ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΕΚΤΕΛΕΣΗΣ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΕΦΑΡΜΟΓΩΝ ΜΕ ΤΗΝ ΕΥΡΕΣΗ ΚΑΤΑΛΛΗΛΟΤΕΡΟΥ ΔΙΑΚΟΜΙΣΤΗ ΣΕ ΔΙΚΤΥΑ ΧΑΜΗΛΗΣ ΧΩΡΗΤΙΚΟΤΗΤΑΣ

Επιμέλεια:
ΤΣΟΤΖΟΛΑΣ ΓΕΩΡΓΙΟΣ (ΜΕ1627)
Επιβλέπων: ΔΗΜΟΣΘΕΝΗΣ ΚΥΡΙΑΖΗΣ,ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΦΕΒ 2018

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

Κατεύθυνση: Προηγμένα Πληροφοριακά Συστήματα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση εκτέλεσης κατανεμημένων εφαρμογών με την εύρεση καταλληλότερου διακομιστή σε δίκτυα χαμηλής χωρητικότητας

Επιμέλεια:

ΤΣΟΤΖΟΛΑΣ ΓΕΩΡΓΙΟΣ (ΜΕ1627)

Επιβλέπων: **ΔΗΜΟΣΘΕΝΗΣ ΚΥΡΙΑΖΗΣ, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ**

ΦΕΒ 2018

Ευχαριστίες

Για την περάτωση της διπλωματικής μου εργασίας, πέραν της προσωπικής προσπάθειας που κατεβλήθη καθ' όλη τη διάρκεια της ακόλουθης μελέτης, βασικό ρόλο στη σύλληψη και τη διαμόρφωση της τελικής εργασίας διαδραμάτισαν τα ερεθίσματα τα οποία αποκόμισα από το περιβάλλον εργασίας μου.

Θα ήθελα να ευχαριστήσω τον επίκουρο Καθηγητή του Πανεπιστημίου Πειραιώς και επιβλέποντα αυτής της εργασίας κ. Δημοσθένη Κυριαζή για την πολύτιμη συμβολή του στη διεκπεραίωση αυτής της εργασίας με την υποστήριξη, τη συμπαράσταση, το χρόνο και την προσωπική γνώση που μου παρείχε για την κατανόηση του αντικειμένου της διπλωματικής εργασίας και το ευχάριστο κλίμα συνεργασίας κάτω από το οποίο αυτή πραγματοποιήθηκε.

Τέλος, θα ήθελα να ευχαριστήσω τους δικούς μου ανθρώπους και κυρίως την γυναίκα η οποίο έδειξε μεγάλη κατανόηση και υπομονή ώστε να ολοκληρώσω με επιτυχία το συγκεκριμένο κύκλο σπουδών μου.

Περίληψη

Η παρούσα διπλωματική εργασία αποβλέπει στο να επιλύσει ένα πραγματικό πρόβλημα που πρόκειται να αντιμετωπίσει ένα μεγάλος οργανισμός στην Ελλάδα το δίκτυο του οποίου εκτείνεται σε όλη την Ελληνική επικράτεια. Το δίκτυο αυτό είναι ένα κλειστό δίκτυο (intranet) χωρίς να έχει σύνδεση με το διαδίκτυο (internet). Ο οργανισμός στο προσεχές μέλλον πρόκειται να να μεταπέσει σε μία κατανεμημένη λειτουργία των διαδικτυακών του εφαρμογών. Το πρόβλημα το οποίο καλείται αυτή η εργασία να επιλύσει είναι να βρεθεί ο καταλληλότερος διακομιστής ιστού (web application server) για να εξυπηρετήσει πιο γρήγορα τα αιτήματα (request) των χρηστών ώστε να παρέχει καλύτερη ποιότητα υπηρεσιών.

Στην συνέχεια αναλύονται αντίστοιχες τεχνολογίες οι οποίες έχουν δώσει λύση σε παρόμοια προβλήματα στο διαδίκτυο και παρουσιάζεται μια προτεινόμενη λύση. Η λύση η οποίο προτείνεται είναι να χρησιμοποιηθεί ένας αντίστροφος διακομιστής μεσολάβησης (reverse proxy) ο οποίος θα εξυπηρετεί τους χρήστες ανά συγκεκριμένη γεωγραφική περιοχή. Αυτός ο διακομιστής μεσολάβησης θα επιλέγει τον καταλληλότερο διακομιστή για να στείλει τα αιτήματα των χρηστών με σκοπό την καλύτερη και γρηγορότερη εξυπηρέτηση των αιτημάτων.

Ακολούθως παρουσιάζεται το δοκιμαστικό περιβάλλον το οποίο υλοποιήθηκε ώστε να μπορέσουν να γίνουν δοκιμές στην προτεινόμενη λύση και αναλύονται οι μετρικές βάση των οποίον γίνονται οι μετρήσεις βάση των οποίων εξάγονται κάποια συμπεράσματα.

Η επιλογή του καταλληλότερου διακομιστή γίνεται με την μέτρηση της βαθμολογίας Apdex , του μέσου φόρτου του τελευταίου λεπτού καθώς και τον ρυθμό μεταβολής του φόρτου αυτού και προτείνεται ένας μαθηματικός τύπος βάση του οποίου βγαίνει ο καταλληλότερος διακομιστής.

Τέλος αναλύονται τα μελλοντικά βήματα το οποία πρέπει να γίνουν ώστε να μπορέσει να μπει στην παραγωγική διαδικασία το προτεινόμενο σύστημα.

Abstract

This diploma thesis aims to solve a real problem facing a large organization in Greece, whose network extends throughout the Greek territory. This network is a closed network (intranet) without internet connectivity. The organization concerned in the near future is about to switch to a distributed online application mode. The problem that this thesis wants to solve is to find the most suitable web application server to serve users faster to provide better service quality.

Next, corresponding technologies that have solved similar problems on the internet are analyzed and a suggested solution is presented. The solution proposed is to use a reverse proxy server that will serve users per geographic area. This proxy server will choose the most suitable server to send user requests for better and faster service requests.

The test environment is then presented to test the proposed solution and analyze the metrics on which the measurements are made and on the basis of which some conclusions are drawn. Selecting the most suitable server is done by measuring the Apdex score, the last minute load average, and the rate of change of that load, and a mathematical formula based on which the most suitable server comes out.

Finally, the future steps that need to be taken to enable the proposed system to go into the production process are analyzed.

Περιεχόμενα

Κατάλογος Εικόνων

1. Εισαγωγή

Στην σημερινή εποχή, με την εδραίωση της νεφοϋπολογιστικής (cloud computing) έχουν απλοποιηθεί σε μεγάλο βαθμό κάποιες διαδικασίες τόσο στον τομέα των συστημάτων όσο και στον τομέα της παροχής υπηρεσιών . Με την χρήση της νεφοϋπολογιστικής, οι εφαρμογές είναι εγκατεστημένες και εκτελούνται σε διακομιστές (servers) οι οποίοι δεν ελέγχονται από τους κατόχους των εφαρμογών και των υπηρεσιών αλλά από τρίτους. Αυτοί οι διακομιστές, διαχειρίζονται από μεγάλες εταιρίες με μεγάλη εμπειρία και με πολλούς πόρους και παρέχουν υψηλής ποιότητας υπηρεσίες προς τους πελάτες τους.

Τα πράγματα όμως γίνονται αρκετά πιο δύσκολα, όταν αναγκαστικά κάποιος πρέπει να εναρμονιστεί σε ένα περιβάλλον το οποίο έχει αρκετούς περιορισμούς και δεν μπορεί να λειτουργήσει σε κάποιο υπολογιστικό σύννεφο (cloud). Σε αυτή την περίπτωση θα πρέπει να βρεθούν λύσεις, ώστε η παροχή υπηρεσιών προς τους χρήστες να είναι η καλύτερη δυνατή.

Η παρούσα διπλωματική εργασία θα προσπαθήσει να δώσει λύση σε ένα πραγματικό πρόβλημα το οποίο υπάρχει σε έναν μεγάλο οργανισμό ο οποίος παρέχει υπηρεσίες σε χρήστες σε όλη την Ελληνική επικράτεια. Θα αναλύσει το πρόβλημα, θα αναφερθεί διεξοδικά στις λύσεις που υπάρχουν στο διαδίκτυο και λύνουν παρόμοια προβλήματα τα οποία υπάρχουν, και θα παρουσιάσει μια προτεινόμενη λύση.

2. Παρουσίαση του προβλήματος

Στην παρούσα ενότητα θα αναλυθεί ποιο είναι το πρόβλημα το οποίο η παρούσα εργασία καλείται να επιλύσει. Στη συνέχεια θα τεθούν τα δεδομένα πάνω στα οποία πρέπει να παρέχονται υπηρεσίες στους χρήστες μέσω διαδικτυακών εφαρμογών (web applications), ενώ επίσης θα γίνει αναφορά στην υπάρχουσα τοπολογία των πληροφοριακών συστημάτων καθώς και στη μελλοντική τοπολογία την οποία πρόκειται να υιοθετήσει ο οργανισμός. Τέλος, θα παρουσιαστεί ένα σενάριο για την ανάλυση του προβλήματος.

2.1. Περιγραφή περιορισμών

Ένας από τους πιο σημαντικούς περιορισμούς στο συγκεκριμένο περιβάλλον, στο οποίο μελετάτε η παρούσα εργασία, είναι η παρουσία κλειστού δικτύου (intranet) το οποίο δεν έχει άμεση επικοινωνία με το διαδίκτυο (World Wide Web). Αυτός και μόνο ο περιορισμός είναι αρκετά δεσμευτικός αφού δεν μπορούν να χρησιμοποιηθούν άλλες υποδομές και υπηρεσίες παρά μόνο αυτές τις οποίες πρέπει να αναπτύξει ο ίδιος ο οργανισμός, αναλαμβάνοντας ταυτόχρονα το κόστος και το καθήκον της συντήρησής τους. Το δίκτυο αυτό εκτείνεται σε όλη την Ελληνική επικράτεια και έχει χρήστες σε περιοχές οι οποίες είτε είναι αστικές, είτε ημιαστικές/αγροτικές ή και νησιωτικές.

Ένας άλλος μεγάλος περιορισμός είναι ότι το προαναφερθέν δίκτυο είναι χαμηλής χωρητικότητας . Η ταχύτητά του ποικίλλει από περιοχή σε περιοχή. Ακόμα η ποιότητα του δικτύου μπορεί να μεταβάλλεται κατά την διάρκεια της ημέρας λόγω της χαμηλής χωρητικότητας του και της πληθώρας υπηρεσιών που εξυπηρετούνται από αυτό. Πιο συγκεκριμένα, μέσα από το δίκτυο αυτό παρέχονται διαδικτυακές υπηρεσίες ιστού (web εφαρμογές), ηλεκτρονικού ταχυδρομείου (email) αλλά και υπηρεσίες οι οποίες δημιουργούν μεγάλο φόρτο στο δίκτυο, όπως είναι υπηρεσίες μεταφοράς εικόνας και ήχου. Η κίνηση του δικτύου ελέγχεται από τον οργανισμό,

χωρίς όμως να επιβάλλει κανένα περιορισμό στους χρήστες του σχετικά με τον όγκο των δεδομένων ή τον τύπο των υπηρεσιών που μπορούν να χρησιμοποιήσουν ή και τη χρονική διάρκεια αυτών.

Οι υπηρεσίες οι οποίες μελετούνται στην παρούσα περίπτωση είναι διαδικτυακές υπηρεσίες ιστού (web applications). Αναφέρονται σε διαχειριστικές εφαρμογές και σε εφαρμογές παροχής πληροφορίας περιεχομένου και δεν είναι εφαρμογές παρουσίασης περιεχομένου. Δεν έχουν μεγάλο φόρτο περιεχομένου (εικόνες, βίντεο) αλλά έχουν μεγάλη αλληλεπίδραση με βάση δεδομένων για την άντληση δεδομένων από αυτήν.

Πάνω από το δίκτυο αυτό ο οργανισμός καλείται να παρέχει όσο το δυνατόν καλύτερη ποιότητα υπηρεσιών (Quality of Service) στις διαδικτυακές εφαρμογές τις οποίες αναπτύσσει και υποστηρίζει ώστε να επιτύχει το μικρότερο δυνατόν χρόνο απόκρισης τους (responce time).

2.2. Υφιστάμενη Τοπολογία

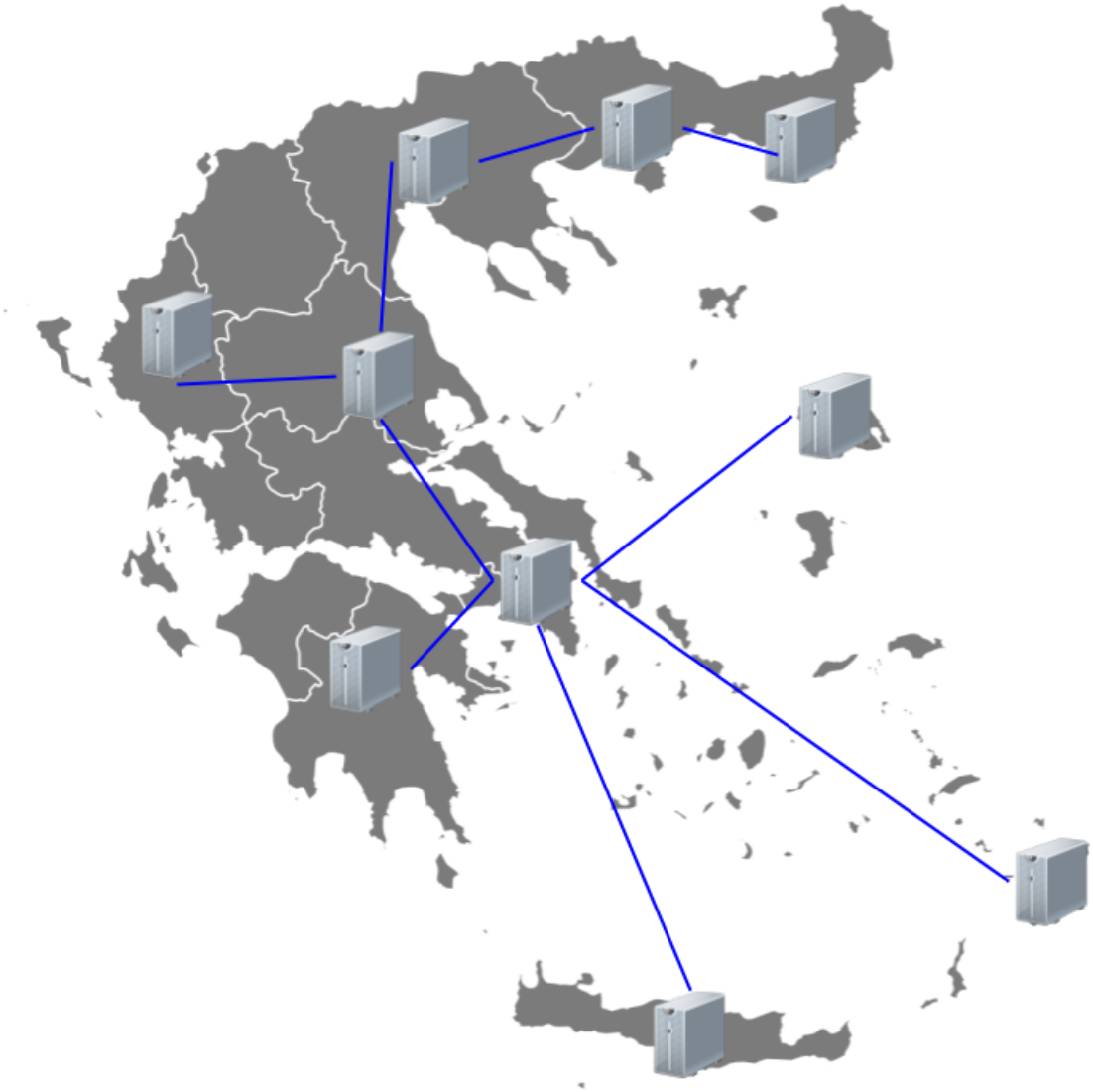
Στην παρούσα κατάσταση η τοπολογία των συστημάτων την οποία έχει ο οργανισμός για την παροχή υπηρεσιών διαδικτύου βασίζεται σε μια πλήρως κεντρικοποιημένη τοπολογία. Οι διακομιστές οι οποίοι παρέχουν τις παρεχόμενες υπηρεσίες βρίσκονται σε δύο κέντρα διακομιστών (data center) τα οποία βρίσκονται σε μικρή απόσταση μεταξύ τους και συνδέονται με δίκτυο πολύ μεγάλης χωρητικότητας . Από τους διακομιστές αυτούς εξυπηρετούνται όλοι οι χρήστες σε όλη την Ελληνική Επικράτεια.

Εικόνα 1: Υφιστάμενη τοπολογία οργανισμού

Αυτή η κατάσταση έχει προφανώς πλεονεκτήματα και μειονεκτήματα. Από την μία η κεντρικοποιημένη λειτουργία έχει το σημαντικό πλεονέκτημα της μειωμένης πολυπλοκότητας και της εύκολης συντήρησης. Αντίθετα στην κατανεμημένη λειτουργία υπάρχει αυξημένη πολυπλοκότητα και δυσκολία συντήρησης. Από την άλλη όμως στην κατανεμημένη λειτουργία έχουμε κάποια πολύ σημαντικά πλεονεκτήματα όπως είναι η μεγαλύτερη ανοχή σφαλμάτων ,η επεκτασιμότητα , η μεγαλύτερη διαθεσιμότητα ,η καλύτερη απόδοση ,η μεγαλύτερη αξιοπιστία , και άλλα.

2.3. Μελλοντική Τοπολογία

Στο άμεσο μέλλον ο οργανισμός έχει προγραμματίσει να αλλάξει την τοπολογία των συστημάτων του και να μεταβεί σε μια πλήρως κατανεμημένη λειτουργία των συστημάτων του. Σε αυτήν θα υπάρχουν διάφοροι διακομιστές ιστού (web application servers) σε επιλεγμένα σημεία της Ελληνικής επικράτειας οι οποίοι θα επικοινωνούν μεταξύ τους μέσω του υπάρχοντος δικτύου χαμηλής χωρητικότητας. Οι διακομιστές (servers) θα εξυπηρετούν εφαρμογές ιστού (web applications) στις οποίες θα έχουν πρόσβαση οι χρήστες.

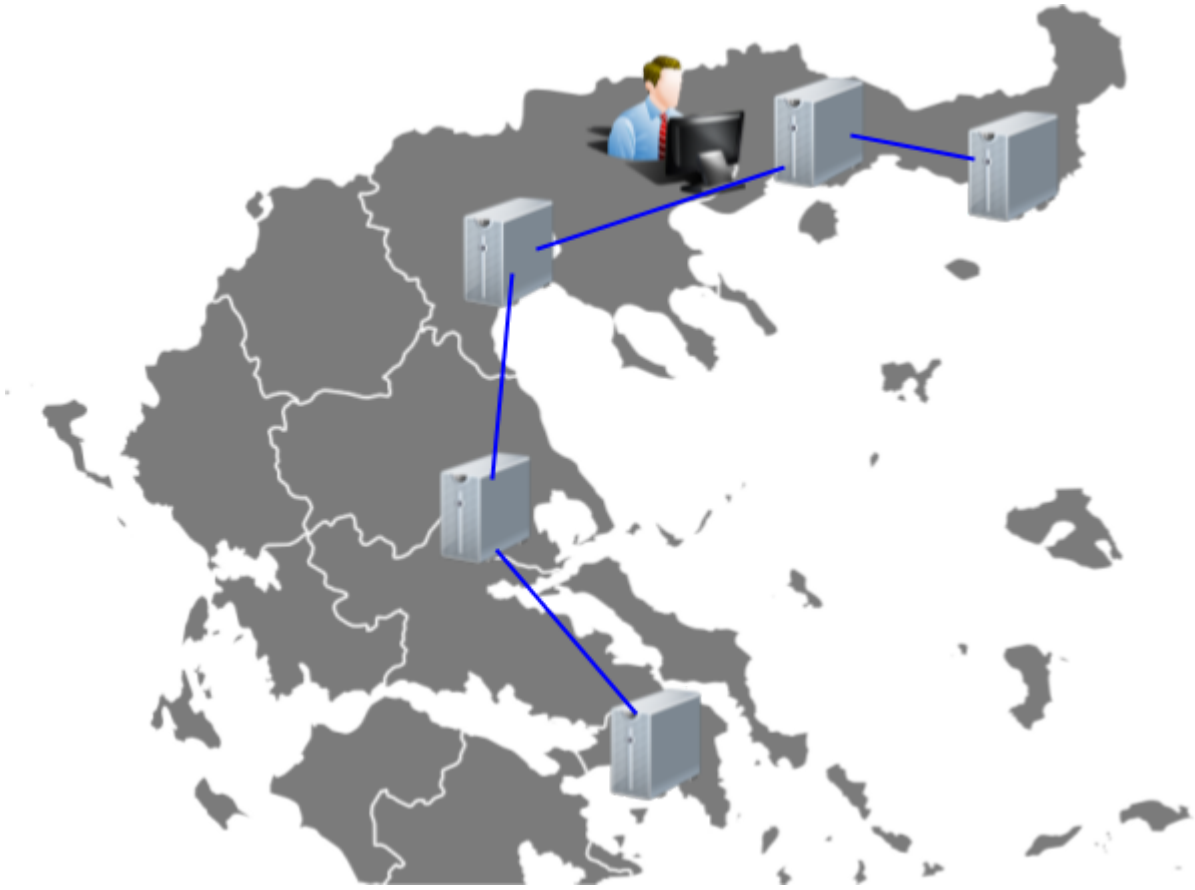


Εικόνα 2: Μελλοντική Τοπολογία οργανισμού

2.4. Σενάριο προβλήματος

Στην παρούσα ενότητα θα παρουσιαστεί το σενάριο ενός προβλήματος το οποίο θα αντιμετωπίσει ο οργανισμός στην κατανεμημένη λειτουργία των συστημάτων του.

Έστω για παράδειγμα ότι υπάρχει ένας διακομιστής ιστού (web application server) στην Αθήνα, ένας στη Λάρισα, ένας στη Θεσσαλονίκη, ένας στη Κομοτηνή και ένας στην Αλεξανδρούπολη. Υπάρχει ένας χρήστης ο οποίος βρίσκεται στην Ξάνθη και θέλει να προσπελάσει την εφαρμογή “Α” η οποία εξυπηρετείται από όλους τους προαναφερθέντες διακομιστές (servers). Το πρόβλημα έγκειται στο να επιλέξει το σύστημα ποιος είναι ο καταλληλότερος διακομιστής για να ανταποκριθεί στα αιτήματα (requests) του συγκεκριμένου χρήστη. Με τον όρο “καταλληλότερος διακομιστής” εννοείται ο διακομιστής εκείνος ο οποίος θα μπορεί να εξυπηρετήσει γρηγορότερα τα αιτήματα του χρήστη προσφέροντας ταυτόχρονα και καλύτερη ποιότητα υπηρεσιών (Quality of Service).



