ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Ψηφιακών Συστημάτων

ΠΜΣ "Ψηφιακά Συστήματα & Υπηρεσίες

Κατεύθυνση: Ψηφιακές Επικοινωνίες & Δίκτυα

«Μελέτη & ανάπτυξη συστήματος VOIP σε περιβάλλον cloud computing με δυνατότητες αυτοματοποιημένων διεργασιών Load balancing»

Καραγιαννάκης Βασίλειος
ΜΕ 1548

Επιβλέπων καθηγητής κ Απόστολος Μηλιώνης

ΠΕΙΡΑΙΑΣ
ΝΟΕΜΒΡΙΟΣ 2017

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Ψηφιακών Συστημάτων

ΠΜΣ "Ψηφιακά Συστήματα & Υπηρεσίες"

Κατεύθυνση: Ψηφιακές Επικοινωνίες & Δίκτυα

«Μελέτη & ανάπτυξη συστήματος VOIP σε περιβάλλον cloud computing με δυνατότητες αυτοματοποιημένων διεργασιών Load balancing»

Καραγιαννάκης Βασίλειος

ΜΕ 1548

Επιβλέπων καθηγητής κ Απόστολος Μηλιώνης

Πειραιάς

Νοέμβριος 2017

Ευχαριστίες

Καθ' όλη την διάρκεια των μεταπτυχιακών μου σπουδών με υποστήριξαν αρκετοί φίλοι, συνάδελφοι και συγγενείς και θα ήθελα να τους ευχαριστήσω πολύ γ αυτό. Ιδιαίτερη όμως αναφορά θα ήθελα να κάνω σε ορισμένους που συνέβαλαν καθοριστικά, συγκεκριμένα:

Στον επιβλέποντα καθηγητή μου κ Απόστολο Μηλιώνη για την δυνατότητα αλλά και την ελευθέρια που μου έδωσε να αναπτύξω ένα τόσο ενδιαφέρον θέμα, σε γνωστικό αντικείμενο αιχμής δίνοντας μου τις απαραίτητες κατευθύνσεις για την σωστή και αποτελεσματική ανάπτυξη του θέματος.

Την οικογένεια μου για όλη την ηθική και υλική υποστήριξη που μου έδωσε ώστε να ολοκληρώσω επιτυχώς τις σπουδές μου.

Τον φίλο μου Γιάννο Σανταμούρη για την προτροπή και ενθάρρυνση του στην λήψη σωστών ακαδημαϊκών επίλογων.

Περίληψη

Τα συστήματα νεφοϋπολογιστικής αποτελούν ένα από τα βασικότερα πεδία αιχμής στον χώρο της πληροφορικής. Προσφέρουν αξιόπιστες δυνατότης σε μεγάλο εύρος εφαρμογών ξεχωριστών επίπεδων. Η παρούσα εργασία ασχολείται με την μελέτη και περιγραφή των δυνατοτήτων ενός συστήματος νεφοϋπολογιστικής στο επίπεδο Iaas. Συγκεκριμένα μελετήθηκε η πλατφόρμα okeanos του δικτύου GRnet τόσο από την πλευρά της συνολικής λειτουργικής συγκρότησης του συστήματος όσο από τις δυνατότητες - εργαλεία που παρέχονται στον τελικό χρήστη. Στην συνέχεια αναπτύχθηκε, με χρήση υπολογιστικών πόρων της πλατφόρμας, σύστημα υποστήριξης επικοινωνιών φωνής με τεχνολογίες VOIP μέσω του λογισμικού Asterisk. Παρατηρήθηκε ότι το αποτέλεσμα είναι εξίσου αξιόπιστο όπως σε φυσικά συστήματα ενώ η δικτυακή διαχείριση που πραγματοποιήθηκε από την υπηρεσία της πλατφόρμας Okeanos, ήταν πλήρως λειτουργική και αποτελεσματική. Όμως ένα cloud σύστημα δεν υποστηρίζει μόνο δυνατότητες virtualization αλλά χαρακτηρίζεται και από τις δυνατότητες εύκολης επέκτασης των παρεχόμενων υπηρεσιών. Έτσι μέσω των Rest Api διεπαφών του okeanos, αναπτύχθηκε σε γλώσσα python πρόγραμμα που επεκτείνει τις δυνατότητες του αρχικού VoIP συστήματος ενσωματώνοντας επιπλέον υπολογιστικούς και δικτυακούς πόρους. Η μετάβαση στο νέο σύστημα γίνεται με απολυτά αυτοματοποιημένο τρόπο και χωρίς την διακοπή της παρεχόμενης υπηρεσίας κατά την μετάβαση, ενώ ο σχεδιασμός του VoIP συστήματος έγινε εξ αρχής με τέτοιο τρόπο ώστε να δεχθεί την προσθήκη νέων συστημάτων χωρίς την ανάγκη συνολικού επανασχεδιασμού.

Abstract

Cloud computing is one of the most challenging concepts on IT industry. These systems provide a big variety of applications on every level. This thesis describes the functionality, tools and architecture of okeanos cloud computing service. Okeanos platform is an Iaas cloud computing system by Grnet that aims to provide a cloud based fully virtualized solution for academic and research community. A basic set of compute and network resources used, in order to develop a fully functional VOIP system. The related system used Asterisk opensource software project to manipulate and coordinate communications between clients. Then an automated monitoring procedure starts a python program. This happens, when networks interface traffic exceeds a minimum threshold. The above Program, communicates with okeanos platform through restful APIs, and creates a new cluster of servers with more compute and network resources. Also, a multi-tier network connectivity schema handles the connection not only between client and servers but also manipulates a trunk connectivity between servers. During the implementation of the above system a smart configuration was designed for the Asterisk servers just to make the transformation easier and with better efficiency.

Ακρωνύμια

API	Application program interface
DAHDI	Digium/Asterisk Hardware Device Interface
Iaas	Infrastucture as a Service
IAX	Inter-Asterisk eXchange
IETFS	Internet Engineering Task Force
IVR	Interactive voice response
KVM	Kernel-based Virtual Machine
MD5	Message-Digest Algorithm
NIST	National Institute of Standards and Technology
Paas	Platform as a Service
PBX	Private Branch Exchange
PSTN	public switched telephone network
QOS	Quality of Service
RFC	Request for Comments
RTP	Real-time Transport Protocol
Saas	Software as a Service
SIP	Session Initiation Protocol
VM	Virtual Machine

Περιεχόμενα

1. Εισαγωγή στις τεχνολογίες cloud computing.

Μέχρι πρότινος, η παρουσίαση του cloud, τόσο στην βιβλιογραφία όσο και σε τεχνικές αναφορές, παρέπεμπε κυρίως στο διαδίκτυο. Άλλωστε, το internet έχει επικρατήσει, ακόμα και σχηματικά, να απεικονίζεται σαν ένα σύννεφο. Ο στόχος ήταν η παρουσίαση με αφηρημένο τρόπο της πολυπλοκότητας που υπάρχει πίσω από αυτό και η εστίαση στην πραγματική υπηρεσία που αυτό προσφέρει, δηλαδή την μεταφορά δεδομένων. Η λογική της αποδέσμευσης των λεπτομερειών και η παρουσίαση τους με αφηρημένο τρόπο έγινε και στους υπολογιστικούς πόρους σε αυτό που αναφέρουμε ως «*cloud computing*». Ένας πλήρης ορισμός για το πότε ένα σύστημα χαρακτηρίζεται ως σύστημα «cloud computing» έχει δοθεί από το διεθνές ινστιτούτο προτύπων και τεχνολογίας (NIST).

~

Cloud computing είναι ένα μοντέλο που υλοποιεί την αδιάλειπτη, εύκολη, κατ' απαίτηση και εξ αποστάσεως πρόσβαση, σε μια κοινόχρηστη συλλογή διαμοιραζόμενων υπολογιστικών πόρων (όπως networks, servers, storage, applications, υπηρεσίες) άμεσα και με το ελάχιστο δυνατό φόρτο διαχείρισης καθώς και αλληλεπίδρασης με τον πάροχο των υπηρεσιών [1].

~

Σαν συνέπεια της παραπάνω αρχιτεκτονικής προκύπτουν τα εξής βασικά χαρακτηριστικά.

- ✓ **Μείωση κόστους** : Η απόκτηση, εγκατάσταση και λειτουργία είναι αποκλειστική υποχρέωση του παρόχου, απαλλάσσοντας τον χρήστη από το cost of ownership.
- ✓ **Ευκολία απόκτησης**: Οι πόροι διατίθενται άμεσα (One click away) με δυνατότητα εύκολης μεταβολής, αν είναι απαραίτητο.
- ✓ **Συντήρηση**: Ο χρήστης δέχεται τις υπηρεσίες απαλλασσόμενος από administrative & maintenance effort .
- ✓ **Ευκολία πρόσβασης**: Μοναδική προϋπόθεση για την αξιοποίηση των υπηρεσιών είναι η δυνατότητα πρόσβασης στο διαδίκτυο.
- ✓ **Αξιοπιστία**: Η παροχή των υπηρεσιών είναι αδιάλειπτη, καθώς η δυνατότητα Migration διαφοροποιεί την διαθεσιμότητα της υπηρεσίας από την κατάσταση του data center.
- ✓ **Επαναχρησιμοποίηση**: Οι παρεχόμενοι πόροι είναι δυνατόν να αξιοποιηθούν ταυτόχρονα από περισσότερους χρήστες .

1.1 Μοντέλα εφαρμογής

- PUBLIC CLOUD

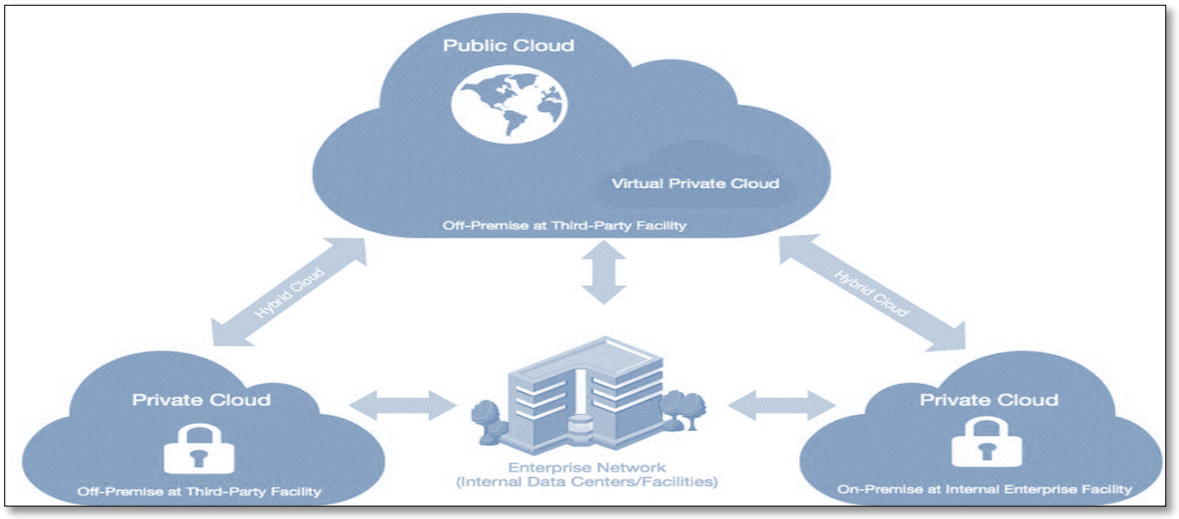
  Η δυνατότητα πρόσβασης στις υπηρεσίες πραγματοποιείται από ένα εκτεταμένο δίκτυο χρηστών. Ανεξάρτητα από την προέλευση τους, (μεμονωμένος χρήστης, επιχειρηματική μονάδα, ακαδημαϊκή κοινότητα) οι χρήστες μπορούν να αιτηθούν την απόκτηση οποιασδήποτε δυνατότητας προσφέρει ο εκάστοτε πάροχος.

- PRIVATE CLOUD

  Η υλοποίηση cloud υποδομών είναι αποδοτική, όταν αφορά ένα εκτεταμένο δίκτυο χρηστών. Τέτοιες υλοποιήσεις πραγματοποιούνται στα πλαίσια επιχειρήσεων ή οργανισμών. Χαρακτηριστικό τους είναι η αποκλειστική χρήση από τα μέλη αυτών των ομάδων, καθώς και η αδυναμία πρόσβασης από τρίτα μέρη.

- HYBRID CLOUD

  Συνδυασμός των προηγουμένων υλοποιήσεων είναι δυνατόν να πραγματοποιήσει ένα μεικτό σχήμα. Συγκεκριμένα, περισσότερα cloud, είτε Public είτε private, μπορούν λειτουργώντας σαν ξεχωριστές οντότητες, να συνδέονται μεταξύ τους για την πραγματοποίηση μιας λειτουργίας πχ load balancing μέσω προκαθορισμένων μηχανισμών [1].



Εικόνα 1 Μοντέλα εφαρμογής cloud computing [23]

1.2 Εικονικοποίηση και Hypervisors

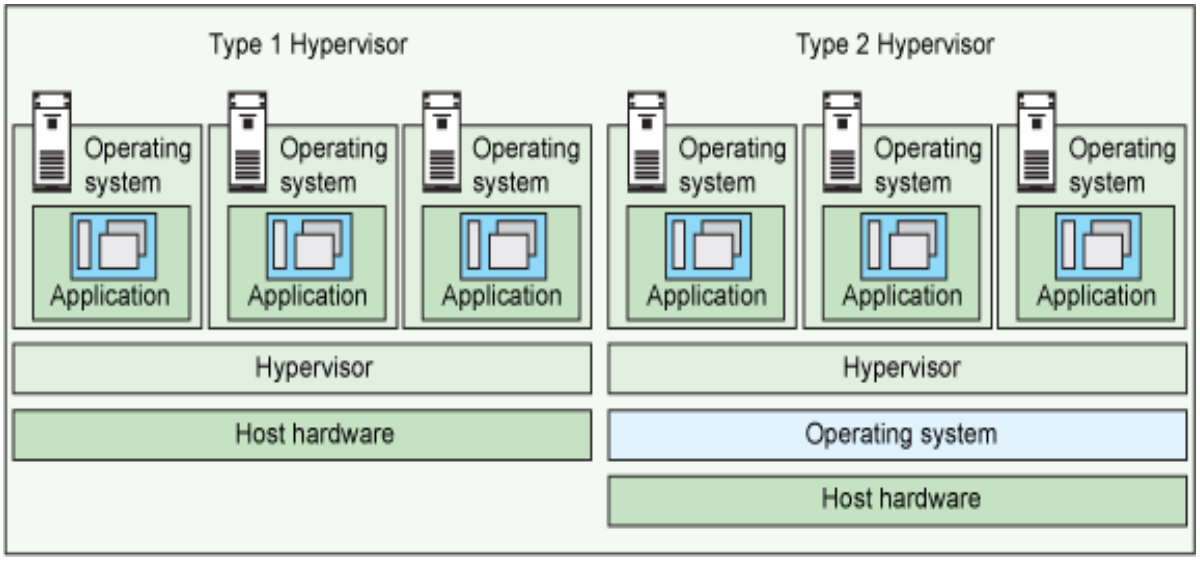
Η λειτουργία κάθε υπολογιστικού συστήματος βασίζεται στην εκτέλεση εντολών μηχανής από κάποιο hardware. Τα δυο αυτά μέρη είναι συνδεδεμένα μεταξύ τους σε ένα σύστημα και μπορούν να επιτελέσουν έναν μοναδικό έργο, αυτό για το οποίο έχουν προγραμματιστεί. Από την περιγραφή του cloud computing, προκύπτει ως αναγκαίο χαρακτηριστικό η ύπαρξη μιας συλλογής πόρων προς διάθεση σε χρήστες που θα εκτελέσουν διαφορετικές εργασίες. Η μέχρι πρότινος λογική της κάλυψης επιπλέον αναγκών σε πόρους με αγορά επιπλέον hardware, αλλά και αδειών software είναι μη αποδοτική. Για τον σκοπό αυτό, είναι αναγκαία η αξιοποίηση της εικονικοποίησης. Σκοπός της εικονικοποίησης είναι η δημιουργία του επιθυμητού αντικειμένου είτε πρόκειται για cpu, ram, storage ή κάποιο operating system εικονικά, δηλαδή να μην έχει άμεσα φυσική υπόσταση. Φυσικά, η ύπαρξη πραγματικών πόρων είναι απαραίτητη [2]. Η δημιουργία των αναφερομένων virtual συσκευών γίνεται με την χρήση ειδικού λογισμικού, που ονομάζεται hypervisor. Η λειτουργικότητα ενός hypervisor βασίζεται στην εισαγωγή ενός επιπλέον λογικού επιπέδου, με σκοπό να αποσυνδεθεί η άμεση σχέση hardware και guest operating system. Στο layer αυτό, αποκτούν υπόσταση τα VMs. Για λογούς αρχιτεκτονικής και ασφάλειας είναι δυνατόν να υπάρξουν hypervisors σε περισσότερα επίπεδα (nested virtualization). Διαθέτουν έναν βασικό hypervisor, στον οποίο τρέχει συνολικά η υποδομή, και σε έναν δεύτερο σε ανώτερο επίπεδο, οπού βρίσκονται τα VMs των χρηστών. Νέες χρήσιμες λειτουργίες που εφαρμόζονται είναι:

- Snapshot: Καθώς λειτουργεί κάθε VM, το μέσο αποθήκευσης που διαθέτει περνάει διαδοχικά στάδια - καταστάσεις. Οι καταστάσεις αυτές απαρτίζονται από τα δεδομένα που έχει το σύστημα κάποια χρονική στιγμή. Σε περίπτωση που είναι αναγκαίο, μπορούμε να ανακτήσουμε μια από αυτές τις καταστάσεις, έτσι ώστε το σύστημα να επανέλθει σε προηγούμενη λειτουργική κατάσταση, εφόσον απαιτείται.
- Migration: Κάθε snapshot αποτελεί ένα σύνολο δεδομένων που μπορεί, με κατάλληλους αλγορίθμους, να μεταφερθεί σε διαφορετικά host systems. Έτσι συμβαίνει Migration σε κάποιο VM. Το κέρδος μας από αυτό είναι η εξασφάλιση high availability.

Τύποι hypervisor

Τύπος 1 – native: Αυτή η κατηγορία Hypervisor εκτελείται απευθείας στο hardware του host συστήματος. Έχει δυο ρόλους. Να εκτελεί όλους τους ελέγχους που απαιτεί το Hardware, για να λειτουργήσει, ενώ ταυτόχρονα να δίνει όλη εκείνη την λειτουργικότητα που απαιτείται για την δημιουργία VMs . Χαρακτηριστικό του είναι ότι δεν απαιτεί κάποιο ήδη εγκατεστημένο λειτουργικό σύστημα. Τα πιο διαδεδομένα προϊόντα αυτού του τύπου είναι τα : VMware ESX/ESXi, Microsoft Hyper-V, XEN .

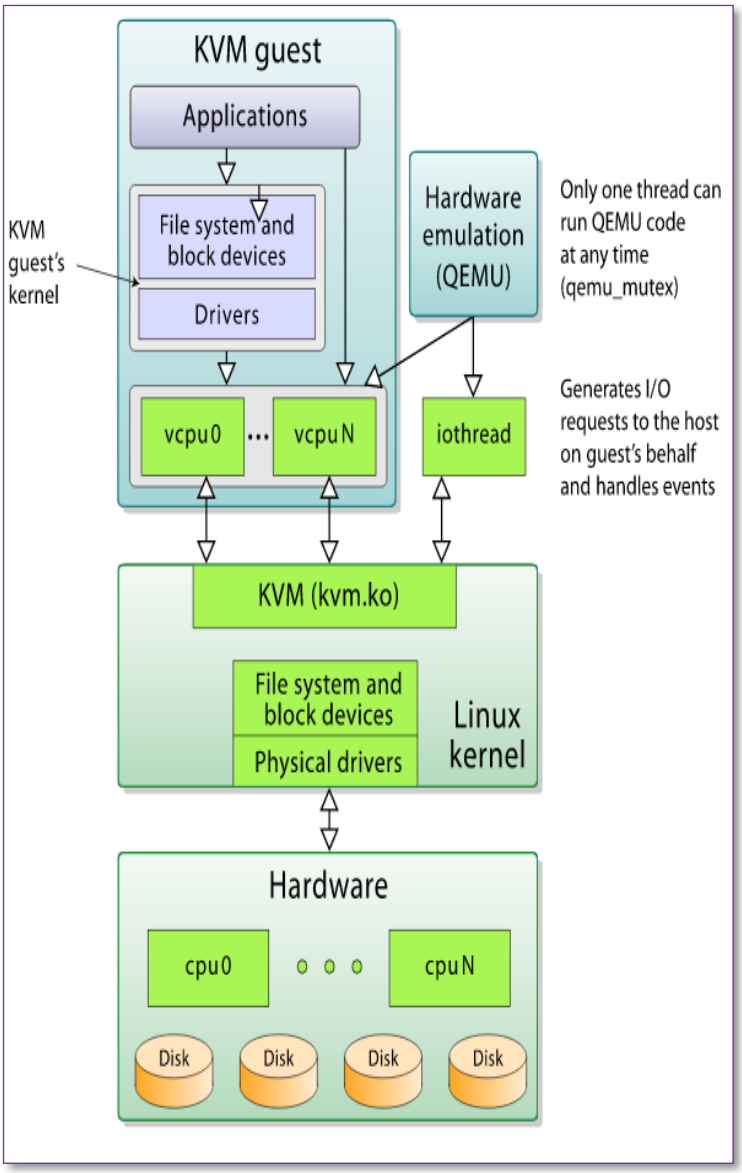
Τύπος 2 – hosted: Η δεύτερη κατηγορία προϋποθέτει ένα ήδη εγκατεστημένο Operating system στο hardware. Ο hypervisor εγκαθίσταται σε επόμενο στάδιο, ενώ για να λειτουργήσει ενθυλακώνει την λειτουργικότητα του στο host OS. Σαν αποτέλεσμα είναι η δημιουργία των VMs να θεωρείται ως μια διεργασία του host OS. Επιπλέον, τα λειτουργικά συστήματα που εκτελούνται στο σύστημα (host & guest OS) είναι σαφώς διαχωρισμένα και δεν υπάρχει καμία άμεση αλληλεπίδραση με το Hardware από τα Host. Τα πιο διαδεδομένα προϊόντα αυτής της κατηγορίας είναι : VMware workstation ,Virtual box by oracle, Linux KVM [2].



Εικόνα 2 Τύποι Hypervisor [3]

KVM hypervisor

Μια από τις πιο διαδεδομένες λύσεις για Virtualization είναι ο KVM hypervisor. Είναι τύπου 2 και απευθύνεται σε Linux based συστήματα αρχιτεκτονικής x86. Είναι ένα Open source software με δυνατότητα υποστήριξης guest OS διαφορετικών παροχών ,όπως της Microsoft, ανεξάρτητα με το αν εγκαθίσταται μόνο σε Linux host OS. Λειτουργεί πραγματοποιώντας τις προσομοιώσεις, όχι στο ίδιο το software, αλλά σε ένα εξωτερικό interface (dev/KVM) σε συνδυασμό με άλλους emulators, όπως τον QEMU. Ο QEMU είναι επίσης, ένα open source software που πραγματοποιεί hardware virtualization με πολλές διαφορετικές εφαρμογές. Στην περίπτωση οπού συνεργάζεται με τον KVM, συμβάλλει στην εικονικοποίηση των πόρων, αλλά η εκτέλεση των images γίνεται από τον KVM. Το High level design ενός τέτοιου συστήματος φαίνεται στο σχήμα [4].



Εικόνα 3 Λειτουργικά στοιχεία KVM [24]

1.3 Επίπεδα παρεχόμενων υπηρεσιών σε cloud συστήματα.

Οι δυνατότητες και υπηρεσίες που προσφέρονται από ένα cloud σύστημα αφορούν ένα πολύ μεγάλο εύρος εφαρμογών με διαφορετικές απαιτήσεις και επιμέρους τεχνικές λεπτομερείες. Στην κατεύθυνση αυτή, έχουν οριστεί τα βασικά πεδία, στα οποία κατηγοριοποιούνται οι υπηρεσίες. Διαφορετικοί πάροχοι για διαφορετικά *επίπεδα* υπηρεσιών προσφέρουν μια πλειάδα δυνατοτήτων, καθώς και εργαλείων διευκολύνοντας και απλοποιώντας τις διαδικασίες απόκτησης, αλλά και ελέγχου του κόστους χρήσης.

Επίπεδο Iaas.

Η δυνατότητα απόκτησης On demand βασικών υπολογιστικών πόρων πραγματοποιείται από το επίπεδο iaas. Με τον ορό «βασικό» αναφέρονται όλες εκείνες οι λειτουργίες που είναι απαραίτητες σε ένα σύστημα. Τέτοιοι πόροι μπορεί να είναι, cpu ,ram, storage, load balancer, καθώς και δικτυακές δυνατότητες. Βρίσκονται, είτε σε φυσική μορφή, (bare metal) είτε σε εικονική. Οι παροχές αυτές πραγματοποιούνται από το layer που βρίσκεται ο hypervisor. Ο χρήστης σε αυτό το επίπεδο, συνήθως, δέχεται ένα εικονικό μηχάνημα με εγκατεστημένο το λειτουργικό σύστημα της επιλογής του, χωρίς να επιδρά στην λειτουργία ή να παραμετροποιεί οτιδήποτε στην βασική υποδομή.

Amazon elastic compute: Είναι η βασική cloud υποδομή της amazon. Διαθέτει web interface για την απόκτηση και εποπτεία των υπηρεσιών, ενώ δίνει την δυνατότητα και για διαχείριση μέσω command line. Εύκολη είναι, επίσης, η αύξηση ή μείωση των δεσμευμένων πόρων. Διαθέτει κάποια βασικά προκαθορισμένα images (Amazon Machine Images) για γρήγορη απόκτηση κάποιου instance. Ενώ η υπηρεσία (Amazon Elastic Block Store) επιτρέπει την σύνδεση των VMs με λογικές μονάδες αποθήκευσης.

Microsoft azure: η πλατφόρμα azure δίνει αντίστοιχες λύσεις για την δημιουργία instances με την διαφορά ότι δίνει και επιπλέον προ εγκατεστημένα κάποια προγράμματα, όπως το share point. Ενσωματώνει ένα διαφορετικό μοντέλο χρέωσης, κατά το οποίο θεωρεί ως αντικείμενο κοστολόγησης τον χρόνο που λειτουργεί το VM, σε αντίθεση με το ECS που χρεώνει την ποσότητα πόρων.

