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

Αυτόματη διαχείριση δικτυακών τοπολογιών και υπηρεσιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γρηγόρης-Παναγιώτης Δ. Μαντάος

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

Αθήνα, Σεπτέμβριος 2018

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

Αυτόματη διαχείριση δικτυακών τοπολογιών και υπηρεσιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γρηγόρης-Παναγιώτης Δ. Μαντάος

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Σεπτεμβρίου 2018.

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

Αθήνα, Σεπτέμβριος 2018

.....
Γρηγόρης-Παναγιώτης Δ. Μαντάος

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

Περίληψη

Η χρήση εικονικών μηχανών τόσο για προσομοίωση, όσο και για κανονική ρύθμιση και χρήση δικτυακών εξαρτημάτων, όπως δρομολογητές και γέφυρες, έχει επηρεάσει σημαντικά τα τελευταία χρόνια το ρόλο των μηχανικών και των διαχειριστών συστημάτων. Οι εικονικοί δρομολογητές είναι λογισμικό που εκτελείται σε εικονικές μηχανές και υποστηρίζει όλα τα πρωτόκολλα και τις λειτουργίες ενός φυσικού δρομολογητή. Η ευελιξία που παρέχουν οι τεχνολογίες εικονικοποίησης (virtualization) επιτρέπουν τη γρήγορη και εύκολη ανάπτυξη τέτοιου λογισμικού σε φυσικούς εξυπηρετητές δίνοντας τη δυνατότητα σε ερευνητές και σπουδαστές να πειραματιστούν με δικτυακές τοπολογίες, χωρίς να περιορίζονται από τους πόρους των τοπικών τους μηχανημάτων. Συνδυάζοντας την άρση των φυσικών περιορισμών με τα εργαλεία και τεχνικές της ανάπτυξης λογισμικού, μας δίνεται πλέον η δυνατότητα να αντιμετωπίσουμε αυτά τα δικτυακά εξαρτήματα με ένα επίπεδο αφαίρεσης (abstraction), δημιουργώντας, ρυθμίζοντας και διαγράφοντας εικονικές μηχανές κατά βούληση. Σκοπός της εργασίας αυτής είναι η δημιουργία ενός γραφικού περιβάλλοντος διαχείρισης, που θα επιτρέπει τον σχεδιασμό δικτυακών τοπολογιών μεταβλητού μεγέθους και πολυπλοκότητας, καθώς και την αυτόματη υλοποίηση των τοπολογιών αυτών στο διαθέσιμο υλικό (hardware).

Λέξεις Κλειδιά

Αυτοματοποίηση, Δικτυακές τοπολογίες, Γραφικό περιβάλλον, Εικονικοί δρομολογητές, VyOS, Ansible.

Abstract

The use of virtual machines for simulation, as well as regular configuration and use of networking components such as routers and bridges, has significantly affected the role of systems engineers and administrators over the last few years. Virtual routers are essentially software running on virtual machines and support all of the routing protocols and functions of a physical router. The flexibility that these virtualization technologies provide allows for the fast and seamless development of such software on physical hosts, giving researchers and students alike the means to experiment with network topologies without being limited by the resources of physical machines. Combining this lift of physical limitations with the tools and methodologies of software development, we now have the ability to treat such networking components with a layer of abstraction, creating, configuring and deleting them at will. The purpose of this project is to develop an administrative graphical user interface that will allow the design of network topologies of variable size and complexity as well the automated deployment of said topologies on the available hardware.

Keywords

Automation, Network topologies, Graphical user interface, Virtual routers, VyOS, Ansible.

Περιεχόμενα

Ευρετήριο Εικόνων

Κεφάλαιο 1

Αυτοματοποίηση Δικτυακών Εφαρμογών

1.1 Software-Defined Networking

Με την επέκταση της δικτύωσης και του διαδικτύου, η απαιτούμενη υποδομή από την προοπτική της δικτύωσης είναι συνήθως περίπλοκή στην σχεδίαση και την εγκατάσταση. Είναι δύσκολη και η παραμετροποίηση των στοιχείων του δικτύου, ώστε να παρέχεται η επιθυμητή δικτύωση, αλλά και η προσαρμογή του σε σφάλματα, αλλαγές στην τοπολογία και επιβαρύνσεις στον όγκο της ροής της πληροφορίας. Επιλύοντας τα παραπάνω προβλήματα, προέκυψε η ανάγκη διαχωρισμού του επιπέδου διαχείρισης και παραμετροποίησης ενός δικτύου από τα στοιχεία του δικτύου αυτού, που επιτελούν τη δρομολόγηση και την προώθηση της κίνησης. Με την εισαγωγή των μεταγωγέων στην τεχνολογική στοίβα της δικτύωσης υπολογιστών, έγινε δυνατός ο έλεγχος των δικτύων από υψηλό επίπεδο, με χρήση προγραμματιστικών διεπαφών. Οι διεπαφές αυτές επιτρέπουν την προγραμματιστική διαχείριση ενός δικτύου, εισάγοντας, έτσι, την έννοια των ορισμένων από λογισμικό δικτύων, **software-defined networks (SDN)**. Η προσέγγιση αυτή για την ενορχήστρωση ενός δικτύου κάνει εύκολη την προσθήκη νέων επιπέδων αφαιρετικότητας (abstraction), τα οποία απλοποιούν την περιγραφή και την εγκατάσταση ενός δικτύου, επιτρέπουν την επίλυση προβλημάτων υψηλότερου επιπέδου και διευκολύνουν την εξέλιξη των δικτύων [22].

Ο διαχωρισμός του πεδίου ελέγχου (control plane) και του επιπέδου δεδομένων (data plane) επιτυγχάνεται από μία καλώς ορισμένη προγραμματιστική διεπαφή μεταξύ των κόμβων του δικτύου και του ελεγκτή. Ο ελεγκτής αυτός παρέχει το λογισμικό, το οποίο ελέγχει την κατάσταση των συσκευών, μέσω της διεπαφής αυτής. Ο διαχωρισμός αυτός είναι σημαντικός για την ανάπτυξη ενός δικτύου, καθώς διαχωρίζει πλήρως τη μέριμνα της σχεδίασης και της περιγραφής των πολιτικών ενός δικτύου από την εφαρμογή τους στο υποκείμενο υλικό. Μέσω της προσέγγισης αυτής, απλοποιείται, επίσης, η περιγραφή δικτυακών πολιτικών, καθώς εγκαταλείπει τις περίπλοκες τεχνολογίες της δικτύωσης υπολογιστών και πλησιάζει την επιστήμη της ανάπτυξης λογισμικού.

1.2 Εικονικοποίηση Δικτυακών Υπηρεσιών

Η εικονικοποίηση υπηρεσιών στο διαδίκτυο (**Network Function Virtualization, NFV**) προτίθεται να βελτιώσει την ευελιξία των υπηρεσιών αυτών, μειώνοντας παράλληλα το χρόνο που απαιτούσαν για εγκατάσταση και εμπόρευση. Χρησιμοποιώντας εμπορικές τεχνολογίες και προγραμματιζόμενα υπολογιστικά συστήματα (hardware), η προσέγγιση του NVF διαχωρίζει την προγραμματιστική υλοποίηση των υπηρεσιών από το υποκείμενο hardware στο οποίο παρέχονται. Οι προκλήσεις που προκύπτουν από την εισαγωγή της εικονικοποίησης είναι η απόδοση των εικονικών αυτών υπηρεσιών, η δυναμική τους εγκατάσταση, η μετακίνησή τους αλλά και η βέλτιστη κατανομή τους στο διαθέσιμο υλικό [23]. Οι τεχνικές NVF συμβαδίζουν με τις προαναφερθείσες τεχνολογίες SDN, απαλείφοντας παράλληλα τους περιορισμούς του υλικού. Μία φυσική συσκευή μπορεί να φιλοξενεί πολλαπλές εικονικές δικτυακές συσκευές και υπηρεσίες, τις οποίες μπορεί να

εγκαθιστά, να μετακινεί και να διαγράφει κατά βούληση. Οι διαχειριστές ενός τέτοιου δικτύου απαλλάσσονται, επίσης, από άλλους φυσικούς περιορισμούς, όπως ο αριθμός των φυσικών διεπαφών μίας δικτυακής συσκευής, καθώς οι τεχνολογίες εικονικοποίησης επιτρέπουν την διασύνδεση εικονικών (paravirtualized) συσκευών στις εικονικές μηχανές. Μία υπηρεσία που υλοποιείται ως NFV σχεδιάζεται έτσι, ώστε να παρέχει μία ή περισσότερες εικονικές λειτουργίες (**Virtual Network Functions, VNFs**). Οι λειτουργίες αυτές δεν αποτελούν από μόνες τους μία υπηρεσία, που είναι εμπορεύσιμη στους χρήστες του εκάστου παρόχου, αλλά χρησιμοποιούνται σε συνοχή, με πολλά VNF μαζί να παρέχουν κάποια υπηρεσία.

Στα πλαίσια της εργασίας αυτής, θα υλοποιηθεί η εγκατάσταση εικονικών ιδιωτικών δικτύων (**Virtual Private Network, VPN**). Τα δίκτυα αυτά αποτελούν μία τέτοια εικονική υπηρεσία, παρέχοντας τη δυνατότητα εγκατάστασης εικονικών ζεύξεων από ένα άκρο σε ένα άλλο, με διευθυνσιοδότηση που ξεπερνά τα φυσικά όρια των τοπικών δικτύων των εν λόγω υπολογιστικών συστημάτων. Έτσι, παρέχεται η δυνατότητα ασφαλούς συνδεσιμότητας των συστημάτων ενός οργανισμού με εικονικές διευθύνσεις σε εικονικές ζεύξεις, που είναι abstract ως προς τη διαδρομή που ακολουθεί η πραγματική ροή πληροφορίας στο δημόσιο διαδίκτυο. Συγκεκριμένα, η εργασία θα επικεντρωθεί στην υλοποίηση των εικονικών δικτύων, από την οπτική της αυτοματοποίησης της εγκατάστασης (deployment) των εικονικών μηχανών στο διαθέσιμο υλικό.

1.3 Αυτοματοποίηση Δικτυακής Υποδομής

Η αυτόματη διαχείριση των δικτυακών πόρων και τοπολογιών είναι σημαντικό κομμάτι της ανάπτυξης οργανισμών οποιουδήποτε μεγέθους, που σκοπεύουν να κάνουν την υποδομή τους ευέλικτη, κλιμακώσιμη και ανθεκτική σε βλάβες. Σε πολλές περιπτώσεις, η ανάπτυξη εφαρμογών στα υποκείμενα δίκτυα πραγματοποιείται ακόμα με χειροκίνητες ενέργειες, οι οποίες μπορούν να καθυστερήσουν την ανάπτυξη, να αυξήσουν το κόστος τους και να τις κάνουν ευάλωτες σε ανθρώπινα σφάλματα [24]. Η αυτοματοποίηση των δικτύων αυξάνει την ανθεκτικότητά τους σε σφάλματα και αλλαγές και εξασφαλίζει αυξημένη παραγωγικότητα των ανθρώπων που αναλαμβάνουν την ανάπτυξή τους. Η αυτοματοποιημένη εφαρμογή κάποιας παραμετροποίησης εξασφαλίζει, επίσης, **idempotence**, κάτι που σημαίνει ότι οι επαναλαμβανόμενες εκτελέσεις τις πραγματοποιούνται ομοιόμορφα και οδηγούν το σύστημα στην ίδια ακριβώς κατάσταση κάθε φορά. Με τις τεχνικές αυτές, η δικτυακή υποδομή μπορεί να αποσυναρμολογείται, να επεκτείνεται και να εγκαθίσταται, χωρίς να απαιτείται ανθρώπινη παρέμβαση στην λεπτομέρεια της παραμετροποίησης. Η προσέγγιση αυτή είναι η λογική συνέχεια του SDN. Δηλαδή, το λογισμικό που αναπτύσσεται με σκοπό τη διαχείριση του δικτύου μπορεί να αποτελέσει τα τμήματα της διαδικασίας αυτοματοποίησης κάποιας δικτυακής εγκατάστασης.

Στα πλαίσια της εργασίας αυτής, θα δοθεί έμφαση στο πώς θα υλοποιηθεί η αυτοματοποίηση της εγκατάστασης των επιθυμητών VPN. Αναπτύσσοντας την εφαρμογή στα πλαίσια του SDN, η εργασία αυτή θα λειτουργήσει ως διεπαφή που παρέχει τον αυτοματισμό αυτό, δίνοντας, έτσι, στον τελικό χρήστη την δυνατότητα να υλοποιήσει abstract σχεδίαση των επιθυμητών τοπολογιών, χωρίς μέριμνα για το πώς οι τοπολογίες αυτές θα εγκατασταθούν στο διαθέσιμο αυτό υλικό. Η αυτόματη αυτή διαχείριση μίας τοπολογίας δίνει στο χρήστη τη δυνατότητα πειραματισμού με δίκτυα μεταβλητού μεγέθους και πολυπλοκότητας.

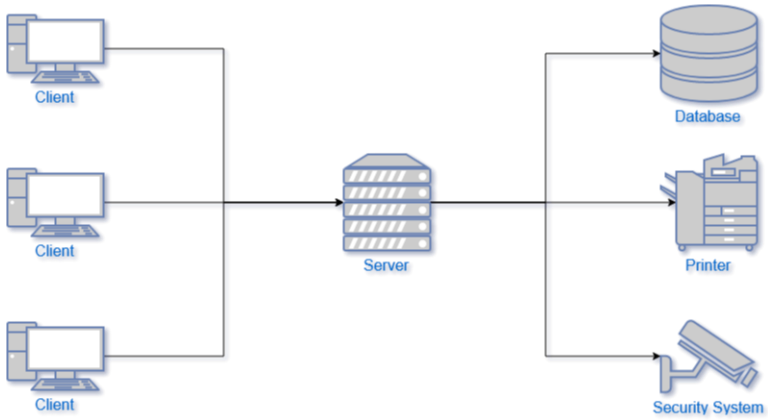
Κεφάλαιο 2

Περιγραφή της Εφαρμογής και Επιλογές Ανάπτυξης

Στα πλαίσια αυτής της εργασίας, θα γίνει η ανάπτυξη λογισμικού, το οποίο θα επιτελεί μία υπηρεσία αυτόματης διαχείρισης δικτυακών τοπολογιών. Μέσω της υπηρεσίας αυτής, κάποιος χρήστης θα έχει τη δυνατότητα να σχεδιάσει κάποια δικτυακή τοπολογία αποτελούμενη από κόμβους συνδεδεμένους μεταξύ τους και να προσδιορίσει διευθύνσεις για τους κόμβους στις ζεύξεις αυτές. Ουσιαστικά, θα επιτρέπει την δημιουργία ενός εικονικού ιδιωτικού δικτύου (**virtual private network, VPN**). Ένα τέτοιο δίκτυο, επεκτείνει ένα ιδιωτικό δίκτυο πάνω από ένα δημόσιο, όπως το διαδίκτυο. Επιτρέπει έμμεσα στους υπολογιστές που συμμετέχουν σε αυτό, να ανταλλάσουν δεδομένα μεταξύ τους σαν να ήτανε άμεσα συνδεδεμένοι σε κάποιο τοπικό ή ιδιωτικό δίκτυο. Επομένως, εφαρμογές που λειτουργούνε επικοινωνώντας μέσω κάποιου VPN εκμεταλλεύονται τη λειτουργικότητα, την ασφάλεια και τη διαχείριση ενός ιδιωτικού δικτύου, ακόμη και αν εκτείνονται στο δημόσιο διαδίκτυο. Στα πλαίσια της εργασίας αυτής, το ενδιαφέρον για τα VPN πηγάζει από ακαδημαϊκό ενδιαφέρον, τόσο για το θεωρητικό υπόβαθρο της μοντελοποίησης και της εγκατάστασης των δικτύων αυτών ως μία εικονική υπηρεσία (NFV), αλλά και για το εκπαιδευτικό περιεχόμενο των υποδομών αυτών, οι οποίες επιτρέπουν την μελέτη δικτύων μεταβλητής κλίμακας ανεξάρτητα από τους υποκείμενους φυσικούς περιορισμούς.

Η τοπολογία που θα δημιουργείται στα πλαίσια της εργασίας θα είναι μεταβλητού μεγέθους και πολυπλοκότητας και θα υλοποιείται χρησιμοποιώντας εικονικούς δρομολογητές με ελεύθερη πρόσβαση στο διαδίκτυο. Το δίκτυο αυτό θα μοντελοποιείται από το λογισμικό που εκτελείται στο περιβάλλον του χρήστη, χρησιμοποιώντας τυποποιημένα μοντέλα για την περιγραφή του, τα οποία ύστερα θα υποβάλλονται στην υπηρεσία για δημιουργία. Το λογισμικό της υπηρεσίας, ύστερα, θα υλοποιεί την τοπολογία, που έχει σχεδιάσει ο χρήστης, εγκαθιστώντας την στο διαθέσιμο υλικό (hardware), μέσω εικονικών μηχανών και εκτελώντας την κατάλληλη παραμετροποίηση στις μηχανές αυτές, ώστε να δημιουργηθεί το επιθυμητό εικονικό ιδιωτικό δίκτυο. Το υλικό που θα φιλοξενεί τις εικονικές μηχανές θα μπορεί να αποτελείται από πολλαπλούς εξυπηρετητές απομακρυσμένους μεταξύ τους στο διαδίκτυο, οπότε το ιδιωτικό δίκτυο που θα δημιουργείται, θα έχει τις ζεύξεις του εγκατεστημένες στο επίπεδο δικτύου. Επεκτείνοντας τη λειτουργικότητα του δικτύου αυτού, οι κόμβοι θα είναι μία λίστα από εικονικούς δρομολογητές, που εκτελώντας κάποιο λειτουργικό σύστημα (πχ. UNIX), θα επιτρέπουν την απομακρυσμένη πρόσβαση (πχ. telnet, SSH), δίνοντας έτσι τη δυνατότητα στο χρήστη να εργαστεί κατευθείαν πάνω στο δίκτυο το οποίο εγκατέστησε. Η ανάπτυξη θα γίνει σε εφαρμογή ιστού (**web application**), προσβάσιμη από φυλλομετρητή (web browser). Η επιλογή αυτή έγινε κυρίως για την προσιτότητα και την διαθεσιμότητα των εφαρμογών αυτών από όλου του είδους τις συσκευές. Επίσης, με τα νέα framework που αναπτύσσονται, οι γλώσσες για απεικόνιση περιεχομένου στο διαδίκτυο είναι πλέον φορητές και επαναχρησιμοποιήσιμες σε πολλές τεχνολογικές στοίβες.

2.1 Το Μοντέλο Πελάτη-Διακομιστή



1. Απεικόνιση Μοντέλου Client-Server

Με μέριμνα για την εμπειρία του χρήστη, αλλά και για την ακεραιότητα της κρίσιμης λογικής του προγράμματος (business logic), οι υπηρεσίες μέσω διαδικτύου (web services) πολύ συχνά βασίζουν την αρχιτεκτονική τους στο μοντέλο πελάτη-διακομιστή (**client-server model**). Σύμφωνα με το μοντέλο αυτό, το περιβάλλον στο οποίο εργάζεται ο χρήστης (πχ. η ιστοσελίδα, client) λειτουργεί μόνο ως μέσω διεπαφής με τον εξυπηρετητή που παρέχει την υπηρεσία. Μάλιστα, στην περίπτωση των υπηρεσιών ιστού, όπου ο χρήστης αλληλεπιδρά με ιστοσελίδες, ο ίδιος ο διακομιστής ιστού (web server) σερβίρει τον κώδικα της ιστοσελίδας σε κάθε επίσκεψη. Η δομή αυτή εξασφαλίζει ότι ο κώδικας που εκτελείται από τη μεριά του client δεν θα περιέχει ποτέ διαδικαστική λογική (business logic). Κάποιοι από τους λόγους για τους οποίους αυτή η προσέγγιση είναι πολύ σημαντική είναι οι παρακάτω:

- **Ενημέρωση του business logic**
  Η λογική του κώδικα που εκτελεί κρίσιμες διεργασίες, όπως ρύθμιση κάποιου συστήματος ή ενημέρωση κάποιας βάσης δεδομένων, οφείλει να υπακούει σε ανασκοπήσεις, αναθεωρήσεις και ενημερώσεις από τους διαχειριστές. Η συμφωνία του business logic, που εκτελείται με τις προδιαγραφές της εφαρμογής, δεν μπορεί να εγγυηθεί εάν το λογισμικό που μοιράζεται στις συσκευές των χρηστών περιέχει τον κώδικα αυτό.

- **Έλεγχος πρόσβασης**
  Το λογισμικό που επιτελεί κάποια υπηρεσία, συνήθως, απαιτεί διαχειριστική πρόσβαση, με δικαιώματα ελέγχου σε πόρους, όπως μια βάση δεδομένων ή αρχεία συστήματος. Το εν λόγω μοντέλο το εξασφαλίζει αυτό, διότι δεν απαιτείται κατοχή των πιστοποιητικών του συστήματος από τον χρήστη, καθώς και η πρόσβαση στους πόρους αυτούς πραγματοποιείται μόνο από τον server.

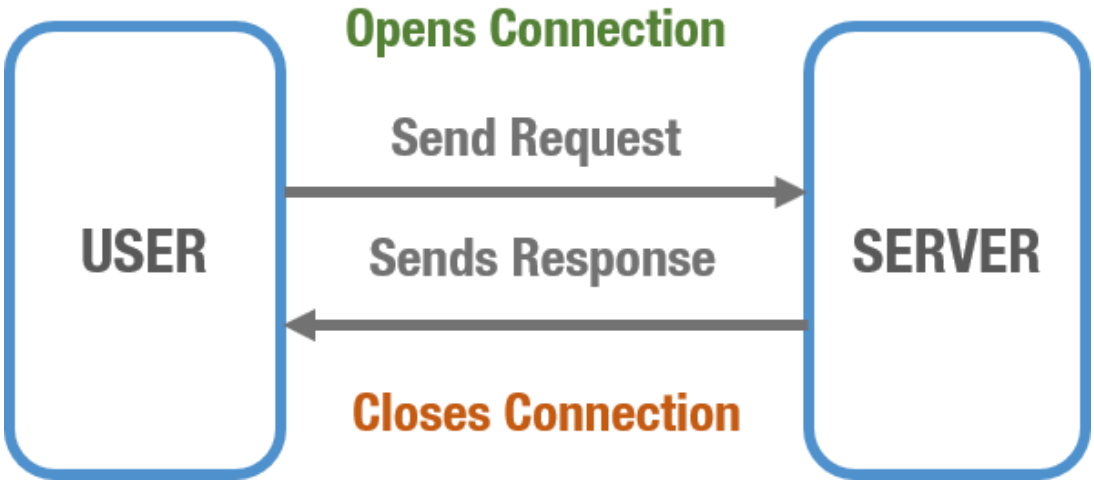
- **Εγκυρότητα της κατάστασης της εφαρμογής και των πόρων**
  Στο μοντέλο αυτό, ο server λειτουργεί ως η επίσημη αρχή (authoritative server), από τον οποίο λαμβάνουν ενημερώσεις οι χρήστες μέσω ερωτημάτων. Αυτό εξασφαλίζει ότι η κατάσταση του συστήματος είναι μονοσήμαντα ορισμένη και ταυτίζεται με την κατάσταση που «αποφασίζει» ο authoritative server.

Η λειτουργία του μοντέλου αυτού στην πράξη είναι απλή. Το λογισμικό διεπαφής του χρήστη (**front-end**) επικοινωνεί με το λογισμικό του διακομιστή (**back-end**) μέσω διακριτών αιτημάτων **(requests**), τα οποία απαντώνται από τον server με διακριτές απαντήσεις (**responses**).

2.1.1 Διεπαφή Προγραμματισμού Εφαρμογών HTTP

Στα πλαίσια της εργασίας, θα πραγματοποιηθεί η ανάπτυξη εφαρμογής ιστού (web application), με το περιβάλλον, από το οποίο ο χρήστης έχει πρόσβαση σε αυτήν, να είναι ιστοσελίδα (web page). Η λογική επιλογή για την υλοποίηση της επικοινωνίας μεταξύ του client και του server, είναι το πρωτόκολλο HTTP. Το πρωτόκολλο αυτό είναι το ίδιο που χρησιμοποιείται από τον web browser του χρήστη για την επικοινωνία με τον web server της εφαρμογής και έχει γνήσια (native) υποστήριξη από τις γλώσσες και τα πρότυπα του W3C για το διαδίκτυο. Τα HTTP requests, πέρα από τις επικεφαλίδες που μπορεί να προσδιορίζουν καλύτερα το είδος του ζητούμενου πόρου, αποτελούνται κυρίως από δύο τμήματα:

- Ένα **request URL,** το οποίο απεικονίζει μονοσήμαντα τον ζητούμενο πόρο. Για παράδειγμα, το URL "**/api/products/list"** θα μπορούσε να χρησιμοποιείται για να ζητά μία λίστα με προϊόντα και το URL **"/api/products/5"** για να ζητά το προϊόν με κωδικό 5.
- Το **request body** (φορτίο) του αιτήματος, το οποίο περιέχει δεδομένα (data), όπως απεικονίσεις αντικειμένων, που δεν γίνεται να κωδικοποιηθούν στο URL.



2. Μοντέλο Επικοινωνίας HTTP

Όπως φαίνεται στο σχήμα, το πρωτόκολλο HTTP θα χρησιμοποιείται από το front-end της εφαρμογής για να γίνονται requests προς τον server. Ένα HTTP request αντιπροσωπεύει είτε ένα αίτημα προς τον server για κάποια πληροφορία ή τις ενέργειες που επιθυμεί να εκτελεί ο χρήστης. Το πρωτόκολλο HTTP ορίζει μεθόδους τις οποίες υλοποιούν τα requests για το σημασιολογικό διαχωρισμό των λειτουργιών που προσφέρει το API. Στην περίπτωση της εργασίας αυτής, που το API δεν θα χειρίζεται αλλαγές ή διαγραφές αντικειμένων και μεταφόρτωση (upload) αρχείων, θα χρησιμοποιηθούν μόνο δύο από τις μεθόδους αυτές.

- **GET**
  Τα request μεθόδου GET χρησιμοποιούνται για να ζητηθεί κάποια απεικόνιση των πόρων του server. Τα ερωτήματα αυτά από τη μεριά του server πρέπει, κατά κανόνα, μόνο να ανακτούν ή να μορφοποιούν τα δεδομένα των πόρων, χωρίς να έχουν κάποια άλλη παρενέργεια ή συνέπεια. Τέτοιου είδους ερωτήματα είναι αυτά που εκτελεί ο web browser για την ανάκτηση ιστοσελίδων και συνήθως δεν περιέχουν request body. Επίσης, τα ερωτήματα αυτά από τη μεριά του server πρέπει να είναι idempotent, δηλαδή οι επαναλαμβανόμενες κλήσεις τους να μην επηρεάζουν το αποτέλεσμά τους.

- **POST**
  Τα request μεθόδου POST χρησιμοποιούνται για να αποσταλούν δεδομένα από τον client στον server, έναντι της μεθόδου GET, που διέπραττε το αντίστροφο. Κατά την απάντηση των ερωτημάτων αυτών, ο server διαβάζει το request body μέχρι το πέρας του, το οποίο συνήθως περιέχει την απεικόνιση κάποιου αντικειμένου, και δρα σύμφωνα με αυτό. Συνήθως, οι απαντήσεις σε αυτά τα ερωτήματα δεν περιέχουν response body, πέρα από κάποια ένδειξη για το αν το αίτημα έγινε αποδεκτό ή για το αν υπήρχε κάποιο σφάλμα.

2.2 Δυνατότητες της Εφαρμογής

Έχοντας περιγράψει τις απαιτήσεις, στο κεφάλαιο αυτό, θα συζητηθεί το πώς θα επιτευχθούν οι στόχοι της ανάπτυξης, που μένει να υλοποιηθεί. Σκοπός είναι η εύρεση ενός συνετού μέσου ανάμεσα στην θεωρία και στην πράξη. Η ανάπτυξη που θα γίνει, δεδομένου του ότι η εργασία αυτή είναι κυρίως κατασκευαστική, οφείλει να είναι πλήρως λειτουργική και εύχρηστη. Οφείλει, όμως, να διατηρήσει παράλληλα επιστημονικό και ερευνητικό χαρακτήρα. Σύμφωνα με αυτό, **η ανάπτυξη θα βασιστεί κυρίως στη σωστή μοντελοποίηση των αντικειμένων και των δεδομένων που θα χρησιμοποιηθούν, καθώς και στις σχεδιαστικές επιλογές που θα προκύψουν κατά την ανάπτυξη**.