Εικόνα 3: Απεικόνιση του προβλήματος

2.5. Σημαντική Σημείωση

Στην μελέτη και την επίλυση του προβλήματος δεν θα ληφθεί υπόψιν ένας πολύ σημαντικός παράγοντα ο οποίος υπό κανονικές συνθήκες θα έπαιζε πάρα πολύ σημαντικό ρόλο: Ο παράγοντας των δεδομένων!

Οι εφαρμογές οι οποίες φιλοξενούνται στους κατανεμημένους διακομιστές, αλληλεπιδρούν με τη βάση δεδομένων προκειμένου να αντλήσουν πληροφορίες από αυτήν και να τις παρουσιάσουν στους χρήστες. Η επικοινωνία μεταξύ διακομιστή της εφαρμογής (web server) με τον διακομιστή της βάσης δεδομένων (database server) καθώς και η απόδοση του τελευταίου, επηρεάζει σε μεγάλο βαθμό και την απόδοση των εφαρμογών.

Στην μελέτη του παρόντος προβλήματος θα θεωρηθεί ότι οι διακομιστές ιστού (web application servers) επικοινωνούν με κάποιον τρόπο με την βάση δεδομένων, και η επικοινωνία αυτή είναι αξιόπιστη και γρήγορη.

2.6. Ανάλυση του προβλήματος

Στην παράγραφο αυτή θα γίνει η προσέγγιση του προβλήματος, προκειμένου να αναζητηθεί η βέλτιστη δυνατή λύση. Η προσέγγιση θα γίνει με όσο το δυνατόν απλούστερο τρόπο έτσι ώστε να επιτευχθεί και η βέλτιστη δυνατή λύση.

Λαμβάνοντας σαν δεδομένο τη χαμηλή χωρητικότητα του δικτύου, τα αιτήματα (requests) του χρήστη θα πρέπει να σταλούν στον διακομιστή (server) ο οποίος θα μπορέσει να τα εξυπηρετήσει γρηγορότερα. Σαν μια υπεραπλουστευμένη λύση, θα μπορούσε να επιλεχθεί ο διακομιστής εκείνος που είναι πλησιέστερα προς το χρήστη. Ωστόσο όμως αυτό μπορεί να μην είναι η ορθότερη επιλογή σύμφωνα με τους περιορισμούς που υπάρχουν στο περιβάλλον στο οποίο εξετάζεται το πρόβλημα. Αυτό συμβαίνει, διότι, σε περίπτωση που μια υπηρεσία (π.χ. μεταφορά εικόνας και ήχου) έχει καταλάβει μεγάλο μέρος της χωρητικότητας του δικτύου, τότε ο συγκεκριμένος διακομιστής δεν θα είναι ο ιδανικότερος για να ανταποκριθεί στα αιτήματα του χρήστη εξαιτίας μεγάλου φόρτου στο δίκτυο, και συνεπώς θα πρέπει να επιλεγεί ένας διακομιστής με μικρότερο φόρτο εργασιών. Έτσι λοιπόν, θα πρέπει να βρεθεί ένας αλγόριθμος ο οποίος θα γνωστοποιεί κάθε φορά ποιος είναι εκείνος ο διακομιστής (server) ο οποίο μπορεί να εξυπηρετήσει πιο γρήγορα τα αιτήματα (request) του χρήστη . Οι παράμετροι οι οποίο θα πρέπει να ληφθούν υπόψιν είναι:

1. Εάν ο διακομιστής (server) είναι σε θέση να εξυπηρετήσει τα αιτήματα των χρηστών;
2. Τι φόρτο έχει ;
3. Τι φόρτο έχει το δίκτυο μεταξύ του χρήστη και του διακομιστή;

Θα πρέπει να είναι γνωστό κάθε φορά ποιος διακομιστής είναι διαθέσιμος για να μπορέσει να δεχτεί τις αιτήσεις των χρηστών. Υπάρχουν αρκετοί λόγοι για τους οποίους έναν διακομιστής μπορεί να μην είναι σε θέση να εξυπηρετήσει αιτήματα

(request). Μπορεί για παράδειγμα ένας διακομιστής να έχει τεθεί εκτός λειτουργίας για λόγους συντήρησης. Ακόμα μπορεί να έχει υπερφορτωθεί (overloaded) και να μην είναι σε θέσει να εξυπηρετήσει άλλα αιτήματα (request). Τέλος μπορεί το δίκτυο με το οποίο συνδέεται ο χρήστης με τον διακομιστή να έχει διακοπεί και να μην είναι δυνατή η δικτυακή επικοινωνία μεταξύ χρήστη και διακομιστή (server).

Μια άλλη παράμετρος η οποίο θα πρέπει να ληφθεί υπόψιν είναι ο φόρτος που έχει ένας διακομιστής (server). Θεωρητικά όταν ένας διακομιστής έχει μεγάλο φόρτο δεν θα μπορεί να εξυπηρετήσει τόσο γρήγορα και άλλα αιτήματα (request) χρηστών. Το τελευταίο βέβαια χωράει αρκετή διερεύνηση και μπορεί να επηρεάζεται από αρκετά πράγματα. Για παράδειγμα αν ένας διακομιστής ο οποίος έχει φόρτο 30% αν μπορεί να εξυπηρετήσει πιο γρήγορα τις αιτήσεις των χρηστών σε σχέση με έναν διακομιστή ο οποίος έχει φόρτο 90% .

Τέλος θα πρέπει να είναι γνωστό ποιο είναι το “γρηγορότερο” δίκτυο μεταξύ των δικτύων που συνδέουν τον χρήστη και τον διακομιστή (server). Θα πρέπει τα αιτήματα (requests) του χρήστη να πάνε όχι στο δίκτυο με την μεγαλύτερη ονομαστική χωρητικότητα αλλά στο δίκτυο εκείνο που έχει την δυνατότητα να εξυπηρετήσει γρηγορότερα τα αιτήματα αυτά.

Για να σταλθούν όμως τα αιτήματα των χρηστών σε συγκεκριμένους διακομιστές θα πρέπει να υπάρχει και ένα μηχανισμός ο οποίος θα επιτρέπει την ανακατεύθυνση των αιτημάτων (request) αυτών. Θα πρέπει επίσης ο μηχανισμός αυτός να μπορεί να λειτουργήσει σύμφωνα με κάποια δεδομένα. Δηλαδή να βασίζεται πάνω σε κάποιες μετρήσεις και σύμφωνα με αυτές, να είναι σε θέση να μπορεί να αποφασίζει ποιος είναι ο καταλληλότερος διακομιστής (server) ,αλλάζοντας κάθε φορά την αποστολή των αιτήσεων προς τον έναν ή τον άλλο διακομιστή (server).

3. Τεχνολογίες

Σίγουρα το πρόβλημα το οποίο η παρούσα διπλωματική καλείται να επιλύσει θα έχει προβληματίσει και άλλους. Σίγουρα στον κόσμο του διαδικτύου θα έχουν υπάρξει παρόμοια προβλήματα και σίγουρα θα έχουν δοθεί λύσεις.

Παρόμοιο πρόβλημα στον χώρο του διαδικτύου λύνουν τα Δίκτυα Διανομής Περιεχομένου (Content Delivery Networks CDNs) στα οποίο θα αναφερθούμε στη συνέχεια. Ακόμα για την ανακατεύθυνση των αιτήσεων (request) των χρηστών χρησιμοποιούνται οι Αντίστροφοι Διακομιστές Μεσολάβησης (Reverse Proxies). Τέλος θα πρέπει να βρεθεί τρόπος με τον οποίο θα υπάρχει μετρήσιμο αποτέλεσμα για την απόδοση μιας εφαρμογής.

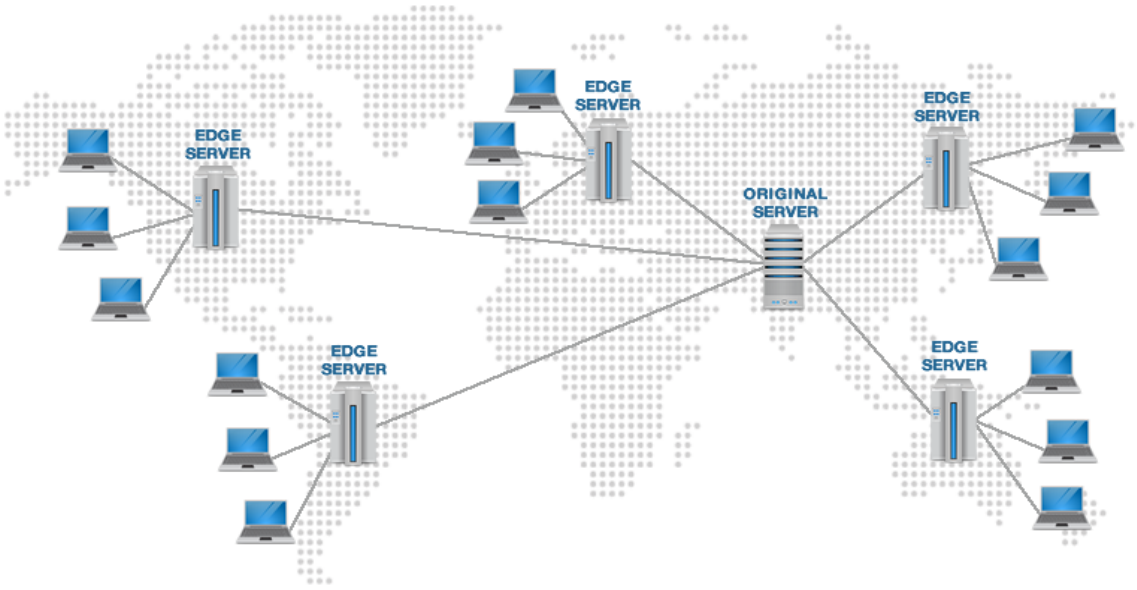
3.1. Δίκτυα Διανομής Περιεχομένου (Content Delivery Networks CDNs)

Το πρόβλημα το οποίο αναφέρθηκε στις προηγούμενες παραγράφους είναι ένα πρόβλημα το οποίο έχει απασχολήσει και τον κόσμο του διαδικτύου στο παρελθόν και φυσικά έχουν βρεθεί λύσεις οι οποίες είναι αρκετά διαδεδομένες .

Στο διαδίκτυο καταρχήν δεν υπάρχει ο περιορισμός της ταχύτητας του δικτύου. Δηλαδή θεωρείται ότι η ταχύτητα του δικτύου υπάρχει και δεν αποτελεί περιοριστικό παράγοντα. Ωστόσο στο διαδίκτυο οι αποστάσεις μπορεί να είναι πολύ μεγάλες και ο χρόνος απόκρισης μπορεί να αυξηθεί σημαντικά όταν πρέπει να προσπελαστεί ένα διακομιστής ο οποίος βρίσκεται σε διαφορετική για παράδειγμα ήπειρο . Για αυτό το λόγο υπάρχουν διάφορες τεχνικές - υλοποιήσεις οι οποίες έχουν σαν σκοπό την παροχή καλύτερης ποιότητας υπηρεσιών στους χρήστες.

Μια λύση για παράδειγμα είναι τα Δίκτυα Διανομής Περιεχομένου (Content Delivery Networks, CDN)[1] . Με τα CDNs , το περιεχόμενο κάθε διακομιστή (server)

που συμμετέχει σε αυτά διατίθεται σε πολλά αντίγραφα σε στρατηγικά τοποθετημένους διακομιστές. Αυτή η πρακτική είναι γνωστή ως αναπαραγωγή περιεχομένου (content replication.) Αντί να προβάλλετε βίντεο, ήχο, JavaScript, CSS και εικόνες από έναν διακομιστή, τα περισσότερα CDNs έχουν δεκάδες διακομιστές άκρων (edge servers) σε όλο τον κόσμο που αναπαράγουν το περιεχόμενό των κύριων διακομιστών. Με άλλα λόγια, τα CDNs εξυπηρετούν το ίδιο περιεχόμενο ανάλογα με το πού βρίσκεται ο τελικός χρήστης.



Εικόνα 4: Δίκτυο CDN

Τα CDNs σχεδιάστηκαν για να είναι αποτελεσματικά ακόμη και κατά τη διάρκεια έντονων περιόδων αιχμής ή ακόμα κατά τη διάρκεια ξαφνικών αιχμών ζήτησης. Τα περισσότερα μεγάλα CDNs αποτελούνται από χιλιάδες διακομιστές, έτσι ώστε εκατοντάδες χιλιάδες τελικοί χρήστες να μπορούν ταυτόχρονα να έχουν πρόσβαση στο ίδιο περιεχόμενο.

Το δίκτυο αυτό είναι σε θέση να μπορεί να εξακριβώσει ποια είναι η βέλτιστη θέση από την οποία πρέπει να παραδοθεί το περιεχόμενο στον τελικό χρήστη. Η ιδανική αυτή θέση είναι :

1) ο διακομιστής που απέχει τη μικρότερη απόσταση από το χρήστη που έκανε την αίτηση για περιεχόμενο (request).

2) ο διακομιστής που απέχει τα λιγότερα δευτερόλεπτα από τον τελικό χρήστη ή

3) ο διακομιστής με την υψηλότερη διαθεσιμότητα όσον αφορά τις επιδόσεις του διακομιστή. Για παράδειγμα, ο διακομιστής με τις λιγότερες αναπηδήσεις δικτύου ή ο διακομιστής με την ταχύτερη ανταπόκριση επιλέγετε για να μεταδώσει το περιεχόμενο.[2]

Στόχος των CDNs είναι η παροχή του περιεχομένου στους τελικούς χρήστες με υψηλές επιδόσεις, διαθεσιμότητα και βέλτιστη απόδοση. Επίσης λόγο του ότι το περιεχόμενο διαμοιράζεται στους χρήστες από διαφορετικούς διακομιστές (servers), δεν περνάει όλη η κίνηση από τον κεντρικό διακομιστή της ιστοσελίδας και αυτό έχει ως αποτέλεσμα να επιτυγχάνεται offloading (εκφόρτωση), που πρακτικά σημαίνει εξοικονόμηση πόρων, άρα και χρημάτων.

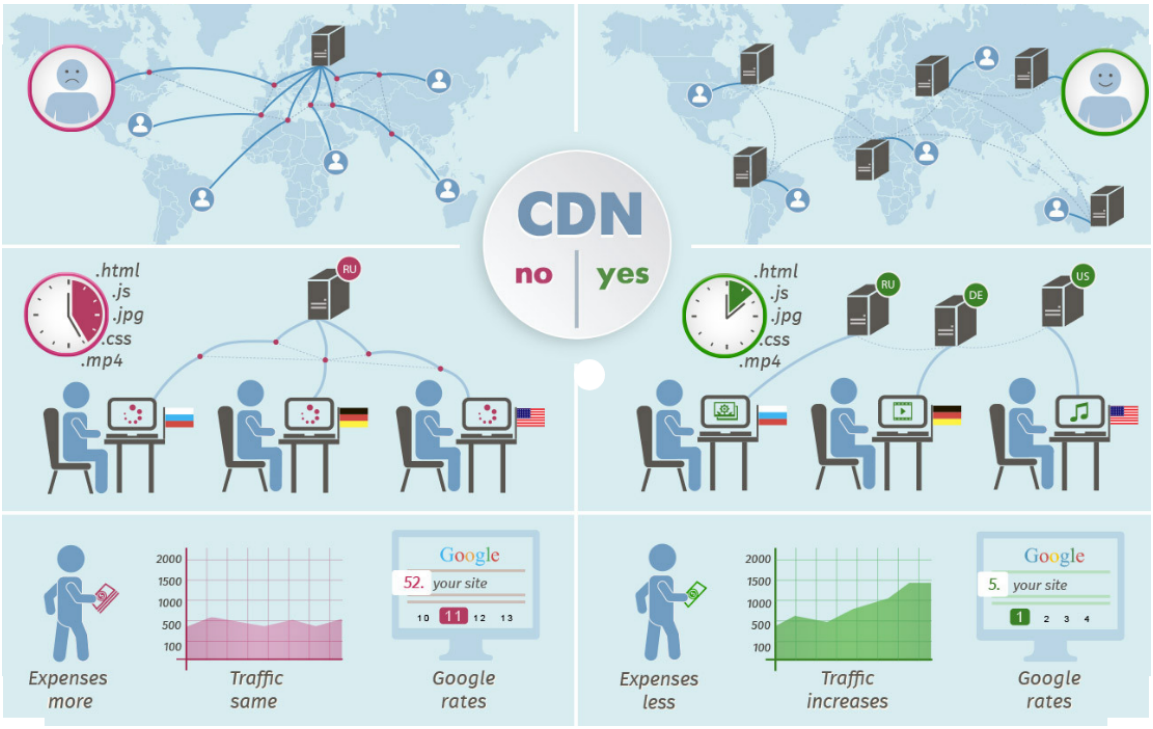
Πρέπει επίσης να επισημανθεί ότι με τα CDNs δεν εξασφαλίζεται μόνο η γρήγορη πρόσβαση σε μια ιστοσελίδα ή ηλεκτρονική υπηρεσία αλλά και κατά κάποιον τρόπο παρέχεται ασφάλεια απέναντι στις διαδικτυακές επιθέσεις, οι λεγόμενες επιθέσεις άρνησης υπηρεσιών (Denial of Service DoS). Αυτές στοχεύουν στην υπερφόρτωση του διακομιστή με σκοπό την κατάρρευση του, καθώς θα αδυνατεί να ανταποκριθεί ορθά σε πληθώρα αιτημάτων (requests). Στην περίπτωση αυτή με την χρήση των CDNs και χάρη στην εύστοχη λειτουργία τους ο φόρτος εργασιών που θα προκληθεί από την επίθεση δεν θα βγάλει εκτός λειτουργίας τον κεντρικό διακομιστή, αφού ο μεγάλος φόρτος αιτημάτων (request) που θα δημιουργηθεί από μια τέτοιου είδους επίθεση θα κατανεμηθεί στους διακομιστές του CDN. Έτσι μια τέτοιου είδους επίθεση δεν θα κατάφερνε να επηρεάσει τη σωστή και ομαλή λειτουργία ενός υπολογιστικού συστήματος.[3]

Τα CDNs είναι ένα παγκόσμιο κατανεμημένο δίκτυο το οποίο είναι εγκατεστημένο μέσα σε όλο το διαδίκτυο. Αποτελείται από γεωγραφικά κατανεμημένους διακομιστές προσωρινής μνήμης (cache servers) παραδίδοντας

προσωρινό περιεχόμενο σε πελάτες παγκοσμίως με βάση τη γεωγραφική τους τοποθεσία. Σε αντίθεση με την γενικότερη φιλοσοφία του υπόλοιπου διαδικτύου, όπου λειτουργεί με την λογική της καλύτερης προσπάθειας (best effort), το περιεχόμενο που παρέχεται μέσω CDN έχει χαμηλή λανθάνουσα κατάσταση (latency) και υψηλή αξιοπιστία και παρέχει την καλύτερη ποιότητα εμπειρίας (Quality of Experience QoE).

Σήμερα, τα CDN εξυπηρετούν ένα μεγάλο μέρος της κίνησης του διαδικτύου. Για παράδειγμα, τα βίντεο που παρέχονται από τα CDNs του Netflix αντιπροσωπεύει το 37% της συνολικής κίνησης του διαδικτύου στη Βόρεια Αμερική κατά τις ώρες αιχμής. Ο κορυφαίος πάροχος υπηρεσιών CDN , η εταιρία Akamai, ισχυρίζεται ότι παρέχει περισσότερο από το 20% της κίνησης του Internet παγκοσμίως [20]

Λόγω του σημαντικού ρόλου του CDNs στο σημερινό οικοσύστημα του Διαδικτύου, εταιρείες όπως η Google, η Microsoft και η Amazon, καθώς και υπηρεσίες παροχής Internet (ISP) όπως η AT&T, η Orange, Swisscom και KT, σχεδιάζουν ή έχουν ήδη ξεκινήσει τα δικά τους CDNs.[20]



Εικόνα 5: Τα CDN με μια ματιά

3.1.1. Δομικά στοιχεία των CDNs

3.1.1.1. Σημεία παρουσίας (Point of Presence)

Τα σημεία παρουσίας των CDNs (PoPs) είναι στρατηγικά τοποθετημένα κέντρα δεδομένων που είναι υπεύθυνα για την επικοινωνία με τους χρήστες σύμφωνα με την γεωγραφική τους γειτνίαση. Βασική λειτουργία τους είναι να μειώσουν το χρόνο ταξιδιού κάποιου αιτήματος (request) , φέρνοντας το περιεχόμενο πιο κοντά στον επισκέπτη του ιστότοπου. Κάθε PoP CDN περιέχει τυπικά πολλούς διακομιστές προσωρινής αποθήκευσης (caching servers).

3.1.1.2. Διακομιστές προσωρινής αποθηκευσης (Caching Servers)

Οι διακομιστές προσωρινής αποθήκευσης είναι υπεύθυνοι για την αποθήκευση και την παράδοση των προσωρινά αποθηκευμένων αρχείων. Η κύρια λειτουργία τους είναι να επιταχύνουν τους χρόνους φόρτωσης του ιστότοπου και να μειώνουν την κατανάλωση εύρους ζώνης (bandwidth). Κάθε διακομιστής προσωρινής αποθήκευσης (cache server) του CDN διατηρεί συνήθως πολλαπλές μονάδες αποθήκευσης και μεγάλες ποσότητες πόρων σε μνήμη RAM.

3.1.1.3. Αποθηκευτικοί χώροι προσωρινής και μόνιμης αποθήκευσης

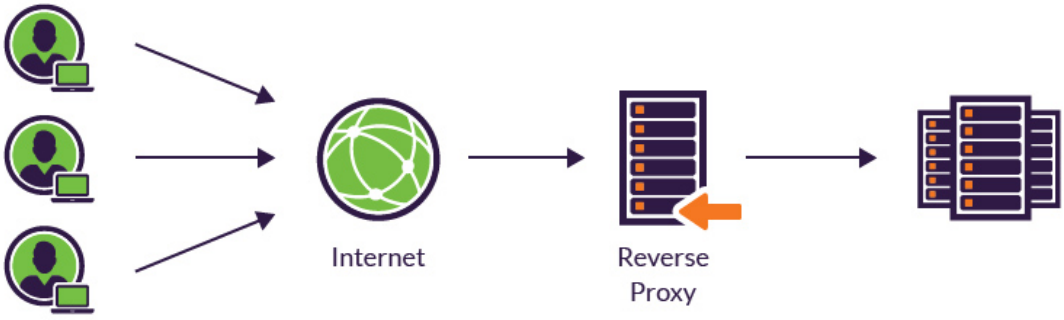
Μέσα στους διακομιστές προσωρινής αποθήκευσης (cache servers) των CDNs, τα αποθηκευμένα αρχεία αποθηκεύονται σε μονάδες Solid State Disk (SSD) και σκληρού δίσκου (Hard Disk Drive HDD) ή σε μνήμη τυχαίας προσπέλασης (Random Access Memory RAM), με τα πιο συχνά χρησιμοποιούμενα αρχεία που φιλοξενούνται στα πιο γρήγορα μέσα. Όντας ο ταχύτερος από τους τρεις, η μνήμη RAM χρησιμοποιείται συνήθως για την αποθήκευση των πιο συχνά προσπελάσιμων στοιχείων.

3.1.2. Τεχνικές δικτύων διανομής περιεχομένου

Τα δίκτυα διανομής περιεχομένου για να καταφέρουν να πραγματοποιήσουν τις κύριες διαδικασίες τους, βασίζονται στις παρακάτω τεχνικές.