Επίπεδο Paas

Η ανάπτυξη και λειτουργία εφαρμογών προϋποθέτει ένα εύρος εργαλείων και αντικειμένων. Συγκεκριμένα, ανάλογα με την γλώσσα προγραμματισμού που χρησιμοποιείται είναι απαραίτητες οι σχετικές βιβλιοθήκες και services. Το επίπεδο iaas συγκεντρώνει όλες αυτές τις δυνατότητες, παρέχοντας στον developer ένα κατάλληλο περιβάλλον ανάπτυξης εφαρμογών, χωρίς να χρειάζεται να λαμβάνει υπ' όψιν τις λειτουργίες κατωτέρου επιπέδου, όπως servers ή networking. Επιπλέον, επιτρέπει και την offsite διαθεσιμότητα του κώδικα από οποιοδήποτε μέρος.

Google app engine: έχει σχεδιαστεί για κατανεμημένες web based εφαρμογές. Υποστηρίζει java, python PHP και GO. Χρησιμοποιεί ένα μοντέλο sandbox, οπού απομονώνει διαφορετικές λειτουργίες, μειώνοντας το ρίσκο απώλειας της λειτουργικότητας της εφαρμογής στην περίπτωση που κάποιος server τεθεί εκτός λειτουργίας.

Red hat open shift: πλατφόρμα που ενσωματώνει ένα ευρύ σύνολο γλωσσών προγραμματισμού και βάσεων δεδομένων. Είναι open source, ενώ η αρχιτεκτονική του στηρίζεται στην τεχνολογία των containers. Δίνει επιλογές customize του περιβάλλοντος λειτουργίας. Έχει 3 βασικές εφαρμογές, on line, σε private data center και στην open source Πλατφόρμα που γίνεται hosting.

Επίπεδο Saas

Ολοκληρώνοντας την παρουσίαση των επίπεδων εφαρμογής ενός cloud, καταλήγουμε στο πιο διαδεδομένο επίπεδο, το οποίο απευθύνεται στο μεγαλύτερο κοινό χρηστών, το επίπεδο SaaS. Στο επίπεδο αυτό βρίσκονται οι εφαρμογές, οι οποίες είναι παντού διαθέσιμες. Το μόνο που χρειάζεται είναι μια σύνδεση στο διαδίκτυο και ένα web interface (συνήθως κάποιος Think client ή web browser). Ο χρήστης έχει δυνατότητα ελάχιστης, έως καθόλου, παραμετροποίησης. Το μόνο που απολαμβάνει είναι οι δυνατότητες του software. Επιπλέον, δεν έχει καμία επίδραση ή εμπλοκή στα updates ή σε οποιοδήποτε άλλο στοιχείο, το οποίο επηρεάζει την διαθεσιμότητα του προγράμματος [1].

2. Διαχείριση εικονικών πόρων με το Google Ganeti

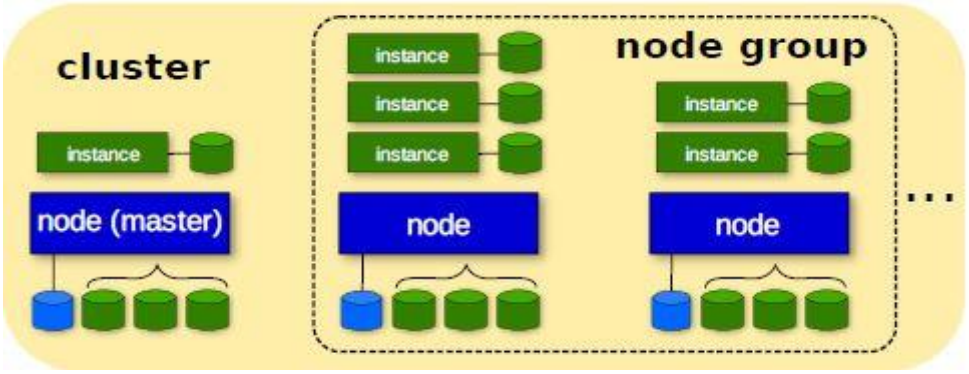
Η πρώτη βασική έννοια που εμπλέκεται στην περιγραφή συστημάτων νεφουπολογιστικής είναι η εικονοικοποίηση. Πολλές φορές, οι 2 αυτές έννοιες μπορεί λανθασμένα να ταυτιστούν. Στην πραγματικότητα, οι υποδομές cloud απαιτούν τεχνολογίες εικονικοποίησης. Για τις ανάγκες εξέλιξης, αλλά και απόδοσης αξιόπιστων λύσεων θεμελίωσης cloud συστημάτων, η Google ανέπτυξε το λογισμικό Ganeti. Σκοπός του λογισμικού είναι να διαχειρίζεται την λειτουργία των VM's, καθώς και την εύρυθμη λειτουργία της κεντρικής υποδομής. Είναι ένα Opensource project, γραμμένο σε γλώσσα python, το οποίο υποστηρίζεται από ένα ευρύτατο community. Η πληθώρα εκδόσεων, αλλά και τα πολλαπλά στάδια testing-debugging, το χαρακτηρίζουν ως ένα ώριμο project. Ο αρχικός σχεδιασμός του προγράμματος ήταν να καλύπτει τις εσωτερικές ανάγκες της Google. Σήμερα αποτελεί ένα πλήρες εργαλείο παροχής Virtualized λύσεων. Να σημειωθεί ότι ένα σύστημα που χρησιμοποιεί το Ganeti δεν είναι cloud σύστημα.

Το Ganeti στηρίζεται στις τεχνολογίες εικονικοποίησης που δίνονται από το KVM και το Xen. Η αρχιτεκτονική του προγράμματος έχει σχεδιαστεί με τέτοιο τρόπο, ώστε να αποσυνδέονται οι λειτουργικές λεπτομέρειες κατωτέρων επίπεδων του hardware από το λογισμικό. Βασικό στοιχείο κάθε συστήματος Ganeti αποτελεί το Node. Είναι η μονάδα, πάνω στην οποία τρέχουν τα παραγόμενα VMs. Μπορούν να εγκατασταθούν από ένα ή και περισσότερα nodes με παράλληλη λειτουργία. Πολλά Node μαζί υλοποιούν ένα cluster. Σημαντικό χαρακτηριστικό αποτελεί και η αντιμετώπιση του συνόλου της υποδομής των clusters ενιαία από το πρόγραμμα. Με αυτόν τον τρόπο, γίνεται πιο αποτελεσματική η λειτουργία του, οι δυνατότητες failover, καθώς και πιο αποτελεσματικό Load balancing. Το Ganeti project σχεδιάστηκε με στόχο την εύκολη εγκατάσταση , διαχείριση, αλλά και αναβάθμιση των εγκατεστημένων υποδομών, ενώ είναι ικανό να λειτουργήσει με έως 200 VMs ανά cluster. [5]

Εγκατάσταση λειτουργικών μονάδων

Τα πλεονεκτήματα virtual λύσεων αναδεικνύονται περισσότερο, όταν εφαρμόζονται σε μεγάλης κλίμακας συστήματα, με μεγάλες απαιτήσεις πόρων, όπως είναι για παράδειγμα, μια επιχείρηση. Έτσι, σε όλες τις περίπτωσης λειτουργίας Ganeti λογισμικού θα παρατηρείται μια εκτεταμένη υποδομή σε κάποιο data Centre που θα απαρτίζεται από πολλά nodes. Τα nodes είναι συνδεδεμένα μεταξύ τους, τόσο με φυσικό τρόπο όσο και σαν ενιαία λειτουργική οντότητα, με αποτέλεσμα η «λογική» του συστήματος να μοιράζεται σε αυτά τα Nodes του cluster.

Το Ganeti χαρακτηρίζεται ως εργαλείο που δίνει Compute resource,s παρόλα αυτά υπάρχει δυνατότητα υποστήριξης storage.



Εικόνα 4 Λογικά στοιχεία cluster [6]

Το βασικό workflow της υποδομής για την διαχείριση του κύκλου ζωής των VM's είναι το εξής

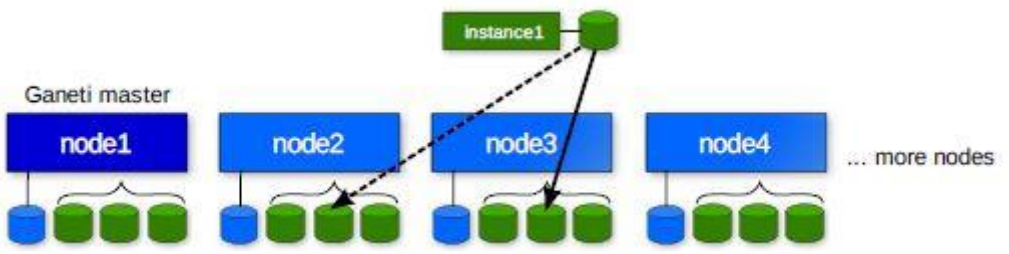
i. Τα VM's εκκινούν την λειτουργία τους σε ένα προκαθορισμένο βασικό Node, ορισμένο για κάθε cluster που ονομάζεται Master node.

ii. Γίνεται σύνδεση του VM με κάποια εικονική μονάδα αποθήκευσης.

iii. Δημιουργείται ένα αντίγραφο του εικονικού σκληρού δίσκου σε κάποιο γειτονικό node.

Διαδικασία migration

Το Ganeti προβλέπει διαδικασίες για την μεταφορά του κόμβου "φιλοξενίας" ενός VM, αν αυτό καταστεί αναγκαίο. Οι λόγοι που μπορεί να επιβάλουν μια τέτοια μετακίνηση είναι, είτε στην περίπτωση, οπού εμφανιστεί αδυναμία ορθής λειτουργίας σε ένα VM, είτε για να γίνει ισοκατανομή στους κόμβους των ενεργών εικονικών μηχανημάτων για τις ανάγκες load balancing.



Εικόνα 5 Αναπαράσταση μεταπήδησης VM σε νέο node [6]

Έτσι, από το αρχικό στάδιο, οπού κατά την έναρξη λειτουργίας του VM βρίσκεται στο master node και ταυτόχρονα διατηρείται ένα αντίγραφο του αποθηκευτικού χώρου σε γειτονικό node, μεταβαίνουμε με τα παρακάτω βήματα σε νέο κόμβο. [5]

Migration

i. Ο scheduler επιλέγει τον κόμβο που θα μεταφερθεί το VM
ii. Διασφαλίζεται ότι και στις 2 θέσεις βρίσκεται συγχρονισμένη εικόνα του σκληρού δίσκου
iii. Αποσυνδέεται το VM από τον σκληρό δίσκο όπου ήταν δεσμευμένο
iv. Γίνεται η μετάβαση στον νέο κόμβο και γίνεται σύνδεση με το αντίγραφο του σκληρού δίσκου, όπως αυτό έγινε στο βήμα ii
v. Δημιουργείται ένα επιπλέον αντίγραφο του σκληρού δίσκου σε γειτονικό node αποδεσμεύοντας ταυτόχρονα την αρχική μονάδα δίσκου.

3. Openstack project

Η ανάπτυξη συστημάτων cloud αποτελεί ένα τεχνικά σύνθετο αντικείμενο. Νέες προκλήσεις γύρω από θέματα, όπως η ασφάλεια πληροφοριών και η δυνατότητα υποστήριξης αξιόπιστων λύσεων για εμπορική χρήση είναι μερικές από αυτές. Σαν συνέπεια, υπήρχε ,κατά κανόνα, στην ανάπτυξη των συστημάτων cloud εμπλοκή, αποκλειστικά από μεγάλων εταιριών του χώρου με άρτια τεχνογνωσία. Η πεποίθηση αυτή ανατράπηκε από την σύμπραξη NASA & Rackspace, όπου το 2010 σε ένα Project δημιούργησαν την πρώτη opensource λύση cloud συστημάτων. Το Project ονομάστηκε OPENSTACK. Το λογισμικό αυτό είχε σαν στόχο να δώσει την δυνατότητα απόκτησης cloud υποδομών από οργανισμούς με αξιοποίηση συμβατικού τύπου hardware.

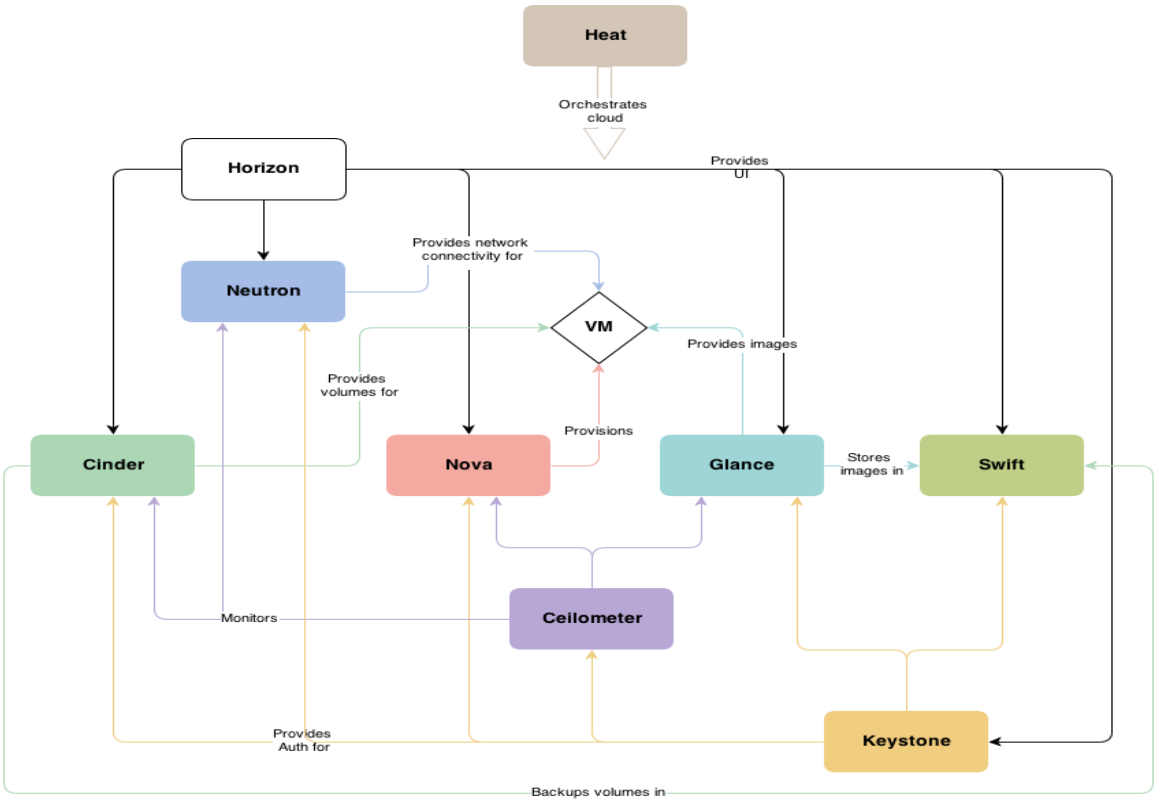
Από τον αρχικό σχεδιασμό έχουν υπάρξει πολλές αλλαγές , βελτιώσεις αλλά και διεύρυνση των δυνατοτήτων που προσφέρει. Ένα από τα ζωτικά κομμάτια που διαθέτει το OPENSTACK είναι το πολύ μεγάλο Community που το υποστηρίζει. Το Community ιδρύθηκε το 2012 ως μη κερδοσκοπικός οργανισμός με σκοπό την προώθηση του λογισμικού. Αποτελείται από έναν μεγάλο αριθμό εταιριών,- πάνω από 500- χωρίς όμως να σταματάει εκεί. Οποιοσδήποτε μπορεί να συμμετάσχει στο Community προσφέροντας, είτε ως developer είτε ως tester, είτε ακόμα και ως ενημέρωση και δημιουργία Bug reports και documentations. Αναλυτικές οδηγίες και κανόνες υπάρχουν στην ιστοσελίδα του Project. Το Community, 2 φορές τον χρόνο, μετα από συνεχή ανάπτυξη και βελτίωση του προγράμματος βγάζει μια νέα έκδοση αναβαθμίζοντας τα Services που το αποτελούν. [7]

Η δομή και η αρχιτεκτονική του openstack είναι αρκετά συνθέτη, ενώ το λογισμικό αποτελείται από αρκετές χιλιάδες γραμμές κώδικα. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Python. Μπορεί να λειτουργήσει σε κάθε μοντέλο εφαρμογής (public-private-hybrid). Η βασική του χρήση είναι η προσφορά υπηρεσιών στο Iaas επίπεδο. Επιπρόσθετα, έχουν αναπτυχθεί και project που μπορούν να ενσωματώσουν λειτουργίες και άλλων επίπεδων, όπως bare metal, containers κα.

Το openstack ΔΕΝ είναι hypervisor

Σε αντιπαραβολή με ένα παραδοσιακό stack εικονικοποίησης, όπως παρουσιάστηκε στο προηγούμενο κεφάλαιο, το openstack βρίσκεται ένα λογικό επίπεδο πάνω από τον hypervisor. Δηλαδή, είναι ένα ενδιάμεσο level, στο οποίο υλοποιείται η αρχιτεκτονική του. Το συνολικό Project το συνθέτουν επιμέρους αυτόνομα τμήματα (Services), καθένα από τα οποία εκτελεί και είναι υπεύθυνο για συγκεκριμένες λειτουργίες. Χαρακτηριστικό των Projects είναι ο βαθμός «ωριμότητας» που έχει επιτευχθεί για κάθε ξεχωριστό service, εννοώντας το κατά ποσό έχουν βγει αρκετές stable releases. Υπάρχουν τα βασικά Project, απαραίτητα σε κάθε υλοποίηση και κάποια επιπλέον προαιρετικά που εγκαθίστανται ανάλογα με τις ανάγκες και υπηρεσίες που δύναται να προσφερθούν. Ορίζοντας, τελικά, τι είναι το Openstack αναφέρεται ότι:

Είναι ένα λογισμικό με Modular δομή, όπου παράγει RESTfull APIs, με στόχο την ενσωμάτωση αποδοτικών και αξιόπιστων λειτουργιών στο επίπεδο Iaas.



Εικόνα 6 Διασύνδεση βασικών services του Openstack [8]

3.1 Openstack core services

Το openstack Project υλοποιεί ένα πολύπλοκο και σύνθετο σύστημα που εγκαθίσταται σε κατανεμημένο περιβάλλον. Η πληθώρα εργασιών που εκτελεί για να επιτελέσει τον λειτουργικό σκοπό γίνεται μεν σε hardware συμβατικής αρχιτεκτονικής, αλλά με υψηλές απαιτήσεις σε πόρους. Για να εγκατασταθεί ένα πλήρες περιβάλλον, ώστε να καλυφθούν παραγωγικές ανάγκες, και όχι απλώς τα βασικά εργαλεία που χρειάζεται ένας μεμονωμένος χρήστης, απαιτούνται πολλοί κόμβοι. Στους κόμβους αυτούς τρέχουν τα βασικά Services που περιγράφονται στην συνέχεια. Οι ονομασίες των κόμβων με τα αντίστοιχα service που φιλοξενούν είναι: [9]

- ✓ Controller node → Keystone & Glance service
- ✓ Network node → Neutron service
- ✓ Compute node → Nova service
- ✓ Block storage node → Cinder service
- ✓ Object storage node →swift service

Identity service (Keystone)

Η εκτέλεση οποιασδήποτε ενέργειας στην υποδομή προϋποθέτει την πιστοποίηση και παραχώρηση των αντίστοιχων δικαιωμάτων του εκάστοτε χρήστη. Η λειτουργία αυτή γίνεται από το Keystone. Επιπλέον, είναι το endpoint που φέρνει σε επαφή όλα τα services. Η λειτουργία του περιλαμβάνει 2 στοιχεία. Πρώτο, τον αρχικό έλεγχο των διαπιστευτηρίων που δίνει ο χρήστης. Εάν είναι επιτυχής ο έλεγχος, στην συνέχεια αποδίδεται ένα token που περιλαμβάνει όλα τα δικαιώματα που έχει και αξιοποιεί όταν ζητείται στο λογισμικό να πραγματοποιήσει οποιαδήποτε εντολή. Δεύτερο, καταχωρεί τις δικτυακές τοποθεσίες των ενεργών services. Η υπηρεσία διαθέτει τα εξής components.

Server: Κεντρικός server που εκτελεί όλες τις λειτουργίες, όπως αυτές ορίζονται από το identity API.

Drivers & modules: είναι υπεύθυνα για την σωστή λειτουργία του server, ενώ αντλούν, ανακτούν και επιστρέφουν οποιεσδήποτε σχετικές πληροφορίες ,είτε από εσωτερικές βάσεις δεδομένων, είτε από εξωτερικές θέσεις αποθήκευσης.

Compute service (Nova)

Το nova service λειτουργεί με σκοπό να εγκαθιστά νέα VMs και να διαχειρίζεται τον κύκλο ζωής τους, κατά την διάρκεια εκτέλεσης εργασιών, όταν "τρέχουν" πάνω στην cloud υποδομή. Η λογική του service βασίζεται σε αρχιτεκτονική ανταλλαγής μηνυμάτων. Με τον τρόπο αυτό αποσυνδέονται οι λειτουργίες από έναν και μόνο server με αποτέλεσμα να δίνεται η δυνατότητα διασποράς των επιμέρους components σε πόλους κόμβους, λειτουργώντας, όμως τελικά, με ενιαίο τρόπο. Οι ενέργειες προς εκτέλεση μπαίνουν σε ένα ειδικό queue και εκτελούνται από το API. Τα επιμέρους components περιλαμβάνουν

Scheduler: για την οργάνωση του τρόπου απόδοσης πόρων σε χρήστες.
Compute: είναι υπεύθυνο για την επικοινωνία του hypervisor με τα VM's
DB: SQL βάση δεδομένων για την αποθήκευση δεδομένων του API
Network: μεταδίδει δεδομένα, δημιουργεί bridges και vlans. Παρόλα αυτά δεν είναι το υπεύθυνο service για την απόδοση network λειτουργικότητας στα VM's

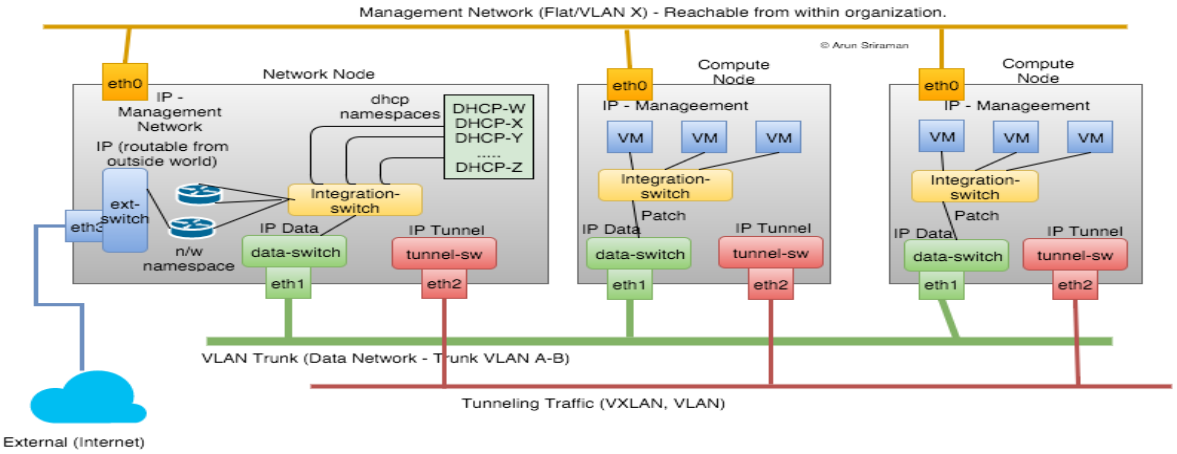
Block storage service (Cinder)

Όπως έχει σημειωθεί τα services του openstack συνεργάζονται για να λειτουργήσουν. Έτσι, αφού το Nova service δημιουργήσει ένα VM συνδέεται με το Cinder, για να προσφέρει δυνατότητες αποθήκευσης. Το API του cinder δεσμεύει αποθηκευτικό χώρο από τους σκληρούς δίσκους της υποδομής και παράγει μια συλλογή από εικονικούς σκληρούς δίσκους μεγέθους ανάλογου των απαιτήσεων του χρήστη και τους συνδέει στο εκάστοτε VM. Επιπλέον, προσφέρει μια ολοκληρωμένη υπηρεσία που υποστηρίζει:

Snapshots: Αποθηκεύει τις καταστάσεις που περνάει κάθε volume κατά την διάρκεια λειτουργίας του επιτρέποντας, είτε την σύνδεση μιας δεδομένης απεικόνισης (snapshot) σε ένα διαφορετικό VM, είτε την δυνατότητα για failover σε περίπτωση που εμφανιστεί κάποιο σφάλμα.
Backup: Κρατάει δεδομένα που βρίσκονταν στο volume κάποια χρονική στιγμή για την περίπτωση, όπου ζητηθεί πάλι ανάκτηση τους.

Network service (Neutron)

Το neutron service είναι υπεύθυνο για την επίτευξη δικτυακής λειτουργικότητας σε κάθε πτυχή της πλατφόρμας. Προσφέρει δυνατότητες για εγκατάσταση εικονικών δικτύων, Interfaces, υποδικτύων και δρομολογητών. Οι δυνατότητες αυτές μιμούνται τις διεργασίες που πραγματοποιούνται στα φυσικά δίκτυα, δηλαδή κάθε δίκτυο αποτελείται από υποδίκτυα, τα οποία δρομολογούν μεταξύ τους πακέτα. Αντίθετα, η ιδιαιτερότητα που υπάρχει είναι ότι δεν μπορεί ένα εικονικό Interface να συνδεθεί άμεσα με δίκτυα εκτός του cloud. Έτσι, κάθε εγκατάσταση περιλαμβάνει τουλάχιστον ένα φυσικό interface. Αυτό στην συνέχεια συνδέεται με έναν εικονικό δρομολογητή, ο οποίος αναλαμβάνει την διαμεσολάβηση για μεταφορά πακέτων στα υποδίκτυα. Να σημειωθεί ότι το Neutron είναι, επίσης, υπεύθυνο για την λειτουργία του εσωτερικού δικτύου της υποδομής εξυπηρετώντας τις διεργασίες μεταξύ των services.