Σημαντικός εδώ είναι ο διαχωρισμός των στόχων της ανάπτυξης από τους στόχους μιας υποθετικής εφαρμογής της ανάπτυξης αυτής. Θα αποφευχθεί δηλαδή η έμφαση και η εξαντλητικότητα στην ανάπτυξη τμημάτων του προγράμματος, των οποίων η υλοποίηση και οι περιορισμοί εξαρτώνται σημαντικά από τις συνθήκες και τις προτιμήσεις των συνθηκών υπό των οποίων θα χρησιμοποιηθεί το πρόγραμμα. Μερικά παραδείγματα τέτοιων τμημάτων ενός λογισμού, που εξαρτώνται από τις προτιμήσεις του περιβάλλοντος χρήσης, οπότε και **δεν θα εξεταστούν στα πλαίσια της εργασίας αυτής**, είναι τα παρακάτω:

- **Ενσωμάτωση (integration) της εφαρμογής με εξωτερικούς παράγοντες (third party vendors)**
  Σύνηθες κομμάτι στην ανάπτυξη εφαρμογών παραμετροποίησης και ελέγχου υπολογιστικών συστημάτων είναι η υλοποίηση μίας διεπαφής προγραμματισμού εφαρμογών (application programming interface, API), η οποία να επιτρέπει προγραμματιστική πρόσβαση στις λειτουργίες της εφαρμογής. Τα είδη των διεπαφών αυτών ποικίλουν. Μπορεί, για παράδειγμα, να έχουν τη μορφή ενός HTTP API ή μίας διαδραστικής γραμμής εντολών (command-line interface, CLI).

- **Έλεγχος πρόσβασης στην εφαρμογή**
  Το πρόγραμμα που θα αναπτυχθεί θα έχει άμεσο διαχειριστικό έλεγχο του συνδεδεμένου υλικού, οπότε σχεδόν οποιαδήποτε περίπτωση υλοποίησής του θα χρειαστεί κάποιου είδους έλεγχο πρόσβασης (access control) και πιθανώς κάποια διαχείριση πιστοποιητικών (credentials) για τους χρήστες της.

- **Ακεραιότητα, έλεγχος και ανάθεση των διαθέσιμων πόρων**
  Εάν το γραφικό περιβάλλον επιτρέπει την πρόσβαση σε πολλαπλούς χρήστες, το back-end κομμάτι της εφαρμογής πρέπει να λειτουργεί με κάποιο βαθμό παραλληλότητας. Στο επίπεδο του λογισμικού και των πόρων, αυτό δημιουργεί σημαντικά ερωτήματα τόσο για την ταυτόχρονη ρύθμιση πολλαπλών στοιχείων ενός συστήματος όσο και για την ανάθεση πόρων με μέριμνα για την αποφυγή συγκρούσεων.

- **Παρακολούθηση και τροποποίηση των αρχικοποιημένων τοπολογιών**
  Δεδομένου του ότι η έμφαση της εργασίας θα είναι στους τρόπους ρύθμισης και ενορχήστρωσης των δικτυακών τοπολογιών και το γεγονός ότι οι τροποποιήσεις των ρυθμίσεων αυτών θα ακολουθούν τις ίδιες τεχνικές χρησιμοποιώντας τις ίδιες τεχνολογίες, η υλοποίηση για την δυνατότητα τροποποίησης των δικτύων που θα δημιουργούνται θεωρείται, επίσης, εκτός του πεδίου εφαρμογής της εργασίας.

2.2.1 Δυνατότητες Γραφικού Περιβάλλοντος (Front-End)

Το γραφικό περιβάλλον (graphical user interface, **GUI**) της εφαρμογής πρέπει να προσφέρει στον χρήστη τη δυνατότητα να σχεδιάσει τοπολογίες σημαντικού μεγέθους, χωρίς περιττή προσπάθεια. Για να διασφαλιστεί η ποιότητα και η χρηστικότητα της εφαρμογής από τον τελικό χρήστη, το γραφικό περιβάλλον οφείλει να έχει τις παρακάτω δυνατότητες:

- Σχεδίαση δικτυακής τοπολογίας.
  - Προσθήκη, αφαίρεση και μετονομασία κόμβων.
  - Προσθήκη και αφαίρεση ζεύξεων μεταξύ των κόμβων.
  - Προσδιορισμός διεύθυνσης IP σε κάθε κόμβο για κάθε ζεύξη που προσπίπτει σε αυτόν.
- Δισδιάστατη απεικόνιση της υπό σχεδίασης τοπολογίας για ευδιάκριτο οραματισμό (**visualization**) κόμβων και των ζεύξεων μεταξύ τους.
- Ευδιάκριτη παρουσία των σημαντικών στοιχείων και δεδομένων πάνω στον χώρο σχεδίασης.
  - Ονομασίες (hostnames) των κόμβων.
  - Διευθύνσεις στις διεπαφές των κόμβων.
- Δυνατότητα αποθήκευσης (**export)** της τρέχουσας τοπολογίας.
- Δυνατότητα επαναφοράς (**import**) αποθηκευμένης τοπολογίας.
- Όψη παρακολούθησης της κατάστασης των εικονικών μηχανών της τοπολογίας σε πραγματικό χρόνο.
- Όψη παρακολούθησης για διαγνωστικά μηνύματα και την έξοδο (output) των εντολών παραμετροποίησης σε πραγματικό χρόνο.
- Δυνατότητα διαγραφής μίας αρχικοποιημένης τοπολογίας, μέσω αποστολής σχετικής εντολής στον server για την απενεργοποίηση και τη διαγραφή των εικονικών μηχανών.

Εξασφαλίζοντας τη χρησιμότητα, όπως αυτή περιγράφηκε παραπάνω, δεν θα εμβαθύνουμε άλλο στο γραφικό περιβάλλον. Όπως προαναφέρθηκε, περεταίρω αναπτύξεις στο λογισμικό του front-end απαιτούν επίγνωση του σεναρίου χρήσης (use case) και θα παραλειφθούν στα πλαίσια αυτής της εργασίας, ώστε το λογισμικό να παραμείνει αφηρημένο (abstract).

2.2.2 Δυνατότητες Server (Back-End)

Back-end μίας αρχιτεκτονικής αφηρημένα αποκαλείται το κομμάτι της εκείνο, που δεν εκτελείται σε περιβάλλον του χρήστη. Στην περίπτωση του μοντέλου client-server που εξετάζουμε, το λογισμικό του server, μαζί με τους διαχειριστικούς πόρους του, όπως βάσεις δεδομένων και εικονικές μηχανές, ανήκει στο τμήμα αυτό. Οι λειτουργίες του θα είναι η κατάλληλη αποκωδικοποίηση, εκτέλεση και απάντηση των ερωτημάτων (requests), που θα προέρχονται από το γραφικό περιβάλλον του χρήστη (front-end). Όπως έχει προαναφερθεί, η έμφαση θα δοθεί στη μοντελοποίηση, τη διαμόρφωση και τη ρύθμιση των τοπολογιών που θα σχεδιάζονται, με λίγη γενικά μέριμνα για την συνολική εμπειρία του χρήστη και ζητήματα που εξαρτώνται από το use case. Σύμφωνα με την παραδοχή αυτή και για να διατηρηθεί το λογισμικό abstract και επαναχρησιμοποιήσιμο, ο server θα επιτελεί κυρίως μία λειτουργία: την δημιουργία της τοπολογίας που υποβάλει ο χρήστης στο διαθέσιμο υλικό (hardware). Αφηρημένα, για κάθε τοπολογία θα εκτελείται η παρακάτω αλληλουχία γεγονότων:

1. Απολογισμός των απαιτούμενων πόρων και σύγκριση με τους διαθέσιμους.
2. Για κάθε εικονική μηχανή (virtual machine, **VM**) που απαιτείται:
   2.1. Δέσμευση δημόσιας διεύθυνσης IP για διαχειριστικό έλεγχο.
   2.2. Επιλογή φιλοξενούντος μηχανήματος (**host**) με επαρκείς διαθέσιμους πόρους.
   2.3. Σύνδεση στο host μηχάνημα.
   2.4. Εγγραφή του VM.
   2.5. Έναρξη (power on) του VM.
   2.6. Σύνδεση στο VM για απόδοση της δεσμευμένης IP διεύθυνσης.
3. Μαζική ρύθμιση των απαραίτητων interfaces, ώστε να επιτυγχάνεται η συνδεσιμότητα στην σχεδιασμένη τοπολογία.

Δεδομένων και των απαιτήσεων του front-end, ο server τελικά θα παρέχει τους εξής πόρους (endpoints) στο HTTP API του.

- **POST /api/create**
  Το endpoint αυτό θα δέχεται στο response body την σειριοποιημένη (serialized) απεικόνιση της προς-δημιουργία τοπολογίας, η μορφή της οποίας θα αναλυθεί στο επόμενο κεφάλαιο περί μοντελοποίησης.
  Response: Μοναδικός κωδικός (ID), που θα ανατεθεί στο αίτημα για αναφορά του client του χρήστη.

- **GET /api/logs/{id}**
  Με αυτό το endpoint, ο client θα ανακτά ενημερωτικά μηνύματα και output εντολών σχετικά με την πρόοδο της εγκατάστασης της τοπολογίας με κωδικό {id}.

Response: Λίστα με τις γραμμές του output, καθώς και κάποιος χαρακτηρισμός του συγκεκριμένου μηνύματος (πχ. πληροφορία, εντολή, μήνυμα σφάλματος).

- **GET /api/topology/{id}**
  Με αυτό το endpoint, ο client θα ανακτά την κατάσταση των εικονικών μηχανών της τοπολογίας με κωδικό {id}.
  Response: Λίστα με τις εικονικές μηχανές, περιέχοντας την ονομασία, την δημόσια IP και την κατάστασή τους (online/offline).

- **POST /api/delete/{id}**
  Εντολή διαγραφής τοπολογίας με κωδικό {id}. Η διαγραφή θα επιτελείται με την απενεργοποίηση και τη διαγραφή όλων των εικονικών μηχανών της τοπολογίας.

Επιπλέον, δεδομένης της ιδιότητας του server και του ρόλου του στην ενορχήστρωση (orchestration) των δικτυακών τοπολογιών, στο υπόλοιπο της εργασίας θα αναφέρεται και ως **ελεγκτής** (**controller**).

2.3 Απεικόνιση Δικτύου στο Υλικό

Στην ενότητα αυτή, θα γίνει η εξερεύνηση των μεθόδων, με τις οποίες θα επιτευχθεί η συνδεσιμότητα μεταξύ των εικονικών μηχανών, σύμφωνα με τις ζεύξεις και τις διευθύνσεις που έχει ορίσει ο χρήστης κατά τη σχεδίαση της τοπολογίας. Οι σχεδιαστικές αποφάσεις αυτές θα γίνουν με βάση τις τεχνολογίες και τις πρακτικές που αναφέρθηκαν στα προηγούμενα κεφάλαια της εργασίας. Τα κρίσιμα τμήματα της ανάπτυξης σε αυτό το σημείο είναι τα παρακάτω:

- Τι είδους συσκευές θα χρησιμοποιηθούν για την υλοποίηση των host.
- Με ποιόν τρόπο θα αποδοθούν οι διευθύνσεις που έχει προσδιορίσει ο χρήστη στις συσκευές αυτές.
- Πώς θα γίνει η δρομολόγηση στο ιδιωτικό δίκτυο που θα προκύψει, ώστε κάθε κόμβος να έχει πρόσβαση στους υπόλοιπους στις προσδιορισμένες διευθύνσεις.

Εξετάζοντας αυτές τις παραμέτρους, είναι συνετό να ληφθεί υπ’ όψιν η συμβατότητα της εφαρμογής αυτής με τα πρότυπα της βιομηχανίας των τηλεπικοινωνιών. Είναι σημαντικό η υλοποίηση των στοιχείων και των πόρων των δικτύων, που θα αναπτύσσονται, να γίνει με τέτοιο τρόπο, ώστε τα στοιχεία αυτά να μπορούν τόσο να μελετηθούν εύκολα από τρίτους, ερευνητές και επαγγελματίες του τομέα, αλλά και να διατηρούν συμβατότητα με το αντίστοιχο φυσικό hardware. Τηρώντας αυτές τις προδιαγραφές, η υλοποίηση θα παρουσιάζει ευελιξία και επαναχρησιμότητα, επιτρέποντας τόσο την ανάπτυξη υπό πολλαπλά σενάρια χρήσης, αλλά ακόμα και την αντικατάσταση των εικονικών μηχανών με πραγματικές φυσικές συσκευές δικτύωσης.

2.3.1 Συσκευές Δικτύου

Τα δίκτυα προς σχεδίαση αποτελούνται από κόμβους και ζεύξεις, απεικονίζοντας αφηρημένα (abstractly) έναν γράφο. Στο τμήμα αυτό, θα εξετάσουμε το πώς θα κατασκευάσουμε τους κόμβους αυτούς. Οι κόμβοι του δικτύου αυτού θα αποτελούν

δικτυακές συσκευές, οι οποίες θα πρέπει να ρυθμιστούν κατάλληλα, ώστε να παρέχουν την επιθυμητή δικτύωση της σχεδιασμένης τοπολογίας. Στο πρώτο κεφάλαιο της εργασίας, αναφέρθηκαν οι πιο συνήθεις συσκευές δικτύωσης που παρέχουν δυνατότητες προώθησης και δρομολόγησης πακέτων: η γέφυρα (**bridge**), ο μεταγωγέας (**switch**) και ο δρομολογητής (**router**).

Επέκταση της γέφυρας αποτελεί ο **μεταγωγέας (Switch)**, ο οποίος, πέρα από την απλή διασύνδεση τμημάτων δικτύου, μπορεί να εκτελέσει και λειτουργίες διανομέα (**Hub**). Οι μεταγωγείς λειτουργούν και σε επίπεδα πέρα του επιπέδου ζεύξης, δυνατότητα που επέτρεψε τον έλεγχο και τη διαχείρισή τους σε υψηλότερο επίπεδο. Με προγράμματα όπως το Openflow, ένας μεταγωγέας μπορεί να ρυθμιστεί διαχειριστικά ξεχωριστά από το επίπεδο διαχείρησης, στο υλικό που αποφασίζει την διαδρομή των δεδομένων.

- **Switch**
  Επέκταση της γέφυρας αποτελεί ο **μεταγωγέας (Switch)**, ο οποίος, πέρα από την απλή διασύνδεση τμημάτων δικτύου, μπορεί να εκτελέσει και λειτουργίες διανομέα (**Hub**). Η απομακρυσμένη διαχείριση, καθώς και η δυνατότητα ρύθμισης των πινάκων προώθησης του επιπέδου δικτύου του OSI καθιέρωσαν τους μεταγωγείς απαραίτητο στοιχείο ενός δικτύου και ενίσχυσαν την δυνατότητα της ορισμένης από λογισμικό δικτύωσης **(software-defined networking, SDN)**. Παραμένουν, όμως, συσκευές χαμηλού επιπέδου, λειτουργώντας μέσω υλικού και ηλεκτρονικών κυκλωμάτων, χωρίς λειτουργικό σύστημα.

- **Router**
  Το ρόλο ενός δρομολογητή μπορεί να επιτελέσει ένα οποιοδήποτε υπολογιστικό σύστημα, δεδομένου του ότι έχει συμβατό λειτουργικό σύστημα και πρόσβαση στο κατάλληλο λογισμικό. Η ευελιξία αυτή δίνει στο ίδιο σύστημα τη δυνατότητα να ελέγχεται και να ρυθμίζεται μέσω λογισμικού, καθώς και να επιτελεί πολλαπλές λειτουργίες ταυτόχρονα.

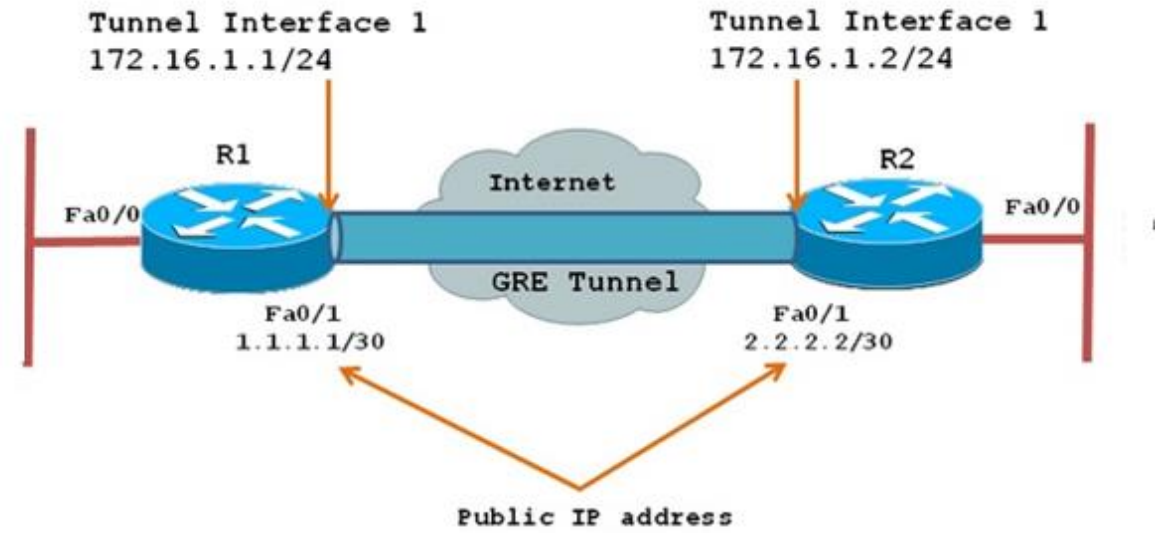
Εν τέλει, το router όχι μόνο πληροί όλες τις προϋποθέσεις που έχουν οριστεί περί δρομολόγησης, αλλά μπορεί και να αποτελείται από λειτουργικά συστήματα οικεία προς το χρήστη (UNIX-like), παρέχοντας, έτσι, τη δυνατότητα πρόσβασης και διαχείρησης μέσω γνωστών και ευρέως υποστηριζόμενων πρωτοκόλλων (SSH). Επιπλέον, η πρόσβαση σε πακέτα ανοιχτού κώδικα, πρακτικά όλων των ευρέως χρησιμοποιημένων λειτουργικών συστημάτων, τα οποία παρέχουν λογισμικό που υλοποιεί αλγορίθμους και πρωτόκολλα δρομολόγησης, προσφέρει σημαντική επεκτασιμότητα στην υπηρεσία, δίνοντας πρόσβαση σε επιπλέον δυνατότητες, όπως η εγκατάσταση επιπλέον υπηρεσιών (πχ. web servers, ftp), η ρύθμιση τοίχων προστασίας (firewalls) και η εκτέλεση πειραματικού ή εκπαιδευτικού λογισμικού του χρήστη πάνω στους κόμβους της τοπολογίας. **Συμπεραίνοντας, οι κόμβοι των ιδιωτικών δικτύων που θα εγκαθίστανται θα υλοποιούνται με routers.**

2.3.2 Υλοποίηση Ζεύξεων και Απόδοση Διευθύνσεων στους Δρομολογητές

Στην ενότητα αυτή, θα συζητηθούν οι τρόποι με τους οποίους μπορούν να αποδοθούν στους δρομολογητές της τοπολογίας οι διευθύνσεις που έχει προσδιορίσει ο χρήστης. Συγκεκριμένα, θα αναφερθούμε στις εικονικές διευθύνσεις των κόμβων του δικτύου πάνω

στις ζεύξεις τους και όχι στη διαχειριστική δημόσια διεύθυνση IP, που θα αποδίδεται στους δρομολογητές για παραμετροποίηση και πρόσβαση εκτός του ιδιωτικού δικτύου. Η μέριμνα στο κομμάτι αυτό είναι κυρίως στην αφαιρετικότητα (abstraction) της υλοποίησης αλλά και στους περιορισμούς τόσο του virtualization όσο και της πιθανής συμβατότητας με τις αντίστοιχες φυσικές συσκευές.

Ο πρώτος περιορισμός, με τον οποίο πρέπει να συμβιβαστεί η υλοποίηση, είναι ότι **δεν πρέπει να απαιτηθούν πολλαπλά interfaces στους δρομολογητές**. Απαίτηση πολλαπλών καρτών δικτύου σε κάποια εικονική μηχανή ενδέχεται, σε ορισμένες περιπτώσεις, να δημιουργήσει θέματα συμβατότητας με λογισμικό φιλοξενίας εικονικών μηχανών που δεν υποστηρίζει αυθαίρετο αριθμό από αυτά. Μεταβλητός αριθμός υποτιθέμενων διεπαφών θα έβλαπτε, επίσης, τη συμβατότητα με τη χρήση της ανάπτυξης αυτής σε συνεργασία με φυσικές συσκευές, καθώς και οι συσκευές αυτές κατασκευάζονται με πεπερασμένο αριθμό καρτών δικτύου. Επιπλέον, δεν υπάρχουν πλεονεκτήματα στο να χρησιμοποιούνται πραγματικά – ή εικονικά – interfaces για εικονικές διευθύνσεις. Όπως προαναφέρθηκε, το δίκτυο που θα δημιουργείται θα εκτείνεται πολλαπλών φυσικών μηχανημάτων, με τις ζεύξεις να υλοποιούνται μέσω του διαδικτύου.



3. Τεχνολογία Tunneling

Για την δημιουργία του overlay δικτύου για την υλοποίηση της εικονικής τοπολογίας πάνω από το διαδίκτυο, θα χρησιμοποιηθούν **τεχνολογίες tunneling**. Σύμφωνα με το πρωτόκολλο σήραγγας που εφαρμόζεται, δημιουργείται ένα **εικονικό tunnel interface**, το οποίο για κάθε σκοπό συμπεριφέρεται σαν μία κανονική διεπαφή, η οποία μπορεί να ορίζει διεύθυνση IP, να συμμετέχει σε πίνακες δρομολόγησης και να αναφέρεται σε κανόνες τοίχους προστασίας (firewall rules). Το tunnel interface δημιουργείται από άκρο σε άκρο, γεφυρώνοντας έτσι τις δημόσιες διευθύνσεις της πηγής και του προορισμού της σήραγγας. Υποθέτοντας ότι δημιουργείται tunnel interface **tun0**, από τον host με δημόσια διεύθυνση **147.102.39.18** και με προορισμό τον host με δημόσια διεύθυνση **178.90.38.12**. Πακέτο IP με διεύθυνση πηγής την **10.0.0.1** και διεύθυνση προορισμού την **10.0.0.2** που στέλνεται στο tunnel interface, θα ενθυλακωθεί (**encapsulated**) εντός πακέτου IP με πηγή 147.102.39.18 και προορισμό 178.90.38.12 και θα δρομολογηθεί εκ νέου. Το ίδιο πακέτο, φτάνοντας στον host με διεύθυνση 178.90.38.12, θα διαπιστωθεί ότι πρόκειται για ενθυλακωμένο πακέτο κάποιου tunneling protocol, θα αποθυλακωθεί και προωθηθεί στο αντίστοιχο tunneling interface, αν υπάρχει.

Κατά συνέπεια, **οι ζεύξεις της τοπολογίας θα υλοποιηθούν με tunnels**. Υπάρχουν αρκετά πρωτόκολλα σήραγγας που χρησιμοποιούνται συχνά, με αυτά που λειτουργούν στο επίπεδο ζεύξης δεδομένων να είναι απρόσιτα στη συγκεκριμένη ανάπτυξη καθώς και επιθυμούμε δρομολόγηση στο επίπεδο δικτύου [2]. Αργότερα στην εργασία, θα συζητηθούν οι μέθοδοι με τις οποίες θα παραμετροποιηθούν τα tunnel αυτά στους δρομολογητές. Εκμεταλλευόμενοι αυτών των μεθόδων και δεδομένου της αρκετά abstract υλοποίησης, η εναλλαγή μεταξύ των πρωτοκόλλων αυτών θα είναι απλό ζήτημα παραμετροποίησης της εφαρμογής. Για απλότητα, στα πλαίσια αυτής της εργασίας, θα χρησιμοποιηθεί το **GRE tunneling**, το οποίο υποστηρίζεται ευρέως από τους δρομολογητές που κυκλοφορούν και είναι εύκολο στη ρύθμιση.

2.3.3 Δρομολόγηση Μεταξύ των Εικονικών Διευθύνσεων

Ανάλογα με τη διάταξη του δικτύου, στη διαδικασία της δρομολόγησης συμμετέχουν κυρίως οι δρομολογητές του δικτύου αλλά και όσοι κανονικοί εξυπηρετητές ενεργοποιήσουν την προώθηση πακέτων στο λειτουργικό τους σύστημα. Στα ιδιωτικά δίκτυα, που είναι και αυτά που μας αφορούν σε αυτή την εργασία, η έμφαση κατά τη σχεδίαση της τοπολογίας του δικτύου δίνεται στη δημιουργία των πινάκων δρομολόγησης. Οι εγγραφές στους πίνακες αυτούς πρέπει να παράγονται έτσι, ώστε η δρομολόγηση να μην αποτυγχάνει όταν υπάρχει ο προορισμός και να υπάρχουν εναλλακτικές διαδρομές σε περίπτωση απώλειας κάποιας ζεύξης ή κάποιου κόμβου.
Η δημιουργία των πινάκων δρομολόγησης για ιδιωτικά δίκτυα μπορεί να γίνει με μερικούς τρόπους, εκμεταλλεύοντας την διαθέσιμη τεχνολογία και τις απαιτήσεις της κάθε περίπτωσης.

- **Στατική Δρομολόγηση (Static Routing)**
  Είναι η διαδικασία δρομολόγησης μέσω της εγκατάστασης στατικών διαδρομών (static routes) στους κόμβους του δικτύου. Στατικές αποκαλούνται οι εγγραφές ενός πίνακα δρομολόγησης που έχουν προέλθει από χειροκίνητη επέμβαση του διαχειριστή του συστήματος. Αυτή η μέθοδος δρομολόγησης είναι μία εύκολη επιλογή για τη γρήγορη ρύθμιση της δρομολόγησης σε μία σχετικά μικρή τοπολογία, αλλά γίνεται πολύ γρήγορα μη-κλιμακώσιμη, καθώς μεγαλώνει ο αριθμός των κόμβων του δικτύου. Η επιλογή των στατικών διαδρομών γίνεται βιώσιμη όταν συμπεριλάβουμε το περιβάλλον του software-defined network**,** όπου και η επιλογή των διαδρομών και η ρύθμιση των στατικών διαδρομών, που ορίζουν τις διαδρομές αυτές, μπορεί να γίνουν δυναμικά σε υψηλό επίπεδο για όλο το γράφο, μέσω λογισμικού.

- **Δυναμική Δρομολόγηση (Dynamic Routing)**
  Αντιμετωπίζοντας τόσο τα θέματα κλιμακωσιμότητας ενός δικτύου, όσο και την απαίτηση για δυναμικές τροποποιήσεις στο δίκτυο, που κατά κανόνα δεν πρέπει να βασίζονται σε ανθρώπινη παρέμβαση, τα περισσότερα δίκτυα εκμεταλλεύονται πρωτόκολλα δρομολόγησης τα οποία κατασκευάζουν αυτόματα τους πίνακες δρομολόγησης στους δρομολογητές του δικτύου. Τα πρωτόκολλα αυτά ανήκουν σε δύο κύριες κατηγορίες:
    - **Interior Gateway Protocols** (IGP): Ανταλλαγή πληροφορίας δρομολόγησης μεταξύ δρομολογητών κοινής περιοχής ή διαχειριστικής οντότητας.

- **Exterior Gateway Protocols** (EGP): Ανταλλαγή πληροφορίας δρομολόγησης μεταξύ δρομολογητών διαφορετικών αυτόνομων συστημάτων (Autonomous Systems, AS).

Στην περίπτωση της ανάπτυξης ενός VPN, που εκτείνεται στο δημόσιο διαδίκτυο, υπάρχουν μερικοί παράγοντες που καθιστούν την εφαρμογή πρωτοκόλλων δρομολόγησης μη επιθυμητή.

- Τα πρωτόκολλα δρομολόγησης την εξερεύνηση γειτονικών δρομολογητών και την εγκατάσταση σχέσεων με αυτούς για ανταλλαγή πληροφοριών. Στην περίπτωσή της εργασίας αυτής, όμως, γνωρίζουμε την ύπαρξη όλως των υπόλοιπων κόμβων του ιδιωτικού δικτύου καθώς και όλες τις σχέσεις γειτνίασης μεταξύ τους.
- Η ανθεκτικότητα των πρωτοκόλλων αυτών σε σφάλματα των ζεύξεων και αλλαγές στην τοπολογία του δικτύου μέσω συνεχών ενημερώσεων δεν είναι χρήσιμη στα πλαίσια της εργασίας διότι, όπως προαναφέρθηκε, οι ζεύξεις είναι σήραγγες από σημείο σε σημείο και η δρομολόγηση που τις συνδέει γίνεται στο διαδίκτυο.
- Ενδέχεται να υπάρχουν περιπτώσεις όπου οι τεχνικές broadcast ή multicast που χρησιμοποιούνται από τα πρωτόκολλα αυτά να μην είναι συμβατές με το πρωτόκολλο σήραγγας που θα χρησιμοποιηθεί για την εγκατάσταση των ζεύξεων στο επίπεδο δικτύου [3].
- Σε κάποια use cases, ο χρήστης μπορεί να επιθυμεί το ιδιωτικό του δίκτυο να αρχικοποιείται με κάποια βασική συνδεσιμότητα μεταξύ των κόμβων, ή και με τη χρήση κάποιου συγκεκριμένου αλγορίθμου, ώστε μετά να μπορεί να παραμετροποιήσει το πώς τελικά θα γίνεται η δρομολόγηση χειροκίνητα.