3.1.2.1. Αντίστροφοι Διακομιστες Μεσολάβησης (Reverse Proxy Server)

Τα CDNs χρησιμοποιούν τους αντίστροφους διακομιστές μεσολάβησης (reverse proxies) μπροστά από τους διακομιστές άκρων (edge servers). Ένας αντίστροφος διακομιστής μεσολάβησης είναι ένα ενδιάμεσο σημείο σύνδεσης τοποθετημένο στην άκρη του δικτύου. Λαμβάνει αρχικά αιτήματα σύνδεσης HTTP, και τα ανακατευθύνει στον πραγματικό διακομιστή ή σε μια ομάδα διακομιστών. Ο reverse proxy χρησιμεύει ως πύλη μεταξύ των χρηστών και του διακομιστή [5].

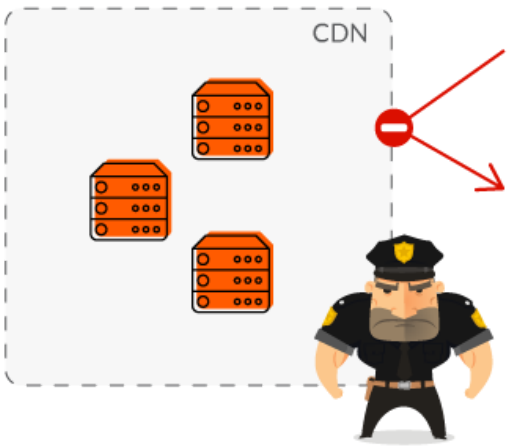


Εικόνα 6: Reverse Proxy

Οι αντίστροφοι διακομιστές μεσολάβησης χρησιμοποούνται για :

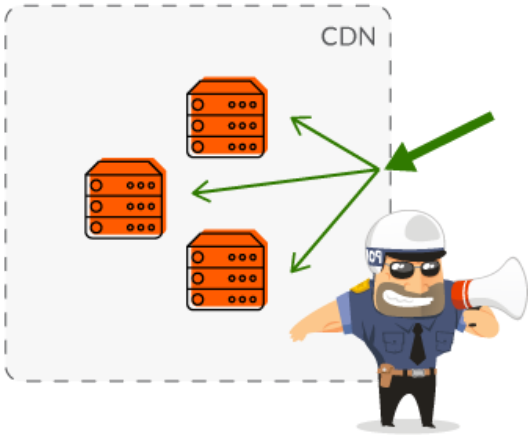
- Ασφάλεια Ιστότοπου (WebSite Security)

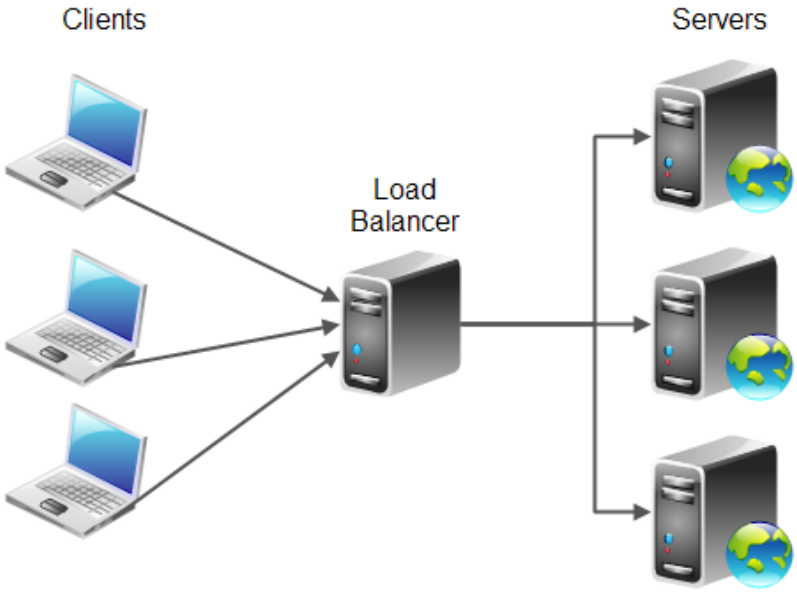
Η ασφάλεια στο διαδίκτυο έχει να κάνει με τη διαχείριση της εξωτερικής πρόσβασης στην “προστατευμένη περιοχή” του διακομιστή. Ελέγχοντας έτσι την κίνηση στους διακομιστές άκρων (edge servers) ,ένα CDN μπορεί να αποτρέπει επιθέσεις στο διακομιστή και κατ’ επέκταση στην ίδια την εφαρμογή. Η δυνατότητα αυτή καθιστά τα CDN ιδανικά για την παρεμπόδιση κατανεμημένων επιθέσεων άρνησης υπηρεσιών (DDoS), οι οποίες πρέπει να μετριάζονται εκτός της υποδομής του κεντρικού δικτύου.



- Κατανομή φόρτου (Load Balancing)

Στην κατανομή φόρτου διαμοιράζουμε τις αιτήσεις των χρηστών σε περισσότερους διακομιστές ώστε να μην έχουμε υπερφόρτωση . Το θετικό εδώ είναι ότι κάνει το σύστημα πιο αποδοτικό αφού παίρνει τον ίδιο χρόνο στις περισσότερες εργασίες μέχρι να ολοκληρωθούν, άρα ολοκληρώνεται πιο γρήγορα και η όλη διαδικασία. Γενικά όλες οι αιτήσεις των χρηστών περνούν μέσα από τον αντίστροφο διακομιστή μεσολάβησης (reverse proxy) και στην συνέχεια αυτός τις διαμοιράζει στους διακομιστές σύμφωνα με διάφορους αλγορίθμους λαμβάνοντας διάφορους παραμέτρους κάθε φορά και σύμφωνα με την ρύθμιση που έχουν κάνει οι διαχειριστές του proxy.[6] Στον αντίστροφο διακομιστή μεσολάβησης (reverse proxy) δεν γίνεται καμία επεξεργαστική διαδικασία. Απλά ανακατευθύνει το αιτήματα (request) του χρήστη προς έναν άλλον διακομιστή .

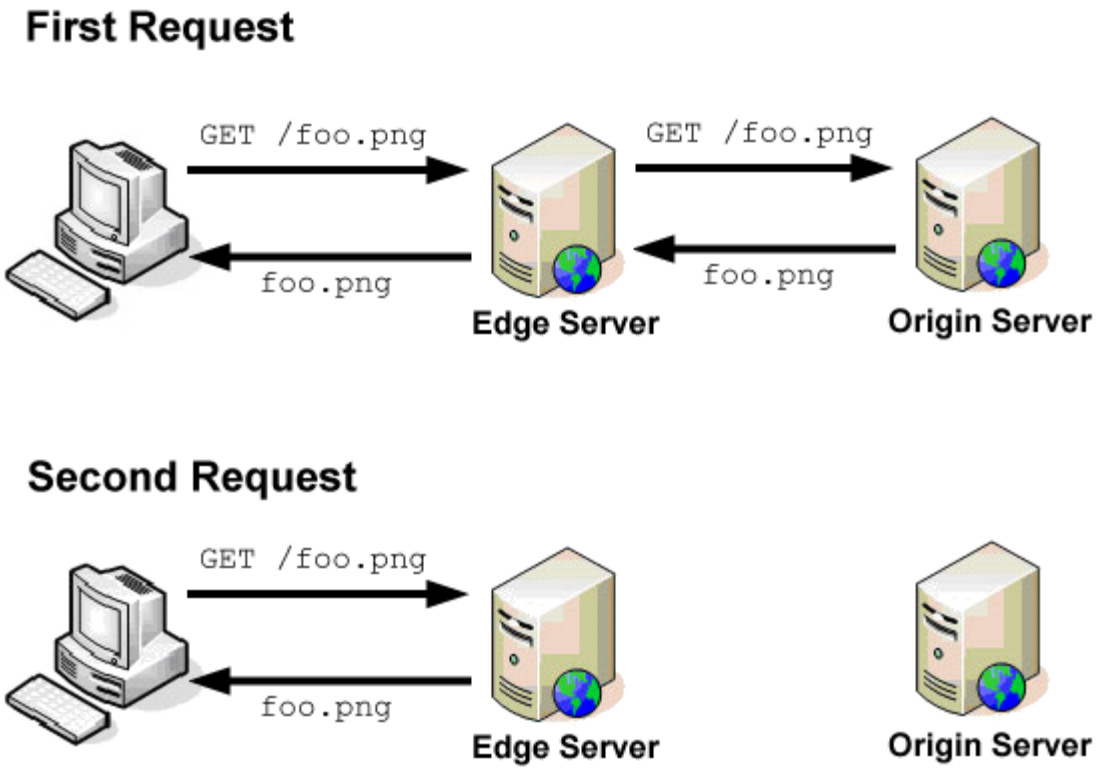




Εικόνα 7: Load Balancing

3.1.2.2. Προσωρινή αποθήκευση (Caching)

Γενικά η τεχνική του προσωρινής αποθήκευσης (caching) είναι αρκετά διαδεδομένη στις τεχνολογίες ιστού και όχι μόνο. Τα CDNs αποθηκεύουν το δημοφιλές περιεχόμενο σε διακομιστές που έχουν τη μεγαλύτερη ζήτηση για αυτό το περιεχόμενο και όταν κάποιος χρήστης το αναζητήσει τότε ο cache server θα το προωθήσει προς τον χρήστη χωρίς να χρειάζεται να φτάσει η αίτηση του χρήστη στον διακομιστή.



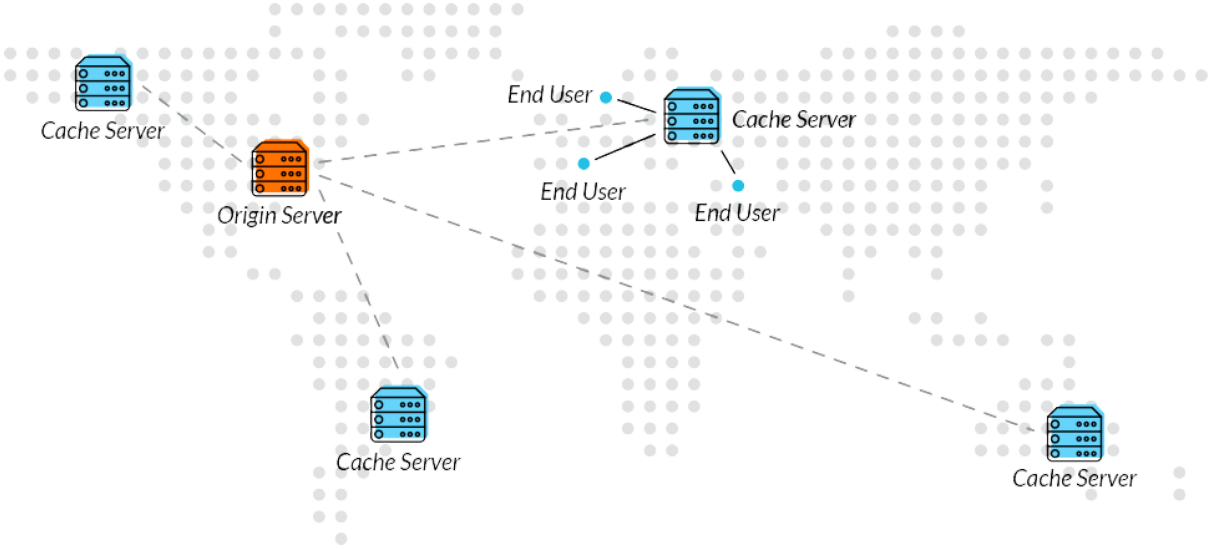
Εικόνα 8: Caching [8]

Τι γίνεται όμως στην περίπτωση όπου το περιεχόμενο το οποίο υπάρχει στον διακομιστή προσωρινής αποθήκευση δεν είναι πια έγκυρο ; Ο μηχανισμός προσωρινής αποθήκευσης το έχει λύσει και αυτό. Κάθε πληροφορία που κρατάει ο cache server έχει μια “ημερομηνία λήξης”. Άμα αυτή η ημερομηνία έχει παρέλθει τότε το αίτημα του χρήστη θα προχωρήσει κανονικά προς τον κανονικό διακομιστή. Γενικώς σκοπός αυτού του μηχανισμού είναι να γίνει αποτελεσματική και αποδοτική η χρήση των πόρων και να αυξηθεί η ταχύτητα μεταφοράς του περιεχομένου στον

τελικό χρήστη. Γενικά το caching είναι μια τεχνική η οποία έχει υιοθετηθεί από πολλές τεχνολογίες στον τομέα της πληροφορικής και έχει σαν σκοπό την [7] :

- Βελτίωση της ταχύτητας (Speed improvement)
- Μεγαλύτερη ανοχή σφαλμάτων (Fault tolerance)
- Επεκτασιμότητα (Scalability)
- Μείωση φόρτου Διακομιστή (Reduce Server Load)

Στην εικόνα 8 φαίνεται χαρακτηριστικά ο τρόπος με τον οποίο λειτουργεί το caching. Αναλύοντας λίγο την εικόνα παρατηρείται ότι το πρώτο αίτημα (request) του χρήστη θα φτάσει μέχρι τον τελικό διακομιστή . Στο δεύτερο αίτημα (request) ο cache server θα εξυπηρετήσει εκείνος το αίτημα (request) του χρήστη αφού έχει την πληροφορία την οποία ζήτησε ο χρήστης. Στην εικόνα 9 φαίνεται η τοπολογία του cache server του τελικού χρήστη και του κεντρικού διαχειριστή.



Εικόνα 9: Cache server [9]

3.1.2.3. Αντιγραφή Περιεχομένου (Content Replication)

Αυτή η τεχνική στα δίκτυα διανομής περιεχομένου έχει ως στόχο να δημιουργεί πολλά αντίγραφα της ίδιας πληροφορίας, με σκοπό να βελτιώσει την ταχύτητα πρόσβασης στην πληροφορία αυτή, να μειώσει τις καθυστερήσεις και άρα να αυξήσει τη διαθεσιμότητα αυτής. Αντί για παράδειγμα να είναι διαθέσιμο ένα βίντεο ή ένα αρχείο ήχου ή εικόνας ή ακόμα και κώδικας JavaScript και CSS από

τον κεντρικό διακομιστή, τα περισσότερα CDNs έχουν δεκάδες διακομιστές άκρων (edge servers) σε όλο τον κόσμο που αναπαράγουν το περιεχόμενα αυτά. Με άλλα λόγια, τα CDNs εξυπηρετούν το ίδιο περιεχόμενο ανάλογα με το πού βρίσκεται ο τελικός χρήστης[1].

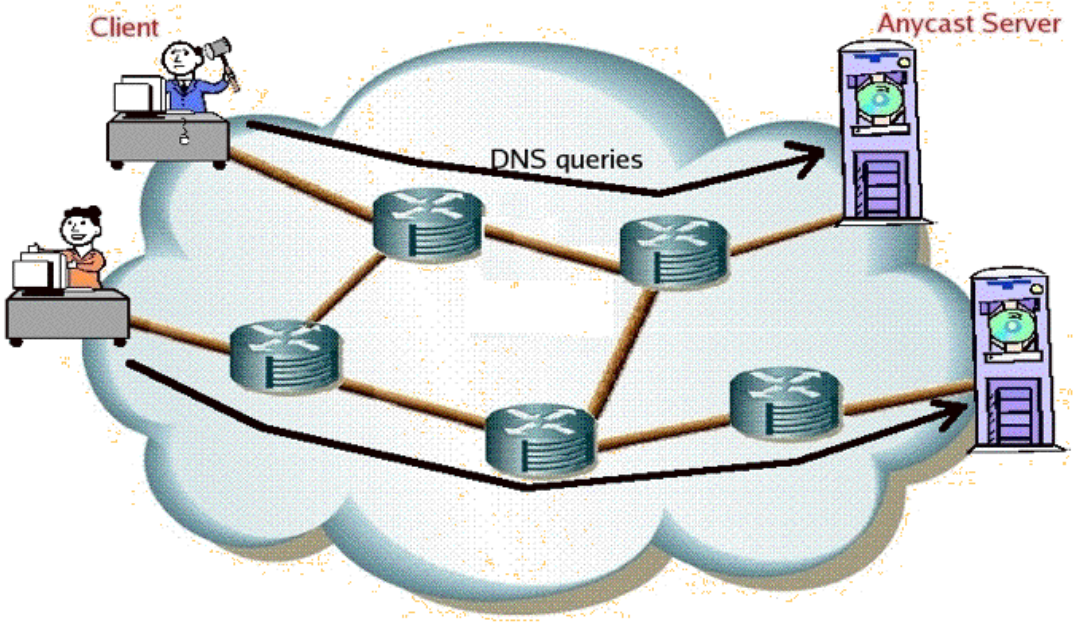
Χρησιμοποιώντας αυτή την τεχνική αντιγράφονται τα περιεχόμενα των κεντρικών διακομιστών στους διακομιστές άκρων του δικτύου (edge servers). Έτσι λοιπόν αν ένας κεντρικός διακομιστής αλλάξει το περιεχόμενο του τότε ενημερώνεται και το περιεχόμενο των διακομιστών άκρων. Όπως αναφέρθηκε και πριν, αυξάνεται γενικότερα η αποδοτικότητα του δικτύου, αφού οι αιτήσεις δεν πάνε στον κεντρικό διακομιστή με το γνήσιο περιεχόμενο, αλλά στον διακομιστή άκρων (edge server).[3]

3.1.2.4. Anycasting

Το Anycast είναι μια μέθοδος διευθυνσιοδότησης και δρομολόγησης δικτύου στην οποία τα αιτήματα από έναν αποστολέα δρομολογούνται σε οποιονδήποτε από τους διάφορους κόμβους προορισμού, οι οποίοι επιλέγονται με βάση κάποιων κριτηρίων. Τα κριτήρια αυτά μπορεί να είναι βάση του πλησιέστερου διακομιστή , του διακομιστή με το χαμηλότερο κόστος, του διακομιστή όπου φαίνεται να είναι πιο “υγιείς”, το διακομιστή με το μικρότερο φόρτο ή με κάποια άλλη μετρική[11] .

Μερικά από τα πλεονεκτήματα του Anycasting είναι:[10]

- Ταχύτερες συνδέσεις (Faster Connections)
- Απλοποίηση των ρυθμήσεων του διακομιστή (server)
- Αυξημένη διαθεσιμότητα (High Availability)
- Προστασία κατανεμημένων επιθέσεων άρνησης υπηρεσιών (DDoS)



Εικόνα 10: Anycast DNS [12]

3.1.2.5. Συμπίεση HTTP

Γενικά με τον όρο συμπίεση δεδομένων (data compression) εννοούμε τη μετατροπή ενός ψηφιακού αρχείου (για αποστολή ή αποθήκευση) σε μικρότερο αρχείο (που περιέχει μικρότερο αριθμό bit) με τρόπο ώστε να είναι δυνατή η επαναμετατροπή του συμπιεσμένου αρχείου στο αρχικό[13]. Αυτό επιτυγχάνεται με την χρήση διάφορων αλγορίθμων .

Η συμπίεση στο επίπεδο του HTTP πρωτοκόλλου είναι η δυνατότητα που μπορεί να ενσωματωθεί σε διακομιστές ιστού και πελάτες ιστού για τη βελτίωση της ταχύτητας μεταφοράς και της χρήσης του εύρους ζώνης (bandwidth). Τα δεδομένα HTTP συμπιέζονται πριν αποσταλούν από το διακομιστή ή σε κάποιον ενδιάμεσο αντίστροφο διακομιστή μεσολάβησης (reverse proxy) ανάλογα με την τοπολογία την οποία έχουν εφαρμόσει οι σχεδιαστές των συστημάτων. Τα συμβατά προγράμματα περιήγησης ιστού (browsers) θα ανακοινώνουν τις μεθόδους συμπίεσης που υποστηρίζονται στον διακομιστή πριν από τη λήψη ώστε ο τελευταίος να στείλει τα

δεδομένα στην σωστή μορφή. Τα προγράμματα περιήγησης που δεν υποστηρίζουν τη συμβατή μέθοδο συμπίεσης θα φορτώσουν τα δεδομένα ασυμπίεστα. Οι πιο διαδεδομένοι μέθοδοι συμπίεσης είναι το gzip και το Deflate, ωστόσο ο πλήρης κατάλογος των διαθέσιμων μεθόδων συμπίεσης διατηρείται από το IANA[15].

Σε αυτό το σημείο αξίζει να αναφερθεί και ο αλγόριθμος Brotli ο οποίος είναι ένας νέος αλγόριθμος συμπίεσης βελτιστοποιημένος για τον ιστό. Προτάθηκε από την Google το 2015 [17], είναι ιδιαίτερα αποδοτικός σε έγγραφα μικρού κειμένου. Η αποσυμπίεση του Brotli είναι όση γρήγορη όσο για το gzip ενώ βελτιώνει σημαντικά την αναλογία συμπίεσης. Είναι όμως πιο αργός στην συμπίεση από ότι του gzip. Συνεπώς, ο Brotli είναι πιο αποτελεσματικός για την εξυπηρέτηση στατικού περιεχομένου, όπως γραμματοσειρές και σελίδες html. [18]

Στις περισσότερες περιπτώσεις η διαπραγμάτευση για την συμπίεση που θα χρησιμοποιήσουν ο διακομιστής (server) και το πρόγραμμα περιήγησης ιστού (browser) γίνεται σε δύο βήματα. Πρώτο το πρόγραμμα περιήγησης ιστού δημοσιοποιεί τις μεθόδους συμπίεσης που υποστηρίζει ενσωματώνοντάς τα στο HTTP Request [14].

```
GET /encrypted-area HTTP/1.1
Host: www.example.com
Accept-Encoding: gzip, deflate
```

Εικόνα 11: Browser Accepted compression[14]

Εάν ο διακομιστής υποστηρίζει ένα ή περισσότερα σχήματα συμπίεσης, τα εξερχόμενα δεδομένα ενδέχεται να συμπιεστούν με μία ή περισσότερες μεθόδους που υποστηρίζονται από αμφότερα και από τους δύο. Σε αυτή την περίπτωση, ο διακομιστής θα προσθέσει ένα πεδίο Content-Encoding ή Transfer-Encoding στην απάντηση HTTP με τις χρησιμοποιούμενες μεθόδους, χωρισμένες με κόμματα.

```
HTTP/1.1 200 OK
Date: mon, 26 June 2016 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
```

Εικόνα 12: HTTP Response με τις χρησιμοποιούμενες μεθόδους[14]

Η συμπίεση γενικά είναι ένας σημαντικός τρόπος για να αυξηθεί η απόδοση μιας εφαρμογής ιστού (web application). Σε ορισμένες περιπτώσεις, η μείωση του μεγέθους μπορεί να φτάσει έως και 70% μειώνοντας κατά πολύ τις ανάγκες χωρητικότητας εύρους ζώνης (bandwidth) καθώς και τον χρόνο απόκρισης (responce time).