Εικόνα 7 Τοπολογίες εσωτερικών και εξωτερικών δικτύων [10]

Το neutron Services υποστηρίζει και δυνατότητες firewalling για τα VMs. Ειδικότερα, σε πρώτο στάδιο ο διαχειριστής της πλατφόρμας ορίζει security group, τα οποία στην συνέχεια εφαρμόζουν firewall rules. Ανάλογα με το group στο οποίο έχει ενταχθεί το κάθε ξεχωριστό VM, επιτρέπεται ή αντίστοιχα εμποδίζεται η κίνηση σε καθορισμένα ports , η καθορισμένου τύπου κίνηση, τόσο στα εσωτερικά δίκτυα όσο και σε κίνηση εκτός της πλατφόρμας. Τέλος, μια εκτεταμένη συλλογή από Modules μπορεί να διευρύνει τις δυνατότητες του service πέρα από τις βασικές λειτουργίες. Τέτοια Plugins μπορούν να προσφέρουν firewalling as a service, VPNs η ακόμα και load balancer as a service. [9]

Object storage service (Swift)

Το πιο σημαντικό κομμάτι, μετα την λειτουργικότητα, σε ένα σύστημα είναι τα δεδομένα και η δυνατότητα διαχείρισης τους. Σε κλίμακα cloud υποδομής, τεράστιος όγκος δεδομένων διακινείται με ασύντακτο τρόπο από τους χρήστες στο ενιαίο backend. Το Swift API αναλαμβάνει να συντονίσει την ροή δεδομένων με κατάλληλους αλγορίθμους, έτσι ώστε να μπορεί να γίνει με γρήγορο και αποδοτικό τρόπο η καταχώρηση - ανάκτηση τους, ενώ επιπλέον έχει δυνατότητα περαιτέρω ανάπτυξης του χώρου αποθήκευσης, ώστε να μην χρειάζεται επανασχεδιασμός και νέα διεύθυνση της θέσης των δεδομένων. Σημαντική είναι η εξασφάλιση της ακεραιότητας δεδομένων με χρήση μηχανισμών replication.

Image service (Glance)

Τέλος, για την ολοκληρωμένη παροχή ενός επιτυχημένου εικονικού συστήματος σε ένα Iaas cloud πρέπει στο παραγόμενο VM να αποδοθεί και ένα λειτουργικό σύστημα. Τον ρολό αυτόν έχει το Glance API. Αποτελεί εναν καταχωρητή που διατηρεί σε μια βάση δεδομένων την λίστα με τα διαθέσιμα λειτουργικά συστήματα, είτε αυτά είναι βασικές εμπορικές εκδόσεις, είτε εξατομικευμένα Images κάποιου χρήστη. Θεωρείται αυτόνομο service, αλλά λειτουργεί σε πολύ στενή σχέση με το Swift. Στο Swift βρίσκονται στην πραγματικότητα αποθηκευμένα τα images. Έτσι, το Glance ενεργεί ως η διεπαφή που θα τα ανακτήσει, ώστε να τα κάνει διαθέσιμα σε κάποιο VM. [9]

4. Okeanos Iaas cloud solutions

Το δίκτυο grnet, στην προσπάθεια του να δώσει την δυνατότητα, κυρίως στην ακαδημαϊκή κοινότητα λύσεων cloud ανέπτυξε την πλατφόρμα OKEANOS Iaas. Σαν στόχο έχει την παροχή ψηφιακών λύσεων για εκπαιδευτικούς και ερευνητικούς σκοπούς. Το λογισμικό, που υποστηρίζει το okeanos, προέρχεται από Opensource άδειες. Σχεδιάστηκε με στόχο την ελάχιστη επιφόρτιση του τελικού χρήστη, κάνοντας το εύχρηστο σε ευρύτερο κοινό. Ταυτόχρονα, δίνεται η δυνατότητα σε χρήστες μεγαλύτερου τεχνικού υπόβαθρου να διαχειριστούν servers, με επαυξημένες δυνατότητες παραμετροποίησης.

Οι δυνατότητες που απολαμβάνει ο τελικός χρήστης παρέχονται μέσω web διεπαφής και περιλαμβάνουν.

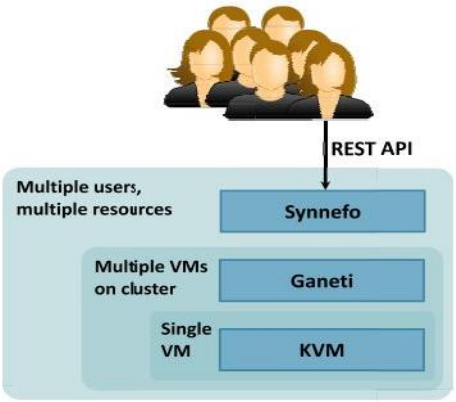
- Εικονικές μηχανές VMs
- Εικονικές δικτυακές διεπαφές ethernet
- Εικονικά μέσα αποθήκευσης
- Firewalling δυνατότητες

Το τελικό αποτέλεσμα προήλθε από την σύνθεση επιμέρους υπηρεσιών με κυριότερα το KVM , το Open stack & το Google Ganeti. Η ενορχήστρωση τους γίνεται από το λογισμικό διαχείρισης με την ονομασία ***synnefo***. Τα κομμάτια από τα οποία αποτελείται το okeanos και η φιλική τους ονομασία είναι:

- Cyclades: networking & compute
- Archipelago: volume storage
- Pithos+: αποθήκευση αρχείων
- Astakos: ταυτοποίηση χρηστών
- Plankton: καταχώρηση των images
- Aquarium: υπηρεσία κοστολόγησης

4.1 Υπηρεσίες πλατφόρμας Okeanos Iaas

Cyclades



Εικόνα 8 Τεχνολογίες που χρησιμοποιούνται από το Okeanos [11]

Η υπηρεσία της πλατφόρμας "okeanos" ,υπεύθυνη για την διαχείριση του δικτυακού και υπολογιστικού μέρους ονομάζεται Cyclades. Αποτελείται από 2 βασικά μέρη, το πρώτο αφορά το κομμάτι του frontend. Είναι το UI αλληλεπίδρασης του χρήστη με τα vm και τα Virtual networks. Η ανάπτυξη του έχει γίνει με χρήση JavaScript/jQuery και τρέχει αποκλειστικά στην πλευρά του client, για καλύτερη απόκριση. Μέσω του αντίστοιχου API μεταφέρει ασύγχρονες κλήσεις REST στον Server. Το δεύτερο κομμάτι περιλαμβάνει το backend. Αποτελείται από ένα σύνολο nodes με εγκατεστημένες Unix εκδόσεις λειτουργικού και χρήση του KVM για virtualization. Η ενορχήστρωση συνολικά του cluster γίνεται από το google ganeti. Το ganeti είναι υπεύθυνο για την εποπτεία και ρύθμιση του cluster των nodes, καθώς αναλαμβάνει την δημιουργία των Vm, το migration μεταξύ των φυσικών κόμβων, αλλά και την διαχείριση τυχόν downtime. Τα σχετικά API για compute management έχουν προέλθει από το Open stack σε συνδυασμό με 3rd. party εργαλεία και βιβλιοθήκες.

Οι δικτυακές δυνατότητες είναι επίσης μια αρμοδιότητα του Cyclades. Προσφέρει λειτουργικότητα για ipv4 & ipv6 συνδεσιμότητας σε κάθε Vm. Είναι σχεδιασμένο, ώστε η παραμετροποίηση να γίνεται ευκολά από το γραφικό περιβάλλον. Είναι δυνατή η εφαρμογή τοπολογιών σε layer 2, είτε για private, είτε για public networks μεταξύ των virtual interfaces. Σημαντικό εργαλείο είναι και το virtual firewalling στα vm με 3 βασικά προφίλ ρυθμίσεων έτσι ώστε να ορίσει τον βαθμό έκθεσης του vm σε εισερχόμενη δικτυακή κίνηση. [11]

Archipelago

Πολύ σημαντικό χαρακτηριστικό κάθε συστήματος αποτελεί η δυνατότητα αποθήκευσης και ανάκτησης δεδομένων από αποθηκευτικούς χώρους. Σε ένα virtualized περιβάλλον, οπού οι φυσικοί πόροι άντλησης είναι κοινοί χρειάζονται τεχνικές διαχείρισης του χώρου δεδομένων από το μεγάλο πλήθος vm's που θα τους χρησιμοποιούν. Έτσι, μέσω της υπηρεσίας του archipelago παράγεται μια συλλογή εικονικών σκληρών δίσκων, άμεσα διαθέσιμων, που επισυνάπτονται τελικά στα VM's. Το μέγεθος των δίσκων μπορεί να αυξηθεί, κατά την διάρκεια χρήσης, να προστεθεί επιπλέον μονάδα ή και να αποδεσμευθεί εντελώς.

Pithos+

Στην πλατφόρμα okeanos τον ρολό του συντονισμού αποθήκευσης δεδομένων παίζει η υπηρεσία Pithos+. Σε τεχνικό επίπεδο, εφαρμόζει τα APIs που δίνονται από το openstack object storage services. Κάθε αποθηκευμένο αρχείο είναι τοποθετημένο σε ένα σύνολο blocks με δεδομένη διεύθυνση. Κάθε φορά που αλλάζει μόνο ένα μέρος του αρχείου ενημερώνεται το αντίστοιχο block και όχι συνολικά το αρχείο με στόχο τον αποδοτικότερο συγχρονισμό αλλαγών. Η αλληλεπίδραση με το χρήστη γίνεται και μέσω της web διεπαφής, αλλά και από το command line της πλατφόρμας με την φιλική ονομασία kamaki. Η υπηρεσία αξιοποιεί κοινό backend με το archipelago service.

Plankton

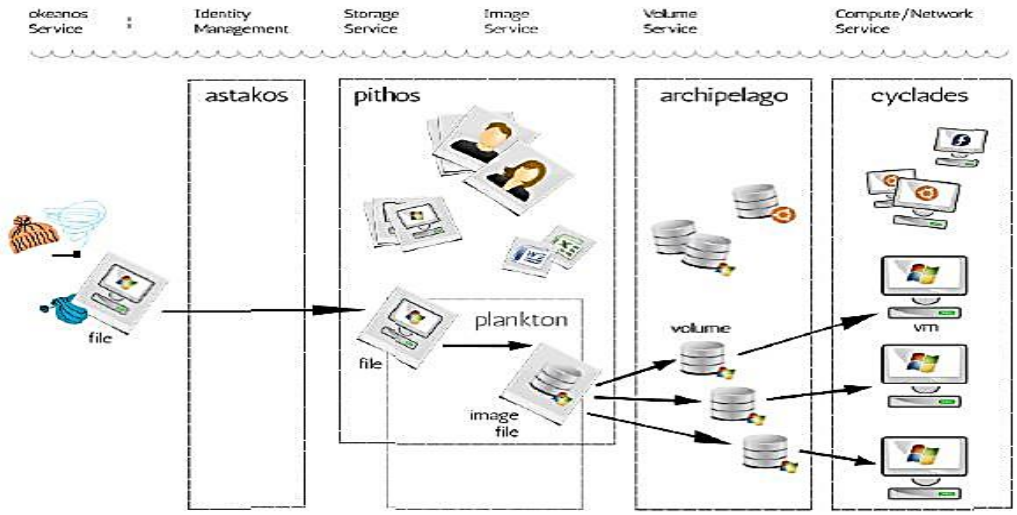
Στην βάση του Pithos+, ενθυλακώνεται και μια επιπλέον υπηρεσία, η οποία για λογούς διαφοροποίησης του σκοπού που επιτελεί έχει την ξεχωριστή ονομασία Plankton. Σκοπός της είναι να αποθηκεύει τα images που περιέχουν τα λειτουργικά συστήματα, ενώ ταυτόχρονα αυτά τα Images καταχωρούνται σε μια βάση, ώστε να είναι ανιχνεύσιμα & ανακτήσιμα από VMs. Η τεχνολογία του βασίζεται στο openstack Glance APi με δυνατότητα χρήσης, τόσο custom images από τους βασικούς vendors, όσο διαφοροποιημένα για εξειδικευμένη χρήση.

Astakos

Η διεπαφή του χρήστη με την πλατφόρμα ,με άμεσο τρόπο, μπορεί να γίνει σε μόλις δυο από τις υπηρεσίες, την υπηρεσία Cyclades και Pithos+. Ο καθορισμός των χρηστών, αλλά και των δικαιωμάτων που κάθε χρήστης διαθέτει σε αυτές τις 2 υπηρεσίες ορίζεται από τον διαχειριστή ταυτότητας Astakos. Είναι το σημείο, όπου πραγματοποιείται το authentication και authorization των χρηστών. Βέβαια, σε πρώτο στάδιο έχει προηγηθεί η εγγραφή του χρήστη, ενώ σε συνεργασία με το Cyclades γίνεται και το login. Διαθέτει λειτουργικότητα με tokens για την διευκόλυνση πραγματοποίησης διαδικασιών από τον χρήστη, χωρίς την ανάγκη συνεχόμενων login για κάθε διαφορετική διεργασία. Τέλος, αναπτύσσονται και δυνατότητες, εκτός από τους τοπικούς λογαριασμός, να αξιοποιούνται APIs κοινωνικών δικτυών, όπως το twitter.

Aquarium

Η διαδικασία κοστολόγησης για κάθε cloud πάροχο γίνεται από την μίξη τεχνικών, αλλά και διαχειριστικών κριτηρίων. Το aquarium είναι μια υπό ανάπτυξη λειτουργία του okeanos για την παροχή δυνατοτήτων billing. Έχει σχεδιαστεί, έτσι ώστε να συλλέγει events που πραγματοποιεί κάθε χρήστης, αλλά και τους πόρους που χρησιμοποίησε στο Cyclades - Pithos+. Με αυτόν τον τρόπο, γίνεται χρέωση των χρηστών σε credits. Σκοπός είναι η αποτροπή αλόγιστης χρήσης πόρων, ενώ λαμβάνονται υπόψιν αντικειμενικά κριτήρια, όπως αριθμός VMs και Gb αποθήκευσης. [12]



Εικόνα 9 Λογική σύνδεση υπηρεσιών του okeanos [11]

4.2 Χαρακτηριστικά πλατφόρμας Okeanos

Google ganeti

Χαρακτηρίζεται ως η καρδιά του συστήματος, διότι σε αυτό υλοποιείται το τελικό σύστημα για τον χρήστη. Αξιοποιεί τις δυνατότητες του kvm hypervisor για την δημιουργία των VMs, υποστηρίζοντας μεγάλο αριθμό λειτουργικών συστημάτων. Είναι υπεύθυνο για την συνολική διαχείριση του backend με αξιόπιστο και αποδοτικό τρόπο. Τα πλεονεκτήματα που προσφέρει αφορούν την δυνατότητα εύκολης επέκτασης του συστήματος, όταν καταστεί αναγκαία η διεύρυνση των πόρων και η αξιόπιστη - αδιάληπτη παροχή των υπηρεσιών – πόρων, μέσω μηχανισμών redundancy, migration & downtime handling . Σημαντικό πλεονέκτημα είναι και ο καλά δομημένος κώδικας με σαφή διαχωρισμό των εργασιών σε αυτόνομα layers.

RESTful API

Κάθε κομμάτι της λειτουργικότητας που παρέχει το okeanos δίνεται και μέσω RESTfull APIs. Προέρχονται από τα Services του Openstack με βάση ανοικτά Standard. Το πλεονέκτημα που δίνει μια τέτοια λύση είναι η δυνατότητα ενσωμάτωσης διαφορετικών εργαλείων διαχείρισης, ενώ δίνει και την δυνατότητα σε developers να αναπτύξουν εξατομικευμένα εργαλεία εκτέλεσης εργασιών από τα services.

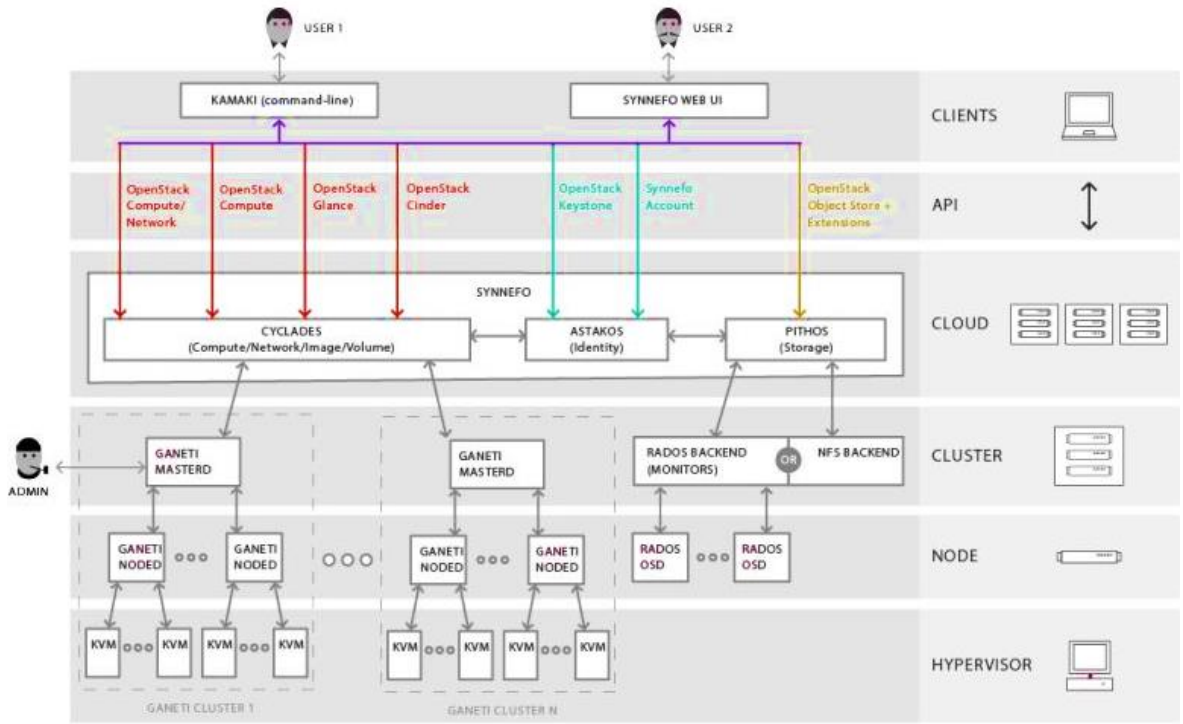
Διαφορετικές δυνατότητες διαχείρισης

Πρωταρχικό μέλημα σχεδιασμού του okeanos ήταν η δυνατότητα χρήσης του από κοινό μικρότερου τεχνικού υποβάθρου. Δεν έπρεπε ταυτόχρονα να λείπει και η δυνατότητα εξοικειωμένων χρηστών να εκτελέσουν πιο συνθέτες διεργασίες. Σαν αποτέλεσμα, δίνεται η αλληλεπίδραση με το σύστημα με δυο ξεχωριστούς τρόπους. Μια γραφική διεπαφή web based που τρέχει στον client και ένα command Line με ,την ονομασία kamaki, με στόχο την εκτέλεση υψηλοτέρου επιπέδου administrative tasks από advanced users & developers.

Software defined networking

Η δικτυακή υποδομή του okeanos στοχεύει στην αποτελεσματική επικοινωνία των VMs, τόσο σε εικονικά ιδιωτικά δίκτυα, όσο και στην διασύνδεση με το Internet. Για την public networking διασύνδεση υποστηρίζονται τα πρωτόκολλα IPV4 & IPV6. Η διευθέτηση των εργασιών γίνεται από το google ganeti με δυνατότητα εξατομικευμένων ρυθμίσεων ανάλογα με τις ανάγκες του χρήστη. Οι διευθύνσεις δίνονται από ένα σύνολο ips χωρίς την διαμεσολάβηση NAT server. Η λειτουργικότητα NAT μπορεί να αξιοποιηθεί μόνο κατά την διαδικασία migration.

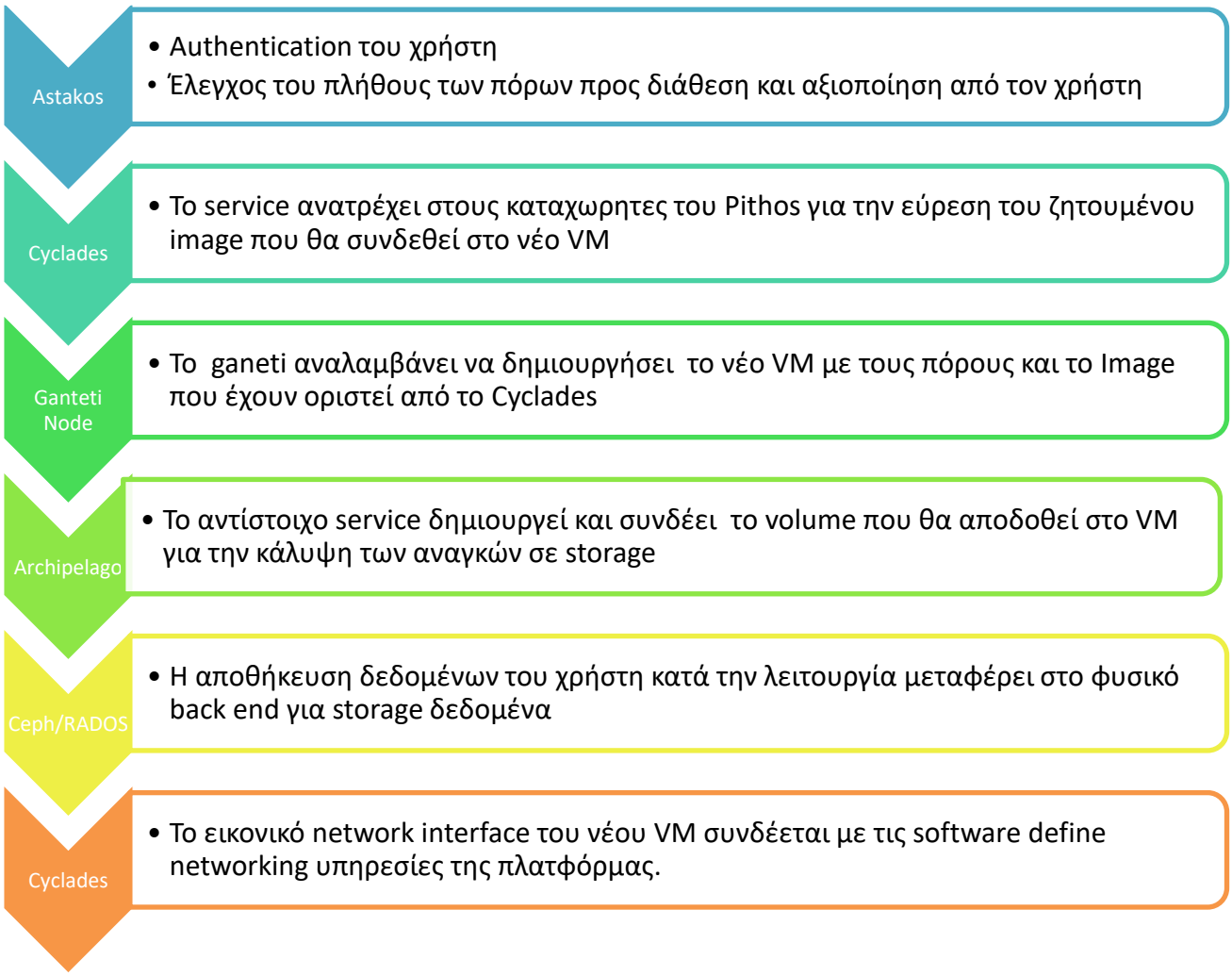
Για τις διεργασίες private networking σε layer 2 δίνεται ένα εύρος βασικών υπηρεσιών. Συγκεκριμένα, η πλατφόρμα παρέχει virtual ethernet resources, στα οποία δίνεται η δυνατότητα υλοποίησης πολυεπίπεδων τοπολογιών. Εικονικές κάρτες δικτύου μπορούν να συνδεθούν σε κάποιο Vm επιτρέποντας την ταυτόχρονη συμμετοχή σε περισσότερα vlans. Η διευθυνσιοδότηση γίνεται από dhcp υπηρεσίες. Συνολικά, οι δικτυακές υπηρεσίες εκτελούνται από το neutron api. [13]



Εικόνα 10 Λογικά επίπεδα πλατφόρμας OKEANOS [11]

Workflow έναρξης λειτουργίας νέου εικονικού μηχανήματος από το Okeanos

Η διαδικασία, κατά την οποία παραμετροποιούνται και εκκινούν τα VM, προϋποθέτει την συνεργασία όλων των υπηρεσιών την πλατφόρμας. Το λογισμικό που συντονίζει την διαδικασία είναι το synnefo σε συνδυασμό με τα εργαλεία του open stack και του ganeti. Οι παρακάτω αλληλουχίες βημάτων ακολουθούνται για την εκάστοτε υπηρεσία.



5. Συστήματα επικοινωνιών VOIP

Τα πρώτα συστήματα επικοινωνιών είχαν σαν στόχο την διάδοση φωνής σε μεγάλες αποστάσεις με αξιόπιστο και αποδοτικό τρόπο. Πολλά συστήματα εμφανίστηκαν, κατά την διάρκεια αυτής της εξέλιξης, με πιο διαδεδομένο το δίκτυο PSTN. Παράλληλα, με την εμφάνιση και εξάπλωση ψηφιακών πληροφοριακών συστημάτων και λόγω της διαφορετικής μορφής πληροφορίας που μεταφέρουν, αναπτύχθηκε ένα διαφορετικό σύστημα για τις ανάγκες αυτές, το TCP/IP. Όμως, η εξέλιξη δεν ήταν ισότροπη. Η εξάπλωση ψηφιακών συστημάτων και η αναπόσπαστη χρήση τους στις καθημερινές δραστηριότητες κάνει αναγκαία την ενοποίηση και αξιοποίηση όλων των λειτουργιών από ένα μοναδικό συστήματα που θα προσφέρει όλες τις υπηρεσίες. Στην λογική αυτή, τα συστήματα φωνής ενσωματώθηκαν στα ψηφιακά συστήματα, δημιουργώντας την δυνατότητα αποστολής πληροφορίας φωνής μέσω του TCP/IP δικτύου. Η τεχνολογία αυτή ονομάστηκε VOIP(Voice Over IP).