Η υλοποίηση της επιθυμητής συνδεσιμότητας της τοπολογίας στο VPN μπορεί να επιτευχθεί μέσω στατικής δρομολόγησης. Δεδομένου αυτού και των παραπάνω, η υλοποίηση θα ακολουθήσει αυτή τη μέθοδο, αφήνοντας παράλληλα τη δυνατότητα στους χρήστες να εφαρμόσουν κάποια άλλη, μέσω της απομακρυσμένης πρόσβασης που θα έχουν στους εικονικούς δρομολογητές.

2.4 Μοντελοποίηση Τοπολογίας Ιδιωτικού Δικτύου

Έχοντας αναλύσει τις συσκευές και τις έννοιες, οι οποίες θα αποτελέσουν τα συστατικά των ιδιωτικών εικονικών δικτύων (VPN), που θα κατασκευάζονται στα πλαίσια της εφαρμογής, το επόμενο σημαντικό βήμα στο υπόβαθρο της συγκεκριμένης υλοποίησης είναι η μοντελοποίηση των δικτύων αυτών. Συμβαδίζοντας με τον τομέα των τηλεπικοινωνιών, ο στόχος της ενότητας αυτής δεν είναι το να επινοηθεί μία γενική μορφοποίηση κειμένου που να απεικονίζει επαρκώς τις ιδιότητες και τα χαρακτηριστικά κάποιας δικτυακής διάταξης εξυπηρετητών. Ο σκοπός είναι το να υλοποιηθεί η απεικόνιση αυτή, κάνοντας χρήση των τυποποιημένων μεθόδων και κανόνων που συμφωνούν με τις αντίστοιχες εφαρμογές της βιομηχανίας αυτής. Στην ενότητα αυτή, λοιπόν, θα μελετηθούν κάποιες από τις δομές απεικόνισης τέτοιων που χρησιμοποιούνται ευρέως από παρόμοιες εφαρμογές.

2.4.1 Το YANG ως Γλώσσα Μοντελοποίησης

Η YANG είναι μία γλώσσα μοντελοποίησης μορφής δεδομένων, που αναπτύχθηκε αρχικά για την περιγραφή παραμετροποιήσεων του πρωτοκόλλου NETCONF [1]. Επεκτάθηκε αργότερα με το RFC 8345, με την πρόταση του προτύπου αυτού, για περιγραφή τοπολογιών δικτύου, εισάγοντας έννοιες ζεύξεων και τερματικών κόμβων [18]. Η γλώσσα αυτή δεν εξαρτάται από κάποιο πρωτόκολλο και μπορεί να κωδικοποιηθεί επίσης σε XML ή JSON, κάνοντάς το, έτσι, παράλληλα φιλικό προς τις τεχνολογίες του διαδικτύου (web technologies). Παρακάτω φαίνεται το παράδειγμα ενός μοντέλου YANG, το οποίο απεικονίζει μία απλοποιημένη λίστα των δικτυακών διεπαφών κάποιας συσκευής.

```
container interfaces{
    ...
    list interfaces {
        key “name”;
        description
            “The list of configured
            interfaces on the device.“;
        ...
    }
    list interfaces-state {
        config false;
        key “name”;
        description
            “Data nodes for the operational
            state of interfaces.”
        ...
    }
}
```

Αν και το πρότυπο αυτό θα μπορούσε να επεκταθεί στα πλαίσια της εφαρμογής, ώστε να περιέχει και τη λίστα των δεδομένων προς απεικόνιση, δεν είναι αυτή η κύρια λειτουργία του. Ο σκοπός του YANG είναι κυρίως η περιγραφή δρομολογητών και υπηρεσιών και όχι οι περιγραφή υποδομών στο cloud.

2.4.2 Ενορχήστρωση στο Cloud με TOSCA

Αισθητά πιο κοντά στον πυρήνα της εφαρμογή μας βρίσκεται το μοντέλο TOSCA. Το μοντέλο αυτό απεικονίζεται κυρίως σε μορφή κειμένου YAML και ιστορικά έχει συσχετιστεί με εφαρμογές ανάπτυξης υπολογιστικών πόρων και δικτύων στο cloud [19]. Η κύρια ιδέα του μοντέλου αυτού είναι η περιγραφή τύπων με τη μορφή templates, την εφαρμογή των τύπων αυτών σε κόμβους και την περιγραφή των σχέσεων μεταξύ των κόμβων αυτών. Σε αντίθεση με το YANG, στο οποίο instances των περιγραφόμενων μοντέλων είναι αυστηρά ορισμένα και υλοποιούν τα ίδια τις προδιαγραφές του μοντέλου, το TOSCA λειτουργεί περισσότερο με την έννοια της κληρονομικότητας του βασικού τύπου, ώστε τα instances να επεκτείνουν τον τύπο αυτό, περιγράφοντας δικές τους ιδιότητες και λειτουργικότητες. Το παράδειγμα ενός τέτοιου τύπου, ο οποίος περιγράφει έναν απλό

εικονικό δρομολογητή, φαίνεται παρακάτω. Η σύνταξη της τοπολογίας σε YAML, με αυτόν τον τρόπο, επιτρέπει, επίσης, την ένα-προς-ένα μετατροπή της από και προς μορφή JSON.

```yaml
topology_template:
  node_templates:
    router:
      type: tosca.nodes.Router
      capabilities:
        host:
          num_cpus: 1
          disk_size: 10 GB
          mem_size: 256 MB
        os:
          properties:
            architecture: x86_64
```

Όπως φαίνεται, ορίζεται το template με όνομα “router”, το οποίο μπορεί, επίσης, να είναι επέκταση κάποιου άλλου template και ορίζει κάποιες χρήσιμες προδιαγραφές για όσους κόμβους το υλοποιήσουν, όπως στην περίπτωση του cloud του αποθηκευτικού χώρου και της μνήμης του εικονικού μηχανήματος. Οπότε τώρα, instances του τύπου αυτού μπορούν να προστεθούν στη μοντελοποίηση της τοπολογίας, ορίζοντας παράλληλα δικές τους παραμέτρους.

```yaml
topology_template:
  node_templates:
    router:
      type: tosca.nodes.Router
      ...
  router2:
    type: tosca.nodes.Router
    properties:
      ...
    ...
```

Επεκτείνοντας την ιδέα αυτή, η προτεινόμενη μοντελοποίηση των ιδιωτικών δικτύων, που θα σχεδιάζονται στα πλαίσια της εργασίας αυτής, φαίνεται παρακάτω. Στο παράδειγμα αυτό, φαίνεται μία απλή τοπολογία, η οποία αποτελείται από δύο δρομολογητές και μία ζεύξη μεταξύ τους.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_0
description: Automatically generated topology
topology_template:
  node_templates:
    router:
      type: tosca.nodes.Router
      capabilities:
        host:
          num_cpus: 1
```

```
        disk_size: 10 GB
        mem_size: 256 MB
      os:
        properties:
          architecture: x86_64
router2:
  type: tosca.nodes.Router
  properties:
    router_id: 2
    hostname: router2
  tunnels:
    - addr: 10.0.0.3
      route: 10.0.0.2
      remote_id: 1
router1:
  type: tosca.nodes.Router
  properties:
    router_id: 1
    hostname: router1
  tunnels:
    - addr: 10.0.0.2
      route: 10.0.0.3
      remote_id: 2
```

Δίνεται, επίσης, μία τυπική επεξήγηση των κόμβων που έχουν οριστεί:

- **properties.router_id**: Μοναδικός κωδικός, που αντιστοιχεί σε έναν κόμβο της τοπολογίας. Χρησιμοποιείται για την εγκατάσταση ζεύξεων από σημείο σε σημείο.
- **properties.hostname**: Το hostname της συσκευής.
- **tunnels[i].addr**: Η διεύθυνση της συσκευής στην i-οστή ζεύξη.
- **tunnels[i].remote_id**: Το router_id της συσκευής στο άλλο άκρο της i-οστής ζεύξης.
- **tunnels[i].route**: Η διεύθυνση του δρομολογητή στο άλλο άκρο της i-οστής ζεύξης, πάνω στη ζεύξη αυτή.

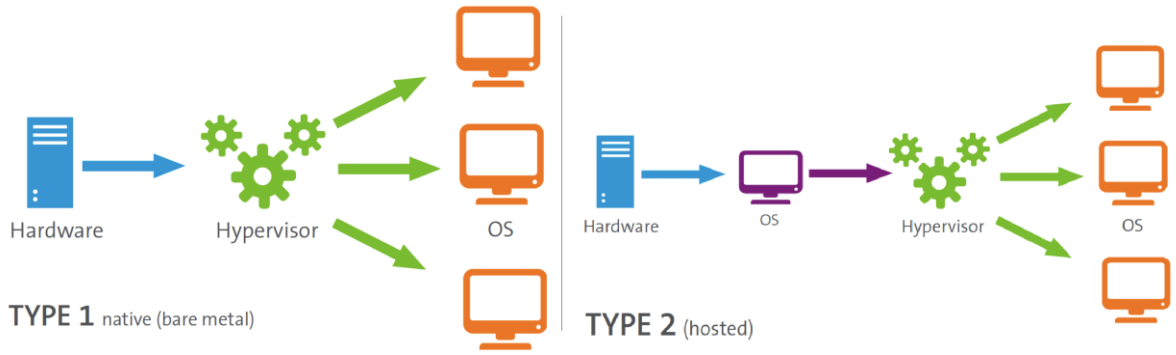
Κεφάλαιο 3

Ανάπτυξη του Ιδιωτικού Δικτύου στους Φυσικούς Εξυπηρετητές

Όπως έχει αναφερθεί, η υποδομή του ιδιωτικού δικτύου, που θα δημιουργείται από την εφαρμογή, θα αποτελείται από δρομολογητές. Αν και θα συζητηθεί η συμβατότητα του λογισμικού που θα αναπτυχθεί με τους φυσικούς δρομολογητές διαφόρων κατασκευαστών, η ανάπτυξη, στα πλαίσια της εργασίας, θα βασιστεί σε εικονικούς δρομολογητές. Η χρήση εικονικών δρομολογητών συμβαδίζει με την κύρια ιδέα της εργασίας, ως προς την άρση των φυσικών περιορισμών, κατά τον πειραματισμό με πολύπλοκες και μεγάλες σε μέγεθος τοπολογίες. Στο κεφάλαιο αυτό, θα γίνει η απαραίτητη έρευνα σχετικά με τις τεχνολογίες και τις μεθόδους που θα χρησιμοποιηθούν για τη φιλοξενία (**hosting**) των εικονικών δρομολογητών στο διαθέσιμο υλικό.

3.1 Hypervisors Τύπου 1 και Τύπου 2

Ένας hypervisor, επίσης αποκαλούμενος και ελεγκτής εικονικών μηχανών (virtual machine monitor, **VMM**), είναι ένας από τους κύριους τρόπους εικονικοποίησης (virtualization) ενός υπολογιστικού συστήματος. Τυπικά, ένας hypervisor καταλαμβάνει το φιλοξενούν φυσικό μηχάνημα (**host**), έτσι ώστε ο διαχειριστής του συστήματος να χρησιμοποιήσει τις δυνατότητες του για να μοιράζει τους διαθέσιμους πόρους, όπως τους επεξεργαστές (CPU) ή τη μνήμη (RAM), μεταξύ εικονικών μηχανών (**guests**). Κάθε guest VM μπορεί να εκτελεί και διαφορετικό λειτουργικό σύστημα, μέσω των διάφορων τεχνικών εικονικοποίησης, αλλά και να λειτουργεί με φαινομενικά τους δικούς του πόρους σε ένα πλήρως απομονωμένο περιβάλλον. Για να επιτευχθεί αυτή η απομόνωση (isolation), ο hypervisor πρέπει μιμηθεί (emulate) το υλικό ενός φυσικού μηχανήματος. Η διαδικασία αυτή είναι από τη φύση της μη αποδοτική και είναι ο κύριος παράγοντας, που επηρεάζει την απόδοση των εικονικών μηχανών. Για να ανακουφιστεί η απόδοση κατά την ακριβή προσομοίωση του υλικού, οι hypervisors συνήθως παρέχουν ειδικούς drivers, εκμεταλλευόμενοι του paravirtualization. Οι drivers αυτοί, αποκαλούμενοι paravirtualized drivers, απεικονίζονται στα guest VM σαν πραγματικές φυσικές συσκευές με απόδοση βελτιστοποιημένη από τον hypervisor. Εάν και η ταχύτητα και απόδοση των εικονικών μηχανών του ιδιωτικού δικτύου δεν είναι από ορισμού μέριμνα της εργασίας αυτής, οφείλουμε τουλάχιστον να εξερευνήσουμε τις επιλογές που υπάρχουν ως προς την εικονικοποίηση δικτυακών συσκευών. Συγκεκριμένα, οι hypervisors υπάγονται σε δύο κύριες κατηγορίες:



4. Σύγκριση Τύπων Hypervisors

- **Type 1**
  Οι hypervisors πρώτου τύπου έχουν τη μορφή λειτουργικού συστήματος, που εγκαθίσταται κατευθείαν στο υλικό και συνήθως αποκαλείται και “Bare Metal” hypervisor. Κάθε άλλο guest λειτουργικό σύστημα εκτελείται πάνω στον hypervisor αυτόν, με ένα από αυτά συνήθως να ορίζεται ως διαχειριστικό με προνομιούχα πρόσβαση στον hypervisor [4].

- **Type 2**
  Ένας hypervisor τύπου δύο είναι πρακτικά λογισμικό που εκτελείται πάνω σε κάποιο υπάρχον λειτουργικό σύστημα, με τα guest λειτουργικά συστήματα να εκτελούνται ως διεργασίες (processes). Αυτού του είδους η εικονικοποίηση συχνά θεωρείται πιο αργή, λόγω του επιπλέον επιπέδου επικοινωνίας του hypervisor με το hardware και χρησιμοποιείται συνήθως από προγράμματα που χρησιμοποιούνται για τη χρήση εικονικών μηχανών, εντός κάποιου περιβάλλοντος εργασίας χρηστών (desktop). Τέτοιου είδους προγράμματα είναι το VirtualBox και το QEMU [5].

3.2 Επιλογή Hypervisor

Στην ενότητα αυτή, θα εξηγηθεί σύντομα η επιλογή του κατάλληλου hypervisor για την φιλοξενία των εικονικών μηχανών των δρομολογητών. Χωρίς ιδιαίτερη μέριμνα για πρόσβαση στον hypervisor από το περιβάλλον του χρήστη και δεδομένου του ότι τα φυσικά μηχανήματα που θα φιλοξενούν τους δρομολογητές κατά κανόνα θα είναι εξυπηρετητές (servers) και όχι τερματικά χρηστών, μπορούμε να παραλείψουμε τους περισσότερους από τους Type 2 hypervisors. Αποφεύγοντας εξαντλητική αναζήτηση, θα επικεντρωθούμε στις χαρακτηριστικές επιλογές κοινών hypervisor που μας παρουσιάζονται.

- **Microsoft Hyper-V**
  Hypervisor, που περιέχεται μαζί με κάθε έκδοση των λειτουργικών συστημάτων Windows από τα “Windows Server 2008”, “Windows 8” και ύστερα [6]. Αν και συνήθως χρησιμοποιείται από το περιβάλλον χρήστη (user space), στην πραγματικότητα είναι type 1 hypervisor. Οι δυνατότητές του ανταγωνίζονται των υπολοίπων, με υποστήριξη για nested virtualization και αλλαγή μεγέθους εικονικών αποθηκευτικών συσκευών με την εικονική μηχανή ενεργή. Όμως, η αναφορά του σε αυτή την ενότητα είναι κυρίως τιμητική, καθώς και η εξάρτηση του από το λειτουργικό σύστημα των Windows τον καθιστούν μη ιδανική επιλογή, τόσο εξαιτίας της πιο περιορισμένης υποστήριξής του, λόγω κόστους, αλλά και της ιστορικά πιο δύσβατης απομακρυσμένης διαχείρισης των συστημάτων αυτών υπό κανονικών συνθηκών.

- **QEMU με KVM**
  Το QEMU είναι ο μόνος Type 2 hypervisor που θα αναφερθεί σε αυτή την ενότητα, λόγω της οικειότητάς του με το περιβάλλον των Linux και της ευελιξίας του ως προς την χρήση από τη γραμμή εντολών. Αν και ικανός hypervisor, η απουσία οποιουδήποτε προγραμματιστικού interface σημαίνει ότι κάθε διεπαφή με αυτόν θα απαιτούσε ειδική ανάπτυξη από τη πλευρά του λογισμικού. Ο κύριος λόγος, όμως, για τον οποίον το QEMU δεν θα αποτελέσει την τελική επιλογή είναι ότι, ως πρόγραμμα του user space, θα απαιτούσε από τους πόρους που θα συμμετέχουν

στην εφαρμογή είτε να εκτελούν κάποιο λειτουργικό σύστημα που να υποστηρίζει το QEMU κατευθείαν πάνω στο hardware ή να αντιμετωπίσουν πιθανά θέματα συμβατότητας, λόγω nested virtualization ή λόγω της εικονικοποίησης των δικτυακών driver.

- **VMware ESXi**
  Ένας ακόμα "Bare Metal" hypervisor, που όμως, σε αντίθεση με το Hyper-V της Microsoft, εγκαθίσταται κατευθείαν στο υλικό με το ρόλο λειτουργικού συστήματος. Σημαντικό είναι το χαρακτηριστικό του συγκεκριμένου hypervisor, της χρήσης ενός εικονικού switch Cisco Nexus 1000v [7] για τη δικτύωση μεταξύ των ESXi hosts και των guest λειτουργικών συστημάτων, παρέχοντάς τους τη δυνατότητα να μοιράζονται ένα φυσικό interface.

Η VMware με το ESXi ηγείται στην αγορά των server hypervisor [8]. Ως συνέπεια, παρέχονται πολλά εργαλεία, τόσο από την ίδια τη VMware όσο και third party, τα οποία έχουν τη δυνατότητα να παρέχουν σημαντικά θεμέλια και δυνατότητες στα πλαίσια της εργασίας. Η ευρέα χρήση και υποστήριξή του, καθώς και το γεγονός ότι χρησιμοποιείται ήδη από το Εργαστήριο Δικτύων Υπολογιστών της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π., καθιστούν ασφαλή την επιλογή του ESXi για την ανάπτυξη αυτής της εφαρμογής.

3.3 Ανάπτυξη και Αυτοματοποίηση στο ESXi

Έχοντας επιλέξει την πλατφόρμα ESXi της VMware για την φιλοξενία των εικονικών μηχανών των δρομολογητών, στην ενότητα αυτή, θα περιγραφεί η διεπαφή προγραμματισμού εφαρμογών (API) του λειτουργικού συστήματος του hypervisor. Ο hypervisor αυτός μπορεί να συνοδεύεται από λογισμικό, που να επεκτείνει το υπάρχον API ή να προσφέρει ένα καινούργιο. Εκμεταλλευόμενοι των τεχνολογιών και το πρωτοκόλλων του διαδικτύου, θα προτιμούσαμε ο ESXi host να είναι προσιτός και παραμετροποιήσιμος μέσω κάποιου HTTP API. Αν και υπάρχει η δυνατότητα για την επέκταση του ESXi, ώστε να υποστηρίζει πρόσβαση μέσω HTTP μέσω επιπρόσθετου λογισμικού, ο σκοπός της εργασίας είναι η ανάπτυξη abstract λογισμικού, που θα είναι συμβατό με το απαιτούμενο hardware και που θα μπορεί να επεκταθεί με την προσθήκη επιπλέον hardware. Το λειτουργικό σύστημα δεν είναι βασισμένο στο UNIX, εκτελείται πάνω από το "vmkernel" της VMware. Όμως, παρά την απουσία του Linux, το περιβάλλον του περιέχει το τερματικό bash και τα βασικά utilities του UNIX μέσω του λογισμικού "BusyBox" [9]. Μέσω του διαχειριστικού περιβάλλοντος του ESXi host, μπορεί να ενεργοποιηθεί η **πρόσβαση μέσω SSH**. Δεδομένου του πόσο διαδεδομένο και ευρέως υποστηριζόμενο είναι το πρωτόκολλο αυτό, καθίσταται ασφαλής επιλογή για την απομακρυσμένη διαχείριση των ESXi host μέσω λογισμικού.

3.3.1 Εντολές του ESXi Shell

Όπως αναφέρθηκε, η διαχείριση ενός συστήματος ESXi γίνεται κυρίως μέσω της γραμμής εντολών. Για διαχείριση από άνθρωπο, παρέχονται περισσότερα εργαλεία από τη VMware, όπως το "VMware vSphere Client" για χρήση από desktop και το "ESXUI" για πρόσβαση από web browser. Προγραμματιστικά, όμως, η πρόσβαση μπορεί ασφαλώς να

πραγματοποιηθεί μέσω του shell του συστήματος. Οι διάφορες λειτουργίες του καλύπτονται από την μονολιθική εντολή **vim-cmd**, η οποία τις ομαδοποιεί κάτω από τις διάφορες εντολές της. Συγκεκριμένα, η δόμηση των εντολών γίνεται με μία δενδρική δομή, που μιμείται οργάνωση σε φακέλους. Για να ανακτήσουμε τις πιθανές εντολές κάποιας ομάδας, χρησιμοποιούμε την εντολή **help**, όπως φαίνεται στα παρακάτω παραδείγματα.

```
$ vim-cmd help
Commands available under /:
hbrsvc/       internalsvc/  solo/         vmsvc/
hostsvc/      proxysvc/     vimsvc/       help
```

Για την ανάκτηση των εντολών της ομάδας **vmsvc** αντίστοιχα.

```
$ vim-cmd vmsvc help
Commands available under vmsvc/:
acquiremksticket                get.snapshotinfo
acquireticket                   get.spaceNeededForConsolidation
connect                         get.summary
convert.toTemplate              get.tasklist
convert.toVm                    getallvms
createdummyvm                   gethostconstraints
destroy                         login
device.connection               logout
device.connusbdev               message
device.ctlradd                  power.getstate
device.ctlrremove               power.hibernate
device.disconnusbdev            power.off
device.diskadd                  power.on
device.diskaddexisting          power.reboot
device.diskremove               power.reset
device.getdevices               power.shutdown
device.toolsSyncSet             power.suspend
device.vmiadd                   power.suspendResume
device.vmiremove                queryftcompat
devices.createnic               reload
get.capability                  setscreenres
get.config                      snapshot.create
get.config.cpuidmask            snapshot.dumpoption
get.configoption                snapshot.get
get.datastores                  snapshot.remove
get.disabledmethods             snapshot.removeall
get.environment                 snapshot.revert
get.filelayout                  snapshot.setoption
get.filelayoutex                tools.cancelinstall
get.guest                       tools.install
get.guestheartbeatStatus        tools.upgrade
get.managedentitystatus         unregister
get.networks                    upgrade
get.runtime
```

Καθοδηγούμενοι, έτσι, από τα μηνύματα βοηθείας και συμβουλευόμενοι του documentation της VMware στο διαδίκτυο [20], συμπεραίνουμε τις εντολές που ενδεχομένως θα χρειαστούν στα πλαίσια της ανάπτυξης αυτής της εργασίας.

vim-cmd vmsvc/getallvms	Εκτυπώνει ως έξοδο έναν πίνακα με τις εικονικές μηχανές που έχουν εγγραφεί αυτή τη στιγμή στον ESXi host.
vim-cmd solo/registervm {vmx file}	Εγγραφή εικονικής μηχανής στον ESXi host. Η διαδικασία αυτή αποδίδει μοναδικό ID

	στην εικονική μηχανή και παράγει MAC διευθύνσεις για όσα interface της χρειάζονται. Εκτυπώνει στην έξοδο το ID που αποδόθηκε.
vim-cmd vmsvc/power.on {vm id}	Ενεργοποίηση εικονικής μηχανής.
vim-cmd vmsvc/power.off {vm id}	Απενεργοποίηση εικονικής μηχανής.
vim-cmd vmsvc/power.getstate {vm id}	Εκτυπώνει την κατάσταση μιας εικονικής μηχανής.
vim-cmd vmsvc/unregister {vm id}	Αφαιρεί μία εικονική μηχανή από το ESXi
vim-cmd vmsvc/get.networks {vm id}	Εκτυπώνει τους δικτυακούς adapters μιας εικονικής μηχανής.

3.3.2 Κλωνοποίηση Εικονικών Μηχανών

Όπως θα δούμε παρακάτω στην εργασία, οι εικονικές μηχανές των δρομολογητών θα προέρχονται από υπάρχουσες εικονικές μηχανές, οι οποίες θα κλωνοποιούνται (clone). Ο κύριος – αναμεταξύ άλλων – λόγος είναι ότι οι δρομολογητές θα χρειαστεί, κατά την εκκίνησή τους, να περιέχουν ορισμένες προϋπάρχουσες ρυθμίσεις, απαραίτητες, ώστε το back-end της εφαρμογής να έχει απομακρυσμένη πρόσβαση σε αυτούς για την παραμετροποίηση τους. Το cloning στο περιβάλλον του ESXi είναι λίγο πιο περίπλοκο από απλή αντιγραφή των αρχείων του, οπότε, στην ενότητα αυτή, θα αναφερθούν οι εντολές με τις οποίες επιτυγχάνεται.

Μια απενεργοποιημένη εικονική μηχανή αποτελείται κυρίως από δύο αρχεία:

- **VMX**: Το αρχείο που περιγράφει το VM, όπως αναφέρθηκε στην προηγούμενη ενότητα.
- **VMDK**: Virtual Machine Disk. Το αρχείο αυτό είναι ο εικονικός δίσκος μίας εικονικής μηχανής. Μια ιδιότητα του αρχείου αυτού είναι ότι μπορεί να δημιουργηθεί με έναν από δύο τρόπους:
    - **Thick provisioning**: Το αρχείο του εικονικού δίσκου καταλαμβάνει τόσο χώρο όσο και η εικονική χωρητικότητα (capacity) του δίσκου.
    - **Thin provisioning**: Το αρχείο του εικονικού δίσκου καταλαμβάνει τόσο χώρο όσο και ο χώρος που καταλαμβάνει το λειτουργικό σύστημα της εικονικής μηχανής.

Για λόγους ασφαλείας, η χωρητικότητα (capacity) των δίσκων των εικονικών δρομολογητών θα ρυθμίζεται σε τιμή της τάξης των 2GB έως 10GB, με το λειτουργικό σύστημα, όσων έχω εξερευνήσει στα πλαίσια της εργασίας, κατά κανόνα να μην ξεπερνάει τα 500MB σε μέγεθος.

Cloning σε thick provisioning γίνεται ως εξής:

```
$ vmkfstools -i disk.vmdk new_disk.vmdk
```

Και cloning σε thin provisioning γίνεται ως εξής:

```
$ vmkfstools -d thin -i disk.vmdk new_disk.vmdk
```

Συγκεκριμένα, εκτελέστηκαν μετρήσεις χρησιμοποιώντας εικονικό δίσκο με capacity 10GB και thin χώρο 268MB, τόσο με χρήση της εντολής **vmkfstools** αλλά και με απλή αντιγραφή με χρήση της εντολής **cp** (copy).

Source \ Destination	Thick Clone	Thin Clone	Copy
Thick	3.88 sec	3.88 sec	83.61 sec
Thin	3.81 sec	3.42 sec	95.20 sec

Τόσο για την οικονομία και την ευελιξία στον αποθηκευτικό χώρο των ESXi host, όσο και με μέριμνα την ταχύτητα της κλωνοποίησης, **οι δίσκοι των εικονικών μηχανών θα υλοποιηθούν με thin provisioning**. Κάνοντας, όμως, απλή αντιγραφή του αρχείου vmdk με χρήση της εντολής **cp**, το νέο αρχείο που θα δημιουργηθεί θα υιοθετήσει thick provisioning, ακόμα και αν το πηγαίο αρχείο είναι σε thin provisioning. **Οπότε θα αφιερωθεί προσοχή στο να διατηρείται το βασικό VM – από το οποίο θα πηγάζουν οι κλώνοι – σε thin provisioning, καθώς να γίνεται το cloning επίσης πάντα σε thin provisioning**.

Τέλος, αφού πραγματοποιηθεί η αντιγραφή του αρχείου **vmx**, πρέπει να αφαιρεθούν οι παρακάτω γραμμές από αυτό, εάν υπάρχουν. Οι παράμετροι αυτοί γράφονται στο αρχείο αυτό από το ESXi, όταν η εικονική μηχανή που είναι συσχετισμένη με το αρχείο αυτό ενεργοποιείται για πρώτη φορά. Κάποιες από τιμές αυτές απλώς δεν επιθυμούμε να κληρονομηθούν από τις καινούργιες εικονικές μηχανές, όπως οι παραγόμενες MAC διευθύνσεις των διεπαφών του. Άλλες τιμές ταυτοποιούν την εικονική μηχανή του αρχείου αυτού και ενδέχεται, εάν παραμείνουν σε εικονική μηχανή που επιχειρήσει να γίνει register μέσω της γραμμής εντολών από καινούργιο φάκελο, να κολλήσουν το ESXi σε κατάσταση ερωτήσεως προς το χρήστη για το αν η συγκεκριμένη εικονική μηχανή μετακινήθηκε ή αντιγράφηκε. Σε κάθε περίπτωση, η αφαίρεση των γραμμών αυτών αποτρέπει όλες τις σχετικές περιπλοκές κατά την εγγραφή της κλωνοποιημένης εικονικής μηχανής.