3.1.3. Οι τέσσερις πυλώνες της σχεδίασης των CDNs

3.1.3.1. Απόδοση (Performance)

Μία από τις κύριες αποστολές των CDNs είναι να ελαχιστοποιήσουν την καθυστέρηση. Από αρχιτεκτονικής άποψη, αυτό σημαίνει ότι πρέπει οι διακομιστές άκρων να τοποθετηθούν σε στρατηγικά σημεία ώστε να είναι πιο αποδοτικοί και να μπορούν να εξυπηρετήσουν μεγαλύτερο αριθμό χρηστών. Έτσι θα πρέπει να βρίσκονται σε μεγάλες “διασταυρώσεις” κόμβων δικτύωσης εκεί όπου ταξιδεύουν ο μεγαλύτερος όγκος των δεδομένων.

3.1.3.2. Αξιοπιστία (Reliability)

Το μέγεθος της υποδομής ενός δικτύου CDN καθιστά ένα σύστημα χωρίς σφάλματα στατιστικά απίθανο. Ωστόσο, η ίδια κλίμακα μπορεί να βοηθήσει στην εξασφάλιση της ανθεκτικότητας των πόρων και της υψηλής διαθεσιμότητας,

επιτρέποντας στους παρόχους CDN να δεσμευτούν σε συμφωνίες επιπέδου υπηρεσιών 99,9% και 99,999% (Service Level Agreements SLA).

Κατά κανόνα, τα εμπορικά CDNs υιοθετούν μια προσέγγιση "χωρίς μοναδικό σημείο αποτυχίας", τόσο με τη προσεκτική διατήρηση των κύκλων συντήρησης όσο και με την ενσωμάτωση επιπλέον πλεονάζοντος υλικού και λογισμικού. Πολλοί επίσης πάροχοι CDNs διαθέτουν συστήματα που μπορούν να ανακάμψουν από κάποια αποτυχία (failover και disaster recovery) που αυτοματοποιούν την δικτυακή κυκλοφορία γύρω από κατεστραμμένους κόμβους. Για επιπλέον εξασφάλιση , οι πάροχοι υπηρεσιών CDN συνάπτουν συμφωνίες με μεγάλους δικτυακούς παρόχους και εξασφαλίζουν αποκλειστικά κανάλια διαχείρισης που τους επιτρέπουν να αλληλεπιδρούν με τους διακομιστές τους σε περίπτωση καταστροφής.

3.1.3.3. Επεκτασιμότητα (Scalability)

Υλοποιημένα για δρομολόγηση υψηλής ταχύτητας και μεγάλου όγκου, τα CDNs αναμένεται να μπορούν να χειριστούν οποιαδήποτε κίνηση όσο μεγάλη και αν είναι αυτή. Η αρχιτεκτονική των CDNs θα πρέπει να αντιμετωπίσει αυτές τις προκλήσεις, παρέχοντας άφθονους πόρους δικτύωσης και επεξεργασίας σε όλα τα επίπεδα - κάτω από τους πόρους υπολογιστών και προσωρινής αποθήκευσης που είναι διαθέσιμοι σε κάθε έναν από τους διακομιστές προσωρινής αποθήκευσης (cache servers).

Όπως θα περίμενε κανείς, τα CDN που προσφέρουν υπηρεσίες προστασίας επιθέσεων κατανεμημένης άρνησης υπηρεσιών (Distributed Denial of Sevice DDoS) έχουν πολύ υψηλότερες απαιτήσεις κλιμάκωσης. Για την αντιμετώπιση αυτών των αναγκών, αναπτύσσουν αποκλειστικούς διακομιστές που είναι κατασκευασμένοι για μετριασμό του DDoS (scrubbers). Αυτά μπορούν να χειρίζονται μεγάλες ποσότητες κίνησης δικτύου, επεξεργάζοντας δεκάδες gigabytes το κάθε δευτερόλεπτο .

3.1.3.4. Απόκριση στις Αλλαγές (Responsiveness)

Με ένα δίκτυο παγκόσμιου μεγέθους, τα CDNs συνεχώς επιδιώκουν να βελτιώσουν την ταχύτητα απόκρισης σε οποιαδήποτε αλλαγή γίνει σε αυτά. Προσπαθούν δηλαδή να μειώσουν τον χρόνο που χρειάζεται για να ενημερωθούν όλοι οι κόμβοι του δικτύου για οποιαδήποτε αλλαγή.

Αν λάβει κανείς υπόψιν του ότι ακόμη και μικρές αλλαγές σε κάποια ρύθμιση, όπως μια εντολή για την διαγραφή μιας συγκεκριμένης εικόνας από την προσωρινή μνήμη ή τη προσθήκη μιας διεύθυνσης ip σε μία λίστα απαγόρευσης (black list), θα πρέπει να γίνουν γνωστές σε όλους τους διακομιστές άκρων (edge servers). Όσο πιο γεωγραφικά απλωμένο είναι το δίκτυο των CDNs, τόσο πιο δύσκολο είναι να ενημερωθούν με γρήγορο και αποδοτικό τρόπο όλα οι κόμβοι του δικτύου.

3.1.4. Αρχιτεκτονική Συστημάτων CDNs

Οι διάφοροι πάροχοι έχουν αναπτύξει διαφορετικές προσεγγίσεις στις αρχιτεκτονικές των CDNs δικτύων που παρέχουν. Τα δύο κυρίαρχα μοντέλα των τοπολογιών ανάπτυξης είναι διεσπαρμένα (scattered) και ενοποιημένα (consolidated) CDNs. Κάθε ένα προσφέρει πολλά κοινά οφέλη μαζί με μερικά ξεχωριστά πλεονεκτήματα και μειονεκτήματα[19] .

3.1.4.1. Διεσπαρμενη Αρχιτεκτονική CDNs (Scattered CDNs)

Τα διεσπαρμένα CDNs λειτουργούν με μεγάλο αριθμό μέτριων και χαμηλής χωρητικότητας διακομιστών άκρων (edge servers), τα οποία γίνονται πιο πυκνά σε επιλεγμένες γεωγραφικές περιοχές. Αυτή η τοπολογία εστιάζει στη βέλτιστη φυσική εγγύτητα. Συνεπώς, δεν είναι ασυνήθιστο να βρεθούν οι διακομιστές άκρων (edge servers) τοποθετημένοι πολύ κοντά ο ένας στον άλλο - συχνά όχι περισσότερο από μερικές δεκάδες χιλιόμετρα μακριά.

Τα πρώιμα CDN, που αναπτύχθηκαν κατά τη διάρκεια μιας μεταβατικής περιόδου μεταξύ καλωδίων χαλκού και οπτικών ινών, βασίστηκαν στο διασκορπισμένο μοντέλο.

Με τον καιρό, καθώς εγκαταστάθηκαν περισσότερα καλώδια οπτικών ινών - και καθώς η παγκόσμια συνδεσιμότητα συνέχισε να βελτιώνεται - το οριακό όφελος της ελαχιστοποίησης της φυσικής απόστασης στους διακομιστές συνέχισε να μειώνεται. Επιπλέον, καθώς τα CDNs συνέχισαν να εισάγουν περισσότερα χαρακτηριστικά προσαρμογής, διαπιστώθηκε επίσης ότι η διάσπαρτη τοπολογία εμποδίζει την απόκριση των συστημάτων.

Ακόμα και σήμερα, η εγγύτητα εξακολουθεί να έχει σημασία. Τα διασκορπισμένα CDN παρέχουν πρόσθετη βελτίωση ταχύτητας, ειδικά σε περιοχές με χαμηλή συνδεσιμότητα . Επιπροσθέτως, οι μικρότερης χωρητικότητας διακομιστές άκρων (edge servers) διευκολύνουν την ανάπτυξη, και επιτρέπουν την ταχεία ανάπτυξη της κάλυψης του δικτύου[19].

Εικόνα 13: Διεσπαρμένη Αρχιτεκτονική CDNs[19]

3.1.4.2. Ενοποιημένη Αρχιτεκνονική CDNs (Consolidated CDN)

Τα ενοποιημένα CDNs λειτουργούν με ένα μικρό αριθμό διακομιστών άκρων (edge servers) μεγάλης χωρητικότητας, τα οποία είναι στρατηγικά τοποθετημένα σε μεγάλα κέντρα δεδομένων (data centers), εξυπηρετώντας ένα ευρύτερο πληθυσμό. Αυτή η τοπολογία δικτύου αντιπροσωπεύει μια πιο σύγχρονη προσέγγιση για την παροχή περιεχομένου που κατέστη δυνατή από την εξέλιξη του διαδίκτυο.

Το κύριο πλεονέκτημα μιας ενοποιημένης τοπολογίας είναι η κεντρική υποδομή της, η οποία επιτρέπει ευέλικτη διαχείριση και γρήγορη εγκατάσταση. Αυτό ωφελεί τόσο τους τελικούς χρήστες όσο και τον διαχειριστή του δικτύου, προσφέροντας μεγαλύτερο έλεγχο και καλύτερη συνολική απόκριση. Επιπλέον, οι διακομιστές άκρων υψηλής χωρητικότητας είναι πιο ανθεκτικοί, ειδικά όταν πρόκειται για μετριασμό επιθέσεων DDoS.

Αντίθετα, μια ενοποιημένη τοπολογία αποδείχθηκε λιγότερο αποτελεσματική στις περιοχές χαμηλής συνδεσιμότητας. Οι διακομιστές άκρων μεγάλης χωρητικότητας απαιτούν πιο πολύπλοκη ανάπτυξη, εμποδίζοντας τις γρήγορες επεκτάσεις του δικτύου[19].

Εικόνα 14: Ενοποιημένη Αρχιτεκτονική CDNs[19]

3.2. Αντίστροφοι Διακομιστές Μεσολάβησης (Reverse Proxies)

Ένας αντίστροφος διακομιστής μεσολάβησης (reverse proxy) HTTP είναι μια εφαρμογή μεσολάβησης που αναπτύσσεται κοντά σε έναν ή περισσότερους διακομιστές ιστού (web servers). Ο διακομιστής μεσολάβησης παρακολουθεί αιτήσεις (request) HTTP για τους διακομιστές ιστού, εκτελεί κάποια πολύ μικρή επεξεργασία και στη συνέχεια διαβιβάζει το αίτημα σε έναν από τους διακομιστές ιστού. Συνήθως, οι αντίστροφοι διακομιστές μεσολάβησης χρησιμοποιούνται για εξισορρόπηση φορτίου (Load Balancing) μεταξύ πολλών διακομιστών ή για αύξηση της απόδοσης με προσωρινή αποθήκευση στατικού περιεχομένου (caching)[22] Ένας αντίστροφος διακομιστής μεσολάβησης (reverse proxy) δέχεται ένα αίτημα (request) από έναν πελάτη και το προωθεί σε έναν διακομιστή (server) που μπορεί να το εξυπηρετήσει και στη συνέχεια επιστρέφει την απάντηση του διακομιστή στον πελάτη. Ένας αντίστροφος διακομιστής μεσολάβησης είναι ένας τύπος διακομιστή μεσολάβησης (proxy) που συνήθως βρίσκεται πίσω από το τείχος προστασίας (firewall) σε ένα ιδιωτικό δίκτυο (private network) και κατευθύνει τις αιτήσεις πελατών στον κατάλληλο διακομιστή(backend). Ένας αντίστροφος διακομιστής μεσολάβησης παρέχει ένα επιπλέον επίπεδο αφαίρεσης και ελέγχου για να διασφαλιστεί η ομαλή ροή της κίνησης δικτύου μεταξύ πελατών και διακομιστών.[23]

Οι κύριες λειτουργίες ενός αντίστροφου διακομιστή μεσολάβησης είναι:

- Διαμοιρασμός Φόρτου (Load Balancing)
- Αύξηση ταχήτητας (Web Accelaration)
- Ασφάλεια και ανωνυμία (Security and anonymity)

3.2.1. Διαμοιρασμός Φόρτου (Load Balancing)

Τα συστήματα εξισορρόπησης φορτίου αναπτύσσονται συνήθως όταν ένας ιστότοπος χρειάζεται πολλούς διακομιστές επειδή ο όγκος των αιτημάτων είναι

υπερβολικός για να εξυπηρετηθεί από μόνο έναν διακομιστή. Η ανάπτυξη πολλαπλών διακομιστών εξαλείφει επίσης το φαινόμενο του μοναδικού σημείου αποτυχίας (Single Point of Failure), καθιστώντας τον ιστότοπο πιο αξιόπιστο. Συχνότερα, πολλοί διακομιστές φιλοξενούν το ίδιο ακριβώς περιεχόμενο και η εργασία εξισορρόπησης φορτίου είναι να διανείμει το φόρτο εργασίας στους διακομιστές αυτούς, κατά τρόπο ούτως ώστε να γίνεται η καλύτερη δυνατή χρήση της χωρητικότητας κάθε διακομιστή, αποτρέποντας την υπερφόρτωση σε κάποιον από αυτούς , κάτι που έχει ως αποτέλεσμα την ταχύτερη δυνατή απόκριση στον πελάτη .

Ένας εξισορροπιστής φορτίου (load balancer) μπορεί επίσης να βελτιώσει την εμπειρία του χρήστη μειώνοντας τον αριθμό των απαντήσεων σφάλματος που βλέπει ο πελάτης. Αυτό γίνεται γιατί με την εξισορρόπηση φορτίου μπορεί να εντοπίσει πότε ένα διακομιστής δεν είναι σε θέση να εξυπηρετήσει κάποιο αίτημα (request) κάποιου χρήστη, οπότε αντίστροφος διακομιστής εκτρέπει τις αιτήσεις αυτές στους άλλους διακομιστές της ομάδας των διακομιστών. Στην απλούστερη υλοποίηση, ο εξισορροπιστής φορτίου ανιχνεύει την “υγεία” του διακομιστή παρακωλύοντας τις απαντήσεις σφάλματος σε τακτικά αιτήματα. Οι έλεγχοι υγείας (health check) των εφαρμογών είναι μια πιο ευέλικτη και εξελιγμένη μέθοδος στην οποία ο εξισορροπιστής φορτίου στέλνει ξεχωριστά αιτήματα ελέγχου της υγείας και απαιτεί έναν συγκεκριμένο τύπο απόκρισης για να θεωρήσει τον διακομιστή υγιή.

Μια άλλη χρήσιμη λειτουργία που παρέχεται από ορισμένους εξισορροπιστές φορτίου (loab balancers) είναι η επιμονή της περιόδου σύνδεσης (sticky session), η οποία σημαίνει την αποστολή όλων των αιτήσεων από ένα συγκεκριμένο πελάτη στον ίδιο διακομιστή. Πιο απλά όταν κάποιος χρήστης έχει πάρει session από κάποιον διακομιστή τότε ο εξισορροπιστής φορτίου (load balancer) θα συνεχίσει να στέλνει τα αιτήματα (request) αυτού του χρήστη στον ίδιο διακομιστή (server). Πολλές εφαρμογές πρέπει να αποθηκεύουν διάφορες πληροφορίες μόνο για να παρέχουν την βασική τους λειτουργικότητα. Όπως για παράδειγμα το καλάθι αγορών σε έναν ιστότοπο ηλεκτρονικού εμπορίου (eshop). Τέτοιες εφαρμογές υποβαθμίζονται ή μπορεί να αποτύχουν σε ένα περιβάλλον ισορροπημένου φορτίου, εάν ο εξισορροπιστής φορτίου (load balancer) διανέμει τα αιτήματα ενός χρήστη που έχει

session σε διαφορετικούς διακομιστές (servers), αντί να τις κατευθύνει στο διακομιστή (server) που ανταποκρίθηκε στο αρχικό αίτημα (request) .[23] Η λειτουργία αυτή είναι γνώστη σαν sticky session [24].

3.2.1.1. Αλγόρυθμοι Διαμοιρασμού Φόρτου (Load Balancing Algorithms)

Υπάρχουν πολυάριθμες τεχνικές και αλγόριθμοι που μπορούν να χρησιμοποιηθούν για τον έξυπνο διαμοιρασμό των αιτήσεων (request) σε μια ομάδα διακομιστών (servers). Ακόμα κάθε εταιρία που παρέχει υπηρεσίες διαμοιρασμό φόρτου , έχει αναπτύξει και δικούς της αλγορίθμους για αυτό το σκοπό. Η επιλεγμένη τεχνική θα εξαρτηθεί από τον τύπο της υπηρεσίας ή την εφαρμογή που εξυπηρετείται και την κατάσταση του δικτύου και των διακομιστών κατά το χρόνο υποβολής της αίτησης. Το είδος και ο όγκος των αιτημάτων προς τον διαμοιραστή φόρτου (load balancer) καθορίζει συχνά ποια μέθοδος διαμοιρασμού φόρτου θα χρησιμοποιηθεί. Όταν το φορτίο αιτημάτων είναι χαμηλό, τότε αρκεί μία από τις απλές μεθόδους εξισορρόπησης φορτίου. Αλλά σε περιόδους υψηλού φορτίου, οι πιο σύνθετες μέθοδοι χρησιμοποιούνται για να διασφαλιστεί η ομοιόμορφη κατανομή των αιτημάτων των χρηστών.

Ακολουθούν μερικές από τις πιο διαδεδομένες τεχνικές που χρησιμοποιούν οι εξισορροπιστές φορτίου (load balancers) για τον διαμοιρασμό του φόρτου στους διακομιστές :

- **Round Robin** : Μια απλή μέθοδο διαμοιρασμού φόρτου η οποία παρέχει ταυτόχρονα και ανοχή σε σφάλματα. Πολλοί ταυτόσημοι διακομιστές έχουν ρυθμιστεί ώστε να παρέχουν ακριβώς τις ίδιες υπηρεσίες. Όλα έχουν ρυθμιστεί ώστε να χρησιμοποιούν το ίδιο όνομα τομέα στο Internet (internet domain), αλλά ο καθένας έχει μια μοναδική διεύθυνση IP. Ένας διακομιστής DNS έχει μια λίστα με όλες τις μοναδικές διευθύνσεις IP που σχετίζονται με το όνομα τομέα Internet. Όταν λαμβάνονται αιτήματα για τη διεύθυνση IP που σχετίζονται με το όνομα τομέα Internet, οι διευθύνσεις επιστρέφονται με διαδοχικό τρόπο.

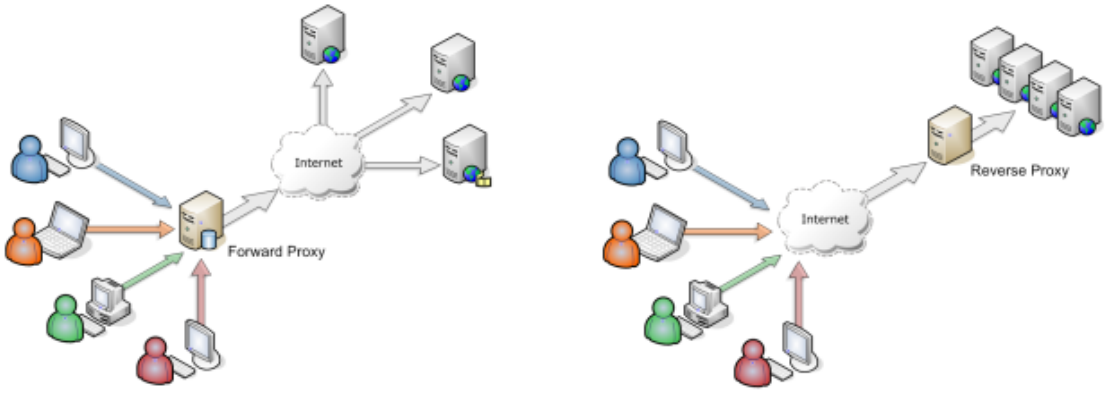
- **Weighted Round Robin:** Αυτός βασίζεται στην απλή μέθοδο εξισορρόπησης φορτίου Round Robin. Στην Weighted Round Robin έκδοση κάθε διακομιστής λαμβάνει ένα “βάρος” . Οι διακομιστές με υψηλότερο “βάρος” λαμβάνουν περισσότερα αιτήματα.
- **Least Connection:** Ούτε ο Round Robin ούτε ο Weighted Round Robin λαμβάνουν υπόψη το τρέχον φορτίο του διακομιστή κατά τη διανομή των αιτημάτων. Η μέθοδος Least Connection λαμβάνει υπόψη τον αριθμό των ενεργών συνόδων του διακομιστή. Το τρέχον αίτημα πηγαίνει στον διακομιστή που εξυπηρετεί τον ελάχιστο αριθμό ενεργών συνόδων κατά την τρέχουσα ώρα.
- **Source IP Hash**: Η μέθοδος εξισορρόπησης φορτίου IP Hash χρησιμοποιεί έναν αλγόριθμο που παίρνει τη διεύθυνση IP προέλευσης και προορισμού του πελάτη και του διακομιστή και τους συνδυάζει για να δημιουργήσει ένα μοναδικό κλειδί κατακερματισμού (key hash). Αυτό το κλειδί χρησιμοποιείται για την κατανομή των αιτημάτων του πελάτη σε έναν συγκεκριμένο διακομιστή. Καθώς το κλειδί μπορεί να αναγεννηθεί εάν η συνεδρία έχει καταστραφεί, αυτή η μέθοδος εξισορρόπησης φορτίου μπορεί να διασφαλίσει ότι το αίτημα του πελάτη κατευθύνεται στον ίδιο διακομιστή που είχε χρησιμοποιήσει προηγουμένως .Αυτό είναι χρήσιμο εάν είναι σημαντικό να συνδεθεί ένας πελάτης σε μια περίοδο λειτουργίας που εξακολουθεί να είναι ενεργή μετά από μια αποσύνδεση. Ένα παράδειγμα στον προαναφερθέν είναι η διατήρηση αντικείμενων σε ένα καλάθι αγορών μεταξύ των περιόδων σύνδεσης.[25]

3.2.1.2. Αντίστροφοι Διακομιστές Μεσολάβησης Vs Διακομιστές Προώθησης (Reverse Proxies VS Forward Proxies)

Συχνά συγχέεται η έννοια του Αντίστροφου Διακομιστή Μεσολάβησης (Reverse Proxy) με τον Διακομιστή Προώθησης (Forward Proxy). Γενικά όταν κάποιος αναφέρεται στην έννοια του proxy αναφέρεται κυρίως στον Forward Proxy. Ένας Διακομιστή Προώθησης (Forward Proxy) είναι ένας διακομιστής μεσολάβησης που είναι ρυθμισμένος από τους τοπικούς διαχειριστές ώστε να χειρίζονται τις αιτήσεις για μια ομάδα χρηστών (clients) προς μια άγνωστη ή αυθαίρετη ομάδα πόρων που βρίσκονται εκτός του ελέγχου τους. Ένα καλό παράδειγμα είναι ένας

διακομιστής μεσολάβησης ιστού (web proxy) που δέχεται αιτήματα ιστού από χρήστες σε ένα τοπικό δίκτυο και τα προωθεί σε διακομιστές στο διαδίκτυο. Ο σκοπός ενός Διακομιστή Προώθησης (Forward Proxy) είναι η διαχείριση της επισκεψιμότητας των χρηστών.

Ο Αντίστροφος Διακομιστής Μεσολάβησης (Reverse proxy) είναι ένας διακομιστής μεσολάβησης που έχει ρυθμιστεί για τη διαχείριση αιτημάτων από μια ομάδα απομακρυσμένων ή αυθαίρετων χρηστών σε μια ομάδα γνωστών πόρων υπό τον έλεγχο του τοπικού διαχειριστή. Ο σκοπός ενός αντιστρόφου διακομιστή μεσολάβησης είναι η διαχείριση των συστημάτων διακομιστή.



