data2 = new JLabel("data2");
                window.add(data2);
                label3 = new JLabel("label3");
                window.add(label3);
                data3 = new JLabel("data3");
                window.add(data3);
                data1.setForeground(Color.red);
                data2.setForeground(Color.red);
                data3.setForeground(Color.red);
                window.setSize(300, 200);;
                window.setVisible(true);
        }

}
```

TestController2.java

```
package sm01.sm01;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Random;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFormattedTextField;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.border.BevelBorder;

import org.jivesoftware.smackx.iot.element.NodeInfo;
import org.jxmpp.jid.BareJid;
import org.jxmpp.jid.EntityBareJid;
import org.jxmpp.jid.EntityFullJid;
import org.jxmpp.jid.impl.JidCreate;
import org.jxmpp.stringprep.XmppStringprepException;

import sm01.Entities.Controller;
import sm01.Entities.Thing;
import sm01.IoTElements.IoTDataField;
import sm01.IoTElements.SetData;
import sm01.Listeners.ControlListener;
import sm01.Listeners.SensorDataListener;
import sm01.Listeners.ThingPresenceListener;
import sm01.Messages.DataFieldListBuilder;
import sm01.Messages.SetDataListBuilder;

public class TestController2 {
            final static String host = "83.212.97.56";
            //final static String localhost = "pikserver.tk";
            //final static String host = "83.212.97.56";
            final static String localhost = "169.254.44.207";
            static String user = "user2";
            static String pass = "pass";
            static String nodeId = "myController";
            final static String roomJid = "test_room@conference."+localhost;
            private Controller controller;
            private JFrame window;
            private JLabel sensorLabel;
            JTextField box1, box2; JFormattedTextField box3;

            JComboBox<String> thing_list, room_list;

            private Map<EntityBareJid,JPanel> room_panels;
            private Map<EntityBareJid,Map<String, JPanel>> thing_panels;

            private void add_room_panel(EntityBareJid room){
                        JPanel room_panel = new JPanel();
                        room_panel.setLayout(new BoxLayout(room_panel, BoxLayout.X_AXIS));
                        room_panel.setBorder(BorderFactory.createCompoundBorder(
                  BorderFactory.createTitledBorder(room.toString()),
                  BorderFactory.createEmptyBorder(5,5,5,5)));
```

```
                Map<String, JPanel> room_things = new HashMap<String, JPanel>();
                List<String> nodes = /*Arrays.asList("node_1", "node_2", "node_3");*/
                                controller.getOnlineThingsNodeIds(room);
                System.out.println(room.toString()+" :"+nodes.size());
                for (String node : nodes){
                        System.out.println("__"+node);
                        JPanel thing_panel = new JPanel();
                        thing_panel.setLayout(new BoxLayout(thing_panel, BoxLayout.X_AXIS));
                        //thing panel.setBorder(BorderFactory.createTitledBorder(node));


        thing_panel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(),
node));
                        thing_panel.setBackground(Color.white);
                        JLabel lab = new JLabel("sensor_data");
                        thing_panel.add(lab);

                        room_things.put(node, thing_panel);
                        room_panel.add(thing_panel);
                }

                //room panel.add(new JLabel("hello"));
                room_panel.setSize(500, 350);
                room_panels.put(room, room_panel);
                thing_panels.put(room, room_things);
                room_list.addItem(room.toString());
        }

        public TestController2(String username, String password){
                user = username; pass = password; nodeId = username+"_client";
                controller = Controller.createController(user, pass, host, nodeId);
                Iterator<EntityBareJid> all_rooms = controller.getAllRooms().iterator();

                room_panels = new HashMap<EntityBareJid,JPanel>();
                thing_panels = new HashMap<EntityBareJid,Map<String, JPanel>>();
                room_list = new JComboBox<String>();
                while (all_rooms.hasNext()){
                        EntityBareJid cur_room = all_rooms.next();
                        if (!controller.joinRoom(cur_room)) {
                                System.out.println(user+" couldn't connect to
"+cur room.toString());
                                //return;
                        }
                        else{
                                add room panel(cur room);
                        }
                }

                createWindow();
                window.addWindowListener(new WindowListener(){

                        public void windowActivated(WindowEvent arg0) {
                                // TODO Auto-generated method stub

                        }

                        public void windowClosed(WindowEvent arg0) {
                                // TODO Auto-generated method stub

                        }

                        public void windowClosing(WindowEvent arg0) {
                                controller.disconnect();
                        }

                        public void windowDeactivated(WindowEvent arg0) {
                                // TODO Auto-generated method stub

                        }

                        public void windowDeiconified(WindowEvent arg0) {
                                // TODO Auto-generated method stub

                        }

                        public void windowIconified(WindowEvent arg0) {
                                // TODO Auto-generated method stub

                        }
```

```
                                public void windowOpened(WindowEvent arg0) {
                                        // TODO Auto-generated method stub