```
sched.swap.derivedName = "..."
uuid.bios = "56 4d 23 c7 3e ed 28 d2-e7 fd 19 60 a0 ed 5c 60"
uuid.location = "56 4d 23 c7 3e ed 28 d2-e7 fd 19 60 a0 ed 5c 60"
ethernet0.generatedAddress = "00:0c:29:ed:5c:60"
ethernet0.generatedAddressOffset = "0"
vmci0.id = "-1595057056"
```

3.4 Εικονικές Μηχανές Δρομολογητών

Στην ενότητα αυτή, θα αναφερθούν τα είδη λογισμικού δρομολογητών, που παρουσιάζουν ενδιαφέρον στα πλαίσια της εργασίας. Για την ικανοποίηση των απαιτήσεων της συγκεκριμένης ανάπτυξης, θα μπορούσαν να χρησιμοποιηθούν τόσο εικονικοποιήσεις λειτουργικών συστημάτων από φυσικούς δρομολογητές όσο και λειτουργικά συστήματα γενικού σκοπού με το κατάλληλο λογισμικό δρομολόγησης. Όπως έχει ήδη αναφερθεί, στόχος της υλοποίησης αποτελεί το να είναι αρκετά abstract και να παρέχει τις κατάλληλες διεπαφές παραμετροποίησης, ώστε να είναι δυνατή η επέκταση για τη λειτουργιά με πολλαπλού είδους δρομολογητές. Δεδομένου αυτού, αναφέρονται συνοπτικά κάποια τέτοια

είδη εικονικών δρομολογητών, που είναι άμεσα σχετικά με την υλοποίηση της εργασίας. Αν και στα πλαίσια της ανάπτυξης αυτής θα χρησιμοποιηθεί χαρακτηριστικά ένας από αυτούς, η κύρια ιδέα είναι ότι κάποιος διαχειριστής θα έχει τη δυνατότητα να επεκτείνει την εφαρμογή, δίνοντας τη δυνατότητα στους χρήστες να επιλέξουν το επιθυμητό είδος δρομολογητή μέσω παραμετροποίησης.

- **FreeBSD με Quagga**
  Το FreeBSD είναι λειτουργικό σύστημα ανοιχτού κώδικα, βασισμένο στο UNIX, η ανάπτυξη του οποίου εστιάζει στην επίδοση, την ασφάλεια και την σταθερότητα για δικτυακές εφαρμογής. Η μετατροπή ενός τέτοιου συστήματος σε δρομολογητή μπορεί να γίνει με εγκατάσταση του Quagga. Η συλλογή λογισμικού, που παρέχεται με την εγκατάσταση αυτή, προσφέρει εργαλεία παραμετροποίησης της συσκευής με υποστήριξη ρυθμίσεων kernel, RIP, OSPF, BGP και IS-IS τόσο για IPv4 όσο και για IPv6.

- **HPE VSR1000**
  Ο δρομολογητής αυτός είναι προϊόν της Hewlett Packard (HP) και παρέχει virtualized εφαρμογές παρόμοιες με αυτές των αντίστοιχων φυσικών δρομολογητών της. Όντας εμπορικό προϊόν, ο εικονικός δρομολογητής αυτός περιέχει σχεδόν όλες τις δυνατότητες τόσο από την πλευρά της δρομολόγησης όσο και από την πλευρά ασφάλειας, με υποστήριξη για IPSec tunneling και προ-εγκατεστημένο τείχος προστασίας. Είναι άξιος αναφοράς για τη συγκεκριμένη εργασία, καθώς και η ίδια η HP προτείνει την εγκατάστασή του με τον VMware ESXi hypervisor [1].

- **Cisco CSR 1000V**
  Ο εικονικός δρομολογητής αυτός αναπτύσσεται από τη Cisco με στόχο την επιχειρησιακή (enterprise) ανάπτυξη στο cloud. Είναι ειδικευμένος για χρήση ως gateway σε VPN [12], υποστηρίζοντας όλα τα σχετικά πρωτόκολλα, και με τη Cisco να ηγείται του τομέα των τηλεπικοινωνιών, κατέχοντας περίπου 60% του μεριδίου της αγοράς [11], συμβατότητα με αυτόν θα επέτρεπε την εφαρμογή σε πολλαπλά επιχειρησιακά σενάρια χρήσης για cloud orchestration.

- **VyOS**
  Μοιράζοντας πολλές ομοιότητες με το Quagga που προαναφέρθηκε, το VyOS αποτελείται ουσιαστικά από ένα πακέτο λογισμικού, αναπτυσσόμενο πάνω στο λειτουργικό των Debian Linux. Είναι ένας πλήρως εξοπλισμένος δρομολογητής, συμβατός με hardware και με πολύ μικρές απαιτήσεις σε χώρο και μνήμη, που τον καθιστούν πολύ καλή επιλογή τόσο για cloud όσο και για δικτυακές συσκευές μικρών δυνατοτήτων.

3.5 Παραδοχές και Περιορισμοί

Η πρόσβαση, η χρήση και η παραμετροποίηση των ESXi hosts, όπως θα προγραμματιστεί κατά την ανάπτυξη του λογισμικού στα επόμενα κεφάλαια, καθώς και η εγκατάσταση εικονικών μηχανών σε αυτούς, θα γίνουν θεωρώντας δεδομένες ορισμένες παραδοχές που θα γίνουν σε αυτή την ενότητα. Κάποιες από τις παραδοχές αυτές υπάρχουν διότι, όπως έχει αναφερθεί στην περιγραφή των δυνατοτήτων της εφαρμογής, η υλοποίηση παραμένει

abstract, αφήνοντας, έτσι, την επίλυση των περιορισμών αυτών στις συνθήκες του κάθε σεναρίου χρήσης (use case).

- **Παρουσία των προ-παραμετροποιημένων εικονικών μηχανών.**
  Αργότερα στην εργασία, θα περιγραφεί η παραμετροποίηση των εικονικών μηχανών με τις ρυθμίσεις που είναι απαραίτητες κατά την εκκίνησή τους (**Zero-Day Configuration**). Το λογισμικό του controller θα θεωρεί ότι τα αρχεία αυτά, από τα οποία θα παράγει τους κλώνους των εικονικών μηχανών των δρομολογητών, θα υπάρχουν ήδη στον αποθηκευτικό χώρο του host. Στα πλαίσια αυτής της εργασίας, τα αρχεία αυτά θα τοποθετούνται στους hosts χειροκίνητα, όμως, για κάποιο σενάριο χρήσης υπάρχει η δυνατότητα επέκτασης της εφαρμογής με κάποιο σύστημα διαχείρισης εκδόσεων (versioning) των αρχείων αυτών και την δυναμική μεταφορά τους στους hosts από τον controller μέσω FTP, όταν δεν υπάρχουν.

- **Πρόσβαση στο χρήστη root μέσω SSH**
  Το λογισμικό αυτοματοποίησης θα επιχειρεί πρόσβαση μέσω SSH στον χρήστη root του ESXi host. Για να είναι δυνατό αυτό, η πρόσβαση μέσω SSH πρέπει να ενεργοποιηθεί ρητά από τη διαχειριστική διεπαφή του ESXi. Επίσης, για αποφυγή χειρισμού πολλαπλών κωδικών από τον controller και για να μην χρειαστεί να εισαχθούν οι κωδικοί αυτοί στο source control της εργασίας, ο controller, στα πλαίσια της εργασίας, θα χρησιμοποιεί ένα μόνο private SSH key για πρόσβαση σε όλα τα συστήματα που ελέγχει. Οπότε, θα θεωρείται, επίσης, δεδομένο ότι το public SSH key του controller θα είναι εγκατεστημένο στους hosts.

Κεφάλαιο 4

Ενορχήστρωση Εικονικού Ιδιωτικού Δικτύου

Προτού εμβαθύνουμε στις λεπτομέρειες της ανάπτυξης της εφαρμογής, μένει να γίνει η μελέτη του πώς θα πραγματοποιηθεί η ρύθμιση των εικονικών δρομολογητών από το διαχειριστικό τους περιβάλλον. Το κεφάλαιο αυτό θα επικεντρωθεί στις σχεδιαστικές αποφάσεις που θα ληφθούν για την υλοποίηση της τοπολογίας, καθώς και το πώς θα γίνουν οι απαραίτητες ρυθμίσεις στο περιβάλλον του VyOS. Τέλος, στα πλαίσια αυτού του κεφαλαίου, θα ολοκληρωθεί η διαδικασία της ρύθμισης της τοπολογίας με χρήση του Ansible.

4.1 Απόδοση Δημόσιων Διευθύνσεων στους Εικονικούς Δρομολογητές

Απαιτώντας διαχείριση μέσω SSH με τη δυνατότητα για απομακρυσμένη πρόσβαση από τον controller στους δρομολογητές, με τους ESXi hosts να είναι προσβάσιμοι από το δημόσιο internet, οι εικονικοί δρομολογητές οφείλουν να είναι προσβάσιμοι από τον controller, με τη γενική περίπτωση να συνεπάγεται πρόσβαση μέσω δημόσιας διεύθυνσης IP. Στην ενότητα αυτή, θα εξερευνηθούν οι επιλογές, μέσω των οποίων οι εικονικοί δρομολογητές, που θα δημιουργούνται, μπορούν να γίνουν ορατοί στο δημόσιο δίκτυο. Αν και στα πλαίσια της εργασίας θα υλοποιηθεί μόνο μία από τις μεθόδους απόδοσης διευθύνσεων που θα αναφερθούν, η μελέτη τους έχει αξία τόσο για την πληρότητα της ανάλυσης πιθανών τρόπων μελλοντικών επεκτάσεων της εφαρμογής αλλά και για υπογράμμιση των διαχειριστικών και σχεδιαστικών προκλήσεων που προκύπτουν κατά την υλοποίηση της εργασίας.

4.1.1 Χρήση DHCP Client

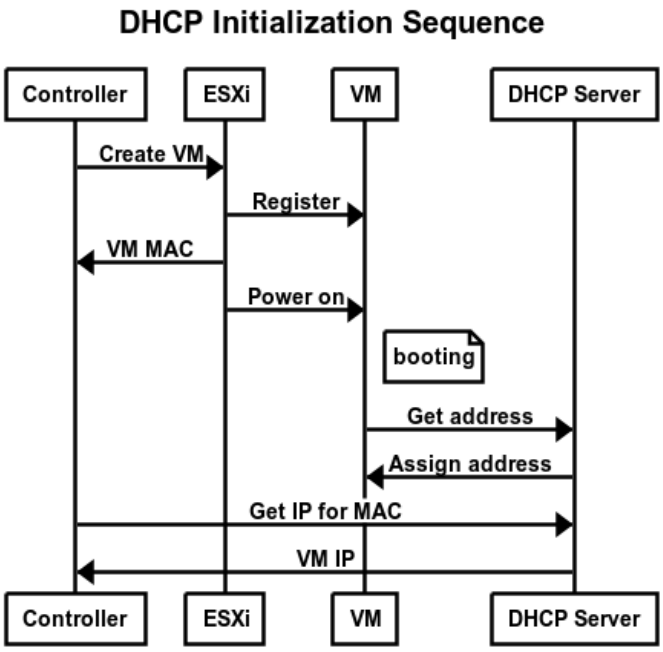
Ίσως η πιο προφανής προσέγγιση αυτού του ζητήματος είναι η απόδοση διευθύνσεων στους καινούργιους δρομολογητές μέσω DHCP. Το πρωτόκολλο αυτό χρησιμοποιεί το πρωτόκολλο Ethernet, ώστε ένας εξυπηρετητής, χωρίς διεύθυνση IP σε κάποια διεπαφή του, να έχει τη δυνατότητα να αποκτήσει διεύθυνση από κάποιον DHCP server, με ισχύ στη συγκεκριμένη περιοχή, κάνοντας ερωτήματα μέσω broadcast. Το μόνο που απαιτείται από τη μέθοδο αυτή, κατά την αρχική παραμετροποίηση των VM των δρομολογητών, είναι η ενεργοποίηση του DHCP client στην ενεργή διεπαφή. Η μέθοδος αυτή έχει τα εξής **πλεονεκτήματα:**

- **Παραλληλία**: Πολλαπλές εικονικές μηχανές μπορούν να κάνουν boot ταυτόχρονα, καθώς και οι τυχαίως παραγόμενες MAC διευθύνσεις τους κατά το register δεν θα προκαλέσουν κάποιο conflict εντός του ίδιου δικτύου. Αφού ολοκληρώσουν το boot, θα αποκτούν μοναδική διεύθυνση από τον DHCP server.

- **Φορητό Zero-Day configuration:** Η μέθοδος αυτή απαιτεί ως zero-day configuration μόνο την ενεργοποίηση του DHCP client. Αυτό είναι πλεονέκτημα ως προς άλλες μεθόδους, που απαιτούν παραμετροποίηση συγκεκριμένης διεύθυνσης

στον εικονικό δίσκο του VM, διότι αυτό το κάνει ενδεχομένως μη επαναχρησιμοποιήσιμο σε άλλο δίκτυο.

Τα μειονεκτήματα, όμως, αυτής της μεθόδου παρουσιάζονται όταν μελετηθεί η αλληλουχία γεγονότων, με την οποία η διεύθυνση IP, που αποδόθηκε σε ένα νέο εικονικό δρομολογητή, θα γνωστοποιείται στον controller. Η αλληλουχία αυτή προέκυψε δεδομένου του ότι δεν μπορεί να γίνει ερώτημα προς το VM που μόλις δημιουργήθηκε χωρίς να γνωρίζουμε τη διεύθυνσή του.



5. Διαδικασία Απόδοσης Διεύθυνσης Μέσω DHCP

Όπως φαίνεται στο παραπάνω sequence diagram, η γνωστοποίηση του controller για την διεύθυνση IP που θα αποδοθεί σε ένα νέο εικονικό δρομολογητή έχει ως συνέπεια τους παρακάτω **περιορισμούς**:

- **Γνωστοποίηση MAC**: Ο controller πρέπει κάπως να ενημερωθεί για τη διεύθυνση MAC που παρήχθη για την κύρια διεπαφή του VM κατά την εγγραφή του (register) στο ESXi. Γνωρίζοντας το Vmid της εικονικής μηχανής, η MAC διεύθυνση μπορεί να ανακτηθεί είτε διαβάζοντας το **vmx** αρχείο μετά το register ή με χρήση της εντολής: `vim-cmd vmsvc/device.getdevices <id>`.

- **Παρουσία DHCP server**: Με την επιλογή της χρήσης DHCP server για απόδοση διευθύνσεων, δημιουργείται η δέσμευση για παρουσία ενός τέτοιου εξυπηρετητή εντός εμβέλειας broadcast από τα VM που δημιουργούνται.

- **Ανάκτηση IP από τον DHCP server**: Η απαίτηση κάποιου προγραμματιστικού interface (API), μέσω του οποίου ο controller θα επικοινωνεί με τον DHCP server για να ανακτά τη διεύθυνση που έχει αποδοθεί σε κάποιο VM δεδομένης της MAC του, προϋποθέτει είτε τη χρήση εξειδικευμένων εξυπηρετητών DHCP ή την ανάπτυξη επιπλέον λογισμικού στο περιβάλλον του δικτύου, που να επιτελεί αυτή τη λειτουργία. Περιορίζεται, λοιπόν, σημαντικά τη συμβατότητα με τα περισσότερα περιβάλλοντα και σενάρια χρήσης.

Δεδομένων των περιορισμών αυτών, η χρήση του DHCP για απόδοση διευθύνσεων δεν θα εξεταστεί στα πλαίσια της εργασίας και θα θεωρηθεί πιθανός τρόπος επέκτασης της λειτουργικότητας.

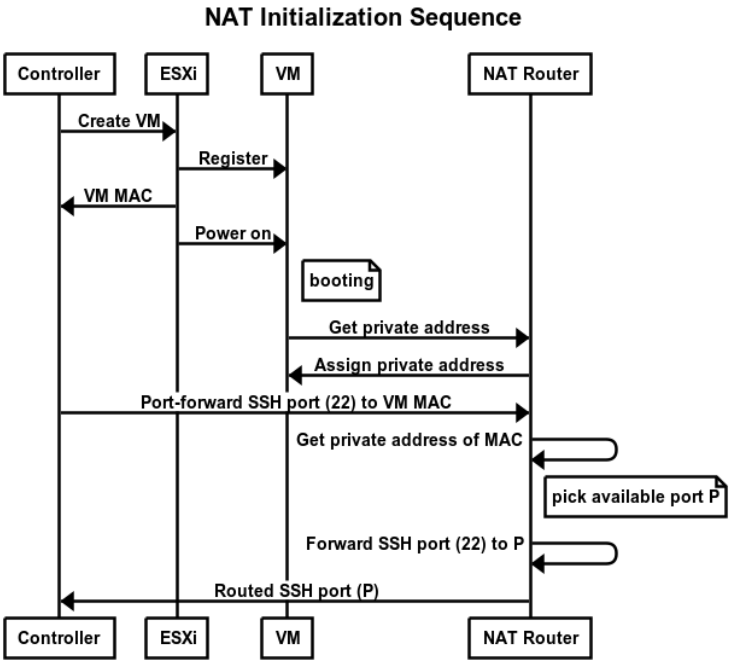
4.1.2 NAT και Αυτόματη Προώθηση Θυρών

Μία πιθανή προσέγγιση είναι η αποφυγή της ανάθεσης δημοσίων διευθύνσεων με χρήση του πρωτοκόλλου **Network Address Translation** (NAT). Το πρωτόκολλο αυτό επιτρέπει την επικοινωνία μεταξύ των κόμβων ενός ιδιωτικού δικτύου αλλά και τη συνδεσιμότητά τους με το διαδίκτυο, χωρίς την δέσμευση κάποιας δημοσίως δρομολογήσιμης διεύθυνσης για τον κάθε κόμβο. Για να επιτευχθεί αυτό, όλη η κίνηση από και προς το συγκεκριμένο δίκτυο δρομολογείται μέσω ενός συγκεκριμένου δρομολογητή, ο οποίος, σύμφωνα με το πρωτόκολλο, μεταγλωττίζει τη μία και μοναδική δημόσια διεύθυνση που του αντιστοιχεί στην κατάλληλη ιδιωτική διεύθυνση κάποιου κόμβου του δικτύου. Η μεταγλώττιση αυτή πραγματοποιείται με βάση τις θύρες (**ports**), που έχει ως πηγή ή προορισμό η κίνηση από και προς το δίκτυο αντίστοιχα και έχει ως βάση το επίπεδο μεταφοράς (transport layer) του OSI. Επίσης, συνήθως οι διευθύνσεις των κόμβων του ιδιωτικού δικτύου αποδίδονται από τον ίδιο δρομολογητή μέσω SSH.

Αναζητώντας τις δυνατότητες του ESXi, διαπιστώνεται ότι δεν παρέχεται η δυνατότητα ούτε για διασύνδεση των εικονικών μηχανών πίσω από NAT ούτε και η απόδοση διευθύνσεων μέσω DHCP, καθώς και ο hypervisor του ESXi δεν λειτουργεί ως δρομολογητής, αλλά χρησιμοποιεί εικονικό μεταγωγέα (switch). Οπότε, μία ανάπτυξη με χρήση του NAT μεταξύ των εικονικών μηχανών ενός ESXi host απαιτεί την ανάπτυξη και την εγκατάσταση ενός νέου **NAT Router**, ο οποίος θα επιτελεί τους παρακάτω ρόλους.

- Λειτουργία DHCP εξυπηρετητή για απόδοση ιδιωτικών διευθύνσεων στους νέους εικονικούς δρομολογητές.
- Μεταγλώττισης διευθύνσεων (NAT) από τη μία δημόσια διεύθυνση που θα του αποδίδεται στις ιδιωτικές των εικονικών δρομολογητών.
- Λογισμικό εξυπηρετητή (server), το οποίο θα ταιριάζει κάποια MAC διεύθυνση με την ιδιωτική IP, που θα της έχει αποδώσει ο DHCP server, και θα παρέχει διεπαφή (API), μέσω της οποίας ο controller θα ζητά τη προώθηση μίας διαθέσιμης θύρας στην θύρα του SSH ενός νέου εικονικού δρομολογητή (**port-forwarding**).

Δεδομένης της ανάπτυξης του **NAT Router** και της εγκατάστασής του σε κάποιον ESXi host, αποδίδοντας του δημόσια δρομολογήσιμη διεύθυνση – γνωστή στον controller, – η διαδικασία δημιουργίας και αρχικοποίησης ενός νέου εικονικού δρομολογητή φαίνεται στο παρακάτω sequence diagram.



6. Εγκατάστασης Δημόσιας Πρόσβασης Μέσω NAT

Αν και πιο περίπλοκη στην υλοποίηση, η ιδέα αυτή παρέχει τα ίδια πλεονεκτήματα με τη χρήση DHCP server, που περιγράφηκε στην προηγούμενη ενότητα. Επιπλέον, η χρήση NAT για αυτή την εφαρμογή είναι σημαντικά πιο οικονομική και κλιμακώσιμη στη χρήση δημόσια δρομολογήσιμων διευθύνσεων – τουλάχιστον όσο χρησιμοποιούμε IPv4 -, καθώς και δεν απαιτεί μία για κάθε εικονικό δρομολογητή.

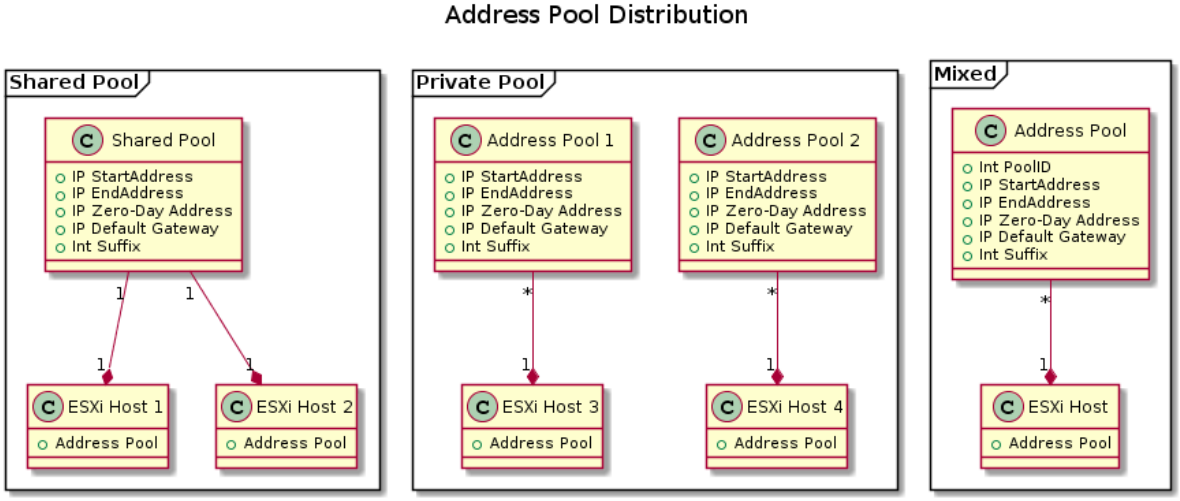
4.1.3 Απόδοση από Pool Διευθύνσεων

Ως pool διευθύνσεων (**Address Pool**) ορίζουμε ένα σετ από διαθέσιμες διευθύνσεις προς ανάθεση. Ούσες, συνήθως, συνεχόμενες και ανήκοντας στο ίδιο υποδίκτυο (subnet), οι διευθύνσεις αυτές επιμελούνται από μία συσκευή, έναν άνθρωπο ή κάποια άλλη διαχειριστική οντότητα και χρησιμοποιούνται για την ανάθεση σε υπολογιστικά συστήματα, που χρειάζονται διευθυνσιοδότηση σε κάποιο δίκτυο. Ο υπεύθυνος του pool συνήθως, επίσης, μεριμνά για την αποτελεσματική χρησιμοποίηση των διευθύνσεων, διατηρώντας κατάλογο με τις διαθέσιμες και δεσμευμένες διευθύνσεις και φροντίζοντας για την επαναχρησιμοποίηση των διευθύνσεων που αποδεσμεύονται.

Η λογική της υλοποίησης αυτής είναι ότι ο controller θα περιέχει στις ρυθμίσεις του pool διευθύνσεων για το σκοπό της διαχείρησης των εικονικών δρομολογητών, που θα δημιουργεί. Ελέγχοντας εσωτερικά το pool αυτό, το οποίο θα είναι δεσμευμένο από τους διαχειριστές της εφαρμογής για το σκοπό αυτό, θα αποδίδει από μία διεύθυνση σε κάθε νέο εικονικό δρομολογητή. Η προσέγγιση αυτή μπορεί να φαίνεται παρόμοια με τη χρήση ενός DHCP server, όπως περιγράφηκε προηγουμένως, είναι, όμως, σημαντικά πιο ευέλικτη, καθώς δεν απαιτεί την παρουσία εξυπηρετητή DHCP στο περιβάλλον των ESXi host και δεν έχει επιπλέον απαιτήσεις ως προς την γνωστοποίηση των ανατιθέμενων διευθύνσεων από τον controller. Στην περίπτωση αυτής της εφαρμογής, θα ορίσουμε ένα pool συνεχόμενων διευθύνσεων ως ένα αντικείμενο με τις παρακάτω ιδιότητες:

- **Start Address**: Η πρώτη διεύθυνση του pool. Ο controller θα ξεκινά την ανάθεση διευθύνσεων από αυτήν, συνεχίζοντας στο pool, αυξάνοντας την διεύθυνση κατά μία μονάδα κάθε φορά.
- **End Address**: Η τελική διεύθυνση του pool.
- **Zero-Day Address**: Η διεύθυνση που είναι δεσμευμένη για την χρήση από μόλις δημιουργημένες εικονικές μηχανές στην εμβέλεια ισχύς του συγκεκριμένου pool, χωρίς πιθανές συγκρούσεις.
- **Default Gateway**: Η προεπιλεγμένη πύλη (default gateway) για τις εικονικές μηχανές, που θα αποκτούν τη δημόσια διεύθυνσή τους από το συγκεκριμένο pool. Στα δίκτυα που χρησιμοποιούν DHCP server για απόδοση διευθύνσεων, συνηθίζεται το default gateway να δίνεται από τον server αυτόν ως επιπλέον πληροφορία μαζί με τη διεύθυνση.
- **Address Suffix**: Μια ακόμη πληροφορία που υπό άλλες συνθήκες διανέμεται από τον υπεύθυνο DHCP είναι η μάσκα υποδικτύου (subnet mask). Καθώς και αυτό πρέπει, επίσης, να συμπεριληφθεί στην προγραμματιστική ρύθμιση της διεύθυνσης ενός host, κάθε pool θα περιέχει και την κατάληξη (suffix) του σχετικού υποδικτύου, που ουσιαστικά είναι ο αριθμός των bit της μάσκας.
- **Pool ID**: Προαιρετικά, για pool διευθύνσεων, που μοιράζεται από πολλαπλούς ESXi hosts, θα ορίζεται και ένας μοναδικός κωδικός (ID) για το συγκεκριμένο pool, κατά την παραμετροποίησή του στον controller, ώστε να μπορεί, σε επίπεδο λογισμικού, να χειρίζεται για όλους τους hosts που το χρησιμοποιούν, με τους hosts αυτούς να αναφέρονται απλά στο ID του pool, αντί να διατηρούν δικό τους αντίγραφο από αυτό.

Όσον αφορά την παραμετροποίηση των address pool στην εφαρμογή, διακρίνονται τρείς μέθοδοι που απεικονίζονται στα παρακάτω διαγράμματα.



7. Μέθοδοι Κατανομής Pool Διεθύνσεων

- **Shared Pool**
  Η εφαρμογή χρησιμοποιεί ένα μοναδικό κοινόχρηστο pool διευθύνσεων, το οποίο χρησιμοποιείται για την απόδοση διευθύνσεων σε όλους τους ESXi hosts που χρησιμοποιούνται από την εφαρμογή. Αποτελεί καλή υλοποίηση, λόγω της απλότητάς της, όταν οι hosts αυτοί βρίσκονται στο ίδιο υποδίκτυο δρομολογήσιμων διευθύνσεων, όμως, περιορίζει την εφαρμογή στη χρήση hosts μόνο στο υποδίκτυο αυτό.

- **Private Pool**
  Ορίζονται ιδιωτικά διακριτά pool διευθύνσεων, ώστε να γίνεται ανάθεση ενός σε κάθε έναν ESXi host. Είναι, επίσης, απλό στην υλοποίηση και, σε αντίθεση με το shared pool, επιτρέπει την χρησιμοποίηση host συστημάτων σε ποικίλα μέρη του διαδικτύου. Το μειονέκτημα είναι ότι, επειδή – χωρίς επιπλέον προγραμματιστικούς ελέγχους – τα pool δεν πρέπει να μοιράζονται διευθύνσεις ή να παρουσιάζουν επικάλυψη (overlap) στο εύρος τους, περιορισμός που μπορεί να οδηγήσει σε μη-βέλτιστη χρησιμοποίηση (utilization) του διαθέσιμου εύρους διευθύνσεων εντός ενός υποδικτύου, εάν αυτό χρειαστεί να διαιρεθεί μεταξύ πολλαπλών pool διευθύνσεων.

- **Mixed**
  Η τεχνική αυτή χρησιμοποιεί την απόδοση μοναδικών κωδικών (ID) στα διαθέσιμα pool, επιτρέποντας έτσι στην παραμετροποίηση (configuration) της εφαρμογής να αντιστοιχίζει τον κάθε διαθέσιμο ESXi host σε ένα από τα pool διευθύνσεων. Ύστερα, στο επίπεδο του λογισμικού, το κάθε address pool θα λειτουργεί ως δικιά του οντότητα (class), αποδίδοντας διευθύνσεις μέσω προγραμματιστικής διεπαφής με το υπόλοιπο πρόγραμμα, ελέγχοντας, έτσι, εσωτερικά την κατάσταση και διαθεσιμότητα των διευθύνσεων.

Συμπεραίνοντας, η mixed τεχνική παραμετροποίησης των address pool παρέχει υπερσύνολο δυνατοτήτων των δύο άλλων τεχνικών. Με την κατάλληλη παραμετροποίηση, μπορεί να προσδιορίζει κοινόχρηστα (shared) pools, ιδιωτικά (private) pools ή ένα μείγμα και των δύο. Δεδομένου του ότι είναι η πιο ευέλικτη, θα επιλεχθεί τελικά για την υλοποίηση του address pool της εφαρμογής, παρά της ελαφρώς μεγαλύτερης πολυπλοκότητας σχετικά με τις άλλες δύο.

4.2 Ρύθμιση Δρομολογητών

Στην ενότητα αυτή, θα γίνει η περιγραφή και η ανάλυση της διαδικασίας ρύθμισης των δρομολογητών, που θα κατασκευάσουν την τοπολογία. Σκοπός είναι η θεωρητική θεμελίωση των μεθόδων παραμετροποίησης και διαχείρισης των συστημάτων που θα αναφερθούν, ώστε να μπορούν να χρησιμοποιηθούν κατά την ανάπτυξη του λογισμικού στο τεχνικό κομμάτι της εργασίας, αλλά και η εξοικείωση του αναγνώστη με τα εν λόγω συστήματα.

4.2.1 Εισαγωγή στο VyOS

Το VyOS, όπως προαναφέρθηκε, αποτελείται από ένα πακέτο λογισμικού, αναπτυσσόμενο πάνω στο λειτουργικό των Debian Linux και είναι ο εικονικός δρομολογητής, που θα χρησιμοποιηθεί στα πλαίσια αυτής της εργασίας ως η υποδομή των εικονικών δρομολογητών των ιδιωτικών δικτύων, που θα υλοποιούνται. Η ανάπτυξή του ξεκίνησε το 2013 ως διακλάδωση του “Vyatta Core 6.6R1”, που υπάγονταν σε άδεια ανοιχτού λογισμικού, όταν διακόπηκε η υποστήριξη για την δωρεάν εκδοχή του [13]. Επίσης, η σύνταξη των ρυθμίσεών ενός συστήματος VyOS, καθώς και ο τρόπος λειτουργίας της γραμμής εντολών του είναι βασισμένα στο λειτουργικό σύστημα “Juniper JUNOS”.

Αν και το περιβάλλον του VyOS είναι βασισμένο στο Linux και μοιράζεται πολλές λειτουργίες και δυνατότητες με αυτό, στην πράξη η γραμμή εντολών συμπεριφέρεται πιο διαδραστικά, με έμφαση στην αυτόματη συμπλήρωση εντολών και παραμέτρων. Για παράδειγμα, χρησιμοποιώντας το πλήκτρο του ερωτηματικού "**?**" στο πληκτρολόγιο, το τερματικό εκτυπώνει στην έξοδό του μία περιγραφική λίστα με όλες τις εντολές παραμετροποίησης, που είναι διαθέσιμες τη συγκεκριμένη στιγμή στο χρήστη. Επίσης, χρησιμοποιώντας το πλήκτρο "**TAB**" του πληκτρολογίου, το τερματικό συμπληρώνει την ημιτελή εντολή που πληκτρολογείται εκείνη τη στιγμή ή, αν η συμπλήρωση δεν είναι δυνατή λόγω ασάφειας από πολλαπλές πιθανές εντολές, παρουσιάζονται οι επιλογές αυτές στο χρήστη. Τέλος, το τερματικό αυτό δεν απαιτεί την πλήρη πληκτρολόγηση των εντολών και αρκείται στην είσοδο της συντομότερης δυνατής ακολουθίας χαρακτήρων, από την οποία είναι δυνατόν να προκύψει το όνομα μίας μόνο από τις διαθέσιμες εντολές. Η γραμμή εντολών του λειτουργικού συστήματος, μέσω της οποίας πραγματοποιείται η παραμετροποίηση του δρομολογητή, έχει δύο καταστάσεις λειτουργίας:

- **Operational Mode**
  Στην κατάσταση αυτή, ο δρομολογητής δεν είναι παραμετροποιήσιμος και επιτρέπεται η εκτέλεση μόνο των εντολών ανάκτησης πηλοφοριών. Μία από τις κύριες εντολές σε αυτή την κατάσταση είναι η **show,** που χρησιμοποιείται για την εκτύπωση της τρέχουσας παραμετροποίησης του δρομολογητή και, συγκεκριμένα, για την ανάκτηση συγκεκριμένων παραμέτρων, όπως της κατάστασης των διεπαφών ή τον πίνακα δρομολόγησης. Άλλες χρήσιμες εντολές είναι οι διαγνωστικές **ping**, **traceroute** και **monitor**, καθώς και η εντολή **configure,** που εισάγει το τερματικό στην κατάσταση παραμετροποίησης.

- **Configuration Mode**
  Στην κατάσταση παραμετροποίησης, ο χρήστης έχει πρόσβαση σε εντολές, με τις οποίες μπορεί να επηρεάσει την τρέχουσα παραμετροποίηση του δρομολογητή. Η λειτουργία στην κατάσταση αυτή σηματοδοτείται από την ύπαρξη ένδειξης "[edit]" πάνω από προτροπή (prompt) της γραμμής εντολών.
  Η κύρια εντολή στο περιβάλλον αυτό είναι η εντολή **set**, μέσω της οποίας μπορούν να οριστούν οι διάφοροι παράμετροι λειτουργίας του δρομολογητή.
  Για να διατηρείται η ακεραιότητα στην κατάσταση αυτή, εντολές που προκαλούν αλλαγές στην παραμετροποίηση δεν έχουν ισχύ όταν καταχωρούνται. Οι αλλαγές αυτές συγκρίνονται με την τρέχουσα παραμετροποίηση και παράγουν μία ενιαία ενημέρωση (patch), που διατηρείται για όλη τη διάρκεια παραμονής στο configuration mode. Για την εφαρμογή αυτής της ενημέρωσης, απαιτείται η χρήση της εντολής **commit**, μέσω της οποίας η ενημέρωση, που έχει συνταχθεί μέχρι στιγμής, εφαρμόζεται πλήρως ως μία μοναδική αλλαγή (atomically) στην τρέχουσα παραμετροποίηση του συστήματος.
  Με τη χρήση της εντολής **commit,** ενημερώνεται μόνο η τρέχουσα παραμετροποίηση, που σημαίνει ότι οι αλλαγές που μόλις εφαρμόστηκαν θα χαθούν την επόμενη φορά που θα εκτελέσει επανεκκίνηση ο δρομολογητής. Για την μόνιμη εφαρμογή των αλλαγών, απαιτείται χρήση της εντολής **save**, η οποία αποθηκεύει την τρέχουσα παραμετροποίηση στο αρχείο "**/config/config.boot**", από το οποίο ο δρομολογητής ανακτά τις ρυθμίσεις του, κατά την έναρξη του συστήματος.
  Τέλος, έξοδος από το configuration mode επιτυγχάνεται με την εντολή **save**.

4.2.2 Ρύθμιση Ζεύξεων και Δρομολόγηση

Έχοντας εξοικειωθεί με το περιβάλλον του VyOS, στην ενότητα αυτή, θα γίνει η εξερεύνηση των εντολών παραμετροποίησης, με τις οποίες θα πραγματοποιηθεί ο προγραμματισμός του δρομολογητή, που απαιτείται στα πλαίσια τις εργασίας. Ο προγραμματισμός αυτός, όπως έχει αναφερθεί στο κεφάλαιο σχετικά με την περιγραφή της εργασίας, περιλαμβάνει την ανάθεση μίας δημόσιας διαχειριστικής διεύθυνσης στην κύρια διεπαφή του δρομολογητή, την δημιουργία διεπαφών σήραγγας (tunnel interfaces) μεταξύ του δρομολογητή και των γειτονικών του στον γράφο της τοπολογίας και την προσθήκη στατικών διαδρομών προς όλους τους άλλους δρομολογητές του ιδιωτικού δικτύου.

Αλλαγή του hostname του δρομολογητή σε **router1**.