Εικόνα 15: Forward Proxy Vs Reverse Proxy

3.3. Απόδοση εφαρμογών ιστού

Η μέτρηση της απόδοσης γενικά είναι κάτι το οποίο έχει μεγάλη σημασία όχι μόνο στον τομέα της πληροφορικής αλλά και σε πολλούς άλλους τομείς της επιστήμης και όχι μόνο. Γενικά με τον όρο απόδοση νοείται η προσπάθεια προσδιορισμού το πόσο καλά “δουλεύει” κάτι.

Το να μετρήσει κανείς την απόδοση μια εφαρμογής ιστού δεν είναι κάτι το οποίο μπορεί να γίνει με σχετική ευκολία. Και αυτό γιατί θα πρέπει να αποφασιστεί τι θα είναι αυτό το οποίο θα πρέπει να μετρηθεί. Μια σελίδα ιστού για παράδειγμα για να εμφανιστεί στον φυλλομέτρητη (browser) του χρήστη θα πρέπει να κάνει αρκετές αιτήσεις (request) στον διακομιστή (server) για να εμφανιστεί στον χρήστη. Ακόμα το

περιεχόμενο το οποίο μπορεί να ζητάει ο χρήστης από τον διακομιστή μπορεί να είναι ένα μεγάλο αρχείο το οποίο θα κάνει αρκετή ώρα για να φτάσει ακέραιο στον χρήστη. Επίσης μπορεί ο διακομιστής να έχει αρκετά μεγάλο φόρτο και να μην μπορεί να εξυπηρετήσει τα αιτήματα των χρηστών. Τέλος μπορεί το δίκτυο μεταξύ χρήστη και διακομιστή να είναι αρκετά φορτωμένο , με αποτέλεσμα να μην μπορούν τα αιτήματα του χρήστη και οι απαντήσεις του διακομιστή να μεταφερθούν με την επιθυμητή ταχύτητα.

Γενικά η απόδοση μια εφαρμογής ιστού μπορεί να επηρεάζεται από διάφορες παραμέτρους οι οποίες πολλές φορές δεν ελέγχονται και από τους διαχειριστές της εφαρμογής. Έτσι θα πρέπει να βρεθούν κάποιες μετρικές ώστε να υπάρχει κάποιο μετρήσιμο αποτέλεσμα.

3.3.1. Χρόνος απόκρισης (Response Time)

Μια μετρική που μπορεί να χρησιμοποιηθεί για να μετρηθεί η απόδοση μιας εφαρμογής ιστού (web application) είναι ο χρόνος απόκρισης (response time) . Ο χρόνος απόκρισης (response time) είναι ο συνολικός χρόνος που χρειάζεται από τη στιγμή που θα υποβληθεί ένα αίτημα (request) σε έναν διακομιστή μέχρι τη στιγμή που θα έρθει η απάντηση από τον διακομιστή (server) [27].

Και εδώ τώρα γεννιούνται διάφορα ερωτήματα. Από που πρέπει να γίνει η αίτηση στον διακομιστή; Θα πρέπει να μετρηθεί ο χρόνος που κάνει το πρώτο αίτημα ή ο συνολικός χρόνος τον οποίο θα κάνουν να έρθουν όλα τα αιτήματα μέχρι να φορτωθεί όλη η σελίδα. Η απάντηση είναι ότι εξαρτάται κάθε φορά τι θέλει κάποιος να δει και να μετρήσει. Αν για παράδειγμα οι διαχειριστές ενός διακομιστή θέλουν να μετρήσουμε τον χρόνο απόκρισης για να κάνουμε διορθώσεις στον διακομιστή ιστού θα μετρήσουν το χρόνο απόκρισης από τον διακομιστή προς τον ίδιο τον διακομιστή και ανάλογα το τι θέλουν να διορθώσουν θα μετρήσουν τον χρόνο της μίας αίτησης ή θα μετρήσουν τον συνολικό χρόνο τον οποίο θέλει για να φορτωθεί η σελίδα. Στην περίπτωση που θέλει κάποιος να μετρήσει τον συνολικό χρόνο που χρειάζεται για να φορτωθεί μια σελίδα θα πρέπει η μέτρηση να γίνει από τον φυλλομετρητή (browser) του χρήστη.[28]

Ένα άλλο μεγάλο ερώτημα τώρα που γεννιέται είναι το ποιος είναι ο αποδεκτός χρόνος απόκρισης (response time) που θα πρέπει να έχει μια ιστοσελίδα. Και σε τι ιστοσελίδα αναφέρεται κάποιος. Αναφέρεται σε μια ιστοσελίδα που περιέχει μόνο απλό περιεχόμενο ή σε μια ιστοσελίδα της οποίος το περιεχόμενο είναι πιο πλούσιο και περιέχει και κείμενο και εικόνες ή ακόμα και βίντεο; Όπως προαναφέρθηκε προηγουμένως οι παράμετροι είναι πολλοί οπότε το καλύτερο είναι να υπάρχει μια σελίδα αναφοράς κάθε φορά ώστε να γίνονται οι μετρήσεις σε αυτήν την σελίδα, έχοντας μεριμνήσει αυτή η σελίδα να έχει ένα μέσο περιεχόμενο.

Η απάντηση στο πρώτο ερώτημα της προηγούμενης παραγράφου έχει απασχολήσει την ακαδημαϊκή κοινότητα από το 1968 όπου ο J. Miller στη δημοσίευσή του για την IBM, "Response time in man-computer conversational transactions " περιγράφει τα τρία κατώτατα όρια προσοχής του χρήστη σχετικά με τον χρόνο απόκρισης:

0,1 δευτερόλεπτα - ο χρήστης βλέπει αυτό ως στιγμιαίο.

1,0 δευτερόλεπτο - θα παρατηρήσουν μια καθυστέρηση αλλά μπορεί να συνεχίσει να λειτουργεί.

10 δευτερόλεπτα - όριο για τη διατήρηση της προσοχής του χρήστη.

Ο Miller αναγνώρισε ένα χρόνο απόκρισης δύο δευτερολέπτων ως ιδανικό[29]. Αργότερα ο Peter Bickford στη δημοσίευσή του “Worth the Wait?” δημιούργησε τον κανόνα των 8 δευτερολέπτων (“8-second rule”) όπου λέει ότι οι μισοί χρήστες εγκατέλειψαν μια σελίδα μετά από τα 8.5 δευτερόλεπτα .

Η πραγματικότητα είναι ότι στα σημερινά χρόνια τα πράγματα έχουν αλλάξει αρκετά σε σχέση με τότε καθώς και οι ταχύτητες γενικότερα έχουν αυξηθεί αλλά και έχουν βρεθεί διάφορες τεχνικές για την πιο γρήγορη φόρτωση του περιεχομένου. Ταυτόχρονα όμως έχει αυξηθεί και ο όγκος των δεδομένων που φορτώνουν οι ιστοσελίδες. Ακόμα οι φυλλομετρητές (browsers) κρατάνε στην κρυφή μνήμη τους (cache) ένα μεγάλο μέρος των πληροφοριών μια ιστοσελίδας. Αυτό έχει σαν αποτέλεσμα την γρηγορότερη φόρτωση της ιστοσελίδας. Χαρακτηριστικά να αναφερθεί ότι στον υπολογιστή όπου γράφεται αυτή τη στιγμή αυτό το κείμενο η

αρχική σελίδα του www.google.com για να φορτωθεί πλήρως την πρώτη φορά κάνει 1,77 sec ενώ για να φορτωθεί ξανά για δεύτερη φορά κάνει 0.8 sec . Και η σελίδα η οποία αναφέρθηκε είναι μια σελίδα χωρίς πλούσιο περιεχόμενο και χωρίς πληροφορία.

3.3.2. Βαθμολογία Apdex (Apdex Scores)

Η βαθμολογία Apdex ήταν αποτέλεσμα της αναζήτησης μιας μετρικής για την μέτρηση της απόδοσης μιας ιστοσελίδας[31]. Είναι ένα αριθμητικό μέτρο ικανοποίησης των χρηστών για την απόδοση των επιχειρησιακών εφαρμογών. Μετατρέπει πολλές μετρήσεις σε έναν αριθμό σε ομοιόμορφη κλίμακα από 0 έως 1 (0 = μη χρήστες ικανοποιημένοι, 1 = όλοι οι χρήστες ικανοποιημένοι). Μπορεί να χρησιμοποιηθεί για να αξιολογήσει τα πάντα από το χρόνο απόκρισης μιας εφαρμογής, την ποιότητα των τροφίμων, τα χειρουργικά αποτελέσματα, το εύρος ζώνης που παρέχεται από έναν πάροχο κ.τ.λ.

Η μέθοδος Apdex στον τομέα της απόδοσης των εφαρμογών ιστού, μεταφράζει πολλούς ατομικούς χρόνους απόκρισης, μετρούμενοι από το επίπεδο του χρήστη, σε έναν μόνο αριθμό. Μια αίτηση σε έναν διακομιστή ιστού είναι μια ατομική αλληλεπίδραση με το σύστημα, μέσα σε μια ευρύτερη διαδικασία. Ο χρόνος απόκρισης του αιτήματος ορίζεται ως ο χρόνος που μεσολαβεί μεταξύ της στιγμής που κάποιος χρήστης κάνει κάτι (κλικ με το ποντίκι, εισέρχεται ή επιστρέφει, κλπ) και του συστήματος (πελάτης, δίκτυο, διακομιστές) που ανταποκρίνεται έτσι ώστε να μπορεί να προχωρήσει η διαδικασία. Αυτή είναι η περίοδος κατά την οποία ο άνθρωπος περιμένει το αποτέλεσμα. Αυτές οι μεμονωμένες περίοδοι αναμονής είναι εκείνες που ορίζουν την "ανταπόκριση" της εφαρμογής στον χρήστη .

3.3.3. Πώς δουλεύει το Apdex Score

Ένα βασικό χαρακτηριστικό της διαδικασίας είναι ότι είναι απλό .Η διαδικασία υπολογισμού της βαθμολογίας Apdex ξεκινά με τον ορισμό του χρόνου αναφοράς T. Ο χρόνος αυτός αναφέρεται στον χρόνο όπου ορίζει κάποιος ότι θα πρέπει να είναι ο αποδεκτός χρόνος, μέσα στον οποίο θα πρέπει να έχει φορτωθεί στον χρήστη η σελίδα την οποία έχει ζητήσει από τον διακομιστή.

Η ένδειξη ικανοποίησης των χρηστών βασίζεται σε τρεις ζώνες απόκρισης της εφαρμογής:

- **Ικανοποιημένος (Satisfied)** - Ο χρήστης είναι πλήρως παραγωγικός. Αυτό αντιπροσωπεύει ότι ο χρόνος φόρτωσης της σελίδας είναι μικρότερος από την χρονική τιμή T . Σε αυτή την περίπτωση οι χρήστες δεν παρεμποδίζονται από τον χρόνο απόκρισης της εφαρμογής ώστε να κάνουν την εργασία τους.
- **Υποφερτό (Tolerating)** - Ο χρήστης παρατηρεί καθυστέρηση απόδοσης σε απαντήσεις μεγαλύτερες από T δευτερόλεπτα αλλά μικρότερες από 4T δευτερόλεπτα. Ο χρήστης είναι σε θέση να εκτελέσει τις εργασίες τις οποίες θέλει , παρατηρώντας βέβαια μια μικρή καθυστέρηση
- **Απογοητευτικός (Frustrated)** - με χρόνο απόκρισης μεγαλύτερο από 4T δευτερόλεπτα. Ο χρόνος αυτός είναι απαγορευτικός για την παραμονή του χρήστη στην εφαρμογή και υπάρχει μεγάλη πιθανότητα να εγκαταλείψει την σελίδα. Ο χρόνος αυτός ακόμα μειώνει την παραγωγικότητα χρηστών.

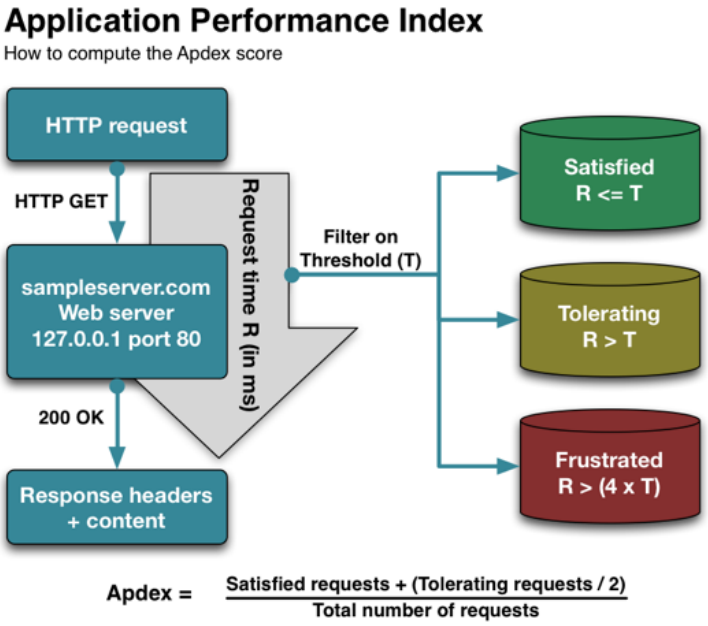
Η βαθμολογία Apdex παράγεται από τον αριθμό των ικανοποιημένων δειγμάτων (satisfied) συν το ήμισυ των υποφερτών δειγμάτων (tolerating) συν κανένα από τα απογοητευτικά δείγματα, διαιρούμενο με όλα τα δείγματα.

$$Apdex_T = \frac{Satisfied + Tolerating\ /2}{Total\ Samples}$$

Για να οριστεί η βαθμολογία Apdex πρέπει να ακολουθηθούν τα παρακάτω βήματα[32]:

- Τίθεται ένας χρόνος (T) ως στόχο για το χρόνο φόρτωσης της σελίδας. Στις εφαρμογές ηλεκτρονικού εμπορίου η επιθυμητή καθυστέρηση ορίζεται συνήθως μεταξύ 1 και 3 δευτερολέπτων .
- Δίνεται η ετικέτα «Ικανοποιημένη» σε όλα τα δείγματα που είναι μικρότερα από αυτό τον χρόνο.
- Πολλαπλασιάζεται το T κατά 4, το οποίο, σύμφωνα με το Apdex, είναι ο μέγιστος χρόνος που οι άνθρωποι θα ανέχονται να περιμένουν τη σελίδα να φορτώσει.

- Δίνεται την ετικέτα «Υποφερτό» στα δείγματα που έχουν χρόνο μεγαλύτερο από T αλλά μικρότερο από 4 T.
- Χρησιμοποιείται ο τύπος του Apdex για να υπολογιστεί ο δείκτης ικανοποίησης



Εικόνα 16: Αναπαράσταση Apdex

4. Προτεινόμενη Λύση

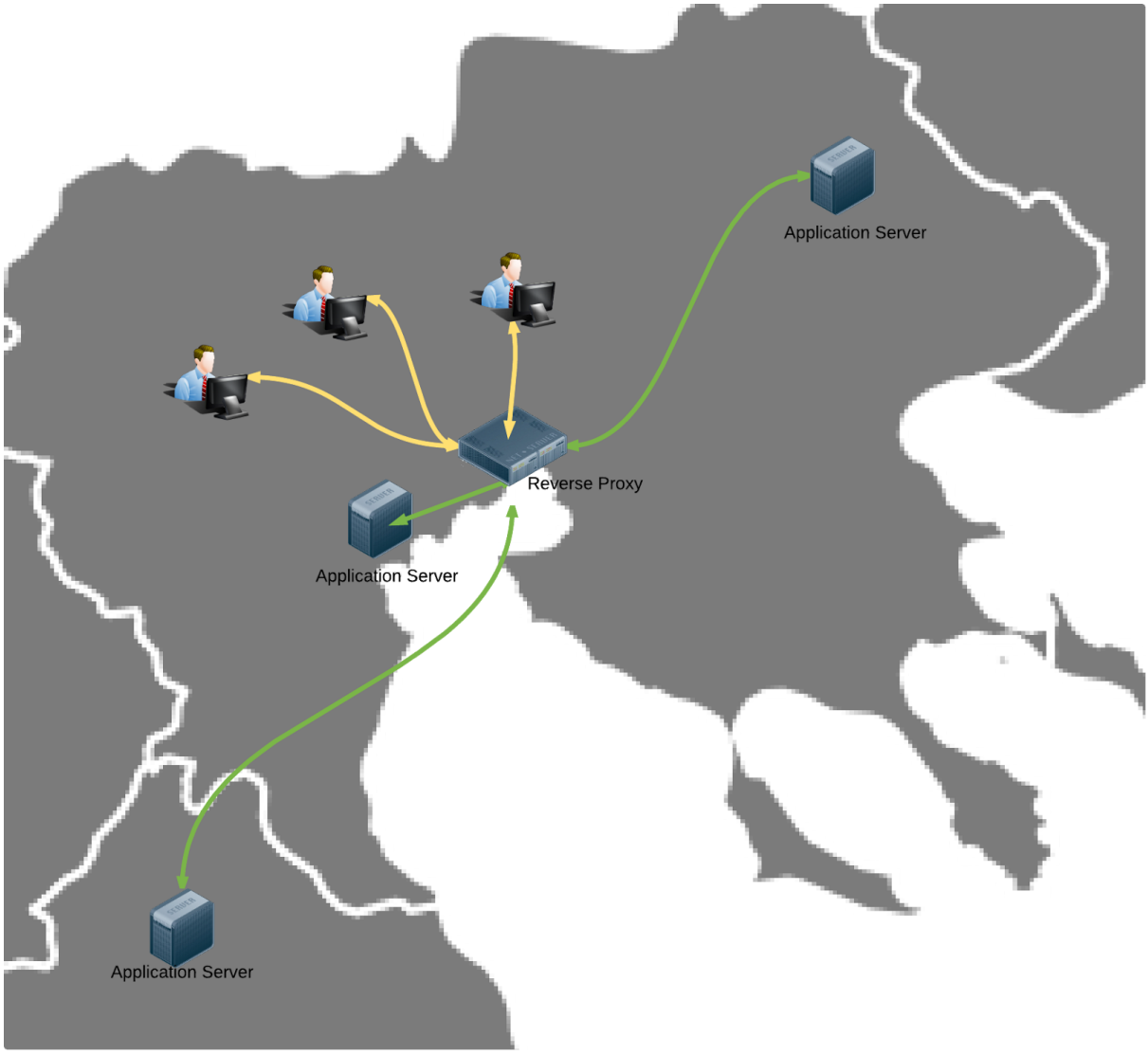
Στην παρούσα ενότητα αναλύεται μια προτεινόμενη λύση στο πρόβλημα το οποίο αναλύθηκε σε προηγούμενες παραγράφους. Ακόμα θα αναφερθεί η αρχιτεκτονική την οποία πρέπει να έχει το σύστημα για την λύση αυτή, καθώς επίσης και οι τεχνικές λεπτομέρειες .

4.1. Λειτουργία - Αρχιτεκτονική

Καταρχήν η διαδικτυακή εφαρμογή την οποία θα υποστηρίζει το σύστημα, θα πρέπει να είναι κατανεμημένη, δηλαδή θα υπάρχει σε διάφορους διακομιστές σε διάφορα επιλεγμένα σημεία ανά την Ελλάδα , οι οποίοι θα μπορούν να δεχτούν τα αιτήματα των χρηστών.

Θα πρέπει να υπάρχει κάποιος τρόπος να γίνεται η ανακατεύθυνση των αιτήσεων των χρηστών στον κατάλληλο διακομιστή (server). Αυτό θα γίνει με την χρήση ενός Αντίστροφου Διακομιστή Μεσολάβησης (Reverse Proxy). Όλοι οι χρήστες θα χωριστούν ανάλογα τις γεωγραφικής τους περιοχής. Σε κάθε γεωγραφική περιοχή θα υπάρχει ένας Αντίστροφος Διακομιστής Μεσολάβησης (Reverse Proxy) όπου θα είναι υπεύθυνος για την ανακατεύθυνση των αιτήσεων των συγκεκριμένων χρηστών. Αυτός θα δέχεται όλα τα αιτήματα των χρηστών και θα τα ανακατευθύνει στον καταλληλότερο διακομιστή βάση κάποιων προϋποθέσεων .

Έτσι όταν έρχεται ένα αίτημα (request) από κάποιον χρήστη ο αντίστροφος διακομιστής μεσολάβησης (reverse proxy) θα πρέπει να ελέγχει άμα το αίτημα (request) του χρήστη έχει session από κάποιον διακομιστή. Άμα έχει θα τον στέλνει στον διακομιστή αυτό. Αν δεν έχει θα τον στέλνει στον καταλληλότερο διακομιστή (server).



Εικόνα 17: Αναπαράσταση Τοπολογίας Γεωγραφικού Διαμερίσματος

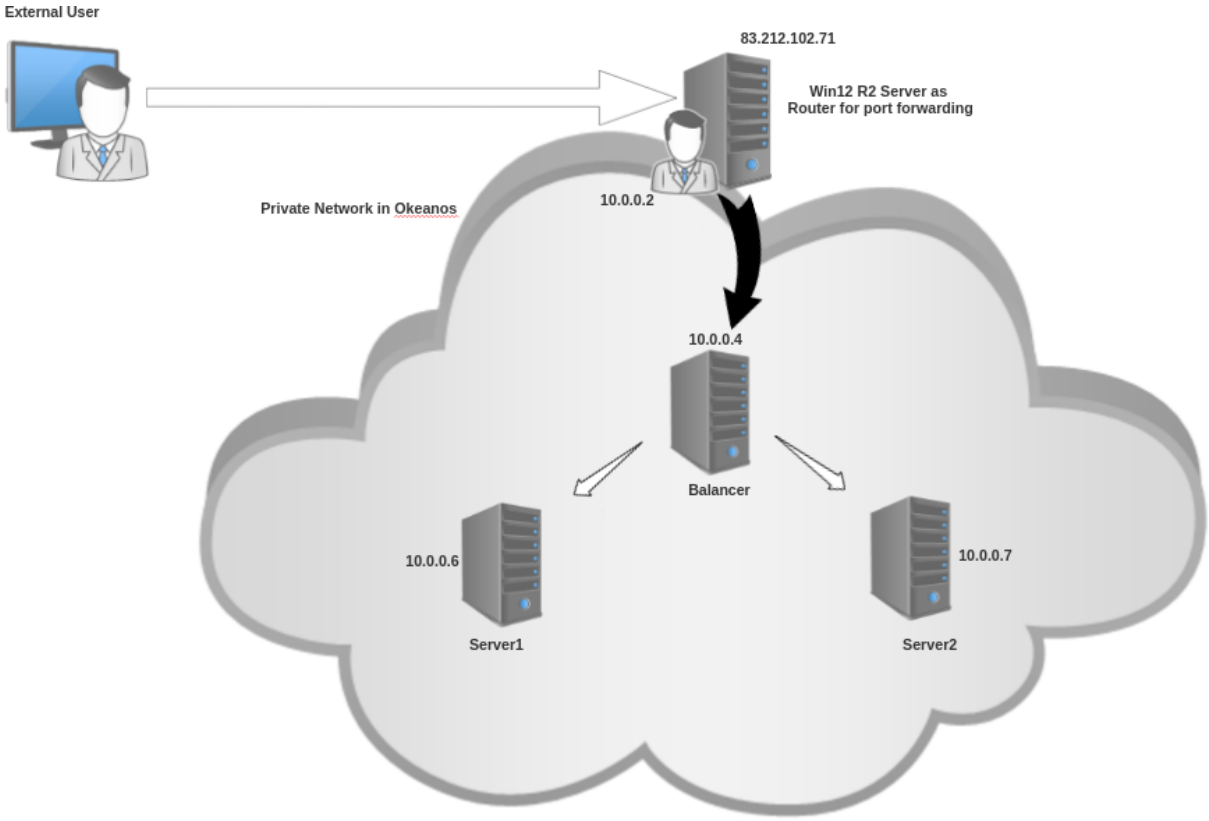
4.2. Στοιχεία Υλοποίησης

Στην παρούσα ενότητα θα παρουσιαστεί με λεπτομέρεια ο τρόπος με τον οποίο υλοποιείται το δοκιμαστικό περιβάλλων καθώς επίσης και τις τεχνολογίες και τις ρυθμίσεις που έγιναν.