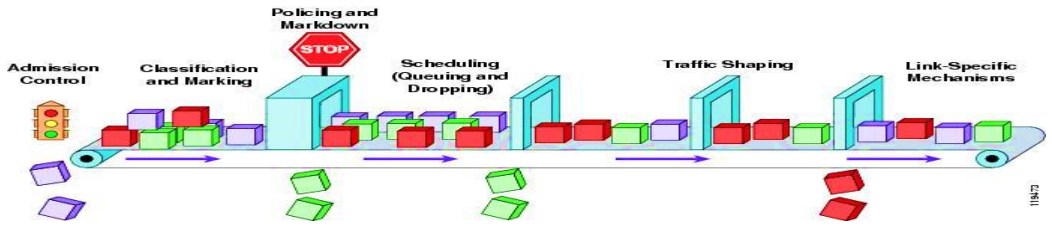
Η λογική των VOIP συστημάτων, δεν διαφέρει πολύ από αυτή ενός συμβατικού συστήματος PSTN. Συγκεκριμένα, η διαδικασία πραγματοποίησης μιας κλήσης σε ένα δίκτυο μεταγωγής κυκλώματος -όπως ονομάζεται- περιλαμβάνει το στάδιο της σηματοδοσίας μεταξύ των χρηστών, διαμέσου των υποδομών του δικτύου που έχει σαν στόχο να εγκαθιδρύσει και να τερματίσει την κλήση, όταν χρειαστεί, ενώ στο επόμενο στάδιο γίνεται η μεταφορά ψηφιοποιημένων δεδομένων (ομιλία , Φαξ κλπ.). Η ουσιαστική διαφορά μεταξύ των 2 συστημάτων είναι ο τρόπος που μεταφέρεται η πληροφορία, καθώς στον συμβατικό τρόπο δεσμεύεται μια διαδρομή μεταξύ των 2 χρηστών, ενώ σε ένα VOIP σύστημα η πληροφορία κομματιάζεται σε πακέτα διαδιδόμενα από το IP δίκτυο. Τεχνικά για να λειτουργήσει το VOIP, στηρίζεται στην αξιοποίηση πολλών διαφορετικών λειτουργιών του δικτύου TCP/IP. Έτσι, για το στάδιο της σηματοδοσίας αξιοποιείται το SIP πρωτόκολλο, ενώ για την μεταφορά δεδομένων το RTP, αφού έχουν μορφοποιηθεί από καταλλήλους codecs. [14]

5.1 Παράμετροι ποιότητας κλησεων

Για να θεωρηθεί επιτυχημένο ένα σύστημα φωνής VoIP πρέπει να αποδίδει την ιδιά ποιότητα κλήσης με αυτή ενός συμβατικού συστήματος. Έτσι, είναι απαραίτητη η επίτευξη υψηλής ποιότητας μετάδοσης. Σαν αποτέλεσμα, υπάρχει η ανάγκη -όπως για κάθε σύστημα real time- επίτευξης ελάχιστου delay και μεγάλου εύρους ζώνης. Επιπλέον, τα μεταδιδόμενα πακέτα να γίνονται drop σε ανεκτό ποσοστό. Τα ακόλουθα standard χρειάζεται να καλυφθούν σε κάθε περίπτωση.

- ✓ Delay με τιμές μικρότερες των 150 millisecond end to end, σύμφωνα με το πρότυπο ITU G.114. Για κλήσεις μέσω δορυφορικών ζεύξεων είναι αποδέκτες και τιμές 300 ms.
- ✓ Πάρα το ότι ιδανικά δεν είναι επιθυμητή η απώλεια πακέτων, όταν λειτουργεί ο codec G.729 packet loss της τάξεως 1% είναι ανεκτό.
- ✓ Η μεταβλητότητα του delay (jitter), δεν δημιουργεί επιπλέον μείωση της ποιότητας, όταν έχει τιμές μικρότερες των 100 ms.

Η ικανοποίηση των παραπάνω επίπεδων ποιότητας μπορεί να την εγγυηθεί το δίκτυο, δίνοντας προτεραιότητα στην κίνηση πακέτων που προέρχεται από την σηματοδοσία και τα πακέτα φωνής. Έτσι, για το αναγκαίο bandwidth, delay & jitter μπορούν να αξιοποιηθούν μηχανισμοί quality of service (QOs), ώστε να φέρνουν το δίκτυο σε προβλέψιμη κατάσταση με την υποστήριξη των παρακάτω τεχνικών. [15]



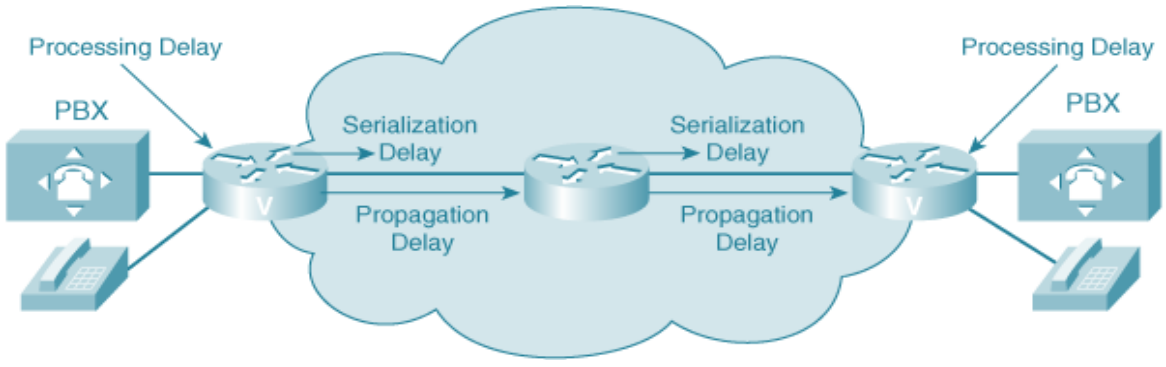
Εικόνα 11Σχηματική αναπαράσταση μηχανισμών Qos [15]

- Εξασφάλιση προκαθορισμένου bandwidth
- Βελτίωση συνθήκων που προκαλούν απώλεια πακέτων
- Αποφυγή συμφόρησης στο δίκτυο
- Shaping δικτυακού traffic
- Καθορισμός προτεραιοτήτων στο δίκτυο

Καθυστέρηση

Το μέγεθος της καθυστέρησης ορίζεται ως ο χρόνος που απαιτείται για την πληροφορία να διανύσει την απόσταση μεταξύ των 2 χρηστών. Πρόκειται για ένα μέγεθος, οπού σημαντική επίδραση έχουν οι μηχανισμοί δρομολόγησης και γενικότερα το φορτίο του δικτύου. Ανάλογα με το στάδιο και την αίτια που εισάγουν καθυστέρηση υπάρχουν 3 περιπτώσεις:

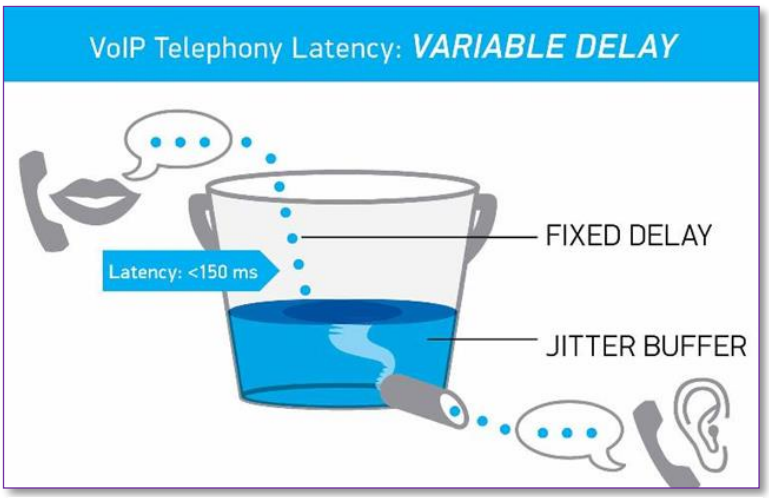
- Καθυστέρηση διάδοσης: ο χρόνος που διαρκεί η εισαγωγή του πακέτου πληροφορίας στο μέσο διάδοσης, καθώς και ο χρόνος διάδοσης που οφείλεται στην απόσταση των 2 χρηστών.
- Καθυστέρηση επεξεργασίας : προκαλείται στα 2 ακραία σημεία της επικοινωνίας, οπού η φωνή από αναλογικό σήμα ψηφιοποιείται και ενθυλακώνεται σε πακέτα πληροφορίας προς μετάδοση. Αντίστοιχα, μόλις το πακέτο φτάσει στον δεκτή μεταφράζεται πάλι σε αναλογικό σήμα φωνής. Φορέας αυτής της καθυστέρησης είναι ο χρόνος επεξεργασίας στα 2 άκρα, αλλά και ο χρόνος, οπού οι δικτυακές συσκευές διαβάζουν το πακέτο για να το δρομολογήσουν σωστά.
- Καθυστέρηση δρομολόγησης: κατά την διάρκεια δρομολόγησης πακέτων υπάρχει ο μηχανισμός των ουρών δρομολόγησης. Ανάλογα με το φορτίο κίνησης του δικτύου, ο χρόνος παραμονής ενός πακέτου σε τέτοιες ουρές μεταβάλλεται, εισάγοντας καθυστέρηση στον συνολικό χρόνο παράδοσης.



Εικόνα 12 Δικτυακές διεργασίες που εισάγουν delay [16]

Jitter

Η επικοινωνία 2 χρηστών με VOIP τεχνολογία πρέπει να αποκτήσει χαρακτηριστικά σχεδόν real time, ώστε να προσφέρει την αναμενόμενη εμπειρία στον χρήστη. Έτσι, λογικό είναι η καθυστέρηση, όπως ορίστηκε προηγουμένως, να παίζει πολύ σημαντικό ρολό. Όμως, η διάδοση πακέτων πολυμέσων πρέπει επιπλέον να γίνεται με σταθερό ρυθμό. Επιπρόσθετα ,λόγω της μεταβλητότητας στην κίνηση πακέτων στο δίκτυο μπορεί για κάποια χρονική στιγμή να προωθηθούν περισσότερα πακέτα φωνής στην μονάδα του χρόνου από ότι σε κάποια άλλη χρονική στιγμή, οπού θα υπάρχει μεγαλύτερο φορτίο στο δίκτυο. Το μέγεθος που μετράει την μεταβλητότητα του delay ονομάζεται jitter.



Εικόνα 13 Τεχνική ελάττωσης του jitter [25]

Η διαχείριση του παράγοντα jitter σε πρώτο στάδιο μπορεί να συμβεί από εξωτερικούς παράγοντες που προέρχονται από την διαχείριση που πραγματοποιείται μέσω των μηχανισμών QOS του δικτύου. Κάτι τέτοιο μπορεί να μην είναι αρκετό. Για τον λόγο αυτό, ένας επιπλέον μηχανισμός που εξαλείφει το jitter μπορεί να εφαρμοστεί με χρήση κατάλληλων buffers. Η λειτουργία τους βασίζεται στην αποθήκευση πακέτων φωνής και προώθησης τους με τέτοιο ρυθμό, ώστε να δημιουργηθεί ομαλή ροή των δεδομένων. Η εφαρμογή του μπορεί να γίνει τόσο στα στοιχεία του δικτύου (routers -switch) ή από την ιδιά εφαρμογή που τρέχει στους clients.

Ηχώ (echo)

Σε παλαιοτέρα ή χαμηλότερου κόστους τηλεφωνικό εξοπλισμό ενός παραδοσιακού τηλεφωνικού δικτύου PSTN εμφανιζόταν το φαινόμενο της ηχούς. Το αποτέλεσμα σε μια τηλεφωνική συνδιάλεξη είναι ο ομιλών να ακούει ετεροχρονισμένα την πληροφορία που μετέδωσε. Σαν συνέπεια, η εμπειρία του χρήστη ήταν ανεπαρκής, όσον αφορά την υπηρεσία που απολάμβανε. Πηγή του φαινομένου αυτού είναι ο δικτυακός εξοπλισμός, οπού σε διαφορετικά στάδια - κυρίως φίλτρα – μέρος του σήματος καθώς διαδίδεται ανακλάται και επιστρέφει στην πηγή. Η εξέλιξη των δικτυών κατάφερε να εξαλείψει την ηχώ ή τουλάχιστον να την κάνει μη ανιχνεύσιμη από τον χρήστη, αφού συμβαίνει σε πολύ μικρό χρονικό διάστημα. Έτσι ο χρήστης δεν είναι ικανός να το αντιληφθεί .

Το ίδιο πρόβλημα μπορεί να εμφανιστεί και σε ένα σύστημα VOIP. Ο καθοριστικός παράγοντας που το αναδεικνύει είναι η καθυστέρηση. Έτσι, σε μια συνοδό 2 χρηστών το delay της μεταξύ τους επικοινωνίας φέρνει τα ετεροχρονισμένα πακέτα που αναφέρθηκαν σε χρονικά διαστήματα ικανά να γίνουν αντιληπτά. [17]

Τεχνικές που εφαρμόζονται για την εξάλειψη είναι:

- Echo suppression: παρακολουθεί συνεχώς την κίνηση δεδομένων στο κανάλι φωνής και με την χρήση ενός voice activated switch δεν επιτρέπει την μεταφορά δεδομένων από κάποιον χρήστη μόλις ανιχνεύει ότι την συγκεκριμένη χρονική στιγμή εκείνος δεν μιλάει. Αντίστοιχα, επαναφέρει το κανάλι μόλις ο χρήστης μιλήσει και πάλι. Δεν αποτελεί αποδοτική περίπτωση, όταν και οι 2 χρήστες μιλάνε ταυτόχρονα.
- Echo cancellation: ο μηχανισμός αυτός χρησιμοποιεί μια μαθηματική φόρμουλα. Σε πρώτο στάδιο, υπολογίζει την ένταση του σήματος που θα δημιουργήσει ηχώ και στην συνέχεια την αφαιρεί από το τελικό σήμα.

5.2 Πρωτόκολλο σηματοδοσίας (SIP)

Η μεταφορά δεδομένων πολυμέσων διαφοροποιείται από την μεταφορά δεδομένων συμβατικών εφαρμογών. Η διαφορά βρίσκεται στην ανάγκη για έλεγχο με την δημιουργία κατάλληλης σηματοδοσίας, ώστε να συντονίζεται η συνέδρια μεταφοράς δεδομένων. Τον σκοπό αυτό επιτελεί το SIP πρωτόκολλο. Ορίζεται ως ένα Application layer protocol, κατά το RFC 3261, που έχει σαν στόχο να δημιουργεί, διαμορφώνει και να τερματίζει ένα session μεταξύ δυο τερματικών στο IP δίκτυο. Σχεδιάστηκε, ώστε η λειτουργία του να εφαρμόζει όλες τις διεργασίες σε χαμηλότερα layers. Το SIP ορίστηκε, ώστε να έχει απλή εφαρμογή χρησιμοποιώντας ένα μικρό αριθμό εντολών. Τα μηνύματα έχουν μορφή απλού κειμένου, κάνοντας τα σήματα του πρωτοκόλλου ευκολά αναγνώσιμα από απλά προγράμματα με μικρές απαιτήσεις σε πόρους.

Τα χαρακτηριστικά του είναι:

- Ακολουθεί την αρχιτεκτονική client – server
- Μπορεί να λειτουργήσει για 2 απλούς χρήστες (unicast) ή ακόμη και για περιπτώσεις συνεδριών πολλών χρηστών (multicast), όπως γίνεται σε εφαρμογές teleconference.
- Χρησιμοποιεί την φόρμα URL & URI του HTTP protocol, καθώς κάθε στοιχείο του δικτύου αποτελεί μέρος ενός Domain.
- Μπορεί να εφαρμόσει UDP και TCP στο επίπεδο μεταφοράς με προτίμηση το UDP στις real time εφαρμογές
- Μπορεί να αξιοποιηθεί και σε περιπτώσεις μεταφοράς αρχείων Online gaming ή instant messaging

Εμπλεκόμενα Δικτυακά στοιχεία

Η εγκαθίδρυση μιας κλήσης αναμεσά σε δυο μέρη προϋποθέτει την αξιοποίηση του πρωτοκόλλου SIP. Για να λειτουργήσει, είναι αναγκαίο να σταλθούν τα καταλληλά σήματα που θα δημιουργήσουν την συνεδρία, καθώς και την διαχείριση μηνυμάτων από server που θα συντονίζουν την σηματοδοσία αυτή.

Clients

Αρχικά, τα 2 βασικά στοιχεία που απαιτούνται είναι clients με εγκατεστημένη την κατάλληλη εφαρμογή softphone ή VoIP συσκευής. Όμως, το SIP είναι ένα πρωτόκολλο αρχιτεκτονικής client server. Έτσι ,κατά την δημιουργία μιας κλήσης τον ρολό του client παίζει το μέρος που προσπαθεί να καλέσει (User agent client -UAC-) και τον ρολό του server αυτός που δέχεται την κλήση. (User Agent Server -UAS-). Δηλαδή, οι ρολόι εναλλάσσονται ανάλογα με το ποιος ξεκινάει την διαδικασία.

Registrar server

Σε πρώτο στάδιο, μόλις ένα UAC συνδεθεί στο δίκτυο, για να γίνει διαθέσιμο θα πρέπει να κάνει register, ώστε να γίνει authenticated σε έναν ειδικό server. Εκεί αποθηκεύεται το URI και η IP του UAC στο domain.

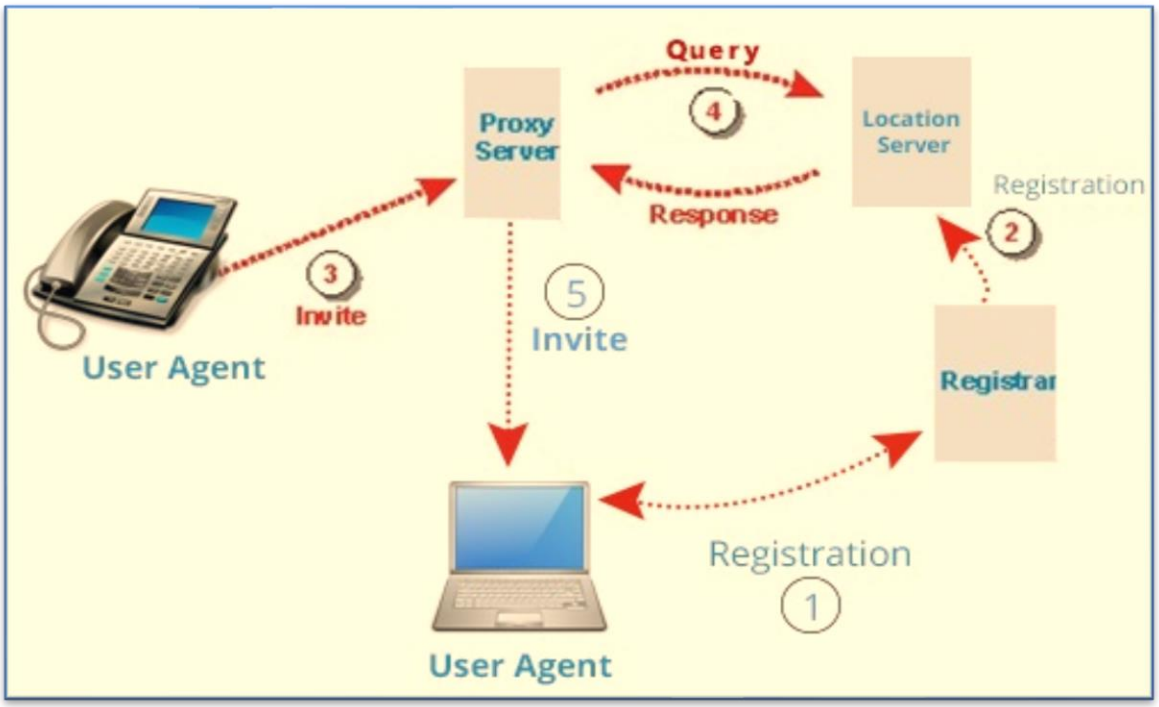
Proxy server

Βασικό και απαραίτητο στοιχείο του δικτύου αποτελεί ο proxy server. Είναι το πρώτο λειτουργικό μέρος, στο οποίο συνδέεται ένας UAC για να στείλει το αίτημα του για κλήση. Ο server αυτός λειτουργεί σαν router που σε συνεργασία με τους υπολοίπους servers, αναζητά την διαδρομή που πρέπει να ακολουθήσουν τα δεδομένα μιας κλήσης. Χωρίζονται σε 2 κατηγορίες:

- ✓ Stateless: προωθεί τα πακέτα, χωρίς να διατηρεί κανενός είδους πληροφορία για την κλήση.
- ✓ Stateful: διατηρεί αρχείο των αιτημάτων, για να βελτιώσει την αξιοπιστία της σηματοδοσίας κατά την κλήση, επαναπροωθώντας πακέτα σε περίπτωση που αυτά δεν παραδοθούν σωστά.

Location server

Μια συσκευή UAC μπορεί να βρίσκεται σε μια συγκεκριμένη θέση με δεδομένα στοιχεία IP διεύθυνσης και MAC address ή να αλλάζει δίκτυο ανάλογα με την κίνηση του χρήστη. Για την πραγματοποίηση μιας κλήσης, απαιτείται να είναι γνωστή η θέση (με ορούς δικτυακούς) του τελικού UAC. Έτσι, με το που συνδεθεί ο χρήστης, για να είναι διαθέσιμος να δεχτεί κλήσεις πρέπει πρώτα να ενημερώνει τον registrar server για την τοποθεσία του και στην συνέχεια ,αν λάβει αίτημα από κάποιον proxy, να αναζητά στις βάσεις δεδομένων που έχουν δημιουργηθεί από τους registrar server, για να μπορεί να γίνει ανιχνεύσιμος από την αναζήτηση άλλων χρηστών που θέλουν να πραγματοποιήσουν μια κλήση.



Εικόνα 14 Workflow σηματοδοσίας μεταξύ των servers σε VOIP συστήματα [19]

Κατηγορίες μηνυμάτων σηματοδοσίας

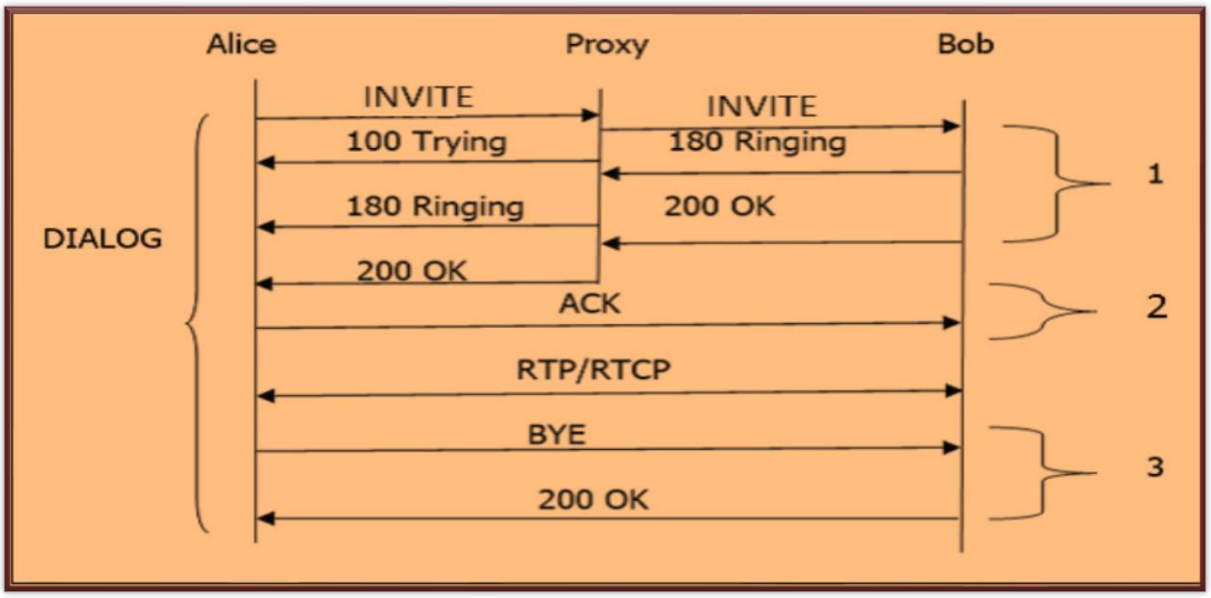
Όλα τα προηγούμενα δικτυακά στοιχεία συντονίζουν τις κλήσεις ανταλλάσσοντας πληροφορίες με προκαθορισμένα μηνύματα , όπως αυτά ορίζονται στο πρωτόκολλο SIP. Τα μηνύματα αυτά έχουν παρόμοια λογική με τα HTTP requests. Χωρίζονται σε 2 κατηγορίες. Στα μηνύματα ερωτήσεων και στα μηνύματα απαντήσεων. Τα μηνύματα ερωτήσεων (request methods) ζητούν από έναν Server να πραγματοποιήσει μια ενέργεια, ενώ τα μηνύματα απαντήσεων (response methods) επιβεβαιώνουν την πραγματοποίηση του αιτήματος ή σε αντίθετη περίπτωση επιστρέφουν κάποιο σφάλμα. Η περιγραφή των απαντήσεων γίνεται από έναν 3ψήφιο αριθμό, οπού το πρώτο ψηφίο δίνει πληροφορίες για το κατά ποσό έχει εκτελεστεί το αίτημα. Συνολικά τα μνήματα είναι: [18]

- Request methods
  - Invite: περιέχει πληροφορίες για τον αποδεκτή της κλήσης
  - BYE: τερματίζει ένα εγκαθιδρυμένο session.
  - REGISTER: ενημερώνει τον location server με τις απαραίτητες πληροφορίες.
- Response methods
  - 1xx: επιβεβαίωση επιτυχούς εκτέλεσης ενός έγκυρου ενδιάμεσου αιτήματος.
  - 2xx-200: επιβεβαιώνει την επιτυχή εκτέλεση ενός request. Για την περίπτωση INVITE request επιβεβαιώνει την έναρξη κλήσης.
  - 3xx: για να ολοκληρωθεί το αίτημα είναι απαραίτητη η δρομολόγηση του σε διαφορετικό σημείο στο δίκτυο.
  - 4xx: το μήνυμα που στάλθηκε δεν είχε σωστή σύνταξη ή δεν μπόρεσε να ολοκληρωθεί από τον server.
  - 5xx: ο server δεν μπόρεσε να ολοκληρώσει το request χωρίς όμως αυτό να περιέχει κάποιο σφάλμα.
  - 6xx: το αίτημα ήταν αδύνατο να ολοκληρωθεί σε κανέναν από τους servers.