                                }

                        });
                        controller.setSensorDataListener(new SensorDataListener(){
                                public void sensorDataReceived(BareJid roomJid, NodeInfo nodeInfo,
                                        Date timestamp, List<? extends IoTDataField> data) {
                                        String txt = "<html>";
                                        txt += "<i
style='color:#999'>"+timestamp.toString()+"</i>";
                                        txt += "<table>";
                                        for (IoTDataField field : data)
                                                txt += "<tr><th>"+field.getName()+ "</th> <td>"
+ field.getValueString() + "</td></tr>";
                                        txt += "</table>";
                                        txt += "</html>";
                                        try{
                                                JLabel l = (JLabel)
thing panels.get(roomJid).get(nodeInfo.getNodeId()).getComponent(0);
                                                l.setText(txt);
                                        }
                                        catch(Exception e){}

                                }
                        });

                        controller.setThingPresenceListener(new ThingPresenceListener(){
                                public void thingPresenceChanged(BareJid roomJid,
                                                List<EntityFullJid> onlineThings) {
                                        List<String> otn = new ArrayList<String>();
                                        for (EntityFullJid th : onlineThings){
                                                otn.add(th.getResourceOrEmpty().toString());
                                                if
(!thing_panels.get(roomJid).containsKey(th.getResourceOrEmpty().toString())){

            System.out.println("__"+th.toString());
                                                                JPanel thing_panel = new JPanel();
                                                                thing_panel.setLayout(new
BoxLayout(thing panel, BoxLayout.X AXIS));

            thing_panel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(),
th.getResourceOrEmpty().toString()));

            thing_panel.setBackground(Color.white);
                                                                thing_panel.add(new
JLabel("sensor_data"));
                                                                thing_panel.setSize(200, 200);


            thing_panels.get(roomJid).put(th.getResourceOrEmpty().toString(), thing_panel);

            room_panels.get(roomJid).add(thing_panel);

                                                }
                                        }



                                        Iterator<String> it =
thing_panels.get(roomJid).keySet().iterator();
                                        while (it.hasNext()){
                                                String th = it.next();
                                                if (!otn.contains(th)) {

            room_panels.get(roomJid).remove(thing_panels.get(roomJid).get(th));

                                                                it.remove();
                                                }
                                        }
                                        window.repaint();

                                }
                        });
            }

            private void createWindow() {
                        window = new JFrame("Client: "+nodeId);
```

```
                        window.setLayout(new BoxLayout(window.getContentPane(),BoxLayout.PAGE_AXIS));
                        window.add(new JLabel("Command1: "));
                        box1 = new JTextField(); box2 = new JTextField();
                        window.add(box1);
                        window.add(new JLabel("Command2: "));
                        window.add(box2);
                        box3 = new JFormattedTextField(NumberFormat.getNumberInstance());
                        window.add(new JLabel("Number1: "));
                        window.add(box3);

                        window.add(new JLabel("Room: "));

                        window.add(room_list);
                        room_list.addActionListener(new ActionListener(){
                                  public void actionPerformed(ActionEvent e) {
                                            JComboBox cb = (JComboBox)e.getSource();
                                  String room_jid = (String)cb.getSelectedItem();
                                  thing_list.removeAllItems();
                                  try {
                                                      for (String st :
thing_panels.get(JidCreate.entityBareFrom(room_jid)).keySet())
                                                                thing_list.addItem(st);

                                            } catch (XmppStringprepException e1) {}
                                  }
                        });

                        window.add(new JLabel("Thing: "));
                        thing_list = new JComboBox<String>();
                        window.add(thing_list);

                        JButton button = new JButton("Send");
                        button.addMouseListener(new MouseListener(){

                                  public void mouseClicked(MouseEvent arg0) {
                                            List<SetData> data =
SetDataListBuilder.createSetDataListBuilder()
                                            .addSetData("Command1", box1.getText())
                                            .addSetData("Command2", box2.getText())
                                            .addSetData("Number1", box3.getValue())
                                            .getList();
                                            try {
                                                      System.out.println("Control:
"+JidCreate.bareFrom((String)room_list.getSelectedItem()).toString()+"
"+(String)thing_list.getSelectedItem());

          controller.sendControlData(JidCreate.bareFrom((String)room_list.getSelectedItem()),
Collections.singletonList((String)thing_list.getSelectedItem()), data);
                                            } catch (XmppStringprepException e) {
                                                      // TODO Auto-generated catch block
                                                      e.printStackTrace();
                                            }
                                  }
                                  public void mouseEntered(MouseEvent arg0) {}
                                  public void mouseExited(MouseEvent arg0) {}
                                  public void mousePressed(MouseEvent arg0) {}
                                  public void mouseReleased(MouseEvent arg0) {}

                        });
                        window.add(button);

                        JPanel rooms = new JPanel();
                        rooms.setLayout(new BoxLayout(rooms, BoxLayout.Y_AXIS));
                        Iterator<JPanel> pit = room_panels.values().iterator();
                        while (pit.hasNext()) {
                                  JPanel cur_panel = pit.next();
                                  rooms.add(cur_panel);
                        }
                        window.add(rooms);
                        window.setSize(500, 500);
                        window.setVisible(true);
          }

}
```