```
$ set system host-name router1
```

Ορισμός της διεύθυνσης **147.102.39.41/26** στη διεπαφή **eth0**.

```
$ set interfaces ethernet eth0 address 147.102.39.41/26
```

Ορισμός της διεύθυνσης **147.102.39.1** ως προεπιλεγμένη πύλη (default gateway).

```
$ set system gateway-address 147.102.39.1
```

Υποθέτοντας ότι η δημόσια διεύθυνση του δρομολογητή είναι η **147.102.39.31,** για τη δημιουργία tunnel interface **tun0**, που λειτουργεί με ενθυλάκωση **GRE**, μεταξύ αυτού και απομακρυσμένου δρομολογητή με δημόσια διεύθυνση **147.102.39.32** χρησιμοποιούνται οι παρακάτω εντολές.

```
$ set interfaces tunnel tun0 encapsulation gre
$ set interfaces tunnel tun0 local-ip 147.102.39.31
$ set interfaces tunnel tun0 remote-ip 147.102.39.32
```

Ανάθεση της διεύθυνσης **10.0.0.1** στο παραπάνω tunnel interface.

```
$ set interfaces tunnel tun0 address 10.0.0.1/32
```

Και τέλος, προσθήκη στατικής διαδρομής (static route), η οποία θα δρομολογεί την κίνηση με προορισμό τη διεύθυνση **10.0.0.2** στο tunnel interface.

```
$ set protocols static interface-route 10.0.0.2/32 next-hop-interface tun0
```

Η τελευταία στατική διαδρομή απαιτείται στην υλοποίηση της εργασίας, διότι χωρίς μάσκα υποδικτύου, δηλαδή πλήρη ελευθερία στην επιλογή διευθύνσεων από τον χρήστη, απαιτείται οδηγία για την δρομολόγηση κίνησης με προορισμό τον απομακρυσμένο δρομολογητή στο άλλο άκρο ενός tunnel interface, μέσω του συγκεκριμένου interface.

4.2.3 Ρυθμίσεις Zero-Day

Ο αγγλικός όρος Zero-Day Configuration αναφέρεται στην **διαμόρφωση ενός συστήματος και την κατάσταση της παραμετροποίησής του κατά την εκκίνησή του για πρώτη φορά**. Ένα υπολογιστικό σύστημα, το οποίο διαχειρίζεται απομακρυσμένα, δεν έχει τη δυνατότητα να λάβει καινούργιες εντολές παραμετροποίησης προτού να γίνει ορατό στο σχετικό δίκτυο και προσβάσιμο μέσω αυτού. Για τον λόγο αυτό, ένα τέτοιο σύστημα πρέπει είτε να έχει τη δυνατότητα να γίνεται ορατό στο δίκτυό του αυτόνομα ή να μπορεί να ανακτήσει ενεργητικά δεδομένα ή ρυθμίσεις επικοινωνώντας με άλλους πόρους ή εξυπηρετητές στο δίκτυο αυτό. Το zero-day configuration, λοιπόν, αφορά τις ρυθμίσεις και παραμέτρους, που τοποθετούνται στο περιβάλλον ενός συστήματος, έτσι ώστε να προϋπάρχουν σε ένα σύστημα, κατά την εκκίνησή του.

Η ενότητα αυτή, θα αναφερθεί στις ρυθμίσεις, που θα εφαρμοστούν στην εικονική μηχανή VyOS, από την οποία θα δημιουργούνται οι εικονικοί δρομολογητές, κατά την εγκατάσταση του εικονικού δικτύου, μέσω κλωνοποίησης στους ESXi hosts, όπως περιγράφηκε στο σχετικό κεφάλαιο. Ακολουθώντας τις προδιαγραφές της εφαρμογής που αναπτύχθηκαν στα προηγούμενα κεφάλαια, οι Zero-Day ρυθμίσεις, που θα τοποθετούνται στην εν λόγω εικονική μηχανή, είναι οι παρακάτω:

Αρχικά, πρέπει να ενεργοποιηθεί η πρόσβαση μέσω **SSH** καθώς και η πρόσβαση για τον χρήστη **root**.

```
$ set service ssh
$ set service ssh allow-root
```

Ύστερα, θα προστίθεται το δημόσιο κλειδί SSH του controller στα εξουσιοδοτημένα κλειδιά του χρήστη **vyos**.

```
$ echo "ssh-rsa AAAA..." > /home/vyos/pub_key
$ loadkey vyos /home/vyos/pub_key
```

Θα ρυθμίζεται η στατική Zero-Day διεύθυνση από το σχετικό pool διευθύνσεων, ώστε ο δρομολογητής να γίνει προσιτός από τον controller μετά την εκκίνησή του.

```
$ set interfaces ethernet eth0 address 147.102.39.41/26
$ set system gateway-address 147.102.39.1
```

Επίσης, για να υπάρχει η δυνατότητα εγγραφής αρχείων, πρέπει να διορθωθεί η ιδιοκτησία του φακέλου του χρήστη **vyos**, καθώς και σε ορισμένες περιπτώσεις ο φάκελος αυτός ανήκει, αρχικά, στον χρήστη **root**. Η ρύθμιση αυτή είναι απαραίτητη, καθώς και η παραμετροποίηση των δρομολογητών θα υλοποιηθεί με χρήση του Ansible, που θα περιγραφεί μετέπειτα στην εργασία, το οποίο δημιουργεί προσωρινά αρχεία κατά τη ρύθμιση ενός συστήματος.

```
$ sudo chown -R vyos /home/vyos
```

Τέλος, κατά τη διάρκεια των δοκιμών λειτουργίας των zero-day ρυθμίσεων κλωνοποιημένων εικονικών μηχανών, παρουσιάστηκε μία περιπλοκή σχετικά με την ισχύ της στατικής zero-day διεύθυνσης στο δίκτυο του πολυτεχνείου, που εκτελέστηκαν τα πειράματα. Καθώς η διεύθυνση αυτή χρησιμοποιούνταν διαδοχικά από πολλαπλούς δρομολογητές μετά την εκκίνηση τους, υπήρχε το ενδεχόμενο κάποιος δρομολογητής να μην είναι προσβάσιμος από εκτός του δικτύου στην διεύθυνση αυτή. Συγκεκριμένα, εάν ένας δρομολογητής ενεργοποιούταν με τη διεύθυνση αυτή, γινόταν ping και, ύστερα, απενεργοποιούταν ή αποδέσμευε τη zero-day διεύθυνση, ο επόμενος εικονικός δρομολογητής, που δημιουργούταν, δεν ήταν προσβάσιμος στη διεύθυνση αυτή. Παρατηρήθηκε ότι ένας δρομολογητής σε αυτή την κατάσταση γινόταν προσβάσιμος εάν ο ίδιος ξεκινούσε μία εξερχόμενη δικτυακή επικοινωνία, όπως, για παράδειγμα, μία εκτέλεση της εντολής ping. Δεδομένης αυτής της παρατήρησης, η κύρια θεωρία για την εξήγηση αυτού του γεγονότος είναι ότι ο δρομολογητής που ελέγχει την δρομολόγηση πακέτων από και προς το συγκεκριμένο τμήμα του δικτύου του εργαστηρίου αποθηκεύει την zero-day διεύθυνση στην ARP cache του την πρώτη φορά που την συναντά, επιχειρώντας, έτσι, να δρομολογήσει πακέτα, με προορισμό τη διεύθυνση αυτή προς τη διεύθυνση MAC της πρώτης εικονικής μηχανής που τη χρησιμοποιεί. Έτσι, σύμφωνα με τη θεωρία αυτή, μία εικονική μηχανή, που επαναχρησιμοποιεί την zero-day διεύθυνση μετά από κάποια άλλη, θα παραμείνει απρόσιτη από το δημόσιο δίκτυο, μέχρι να λήξει η σχετική εγγραφή στην ARP cache του τοπικού δρομολογητή.

Για την επίλυση αυτού του προβλήματος, θα επιλεχθεί λύση που δεν απαιτεί παρεμβάσεις στο υπόλοιπο δίκτυο ή την εισαγωγή επιπλέον περιορισμών και παραδοχών στην ανάπτυξη της εφαρμογής. Χρησιμοποιώντας το αρχείο **"/config/scripts/vyatta-postconfig-bootup.script**", επιτρέπεται ο ορισμός εντολών προς εκτέλεση κατά την εκκίνηση του λειτουργικού συστήματος, αμέσως μετά από την εφαρμογή των αποθηκευμένων ρυθμίσεων. Συγκεκριμένα, αρκεί η εκτέλεση μίας μοναδικής επανάληψης της εντολής ping, ώστε να ενημερωθεί ο δρομολογητής και η εικονική μηχανή να γίνει προσβάσιμη στην zero-day διεύθυνση. Οπότε, στο αρχείο αυτό θα προστεθεί το παρακάτω:

```
# /config/scripts/vyatta-postconfig-bootup.script

ping -c 1 8.8.8.8
```

4.3 Παραμετροποίηση των Εικονικών Δρομολογητών του Δικτύου με Χρήση του Ansible

Η ενότητα αυτή θα ασχοληθεί με το τελευταίο – και κυριότερο – στάδιο της ενορχήστρωσης των εικονικών ιδιωτικών δικτύων (VPN) που θα δημιουργούνται. Το στάδιο αυτό θα αποτελείται από τις κατάλληλες ενέργειες, ώστε η ήδη δημιουργημένες και αρχικοποιημένες εικονικές μηχανές δρομολογητών να παραμετροποιηθούν κατάλληλα, ώστε η συνδεσιμότητα μεταξύ τους να ταιριάζει με αυτή που περιγράφεται στη δικτυακή τοπολογία, που έχει σχεδιάσει και υποβάλει ο χρήστης στην εφαρμογή. Στα πλαίσια αυτής της παραμετροποίησης, οι εικονικές μηχανές του δικτύου έχουν ολοκληρώσει την εκκίνησή τους και τους έχει αποδοθεί δημόσια διαχειριστική διεύθυνση, μέσω της οποίας θα έχει πρόσβαση σε αυτές το λογισμικό, που θα αναλάβει την παραμετροποίηση τους. Στην υλοποίηση αυτού του σταδίου, ενδιαφέρον παρουσιάζουν οι ακόλουθοι παράγοντες:

- **Ικανότητα προσαρμογής σε εναλλακτικά σενάρια.**
  Όπως έχει προαναφερθεί, η μέθοδος της αναπαράστασης της συνδεσιμότητας των δρομολογητών στις ρυθμίσεις τους είναι αντικείμενο επιλογής τους σεναρίου χρήσης της εφαρμογής. Η υλοποίηση στα πλαίσια της εργασίας αυτής, για παράδειγμα, θα ακολουθήσει τη μέθοδο της συνδεσιμότητας μέσω σηράγγων με GRE ενθυλάκωση, όμως, κάποια άλλη υλοποίηση θα έπρεπε να έχει τη δυνατότητα να χρησιμοποιήσει κάποιο άλλο πρωτόκολλο όπως IPSec ή OpenVPN.

- **Πρόσβαση στις εικονικές μηχανές αδιάφορα του λειτουργικού συστήματός τους.**
  Η πρόσβαση στα συστήματα του δικτύου για την παραμετροποίησή τους είναι σημαντικό να πραγματοποιείται με τρόπο που να μην εξαρτάται σε μεγάλο βαθμό από τα συστήματά αυτά. Δηλαδή, ιδανικά, πρέπει να μην απαιτείται επιπλέον λογισμικό από την πλευρά του υπό-διαχείριση συστήματος, καθώς και να χρησιμοποιείται κάποιο ευρέως διαθέσιμο πρωτόκολλο, όπως το SSH. Με τη δυνατότητα αυτή, ενισχύονται η συμβατότητα και η επεκτασιμότητα της εφαρμογής, καθώς και επιτρέπει την υλοποίηση με χρήση διαφορετικών εικονικών – αλλά και φυσικών – συστημάτων.

- **Παραμετροποίηση της εφαρμογής από τον χρήστη χωρίς να απαιτείται επέμβαση στο λογισμικό.**
  Με μέριμνα για την επεκτασιμότητα και επαναχρησιμοποίηση της εφαρμογής, σημαντική είναι η δυνατότητα προσαρμογής της στις απαιτήσεις κάπου σεναρίου χρήσης. Ο διαχειριστής του συστήματός, που επιθυμεί να υλοποιήσει την ανάπτυξη αυτή, πρέπει να είναι σε θέση να το επιτελέσει από την προοπτική χρήστη και όχι από την προοπτική προγραμματιστή. Αντί να απαιτείται χρονοβόρα εξοικείωση και παρέμβαση στο περίπλοκο λογισμικό της εφαρμογής, το πρόγραμμα ιδανικά θα χρησιμοποιείται ως κλειστό εργαλείο από τον τελικό χρήστη, με την επέκτασή του να πραγματοποιείται σε υψηλό επίπεδο, είτε μέσω αρχείων παραμετροποίησης ή με τη χρήση δημοφιλών τρίτων (third party) εργαλείων.

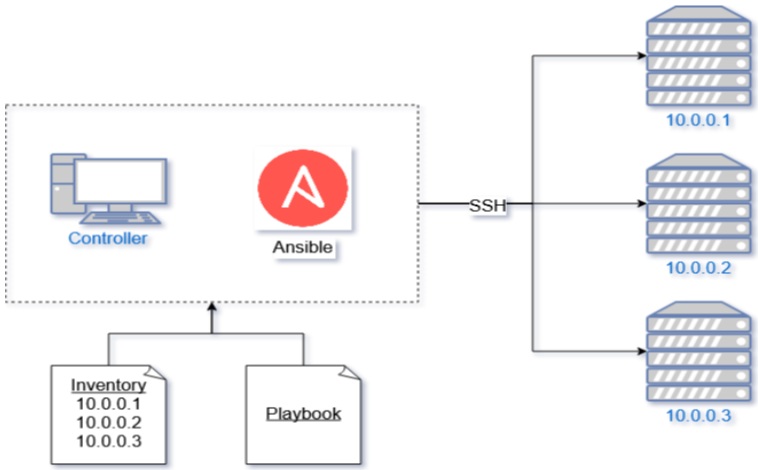
Κατανοώντας αυτές τις προδιαγραφές και έχοντας εξερευνήσει ορισμένες εναλλακτικές, ασφαλής επιλογή είναι η χρήση του **Ansible** για την λειτουργία αυτή.

4.3.1 Εισαγωγή στο Ansible

Το Ansible είναι λογισμικό ανοιχτού κώδικα (open source), το οποίο χρησιμοποιείται ευρέως από διαχειριστές δικτύων και συστημάτων για την αυτοματοποίηση διαχείρισης ρυθμίσεων και εγκαταστάσεως λογισμικού. Χρησιμοποιώντας δικτυακά πρωτόκολλα, όπως το SSH και το απομακρυσμένο PowerShell για την πρόσβαση στα υπό-διαχείριση συστήματα [14], το Ansible έχει τη δυνατότητα προγραμματισμού απομακρυσμένων λειτουργικών συστημάτων, χρησιμοποιώντας μόνο το client-side πρόγραμμα, χωρίς να απαιτείται εγκατάσταση ταιριαστού λογισμικού στα συστήματα αυτά. Η λειτουργικότητα αυτή είναι πολύ σημαντική για την ευελιξία της εφαρμογής, καθώς και δεν υπάρχει σε κάποια από τα άλλα παρόμοια προγράμματα όπως το Chef, το οποίο χρειάζεται και εγκατάσταση λογισμικού (chef client) στα προς παραμετροποίηση μηχανήματα [15]. Η δομή αυτή αποκαλείται “agentless architecture”. Στην ενότητα αυτή, λοιπόν, θα περιγραφούν οι δυνατότητες του Ansible, το πώς αυτές θα εκμεταλλευτούν για την

υλοποίηση της εργασίας και το πώς χρησιμοποιείται το Ansible, επικεντρώνοντας το ενδιαφέρον στις λειτουργίες που αφορούν την εν λόγω υλοποίηση.

Όπως και με τα περισσότερα παρόμοια προγράμματα, το Ansible χειρίζεται δύο είδη συστημάτων, έναν ελεγκτή (**controller)** και τους κόμβους (**nodes**) που ελέγχονται από αυτόν μέσω SSH. Το κύριο εκτελέσιμο πρόγραμμα του Ansible δεν εκτελεί διεργασίες συνεχόμενα (στο background) του συστήματος του controller, οπότε δεν καταναλώνει πόρους, εκτός από όταν εκτελείται ρητά. Η κύρια λειτουργία του Ansible επιτελείται με την προσωρινή εγκατάσταση και εκτέλεση τμημάτων λογισμικού, που αποκαλούνται **modules** στους κόμβους του δικτύου. Κατά την εκτέλεση του, το Ansible, στο περιβάλλον του controller, τροφοδοτείται με μία κόμβων, πάνω στους οποίους πρέπει να ενεργήσει, την οποία αποκαλεί **inventory**, εκτελώντας στους κόμβους αυτούς modules, όπως αυτά επικαλούνται από το κύριο αρχείο που περιγράφει τη διαδικασία προς αυτοματοποίηση, το οποίο το Ansible αποκαλεί **playbook**.



8. Αρχιτεκτονική Χρήσης του Ansible

Τα **modules** είναι αυτόνομα προγράμματα (scripts) και μπορούν να υλοποιούνται σε οποιαδήποτε σεναριακή (scripted) γλώσσα, όπως η Python και η Bash. Ένα module μπορεί να δέχεται παραμέτρους ορισμένες στο playbook και συνήθως σχεδιάζεται για επιτελέσει ένα συγκεκριμένο σκοπό, όπως την παραμετροποίηση κάποιου συγκεκριμένου λειτουργικού συστήματος ή την εγκατάσταση λογισμικού. Τα modules, επίσης, σχεδιάζονται κατά κανόνα, έτσι ώστε να είναι **indempotent**, που σημαίνει ότι, εκτελώντας τα πολλαπλές φορές, το σύστημα θα καταλήγει πάντα στην ίδια κατάσταση. Η κοινότητα των χρηστών του Ansible, συνεχίζοντας την φιλοσοφία του ανοιχτού λογισμικού, έχει αναπτύξει πάνω από 1800 modules, τα οποία είναι δημοσίως αναρτημένα για χρήση [16]. Μάλιστα, ακόμη περισσότερα modules μπορούν να βρεθούν στο διαδίκτυο, αναζητώντας τα σε πλατφόρμες ανοιχτού κώδικα, όπως το GitHub.

Με τον όρο **inventory** το Ansible αναφέρεται σε μία συλλογή από αρχεία, τα οποία απαριθμούν τους κόμβους, στους οποίους θα εκτελεστεί ένα playbook, και ορίζουν και ειδικές παραμέτρους σχετικά με τους κόμβους αυτούς. Ένα inventory αποτελείται από ένα δένδρο φακέλων (directory tree), το οποίο περιέχει τα σχετικά αρχεία σε συγκεκριμένη δομή. Η τοποθεσία (path) του inventory μπορεί να προσδιοριστεί, κατά την εκτέλεση του Ansible, μέσω σχετικής παραμέτρου και, εάν παραληφθεί, χρησιμοποιείται το inventory στο path "/etc/ansible". Αρχικά, ένα inventory περιέχει ένα αρχείο που λέγεται **hosts** και περιέχει τη λίστα με τους κόμβους σε μορφή INI ή YAML. Παρακάτω φαίνεται ένα παράδειγμα ενός τέτοιου αρχείου.