Βασική σηματοδοσία εκτέλεσης κλήσεων

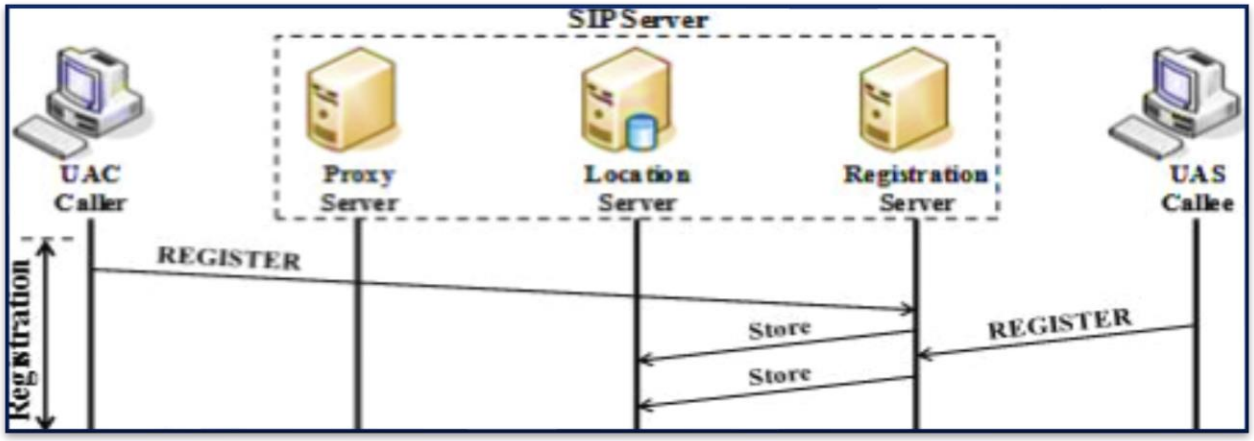
Η ροή μηνυμάτων ακολουθεί μια προκαθορισμένη αλληλουχία ερωταπαντήσεων μεταξύ UAC & UAS. Στην συνέχεια, παρουσιάζεται η σηματοδοσία για την πραγματοποίηση κλήσης μεταξύ δυο χρηστών με την διαμεσολάβηση ενός proxy server. Ένα τέτοιο μοντέλο αποτελεί την πιο απλή περίπτωση. Παρ όλα αυτά, αν διατίθεται ένα εκτεταμένο δίκτυο χρηστών, οπού για παράδειγμα, θα έχουμε περισσοτέρους proxy servers τα βήματα είναι το ιδιά με την διαφορά ότι επαναλαμβάνονται τα βήματα μεταξύ των proxies. Τα μηνύματα που θα αναφερθούν στο παράδειγμα θεωρούν την κλήση επιτυχημένη, οπότε δεν θα γίνουν αναφορές για τα μηνύματα σφάλματος. Ακόμα και σε αυτή την περίπτωση, η σηματοδοσία θα έληγε με αποστολή από τον proxy ή τον UAS απάντησης σφάλματος, όπως αναφέρθηκε στις κατηγορίες μηνυμάτων με επιτυχημένη έκβαση.

Παράδειγμα: ο χρήστης Alice θέλει να επικοινωνήσει με τον bob μέσω του proxy server.



Εικόνα 15 Ανταλλαγή μηνυμάτων σηματοδοσίας κατά την πραγματοποίηση κλήσεων [19]

i. Η Alice στέλνει Invite msg στον proxy με τα στοιχεία του Bob.
ii. Ο proxy στέλνει απάντηση, ότι αναζητά τον χρήστη Bob. Αυτό συμβαίνει, έτσι ώστε να ενημερωθεί η allice ότι έλαβε το msg από τον proxy και να μην κάνει επαναποστολη, θεωρώντας ότι αυτό δεν έφτασε.
iii. Ο proxy προωθεί το invite msg στον Bob.
iv. Αφού φτάσει το αίτημα στον Bob, μηνύματα επιβεβαίωσης στέλνονται προς την allice μέσω του proxy και ξεκινάει το session μεταξύ allice & bob.
v. Μόλις ολοκληρωθεί η συνομιλία, τα 2 μέρη ανταλλάσσουν μηνύματα τερματισμού του session.



Εικόνα 16 Σηματοδοσία Register νέου χρήστη στο VOIP δίκτυο [14]

Η παραπάνω λειτουργικότητα μπορεί να πραγματοποιηθεί με δεδομένο ότι όλη η απαιτούμενη προετοιμασία για τα μέρη του δικτύου έχει πραγματοποιηθεί. Η προετοιμασία περιλαμβάνει το authentication του χρήστη στον register server και στην συνέχεια η ενημέρωση του Location server για την θέση του UAC. Με αυτό τον τρόπο, ο proxy θα βρει όλες τις απαραίτητες πληροφορίες για να εντοπίσει τους χρήστες και στην συνέχεια να δημιουργηθεί το επιθυμητό session. Στην παραπάνω εικόνα, φαίνεται η σηματοδοσία που παράγεται για τον σκοπό αυτό. Τα σχετικά μηνύματα πάντα περιλαμβάνουν και ack.

5.3 Πρωτόκολλο real time εφαρμογών (RTP)

Το στάνταρ που καθορίζει τον τρόπο, με τον οποίο πολυμεσικές εφαρμογές αποστέλλουν δεδομένα, είναι το πρωτόκολλο RTP της σουίτας TCP/IP, με προτυποποίηση, όπως ορίζεται από το RFC 1889 by IETF's. Η μεταφορά πακέτων ακολουθεί μια διαδικασία, όπως ορίζει το πρωτόκολλο, ενώ κατά την εφαρμογή του εμπλέκει και άλλα πρωτόκολλα του επιπέδου μεταφοράς και κυρίως του UDP. Πιο ειδικά τα βήματα είναι τα εξής:

- Το SIP σε πρώτο στάδιο έχει δημιουργήσει ένα session, στο οποίο οι 2 χρήστες έχουν "συμφωνήσει" σε ποιές διευθύνσεις μιλάνε και σε ποιά port θα δέχονται δεδομένα.
- Η ψηφιοποιημένη πληροφορία, μόλις διαμορφωθεί κατάλληλα από τον εκάστοτε codec, χωρίζεται σε πακέτα, ώστε να γίνει η μεταφορά τους. Το ρόλο αυτό αναλαμβάνει το UDP και το Network layer.
- Έλεγχος της ποιότητας και των χαρακτηριστικών της κλήσης, μέσω των ανταλλασσόμενων πληροφοριών στα header των πακέτων.

Ο ειδικός στόχος που έχει το RTP σε σχέση με άλλα πρωτόκολλα μεταφοράς πακέτων είναι η παροχή υπηρεσιών real time. Για να πραγματοποιηθεί ο στόχος αυτός, πρέπει να επιτευχθούν υψηλά στάνταρ QOS. Για τον λόγο αυτό, κάθε πακέτο ανταλλάσσει ορισμένες πληροφορίες αξιοποιήσιμες από αντίστοιχους μηχανισμούς. Οι πληροφορίες αυτές περιλαμβάνουν.

i. Συνεχής αρίθμηση πακέτων (Sequence number) με στόχο τον εντοπισμό πιθανού packet loss
ii. Χρονική σήμανση πακέτων για τον καλύτερο συγχρονισμό μιας κλήσης.
iii. Πληροφορίες σχετικές με τις μορφές κωδικοποίησης η συμπίεση που ακολουθήθηκαν από το πρόγραμμα του αποστολέα

RTCP

Στην κατεύθυνση αυτή μια επέκταση του RTP είναι το RTCP (IETF's RFC 3550). Το επιπλέον στοιχείο που διαθέτει είναι η ενθυλάκωση στο πακέτο feedback στατιστικών στοιχείων για την σύνδεση, καθώς και μηνυμάτων ελέγχου. Στόχος είναι οι παραπάνω πληροφορίες να δώσουν δεδομένα στους μηχανισμούς QOS, ώστε να βελτιωθεί η ποιότητα της κλήσης ή και ακόμα να χρησιμοποιηθούν σαν κριτήριο επιλογής ενός διαφορετικού codec, που θα εξομαλύνει την ροή πακέτων. Τα πακέτα του πρωτοκόλλου RTCP δεν μεταφέρουν δεδομένα των Multimedia εφαρμογών, αλλά για τον σκοπό αυτό λειτουργούν σε συνεργασία με το RTP. Συνολικά, οι επιπλέον πληροφορίες που παρέχει είναι:

- ✓ Packet loss
- ✓ Μεταβλητότητα καθυστέρησης (jitter)
- ✓ αριθμός μεταδιδόμενων bytes ανά πακέτο
- ✓ σειριακή αρίθμηση πακέτων
- ✓ Round trip delay

Το RTCP είναι ένα ευέλικτο πρωτόκολλο, όπου επιτρέπει την δημιουργία ειδικών πακέτων ελέγχου, που ορίζονται από τον σχεδιαστή του εκάστοτε συστήματος. Παρ' όλα αυτά, ορίζονται κάποια βασικά πακέτα που ανταλλάσσονται μεταξύ των μερών. Πρόκειται για τα:

- ✓ SR(sender report): πακέτο ελέγχου που βοηθάει κυρίως τον συγχρονισμό πακέτων video &audio.
- ✓ RR(receiver report): δίνει στοιχεία QOS

5.4 Πρωτόκολλο διασύνδεσης PBXs (IAX)

Το IAX είναι ένα πρωτόκολλο επικοινωνιών παρόμοιο με το SIP. Αναπτύχθηκε από την Digium με στόχο να συντονίσει την επικοινωνία μεταξύ 2 και περισσότερων pbxs . Επιπλέον, η χρήση του είναι εκτεταμένη και σε διαφορετικούς κατασκευαστές softswitch , PBX και softphone. Χρησιμοποιείται σε περιπτώσεις, όπου είναι επιθυμητό να περάσουν πολλαπλές σύνοδοι επικοινωνιών VOIP πάνω από ένα κανάλι επικοινωνίας. Με τον τρόπο αυτό, εξοικονομείται εύρος ζώνης, αν συγκριθεί με το εύρος ζώνης, που θα χρειαζόταν το ίδιο πλήθος συνοδών να περάσουν με σηματοδοσία και συντονισμό από το sip πρωτόκολλο. [20]

Αν και ο αρχικός σχεδιασμός του ήταν η μεταφορά φωνής μπορεί με εξίσου αποδοτικό τρόπο να λειτουργήσει με multimedia εφαρμογές. Είναι ένα μεικτό πρωτόκολλο που ταυτόχρονα αναλαμβάνει την σηματοδοσία και την μεταφορά δεδομένων ταυτόχρονα. Ο τρόπος μεταφοράς πακέτων ονομάζεται trunnking, καθώς σε ένα ενιαίο πακέτο με κοινό Header περνάει ευρύτερο πλήθος συνεδριών. Έτσι, προτιμάται, κατά τον σχεδιασμό core επιπέδου διασύνδεσης pbx και όχι τόσο ανέμεσα σε 2 softphone clients.

Η διαδικασία του authenticate πραγματοποιείται με:

- RSA keys
- Αρχείο απλού Text
- MD5 hashing

Θέματα ασφαλείας αποτελούν σημεία, όπου παρουσιάζει το IAX ορισμένες αδυναμίες. Τα πακέτα δεδομένων δεν είναι κρυπτογραφημένα. Μια τεχνική που θα μπορούσε να φανεί αξιόπιστη είναι η χρήση vpn. Αφού οι βασικές εφαρμογές είναι σε συνδέσεις μεταξύ PBX σταθερών IPs, η λύση του VPN είναι εφικτή. Στην περίπτωση του απλού χρήστη ενός softphone, όπου αλλάζει συνεχώς IP, θα είναι δυσκολότερη η παραμετροποίηση αντίστοιχων vpn clients.

6. Λογισμικό ανοικτού κώδικα Asterisk

Με την προηγούμενη συνολική αποτύπωση end to end επικοινωνίας με χρήση VOIP τεχνολογιών, έγινε σαφές, ότι πρόκειται για μια σύνθετη δομή που εμπλέκει, τόσο δικτυακές τεχνολογίες και πρωτοκολλά, όσο και ειδικούς Server για τον καθορισμό και καθοδήγηση των λειτουργιών. Το έργο αυτό επιτελούν ειδικοί server που ονομάζονται PBXs. Ειδικότερα, στα πλαίσια της εργασίας θα χρησιμοποιηθούν οι δυνατότητες του open source λογισμικού ***ASTERISK***. Η εγκατάσταση του γίνεται σε υλικό συμβατικής αρχιτεκτονικής που διαθέτει εγκατεστημένη μια διανομή Linux. Εκεί, καταχωρούνται όλες οι VoIP συσκευές που εμπλέκονται στο δίκτυο και με κατάλληλη παραμετροποίηση παρέχει άρτιες υπηρεσίες μετάδοσης πολυμέσων.

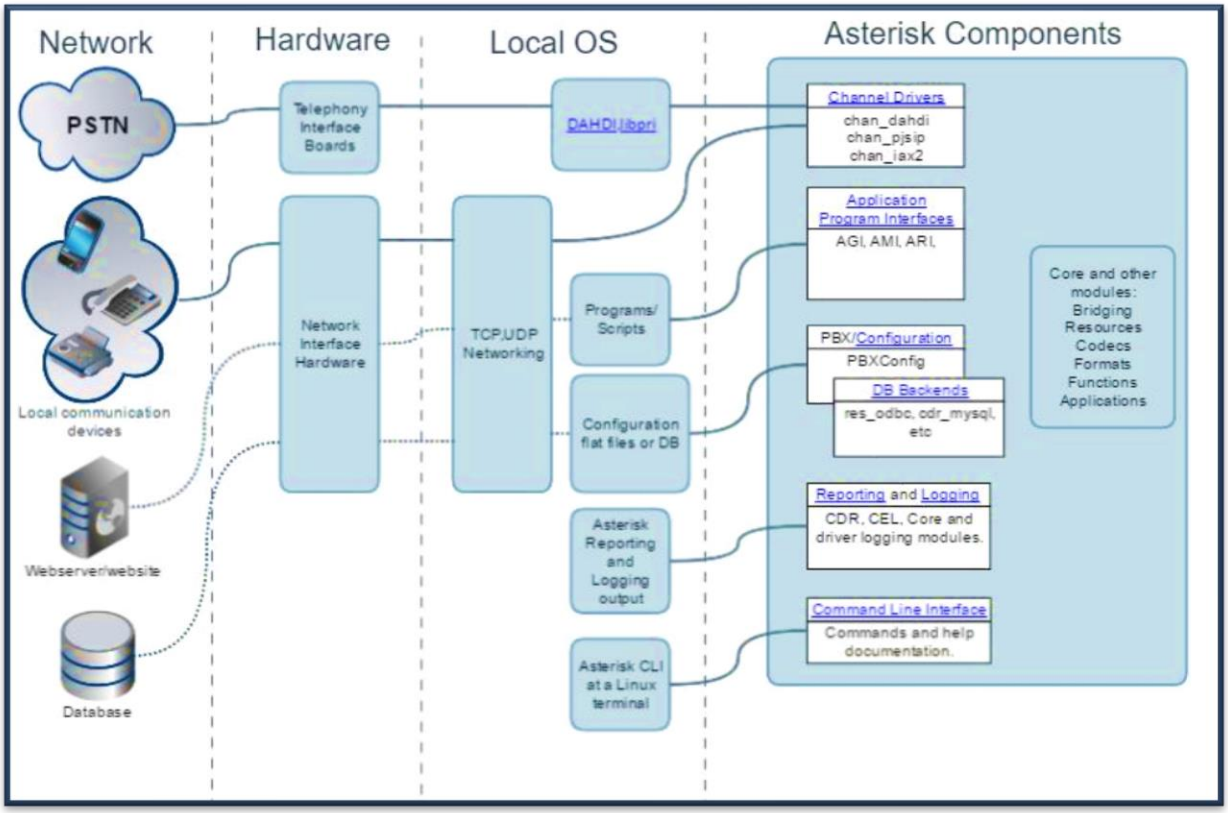
Οι δυνατότητες του asterisk είναι πολλές. Υποστηρίζει το μεγαλύτερο εύρος VOIP IP πρωτοκόλλων (SIP , IAX ,H323) και παράλληλα διαχειρίζεται όλες τις αναγκαίες υπηρεσίες (registrar , location server etc.). Σημαντική είναι και η ικανότητα του να διασυνδέει ένα δίκτυο VOIP με το συμβατικό δίκτυο PSTN, μέσω κατάλληλων interfaces και gateways. Η γλώσσα, στην οποία έχει γραφεί ο κυρίως κώδικας, είναι η C, ενώ διαθέτει εξατομικευμένους κανόνες υλοποίησης script για την παραμετροποίηση του συστήματος. Οι δυνατότητες του διευρύνονται από το ειδικό gateway (AGI). Το AGI δίνει την δυνατότητα σε προγραμματιστές να προσθέσουν επιπλέον υπηρεσίες σε όποια γλώσσα προγραμματισμού επιθυμούν, καθώς και να αλληλοεπιδράσουν με το σύστημα από το συγκεκριμένο interface. Ο σκοπός του Asterisk δεν περιορίζεται στην διαχείριση κλήσεων. Είναι ένα πρόγραμμα που προσφέρει ολοκληρωμένες λύσεις και υπηρεσίες χρήσιμες σε ένα σύγχρονο τηλεφωνικό δίκτυο.

- Voice mail
- Conference κλήσεις
- Ανακατεύθυνση κλήσεων με υπηρεσία IVR
- Αυτοματοποιημένες απαντήσεις φωνής

6.1 Αρχιτεκτονική του Asterisk

Ο Asterisk, παρα την απλή λογική που έχει, είναι ένα σύνθετο πρόγραμμα με πολλαπλές συσχετίσεις ανάμεσα στα μέρη που το αποτελούν. Το γεγονός ότι αναπόσπαστο μέρος της χρησιμότητας του είναι η διασύνδεση με άλλες συσκευές ή ακόμα και με εξωτερικούς αποθηκευτικούς χώρους ή βάσεις δεδομένων, κάνει ακόμα πιο πολύπλοκη την δομή του. Επιπλέον, η modular αρχιτεκτονική δίνει την δυνατότητα στον χρήστη να εξατομικεύσει την εγκατάσταση που θα πραγματοποιήσει ανάλογα με τις ανάγκες. Υπάρχουν 3 βασικά στοιχεία απαραίτητα σε κάθε εγκατάσταση. [21]

- Core
- Channels
- Bridges



Εικόνα 17 Διασύνδεση λειτουργικών μονάδων του asterisk και αλληλεπίδραση με εξωτερικά interfaces [20]

Πυρήνας (Core)

Ο πυρήνας αποτελεί τον ακρογωνιαίο λίθο του προγράμματος. Είναι το βασικό component, από το οποίο αντλείται όλη η λειτουργικότητα της υπηρεσίας, αλλά και των Modules. Εκτός από την διαχείριση του βασικού backend, ο πυρήνας είναι και το μέσο, από το οποίο συντονίζονται και περνάνε όλες οι κλήσεις. Ειδικότερα, σκοπός του πυρήνα είναι να δρομολογήσει κλήσεις, με βάση το dial plan, που του έχει ενσωματωθεί από το σχετικό configuration file. Dial plan είναι το σύνολο από κανόνες που πρέπει να ακολουθήσει το πρόγραμμα στην περίπτωση που δεχτεί κλήση από κάποιον client με αντίστοιχο αριθμό, ή όπως ονομάζεται extension. Υπάρχει σε μορφή Text file συνήθως με την ονομασία extensions.conf. Μπορεί να βρεθεί, είτε τοπικά, είτε να ανακτηθεί από κάποιο εξωτερικό πρόγραμμα ή database. Εκτός από το dial plan, ο πυρήνας φορτώνει τα Modules.

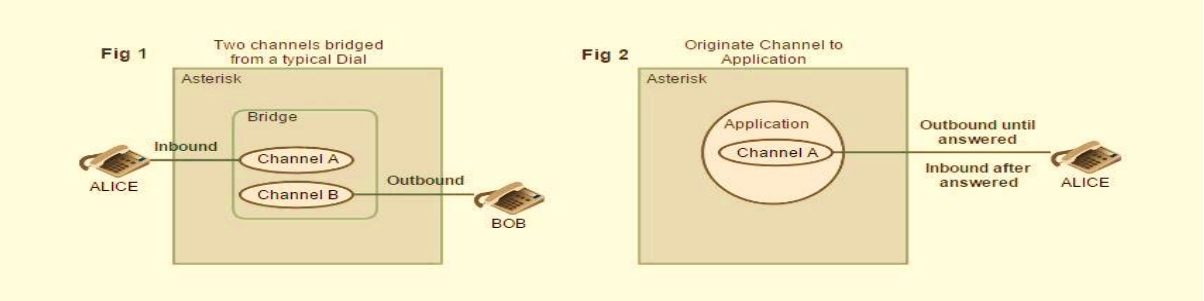
Τα λογικά στοιχεία του πυρήνα είναι τα ακόλουθα:

- **PBX switch core**: Αντικείμενο υπεύθυνο για την υλοποίηση της κυρίως λειτουργίας του asterisk, δηλαδή την λήψη και μεταγωγή κλήσεων στον σωστό παραλήπτη.
- **Application luncher:** Ξεκινάει την λειτουργία ειδικών εφαρμογών που προσφέρει η πλατφόρμα για την ικανοποίηση προχωρημένων υπηρεσιών, όπως είναι ο αυτόματος τηλεφωνητής το IVR κλπ.
- **I/O Manager:** Αναλαμβάνει τον χρονοπρογραμματισμό και τον συντονισμό των εργασιών που εκτελεί το πρόγραμμα.
- **Codec Translator:** Ενσωματώνει τους κατάλληλους codecs στα channels, ώστε τα μεταδιδόμενα δεδομένα να μεταφράζονται σε «γλώσσα» αναγνωρίσιμη απο τον πυρήνα και στην συνέχεια μέσω αυτού να εφαρμοστούν, τόσο οι βασικές, όσο και οι προχωρημένες λειτουργίες.

Channels

Η πραγματοποίηση οποιασδήποτε κλήσης μέσω του asterisk προϋποθέτει την ύπαρξη και εγκατάσταση κατάλληλου δίαυλου επικοινωνίας αναμεσά στους χρήστες. Μόλις δημιουργηθούν οι δίαυλοι αυτοί «channels», είναι διαθέσιμοι να αλληλοεπιδράσουν και με άλλες εφαρμογές ή λειτουργίες του asterisk. Είναι προσβασιμο από τον πυρήνα, ώστε να πραγματοποιήσει αλληλεπίδραση με το dial plan και σε συνέχεια να κάνει μεταγωγή «bridge» με κάποιο άλλο κανάλι. Ακόμη, μπορεί να γίνει μορφοποίηση των ιδιοτήτων του ή της κατάστασης του από εφαρμογές ή και να διαμορφωθεί καταλληλά με στόχο να διοχετεύσει πληροφορίες φωνής, μηνυμάτων αλλά και βίντεο. Για την παραμετροποίηση τους υπάρχουν καθορισμένα directories, οπού όλες οι οδηγίες και πληροφορίες, είναι αποτυπωμένες σε απλά text αρχεία. Για μεγαλύτερη ευελιξία και διευκόλυνση μπορεί στις νεότερες εκδόσεις του λογισμικού να ανακτηθούν στοιχεία παραμετροποιήσεις από βάσεις δεδομένων, είτε τοπικά στον ίδιο server είτε απομακρυσμένα. Στα πλαίσια της εργασίας θα γίνει χρήση μόνο των SIP channels για τους clients και των IAX για την διασύνδεση των servers.

Η διαδικασία ορισμού ενός καναλιού στο αντίστοιχο configuration file δεν το καθιστά άμεσα λειτουργικό. Για να μπορεί ένα Channel να μεταδώσει δεδομένα θα πρέπει πρώτα να έχει γίνει originate από κάποιο module ή τον core. Επιπλέον, τα κανάλια μπορούν να διαχωριστούν σε 2 κατηγορίες ανάλογα με την κατεύθυνση που έχει το originate process. Στα *εισερχόμενα*, τα οποία δημιουργούνται από τον client, ο οποίος θέλει να πραγματοποιήσει κλήση και στα *εξερχόμενα* που γίνονται σε δεύτερο χρόνο από το λογισμικό, για να εγκαθιδρύσει την ζητούμενη από τον client κλήση ή να επικοινωνήσει κάποιο application module με client.



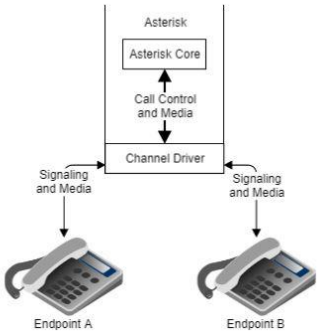
Εικόνα 18 Βασικοί τρόποι διασύνδεσης clients και του asterisk μέσω των channels [21]

Bridges

Η εγκαθίδρυση channels δεν είναι ικανή από μονή της να κάνει 2 χρήστες να επικοινωνήσουν. Η λειτουργικότητα ολοκληρώνεται με την βοήθεια των bridges. Ο ρόλος τους είναι να «γεφυρώσουν» τα channels και να επιτρέψουν την μεταφορά πολυμέσων. Σε ειδικές περιπτώσεις, Bridge μπορεί να δημιουργήσει και ο core με κάποιο Channel με σκοπό να λειτουργήσει κάποιο application. Σε πιο πρόσφατες εκδόσεις του asterisk, δίνεται η δυνατότητα κράτησης μιας κλήσης σε ουρά μέχρι ο λήπτης να είναι διαθέσιμος. Και σε αυτό το σενάριο μεσολαβεί η λειτουργία του bridge. Σε τελικό στάδιο, μόλις κάποιο μέρος σταματήσει την κλήση, ή σε περίπτωση ομαδικών κλήσεων αποχωρήσει ο διαχειριστής το bridge, παύει να υπάρχει και σταματάει η μετάδοση δεδομένων.

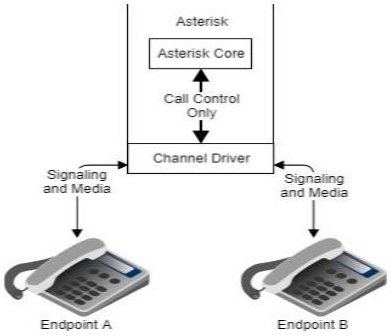
Ο Asterisk διαθέτει ένα εύρος ξεχωριστών τύπων Bridge. Σε γενικές γραμμές ,όμως, ο διαχωρισμός γίνεται σε 2 βασικές κατηγορίες, στην κατηγορία two party και στην multiparty.

- **Two party:** Είναι το Bridge που συνδέει 2 χρήστες με δυνατότητες βελτιστοποίησης, ανάλογα με τον τύπο καναλιού που συνδέουν. Σε αυτή την κατηγορία, υπάρχουν 3 υποκατηγορίες.