4.2.1. Δοκιμαστικό Περιβάλλον, Αρχιτεκτονική

Για την προσέγγιση του πρόβλημα υλοποιήθηκε ένα δοκιμαστικό περιβάλλον όπου προσπαθεί να εξομοιωθεί η τοπολογία που αναφέρθηκε.

Για την υλοποίηση του δοκιμαστικού περιβάλλοντος χρησιμοποιήθηκαν οι cloud υποδομές που προσφέρει ο Okeanos[34]. Εκεί έχει υλοποιηθεί ένα ιδιωτικό δίκτυο, όπου μέσα σε αυτό υπάρχουν κάποια εικονικά μηχανήματα (Virtual Machines VMs). Τα VMs αυτά έχουν IP διεύθυνση από το ιδιωτικό δίκτυο και φυσικά έχουν και δικτυακή επικοινωνία μεταξύ τους. Το λειτουργικό σύστημα αυτών ειναι Ubuntu Mate 16.04 και είναι 3 στον αριθμό. Το ένα λειτουργεί ως Reverse Proxy και τα άλλα δύο έχουν τον διακομιστή ιστού (web server) όπου φιλοξενείται μια δοκιμαστική εφαρμογή. Η εφαρμογή αυτή τρέχει πάνω Wildfly 10.0.1 [40] application server και πρόκειται για μια JSF[41] εφαρμογή. Η επικοινωνία με τον “έξω” κόσμο του διαδικτύου γίνεται μέσω ενός ένα άλλου VM το οποίο έχει δημόσια ip (public ip) και χρησιμεύει σαν δρομολογητής (router) κάνοντας προώθηση θυρών (port forwarding). Το VM αυτό έχει λειτουργικό σύστημα Windows Server 2012 R2 .

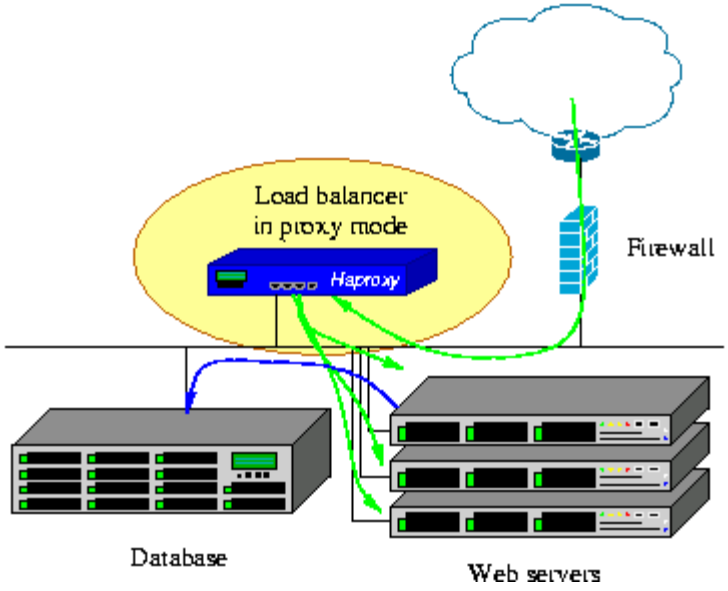


Εικόνα 18: Τοπολογία Okeanos

Στο ιδιωτικό δίκτυο ο Αντίστροφος Διακομιστής Μεσολάβησης έχει ip 10.0.0.4 , ο ένας διακομιστής με το όνομα “Server1” έχει ip 10.0.0.6 και ο άλλος διακομιστής με όνομα “Server2” έχει ip “10.0.0.7”. Ο διακομιστής που έχει τον ρόλο του router έχει ip 10.0.0.2 .

4.2.2. Αντίστροφος Διακομιστής Μεσολάβησης (Reverse Proxy)

Ο Αντίστροφος Διακομιστής Μεσολάβησης ο οποίος επιλέχτηκε είναι ο HAproxy . Ο HAProxy είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα που παρέχει ένα υψηλής διαθεσιμότητας διαμοιρασμό φόρτου (load balancing) και διακομιστή μεσολάβησης (proxy server) για εφαρμογές που βασίζονται στο TCP και HTTP και διαμοιράζει αιτήματα χρηστών σε πολλούς διακομιστές. Είναι γραμμένος στη προγραμματιστική γλώσσα C και έχει τη φήμη ότι είναι γρήγορος και αποδοτικός ακόμα και όταν η διαθέσιμη επεξεργαστική ισχύς και μνήμη είναι χαμηλή.[35]

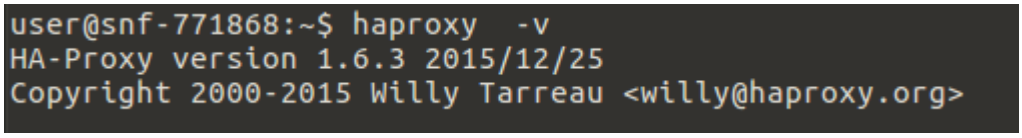


Εικόνα 19: HAproxy[35]

Ο HAProxy χρησιμοποιείται από διάφορους ιστότοπους υψηλού προφίλ όπως GitHub, Bitbucket, StackOverflow, Reddit, Speedtest.net, Tumblr, Twitter , Tuenti , όπως επίσης και το προϊόν OpsWorks της Amazon Web Services [36]. Η κύρια λειτουργία του HAproxy είναι ο διαμοιρασμός φόρτου (load balancing) αλλά υποστηρίζει και πολλές ακόμα λειτουργίες όπως για παράδειγμα λειτουργίες caching proxying και άλλα.

Η έκδοση η οποία χρησιμοποιήθηκε για την πειραματική υλοποίηση της παρούσας διπλωματικής εργασίας, είναι η έκδοση 1.6.3. Ο HAproxy από την έκδοσή

του 1.6.13 υποστηρίζει και την προγραμματιστική γλώσσα Lua[37] . Η Lua είναι μια ισχυρή, αποτελεσματική, ελαφριά, ενσωματωμένη γλώσσα scripting. Υποστηρίζει προγραμματισμό, αντικειμενοστραφή προγραμματισμό, λειτουργικό προγραμματισμό, προγραμματισμό βάσεων δεδομένων και περιγραφής δεδομένων[38].



Εικόνα 20: Έκδοση HAproxy

Οι ρυθμίσεις που εφαρμόσαμε στον HAProxy φαίνονται παρακάτω. Το αρχείο που παρουσιάζετε είναι το haproxy.cfg που είναι το αρχείο ρυθμίσεων του HAproxy:

```
###### HAproxy Configuration File #######
# Created by George Tsotzolas #

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 socket /home/user/haproxy.stat
        stats timeout 10m
        user haproxy
        group haproxy
        daemon

        #Read from Lua code
        #When the Haproxy start run the lua script
        lua-load etc/haproxy/scripts/read_from_file.lua

defaults
        log     global
        mode    http
        option  httplog
        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

        compression algo gzip
 application/plain application/x-javascript application/json
application/xml text/xml

    stats enable
    stats uri /haproxy?stats
    stats auth admin:admin!234@
    stats realm Haproxy\ Statistics

############ FrontEnds #################
frontend localnodes
    bind *:80
    mode http

 compression algo gzip
     compression type text/html text/plain text/css text/json
text/javascript application/plain application/x-javascript
application/json application/xml text/xml

      acl is_app hdr_end(host) -i app.zapto.org

     #use_backend nodes if is_app
     redirect location https://app.zapto.org/lab/login.jsf if
is_app


frontend internal
    #self signed ssl certificate
    bind :443 ssl crt /etc/ssl/haproxy/private/haproxy.pem

    acl is_app hdr_end(host) -i app.zapto.org
```

```
    acl app url_beg -i /lab

    #Check if the request have a cookie
    acl cookie_found hdr_sub(cookie) JSESSIONID=

    redirect prefix https://app.zapto.org/lab if is_app !app


    #if the request have a cookie go to all server to balance from
sticky session
    use_backend serverAll if cookie_found

    #if no cookie balance with lua script
    use_backend %[lua.choose_backend] if is_app !cookie_found

    #if something goes wrong send in default_backend
    default_backend serverAll


############ Backends ##################

backend server1
   mode http
   balance  first
   compression algo gzip
   application/xml text/xml
   option forwardfor
   option tcpka
   option http-server-close

   #This is for sticky Session
   cookie JSESSIONID prefix nocache

   option httpchk HEAD /lab/login.jsf HTTP/1.0
   reqadd X-Forwarded-Proto:\ https

   server web01 10.0.0.6:8080 cookie appA check inter 10000
   #Backup server in case that the active server in down
   server web02 10.0.0.7:8080 cookie appB check inter 10000 backup
```

```
backend server2
    mode http
    balance  first
    compression algo gzip
    option forwardfor
    option tcpka
    option http-server-close

    #This is for sticky Session
    cookie JSESSIONID prefix nocache

    option httpchk HEAD /lab/login.jsf HTTP/1.0
    reqadd X-Forwarded-Proto:\ https

    server web02 10.0.0.7:8080 cookie appB check inter 10000
    #Backup server in case that the active server in down
    server web01 10.0.0.6:8080 cookie appA check inter 10000
backup


#Backend without lua decision when the user have session  or
something goes wrong
backend serverAll
    mode http
    balance  roundrobin
    compression algo gzip
    option forwardfor
    option tcpka
    option http-server-close
    #This is for sticky Session
    cookie JSESSIONID prefix nocache

    option httpchk HEAD /lab/login.jsf HTTP/1.0
    option httpclose
    reqadd X-Forwarded-Proto:\ https

    server web01 10.0.0.6:8080 cookie appA check inter 10000
    server web02 10.0.0.7:8080 cookie appB check inter 10000
```

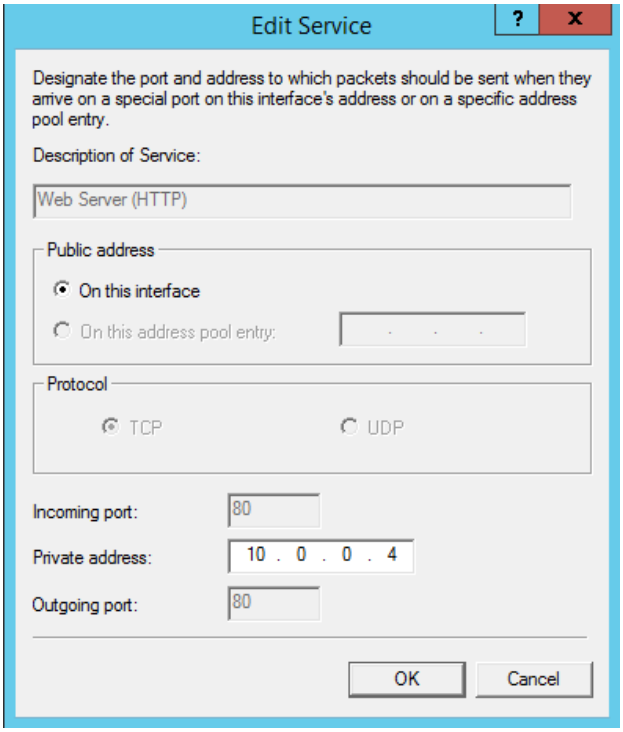
Το Lua Script το οποίο διαβάζει από αρχείο για να ενημερωθεί ο HAproxy για τον προτεινόμενο διακομιστή φαίνεται παρακάτω:

```
content = Map.new("/etc/haproxy/scripts/ip.map", Map.str);
local function find_backend(cn)
    core.log(core.info, "************************************")
    core.log(core.info, "The Ip is ");
    core.log(core.info, cn);
    core.log(core.info, "************************************")

    -- This line find the ip
    local metadata = content:lookup(cn);
    local metadata1 = content1;

    if (metadata == nil) then
        core.log(core.info, "*********************************")
        core.log(core.info, "The Ip not found")
        core.log(core.info, "*********************************")
        return "serverAll"
    else
        return metadata
    end
end

function choose_backend(txn)
    local arg1 = txn.f:src();
    backend = find_backend(arg1)

    if backend==nil then
        core.log(core.info, "*********************************")
        core.log(core.info, "Null Backend go to default server")
        core.log(core.info, "*********************************")
        backend=serverAll;
    end
    return backend
end

core.register_fetches("choose_backend", choose_backend)
```

4.2.3. Συμπληρωματικά στοιχεία υλοποίησης

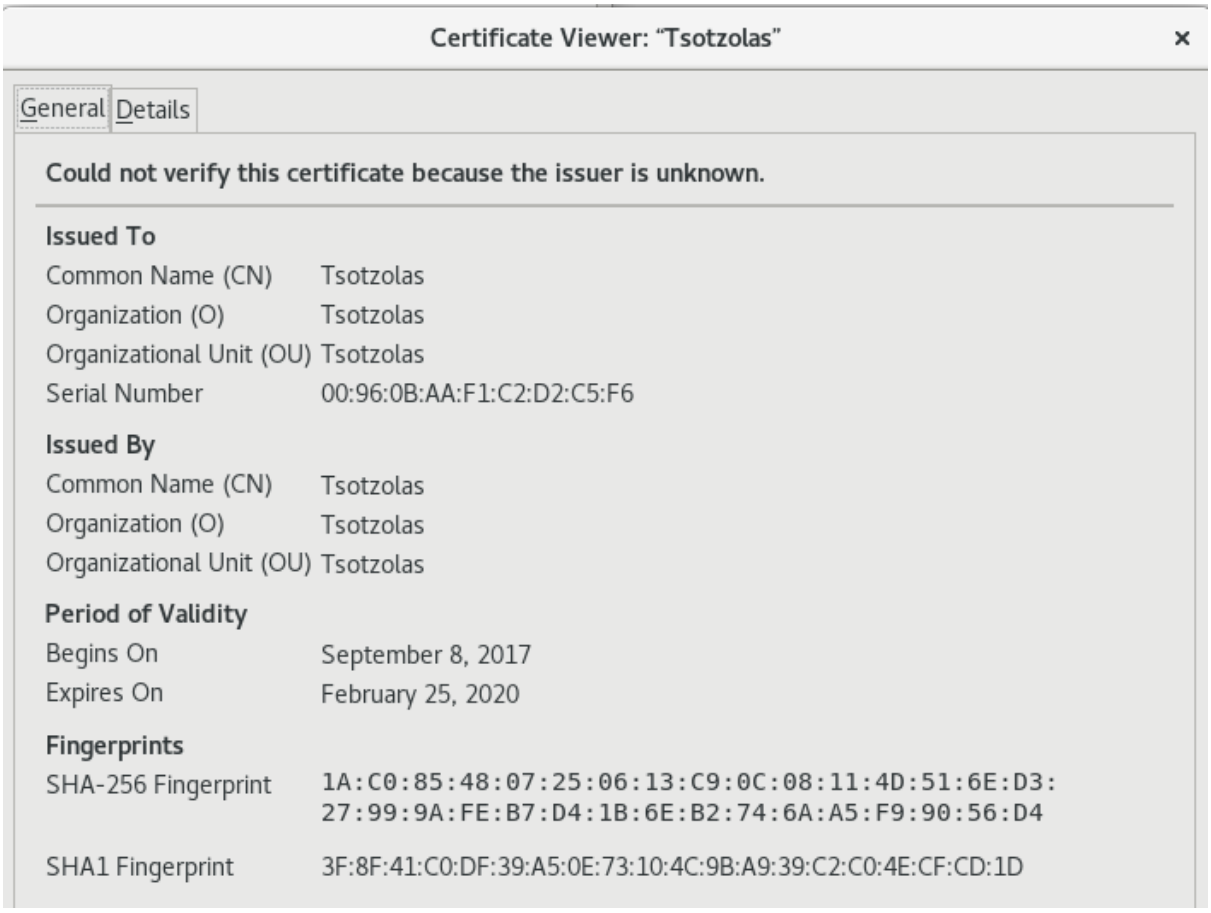
Στην παρούσα παράγραφο θα αναφέρονται κάποια συμπληρωματικά στοιχεία τα οποία έχουν υλοποιηθεί για την πιο ολοκληρωμένη παρουσίαση της λύσης .

- **DNS όνομα**: Καταχωρήθηκε ένα δωρεάν όνομα DNS από το https://www.noip.com/ ώστε να προσομοιαστούν όσο το δυνατόν περισσότερο οι πραγματικές συνθήκες αλλά και για λόγους ευκολίας ώστε να μην χρειάζεται να πληκτρολογείται κάθε φορά η διεύθυνση ip. Η καταχώρηση που έχει γίνει έχει το όνομα “app.zapto.org” να αντιστοιχεί στην ip 83.212.102.71. Έτσι για να προσπελαστεί η δοκιμαστική εφαρμογή, αντί να χρειάζεται να πληκτρολογηθεί η ip 83.212.102.71, πληκτρολογείται το “app.zapto.org”. Να σημειωθεί σε αυτό το σημείο ότι η ip 83.212.102.71 αντιστοιχεί στην εξωτερική ip που έχει το VM με τα Windows Server 2012 R2 που χρησιμεύει ως δρομολογητής (router). Σε αυτόν έχει γίνει καταχώρηση όπου προωθούνται τα αιτήματα στον Διακομιστή Αντίστροφης Δρομολόγησης (Reverse Proxy).



Εικόνα 21: Καταχώρηση Port Forwarding

- **Πιστοποιητικό SSL**: Για να μπορεί να υπάρχει κρυπτογραφημένη η κίνηση των αιτημάτων και των απαντήσεων του διακομιστή , έχει υλοποιηθεί ένα self sign πιστοποιητικό SSL . Αυτό υλοποιήθηκε με την βοήθεια του OpenSSL[38]. Οι λεπτομέρειες του πιστοποιητικού φαίνονται στη εικόνα 22 όπως παρουσιάζονται από τον φυλλομέτρητη Mozilla Firefox.



Εικόνα 22: Πιστοποιητικό SSL

Όταν κάποιος χρήστης πληκτρολογήσει στον φυλλομετρητή του το "http://app.zapto.org" τότε ο Αντίστροφος Διακομιστής Μεσολάβησης (Reverse Proxy) θα αλλάξει το url σε "https://app.zapto.org". Δηλαδή ο Αντίστροφος Διακομιστής Μεσολάβησης (Reverse Proxy) αλλάζει το url ώστε πάντα να υπάρχει κρυπτογραφημένη επικοινωνία μεταξύ του χρήστη και του διακομιστή. Στην συγκεκριμένη τοπολογία οι απαντήσεις (responses) του που έρχονται από τον διακομιστή (server) θα πρέπει να έρχονται κρυπτογραφημένες .Οπότε μπορεί να είναι και υπερβολική η παρουσία της προσθήκης του SSL στο επίπεδο του

Αντίστροφου Διακομιστή Μεσολάβησης στην συγκεκριμένη τοπολογία. Στο δοκιμαστικό περιβάλλων δεν έχει υλοποιηθεί η προσθήκη SSL να γίνεται στο επίπεδο του διακομιστή (server) ώστε να είναι κρυπτογραφημένη η επικοινωνία μεταξύ του διακομιστή και του Αντίστροφου Διακομιστή Μεσολάβησης (Reverse Proxy). Σε παραγωγικό περιβάλλων όμως θα πρέπει η επικοινωνία μεταξύ του Αντίστροφου Διακομιστή Μεσολάβησης (Reverse Proxy) και του διακομιστή να είναι κρυπτογραφημένη.

- **Συμπίεση δεδομένων:** Για την συμπίεση των δεδομένων έχει χρησιμοποιηθεί ο διαδεδομένος αλγόριθμος gzip. Η συμπίεση των δεδομένων γίνεται για να μειωθεί ο όγκος των δεδομένων που διακινούνται μεταξύ του χρήστη και το διακομιστή. Η συμπίεση των δεδομένων γίνεται και στο επίπεδο του διακομιστή. Στο επίπεδο του Αντίστροφου Διακομιστή Μεσολάβησης (Reverse Proxy) έχει τοποθετηθεί για μεγαλύτερη εξασφάλιση ώστε τα δεδομένα μεταξύ του χρήστη και του διακομιστή να είναι συμπιεσμένα και να έχουμε όσο το δυνατό καλύτερη απόδοση του συστήματος.

- **Sticky Session:** Ο HAproxy σε κάθε αίτημα (request) το οποίο δέχεται ελέγχει άμα το αίτημα (request) έχει session από κάποιον διακομιστή (server). Αν έχει τότε συνεχίζει να στέλνει τα αιτήματα του συγκεκριμένου χρήστη στον διακομιστή. Αυτό πολύ σημαντικό για την ομαλή λειτουργία μιας εφαρμογής. Για παράδειγμα όταν ένας χρήστης έχει συνδεθεί σε ένα διακομιστή και έχει περάσει την διαδικασία της αυθεντικοποίησης του χρήστη , τότε θα πρέπει όλα τα αιτήματα του συγκεκριμένου χρήστη να στέλνονται στον συγκεκριμένο διακομιστή. Ειδάλλως άμα τον στείλει σε άλλον θα πρέπει να ξαναπεράσει πάλι την διαδικασία της αυθεντικοποίησης του χρήστη.

- **Default/ Failover Backend:** Σε όλες τις περιπτώσεις χρήσης έχουν προβλεφθεί και εναλλακτικοί διακομιστές ώστε να υπάρχει εξασφάλιση ότι δεν θα υπάρχει αίτημα χρήστη το οποίο δεν θα εξυπηρετηθεί από κάποιον διακομιστή, εφόσον φυσικά υπάρχει τουλάχιστον έναν διακομιστής ο οποίος είναι λειτουργικός και είναι σε θέση να εξυπηρετήσει κάποιο αίτημα. Έτσι σε κάθε backend διακομιστή

υπάρχει και ένας εφεδρικός διακομιστής για να δεκτεί τα αιτήματα του χρήστη καθώς και μια default ομάδα διακομιστών όπου είναι σε θέση βάση διαθεσιμότητας να δεκτή το αίτημα του χρήστη σε περίπτωση που κάτι πάει στραβά.