```
192.168.6.1

[webservers]
ws1.example.com
ws2.example.com

[dbservers]
ws1.example.com
db2.example.com
```

Όπως φαίνεται στο παράδειγμα, το αρχείο αυτό υποστηρίζει ανάμεικτη ομαδοποίηση των κόμβων, αυτό χρησιμοποιείται για τον ορισμό παραμετροποιήσεων, που αντιστοιχούν μόνο σε συγκεκριμένες ομάδες κόμβων. Ύστερα, υποστηρίζεται ο ορισμός παραμέτρων για συγκεκριμένες ομάδες, τοποθετώντας αντίστοιχα αρχεία στον φάκελο **group_vars**. Για παράδειγμα, για τον ορισμό μεταβλητών σε μορφή YAML, με ισχύ για ολόκληρη την ομάδα "**webservers**", θα δημιουργούσαμε το παρακάτω αρχείο:

```
# inventory/group_vars/webservers.yml

ansible_user: admin
ansible_connection: network_cli
ansible_network_os: debian
```

Με ταυτόσημο τρόπο μπορούν να οριστούν μεταβλητές για συγκεκριμένους κόμβους, με χρήση του φακέλου **host_vars**.

Μένει να αναφερθούμε στο κύριο κομμάτι της παραμετροποίησης του Ansible – το **playbook**, το οποίο είναι ένα YAML αρχείο, που αποτελείται από **plays**. Κάθε play στο αρχείο αυτό, έχει ως στόχο μία συγκεκριμένη ομάδα από hosts και ορίζει **tasks,** τα οποία θα εκτελεστούν σειριακά στους host αυτούς. Κάθε task συνήθως αποτελείται από μία αναφορά σε κάποιο module καθώς και τις παραμέτρους προς αυτό. Παρακάτω φαίνεται το παράδειγμα ενός playbook, που αποτελείται από δύο plays. Το πρώτο εγκαθιστά κάποια βασικά πακέτα σε όλους τους κόμβους, ενώ το δεύτερο εγκαθιστά εξυπηρετητή ιστού (web server) σε όσους κόμβους βρίσκονται στην ομάδα "**webservers**".

```
- hosts: all
  sudo: true
  vars:
     packages: [ 'vim', 'git', 'curl' ]
  tasks:
    - name: Install Package
      apt: name={{ item }} state=latest
      with_items: packages

- hosts: webservers
  tasks:
    - name: Install httpd Package
      apt: name=httpd state=latest
```

Τέλος, ένα playbook μπορεί να εκτελεστεί από τη γραμμή εντολών, πάνω σε ένα συγκεκριμένο inventory, χρησιμοποιώντας την παρακάτω εντολή:

```
$ ansible-playbook -i /path/to/inventory playbook.yml
```

Η σύνταξη του playbook, σε συνδυασμό με τον ορισμό παραμέτρων, που να αφορούν συγκεκριμένες ομάδες ή hosts, δίνουν τη δυνατότητα στο αρχείο αυτό να ορίσει οποιαδήποτε αλληλουχία δράσεων απαιτείται για τον προγραμματισμό ενός συστήματος.

4.3.2 Παραγωγή Δυναμικού Inventory

Για την παραμετροποίηση της δικτυακής τοπολογίας με χρήση του Ansible, το back-end της εφαρμογής θα εκτελεί προκαθορισμένα playbook πάνω σε inventory, το οποίο θα ορίζει τόσο τους δρομολογητές της του δικτύου, αλλά και τις παραμέτρους που θα τους αντιστοιχούνε. Το inventory αυτό θα πρέπει να παράγεται προγραμματιστικά από το λογισμικό του controller και να προσδιορίζει όλες τις απαιτούμενες παραμέτρους ως μεταβλητές προς το Ansible, ώστε να μην απαιτείται παρέμβαση στο ίδιο το playbook.

Αρχικά, θα κατασκευάζεται το αρχείο **hosts**, στο οποίο θα προσδιορίζονται και θα ομαδοποιούνται οι δρομολογητές πάνω στους οποίους θα πρέπει να δράσει το Ansible.

```
# inventory/hosts

[routers_vyos]
147.102.39.30
147.102.39.31

[routers_cisco]
147.102.39.32
```

Όπως φαίνεται στο παραπάνω παράδειγμα, οι ομάδες από hosts θα χρησιμοποιούνται για τον διαχωρισμό των δρομολογητών, βάσει του λειτουργικού τους συστήματος, επιτρέποντας, έτσι, στην εφαρμογή να υποστηρίζει πολλαπλά είδη δρομολογητών ταυτόχρονα, ορίζοντας ξεχωριστό play για το καθένα. Έπειτα, εάν αυτό απαιτείται, θα ορίζονται μεταβλητές κάποιας συγκεκριμένης ομάδας, δημιουργώντας το αντίστοιχο αρχείο στον φάκελο **group_vars**. Στα πλαίσια της ανάπτυξης αυτής, οι δρομολογητές θα έχουν το **VyOS** για λειτουργικό σύστημα, το οποίο απαιτεί τις παρακάτω παραμέτρους στο αντίστοιχο αρχείο:

```
# inventory/group_vars/routers_vyos.yml

# Specify the user to login as
ansible_user: vyos

# Necessary to support the shell that VyOS runs
ansible_connection: network_cli
ansible_network_os: vyos
```

Οι τελευταίες δύο παράμετροι είναι απαραίτητες για να λειτουργήσει το Ansible, καθώς και το περιβάλλον του VyOS δεν υποστηρίζει ρητές εκτελέσεις εντολών μέσω SSH και η συνδεσιμότητα πρέπει να πραγματοποιείται με streams. Το Ansible υποστηρίζει αυτού του είδους συνδέσεις, οπότε απαιτείται μόνο να αναφέρεται ρητά στις σχετικές παραμέτρους. Στην πράξη, το παραπάνω αρχείο δεν είναι απαραίτητο να δημιουργείται προγραμματιστικά και αρκεί να υπάρχει στο περιβάλλον του controller και να αντιγράφεται στο inventory που δημιουργείται. Τέλος, μένει να παραχθούν οι παράμετροι, που προσδιορίζουν τη συνδεσιμότητα μεταξύ των κόμβων, ώστε να χρησιμοποιηθούν από τον Ansible. Είναι εμφανές ότι, αφού οι παράμετροι αυτοί περιέχουν πληροφορίες, όπως διευθύνσεις σε εικονικά interfaces και στατικές διαδρομές, ορίζονται ξεχωριστά για κάθε δρομολογητή. Για το σκοπό αυτό, θα δημιουργείται ένα αρχείο μεταβλητών για κάθε έναν από αυτούς στον φάκελο **host_vars**. Παρακάτω φαίνεται ένα χαρακτηριστικό παράδειγμα του πως θα δομηθούν τα αρχεία αυτά:

```yaml
# inventory/host_vars/147.102.39.30.yml

hostname: router1

tunnels:
  - interface: tun0
    addr: 10.0.0.1
    route: 10.0.0.2
    remote: 147.102.39.31

static_routes:
  - dest: 10.0.0.3
    next_hop: 10.0.0.2
  - dest: 10.0.0.4
    next_hop: 10.0.0.2
```

Στο παραπάνω παράδειγμα, ορίζεται ότι ο δρομολογητής με διεύθυνση **147.102.39.30** θα λάβει τις ακόλουθες παραμετροποιήσεις:

- Αλλαγή του hostname του σε "**router1**".
- Δημιουργία ενός tunnel interface με όνομα **tun0** μεταξύ αυτού και του δρομολογητή με δημόσια διεύθυνση **147.102.39.31**.
- Απόδοση της διεύθυνσης **10.0.0.1** στο interface **tun0**.
- Δήλωση ότι ο γειτονικός του δρομολογητής στο **tun0** έχει τη διεύθυνση **10.0.0.2**, για τη δημιουργία στατικής διαδρομής προς αυτόν μέσω του **tun0**.
- Προσθήκη στατικής διαδρομής προς τον **10.0.0.3,** με επόμενο βήμα τη διεύθυνση **10.0.0.2**.
- Προσθήκη στατικής διαδρομής προς τον **10.0.0.4,** με επόμενο βήμα τη διεύθυνση **10.0.0.2**.

4.3.3 Υλοποίηση του Playbook

Έχοντας κατασκευάσει το inventory, επόμενο βήμα, κατά την ανάπτυξη της ενορχήστρωσης του δικτύου μέσω Ansible, είναι η σχεδίαση του playbook. Όπως προαναφέρθηκε, τα playbooks θα θεωρούνται αρχεία παραμετροποίησης της εφαρμογής και θα χρησιμοποιούνται για την εύκολη επέκταση της εφαρμογής με επιπλέον λειτουργικά συστήματα δρομολογητών. Το inventory δημιουργήθηκε με τέτοιο τρόπο, ώστε να περιέχει την απαραίτητη πληροφορία σε μεταβλητές ανά κόμβο (host_vars) για να την αναπαραγωγή της επιθυμητής συνδεσιμότητας, χωρίς μέριμνα για το είδος του συγκεκριμένου κόμβου. Η abstract υλοποίηση αυτή επιτρέπει την προσθήκη νέων playbook, ή νέων plays στα υπάρχοντα playbook στο περιβάλλον των αρχείων της εφαρμογής, χωρίς να απαιτείται παρέμβαση στον κώδικα που παράγει το inventory. Οπότε, στην ενότητα αυτή, θα περιγραφεί η υλοποίηση του πρωτότυπου playbook της εφαρμογής, το οποίο θα περιέχει **ένα play, μέσω του οποίου θα ρυθμίζονται οι δρομολογητές που λειτουργούν με VyOS**.

Αρχικά, ορίζεται η βασική δομή του **play**, στην οποία αποδίδεται κάποιο όνομα (name) στο play και προσδιορίζεται η ομάδα κόμβων, στην οποία θα ενεργεί. Η λειτουργία **gather_facts** είναι απαραίτητο να απενεργοποιηθεί, όπως φαίνεται παρακάτω, διότι από τις προσπάθειες που εκτελέστηκαν παρουσιάστηκε ασυμβατότητα με το λειτουργικό σύστημα του VyOS.

```
- name: "Initialization of VyOS routers"

  hosts: routers_vyos

  gather_facts: no

  tasks:

  - ...
```

Ύστερα, θα οριστούν τα **tasks** αυτού του play, τα οποία θα χρησιμοποιούνε τα διαθέσιμα modules του Ansible για το VyOS, σε συνδυασμό με τις μεταβλητές, που ορίζονται στο inventory. Πρώτα, έχουμε το task, το οποίο θα χρησιμοποιήσει το **vyos_system** module, για να ενημερώσει το hostname του δρομολογητή.

```
  - name: "setting router hostnames"
    vyos_system:
      host_name: "{{ hostname }}"
```

Όπως είναι εμφανές, το task αυτό ορίζει ένα όνομα (name), το οποίο θα εμφανίζεται στα διαγνωστικά μηνύματα κατά την εκτέλεση του playbook, καθώς και την κλήση του module, με τις παραμέτρους προς αυτό να ορίζονται ως αντικείμενο πάνω στο κλειδί (key) με το όνομα του module. Ύστερα, θα προστεθεί ένα task, το οποίο θα αφαιρεί από τον δρομολογητή όλα τα tunnel interfaces που ορίζονται στις παραμέτρους του αλλά υπάρχουν ήδη. Αυτό το βήμα συμπεριλαμβάνεται για να διατηρεί το idempotence του playbook, δηλαδή ότι επαναλαμβανόμενες εκτελέσεις του playbook θα οδηγούν το σύστημα στην ίδια ακριβώς κατάσταση. Σύμφωνα με τη δομή του αρχείου μεταβλητών των δρομολογητών, που ορίστηκε στην προηγούμενη ενότητα, τα tunnel interface του δρομολογητή περιέχονται ως λίστα από αντικείμενα. Οπότε, για την ανάγνωσή τους, θα χρησιμοποιηθεί η δυνατότητα του Ansible για ορισμό βρόχων επαναλήψεων στο playbook.

```
  - name: "clearing any existing tunnels"
    loop: "{{ tunnels }}"
    ignore_errors: yes
    vyos_config:
      lines:
        - delete interfaces tunnel "{{ item.interface }}"
```

Όπως φαίνεται παραπάνω, μέσω της λέξης-κλειδί **loop,** ορίζουμε ότι το task αυτό πρέπει να εκτελεστεί επαναλαμβανόμενα για κάθε αντικείμενο στη λίστα της μεταβλητής **tunnels**. Κάθε αντικείμενο γίνεται ορατό στο περιβάλλον του task μέσω της μεταβλητής **item**. Χρησιμοποιώντας, λοιπόν, το module **vyos_config**, το οποίο είναι υπεύθυνο για την εκτέλεση εντολών παραμετροποίησης στο σύστημα, εκτελείται η εντολή “delete interfaces” μία φορά για κάθε όνομα interface στη λίστα, το οποίο, στο σημείο αυτό, βρίσκεται στην

μεταβλητή **item.interface**. Επιπλέον, εισάγοντας τη γραμμή “ignore_errors: yes”, καθοδηγούμε το Ansible να μην διακόψει την εκτέλεση του task ή του playbook, στην περίπτωση που η έξοδος κάποιας εντολής του task αυτού υποδηλώνει αποτυχία ή σφάλμα, καθώς, έτσι, επιτρέπουμε στο task αυτό να λειτουργεί το ίδιο και όταν δεν υπάρχει κάποιο από τα interface που επιχειρεί να αφαιρέσει.

Το επόμενο task θα εκτελεί πάλι επανάληψη (loop) πάνω στα ορισμένα tunnel interfaces, αυτή τη φορά δημιουργώντας τα. Οι εντολές στο task αυτό είναι οι ίδιες με αυτές που αναφέρθηκαν στα πλαίσια της ενότητας **4.2.2** περί παραμετροποίησης του VyOS.

```
  - name: "configuring interface tunnels"
    loop: "{{ tunnels }}"
    vyos_config:
      save: yes
      lines:
        - set interfaces tunnel "{{ item.interface }}" encapsulation gre
        - set interfaces tunnel "{{ item.interface }}" local-ip "{{ inventory_hostname }}"
        - set interfaces tunnel "{{ item.interface }}" remote-ip "{{ item.remote }}"
        - set interfaces tunnel "{{ item.interface }}" address "{{ item.addr }}/32"
        - set protocols static interface-route "{{ item.route }}/32" next-hop-interface "{{
item.interface }}"
```

Καινούργιες αναφορές στο task αυτό είναι η παράμετρος **save**, που καθοδηγεί το module να γράψει τη νέα παραμετροποίηση στο δίσκο με χρήση της εντολής save, καθώς και η μεταβλητή **inventory_hostname**, η οποία περιέχει τη διεύθυνση του κόμβου, μέσω της οποίας είναι προσβάσιμος, όπως αυτή αναφέρεται στο αρχείο **hosts**. Συλλέγοντας τα παραπάνω tasks, προκύπτει το τελικό playbook που φαίνεται παρακάτω:

```
# inventory/playbook.yml

- name: "Initialization of VyOS routers"

  hosts: routers_vyos

  gather_facts: no

  tasks:

  - name: "setting router hostnames"
    vyos_system:
      host_name: "{{ hostname }}"

  - name: "clearing any existing tunnels"
    loop: "{{ tunnels }}"
    ignore_errors: yes
    vyos_config:
      lines:
        - delete interfaces tunnel "{{ item.interface }}"
```

```
  - name: "configuring interface tunnels"
    loop: "{{ tunnels }}"
    vyos_config:
      save: yes
      lines:
        - set interfaces tunnel "{{ item.interface }}" encapsulation gre
        - set interfaces tunnel "{{ item.interface }}" local-ip "{{ inventory_hostname
}}"
        - set interfaces tunnel "{{ item.interface }}" remote-ip "{{ item.remote }}"
        - set interfaces tunnel "{{ item.interface }}" address "{{ item.addr }}/32"
        - set protocols static interface-route "{{ item.route }}/32" next-hop-interface
"{{ item.interface }}"

  - name: "configuring static routes"
    loop: "{{ static_routes }}"
    vyos_static_route:
      prefix: "{{ item.dest }}"
      mask: 32
      next_hop: "{{ item.next_hop }}"
```

Κεφάλαιο 5

Τεχνική Ανάπτυξη της Εφαρμογής

5.1 Δημιουργία Εικονικών Μηχανών στο ESXi

Στο πλαίσιο αυτής της ενότητας, θα υποθέσουμε ότι υπάρχει το **vmx** αρχείο κάποιας εικονικής μηχανής. Το αρχείο αυτό περιέχει παραμέτρους της εικονικής μηχανής προς τον hypervisor, όπως το πόση RAM και επεξεργαστές πρέπει να δεσμευτούν για το συγκεκριμένο VM, τις συνδεδεμένες περιφερικές συσκευές ή την επιλεγμένη διεύθυνση MAC για την εικονική κάρτα δικτύου. Το αρχείο αυτό, δηλαδή, περιγράφει την εικονική μηχανή και μπορεί να έχει προκύψει από κάποιο άλλο πρόγραμμα εικονικοποίησης όπως το VirtualBox. Όλες οι εντολές, που θα αναφερθούν παρακάτω, εκτελούνται στο shell του ESXi με τον χρήστη **root**. Επίσης, θα περιέχεται και η έξοδος (output) των εντολών, όπου είναι σχετικό.

Για να αρχικοποιηθεί μια εικονική μηχανή, πρέπει να γίνει εγγραφή (register) του vmx αρχείου μέσω του ESXi shell και, συγκεκριμένα, με την παρακάτω εντολή:

```
$ vim-cmd solo/registervm /vmfs/volumes/datastore1/vyos/vyos.vmx
24
```

Όπου ο κατάλογος (directory) "**/vmfs/volumes/datastore1"** είναι ένα Datastore του συγκεκριμένου ESXi host, που λειτουργεί ως ο αποθηκευτικός του χώρος.

Η έξοδος της παραπάνω εντολής υποδηλώνει ότι το VM έγινε register στον hypervisor και ότι του αποδόθηκε για ID ο αριθμός **24**. Για επαλήθευση, μπορούμε να εκτελέσουμε την παρακάτω εντολή, της οποίας η έξοδος περιέχει κάποιες πληροφορίες για όλα τα εγγεγραμμένα VM:

```
$ vim-cmd vmsvc/getallvms
Vmid   Name              File                 Guest OS       Version
Annotation
24     vyos   [datastore1] vyos/vyos.vmx   debian6_64Guest   vmx-11
```

Για την εκκίνηση ή την απενεργοποίηση του VM, χρησιμοποιούνται οι δύο παρακάτω εντολές:

```
$ vim-cmd vmsvc/power.on 24
```

```
$ vim-cmd vmsvc/power.off 24
```

Για ανάκτηση της κατάστασης του VM χρησιμοποιείται η εντολή **vmsvc/power.getstate**.

```
$ vim-cmd vmsvc/power.getstate 24
Retrieved runtime info
Powered off
```

Για την αφαίρεση ενός VM από τον hypervisor, χρησιμοποιείται η εντολή **vmsvc/unregister,** όπως φαίνεται παρακάτω:

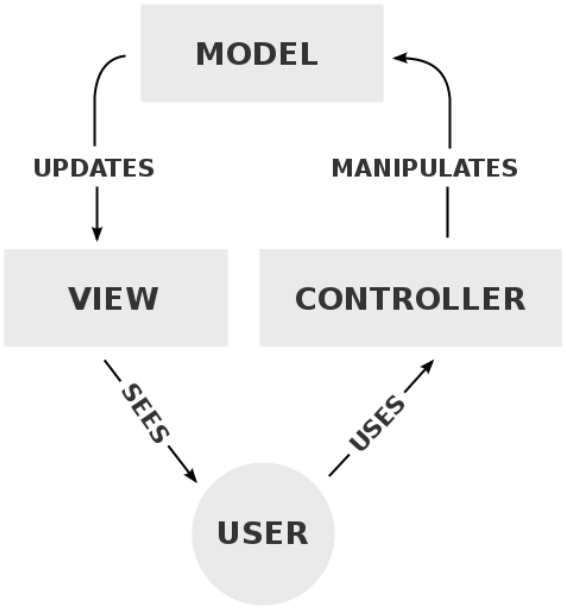
```
$ vim-cmd vmsvc/unregister 24
```

Αφαίρεση (unregister) μιας εικονικής μηχανής δεν διαγράφει τα αρχεία της.

5.2 Εφαρμογή του Μοντέλου MVC

Όπως έχει προαναφερθεί στην εργασία, η υλοποίηση του λογισμικού της ανάπτυξης θα ακολουθήσει τη δομή πελάτη-διακομιστή, με το front-end της εφαρμογής να είναι ιστοσελίδα προσβάσιμη μέσω web browser. Επίσης, η επικοινωνία του λογισμικού του client στο περιβάλλον του πελάτη με τον server θα πραγματοποιείται με τη χρήση ερωτημάτων (requests) και απαντήσεων (responses) του πρωτοκόλλου HTTP. Στην πράξη, αυτή η μέθοδος επικοινωνίας μπορεί να γεφυρώσει δύο αναπτύξεις λογισμικού, που χρησιμοποιούν διαφορετικές τεχνικές δομές και γλώσσες. Η δυνατότητα αυτή επιτρέπει θεωρητικά την ανάπτυξη του front-end και του back-end ως δύο ξεχωριστά έργα λογισμικού. Μία τέτοια προσέγγιση συνοδεύεται από το κατάλληλο επίπεδο abstraction, που επιτρέπει την επαναχρησιμοποίηση των διακριτών υποέργων ή την ενσωμάτωση τμημάτων της λειτουργικότητάς τους σε τρίτα έργα. Για παράδειγμα, το HTTP API του back-end μπορεί να χρησιμοποιηθεί ταυτόσημα από μία ιστοσελίδα, μία εφαρμογή για κινητές συσκευές ή από ένα εργαλείο της γραμμής εντολών (command-line interface, CLI).

Για την συγκεκριμένη ανάπτυξη, όμως, θα χρησιμοποιηθεί το μοντέλο MVC. Το **Model-View-Controller** είναι ένα αρχιτεκτονικό πρότυπο, που χρησιμοποιείται συχνά κατά την ανάπτυξη διεπαφών χρήστη (user interfaces). Το πρότυπο αυτό δεν συνοδεύεται από τεχνικές προδιαγραφές για το λογισμικό, είναι ένας θεωρητικός τρόπος σχεδίασης τέτοιων εφαρμογών, ο οποίος διαιρεί την εφαρμογή σε τρία διακριτά και διασυνδεμένα μέλη. Η κύρια ιδέα είναι ο διαχωρισμός και η διαχείριση της εσωτερικής μοντελοποίησης της πληροφορίας από τον τρόπο με τον οποίο η πληροφορία αυτή συλλέγεται και απεικονίζεται στον χρήστη.



9. Μοντέλο Model-View-Controller (MVC)

- **Model**
  Το μοντέλο είναι το κεντρικό συστατικό της δομής της πληροφορίας, που χειρίζεται από το back-end της εφαρμογής. Στο λογισμικό σχεδιάζεται έτσι, ώστε να λειτουργεί ως αντικειμενοστραφής απεικόνιση των δεδομένων, επιτρέποντας, με τον τρόπο αυτό, τη χρήση, καθώς και την ανταλλαγή τους, μεταξύ του client και του server, σε υψηλό επίπεδο με επίγνωση της μοντελοποίησης.

- **View**
  Η όψη είναι η αναπαράσταση της δομημένης πληροφορίας στο χρήστη. Συνήθως, κατασκευάζεται χρησιμοποιώντας τα μοντέλα ως συστατικά και απεικονίζεται στον χρήστη.

- **Controller**
  Ο ελεγκτής, που συνήθως εκτελείται στο back-end της εφαρμογής, είναι το κομμάτι της εφαρμογής, που είναι υπεύθυνο για να δέχεται τα ερωτήματα (requests) του χρήστη και να συντάσσει τις κατάλληλες απαντήσεις (responses), ανακτώντας τα μοντέλα (models) και συντάσσοντας τις όψεις (views).

5.3 Επιλογή Τεχνολογικής Στοίβας

Τα περισσότερα περίπλοκα έργα λογισμικού πολλών επιπέδων δεν είναι δυνατόν να αναπτυχθούν πλήρως με τη χρήση μίας μόνο γλώσσας ή ενός μόνο περιβάλλοντος. Με τον όρο της τεχνολογικής στοίβας (**Tech Stack**), αναφερόμαστε στα επιμέρους τμήματα ενός έργου λογισμικού, τα οποία είναι δομημένα πάνω σε γλώσσες και τεχνολογίες, που διαφέρουν μεταξύ τους. Τα επίπεδα της στοίβας περιέχουν τις τεχνολογίες που χρησιμοποιούνται στα διάφορα τμήματα και στάδια της ανάπτυξης, καθώς και τις μεθόδους με τις οποίες επιτυγχάνεται η διασύνδεση μεταξύ τους. Στην ενότητα αυτή, θα γίνει η ανάλυση της επιλογής των επιπέδων αυτής της στοίβας, που, λόγω του ότι βασίζεται κυρίως σε προσωπικές προτιμήσεις, θα διατηρηθεί σχετικά σύντομη.

- **ASP.NET Core MVC**
  - **C#, .NET**
  - **Razor HTML**

  Για την υλοποίηση του back-end της εφαρμογής, θα χρησιμοποιηθεί η MVC δομή του ASP.NET Core της Microsoft. Το .NET framework της Microsoft κάποτε συνοδευόταν από πολύ περιοριστικές προδιαγραφές και περιορισμούς ανάπτυξης, που απαιτούσαν πολύ συγκεκριμένο περιβάλλον Microsoft τόσο για την ανάπτυξη όσο και για τη χρήση του λογισμικού. Με την εισαγωγή του .NET Core, όμως, η δυνατότητα ανάπτυξης τέτοιων εφαρμογών έχει γίνει συμβατή με άλλα λειτουργικά συστήματα, όπως το macOS και τις περισσότερες διανομές Linux.

- **HTML**
  Κατασκευάζοντας ιστοσελίδες για το διαδίκτυο θεωρητικά δεν υπάρχουν εναλλακτικές της χρήσης της HTML. Οι σελίδες HTML, στα πλαίσια της ανάπτυξης, θα περιγράφονται ως views στο ASP.NET MVC και θα κατασκευάζονται με τη βοήθεια της σύνταξης **Razor,** που παρέχεται από αυτό.

- **Less.js**
  Πάλι λόγω των σχετικών προτύπων, οι web browsers κατά κανόνα υποστηρίζουνε τη μορφοποίηση μιας HTML σελίδας μόνο με χρήση **CSS**. Για την ανάπτυξη της λειτουργικότητας αυτής, στη συγκεκριμένη εφαρμογή, θα χρησιμοποιηθεί η Less. Η γλώσσα αυτή χρησιμοποιεί το συντακτικό της CSS, επεκτείνοντάς το παράλληλα με δυνατότητες, που επιτρέπουν τον πιο ευέλικτο και αποτελεσματικό κώδικα μορφοποίησης (styling). Κώδικας Less γράφεται σε αρχεία με επέκταση "**.less**" και γίνεται compile σε κανονική CSS, ώστε να είναι συμβατός με κάθε web browser.

- **TypeScript**
  Για παρόμοιους λόγους συμβατότητας με τα πρότυπα του διαδικτύου, ο μόνος επιτρεπτός τρόπος προγραμματισμού της λειτουργικότητας μιας ιστοσελίδας είναι ο κώδικας **Javascript**. Με παρόμοια προσέγγιση με τη Less, θα χρησιμοποιηθεί η γλώσσα TypeScript για το σκοπό αυτό. Η γλώσσα αυτή επεκτείνει το συντακτικό της Javascript, προσθέτοντας αντικειμενοστραφείς ιδιότητες, σύστημα τμηματοποίησης του κώδικα σε modules και αυστηρό σύστημα τύπων, που επιβάλλεται από τον compiler της. Κώδικας TypeScript γράφεται σε αρχεία με επέκταση "**.ts**" και γίνεται compile σε κανονική Javascript, ώστε να είναι συμβατός με κάθε web browser.

5.3.1 Frameworks και Βιβλιοθήκες

Στην ενότητα αυτή, θα αναφερθούν οι βιβλιοθήκες που θα χρησιμοποιηθούνε στα πλαίσια της ανάπτυξης, μαζί με τη συγκεκριμένη έκδοση που εγκαταστάθηκε. Όλες είναι έργα ανοιχτού κώδικα και παρέχουν την κατάλληλη άδεια για τη χρήση τους από την εργασία αυτή.

Front-end

- **Bootstrap** – v4.1.1 - https://getbootstrap.com/docs/4.0/getting-started/introduction/

Βιβλιοθήκη που παρέχει κλάσεις CSS, που διευκολύνουν σημαντικά την διάταξη στοιχείων σε μία HTML σελίδα.

- **jQuery** – v3.3.1 - https://jquery.com/
  Javascript βιβλιοθήκη, που παρέχει μεθόδους για την επεξεργασία των στοιχείων μίας HTML σελίδας.

- **Font Awesome –** Free v5.0.13 - https://fontawesome.com/
  Συλλογή από χρήσιμα εικονίδια σε μορφή κλιμακώσιμων γραφικών vector, για απεικόνιση στο γραφικό περιβάλλον του χρήστη.

- **Highlight.js** – v9.12.0 - https://highlightjs.org/
  Μορφοποίηση κώδικα σε μία HTML σελίδα. Χρησιμοποιείται για την καλύτερη απεικόνιση δεδομένων στο χρήστη, που βρίσκονται σε μορφή YAML ή JSON.

- **js-yaml** – v3.11.0 - https://github.com/nodeca/js-yaml
  Βιβλιοθήκη Javascript για την μετατροπή YAML κειμένου σε JSON και το αντίστροφο.

- **Pixi.js** – v4.7.3 - http://www.pixijs.com/
  Το κύριο framework, που θα αναλάβει τη σχεδίαση, την απεικόνιση και τη λειτουργία του σχεδιαστικού περιβάλλοντος του χρήστη.

Back-end

- **YamlDotNet** – v4.3.1 - https://github.com/aaubry/YamlDotNet/wiki
  Βιβλιοθήκη για την σειριοποίηση (serialization) και αποσειριοποίηση (deserialization) μεταξύ κειμένου YAML και αντικειμένων της C#.

- **SSH.NET** – v2016.1.0 - https://github.com/sshnet/SSH.NET/
  Secure Shell (SSH) client για C#.

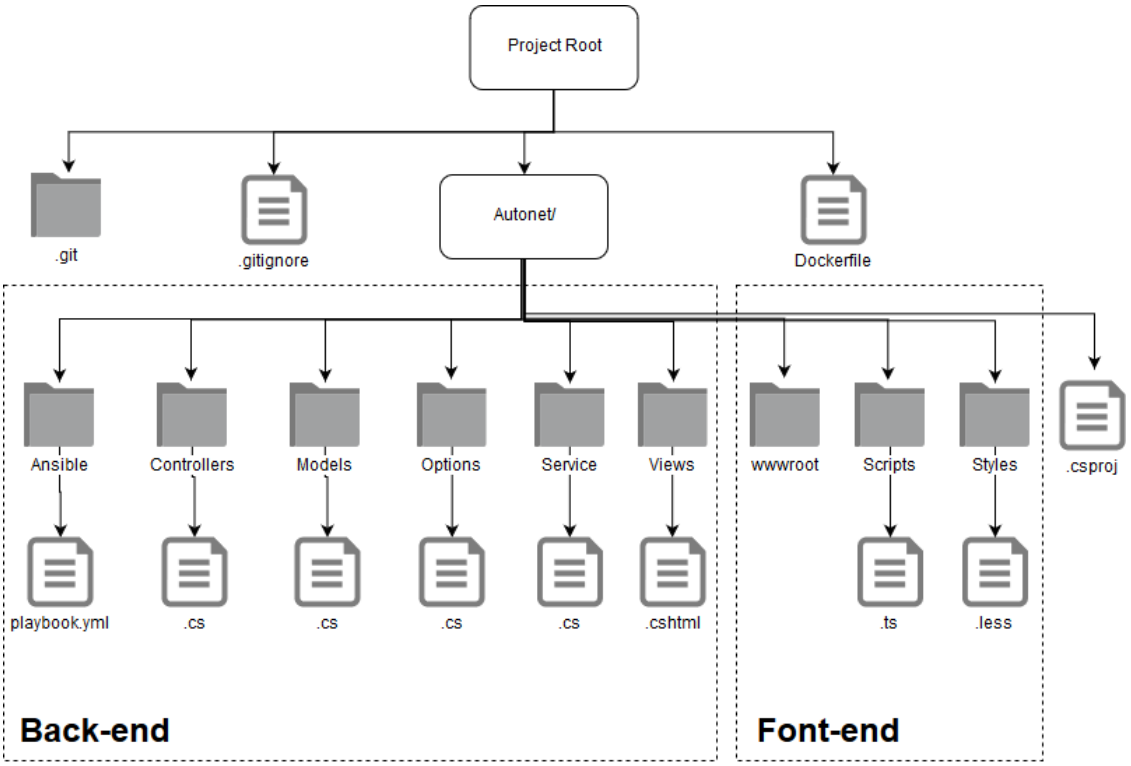
- **Json.NET** – v10.0.1 - https://www.newtonsoft.com/json
  Βιβλιοθήκη για την σειριοποίηση (serialization) και αποσειριοποίηση (deserialization) μεταξύ κειμένου JSON και αντικειμένων της C#.

- **ansible-net** – v1.0.2 - https://github.com/xplicit/ansible-net
  Περικάλυμμα εκτελέσεων εντολών του Ansible στη γραμμή εντολών μέσω δημιουργία διαδικασιών από τη C#.

- **Node.js** – v8.11.2 - https://nodejs.org/en/
  Javascript framework, το οποίο, στα πλαίσια του έργου, θα χρησιμοποιηθεί για το χτίσιμο του λογισμικού και διαχείριση εξαρτήσεων (dependency management).

5.3.2 Δομή Αρχείων της Εργασίας

Στην ενότητα αυτή, θα δοθεί μία σύντομη περιγραφή της δομής των αρχείων και των φακέλων της ανάπτυξης, καθώς και κάποιων από τους ρόλους τους.



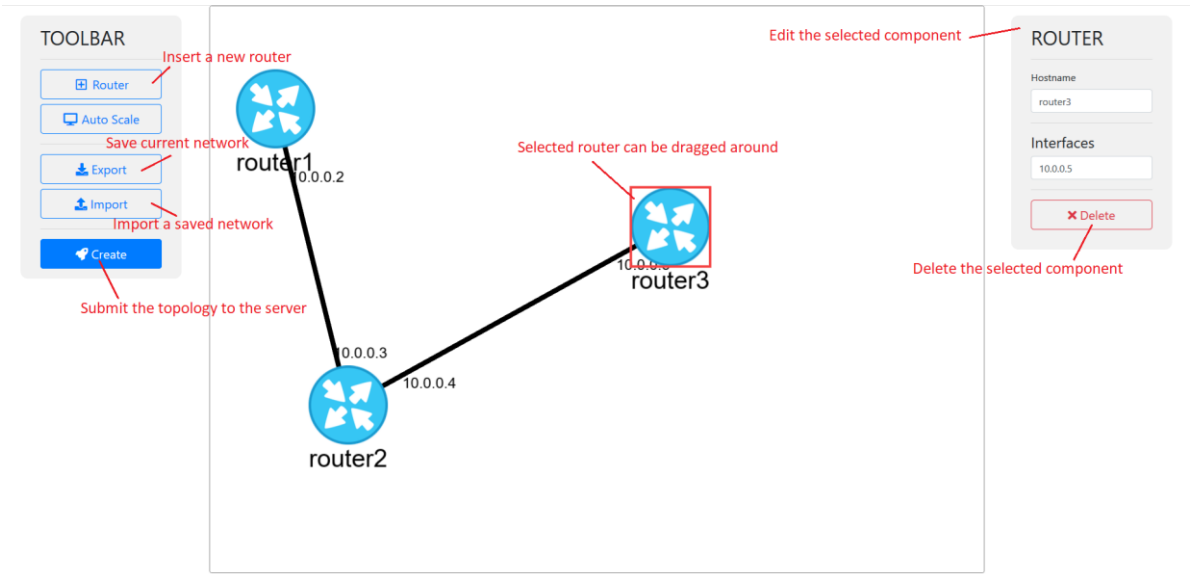
10. Δομή Αρχείων της Εργασίας

- **Ansible/**
  Ο φάκελος αυτός θα περιέχει αρχεία σχετικά με το Ansible, όπως τα playbook και τα αρχεία παραμετροποίησης.
- **Options/**
  Κώδικας και μοντέλα σχετικά με την παραμετροποίηση της εφαρμογής
- **Service/**
  Κώδικας που περιέχει την κύρια λειτουργικότητα της εφαρμογής.
- **wwwroot/**
  Στο ASP.NET, ο φάκελος αυτός χρησιμοποιείται για να κρατάει και να σερβίρει στατικά αρχεία περιεχομένου, όπως εικόνες και αρχεία Javascript και CSS.
- **Scripts/**
  Κώδικας TypeScript.
- **Styles/**
  Κώδικας Less.
- **.csproj**
  Το αρχείο παραμέτρων ενός έργου (project) στο Visual Studio.

5.4 Ανάπτυξη Γραφικού Περιβάλλοντος (Front-end)

5.4.1 Οθόνη Σχεδίασης Τοπολογίας

Το γραφικό περιβάλλον της εφαρμογής, μέσω του οποίου ο οποίου ο χρήστης θα σχεδιάζει το επιθυμητό εικονικό ιδιωτικό δίκτυο, πρέπει να είναι λιτό και κατανοητό στην όψη του. Θα χρησιμοποιηθεί μία απλή διάταξη τριών στηλών, με το κύριο σχεδιαστικό περιβάλλον στη μέση και δύο βοηθητικά μενού στις δύο πλευρές του.



11. Οθόνη Σχεδίασης Τοπολογίας

Η παραπάνω διάταξη υλοποιείται με ευκολία εκμεταλλεύοντας τη διάταξη σε πλέγμα (grid layout) της Bootstrap. Η δομή του αντίστοιχου αρχείου φαίνεται παρακάτω:

```
<!-- Views/Home/Index.cshtml -->
<div class="container-fluid body-content">
    <div class="row">
        <div class="col-md-2">
            <div class="container-fluid">
                <div class="sidebar sidebar-left">
                      ...
                </div>
            </div>
        </div>

        <div class="col-md-8">
            <canvas id="designer-canvas"></canvas>
        </div>

        <div class="col-md-2">
            <div class="container-fluid">
                <div class="sidebar sidebar-right properties" id="router-properties" >
                      ...
                </div>
            </div>
        </div>
    </div>
</div>
```

Ο κόμβος τύπου **canvas** στη μεσαία στήλη λειτουργεί ως η όψη, πάνω στην οποία εγκαθίσταται το αντικείμενο **Application** της βιβλιοθήκης **Pixi.js**. Χρησιμοποιώντας το canvas αυτό ως παράμετρο, αρχικοποιείται η κλάση **Designer**, η οποία θα είναι η κύρια κλάση αυτής της σελίδας. Στην ενότητα αυτή, θα επεξηγηθούνε τα Typescript αρχεία, από τα οποία αποτελείται το λειτουργικό κομμάτι της σελίδας αυτής.

- **Views/_ViewStart.cshtml**
  Από τη δομή του ASP.NET, το αρχείο αυτό περιέχει την πρότυπη μορφή (template) της εφαρμογής. Το template αυτό χρησιμοποιείται για κάθε σελίδα, εμφανίζοντας το κατάλληλο view μέσα στο template.

- **Scripts/app.ts**
  Το σημείο εισαγωγής στην εφαρμογή. Χρησιμοποιώντας event handler, που ενεργοποιείται όταν φορτώσει πλήρως η σελίδα, ο κώδικας στο αρχείο αυτό ανακτά το στοιχείο canvas από τη σελίδα με χρήση jQuery και αρχικοποιεί τον designer.
  Επίσης, το αρχείο αυτό δηλώνει event handlers, που χειρίζονται το πάτημα των κουμπιών των μενού και που επιβάλλουν κανόνες μορφής του πεδίου, που ο χρήστης ορίζει hostname.

- **Scripts/dataPort.ts**
  Το αρχείο αυτό περιέχει τις συναρτήσεις που επιτελούν την εξαγωγή (export) και εισαγωγή (import) από και προς την εφαρμογή αντίστοιχα, σε μορφή κειμένου σε μορφή YAML ή JSON.

- **Scripts/options.ts**
  Περιέχει σταθερές (constants) με τις ρυθμίσεις της εφαρμογής.

- **Scripts/designer/designer.ts**
  Η κύρια κλάση της εφαρμογής. Διατηρεί εφαρμογή στο **PIXI.Application** της βιβλιοθήκης Pixi.js, χειρίζεται το κύριο **PIXI.Container** της ίδιας βιβλιοθήκης, στο οποίο προσθέτει όλα τα components που προστίθενται στο δίκτυο, και, μέσω event handlers για το input από το ποντίκι του χρήστη, πραγματοποιεί τη μεγέθυνση και σμίκρυνση (zoom) του παραθύρου καθώς.

- **Scripts/designer/input.ts**
  Περιέχει βοηθητικές συναρτήσεις για το χειρισμό input του χρήστη.

- **Scripts/designer/line.ts**
  Περιέχει την κλάση **Line**, η οποία κληρονομεί ιδιότητες PIXI.Graphics, ώστε να ζωγραφίζεται στην οθόνη μεταξύ δύο σημείων. Η μέθοδος **update()** της κλάσης αυτής πρέπει να καλείται κάθε φορά που ένα από τα άκρα της μετακινείται, ώστε να ζωγραφίζεται ξανά.

- **Scripts/designer/properties.ts**
  Συναρτήσεις που επιτρέπουν την επεξεργασία ενός δρομολογητή ή μίας ζεύξης, εμφανίζοντας το δεξί μενού στην οθόνη, ενημερώνοντας τα πεδία σε αυτό με τις λεπτομέρειες του επιλεγμένου αντικειμένου και ορίζοντας event handlers τα πεδία του μενού για την ενημέρωσή τους.

- **Scripts/models/draggable.ts**
  Κλάση που κληρονομεί τις ιδιότητας ενός PIXI.Container. Αντικείμενο που προστίθεται στο container **Draggable** γίνεται αυτόματα μετακινήσιμο με το ποντίκι στον σχεδιαστή. Χρησιμοποιεί event handlers για input από το ποντίκι.

- **Scripts/models/selectable.ts**
  Περιέχει το Interface **Selectable**. Υλοποίηση του interface αυτού γίνεται από αντικείμενα που είναι επιλέξιμα στην οθόνη.

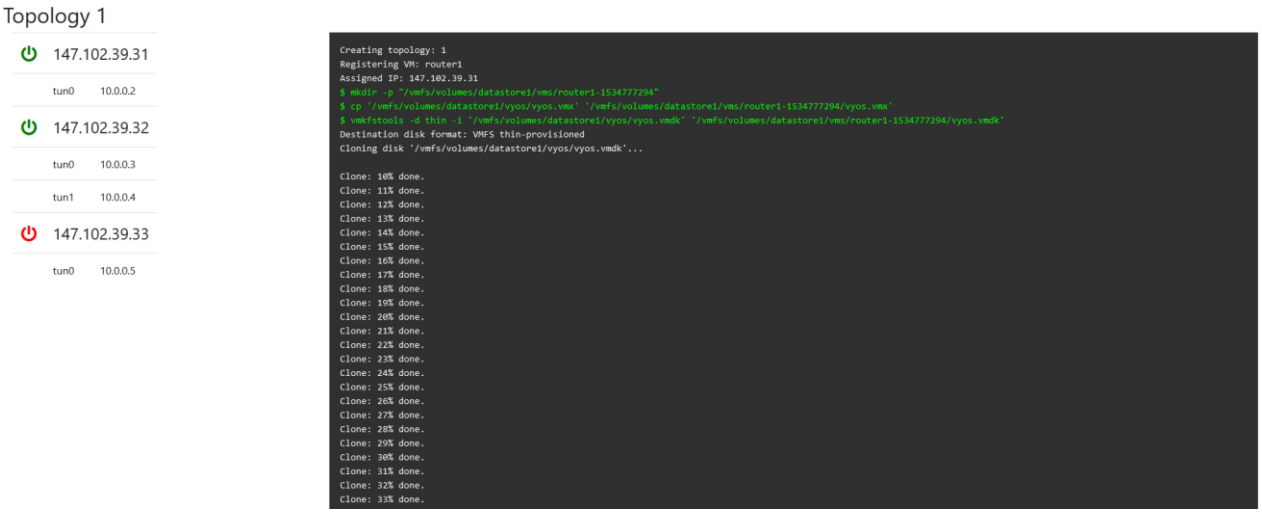
- **Scripts/models/component.ts**
  Η κλάση **Component** είναι ένα αντικείμενο προς απεικόνιση στον σχεδιαστή. Είναι Draggable και Selectable, κρατάει αναφορές σε ζεύξεις μεταξύ αυτού και άλλα components και αποτελείται από μία εικόνα και μία επιγραφή (label).

- **Scripts/models/interface.ts**
  Η κλάση **Interface** στο αρχείο αυτό είναι ένα βοηθητικό αντικείμενο που αντιπροσωπεύει την IPv4 διεύθυνση ενός δρομολογητή σε κάποια διεπαφή του.

- **Scripts/models/router.ts**
  Περιέχει την κλάση **Router**, που κληρονομεί ιδιώτητες Component και διατηρεί διεπαφες. Στιγμιότυπο της κλάσης αυτής προστίθεται στον Designer με το πάτημα του κουμπιού προσθήκης δρομολογητή.

5.4.2 Οθόνη Παρακολούθησης Τοπολογίας

Η δεύτερη οθόνη της εφαρμογής αυτής είναι το γραφικό περιβάλλον, στο οποίο ο χρήστης θα μπορεί να παρακολουθεί την εξέλιξη της εγκατάστασης του ιδιωτικού δικτύου, που υπέβαλε στον server. Χρησιμοποιώντας το κουμπί “Create” της προηγούμενης σελίδας, ο server θα απαντάει με το μοναδικό ID που ανέθεσε στην συγκεκριμένη τοπολογία. Ύστερα, ο client του χρήστη θα διευθύνει τον χρήστη στο URL “**/topology/{id}**”, στο οποίο θα μπορεί να παρακολουθεί διαγνωστικά μηνάνηματα από την εξέλιξη της τοπολογίας με το συγκεκριμένο ID σε πραγματικό χρόνο.



12. Οθόνη Παρακολούθησης Τοπολογίας

Η δομή της σελίδας αυτής είναι εξίσου απλή, με χρήση του πλέγματος Bootstrap.

```
<!-- Views/Home/Topology.cshtml -->

<h2>Topology @ViewBag.TopologyId</h2>

<input type="hidden" value="@ViewBag.TopologyId" id="topology-id" />

<div class="container-fluid">
    <div class="row h-100">
        <div class="col-md-3">
            <table id="router-status-table" class="table table-hover table-striped"></table>
        </div>
        <div class="col-md-9">
            <div class="log-container">
                <div id="output">

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

Όλος ο κώδικας της σελίδας αυτής περιέχεται στο αρχείο **topology.ts**. Το αρχείο αυτό χρησιμοποιεί event handler, που ενεργοποιείται όταν φορτώσει η σελίδα.

```
$(document).ready(function ()
{
    // Will run at page load
    setInterval(refreshLog, Options.LOG_CHECK_INTERVAL);
    setInterval(refreshStatus, Options.LOG_CHECK_INTERVAL);
});
```

Η **setInterval** είναι μία από τις βασικές συναρτήσεις της Javascript. Δέχεται ως πρώτο όρισμα μία συνάρτηση και ως δεύτερο έναν αριθμό, που αντιπροσωπεύει χρονική διάρκεια σε millisecond. Η setInterval, τότε, καλεί την συνάρτηση του πρώτου ορίσματος επαναλαμβανόμενα κάθε τόσα millisecond όσα ορίζονται στη δεύτερη παράμετρο. Έτσι, για παράδειγμα, ανάλογα με την τιμή που ορίζεται στο options.ts, στη σελίδα αυτή θα καλούνται, για παράδειγμα, οι παρακάτω δύο συναρτήσεις κάθε 400ms:

- **refreshLog**
  Η συνάρτηση αυτή εκτελεί ασύγχρονο ερώτημα στο server στο URL "/api/logs/{id}", για να ανακτήσει τις διαγνωστικές και πληροφοριακές γραμμές που προκύψανε από μηνύματα του server ή από εξόδους εντολών παραμετροποίησης. Η ίδια συνάρτηση δημιουργεί HTML γραμμές με το κατάλληλο χρώμα και τις προσθέτει στο div με id "output" στη δεξιά στήλη της σελίδας.

- **refreshStatus**
  Η συνάρτηση αυτή εκτελεί ασύγχρονο ερώτημα στο server στο URL "/api/topology/{id}", για να ανακτήσει πληροφορίες και την κατάσταση των εικονικών δρομολογητών του δικτύου καθώς αυτό δημιουργείται. Η ίδια συνάρτηση δημιουργεί HTML γραμμές με τις πληροφορίες αυτές και τις προσθέτει στο table με id "router-status-table".

5.4.3 Χτίσιμο του Γραφικού Περιβάλλοντος

Το χτίσιμο του κομματιού της εφαρμογής αποτελείται από το compile του κώδικα TypeScript, αλλά και των stylesheet σε Less, παράγοντας Javascript και CSS αρχεία αντίστοιχα, που είναι συμβατά με τον web browser του χρήστη. Τα αρχεία αυτά θα καταλήγουν στον φάκελο **wwwroot**, από τον οποίο, όπως έχει προαναφερθεί, θα σερβίρονται ως στατικά assets. Για την υλοποίηση αυτή, θα αναπτυχθεί μία διαδικασία χτισίματος (build pipeline), χρησιμοποιώντας το πακέτο **Gulp.js**. Το πακέτο αυτό χρησιμοποιείται για την περιγραφή ενός build pipeline σε μορφή κώδικα Javascript, καθώς και για την εκτέλεσή του. Το pipeline περιγράφεται σε αρχείο που ονομάζεται **gulpfile.js**, το οποίο εκτελείται από τη γραμμή εντολών με χρήση της εντολής **gulp**. Το αρχείο αυτό περιγράφει tasks, το κάθε ένα από τα οποία χρησιμοποιεί τη συνάρτηση **gulp.src(),** για να επιλέξει κάποια αρχεία από το έργο, και την συνάρτηση **pipe()** επαναλαμβανόμενα, για να περάσει τα αρχεία αυτά μέσα από διάφορες συναρτήσεις σειριακά, εκτελώντας πάνω τους όσες ενέργειες χρειάζεται. Για παράδειγμα, για το χτίσιμο του κώδικα Less ορίζεται το παρακάτω task, το οποίο ανοίγει όλα τα αρχεία με επέκταση **.less**, τα τροφοδοτεί στη

συνάρτηση **less()**, η οποία τα κάνει compile σε CSS, και τα αποθηκεύει στον φάκελο **wwwroot/css**.

```
gulp.task("compile:less", compileLess);
function compileLess()
{
    return gulp.src('./Styles/**/*.less')
               .pipe(less())
               .pipe(gulp.dest('./wwwroot/css/'));
}
```

Το παραπάνω task μπορεί να εκτελεστεί απομονωμένα με χρήση της παρακάτω εντολής.

```
$ gulp compile:less
```

Τα task μπορούν, επίσης, να ομαδοποιηθούν, επιτρέποντας, έτσι, την παράλληλη εκτέλεσή τους.

```
gulp.task("compile", ["compile:less", "compile:typescript"]);
```

Το χτίσιμο του κώδικα TypeScript είναι αρκετά πιο περίπλοκο. Όπως αναφέρθηκε στην ενότητα **5.3**, η γλώσσα αυτή λειτουργεί ορίζοντας modules, τα οποία εισάγουν (import) το ένα το άλλο. Ο web browser, όμως, δεν φέρει αυτή τη λειτουργικότητα, οπότε το χτίσιμο του κώδικα TypeScript θα πραγματοποιηθεί με χρήση του πακέτου **browserify**, το οποίο, χρησιμοποιώντας το εισαγωγικό αρχείο της εφαρμογής (entrypoint), μπορεί να πακετάρει αυτό με όσα άλλα αρχεία χρησιμοποιεί αναδρομικά σε ένα μοναδικό αρχείο Javascript, για χρήση στον browser. Το task με το pipeline, που το παράγει, φαίνεται παρακάτω:

```
function bundleTypescript(entrypoint, bundleName)
{
    return browserify()
        .add(entrypoint)                            // Add the entrypoint
        .plugin("tsify", tsConfig.compilerOptions)  // Load the TypeScript plugin
        .bundle()                                   // Compile
        .pipe(source(bundleName))                   // Get the bundle
        .pipe(buffer())                             // Transform it to a gulp pipe stream
        .pipe(babel({                               // Fix incompatible TS syntax
            presets: ['env']                        // with the older ES2015 standards
        }))
        .pipe(gulp.dest("./wwwroot/js/"));          // Write the bundle
}
```

5.5 Ανάπτυξη Εφαρμογής Server

Στην ενότητα αυτή, θα περιγραφεί η ανάπτυξη του κύριου τμήματος λογισμικού της εργασίας – του server. Η εφαρμογή αυτή θα λαμβάνει και θα χειρίζεται το input του χρήστη από τον client, έτσι ώστε να δημιουργήσει την τοπολογία στο υλικό. Ο server θα είναι υπεύθυνος για όλες τις ενδιάμεσες διαδικασίες, όπως την απόδοση διευθύνσεων, την

δημιουργία των εικονικών μηχανών, την αρχική παραμετροποίηση τους καθώς και την παραγωγή ενός Ansible inventory και την εκτέλεση του σχετικού playbook πάνω σε αυτό.

5.5.1 Δομή του Κώδικα

Παραπέμποντας στην παραπάνω ενότητα, στην οποία περιγράφηκε η δομή του έργου από την άποψη των αρχείων και των φακέλων του, η ενότητα αυτή θα επεξηγήσει συνοπτικά το πώς είναι δομημένος ο κώδικας στους φακέλους αυτούς, καθώς και το ποια είναι η λειτουργία τους.

- **Controllers/**
  Φάκελος, που περιέχει κλάσεις, οι οποίες χειρίζονται τα HTTP requests, που λαμβάνει ο server. Κληρονομούν την κλάση **Controller** του ASP.NET και καθορίζουν το ποιος controller θα καλείται για κάθε request με χρήση του attribute notation της C#, που θα περιγραφεί παρακάτω.

- **Models/**
  Περιέχει κλάσεις με τα αντικείμενα της εφαρμογής. Τα αντικείμενα αυτά αποτελούνται, κυρίως, από πεδία και attributes στα πεδία αυτά, που περιγράφουν το πώς γίνονται serialize και deserialize, καθώς και μερικές βοηθητικές μεθόδους. Τέτοια αντικείμενα είναι, για παράδειγμα, η κλάση **Router** για τους δρομολογητές και η κλάση **Tunnel**, που περιγράφει της παραμέτρους ενός tunnel interface.

- **Options/**
  Περιέχει απλές κλάσεις, που κρατάνε την παραμετροποίηση της εφαρμογής.

- **Service/**
  - **Logging/**: Απλές κλάσεις σχετικά με την καταχώρηση των logs.
  - **Routing/**: Κλάσεις για υλοποίηση αλγορίθμων εύρεσης μονοπατιών για κατασκευή στατικών διαδρομών.

  Το κύριο business logic της εφαρμογής. Περιέχει κλάσεις που συνδέονται, μέσω SSH, σε εικονικές μηχανές δρομολογητών, που αναπαριστούν μια τοπολογία και που παράγουν inventory για το Ansible.

- **Views/**
  Razor HTML templates, που παράγουν τις σελίδες της εφαρμογής. Αν και υπάρχουν ορισμένες περιπτώσεις που το back-end παρέχει κάποιες μεταβλητές στην κατασκευή αυτής της σελίδας, δεν υπάρχει αξία να αναφερθεί από την οπτική του server.

5.5.2 Επικοινωνία με το Γραφικό Περιβάλλον

Το έργο είναι χτισμένο πάνω στο ASP.NET MVC framework. Λειτουργώντας με το framework αυτό, ο κώδικας ορίζει controllers, οι οποίοι χειρίζονται τα αιτήματα που γίνονται στον HTTP server, που είναι ενσωματωμένος στο framework αυτό. Το framework χειρίζεται, επίσης, το serialization και deserialization των δεδομένων, που χειρίζονται από τους controllers, που σημαίνει ότι, ενώ το πρωτόκολλο HTTP χειρίζεται με την αποστολή

κειμένου JSON – που είναι ο κύριος τρόπος serialization αντικειμένων στο διαδίκτυο, - οι controllers χειρίζονται τα αντικείμενα αυτά κατευθείαν, χωρίς μέριμνα για την χρήση του πρωτοκόλλου.

Οι controllers ορίζονται σε κλάσεις που επεκτείνουν την κλάση **Controller** και ορίζουν τις διαδρομές των μεθόδων τους μέσω Attributes. Στο παρακάτω παράδειγμα, ορίζεται ο controller που χειρίζεται τα HTTP requests στο path "/api" της εφαρμογής. Συγκεκριμένα, για "**/api/create**", καλείται η μέθοδος **Create(…)**, για "**/api/topology/5**", καλείται η μέθοδος **Topology(5)** και, για "**/api/logs/12**", καλείται η **Output(12)**.

```
[Route("api")]
public class ApiController : Controller
{
	[HttpPost("create")]
	public IActionResult Create([FromBody]JObject topologyJson)
	{

	}

	[HttpGet("topology/{id}")]
	public IEnumerable<Router> Topology(int id)
	{

	}

	[HttpGet("logs/{id}")]
	public IEnumerable<LogMessage> Output(int id)
	{

	}

	[HttpPost ("delete/{id}")]
	public IActionResult Output(int id)
	{

	}

}
```

Συνηθίζεται οι controllers να μην περιέχουν πολύ business logic και να λειτουργούν αμιγώς ως προγραμματίστηκες διεπαφες με το υπόλοιπο λογισμικό. Έτσι, με τις κλάσεις στον φάκελο **Service** να υλοποιούν τις λειτουργίες της εφαρμογής, οι controllers, χειριζόμενοι των αιτημάτων του χρήστη, είναι υπεύθυνοι για να αποκωδικοποιούν τα δεδομένα που στέλνει ο client, να χρησιμοποιεί τις υπόλοιπες κλάσεις για την εκτέλεση του αιτήματος και την κωδικοποίηση των δεδομένων στην απάντηση του χρήστη. Στο παράδειγμα, επίσης, φαίνεται και το πώς το framework χειρίζεται το serialization τον αντικειμένων, καθώς και η μέθοδος `Output(int id)`, για παράδειγμα, επιστρέφει λίστα από αντικείμενα LogMessage, η οποία γίνεται serialize σε κείμενο, που απεικονίζει μια JSON λίστα.