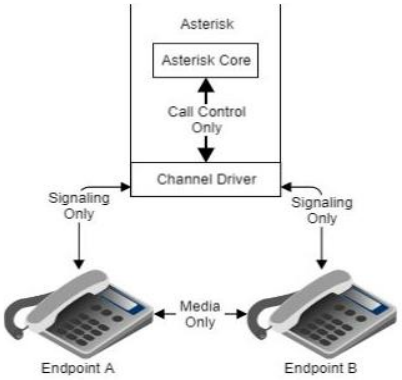


i. **Core:** Δίνει την δυνατότητα διασύνδεσης καναλιών διαφορετικού τύπου. Τα δεδομένα μετατρέπονται σε μια κοινή μορφή, αναγνωρίσιμη από τον asterisk. Με αυτό τον τρόπο, δεν λειτουργεί απλά ως δρομολογητής κλήσεων, αλλά μπορεί και να επιδράσει στα δεδομένα που μεταφέρει με εφαρμογές, όπως voice recording. Προτιμάται σε περιπτώσεις, οπού η παραμετροποίηση του δικτυού δεν είναι ικανοποιητική. Το μειονέκτημα που παρουσιάζει η λύση αυτή η καθυστέρηση που εισάγει, καθώς εισέρχεται Process delay.

ii. **Native:** Συνηθισμένη περίπτωση οπού συναντάται σε εφαρμογές με ίδιου τύπου channels. Η μεταφορά δεδομένων συνεχίζει να γίνεται μέσω του Asterisk. Διαφορά με την core περίπτωση είναι ότι η συμμετοχή του πυρήνα στην μεταφορά δεδομένων γίνεται μόνο για τον έλεγχο και την εποπτεία της κλήση ή για τον τερματισμό του bridge. Μειονέκτημα σε μια τέτοια περίπτωση είναι η αδυναμία του asterisk να επιδρά στην μεταφορά των πολυμέσων



iii. **Remote:** Στο συγκεκριμένο σχεδιασμό, το λογισμικού λειτουργεί με μοναδικό σκοπό την εγκαθίδρυση της κλήσης. Μόλις αυτή πραγματοποιηθεί, η διαδρομή που θα ακολουθείται ορίζεται από το δίκτυο. Είναι μια αρχιτεκτονική αποκεντρωμένη, καθώς δεν περνάει αποκλειστικά από τον asterisk η λειτουργικότητα της κλήσης, αλλά και από τις δικτυακές υποδομές. Ένα τέτοιο σχήμα δίνει δυνατότητα για πιο ευέλικτο σύστημα με περισσότερες δυνατότητες βελτιστοποίησης. Μειονέκτημα αποτελεί η ανάγκη για επιπλέον φόρτο παραμετροποίησης.



- **Multi party:** Τα bridges αυτού του τύπου μπορούν να αλληλοεπιδράσουν με ένα ή και περισσότερα channels και να δρομολογήσουν κλήσεις σε αυτά. Μπορούν να θεωρηθούν ως μια επέκταση του two party core bridge, οπού τα διαδιδόμενα πολυμέσα συνενώνονται και προωθούνται στα κανάλια που συμμετέχουν στην συνέδρια. Να σημειωθεί ότι το σύνολο των δυνατοτήτων για τα two party μπορεί να εφαρμοστεί και σε αυτή την περίπτωση, ανάλογα με τον τρόπο χρήσης που θα προτιμηθεί. [22]

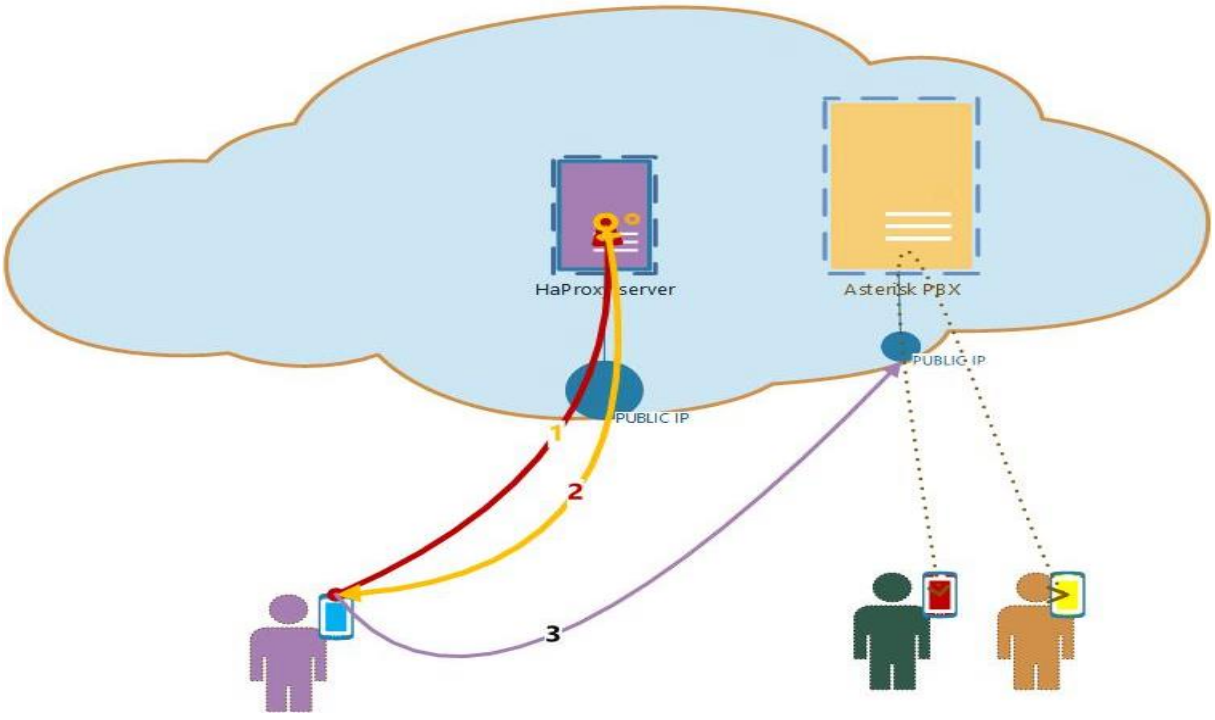
7. Ανάπτυξη VOIP υπηρεσιών & αυτοματοποιημένων διεργασιών load balancing σε Iaas cloud computing περιβάλλον

Κάθε περιβάλλον νεφουπολογιστικής μπορεί να αποδώσει εξίσου καλά τις ίδιες υπηρεσίες με ένα συμβατικού τύπου υπολογιστικό σύστημα, προσθέτοντας και τα επιπλέον πλεονεκτήματα που προσφέρονται από την εκάστοτε υποδομή. Στην προσπάθεια ανάδειξης των σχετικών πλεονεκτημάτων, αναπτύχθηκε στα πλαίσια της εργασίας σύστημα που στοχεύει στην παροχή αδιάλειπτων υπηρεσιών VoIP τηλεφωνίας. Η επιλογή VoIP υπηρεσιών έγινε, ώστε να αποδειχθεί, πως απαιτητικές υπηρεσίες, όπως είναι οι πολυμεσικές μπορούν να εξασφαλίσουν τα απαραίτητα επίπεδα QoS & QoE. Επιπλέον, ο κεντρικοποιημένος σχεδιασμός επιτρέπει την διαθεσιμότητα της υπηρεσίας από οποιοδήποτε σημείο προσφέρει σύνδεση στο διαδίκτυο. Στο τελικό στάδιο, μετά την επιτυχή απόδοση της υπηρεσίας, δίνεται η δυνατότητα με αυτοματοποιημένες διεργασίες να αξιοποιηθούν τα APIs της cloud υποδομής, έτσι ώστε να διευρυνθούν οι υπολογιστικές δυνατότητες, όταν αυτό καταστεί αναγκαίο.

Για τις ανάγκες υλοποίησης του παραπάνω μοντέλου, αρχικά, έγινε ο σχετικός σχεδιασμός και σενάριο εφαρμογής, ώστε να εξασφαλιστούν οι αναγκαίοι πόροι. Η πλατφόρμα που χρησιμοποιήθηκε είναι το **okeanos**. Η διαδικασία απόδοσης πόρων περιλαμβάνει το αίτημα στην ομάδα διαχείρισης με αντίστοιχη τεκμηρίωση του σκοπού, για τον οποίο ζητούνται οι πόροι. Εφόσον το αίτημα εγκριθεί στο account που δημιουργείται αυτόματα μέσω της ακαδημαϊκή ιδιότητας, συνδέεται το project με τους αποδιδόμενους πόρους. Η λειτουργία των clients , έγινε μέσω του λογισμικού **Zoiper** από το google store για android συσκευές. Η μόνη παραμετροποίηση που απαιτήθηκε, ήταν να εισάγουμε τα στοιχεία του server και του user account «με βάση τον σχεδιασμό της εργασίας σαν server IP, εισήχθη η Public IP του load balancer»

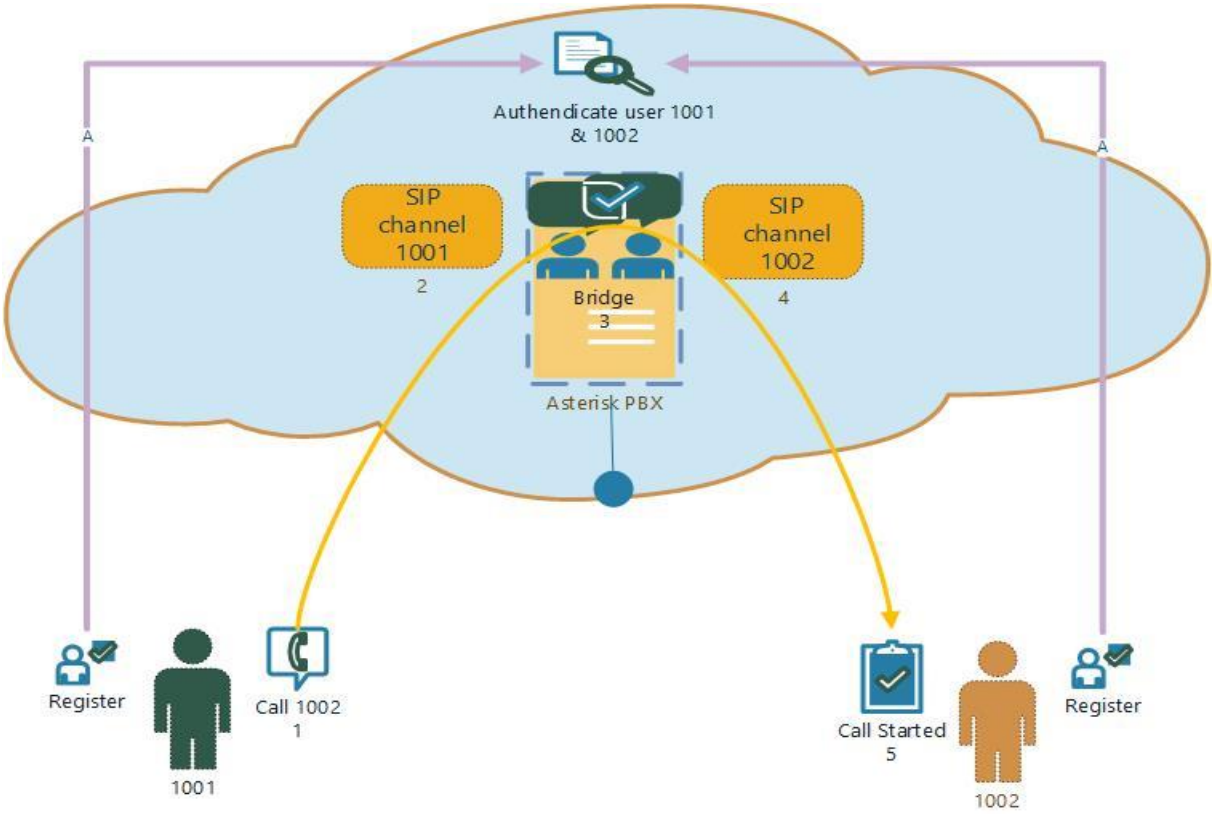
- Αρχική Τοπολογία και λειτουργικότητα του Συστήματος

Το σενάριο εφαρμογής, αρχικά, περιλαμβάνει έναν Virtual Server με εγκατεστημένο τον asterisk. Ο εν λόγω server έχει συνδεθεί και με ένα Public interface. Ένας επιπλέον ubuntu server με εγκατεστημένο το πρόγραμμα **HaProxy** και με ξεχωριστή public ip θα ενεργοποιηθεί, παίζοντας τον ρολό του controller. Τα softphones των clients θα έχουν ρυθμιστεί στην public ip του controller. Στο αρχικό cluster κάθε κλήση ή αίτημα για σύνδεση στο PBX, θα πηγαίνει πρώτα στον controller και στην συνέχεια θα παραπέμπεται από το HaProxy στον Asterisk. Σε αυτό το σημείο, δηλαδή, δεν θα τρέχει κάποιος αλγόριθμος **load balancing,** αλλά το HaProxy θα λειτουργεί ως δείκτης για την Ip του μοναδικού PBX server σε λειτουργία. Μετα την καθοδήγηση του controller η επικοινωνία server – client θα γίνεται από το public interface του Asterisk. Μόλις περισσότεροι χρήστες συνδεθούν με τα παραπάνω βήματα και ολοκληρωθεί η καθοδήγηση τους στο PBX οι επικοινωνίες πλέον θα γίνονται αποκλειστικά μέσω της ip του Asterisk.



Εικόνα 19 Δικτυακό διάγραμμα αρχικού συστήματος

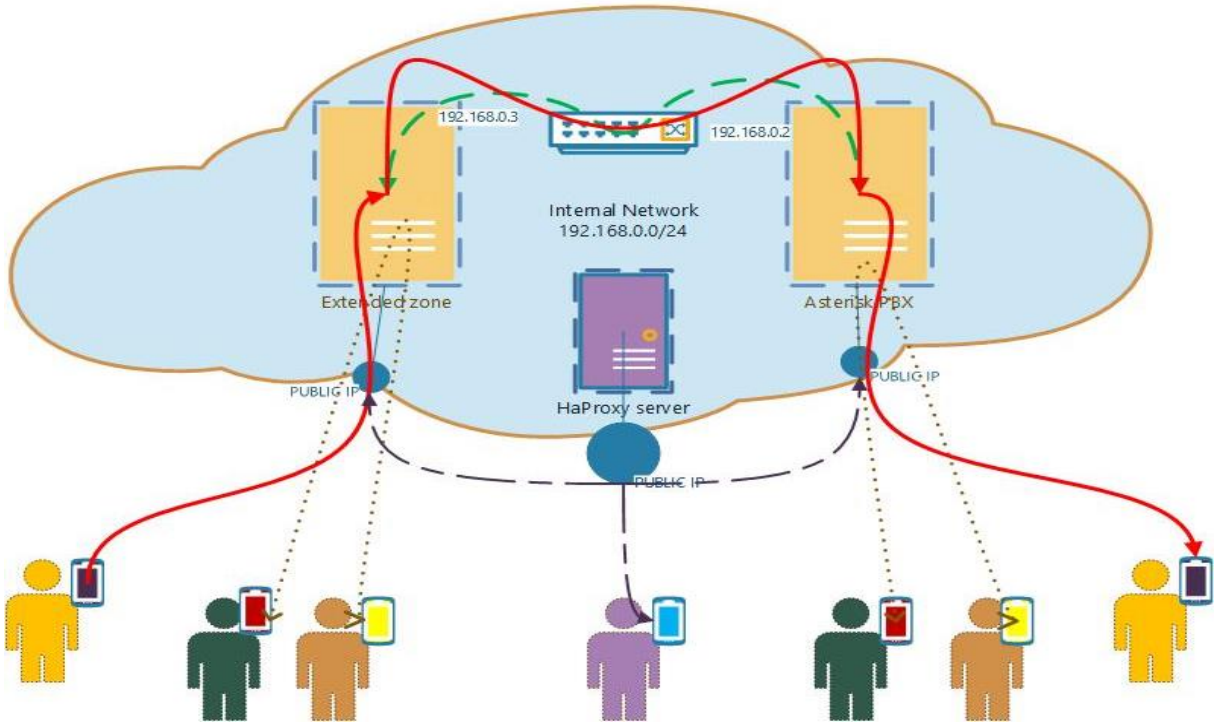
Η προηγούμενη δικτυακή ανακατεύθυνση των κλήσεων δεν είναι ικανή από μόνη της να δώσει υπηρεσίες επικοινωνίας. Ζωτικής σημασίας είναι και τα processes που εμπλέκονται από τους servers. Τα βήματα εκτέλεσης των processes έχουν την εξής σειρά. Το σύστημα, αρχικά, αναλαμβάνει το register των χρηστών. Οι χρήστες έχουν ορισθεί, κατά την παραμετροποίηση των sip channels και περιλαμβάνουν πληροφορίες για το extension ,username, αλλά και το password. Με αυτόν τον τρόπο, ο server έχει αναγνωρίσει την ταυτότητα των χρηστών και έχει καταχωρήσει την δικτυακή τους θέση, έτσι ώστε να είναι ανιχνεύσιμα «εάν χρειαστεί» να προωθήσει ή να δεχθεί κλήση. Μόλις ο Server δεχτεί αίτημα για κλήση, το πρώτο που συμβαίνει είναι να ελέγξει το αρχείο **extensions.conf.** Εκεί βρίσκονται όλες οι οδηγίες, για τον τρόπο δρομολόγησης. Οι παράμετροι έχουν ορισθεί με τέτοιο τρόπο, στο αρχικό set up, όπου έχουμε έναν server ενεργό, ώστε οι ενέργειες που πραγματοποιεί το Pbx είναι το origination του καναλιού και το bridge με το κανάλι του χρήστη που καλεί.



Εικόνα 20 Λειτουργικό διάγραμμα αρχικού συστήματος

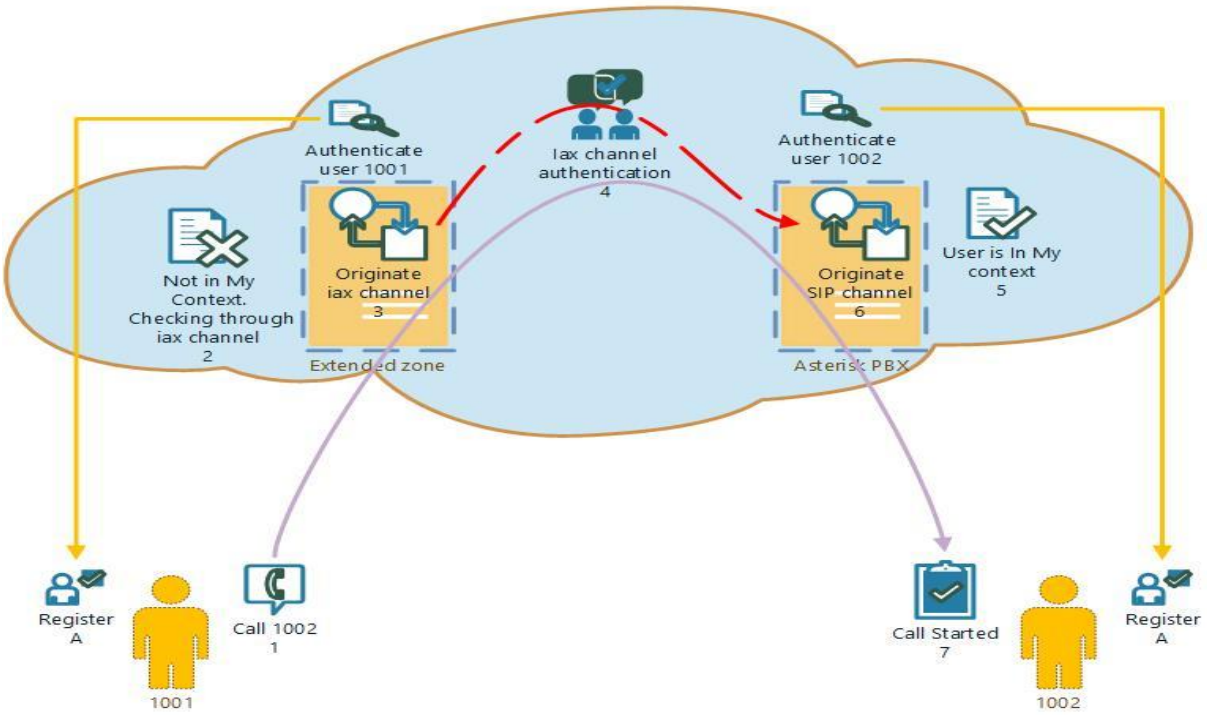
- Τοπολογία και λειτουργικότητα συστήματος μετά την διεύρυνση του.

Το προηγούμενο setup αποτελεί ένα πλήρες σύστημα επικοινωνιών VOIP που υποστηρίζεται από μια cloud Υποδομή. Σε περίπτωση που χρειαστεί να μετάβουμε στο νέο σύστημα, το οποίο θα διαθέτει περισσότερες δυνατότητες, εκτελούνται οι παρακάτω εργασίες. ένας αυτοματοποιημένος έλεγχος παρατηρεί, αν κάποια κριτήρια πληρούνται και στην συνέχεια με το **Python** πρόγραμμα που βρίσκεται αναλυτικά στο παράρτημα αποδίδει τελικώς ένα Ολοκληρωμένο σύστημα που αποτελείται πλέον από 2 servers με 2 Public IPs. Οι 2 αυτοί νέοι server εξυπηρετούν διαφορετικούς λογαριασμούς χρηστών. Η κατεύθυνση για το που θα συνδεθεί τελικώς ένας χρήστης γίνεται από τον **HAproxy,** ο οποίος με βάση τον αλγόριθμο **Load balancing** που του έχει οριστεί, ανακατευθύνει αντίστοιχα τους χρήστες. Να σημειωθεί, ότι πλέον δεν λειτουργεί απλά ως δείκτης, όπως στο προηγούμενο setup. Για να είναι ,όμως, το σύστημα ενιαίο, πρέπει οι δυο νέοι server να μπορούν να επικοινωνήσουν και κάποιος χρήστης από τον 1 server να μπορεί να επικοινωνήσει με κάποιον από τον 2. Για τον σκοπό αυτό, γίνεται η σύνδεση των 2 server και από ένα εσωτερικό **private δίκτυο** σε διαφορετικό network interface. Η επικοινωνία αυτή θα λειτουργήσει ως trunk σύνδεση με χρήση του πρωτοκόλλου IAX.



Εικόνα 21 Δικτυακό διάγραμμα διευρυμένου συστήματος

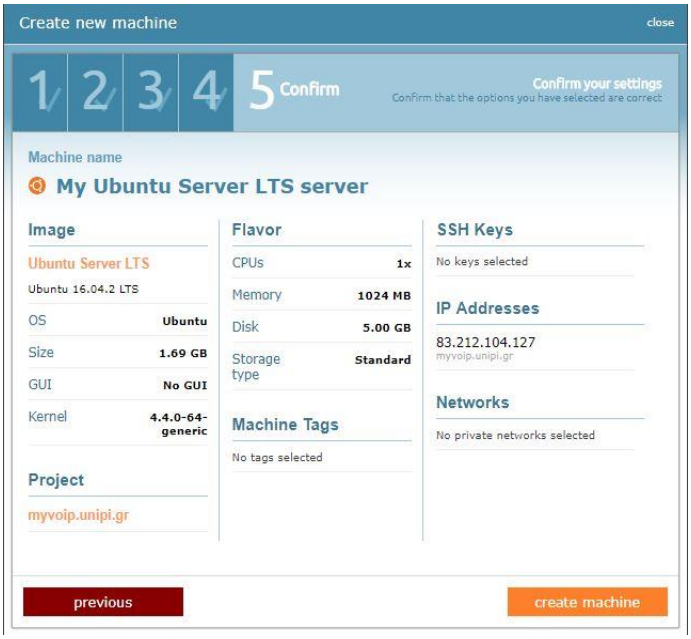
Η δομή που υλοποιείται στο νέο εκτεταμένο σύστημα είναι παρόμοια σε λειτουργικότητα με εκείνο ενός μεμονωμένου server με ορισμένες όμως διαφορές. Η πρώτη βασική διαφορά παρουσιάζεται στην θέση, όπου έχει γίνει register κάποιος χρήστης. Πλέον, ο load balancer έχει υπό την εποπτεία του και τους 2 server. Έτσι, μόλις λάβει κάποιο αίτημα για σύνδεση το δρομολογεί εναλλάξ. Επειδή τα στοιχεία χρηστών είναι κοινά και στα 2 pbxs, εξασφαλίζουμε ότι ο χρήστης πάντα θα γίνει register, είτε στον ένα, είτε στον άλλον server. Η δεύτερη διαφορά σχετίζεται με την διαδικασία κλήσης. Ένα αίτημα κλήσης, πλέον, δεν είναι δεδομένο ότι θα καλυφθεί από τον ίδιο server. Στην περίπτωση, όπου ένας client του server 1, θέλει να επικοινωνήσει με έναν client του server 2 γίνονται τα εξής βήματα. Αρχικά, το pbx ,όπου είναι συνδεδεμένος ο καλών, ελέγχει τοπικά, μήπως το extension βρίσκεται κάτω από το δικό του dialplan. Αν δεν το εντοπίσει, θεωρεί, ότι ο καλών βρίσκεται στο δεύτερο Pbx, οπότε μέσω iax σύνδεσης στέλνει την κλήση σε αυτόν. Εκεί, αναλαμβάνει ο δεύτερος server να ελέγξει το δικό του dial plan, ώστε να εντοπίσει τον χρήστη και αν τον βρει να κάνει originate ένα sip κανάλι δημιουργώντας και τα αντίστοιχα bridges. Έτσι, μια τέτοια επικοινωνία θα περιλαμβάνει ένα sip κανάλι για κάθε χρήστη και ένα iax ανάμεσα στα 2 Pbxs.



Εικόνα 22 Λειτουργικό διάγραμμα διευρυμένου συστήματος

7.1 Δημιουργία & εγκατάσταση εξατομικευμένου Image

Η γρήγορη και χωρίς παρέμβαση χρήστη εγκατάσταση του νέου cluster προϋποθέτει την ύπαρξη άμεσης διαθεσιμότητας ενός συστήματος με εγκατεστημένο τον asterisk. Για τον σκοπό αυτό, θα αξιοποιηθεί η δυνατότητα της πλατφόρμας okeanos, δημιουργίας εξατομικευμένου Image, το οποίο θα ανακτάται από την υπηρεσία Plankton, μέσω του κώδικα, μόλις αυτό ζητηθεί. Παρακάτω περιγράφεται η διαδικασία.



✓ Αρχικά, εκκινείτε ένα σύστημα με εγκατεστημένο το λειτουργικό Ubuntu server από τους διαθεσίμους πόρους του account που έχει αποδοθεί. Το ποσό της ram ,CPU & volume δεν είναι δεσμευτικό για τα μετέπειτα συστήματα που θα εγκατασταθούν. Δηλαδή, μπορεί ένα επόμενο σύστημα που θα συσχετιστεί με αυτό το Image να έχει αλλά χαρακτηριστικά hardware.

✓ Με την αλλαγή του status σε running, με έναν ssh client, όπως για παράδειγμα, το Putty μπαίνουμε remote στο σύστημα, για να πραγματοποιήσουμε την εγκατάσταση του asterisk. Από τις πολλές δυνατότητες που δίνει ο asterisk για εγκατάσταση, θα επιλεχθεί η εγκατάσταση από source code. Η δυνατότητα αυτή επιτρέπει την επιλογή των modules που θα εγκατασταθούν. Επειδή το μοντέλο της εργασίας αφορά αποκλειστικά επικοινωνίες με VoIP τα Modules LibPRI, pjProject & DAHDI δεν θα εγκατασταθούν, καθώς χρησιμεύουν για την διασύνδεση του VoIP δικτύου με το Pstn.