- **Health Check:** Ο HAproxy ελέγχει εάν όλοι οι διακομιστές είναι σε θέση να δεχτούν και να εξυπηρετήσουν τα αιτήματα των χρηστών. Στις συγκεκριμένες ρυθμίσεις που έχουν παρατεθεί παραπάνω ,ο HAproxy ελέγχει για την “υγεία” των διακομιστών κάθε 10 δευτερόλεπτα. Ο HAproxy έχει συγκεκριμένη διαδικασία όπου ελέγχει την κατάσταση των διακομιστών. Στέλνει συγκεκριμένο τύπου αιτήματος (request) και περιμένει συγκεκριμένου τύπου απάντηση ώστε να βεβαιωθεί ότι ο διακομιστής είναι σε θέση να εξυπηρετήσει τα αιτήματα των χρηστών. Αν διαπιστώσει ότι κάποιος διακομιστής δεν είναι σε θέση να εξυπηρετήσει τα αιτήματα αυτά τότε ο HAproxy δεν θα στείλει σε αυτόν κάποιο αίτημα (request). Θα ξαναστείλει σε αυτόν κάποιο αίτημα (request) όταν διαπιστώσει ότι είναι σε θέση να δεχτεί ξανά και να εξυπηρετήσει αιτήματα.

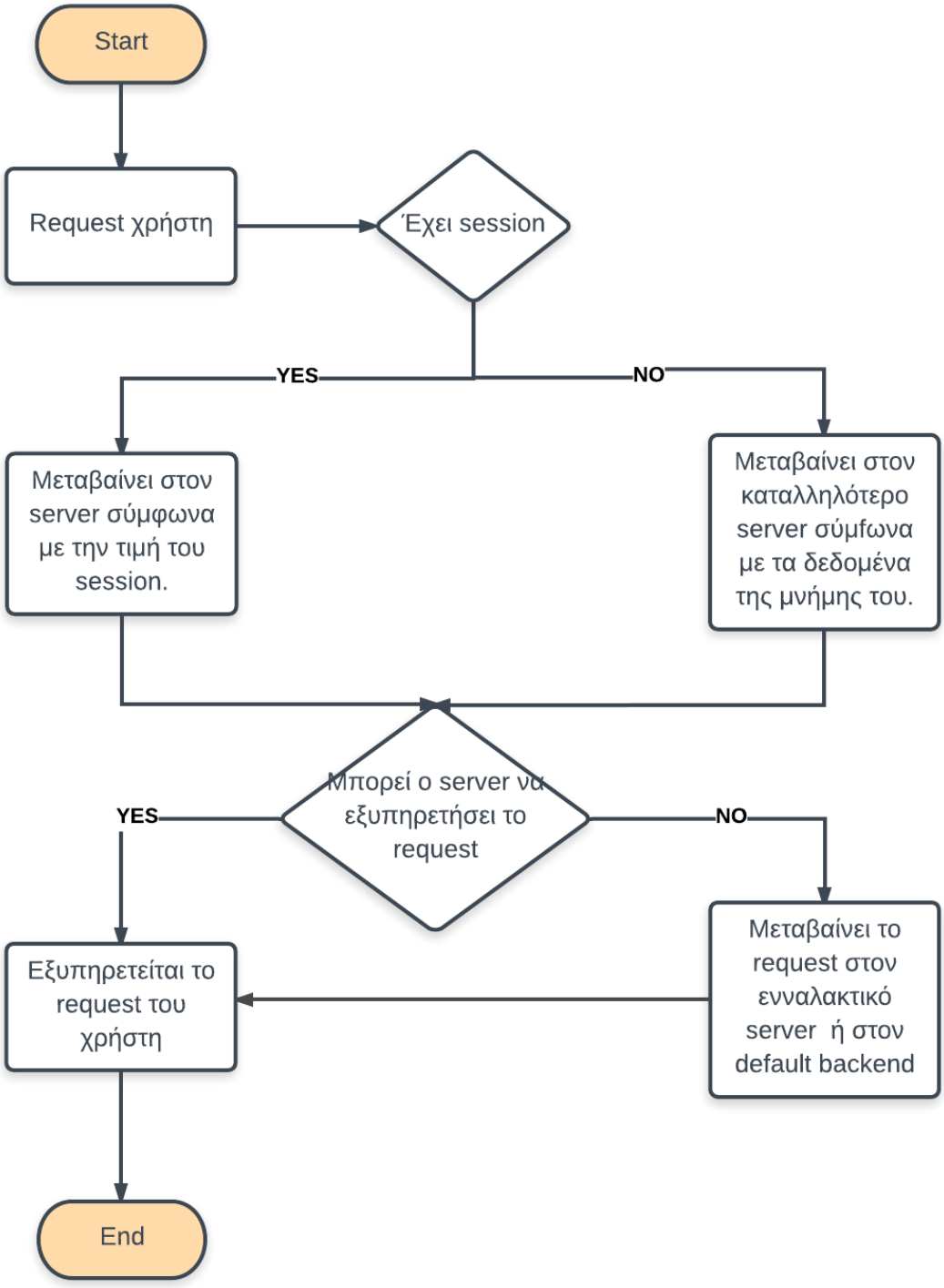
4.2.4. Λειτουργικότητα

Όταν ο Αντίστροφος Διακομιστής Μεσολάβησης (Reverse Proxy) ξεκινάει, διαβάζει μέσω ενός script από ένα αρχείο όπου του λέει ποιος είναι ο καταλληλότερος διακομιστής για τον συγκεκριμένο αντίστροφο διακομιστή μεσολάβησης (reverse proxy). Το script αυτό είναι γραμμένο στην προγραμματιστική γλώσσα Lua που υποστηρίζει ο HAproxy. Τα δεδομένα τα οποία διαβάζει από το script κρατιούνται στην μνήμη του HAproxy και είναι άμεσα διαθέσιμα οπότε αυτά χρειαστούν.

Όταν έρχεται ένα αίτημα (request) από έναν χρήστη τότε ο HAproxy ελέγχει αν έχει session με κάποιον διακομιστή. Αν δεν έχει, τότε ανακατευθύνει το αίτημα (request) του χρήστη στον καταλληλότερο διακομιστή σύμφωνα με τα δεδομένα που έχει στην μνήμη του από το Lua script. Αν το αίτημα (request) έχει session από κάποιον διακομιστή τότε ο HAproxy θα στείλει τα αιτήματα (request) στον

συγκεκριμένο διακομιστή. Αυτή η διαδικασία ακολουθείτε για όλα τα αιτήματα (request) τα οποία δέχεται ο HAproxy.

Αν τώρα κάποιο αίτημα χρήστη σταλθεί σε έναν διακομιστή ο οποίος δεν έχει την δυνατότητα να δεχτεί κάποιο αίτημα, τότε υπάρχει τουλάχιστον ένας εφεδρικός διακομιστής. Αν πάλι για κάποιο λόγο πάλι δεν μπορεί και εκείνος ο διακομιστής να δεχτεί τα αιτήματα του χρήστη τότε υπάρχει ένα “default backend” όπου θα εξυπηρετήσει τα αιτήματα αυτά.



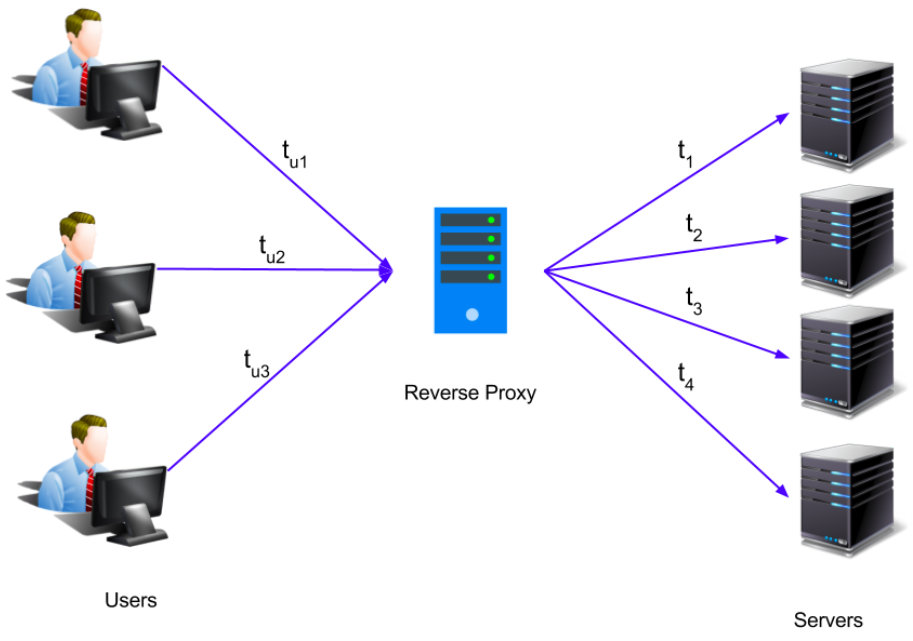
Εικόνα 23: Διάγραμμα εξυπηρέτησης αιτήματος χρήστη.

4.2.5. Προσέγγιση Λύσης

Στην παράγραφο αυτή γίνετε μια προσέγγιση του τρόπου εύρεσης του καταλληλότερου διακομιστή (server) για να εξυπηρετήσει τα αιτήματα των χρηστών.

Θα πρέπει να βρεθεί ένας τρόπος ώστε να ενημερώνεται το αρχείο από το οποίο διαβάζει το Lua script και στην συνέχεια να ενημερώνεται ξανά η τιμή στην μνήμη του HAproxy σχετικά με τον καταλληλότερο διακομιστή.

Επειδή όλα τα αιτήματα των χρηστών θα περνάνε μέσα από τον αντίστροφο διακομιστή μεσολάβησης, αρκεί να βρεθεί ποιος είναι ο καταλληλότερος διακομιστής για να εξυπηρετήσει τα υποθετικά αιτήματα που θα έκανε ο αντίστροφος διακομιστής μεσολάβησης (reverse proxy) προς τον διακομιστή. Και αυτό γιατί δεν μπορεί να είναι γνωστό τι γίνεται από τον χρήστη μέχρι τον αντίστροφο διακομιστή μεσολάβησης (reverse proxy) .



Εικόνα 24: Χρόνοι απόκρισης

Επειδή αυτό το οποίο έχει σημασία είναι η συνολική εμπειρία του χρήστη και το ποιος διακομιστής είναι εκείνος ο οποίος θα μπορέσει να εξυπηρετήσει πιο γρήγορα τα αιτήματα (request) του χρήστη , πρέπει να μετρηθεί ο συνολικός χρόνος τον οποίο κάνει η μία σελίδα αναφοράς να φορτωθεί. Σύμφωνα με τον χρόνο αυτό μπορεί και βρεθεί η βαθμολογία Apdex του συγκεκριμένου διακομιστή. Ταυτόχρονα

μετριέται και ο μέσος φόρτος που έχει ο διακομιστής καθώς και ο ρυθμός αύξησης ή μείωσης του φόρτου του διακομιστή.

Πιο συγκεκριμένα για να βρεθεί ο καταλληλότερος διακομιστής, μετριέται δέκα φορές ο χρόνος που χρειάζεται για να φορτωθεί η σελίδα αναφοράς. Έχει τεθεί ο χρόνος T τον οποίο απαιτεί η μεθοδολογία Apdex που αναφέρθηκε παραπάνω και έτσι μπορεί να βγει την βαθμολογία Apdex για τον συγκεκριμένο διακομιστή. Κάθε φορά που γίνεται μέτρηση καταγράφεται ο χρόνος που κάνει να φορτωθεί η σελίδα αναφοράς ενώ ταυτόχρονα καταγράφεται και ο φόρτος που έχει κάθε διακομιστής εκείνη την στιγμή. Ακόμα δίνεται ιδιαίτερη σημασία στην πρώτη και στην τελευταία μέτρηση του φόρτου του διακομιστή καθώς και στον συνολικό χρόνο τον οποίο χρειάζεται για να τελειώσουν και οι δέκα μετρήσεις.

Η μέτρηση του φόρτου που αναφέρθηκε δεν είναι η στιγμιαία μέτρηση του φόρτου αλλά ο φόρτος που έχει ο διακομιστής το τελευταίο λεπτό. Η στιγμιαία μέτρηση του φόρτου θα ήταν πλασματική καθώς η τιμή του φόρτου μπορεί να αλλάζει στιγμιαία για διάφορους λόγους. Οπότε η μέτρηση του φόρτου το τελευταίο λεπτό μας δίνει μια πιο ξεκάθαρη εικόνα σχετικά με τον φόρτο τον οποίο έχει ο διακομιστής.

Έχοντας συλλέξει όλη αυτή την πληροφορία υπάρχουν κάποια πολύ σημαντικά δεδομένα τα οποία επιτρέπουν να βγουν κάποια ασφαλή συμπεράσματα σχετικά με τον καταλληλότερο διακομιστή. Τα δεδομένα που έχουμε είναι:

- **Βαθμολογία Apdex.** Η βαθμολογία είναι ο πιο σημαντικός παράγοντας για να υποδείξει ποιος είναι ο καταλληλότερος διακομιστή γιατί ουσιαστικά αυτή η πληροφορία υποδηλώνει ποιος διακομιστής (server) μπορεί να εξυπηρετήσει καλύτερα τα αιτήματα του χρήστη. Η βαθμολογία Apdex υπολογίζεται σύμφωνα με τον τύπο .

$$Apdex_T = \frac{Satisfied + Tolerating\ /2}{Total\ Samples}$$

- **Μέσος φόρτος του διακομιστή:** Κάθε φορά που μετριέται ο χρόνος φόρτωσης της σελίδας αναφοράς μετριέται και ο φόρτος του διακομιστή στο τελευταίο λεπτό. Έτσι έχοντας δέκα μετρήσεις μπορεί να βγει και ο μέσος φόρτος που έχει ο διακομιστής (server) . Τον μέσω φόρτο τον μετράμε σύμφωνα με τον παρακάτω τύπο:

$$Load = \frac{\sum_{1}^{10} Load}{10}$$

- **Μέσος ρυθμός αύξησης ή μείωσης του φόρτου του διακομιστή**. Κατά την διάρκεια των μετρήσεων κρατιέται και ο συνολικός χρόνος ο οποίος χρειάζεται για να υλοποιηθούν και οι δέκα μετρήσεις στον κάθε διακομιστή (server). Έχοντας την πρώτη αλλά και την τελευταία τιμή της μέτρησης του φόρτου του διακομιστή μπορεί να βρεθεί ο μέσος ρυθμός αύξησης ή μείωσης του φόρτου του διακομιστή σύμφωνα με τον παρακάτω τύπο:

$$avgLoad = \lim \left(\frac{Load_{10} - Load_{1}}{t_{10} - t_{1}} \right)$$

Το $Load_{10}$ είναι ο φόρτος που έχει ο διακομιστής στη δέκατη φορά που φορτώνεται η σελίδα αναφοράς , $Load_1$ είναι ο φόρτος όταν φορτώνεται για πρώτη φορά η σελίδα αναφοράς και t_{10} - t_1 είναι ο χρόνος ο οποίος χρειάζεται για να γίνουν οι μετρήσεις.

4.2.6. Ανάλυση μετρικών

Οι μετρικές οι οποίες αναφέρθηκαν στην προηγούμενη παράγραφο δεν έχουν επιλέγει τυχαία αλλά μετά από αρκετές πειραματικές μετρήσεις και προσπάθεια ερμηνείας των αποτελεσμάτων με άλλες μετρικές.

Η βαθμολογία Apdex δίνει την πληροφορία του κατά πόσο γρήγορα μπορούν να εξυπηρετηθούν τα αιτήματα του χρήστη και είναι μια μετρική η οποίο παρέχει πληροφορία από τον αντίστροφο διακομιστή μεσολάβησης (reverse proxy) μέχρι τον διακομιστή (server). Αυτή όμως και μόνο η μετρική δεν είναι αρκετή ώστε να βγουν

ασφαλή συμπεράσματα για κάποιο διακομιστή (server). Και αυτό γιατί και ο φόρτος τον οποίο έχει ο διακομιστής είναι πολύ υψηλής σημασίας για την απόδοση του διακομιστή (server). Για παράδειγμα μπορεί ένας διακομιστής να έχει βαθμολογία Apdex 0.9 και η τιμή του φόρτου να είναι υψηλή , ενώ ένας άλλος διακομιστής να έχει και αυτός βαθμολογίας Apdex 0.9 αλλά η τιμή του φόρτου να είναι αρκετά υψηλή. Οπότε θα πρέπει να επιλεγεί ο διακομιστής εκείνος του οποίου η τιμή του φόρτου είναι χαμηλή.

Μια άλλη παρατήρηση η οποία έγινε είναι ότι έχει σημασία να μετριέται και ο ρυθμός μεταβολής του φόρτου του διακομιστή. Και αυτό γιατί όταν για παράδειγμα ένας διακομιστής έχει υψηλό και συνεχώς αυξανόμενο φόρτο τότε ο φόρτος του τελευταίου λεπτού αυξάνεται σταδιακά. Όταν όμως σταματήσει ξαφνικά ο φόρτος τότε ο διακομιστής (server) θα μπορεί να εξυπηρετήσει με πολύ καλή βαθμολογία Apdex τα αιτήματα των χρηστών έχοντας όμως ένα μεγάλο μέσο φόρτο. Ο ρυθμός όμως μείωσης του φόρτου τότε θα είναι μεγάλος. Οπότε άμα έχει έναν μεγάλο φόρτο αλλά και μια μεγάλη μείωση στο ρυθμό μεταβολής του φόρτου , τότε θα είναι σε θέση να εξυπηρετήσει με πολύ καλή βαθμολογία Apdex τα αιτήματα των χρηστών.

4.3. Πειραματικές Μετρήσεις

4.3.1 Υλοποίηση μετρήσεων

Για να μπορέσουν να γίνουν οι μετρήσεις υλοποιήθηκε ένα πρόγραμμα στην προγραμματιστική γλώσσα Java το οποίο δίνει την δυνατότητα να συλλεχθούν τα πειραματικά δεδομένα. Οι πειραματικές μετρήσεις έγιναν από τον αντίστροφο διακομιστή μεσολάβησης όπως θα γινόταν και σε πραγματικές συνθήκες.

Ο κώδικας ο οποίος πραγματοποιεί τις μετρήσεις αυτές είναι:

```
import java.io.FileWriter;
import java.io.IOException;
public class MainForServer2 {

    private static final String SERVER2 = "10.0.0.7";
    private static final String FILENAME =
"/home/user/filename.txt";
```

```
    private static final double T = 1;
    public static void main(String[] args) throws IOException {
        String finalString = "";
        try {
            double satisfied = 0;
            double tolerating = 0;
            double finalScore = 0;
            double averageLoad = 0;

            FileWriter fw = new FileWriter(FILENAME, true);
            try {
                for (int i = 0; i < 10; i++) {
                    System.out.println(i);
                    double rt = 0;
                    //Measure the load time for the reference page
                    rt = ApdexMeasure.loadTime();
                    System.out.println(rt);
                    Thread.sleep(500);
                    double ut = 0;
                    String uptime = "";
                    //Measure the load of the server
                    uptime = Uptime.callUptime(SERVER2, 22,
SERVER2);
                    System.out.println("Uptime----->" + uptime);
                    if (uptime.contains(",")) {
                        uptime = uptime.replace(",", ".");
                    }
                    ut = Double.valueOf(uptime);
                    if (rt <= T) {
                        satisfied += 1;
                    }

                    if (rt < 4 * T && rt > T) {
                        tolerating += 1;
                    }
                    averageLoad += ut;
                }
                 //Calculate the final Apdex Score
                finalScore = (satisfied + (tolerating / 2)) / 10;
```

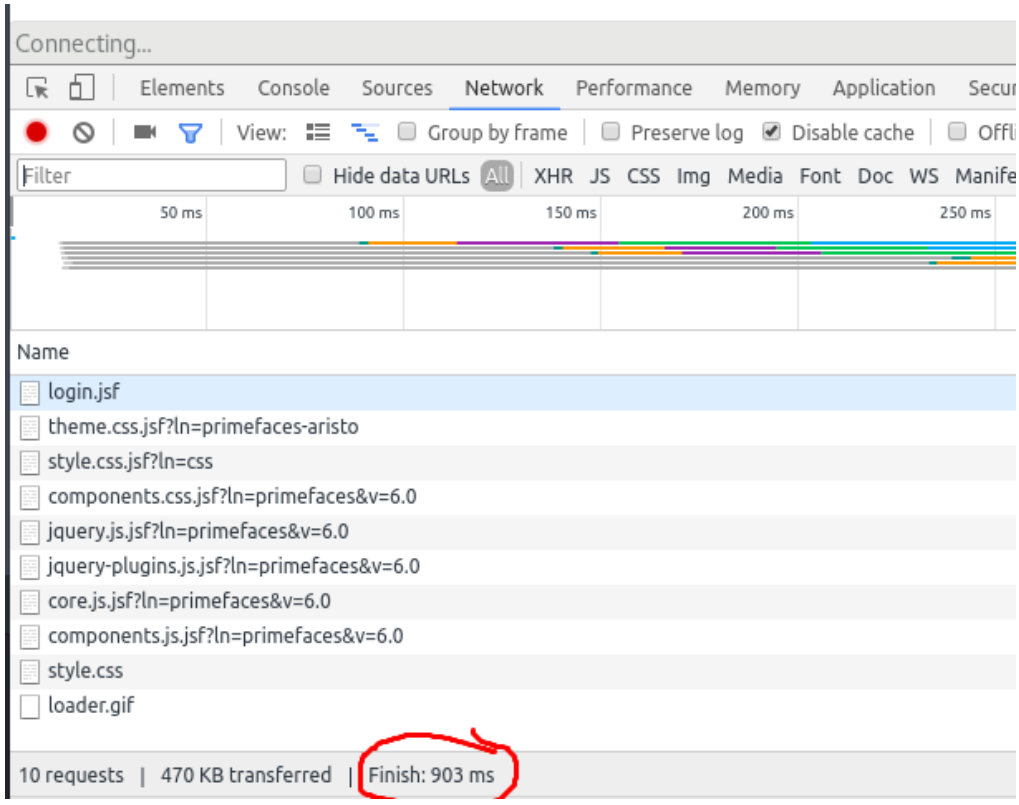
```
                System.out.println("Final Score : " + finalScore);
                fw.write("Apdex Score   |   Load       | \n");
                fw.write(finalScore + "           |   " +
averageLoad / 10 + "   | \n");
            } catch (Exception ex) {
                System.out.println(ex);
               } finally {
                fw.close();
            }
            System.out.println("Command Executed");
            System.out.println("DONE");
            System.out.println(finalString);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Στον παραπάνω κώδικα μετριέται ο χρόνος τον οποίο κάνει να φορτωθεί η σελίδα αναφοράς και βγαίνει η βαθμολογία Apdex. Για να μετρηθεί ο χρόνος τον οποίο κάνει να φορτωθεί η σελίδα αναφοράς χρησιμοποιείται ο παρακάτω κώδικας:

```
    public static double loadTime() throws InterruptedException {
        WebDriver driver = null;
        try {

            File file = new
File("/home/user/phantomjs-2.1.1-linux-x86_64/bin/phantomjs");
            System.setProperty("phantomjs.binary.path",
file.getAbsolutePath());

            DesiredCapabilities caps = new DesiredCapabilities();
            caps.setJavascriptEnabled(true);
            caps.setCapability("cssSelectorsEnabled", false);
            caps.setCapability("applicationCacheEnabled", true);
            caps.setCapability("acceptSslCerts", true);
```

```
            driver = new PhantomJSDriver();
            driver = new PhantomJSDriver(caps);
            driver.get("http://10.0.0.7:8080/lab/login.jsf");
            final JavascriptExecutor js = (JavascriptExecutor)
driver;
            // time of the process of navigation and page load
            double loadTime = (Double) js.executeScript(
                    "return
(window.performance.timing.loadEventEnd -
window.performance.timing.navigationStart) / 1000");
            System.out.print(loadTime + " seconds \n");
            return loadTime;
        }catch (Exception ex){
            System.out.println("****************"+ ex);
        }finally {
            driver.quit();
        }
        return 0;
    }
```

Στον παραπάνω κώδικα μετριέται ο χρόνος τον οποίο κάνει για να φορτωθεί η σελίδα αναφοράς με την χρήση ενός headless browser ο οποίος είναι ο PhantomJs[38]. Ουσιαστικά σηκώνεται προγραμματιστικά ένας φυλλομετρητής και μετριέται ο χρόνος που θα έδειχνε και ένας κανονικός φυλλομετρητής όπως φαίνεται στη εικόνα 25.



Εικόνα 25: Χρόνος φόρτωσης σελίδας από τον Browser

Για να μετρηθεί ο φόρτος του διακομιστή χρησιμοποιείται η εντολή των linux “uptime”.