Εγκατάσταση asterisk server

Η διαδικασία εγκατάστασης του Asterisk γίνεται ευκολά με την εκτέλεση bash scripts και εντολών. Πρώτα, κατεβάζουμε την επιθυμητή έκδοση (κατά την υλοποίηση της εργασίας χρησιμοποιήθηκε η έκδοση Asterisk-14.6.0). Αφού ληφθεί και αποσυμπιεστεί το αρχείο, γίνονται τα παρακάτω βήματα.

- ✓ Εκτέλεση του **./configure** από τον φάκελο λήψης. Το script ελέγχει κατά ποσό το σύστημα έχει εγκατεστημένα όλα τα απαραίτητα περιφερειακά προγράμματα που χρησιμοποιούνται από τον asterisk.
- ✓ Κατά πασά πιθανότητα κάποια προγράμματα θα λείπουν. Έτσι το **./install_prereq install-unpackaged** θα εντοπίσει τα repositories και θα τα εγκαταστήσει αυτόματα
- ✓ Για να εξασφαλιστεί, ότι όλα τα dependencies είναι εγκατεστημένα, εκτελείται πάλι το **./configure** και αν όλα είναι σωστά εμφανίζεται ένα μήνυμα, ότι μπορούμε να συνεχίσουμε στην εγκατάσταση
- ✓ Τέλος, πριν ξεκινήσει η εγκατάσταση, γίνεται compile ο κώδικας με την εντολή **make** και στην συνέχεια με την εντολή **make install** εγκαθίσταται το πρόγραμμα στο σύστημα μας.

Επειδή η παραμετροποίηση του συστήματος είναι αρκετά συνθέτη, για να δοθεί η δυνατότητα σε έναν νέο χρήστη να μπορέσει βήμα βήμα να κατανοήσει και να εξατομικεύσει στις ανάγκες του, το πρόγραμμα δίνει ένα βασικό πακέτο με αρχεία παραμετροποίησης.

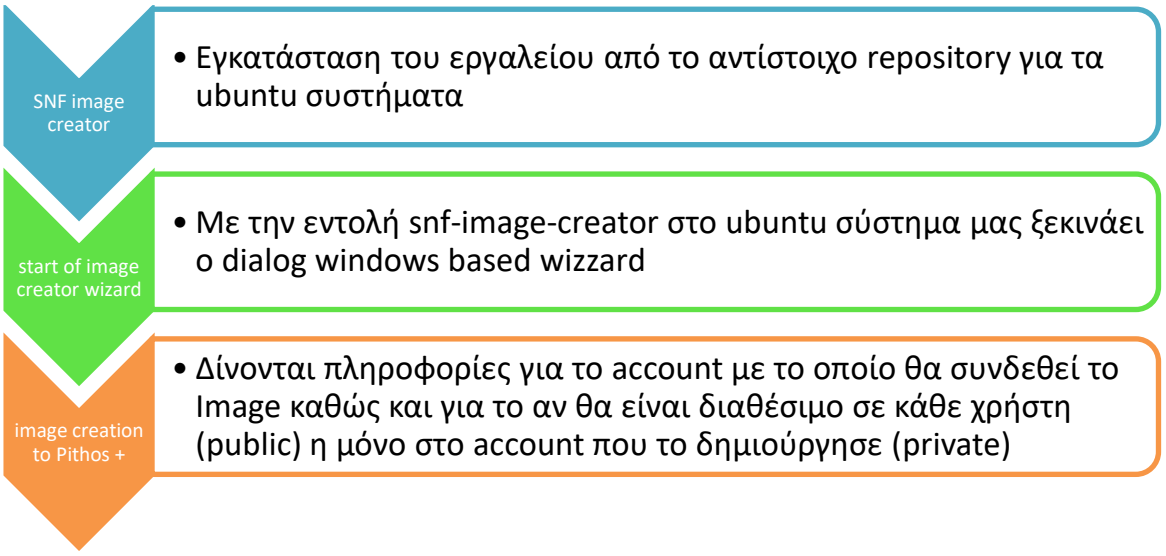
- ✓ Με την εντολή **make samples** φτιάχνονται αυτόματα αρχεία με πληροφορίες απαραίτητες για ένα βασικό configuration.

```
user@snf-761179:/etc/asterisk$ ls
acl.conf                cdr_manager.conf        cli_permissions.conf    festival.conf       muted.conf              res_fax.conf            sorcery.conf
adsi.conf               cdr_mysql.conf          codecs.conf             followme.conf       ooh323.conf             res_ldap.conf           ss7.timers
agents.conf             cdr_odbc.conf           confbridge.conf         func_odbc.conf      osp.conf                res_odbc.conf           stasis.conf
alarmreceiver.conf      cdr_pgsql.conf          config_test.conf        hep.conf            oss.conf                resolver_unbound.conf   statsd.conf
alsa.conf               cdr_sqlite3_custom.conf console.conf            http.conf           phone.conf              res_parking.conf        telcordia-1.adsi
amd.conf                cdr_syslog.conf         dbsep.conf              iax.conf            phoneprov.conf          res_pgsql.conf          test_sorcery.conf
app_mysql.conf          cdr_tds.conf            dnsmgr.conf             iaxprov.conf        pjproject.conf          res_pktccops.conf       udptl.conf
app_skel.conf           cel.conf                dsp.conf                indications.conf    pjsip.conf              res_snmp.conf           unistim.conf
ari.conf                cel_custom.conf         dundi.conf              logger.conf         pjsip_notify.conf       res_stun_monitor.conf   users.conf
ast_debug_tools.conf    cel_odbc.conf           enum.conf               manager.conf        pjsip_wizard.conf       rtp.conf                voicemail.conf
asterisk.adsi           cel_pgsql.conf          extconfig.conf          meetme.conf         queuerules.conf         say.conf                vpb.conf
asterisk.conf           cel_sqlite3_custom.conf extensions.ael          mgcp.conf           queues.conf             sip.conf                xmpp.conf
calendar.conf           cel_tds.conf            extensions.conf         minivm.conf         res_config_mysql.conf   sip.conf.sample
ccss.conf               chan_dahdi.conf         extensions.lua          misdn.conf          res_config_sqlite3.conf sip_notify.conf
cdr_adaptive_odbc.conf  chan_mobile.conf        extensions_minivm.conf  modules.conf        res_config_sqlite.conf  skinny.conf
cdr.conf                cli_aliases.conf        extensions.sample       motif.conf          res_corosync.conf       sla.conf
cdr_custom.conf         cli.conf                features.conf           musiconhold.conf    res_curl.conf           smdi.conf
user@snf-761179:/etc/asterisk$
```

Εικόνα 23 Βασικά αρχεία παραμετροποίησης από την εντολή Make samples

Καταχώρηση image στο okeanos

Για να ολοκληρωθεί η εγκατάσταση και η καταχώρηση του εξατομικευμένου image στο Okeanos, θα γίνει χρήση του εργαλείου **snf-image-creator**. Το εργαλείο αυτό αλληλεπιδρά με το λογισμικό διαχείρισης της πλατφόρμας synnefo. Στην πραγματικότητα, το εργαλείο αυτό παίρνει ένα στιγμιότυπο από το volume του VM και το μετατρέπει σε Image. Έτσι, διευρύνει το χαρακτηριστικό του snapshotting, όχι απλά ως μια δυνατότητα failover,r αλλά ως μια πηγή παραγωγής images διαθέσιμα σε περισσότερα συστήματα. Με την εκτέλεση των παρακάτω βημάτων, θα επιτευχθεί η διαθεσιμότητα λειτουργικού συστήματος με προ εγκατεστημένο τον Asterisk και των κατάλληλων configuration files. Το όφελος θα προκύψει, καθώς δεν θα πρέπει σε κάθε νέο VM να εγκαθίσταται ο Asterisk κάθε φορά σε έναν νέο server, ενώ ταυτόχρονα βοηθάει στην αυτοματοποίηση εγκατάστασης του νέου ειδικού server μέσω των **Kamaki API** εργαλείων.



7.2 Παραμετροποίηση του Asterisk server

Οι δυνατότητες που δίνει ο Asterisk για την ανάπτυξη PBXs, είναι πολλές με αρκετές φορές πολύπλοκη παραμετροποίηση και διαφορετική φιλοσοφία για το κάθε ξεχωριστό set up. Έτσι, για τις ανάγκες της εργασίας η ανάπτυξη ενός ενιαίου συστήματος, του οποίου η λειτουργικότητα δεν θα εξαρτάται από την επιμέρους παραμετροποίηση κάθε ξεχωριστού server που θα το συνθέτουν, θα επιλεχθεί παραμετροποίηση τέτοια που να επιτρέπει την εύκολη, άμεση και αυτοματοποιημένη ενσωμάτωση νέων server, χωρίς την ανάγκη παρέμβασης του administrator για νέο configuration. Η παραμετροποίηση που θα περιγράφει στην συνέχεια θα πραγματοποιηθεί την χρονική στιγμή προετοιμασίας του εξατομικευμένου image από το **snf image creator,** έτσι ώστε να αποτυπωθούν στο snapshot και στην συνέχεια να καταχωρηθούν μαζί με το Image. Έτσι, όταν συνδεθεί ο επιπλέον server στο cluster, να είναι πλήρως λειτουργικός. Τα 3 σημεία που θα παραμετροποιηθούν στον asterisk αφορούν:

- Αρχείο sip.conf

Σε αυτό είναι αποτυπωμένα τα στοιχεία για τον κάθε ξεχωριστό client. Με τα στοιχεία αυτά και το password που είναι καταχωρημένο με το λεκτικό secret, ο χρήστης γίνεται authenticate ως προς το **sip account** από τον Asterisk. Για τις ανάγκες συμβατότητας με τα υπόλοιπα λειτουργικά μέρη του cluster έχει οριστεί και το transport protocol για τις εν λόγω συνδέσεις σε tcp. Στο αρχικό σκέλος με την ονομασία **[general]** καθορίζεται και πάλι το tcp ως πρωτόκολλο μεταφοράς, ενώ επιπλέον ενημερώνει τον Asterisk ,ώστε να δέχεται αιτήματα registration κάποιου sip client από κάθε Ip και όχι σε συγκεκριμένο gateway. Στο επόμενο ακριβώς σκέλος έχουν οριστεί κάποιες default τιμές για το ένα και μοναδικό context που θα χρησιμοποιηθεί από το cluster, με την ονομασία **[admin]**. Τα context είναι η δυνατότητα του asterisk να διαφοροποιεί σε ξεχωριστές ομάδες λογικά τμήματα με ξεχωριστές πολιτικές ασφάλειας, παραμετροποιήσεις ή και λειτουργείες, έτσι ώστε να επιτρέπει στο dial plan τον διαχωρισμό των εργασιών σε πολλαπλά επίπεδα.

- Αρχείο iax.conf

Ο σχεδιασμός της εργασίας απαιτεί η επικοινωνία μεταξύ των 2 Server του cluster να γίνεται, όχι από public interfaces, αλλά μέσω του Internal δικτύου στο subnet **192.168.0.0/24**. Το πρωτόκολλο για την παραπάνω επικοινωνία θα είναι το **IAX**. Ο λόγος που επιλέχθηκε είναι η δυνατότητα ευκολότερης διάδοσης δεδομένων, αξιόπιστα και με μεγαλύτερη ταχύτητα από ότι το SIP. Το IAX αποσκοπεί στο να διευκολύνει την trunk σύνδεση στους 2 servers, ενώ ο περιορισμός του στο εσωτερικό και μόνον δίκτυο εξασφαλίζει και θέματα ασφάλειας. Στο αντίστοιχο αρχείο παραμετροποίησης της παραπάνω λειτουργίας **(iax.conf)** περιέχονται τα εξής τμήματα. Ένα context για τον ορισμό των καναλιών (Όπως αντίστοιχα με το sip) . Στο context αυτό θα υπάρχουν στοιχεία authentication, ενώ θα δείχνουν και το context, οπού θα ενεργήσει εσωτερικά, το όποιο flow έρθει μέσω του συγκεκριμένου καναλιού. Το πιο σημαντικό κομμάτι βρίσκεται στο **[general]** section, καθώς με την εντολή **register** γίνεται καταχώρηση της θέσης του πρώτου server στον δεύτερο και αντίστροφα. Η διαφορά στα αρχεία του πρώτου και του δευτέρου server είναι οι ονομασίες τους και η ip, οπού θα γίνει το lookup. Τα εσωτερικά context έχουν κοινή ονομασία, όπως στο κοινό αρχείο sip.conf.

- Αρχείο extensions.conf

Το τελευταίο και σημαντικότερο αρχείο είναι το extensions.conf, «διαβάζεται» από το dialplan του core, έτσι ώστε να λάβει της οδηγίες ,για το πως θα διαχειριστεί τα αιτήματα κλήσεων των extensions. Μια κλήση μπορεί να δρομολογηθεί άμεσα, να επιδράσουν πάνω της applications ή και ακόμα σε περίπτωση αποτυχίας να δοθούν νέες κατευθυντήριες για τον τρόπο διαχείρισης της. Όλα τα παραπάνω ορίζονται σε αυτό το αρχείο. Η λογική που επιλέχθηκε για το cluster περιλαμβάνει 2 βήματα. Το πρώτο κάνει έλεγχο, ώστε να αναζητήσει, αν υπάρχει ο λήπτης της κλήσης στον δεύτερο server. Στην περίπτωση, όπου είτε ο δεύτερος server δεν είναι σε λειτουργία ,είτε ο load balancer δεν τον έχει δρομολογήσει, ώστε να γίνει register, εκτελείται το επόμενο βήμα, το οποίο αναζητά στα τοπικά sip account τον χρήστη.

Το σύνολο των παραπάνω αρχείων βρίσκεται αναλυτικά στο παράρτημα, ενώ στο σύστημα βρίσκονται κάτω από το directory **/etc/asterisk**.

7.3 Crontab Job

Για την μετάβαση στο νέο σύστημα, που θα παρέχει πλήρη λειτουργικότητα, πρέπει να εκτελεστούν όλες οι απαραίτητες εντολές προς την πλατφόρμα για την εγκατάσταση και παραμετροποίηση του νέου συστήματος. Πριν συμβεί, όμως, κάτι τέτοιο, πρέπει να γίνει ο σχετικός έλεγχος με βάση καθορισμένα κριτήρια, ώστε να εκκινήσει η εκτέλεση του κώδικα. Συγκεκριμένα, στο σύστημα μας θεωρούμε, ότι το στοιχείο που θα οδηγήσει στην εγκατάσταση του επιπλέον server asterisk είναι η δικτυακή κίνηση από το δικτυακό interface του αρχικού server . Δηλαδή, αφού υπολογιστεί το φορτίο πακέτων, ορίζετε ένα όριο πάνω από το οποίο θα επιστρατευθούν επιπλέον εικονικά συστήματα και πόροι για να μην μειωθεί η ποιότητα κλήσεων. Την αυτοματοποιημένη διεργασία εποπτείας της κίνησης θα την εκτελέσει το εργαλείο των λειτουργικών συστημάτων Linux που ονομάζεται crontab.

Ο crontab είναι ένα process που εκτελείται από Linux λειτουργικά συστήματα με στόχο την εκτέλεση προγραμμάτων σε καθορισμένες χρονικές στιγμές. Στο πρόγραμμα που υλοποιήθηκε στα πλαίσια της εργασίας ορίστηκε να «τρέχει» από τον crontab, ένα συγκεκριμένο python script με το όνομα ***trigger***, κάθε μέρα το μεσημέρι. Η επιλογή της συγκεκριμένης περιόδου είναι τυχαία και ορίστηκε στην λογική, ότι εκείνες τις ώρες υπάρχει μεγαλύτερη πιθανότητα αύξησης του φορτιού κλήσεων. Στην παραμετροποίηση το * σημαίνει για κάθε τιμή, αλλιώς όταν ορίζεται με αριθμούς, εκτελείται όποτε ορίζει η τιμή του συγκεκριμένου πεδίου.

```
0   12   *  *   *      python trigger
-   -   -  -   -
|   |   |  |   +----- day of week (0 - 6) (Sunday=0)
|   |   |  +------- month (1 - 12)
|   |   +---------- day of month (1 - 31)
|   +-------------- hour (0 - 23)
+------------------ min (0 - 59)
```

Το trigger script παίρνει 2 στιγμιότυπα από τα στατιστικά του network interface με την βιβλιοθήκη ***psutil*** και υπολογίζει το throughput. Αν βρει τιμή μεγαλύτερη από ένα όριο και ταυτόχρονα ο αριθμός των ενεργών Server είναι 1, τότε ξεκινάει να φτιάχνει το νέο cluster με το επόμενο script που καλεί.

7.4 Cluster creator

Οι υπηρεσίες της πλατφόρμας Okeanos είναι δυνατόν να αλληλοεπιδράσουν σε εντολές από οποιοδήποτε πρόγραμμα μέσω rest κλήσεων. Συγκεκριμένα, ένας μεγάλος κατάλογος από Uri's δίνει την δυνατότητα σε developers να προγραμματίσουν διεργασίες προς εκτέλεση από το cloud. Αυτή η δυνατότητα των rest APIs, αξιοποιήθηκε στα πλαίσια τις εργασίας. Ο στόχος ήταν να συντονιστούν με κατάλληλη σειρά οι διεργασίες που χρειάζονται, ώστε να έχουμε μετάβαση από το πρώτο σχήμα του ενός server, στο νέο σχήμα με δυο servers, ενώ παράλληλα να ορισθούν και τα νέα δικτυακά στοιχεία.

Από την κοινότητα του λογισμικού synnefo (που χρησιμοποιείται από το okeanos) έχει αναπτυχθεί η βιβλιοθήκη **kamaki API library**. Η συγκεκριμένη βιβλιοθήκη συγκεντρώνει όλες τις λειτουργίες της υποδομής. Έχει σχεδιαστεί με τέτοιο τρόπο, ώστε να διευκολύνει τον προγραμματισμό εργασιών, μέσω της κλήσης ορισμένων μεθόδων και κλάσεων που αναλαμβάνουν να εκτελέσουν την εκάστοτε εργασία, χωρίς να είναι αναγκαίο από τον developer να γνωρίζει το σύνολο των αλλεπαλλήλων rest κλήσεων που θα εκτελεστεί από την κλάση.

Η εγκατάσταση της παραπάνω βιβλιοθήκης έγινε στο αρχικό σύστημα, στο οποίο τρέχει το script trigger μέσω του crontab. Οι κλάσεις της θα κληθούν από το πρόγραμμα που αναπτύχθηκε με στόχο να υλοποιηθεί το καθορισμένο σενάριο που έχει σχεδιαστεί. Το Module που αναπτύχθηκε ονομάστηκε **Cluster_Creator.py,** το όποιο περιέχει την κλάση **Core**. Μετά την ολοκλήρωση του ελέγχου των network interfaces από τον trigger, ένα αντικείμενο της Core κλάσης δημιουργείται. Η μοναδική μεταβλητή, που χρειάζεται ο constructor της core κλάσης, είναι ένα ισχύον TOKEN του λογαριασμού χρήστη που διατηρείται στο okeanos.

Στην συνέχεια παρουσιάζεται αναλυτικά κάθε μέθοδος και πως συμβάλει βήμα βήμα στην κατασκευή του cluster.

init

Κάθε πρόγραμμα γραμμένο σε αντικειμενοστραφή γλώσσα, έχει δεσμευμένα keywords για τους Contractors κάθε κλάσης. Για την Python, το λεκτικό αυτό είναι μια συνάρτηση με το όνομα **_init_**. Στο setup της εργασίας ο constructor της core παίρνει και μια string μεταβλητή με το token, από το οποίο φτιάχνει μεταβλητές για το εν λόγω Initialized object. Οι μεταβλητές αυτές είναι με την σειρά τους, εκείνες που παράγουν τα objects της βιβλιοθήκης kamaki και υλοποιούν εργασίες στα αντίστοιχα services του okeanos. Σχεδιάστηκαν με στόχο να διευκολύνουν την αλληλεπίδραση με τα Open stack API. Το Initialize για τα object αυτά απαιτεί, εκτός από το token, που πέρασε στην συνάρτηση μέσω της Κλήσης της core και κάποια URLs με την ονομασία endpoints. Τα endpoints είναι οι δικτυακές θέσεις, στις οποίες θα σταλούν οι εντολές προς το κάθε service. Τα object – variables έχουν global scope , έτσι ώστε οποιαδήποτε άλλη συνάρτηση χρειαστεί, να αντλήσει πληροφορίες από αυτά και να μπορεί ευκολά να κάνει κλήση. Από το σύνολο των service που είναι προσπελάσιμα, η εργασία θα ασχοληθεί με:

Authentication service → μέσω του Object της κλάσης AstakosClient

Compute service → μέσω του Object της κλάσης CycladesComputeClient

Network service→ μέσω του Object της κλάσης CycladesNetworkClient

Σε κάθε περίπτωση το αποτέλεσμα της κλήσης συναρτήσεων των παραπάνω αντικειμένων επιστρέφει πληροφορίες σε μορφή json σε πεπλεγμένη μορφή. Δηλαδή, η διαδικασία εντολών και κλήσεων στο okeanos θα μπορούσε να περιγράφει ως μια ανταλλαγή δομημένων αντικειμένων json σε καθορισμένα APIs. Για λογούς ευκολίας, στον constructor της core ορίστηκε και μια global μεταβλητή που αποθηκεύει το id του project. Το project id ζητείται από την πλατφόρμα κάθε φορά που γίνεται κάποια κλήση για νέους πόρους ή πληροφορίες, έτσι ώστε να ελέγξει, αν το συγκεκριμένο Project έχει δικαίωμα να χρησιμοποιήσει τους ζητούμενους πόρους. Όταν δεν έχει τέτοιο δικαίωμα, εγείρεται αντίστοιχο error.

get_active_servers | get_vm_id

Οι δυο αυτές συναρτήσεις γράφτηκαν για λόγους οικονομίας κώδικα, αλλά και για την καλύτερη και πιο άρτια δομή της κλάσης core. Η πρώτη, με δεδομένα τα στοιχειά του account και του project id δίνει την πληροφορία στην trigger για τον αριθμό των servers, που είναι ενεργοί εκείνη την στιγμή. Έτσι, η κλάση Trigger, μόλις εντοπίσει υψηλό traffic ,ενώ ταυτόχρονα βρει ότι μόνο ένας server είναι active, προχωράει στα επόμενα βήματα επέκτασης του συστήματος. Η δεύτερη μέθοδος, διευκολύνει την ανάγνωση του κώδικα, καθώς παίρνει σαν είσοδο τις ονομασίες που δόθηκαν ή θα δοθούν στους servers και επιστρέφει το id τους. Με αυτό το id γίνεται στην συνέχεια δυνατή η δημιουργία vm, αλλά και σύνδεσης τους με την δικτυακή υποδομή.

create_new_Asterisk_server

Καθώς συνεχίζεται η εκτέλεση των βημάτων εγκατάστασης του νέου cluster και αφού έχουν ενεργοποιηθεί επιτυχώς όλα τα object για την διεπαφη με την πλατφόρμα, ξεκίνα η δημιουργία του νέου VM. Κατά την εκτέλεση της μεθόδου, πραγματοποιούνται 3 λειτουργιές. Η πρώτη, αναζητά το id στην λίστα με τα διαθέσιμα Images, εκείνο το εξατομικευμένο Image που έχει εγκατεστημένο τον asterisk. Η αναζήτηση γίνεται μέσω του ονόματος ,όπως ορίστηκε, κατά το set up του συστήματος από το εργαλείο ***snf image creator***. Στο δεύτερο τμήμα, με μια απλή κλήση της κατάλληλης μεθόδου του compute endpoint object δίνοντας το project id, το image id, μια συμβολοσειρά με το όνομα του server και το flavor id δημιουργείται το νέο VM ,χωρίς όμως κάποιο set up για δικτυακές λειτουργιές. Το δίκτυο θα οριστεί από τις επόμενες μεθόδους. Να σημειωθεί ότι το flavor id δείχνει ποιο πακέτο ram,cpu ,volume θα δοθεί στο VM. Υπάρχει πληθώρα διαφορετικών επίλογων απλώς για τις ανάγκες τις εργασίας ορίστηκε το 29. Το τελευταίο τμήμα καλεί μια μέθοδο, που ουσιαστικά θα σταματήσει την εκτέλεση του κώδικα, μέχρι το νέο VM να περάσει σε status ACTIVE.

attach_public_ip_to_server

Επόμενο στάδιο στην κατασκευή του νέου cluster αποτελεί η παραμετροποίηση των δικτυακών στοιχείων. Ο νέος εγκατεστημένος και λειτουργικός πλέον server, χρειάζεται μια Public IP ,από την οποία θα γίνεται η δικτυακή κίνηση από και προς τους VoIP clients. Τα βήματα που ακολουθεί η μέθοδος, για να καθορίσει την λειτουργιά αυτή, είναι τα έξης. Με μια δομή επανάληψης, μέσω του **Cyclades network** client αναζητά εκείνη την Public Ip που δεν έχει συνδεθεί με κάποιο VM. Μόλις την εντοπίσει, φτιάχνει ένα **Port** σύνδεσης, και με αυτό τον τρόπο την συσχετίζει με το αντίστοιχο VM. Να σημειωθεί ότι από μόνο του το παραγόμενο δικτυακό αντικείμενο δεν συνδέεται σε κάποιο resource, αλλά μέσω της εγκατάστασης ενός εικονικού Port με την εντολή **create port**.

create_internal_network

Τέλος, με την μέθοδο αυτή ολοκληρώνεται η δικτυακή διασύνδεση. Λαμβάνει παραμέτρους με τα ονόματα των 2 servers. Επίσης, δέχεται και μια συμβολοσειρά με το όνομα του νέου δικτύου. Αρχικά, δημιουργείται το νέο private network, δίνοντας στην είσοδο της αντίστοιχης μεθόδου (**createNetwork**) το ID του project από την global μεταβλητή της core μεθόδου , το όνομα του δικτύου και ο τύπος του (**mac_ filtered**) από την static μεταβλητή της κλάσης **CycladesNetworkClient** στο kamaki library. Στο επόμενο βήμα , ορίζεται το υποδίκτυο 192,168,0,0/24 με την μέθοδο **createsubnet** της υπερ κλάσης **NetwokClient,** πάνω στο οποίο θα γίνει η internal επικοινωνία των servers. Τέλος ,όπως αντίστοιχα στα Public interface, θα γίνει η σύνδεση του private network μέσω Ports στα VM με ταυτόχρονο ορισμό σταθερής IP που να προέρχεται, όμως, από το subnet που ορίστηκε στο προηγούμενο βήμα.

7.5 Λογισμικο load balancing HaProxy

Με την ολοκλήρωση της εγκατάστασης των 2 Server είναι απαραίτητο να ορισθούν τα κριτήρια, με τα οποία επιλέγεται ένας χρήστης, για το που θα γίνει register. Για τον σκοπό αυτό, θα χρησιμοποιηθεί το εργαλείο HaProxy. Το HaProxy είναι opensource λογισμικό που προσφέρει δυνατότητες high availability (μέσω load balancing), κατανέμοντας δικτυακή κίνηση εφαρμογών με καταλληλά κριτήρια, είτε με βάση το Layer 4, είτε με βάση το layer 7. Για τις ανάγκες της εργασίας, ο HaProxy εγκαταστάθηκε σε ένα vm με λογισμικό Ubuntu server συνδεδεμένο σε μια Public Ip. Η παραπάνω public ip είναι η ip που θα έχει καθοριστεί στα Zoiper softphones. Αναμενόμενο θα ήταν μια τέτοια σύνδεση να γίνει time out, αλλά, μέσω του αλγορίθμου load balancing, η κίνηση ανακατευθύνεται στον εκάστοτε server, οπού και τελικά θα γίνει το register με επιτυχία.

Οι δυνατότητες load balancing υποστηρίζονται από ένα εύρος διαφορετικών αλγορίθμων που χρησιμοποιεί το HaProxy, οι οποίοι είναι:

- ✓ Round robin: προωθεί συνδέσεις σε όλους τους servers σε σειρά, χωρίς να εξετάζει οποιοδήποτε άλλο χαρακτηριστικό.
- ✓ Least conn: διατηρεί στατιστικά με τον αριθμό των συνδέσεων ανά host, έτσι ώστε κάθε νέα σύνδεση, να προωθείται σε εκείνον τον server που έχει τον μικρότερο αριθμό συνδέσεων
- ✓ Source: ο συγκεκριμένος αλγόριθμος αποτελεί μια πιο ειδική περίπτωση. Συνδέει τον εκάστοτε client με συγκεκριμένο server κάθε χρονική στιγμή. Η επιλογή αυτή γίνεται με βάση την source IP και προτιμάται σε περιπτώσεις συνδέσεων μεγάλης χρονικής διάρκειας.

Παραμετροποίηση του HaProxy server

Αφού εγκατασταθεί το πρόγραμμα από τα βασικά repositories για το ubuntu, εκκινείται το Services, ώστε να βρίσκεται σε status start. Ο καθορισμός περαιτέρω παραμέτρων γίνεται στο αρχείο haproxy.cfg που βρίσκεται στο directory **/etc/haproxy**. Το παραπάνω αρχείο είναι χωρισμένο σε τμήματα που ορίζουν παραμέτρους για διαφορετικές λειτουργίες. Από το αρχικό αρχείο που δημιουργείται κατά την εγκατάσταση, δεν γίνονται αλλαγές στις default τιμές, αλλά ορίζονται επιπλέον κάποια τμήματα που περιγράφονται στην συνέχεια.

- Frontend section

Το τμήμα αυτό αποτελεί το λογικό σημείο σύνδεσης των χρηστών στον load balancer. Περιέχει μια εγγραφή για τον ορισμό των source ips και ports ,από τα οποία δέχεται αιτήματα. Η επιλογή για το σύστημα μας έγινε για οποιαδήποτε IP, αλλά για καθορισμένο port, το 5060. Το 5060 είναι το default port του πρωτοκόλλου SIP. Δεν υπήρξε πρόβλεψη για τα ports του RTP πρωτοκόλλου, καθώς, κατά την εκτέλεση της σηματοδοσίας του SIP, θα έχει ήδη γίνει μεταφορά στον asterisk server και δεν θα υπάρχει λόγος περαιτέρω παραμετροποίησης στον haproxy. Στην τελευταία εγγραφή γίνεται η παραπομπή στο τμήμα του αρχείου παραμετροποίησης που ουσιαστικά θα κάνει το load balancing,στην περίπτωση μας το section αυτό ονομάζεται nodes.

- Backend section

Αφού γίνει η αποδοχή κάποιου αιτήματος σύνδεσης στο front-end ,αναλαμβάνει το backend να εκτελέσει τις λειτουργίες. Στο τμήμα αυτό καθορίζονται οι περιοχές που δύναται να εξυπηρετεί ο proxy μέσω των IP's των servers. Επιπλέον, η εντολή check ελέγχει την διαθεσιμότητα του server για την αποφυγή προώθησης χρήστη σε μη λειτουργικό server. Τέλος, καθορίζεται και ο αλγόριθμος κατανομής των χρηστών ανά τους servers. Για την περίπτωση μας επιλέχθηκε ο βασικός αλγόριθμος round robin.

7.6 Παρουσιαση λειτουργιας Υπηρεσιας VOIP.

Οι ρυθμίσεις που ορίστηκαν προηγουμένως χρησιμεύουν, ώστε να κάνουν το service διαθέσιμο. Όμως, είναι αναγκαίο και από την μεριά του client να οριστούν οι κατάλληλοι παράμετροι. Στην συνέχεια, θα παρουσιαστεί η σύνδεση ενός μεμονωμένου softphone (zoipper) στην υποδομή και πως αντίστοιχα αποκρίνονται οι servers μέσα από τα αντίστοιχα Logs. Ο χρήστης που θα εμφανίζεται είναι εκείνος με το extension 1001

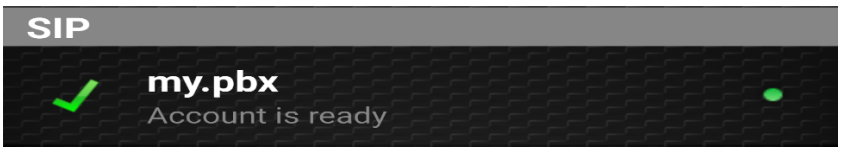
- Στάδιο 0

Στην παρακάτω εικόνα βρίσκεται το αποτέλεσμα της εντολής **sip show peers** στο cli του asterisk. Εκεί εμφανίζονται οι χρήστες που έχουν ορισθεί στο αρχείο sip.conf. Όμως, κανένας ακόμα δεν είναι συνδεδεμένος. Αυτή είναι η αρχική κατάσταση του συστήματος, πριν ακόμα αρχίσει να διαχειρίζεται χρήστες και κλήσεις.

```
snf-761179*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport C
omedia     ACL Port     Status      Description
1001/1001                 (Unspecified)                            D  Auto (No)  N
o                0        Unmonitored
1002/1002                 (Unspecified)                            D  Auto (Yes) N
o                0        Unmonitored
1003/1003                 (Unspecified)                            D  Auto (No)  N
o                0        Unmonitored
1004                      (Unspecified)                            D  Auto (No)  N
o                0        Unmonitored
4 sip peers [Monitored: 0 online, 0 offline Unmonitored: 0 online, 4 offline]
```

- Στάδιο 1

Έπειτα, η σύνδεση κάποιου εγκατεστημένου χρήστη απαιτεί την εισαγωγή στο softphone την ip του load balancer με τα αντίστοιχα εξατομικευμένα στοιχεία (extension , password). Με τον τρόπο αυτό γίνεται το authentication και ο χρήστης πλέον μπορεί να χρησιμοποιήσει την υπηρεσία VOIP.



- Στάδιο 3

Στην συνέχεια γίνεται σύνδεση περισσοτέρων χρηστών και με την εντολή sip show peers παρατηρούνται οι αλλαγές που έχουν γίνει στο status των accounts. Να σημειωθεί, ότι ,παρόλο που οι 2 χρήστες (1001 & 1002) έχουν συνδεθεί από διαφορετικά δίκτυα, το pbx βλέπει την IP, από οπού έφτασε το αίτημα για register. Δηλαδή ,αναγνωρίζει την IP του Load balancer.

```
snf-761179*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport C
omedia    ACL Port     Status      Description
1001/1001                 83.212.105.127                           D  Auto (No)  N
o             50602    Unmonitored
1002/1002                 83.212.105.127                           D  Auto (No)  N
o             49872    Unmonitored
1003/1003                 (Unspecified)                            D  Auto (No)  N
o             0        Unmonitored
1004                      (Unspecified)                            D  Auto (No)  N
o             0        Unmonitored
4 sip peers [Monitored: 0 online, 0 offline Unmonitored: 2 online, 2 offline]
```

- Στάδιο 4

Στο σημείο αυτό θα γίνει προσπάθεια κλήσης από τον χρήστη 1001 στον 1002. Η κλήση πραγματοποιήθηκε κανονικά κατά τον έλεγχο. Επιπλέον, παρακάτω φαίνεται πως κατέγραψε το Log του συστήματος την διαδικασία. Από το αρχείο στο directory **/var/log/asterisk/cdr-csv** παρατηρείται το συμβάν κλήση του 1001 στον 1002. Όλες οι λεπτομερείς που σχετίζονται με το συμβάν βρίσκονται εδώ. Να επισημάνουμε ότι στην παραμετροποίηση του dial plan ορίσαμε πρώτα τον server να ελέγχει τα context του τοπικού server και στην συνέχεια ελέγχει το δεύτερο PBX. Στην περίπτωση μας έχουμε έναν ενεργό server, οπότε η κλήση έγινε μέσω sip.

```
"","1001","1002","admin","""" <1001>","SIP/1001-00000000","SIP/1002-00000001","
Dial","IAX2/zone1/1002@admin,15,r","2017-11-25 09:49:53","2017-11-25 09:50:02","
2017-11-25 09:50:14",21,12,"ANSWERED","DOCUMENTATION","1511603393.0",""
```

Στην συνέχεια θα παρουσιαστούν οι διαδικασίες που λαμβάνουν μέρος κατά την είσοδο ,αλλά και κατά την πραγματοποίηση κλήσεων στο διευρυμένο set up. Να θυμίσουμε ,ότι οι διαφορές παρουσιάζονται στο δικτυακό μέρος με ταυτόχρονη συμμετοχή ,όμως, και των processes για την επίτευξη της επιθυμητής λειτουργικότητας. Ειδικότερα, εκτός των 2 pbxs και του Load balancer χρησιμοποιείται και ένα επιπλέον εικονικό δίκτυο που δρομολογεί κλήσεις μεταξύ των χρηστών αναμεσά στους 2 διαφορετικούς Server. Η δρομολόγηση αυτή πραγματοποιείται από το εικονικό subnet, αλλά συντονίζεται, παράλληλα, από το dialplans των Pbxs.

- Βήμα 1

Μόλις ένας χρήστης προσπαθήσει να συνδεθεί στην πλατφόρμα, αναλαμβάνει ο Load balancer να τον καθοδηγήσει καταλληλά. Στο παράδειγμα που ακολουθεί ,ο haproxy έχει παραμετροποιήσεις ,ώστε να ακολουθεί αλγόριθμο round robin. Έτσι, μόλις προσπαθήσουν οι χρήστες του παραδείγματος (1001 & 1002) να συνδεθούν ,θα τους παραπέμψει εναλλάξ στα 2 διαφορετικά Pbxs. Παρακάτω βλέπουμε το log αρχείο από τον haproxy ,οπού αποτυπώνει την παραπομπή αρχικά στον Asterisk PBX για τον πρώτο χρήστη 1001 και στο Extended zone για τον δεύτερο 1002. Επίσης, αποτυπώνεται και το event ,οπού ο Load balancer πλέον βλέπει και τον νέο server Extended zone.

```
Nov 26 15:02:04 snf-769593 haproxy[880]: Server nodes/Extended_zone is UP, reason: Layer4 check passed, check duration: 1ms. 2 active and 0 backup servers online. 0 sessions req
ueued, 0 total in queue.
Nov 26 15:13:31 snf-769593 haproxy[880]: 5.55.9.97:34847 [26/Nov/2017:15:10:32.243] localnodes nodes/Asterisk_PBX 1/1/179003 3458 cD 1/1/1/0/0 0/0
Nov 26 15:16:06 snf-769593 haproxy[880]: 5.55.9.97:44306 [26/Nov/2017:15:13:31.521] localnodes nodes/Asterisk_PBX 1/1/155034 4562 cD 1/1/1/0/0 0/0
Nov 26 15:19:31 snf-769593 haproxy[880]: 5.55.9.97:36570 [26/Nov/2017:15:16:06.773] localnodes nodes/Extended_zone 1/1/204744 4562 cD 1/1/1/1/0 0/0
Nov 26 15:20:56 snf-769593 haproxy[880]: 5.55.9.97:50989 [26/Nov/2017:15:19:31.977] localnodes nodes/Asterisk_PBX 1/1/84865 4562 cD 1/1/1/0/0 0/0
```

Για να επιβεβαιωθεί και το επιτυχές register στους 2 server παρουσιάζουμε και το αποτέλεσμα της εντολής sip show peers στους 2 ξεχωριστά CLIs που αποτυπώνουν τα διαφορετικά ενεργά extensions εκείνη την στιγμή.

```
snf-761179*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport
 Comedia    ACL Port     Status      Description
1001/1001                  83.212.105.127                          D  Auto (No)
 No            33544     Unmonitored
1002/1002                  (Unspecified)                           D  Auto (Yes)
 No            0         Unmonitored
1003/1003                  (Unspecified)                           D  Auto (No)
 No            0         Unmonitored
1004                       (Unspecified)                           D  Auto (No)
 No            0         Unmonitored
4 sip peers [Monitored: 0 online, 0 offline Unmonitored: 1 online, 3 offline]
```

```
snf-789478*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport
 Comedia    ACL Port     Status      Description
1001/1001                  (Unspecified)                           D  Auto (No)
 No            0         Unmonitored
1002/1002                  83.212.105.127                          D  Auto (Yes)
 No            47756     Unmonitored
1003                       (Unspecified)                           D  Auto (No)
 No            0         Unmonitored
3 sip peers [Monitored: 0 online, 0 offline Unmonitored: 1 online, 2 offline]
snf-789478*CLI>
```

- Βήμα 2

Τέλος, θα παρουσιαστούν οι εγγραφές στο cdr αρχείο των 2 pbxs, ώστε να παρουσιαστεί ο τρόπος και τα βήματα, με τα οποία γίνεται η αναζήτηση του χρήστη και πως αυτή δρομολογείται μέσω της σύνδεσης trunk με το πρωτόκολλο Iax.

Asterisk pbx

Για τον πρώτο server το συμβάν κλήση του χρήστη 1001 στον 1002 γίνεται με κλήση στον γειτονικό server (zone2) με χρήση του IAX. Παρατηρείται, ότι από την στιγμή που το συμβάν περάσει σε status ANSWERED δεν δίνει άλλες λεπτομέρειες για την τύχη της κλήσης, καθώς πλέον έχει φύγει από το πεδίο εφαρμογής του Asterisk PBX και έχει περάσει στο Extended zone.

```
"","1001","1002","admin","""" <1001>","IAX2/zone2-13918","IAX2/zone2-5182","Dia
l","IAX2/zone2/1002@admin,30,r","2017-11-26 13:43:07","2017-11-26 13:43:19","201
7-11-26 13:43:24",17,5,"ANSWERED","DOCUMENTATION","1511703787.1001",""
root@snf-789478:/var/log/asterisk/cdr-csv#
```

Extended zone

Τα επόμενα στοιχεία παρουσιάζουν το συμβάν λήψης της κλήσης από την zone 1. Στην συνέχεια ,ο server2, ακολουθώντας το δικό του dialplan, δημιουργεί την κλήση με SIP πρωτόκολλο στον client 1002

```
"","1001","1002","admin","""" <1001>","IAX2/zone1-12443","SIP/1002-00000020","Dial","SIP
ENTATION","1511689207.3500",""
"","1001","1002","admin","""" <1001>","SIP/1001-00000021","IAX2/zone1-6654","Dial","IAX2
NSWER","DOCUMENTATION","1511703753.3750",""
```

7.7 Επίλογος – Προτάσεις για μελλοντική έρευνα

Το μοντέλο που παρουσιάστηκε στην εργασία ανέδειξε το μεγάλο εύρος δυνατοτήτων και εφαρμογών που μπορεί να υποστηριχθεί από μια cloud πλατφόρμα. Επιπλέον αποδείχθηκε ότι οι διεργασίες που πραγματοποιεί κάθε λειτουργική μονάδα (VM η δίκτυο) δεν είναι ανεξάρτητες από την υπόλοιπη υποδομή, αλλά μπορούν να αλληλοεπιδράσουν με το σύστημα μέσω των αντίστοιχων rest APIs. Στην κατεύθυνση αυτή, παρατίθενται ορισμένες προτάσεις για περαιτέρω αξιοποίηση των δυνατοτήτων σε setup που μπορεί να μην σχετίζονται με VoIP εφαρμογές.

- ✓ *Εφαρμογή διεργασιών διαχείρισης αποφάσεων για χρήση, επέκταση ή και αποδέσμευσης πόρων, λαμβάνοντας υπ οψιν όχι μόνο τεχνικά κριτήρια, αλλά και κριτήρια κόστους, προσαρμοσμένα φυσικά στην τιμολογιακή πολιτική του εκάστοτε παρόδου.*
- ✓ *Ανάπτυξη αυτοματοποιημένων διεργασιών επέκτασης συστήματος σε περιπτώσεις, οπού οι ανάγκες μπορεί να μεταβληθούν ραγδαία.*
- ✓ *Προεγκατάσταση προγραμμάτων και λειτουργιών σε εξατομικευμένα images, με στόχο την αξιοποίηση της πλατφόρμας και για εμπορικούς σκοπούς με δυνατότητα εύκολης απόκτησης των παρεχόμενων υπηρεσιών από τρίτα μέρη.*
- ✓ *Ο συντονισμός των διαφορετικών services δίνει δυνατότητες για ανάπτυξη αποκεντρωμένων συστημάτων με διαμοιρασμό των λειτουργιών σε πολλά αυτόνομα στοιχεία, όπως για παράδειγμα, η αξιοποίηση του pithos ως backend διαμοιρασμού στοιχείων σε άλλες μονάδες.*
- ✓ *Migration σε νέο σύστημα με μηδενισμό του downtime της αρχικά παρεχόμενης υπηρεσίας.*

References

[1] «The NIST definition of cloud computing,» *National Institute of Standards and Technology,* 2011.

[2] A. P. D. D. ,. N. B. K C Gouda, «Virtualization Approaches in Cloud Computing,» *International Journal of Computer Trends and Technology (IJCTT) ,* 2014.

[3] B. P. Tholeti, September 23, 2011. [Ηλεκτρονικό]. Available: https://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/.

[4] R. H. Linux, «https://www.linux-kvm.org/page/Main_Page,» [Ηλεκτρονικό]. [Πρόσβαση 2017].

[5] «Ganeti's documentation,» [Ηλεκτρονικό]. Available: http://docs.ganeti.org/ganeti/2.15/html/.

[6] L. Albertson, «Openstack vs Ganeti,» Open source lab.

[7] D. Radez, Openstack essentials, Packt Publishing LTD , 2015.

[8] P. Godse, June 18, 2015. [Ηλεκτρονικό]. Available: http://kb.bodhost.com/introduction-to-the-eleven-key-components-of-openstack-part-1/.

[9] O. C. release, «Openstack documentation,» 2017. [Ηλεκτρονικό]. Available: https://docs.openstack.org/ocata/.

[10] V. a. G. t. u. N. &. O. Running VLAN. [Ηλεκτρονικό]. Available: http://blog.arunsriraman.com/2015/12/running-vlan-vxlan-and-gre-together.html.

[11] E. Koukis και P. Louridas, «Okeanos IaaS,» *PROCEEDINGS of SCIENCE,* 2012.

[12] «Greek Research and Technology Network's projects,» 2017. [Ηλεκτρονικό]. Available: https://code.grnet.gr/projects.

[13] T. W. Paper, alpha version 2011 v0.1. [Ηλεκτρονικό]. Available: https://okeanos.grnet.gr/media/medialibrary/2012/03/okeanos-grnet_whitepaper_alpha_11v01-20111128.pdf.

[14] A. S. TANENBAUM και D. J. WETHERALL, Computer Networks, PRENTICE HALL.

[15] C. H. R. B. K. B. Tim Szigeti, End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks, Cisco Press, 2013.

[16] N. S. e.V.. [Ηλεκτρονικό]. Available: http://www.networxsecurity.org/members-area/glossary/n/network-delay.html.

[17] B. A. Forouzan, Data Communications and Networking, 2000.

[18] A. B. Johnston, SIP: Understanding the Session Initiation Protocol, 2001.

[19] «Tutorial Point,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/session_initiation_protocol/session_initiation_protocol_network_elements.htm.

[20] M. S. -Digium.Inc, «IAX: Inter-Asterisk eXchange Version 2,» Internet Engineering Task Force (IETF).

[21] F. E. Goncalves, Configuration Guide for Asterisk PBX, 2007.

[22] J. V. Meggelen, L. Madsen και R. Bryant, Asterisk: The Definitive Guide, 4th Edition, O'Reilly Media, 2013.

[23] Srikanth, October 23, 2017. [Ηλεκτρονικό]. Available: https://www.techiexpert.com/india-tackled-cloud-unique-way/.

[24] h. Huynh και S. Hajnoczi, «KVM/QEMU Storage Stack Performance Discussion,» σε *Linux Plumbers Conference.*, 2010.

[25] D. S.A. [Ηλεκτρονικό]. Available: http://www.dinecom.cl/blogs/diccionario-de-conceptos/videoconferencia-que-se-supone-que-mide-el-jitter-buffer/.

Παράρτημα

- Python πρόγραμμα υλοποίησης του νέου cluster

  - ✓ Trigger.py στο home directory του Asterisk PBX Server

```
from psutil import net_io_counters
from time import sleep
from Cluster_Creator import Core
from kamaki.clients.utils import https

https.patch_ignore_ssl()

snap1=net_io_counters(pernic=False)
sleep(30)
snap2=net_io_counters(pernic=False)
throughput = ((snap2[0]+snap2[1])-(snap1[0]+snap1[1]))/30

TOKEN="USER-TOKEN"
main_server_name="Asterisk PBX"
new_server_name="Extended_zone"
network_name="Internal Network"

if throughput > 1:
   obj=Core(TOKEN)
   if obj.get_active_servers()==1:
      obj.create_new_Asterisk_server(new_server_name)
      obj.attach_public_ip_to_server(new_server_name)
      obj.create_internal_network(network_name,main_server_name,new_server_name)
```

✓ Cluster_Creator.py στο home directory του Asterisk PBX Server

```
from astakosclient import AstakosClient
from kamaki.clients.cyclades import CycladesComputeClient , CycladesNetworkClient
from kamaki.clients.utils import https

class Core(object):

    def __init__(self,TOKEN):

self.identity_res=AstakosClient(TOKEN,"https://accounts.okeanos.grnet.gr/identity/v
2.0")
self.network_res=CycladesNetworkClient('https://cyclades.okeanos.grnet.gr/network',
TOKEN)
self.compute_res=CycladesComputeClient('https://cyclades.okeanos.grnet.gr/compute/v
2.0',TOKEN)
self.project_id=self.identity_res.authenticate()["access"]["user"]["projects"][2]

    def get_active_servers(self):
        self.identity_res.authenticate()
        resources= self.identity_res.get_quotas()
        active_servers=resources[self.project_id]['cyclades.vm']['project_usage']
        return active_servers

    def get_vm_id(self,server):
        for i in self.compute_res.list_servers():
            if i["name"]==server:
                return i["id"]

    def create_new_Asterisk_server(self,new_server_name):
        https.patch_ignore_ssl()

        all_images=self.compute_res.list_images()
        for i in range(len(all_images)):
            if all_images[i]["name"] == "Asterisk-PBX":
                asterisk_id=all_images[i]["id"]
                break

compute=self.compute_res.create_server(name=new_server_name,flavor_id=29,image_id=a
sterisk_id , networks=[], project_id=self.project_id)

self.compute_res.wait_server_until(server_id=compute["id"],target_status="ACTIVE",m
ax_wait=1000,delay=30)

    def attach_public_ip_to_server(self,server_name):
        for i in self.network_res.list_floatingips():
            if i["instance_id"]==None:
                network_id=i["floating_network_id"]
                ip=[dict(ip_address=i["floating_ip_address"]), ]

self.network_res.create_port(device_id=self.get_vm_id(server_name),network_id=netwo
rk_id,fixed_ips=ip)

    def create_internal_network(self,network_name,server1,server2):
        network_type = self.network_res.network_types[1]

net=self.network_res.create_network(type=network_type,name=network_name,project_id=
self.project_id)
        self.network_res.create_subnet(network_id=net["id"],cidr="192.168.0.0/24")
id1=self.network_res.create_port(network_id=net["id"],device_id=self.get_vm_id(serv
er1),fixed_ips=[{"ip_address":"192.168.0.2"}])
        self.network_res.wait_port_until(port_id=id1["id"],target_status="ACTIVE")

self.network_res.create_port(network_id=net["id"],device_id=self.get_vm_id(server2)
,fixed_ips=[{"ip_address":"192.168.0.3"}])
```

- Αρχεία παραμετροποίησης asterisk server στο directory /etc/asterisk.

  - ✓ Sip.conf: κοινό και στους 2 server

```
[general]
tcpenable=yes
tcpbindaddr=0.0.0.0

[admin](!)
type=friend
host=dynamic
context=admin
disallow=all
allow=ulaw


[1001](admin)
secret=1234
transport=tcp

[1002](admin)
secret=1234
transport=tcp

[1003](admin)
secret=1234
transport=tcp

[1004](admin)
secret=1234
transport=tcp
```

  - ✓ extensions.conf: για τον server Asterisk PBX

```
[admin]

exten=>_1XXX,1,Dial(IAX2/zone1/${EXTEN}@admin,30,r)
same=> n,Dial(SIP/${EXTEN},15,r)
```

  - ✓ extensions.conf: για τον server Extended zone

```
[admin]

exten=>_1XXX,1,Dial(IAX2/zone2/${EXTEN}@admin,30,r)
same=> n,Dial(SIP/${EXTEN},15,r)
```

- ✓ iax.conf: για τον server Asterisk PBX

```
[general]

register => zone2:1234@192.168.0.3


[zone1]

type=friend
host=dynamic
secret=1234
context=admin
peercontext=admin
qualify=yes
```

- ✓ iax.conf: για τον server Extended Zone

```
[general]

register => zone1:1234@192.168.0.2


[zone2]

type=friend
host=dynamic
secret=1234
context=admin
peercontext=admin
qualify=yes
```

- Αρχειο παραμετροποιησης HaProxy server στο directory /etc/haproxy.

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+
AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:
!DSS
        ssl-default-bind-options no-sslv3
defaults
        log     global
        mode    tcp
        option  tcplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

frontend localnodes
    bind *:5060
    mode tcp
    option tcplog
    default_backend nodes


backend nodes
    mode tcp
    option tcplog
    balance roundrobin
    server Asterisk_PBX (SERVER 1 IP):5060 check
    server Extended_zone (SERVER 2 IP):5060 check
```