```
08:11:22 up 146 days, 34 min, 3 users, load average: 0.28, 0.45, 0.38
```

Η εντολή αυτή μας δίνει διάφορες πληροφορίες όπως είναι η ώρα , ο χρόνος τον οποίο είναι σε λειτουργία ο διακομιστής , οι χρήστες που είναι συνδεδεμένοι στον διακομιστή και ο φόρτος που έχει ο διακομιστής τα τελευταία 1 λεπτό , 5 λεπτά και 15 λεπτά αντίστοιχα [39].

Για να μετρηθεί ο φόρτος του διακομιστή το τελευταίο λεπτό χρησιμοποιείται ο παρακάτω κώδικας:

```
 public static String callUptime(String host, Integer port, String
server) throws JSchException, IOException {

        try {
```

```
            JSch jsch = new JSch();
            //Connect with ssh to the server
            Session session = jsch.getSession(USER, host, port);
            session.setPassword(PASSWORD);
            session.setConfig("StrictHostKeyChecking", "no");
            System.out.println("Establishing Connection...");
            session.connect();
            System.out.println("Connection established.");

            String finalstr = "";
            //Execute the command that we want
            Channel channel = session.openChannel("exec");
            System.out.println("Execute command 'uptime  " +
server);
            ((ChannelExec) channel).setCommand("uptime | grep -ohe
'load average[s:][: ].*' | awk '{ print $3 }'");

            channel.setInputStream(null);
            ((ChannelExec) channel).setErrStream(System.err);

            InputStream in = channel.getInputStream();
            channel.connect();

            byte[] tmp = new byte[1024];
            while (true) {
                while (in.available() > 0) {
                    int i = in.read(tmp, 0, 1024);
                    if (i < 0) break;
                    String tempString = new String(tmp, 0, i);
                    System.out.println(tempString);
                    finalstr += tempString;
                }
                if (channel.isClosed()) {
                    System.out.println("exit-status: " +
channel.getExitStatus());
                    break;
                }
            }
            in.close();
            channel.disconnect();
            //Take the value that we want
```

```
            String t = finalstr.substring(0, finalstr.length() -
2);

System.out.println("----------------------------------------");
                System.out.println("Server " + server + " average
load in last 1 minute:" +t );

System.out.println("----------------------------------------");
                session.disconnect();
                System.out.println("Command Executed");
                System.out.println("DONE");
                return t;
        } catch (Exception ex) {
                System.out.println(ex);
        }
        return "20";
    }
```

Στον κώδικα αυτό γίνεται προγραμματιστική σύνδεση με ssh στον διακομιστή τον οποίο πρέπει να υλοποιηθεί η μέτρηση του φόρτο και εκτελείται η εντολή “uptime”, παίρνοντας μόνο την πληροφορία η οποία χρειάζεται, δηλαδή τον φόρτο του διακομιστή στο τελευταίο λεπτό.

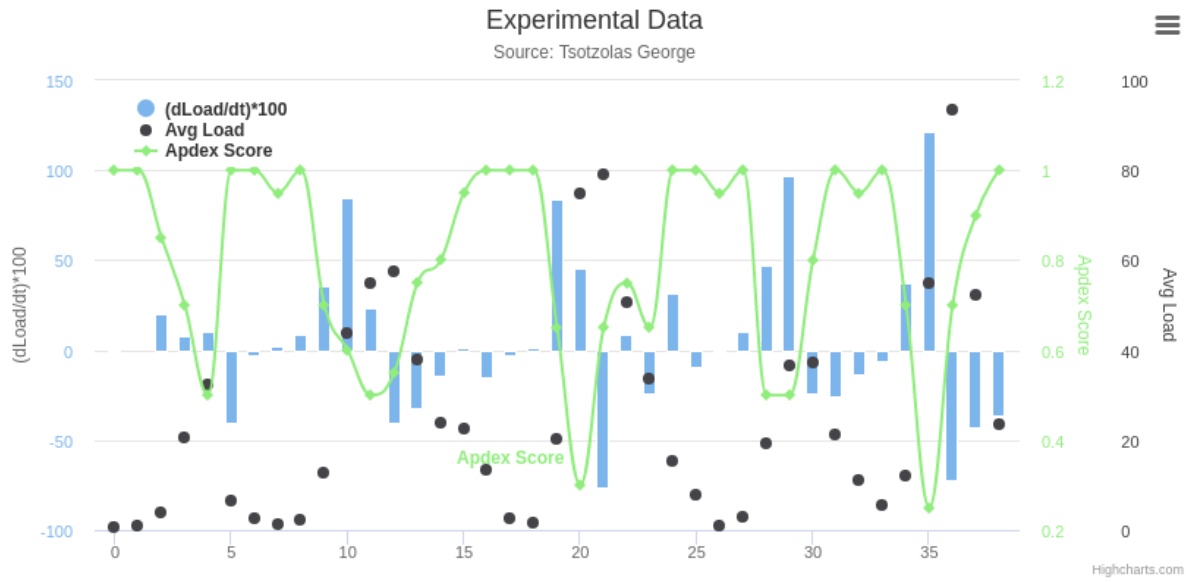
4.3.2. Μετρήσεις

Τα αποτελέσματα των πειραματικών μετρήσεων φαίνονται στον παρακάτω πίνακα.

Apdex Score	AvgLoad	(dLoad/dt)*100
1	0,642	0,22
1	0,875	-0,27
0,85	3,951	20,36
0,7	20,586	7,56
0,5	32,239	10,68
1	6,512	-40,33

1	2,585	-2,63
0,95	1,266	2,02
1	2,294	8,82
0,7	12,772	35,96
0,6	43,932	84,40
0,5	54,853	23,51
0,55	57,64	-40,63
0,75	37,884	-32,53
0,8	23,942	-13,85
0,95	22,566	1,51
1	13,32	-15,33
1	2,714	-2,72
1	1,699	1,15
0,65	20,33	83,78
0,3	74,891	45,62
0,65	79,214	-76,42
0,75	50,795	8,32
0,65	33,605	-24,17
1	15,253	31,22
1	7,959	-9,43
0,95	867	0,51
1	2,874	10,30
0,5	19,201	47,12
0,5	36,603	96,57
0,8	37,418	-24,02
1	21,122	-25,32
0,95	11,145	-13,24
1	5,503	-6,25
0,7	12,019	37,19
0,25	54,849	121,40
0,7	93,319	-72,08
0,9	52,446	-42,70
1	23,4	-36,48

Στο παρακάτω διάγραμμα έγινε μια προσπάθεια γραφικής αναπαράστασης των μετρήσεων αυτών



Εικόνα 26: Απεικόνιση πειραματικών μετρήσεων

4.3.3 Ανάλυση πειραματικών μετρήσεων

Βλέποντας προσεκτικά τα πειραματικά δεδομένα παρατηρείται ότι υπάρχει μια σχέση μεταξύ της βαθμολογίας Apdex και των άλλων δύο μετρικών, του μέσου φόρτου και του ρυθμού μεταβολής του φόρτου. Όταν υπάρχει μικρός μέσος φόρτος και μικρή αύξηση του ρυθμού του φόρτου υπάρχει καλή βαθμολογία Apdex. Όταν υπάρχει μεγάλος φόρτος και μεγάλη αύξηση του ρυθμού μεταβολής του φόρτου υπάρχει μικρή βαθμολογία Apdex. Ακόμα παρατηρούμε ότι όταν ο φόρτος είναι μεγάλος και ο ρυθμός μείωσης του φόρτου μειώνεται πολύ , τότε βελτιώνεται και η βαθμολογία Apdex. Οπότε ίσως θα μπορούσε να βρεθεί η σχέση που έχει η βαθμολογία Apdex σε σχέση με τις άλλες δύο μετρικές και να υπολογιστεί με σχετική ακρίβεια από αυτές.

Με την βοήθεια ενός νευρωνικού δικτύου βρέθηκε η σχέση που έχουν τα δεδομένα αυτά μεταξύ τους με ένα σφάλμα 28% η οποίο είναι η ακόλουθη:

$$Apdex\,Score = 0.7965 + (0.0002 \times avgLoad) - (0.0025 \times (dLoad/dt) * 100)$$

Θεωρητικά θα μπορούσε να αφαιρεθεί η μέτρηση του Apdex Score, αφού μπορεί να μετρηθεί με σχετική ακρίβεια από τις άλλες δύο παραμέτρους αλλά κάτι τέτοιο θα ήταν λάθος. Και θα ήταν λάθος γιατί άμα αφαιρεθεί η παράμετρος του

Apdex Score δεν θα λαμβάνεται καθόλου υπόψιν η ταχύτητα του δικτύου. Η μετρήσεις που έγιναν στο πειραματικό περιβάλλον είχαν σταθερή ταχύτητα δικτύου. Σε πραγματικές συνθήκες όμως η ταχύτητα του δικτύου θα μεταβάλλεται συνεχώς καθώς και η ταχύτητά μεταξύ του αντίστροφου διακομιστή μεσολάβησης και των διακομιστών μπορεί να ποικίλει. Άρα δεν είναι δυνατή η αφαίρεση κάποιας από τις τρεις μετρικές.

4.4 Εύρεση καταλληλότερου διακομιστή

Στην παράγραφο αυτή θα δοθούν οι τελευταίες λεπτομέρειες για την εύρεση του καταλληλότερου διακομιστή. Όπως έχει αναφερθεί και παραπάνω αυτό το οποίο είναι μέγιστης σημασίας είναι να βρεθεί ο διακομιστής ο οποίος θα μπορέσει να εξυπηρετήσει πιο γρήγορα τα αιτήματα του χρήστη.

Ο αντίστροφος διακομιστής μεσολάβησης (reverse proxy) θα πρέπει να γνωρίσει ποιοι διακομιστές εξυπηρετούν την εφαρμογή. Ακόμα θα πρέπει να υπάρχει μια περιοδική διαδικασία η οποία θα κάνει μετρήσεις και θα συλλέγει τις πληροφορίες που αναφέρθηκαν παραπάνω (Apdex Score , μέσος φόρτος και ρυθμός μεταβολής του φόρτου) για κάθε διακομιστή (server). Τέλος θα πρέπει να υπάρχει μία διαδικασία η οποία θα λαμβάνει τα δεδομένα αυτά , θα υπολογίζει βάση μιας σχέσης τον καταλληλότερο διακομιστή , θα ενημερώνει το αρχείο με τον καταλληλότερο διακομιστή από το οποίο διαβάζει το Lua Script του HAproxy και θα κάνει επανεκκίνηση (reload) του HAproxy ώστε να ενημερωθεί για τα νέα δεδομένα. Η περιοδικότητα της διαδικασίας αυτής θα πρέπει να μελετηθεί από τον οργανισμό ανάλογα του αριθμού των διακομιστών (server) , του χρόνου που χρειάζεται για να γίνει καθώς και γενικότερο φόρτο των διακομιστών .

Η διαδικασία που θα υπολογίζει τον καταλληλότερο διακομιστή θα πρέπει, αφού λάβει τις μετρήσεις από τους διακομιστές, να υπολογίζει βάση μιας σχέσης τον καταλληλότερο διακομιστή. Στη σχέση αυτή η ύπαρξη μεγάλου Apdex Score θα πρέπει να λαμβάνεται σαν θετικό στοιχείο στην επιλογή του διακομιστή. Ακόμα ο

μέσος φόρτος θα πρέπει να λαμβάνεται σαν αρνητικό στοιχείο για τον υπολογισμό. Τέλος ο ρυθμός μεταβολής του φόρτου θα πρέπει να λαμβάνεται σαν αρνητικό στοιχείο όταν είναι θετικά αυξανόμενος ενώ θα πρέπει να λαμβάνεται σαν θετικό στοιχείο όταν μειώνεται.

Μια σχέση η οποίο μπορεί να μας δώσει το προαναφερθέν αποτέλεσμα είναι η ακόλουθη.

$$Selection = Apdex\,Score + \frac{10}{avgLoad} - \frac{dLoad}{dt}$$

Έτσι βάση της παραπάνω σχέσης ο διακομιστής (server) ο οποίος θα έχει το μεγαλύτερο αποτέλεσμα θα είναι και ο καταλληλότερος. Η παραπάνω σχέση βγήκε μετά από αξιολόγηση και αξιοποίηση των πειραματικών δεδομένων.

5. Μελλοντικές Ενέργειες.

Στην παρούσα διπλωματική υλοποιήθηκε ένα δοκιμαστικό περιβάλλον ώστε να μπορέσει να προσομοιωθεί όσο το δυνατό το πραγματικό περιβάλλον. Παρόλα αυτά, κάποια πράγματα δεν θα μπορούσαν να γίνουν όπως ακριβώς στο πραγματικό περιβάλλον . Και αυτό γιατί απαιτείτε ένα μεγάλο δίκτυο από διακομιστές κάτι το οποίο δεν ήταν εφικτό να πραγματοποιηθεί σε πειραματικό περιβάλλον. Ακόμα στο πραγματικό περιβάλλον θα υπάρχει ένα δίκτυο το οποίο είναι χαμηλής χωρητικότητας και η ταχύτητα του οποίου δεν θα μπορεί να είναι γνωστή στους διαχειριστές του συστήματος.

Τα επόμενα βήματα που πρέπει να γίνουν είναι να υλοποιηθεί η προτεινόμενη αρχιτεκτονική στο δίκτυο του οργανισμού και να γίνουν δοκιμές σε πραγματικές συνθήκες και με αρκετούς διακομιστές. Θα πρέπει να αξιολογηθούν ξανά τα πειραματικά δεδομένα και να αναθεωρηθούν ξανά οι μετρικές ώστε να μπορέσει και να βελτιστοποιηθεί και ο χρόνος ο οποίος χρειάζεται για να γίνουν οι μετρήσεις. Πιο συγκεκριμένα στην προτεινόμενη λύση που δόθηκε στην παρούσα εργασία, για να εξαχθεί η πληροφορία του Apdex Score πρέπει να φορτωθεί δέκα φορές η σελίδα αναφοράς , κάτι το οποίο χρειάζεται χρόνο και δημιουργεί και επιπλέον φόρτο στο δίκτυο. Θα πρέπει να μελετηθεί μήπως είναι δυνατή η προσθήκη μιας ακόμα μετρικής η οποίο θα συνυπολογίζει την ταχύτητα του δικτύου μεταξύ του αντίστροφου διακομιστή μεσολάβησης (reverse proxy) και του διακομιστή (server), ώστε να μπορεί να βγει από τις μετρήσεις αυτές το Apdex Score. Ακόμα άμα παραμείνει η μετρική του Apdex Score θα πρέπει να αξιολογηθεί ξανά και ο χρόνος T τον οποίο απαιτεί η μέτρησή του.

Τέλος θα πρέπει να γίνουν δοκιμές με πραγματικά σενάρια ώστε να διαπιστωθεί η αξιοπιστία του συστήματος πριν μπει στην παραγωγή, καθώς επίσης

θα πρέπει να υπάρχει μια δοκιμαστική περίοδος ώστε να μπορέσουν να επιλυθούν πιθανά προβλήματα.

6. Επίλογος

Στην παρούσα διπλωματική εργασία έγινε προσέγγιση σε ένα πρόβλημα το οποίο πρόκειται να αντιμετώπιση ένας μεγάλος οργανισμός στην Ελλάδα μετά την αλλαγή της τοπολογίας του από κεντρικοποιημένη σε πλήρως κατανεμημένη. Αναλύθηκε η παρούσα τοπολογία του οργανισμού καθώς και η μελλοντική τοπολογία των συστημάτων του. Αναφέρθηκε ένα σενάριο του προβλήματος που θα αντιμετωπίσει ο οργανισμός και έγινε ανάλυση του προβλήματος αυτού.

Αναζητήθηκαν οι τεχνολογίες οι οποίες έχουν επιλύσει παρόμοια προβλήματα στο διαδικτύου και αναλύθηκαν οι σχετικές τεχνολογίες. Πιο συγκεκριμένα αναλύθηκε η λειτουργία των Δικτύων Κατανομής Περιεχομένου (Content Delivery Networks) , των Αντίστροφων Διακομιστών Μεσολάβησης (Reverse Proxy) και έγινε αναφορά στην απόδοση εφαρμογών διαδικτύου (web applications).

Παρουσιάστηκε μια προτεινόμενη λύση, αναλύθηκε η λειτουργία και η αρχιτεκτονική του συστήματος που πρέπει να υλοποιηθεί και δόθηκαν οι επιμέρους λεπτομέρειες. Έγινε ανάλυση του δοκιμαστικού περιβάλλοντος το οποίο δημιουργήθηκε για αυτό τον σκοπό καθώς και των ρυθμίσεων που έγιναν στον HAproxy ο οποίος είναι ένας αντίστροφος διακομιστής μεσολάβησης (reverse proxy). Στη συνέχεια αναζητήθηκαν και βρέθηκαν κατάλληλες μετρικές οι οποίες επιτρέπουν να βγουν κάποια ασφαλή συμπεράσματα για την εύρεση καταλληλότερου διακομιστή ο οποίος θα είναι σε θέση να εξυπηρετήσει πιο γρήγορα τα αιτήματα των χρηστών. Έγιναν πειραματικές μετρήσεις ώστε να βρεθούν οι σχέσεις που έχουν οι μετρικές μεταξύ τους και έγινε ανάλυση των αποτελεσμάτων αυτών. Τέλος προτάθηκε ένας τύπος βάση του οποίου θα μπορεί να βρεθεί ο προτεινόμενος εξυπηρετητής (server).

Τέλος αναφέρθηκαν οι μελλοντικές ενέργειες οι οποίες πρέπει να γίνουν προκειμένου το μπορέσει το μπει στην παραγωγική διαδικασία το προτεινόμενο σύστημα.

Παραπομπές

[1] What is a Content Delivery Network? http://www.cdnreviews.com/what-is-cdn/

[2] Πώς λειτουργούν οι CDNs http://entercdn.com/el/cdn/how-cdn-works

[3] ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ Δίκτυα Διανομής Περιεχομένου(CDN) (Πουλερέ Ευαγγελία Κουτούμπα Όλγα.)
http://apothetirio.teiep.gr/xmlui/bitstream/handle/123456789/4862/1358.pdf?sequence=1

[4] What is a CDN
https://www.incapsula.com/cdn-guide/what-is-cdn-how-it-works.html

[5] What Is Reverse Proxy Server
https://www.incapsula.com/cdn-guide/glossary/reverse-proxy.html

[6]Load Balancing
http://tutorials.jenkov.com/software-architecture/load-balancing.html

[7] REST better: HTTP cache http://odino.org/rest-better-http-cache/

[8] How content delivery networks (CDNs) work
https://www.nczonline.net/blog/2011/11/29/how-content-delivery-networks-cdns-work

[9] CDN Caching https://www.incapsula.com/cdn-guide/cdn-caching.html

[10] What Is Anycast https://www.incapsula.com/cdn-guide/glossary/anycast.html

[11] Anycast https://en.wikipedia.org/wiki/Anycast

[12] DNS Performance
http://hinrg.cs.jhu.edu/joomla/projects/past-projects/154-dns.html

[13] Συμπίεση δεδομένων https://el.wikipedia.org/wiki/Συμπίεση_δεδομένων

[14] Http Compression https://en.wikipedia.org/wiki/HTTP_compression

[15] IANA https://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority

[16] Compression https://developer.mozilla.org/en-US/docs/Web/HTTP/Compression

[17]Brotli Compressed Data Format
https://www.rfc-editor.org/rfc/pdfrfc/rfc7932.txt.pdf
[18] Compression Benchmarks: brotli, gzip, xz, bz2
https://www.opencpu.org/posts/brotli-benchmarks/
[19] CDN Infrastructure https://www.incapsula.com/cdn-guide/cdn-architecture.html
[20] Rethinking CDN Design with Distributed Time-Varying Traffic Demands http://wulab.cs.uvic.ca/cdn.pdf
[21] Variety Digital News, “Netflix bandwidth usage climbs to nearly 37% of internet traffic at peak hours
http://variety.com/2015/digital/news/netflix-bandwidth-usage-internet-traffic-1201507187/
[22] An Anomaly-driven Reverse Proxy for Web Applications
https://dl.acm.org/citation.cfm?id=1141361
[23] What Is A Reverse Proxy Vs. Load Balancer
https://www.nginx.com/resources/glossary/reverse-proxy-vs-load-balancer/
[24] Sticky Sessions http://wiki.metawerx.net/wiki/StickySessions
[25] Load Balancing Algorithms
https://kemptechnologies.com/emea/load-balancer/load-balancing-algorithms-techniques/
[26] Difference between proxy server and reverse proxy server
https://stackoverflow.com/questions/224664/difference-between-proxy-server-and-reverse-proxy-server/366212#366212
[27] Response Time vs. Latency
http://www.javidjamae.com/2005/04/07/response-time-vs-latency/
[28] Why response times are often measured incorrectly
https://www.dynatrace.com/blog/why-response-times-are-often-measured-incorrectly/
[29] Response time in man-computer conversational transactions
http://yusufarslan.net/sites/yusufarslan.net/files/upload/content/Miller1968.pdf
[30] A Methodology for Determining Response Time Baselines: Defining the “8 Second” Rule
https://pdfs.semanticscholar.org/6bf0/c34fa3396fd366a1dfa52de71f61c0f9aac3.pdf
[31] Defining The Application Performance Index
http://apdex.org/docs/Defining_The_Application_Performance_Index.pdf
[32] Web Application Performance: What Apdex Doesn’t Tell You

https://www.coscale.com/blog/web-application-performance-what-apdex-doesnt-tell-you

[33] Apdex: Measuring user satisfaction

https://docs.newrelic.com/docs/apm/new-relic-apm/apdex/apdex-measuring-user-satisfaction

[34] Οκεανος https://okeanos.grnet.gr/home/

[35] Haproxy http://www.haproxy.org/

[36] HAproxy https://en.wikipedia.org/wiki/HAProxy

[37] Lua http://www.lua.org/

[38] OpenSSL https://www.openssl.org/

[38] PhantomJs http://phantomjs.org/

[39] Understanding Linux CPU Load - when should you be worried?

http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages

[40] http://wildfly.org/

[41] JavaServer Faces Technology

http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html