5.5.3 Απόδοση Διευθύνσεων από το Pool

Ενδιαφέρον παρουσιάζει και η υλοποίηση του pool διευθύνσεων στο επίπεδο λογισμικού, η οποία πρέπει να μπορεί να είναι σε θέση να αποφεύγει συγκρούσεις (conflicts) στις διευθύνσεις που αποδίδονται, αλλά και να επιτρέπει την ασφαλή παράλληλη πρόσβαση σε

αυτό (thread safe). Για να επιτυγχάνεται αυτό, η υλοποίηση ακολουθεί τις παρακάτω σχεδιαστικές αποφάσεις για την κλάση **AddressPool,** που βρίσκεται στο αρχείο **Options/AddressPool.cs**:

- **Αποφυγή εσωτερικής κατάστασης**. Το pool δεν θα διατηρεί λίστα με τις διευθύνσεις που αποδίδονται, καθώς αυτό, από τη μία, δεν είναι πραγματικά αξιόπιστο και, από την άλλη, οδηγεί την εφαρμογή στο να λειτουργεί διαφορά μετά από επανεκκίνηση. Όταν το pool επιχειρεί να αποδόσει κάποια διεύθυνση, θα ελέγχει διαθεσιμότητα στο εύρος του κάνοντας ελέγχους μέσω ping, καθώς και το επιπλέον κόστος του περίπου 400ms ανά διεύθυνση αξίζει την αποφυγή συγκρούσεων με διευθύνσεις που χρησιμοποιούνται ήδη.

- **Κλείδωμα της διεύθυνσης zero-day.** Το pool θα υλοποιεί software lock με αντικείμενο τη δική του zero-day διεύθυνση, επιτρέποντας, έτσι, σε παράλληλες εκτελέσεις της εφαρμογής να μην έλθουν σε σύγκρουση, χρησιμοποιώντας τη διεύθυνση αυτή. Το υπόλοιπο πρόγραμμα θα έχει πρόσβαση σε αυτό, μέσω των μεθόδων **LockZeroDay()** και **ReleaseZeroDay()**.

- **Διατήρηση λίστας διευθύνσεων που έχουν αποδοθεί αλλά δεν είναι ακόμη υποκρίσιμες σε ping**. Το pool θα βασίζεται σε ελέγχους μέσω ping για την απόδοση διευθύνσεων, όμως, υπάρχει το ενδεχόμενο μία διεύθυνση να έχει δεσμευτεί από κάποιον δρομολογητή και ο δρομολογητής αυτός είναι να βρίσκεται, ακόμα, στη διαδικασία εκκίνησης ή να μην έχει αποκτήσει ακόμα τη διεύθυνση που του έχει αποδοθεί. Για την αποφυγή αυτού, το pool θα διατηρεί μία λίστα με αυτές τις διευθύνσεις, στην οποία θα προσθέτει κάθε διεύθυνση που αποδίδει και από την οποία θα αφαιρεί διευθύνσεις όταν ο TopologyController καλέσει τη μέθοδο **ReleaseAddress(),** αφού την αποδόσει επιτυχώς σε κάποιον δρομολογητή.

5.5.4 Deserialization της Μοντελοποιημένης Τοπολογίας

Όπως έχει προαναφερθεί, η τοπολογία υποβάλλεται από τον client στον server, μέσω JSON κειμένου στο HTTP path "/api/create". Στην ενότητα αυτή, θα δούμε το πώς ο server θα αναλύσει την απεικόνιση αυτή, ώστε να δημιουργήσει την τοπολογία αυτή σε αντικείμενα της C#, τα οποία μπορεί να επεξεργαστεί. Αυτό επιτελείται στο αρχείο **Service/TopologyController.cs** και, συγκεκριμένα, από τη μέθοδο **Deserialize**. Η λειτουργία της μεθόδου αυτής περιγράφεται από τον παρακάτω ψευδοκώδικα:

```
Topology Deserialize(JObject topologyJson)
{
    // Assign an auto-increment id to this topology
    int id = AssignId();

    Topology topology = new Topology(id);

    /*
    * Deserialize routers and their tunnels, while assigning addresses from the pool.
    */
    foreach (routerObj in topologyJson."topology_template")
    {
        // Initialize the router's properties
        Router router = new Router(routerObj.proprties.router_id, routerObj.properties.hostname);

        EsxiClient esxi = PickEsxiHostFor(router);
```

```
            IPAddress assignedAddress = esxi.GetAddressPool().AssignAddress();

            router.PublicIP = assignedAddress;

            topology.Add(router);
        }

        /*
        * Match the remote router of the tunnels with their assigned addresses
        */
        foreach (router in topology)
        {
            foreach (tunnel in router.Tunnels)
            {
                tunnel.RemoteRouter = topology[tunnel.RemoteId];
            }
        }

        /*
        * Construct the static routes for each router
        */
        foreach (router in topology)
        {
            // Use the BFS method for now
            IStaticRouteProvider routeProvider = new BreadthFirstRouteProvider();

            router.StaticRoutes = routeProvider.GetRoutesFor(router);
        }

        return topology;
    }
```

5.5.5 Διασύνδεση με τους ESXi Hosts

Για την χρήση των ESXi hosts, ο server θα χρησιμοποιεί κανάλι SSH, στο οποίο θα στέλνει τις κατάλληλες εντολές. Για τη λειτουργικότητα αυτή, θα χρησιμοποιηθεί η βιβλιοθήκη SSH.NET, η λειτουργικότητα της οποίας κρύβεται από το υπόλοιπο πρόγραμμα μέσα στην κλάση **VMClient** του αρχείου **Service/VMClient.cs**. Η κλάση αυτή χρησιμοποιεί την προγραμματιστική διεπαφή (API) της βιβλιοθήκης αυτής, ώστε να παρέχει σε όποια κλάση την επεκτείνει να εκτελεί εντολές σε κάποιο απομακρυσμένο μηχάνημα. Η **EsxiClient** είναι μία τέτοια κλάση, που, επεκτείνοντας την παραπάνω κλάση, έχει πρόσβαση στην μέθοδο **RunCommand**. Χρησιμοποιώντας τη μέθοδο αυτή, η κλάση υλοποιεί την συνάρτηση **RegisterVm**, μέσω της οποίας εγγράφει και ενεργοποιεί καινούργια εικονική μηχανή στον συγκεκριμένο ESXi host, όπως φαίνεται παρακάτω:

```
public async Task RegisterVm(string name)
{
        string imageDir = hostInfo.GetImageDir("vyos");
        string vmxDir = hostInfo.GetPath(string.Format("vms/{0}-{1}", name,
DateTimeOffset.UtcNow.ToUnixTimeSeconds()));
        string vmx = Path.Combine(vmxDir, "vyos.vmx").Replace('\\', '/');

        /*
        * Create the destination folder
        */
        await RunCommand("mkdir -p \"{0}\"", vmxDir);

        /*
        * Copy the VMX file
        */
        await RunCommand("cp '{0}/vyos.vmx' '{1}'", imageDir, vmx);

        /*
        * Clone the virtual disk
        */
```

```
        await RunCommand("vmkfstools -d thin -i '{0}/vyos.vmdk' '{1}/vyos.vmdk'", imageDir, vmxDir);

        /*
        * Register a new VM at the new dir
        */
        string register = await RunCommand("vim-cmd solo/registervm '{0}' '{1}'", vmx, name);

        /*
        * Power-On the new VM
        */
        int vmId = int.Parse(register.Trim());
        await RunCommand("vim-cmd vmsvc/power.on {0}", vmId);
}
```

Όπως φαίνεται, δημιουργείται νέο directory στον host, χρησιμοποιώντας το τωρινό UNIX timestamp και το όνομα του VM για την πρόχειρη αποφυγή συγκρούσεων. Επίσης, χρησιμοποιεί την έξοδο της εντολής **solo/registervm,** για να μάθει το ID που ανέθεσε το ESXi στο νέο VM, έτσι ώστε να το ενεργοποιήσει.

5.5.6 Αρχικοποίηση Εικονικού Δρομολογητή

Στις προηγούμενες ενότητες, κάθε δρομολογητής στην εν λόγω τοπολογία δημιουργήθηκε σε κάποιον από τους διαθέσιμους ESXi hosts, αφού πρώτα του ανατέθηκε κάποια δημόσια διεύθυνση. Το επόμενο βήμα είναι να πραγματοποιηθεί σύνδεση μέσω SSH από τον server στο νέο VM, ώστε να του αποδοθεί η διεύθυνση αυτή. Με προσέγγιση παρόμοια με την ενότητα **5.5.5**, υλοποιήθηκε η κλάση **VyosClient**, η οποία επεκτείνει την κλάση **VmClient**. Μια σημαντική διαφορά, στην περίπτωση αυτή, είναι ότι λειτουργικά συστήματα, όπως του VyOS, δεν υποστηρίζουν τη συνηθισμένη δυνατότητα του SSH για εκτέλεση διακριτών εντολών. Στην πράξη, το shell του υλοποιείται μέσω απλού καναλιού TCP, στο οποίο γράφονται οι εντολές ως είσοδο και διαβάζεται το standard output του shell ως έξοδος. Αφού η κλάση προσαρμόστηκε στην απαίτηση αυτή, χρησιμοποιώντας την κλάση **ShellStream** της βιβλιοθήκης SSH.NET, υλοποιήθηκε η συνάρτηση **ConfigureAddress,** που εκτελεί στο συγκεκριμένο δρομολογητή την κατάλληλη αλληλουχία εντολών για την ρύθμιση της δημόσιας διεύθυνσης. Η υλοποίησή της φαίνεται παρακάτω με ψευδοκώδικα:

```
void ConfigureAddress(ipAddress, suffix, gateway)
{
    // Run 'configure'
    EnterConfiguration();

    /*
    * Clear the interface
    */
    RunCommand("delete interfaces ethernet eth0");

    /*
    * Set default gateway
    */
    RunCommand("set system gateway-address {0}", gateway);

    /*
    * Set the the interface address
    */
    RunCommand("set interfaces ethernet {0} address {1}/{2}", INTERFACE, ipAddress, suffix);

    // Run 'commit', 'save' and 'exit'
    SaveConfiguration();
}
```

5.5.7 Διασύνδεση με το Ansible

Η χρήση του Ansible από τον server, που θεωρεί δεδομένο ότι το Ansible είναι εγκατεστημένο στο περιβάλλον του, γίνεται σε δύο βήματα. Στο πρώτο βήμα, το **TopologyController** δημιουργεί το δυναμικό inventory, όπως αυτό περιγράφηκε στην ενότητα **4.3.2** με χρήση της κλάσης **AnsibleInventory** στο αρχείο **Service/AnsibleInventory.cs**. Η κλάση αυτή επιτελεί ένα απλό περιτύλιγμα της δημιουργίας των αρχείων του inventory. Διατηρεί μία λίστα με τους δρομολογητές της τοπολογίας, ομαδοποιημένους με το λειτουργικό τους, στην οποία προστίθενται οι δρομολογητές με χρήση της μεθόδου **AddHost**. Καλώντας την συνάρτηση **Export**, η κλάση αυτή δημιουργεί προσωρινό φάκελο, στον οποίο γράφει όλα τα απαραίτητα αρχεία. Το αρχείο **hosts** το παράγει η ίδια η κλάση, ενώ τα αρχεία του φακέλου **host_vars** παράγονται κάνοντας deserialize το κάθε αντικείμενο **Router** σε YAML, το οποίο εκδίδει τις απαραίτητες παραμέτρους του, όπως τα tunnel interfaces του και τις στατικές διαδρομές του. Η υλοποίηση της συνάρτησης Export σε ψευδοκώδικα, η οποία, επίσης, επιστρέφει το path του φακέλου στον οποίο έγραψε το inventory, φαίνεται παρακάτω:

```
string Export()
{
    string sourceDir = Options.AnsibleBaseConfigurationDir;

    /*
    * Copy the base Ansible files from the project onto a temp directory
    */
    CopyDirectoryRecursive(sourceDir, targetDir);

    /*
    * Generate and write the hosts file
    */
    string hostsFile = targetDir + "inventory/hosts";
    string hostsFileText = GenerateHostsFile();
    File.Write(hostsFile, hostsFileText);

    /*
    * Generate and writ the host_vars files
    */
    string hostVarsDir = targetDir + "inventory/host_vars";
    Directory.CreateDirectory(hostVarsDir);
    foreach (router in routers)
    {
        string hostVarsFile = hostVarsDir + router.PublicIP + ".yml";

        string routerVarsYaml = SerializeYaml(router);

        File.Write(hostVarsFile, routerVarsYaml);
    }

    return targetDir;
}
```

Όπως φαίνεται, αρχικά αντιγράφεται το template του inventory, το οποίο υπάρχει μαζί με τον κώδικα. Στο template αυτό περιέχονται τόσο τα **group_vars,** που ορίζονται, όπως περιγράφηκε στο κεφάλαιο **4.3.2**, αλλά και το αρχείο **ansible.cfg,** που φαίνεται παρακάτω. Στο αρχείο αυτό, απενεργοποιούμε το **host_key_checking**, διότι πρέπει το Ansible αυτόματα να εμπιστεύεται τις ψηφιακές υπογραφές των νέων δημιουργημένων εικονικών δρομολογητών. Επίσης, προσδιορίζει ότι το inventory θα βρίσκεται σε φάκελο στο ίδιο επίπεδο με το αρχείο αυτό και με όνομα “inventory”. Οι ρυθμίσεις του αρχείου αυτού είναι σε ισχύ όταν εκτελείται κάποια εντολή του Ansible με το working directory να είναι το directory, που περιέχει το αρχείο αυτό.

```
[defaults]
inventory = inventory
private_key_file = ~/.ssh/id_rsa
host_key_checking = False
```

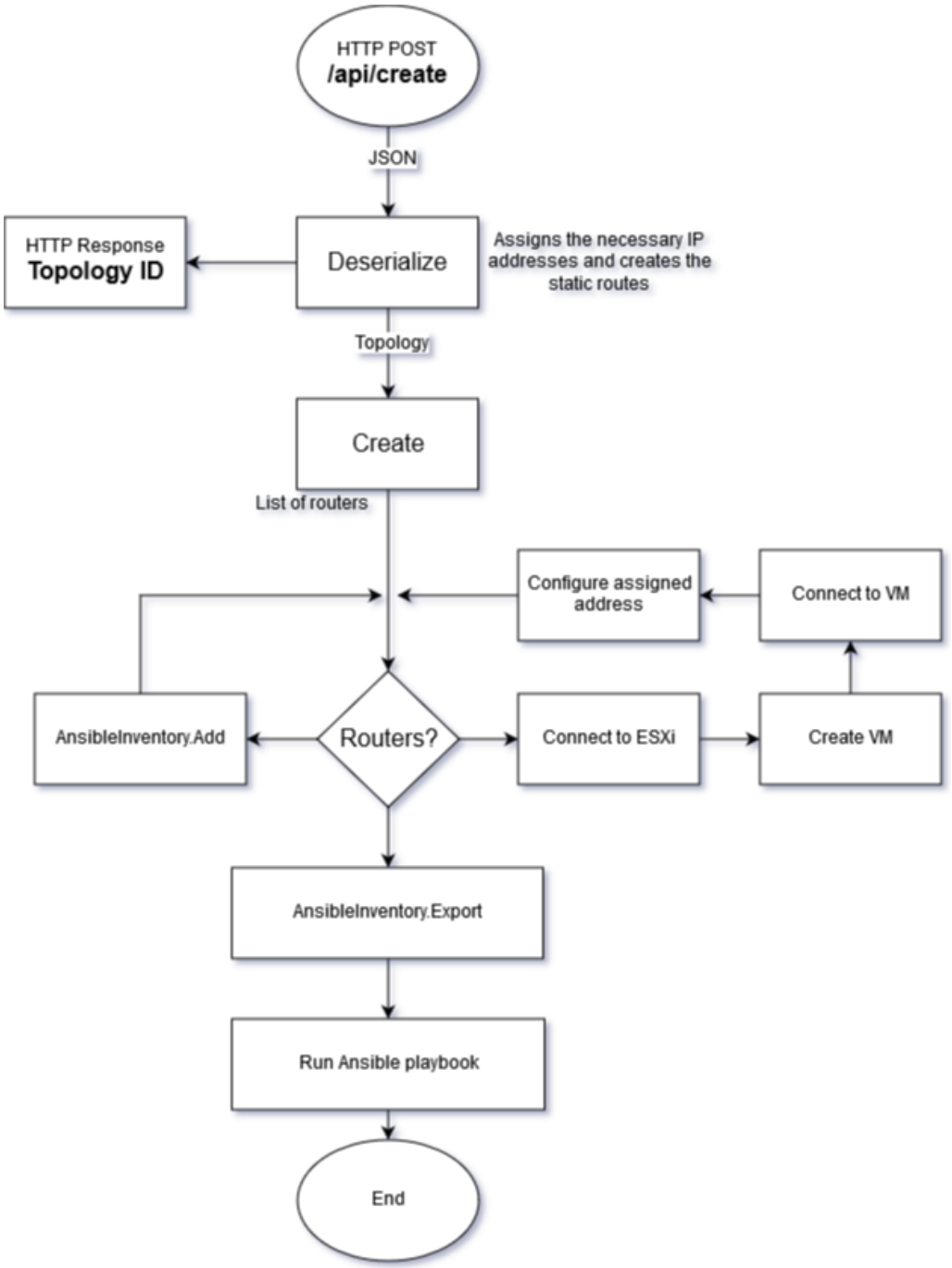
Τέλος, η εκτέλεση του playbook στο inventory, που παράγεται με τη μέθοδο Export, γίνεται εύκολα με τη χρήση της βιβλιοθήκης **ansible-net**, όπως φαίνεται στον παρακάτω ψευδοκώδικα:

```
using Ansible;

Playbook playbookCmd = new Playbook("playbook.yml");
playbookCmd.WorkingDirectory = targetDir;
playbookCmd.Execute();
```

5.5.8 Διαδικασία Ενορχήστρωσης της Τοπολογίας

Στο παρακάτω διάγραμμα, φαίνεται η διαδικασία που ακολουθείται από το πρόγραμμα, κατά τη λήψη ενός POST request, για τη δημιουργία κάποιας τοπολογίας. Η πρώτη διακλάδωση συμβαίνει διότι η διαδικασία αυτή, όπως και ο κώδικας σε όλο το υπόλοιπο έργο, εκμεταλλεύονται τις δυνατότητες ασύγχρονου προγραμματισμού του .NET framework. Οπότε, αφού η διαδικασία ξεκινάει χωρίς να μπλοκάρει το πρόγραμμα, ο controller επιστρέφει απλά στο χρήστη το ID, που αποδόθηκε στην τοπολογία από τη διαδικασία Deserialize – η οποία είναι σύγχρονη, - χωρίς να χρειάζεται να διατηρήσει ανοιχτή τη σύνδεση. Έτσι, η σελίδα του client μπορεί να χρησιμοποιήσει το ID αυτό, για να ανακατευθύνει το χρήστη στη σελίδα, στην οποία μπορεί να παρακολουθεί την εξέλιξη της τοπολογίας σε πραγματικό χρόνο.



13. Διάγραμμα Ροής Ενορχήστρωσης Τοπολογίας

5.6 Διαδικασία Χτισίματος της Εφαρμογής

Σε τέτοιου είδους εργασίες λογισμικού, που αποτελούνται από πολλαπλά στάδια στην τεχνολογική τους στοίβα και που υλοποιούν πολλαπλές διεπαφές για τη χρήση τους, η διαδικασία χτισίματος (**build pipeline**) συνήθως καταλήγει να είναι περίπλοκη διαδικασία πολλαπλών σταδίων. Για το λόγο αυτό, θα αφιερωθεί και αυτή η ενότητα στο πώς ο κώδικας μπορεί να χτίζεται, παράγοντας το τελικό του εκτελέσιμο και όλα του τα asset, με συνέπεια και ομοιομορφία κάθε φορά. Οι βιβλιοθήκες που χρησιμοποιούνται από το λογισμικό, όπως αυτές αναφέρθηκαν προηγουμένως, δεν συμπεριλαμβάνονται στον πηγαίο κώδικα της ανάπτυξης. Οι σύγχρονοι διαχειριστές πακέτων (package managers)

λειτουργούν προσδιορίζοντας αναφορές (references) στις απαιτούμενες βιβλιοθήκες, παρέχοντας παράλληλα τη δυνατότητα ανάκτησης των βιβλιοθηκών αυτών, μέσω των δημόσιων καταλόγων που φιλοξενούν τα πακέτα αυτά. Εκμεταλλευόμενοι αυτής της δυνατότητας, αποφεύγεται κατά κανόνα η συμπερίληψη των εξωτερικών αυτών βιβλιοθηκών στον έλεγχο πηγαίου κώδικα (source control) της εφαρμογής. Η διαδικασία αυτή έχει τις παρακάτω δύο απαιτήσεις (dependencies) από το περιβάλλον χτισίματος (build environment):

- **Node.js**
  Περιβάλλον εκτέλεσης (runtime) για τη Javascript. Χρησιμοποιείται ευρέως για την ανάπτυξη λογισμικού Javascript, το οποίο μπορεί να εκτελεστεί σε οποιοδήποτε περιβάλλον. Η εγκατάσταση της Node.js συνοδεύεται και από το εργαλείο **npm**, που είναι ο διαχειριστής πακέτων (package manager) για τις αναπτύξεις αυτές. Στην περίπτωση της εργασίας αυτής, η Node δεν χρησιμοποιείται για την εκτέλεση του προγράμματος, απαιτείται, όμως, για τον χειρισμό πακέτων, που απαιτούνται για το χτίσιμο του front-end κομματιού, καθώς και για το runtime των πακέτων αυτών.

- **Gulp.js**
  Τεχνικά, δεν είναι απαραίτητο να υπάρχει στο build environment, καθώς γίνεται αναφορά σε αυτό ως build dependency του έργου, οπότε θα ανακτάται αυτόματα σε κάθε χτίσιμο με χρήση του **npm**.

- **.NET Core SDK**
  Το πακέτο αυτό είναι διαθέσιμο από τη Microsoft και περιέχει το .NET Framework, καθώς και το εκτελέσιμο πρόγραμμα **dotnet**. Το πρόγραμμα αυτό περιέχει τον compiler για τα έργα σε .NET, το runtime των εκτελέσιμων προγραμμάτων που παράγει και τη λειτουργικότητα του **NuGet**, που είναι ο package manager για τα έργα αυτά.

Για το χτίσιμο, έχουμε ως working directory τον φάκελο του project. Δηλαδή, αυτόν που περιέχει τα αρχεία .**csproj**, **gulpfile.js** και **package.json**. Αρχικά, εκτελούμε επαναφορά των πακέτων του κώδικα C#.

```
$ dotnet restore
```

Η παραπάνω εντολή χρησιμοποιεί τα references που έχει το back-end του project σε πακέτα, ώστε να τα επαναφέρει από τον επίσημο κατάλογο του NuGet. Ύστερα, εκτελούμε την ίδια διαδικασία, αλλά για τα πακέτα του front-end μέσω του npm.

```
$ npm install
```

Μαζί με τα dependencies, η παραπάνω εντολή ανακτά και το εκτελέσιμο του gulp. Οπότε, εκτελούμε τώρα και το pipeline για το χτίσιμο του front-end, το οποίο θα κατασκευάσει τον φάκελο **wwwroot**, τοποθετώντας εκεί τα αρχεία που θα προκύψουν.

```
$ gulp build
$ gulp min
```

Με τα assets της εφαρμογής και τα dependencies παρών, μένει μόνο να γίνει compile το ASP.NET κομμάτι του έργου.

```
$ dotnet publish -c Release -o out
```

Η παραπάνω εντολή χτίζει το έργο στη ρύθμιση “Release”, γράφοντας το αποτέλεσμα μαζί με όλα τα απαραίτητα assets στον φάκελο **out**. Η παραπάνω διαδικασία παράγει αρχείο τύπου **dll**, το οποίο εκτελείται μέσω του framework για την έναρξη του server, όπως φαίνεται παρακάτω:

```
$ cd out
$ dotnet Autonet.dll
```

Κεφάλαιο 6

Λειτουργία και Χρήση της Εφαρμογής

6.1 Χρήση του Γραφικού Περιβάλλοντος

Παραπέμποντας τις εικόνες του κεφαλαίου **4.4**, η ενότητα αυτή θα αναφερθεί στον τρόπο με τον οποίο το γραφικό περιβάλλον της εφαρμογής χρησιμοποιείται από τον χρήστη. Όπως φαίνεται, η αρχική σελίδα της εφαρμογής αποτελείται από έναν κύριο σχεδιαστικό χώρο στο κέντρο, ανάμεσα από δύο μενού, το ένα εκ των οποίων εμφανίζεται επιλεκτικά.

- **Σχεδιαστικός χώρος**
  Το κέντρο της εφαρμογής αποτελείται από ένα τετραγωνικό χωρίο, μέσα στο οποίο ο χρήστης μπορεί να σχεδιάσει την τοπολογία. Η περιοχή αυτή λειτουργεί ως καμβάς (canvas), δίνοντας στο χρήστη τη δυνατότητα να μετακινήσει κόμβους με το ποντίκι αλλά και να τους επιλέξει για επεξεργασία. Η επιλογή ενός κόμβου, ενώ ο χρήστης έχει ήδη επιλεγμένο κάποιον άλλον κόμβο, θα δημιουργήσει ζεύξη (link) μεταξύ των δύο αυτών κόμβων. Κάθε τέτοια ζεύξη αντιπροσωπεύει και ένα tunnel interface μεταξύ των δύο δρομολογητών. Ο χρήστης έχει, επίσης, τη δυνατότητα να μεγενθύνει τον σχεδιαστικό χώρο, χρησιμοποιώντας το scroll του ποντικιού, αλλά και να μετακινήσει ολόκληρη την όψη κάνοντας drag από ένα κενό σημείο.

- **Γραμμή εργαλείων (toolbar)**
  Στα αριστερά αυτής της σελίδας, υπάρχει ένα μενού με τα παρακάτω κουμπιά:
  - **+ Router**: Προσθήκη νέου δρομολογητή στην τοπολογία.
  - **Auto Scale**: Αυτόματη προσαρμογή ή διόρθωση του μεγέθους και της θέσης του σχεδιαστικού περιβάλλοντος, ώστε να γίνουν εμφανείς όλοι οι κόμβοι.
  - **Export**: Παραγωγή της μοντελοποίησης της τοπολογίας σε μορφή YAML ή JSON, ώστε να την αποθηκεύσει ο χρήστης.
  - **Import**: Εισαγωγή αποθηκευμένης μοντελοποιημένης τοπολογίας από τον χρήστη, ώστε να φορτωθεί στο σχεδιαστικό περιβάλλον.
  - **Deploy**: Αποστολή της τοπολογίας στον server για εγκατάσταση. Μετά την αποστολή, ο χρήστης θα συνδιευθυνθεί στην επόμενη σελίδα.

- **Μενού επεξεργασίας αντικειμένου**
  Στα δεξιά της οθόνης αυτής, εμφανίζεται ένα μενού, όταν ο χρήστης επιλέγει κάποιο αντικείμενο στο σχεδιαστικό περιβάλλον. Στο μενού αυτό, ο χρήστης μπορεί να παραμετροποιήσει το επιλεγμένο αντικείμενο ή να το διαγράψει με χρήση του κουμπιού “Delete”.

Κάνοντας **“Deploy**” την τοπολογία του, ο χρήστης οδηγείται στην επόμενη σελίδα της εφαρμογής, όπου έχει τη δυνατότητα να παρακολουθεί την εξέλιξη της εγκατάστασής της. Στα αριστερά της οθόνης αυτής, φαίνεται η κατάσταση των δρομολογητών αυτών, μαζί με πληροφορίες σχετικά με τις ζεύξεις τους, καθώς και το κουμπί “**Delete**”, με το οποίο ο χρήστης μπορεί να στείλει εντολή στον server για τη διαγραφή της συγκεκριμένης τοπολογίας. Στα δεξιά της οθόνης, τυπώνονται μηνύματα από τον server, που, εκτός από ενημερωτικά μηνύματα, περιέχουν και την έξοδο (output) των εντολών παραμετροποίησης.

6.2 Παραμετροποίηση του Server

Στην ενότητα αυτή, θα γίνει η περιγραφή του πώς κάποιος χρήστης της εφαρμογής θα παραμετροποιήσει το back-end της εφαρμογής, ώστε να την επεκτείνει με δική του λειτουργικότητα. Όλες οι παράμετροι που υποστηρίζονται βρίσκονται στο αρχείο **options.yml**, που βρίσκεται στον φάκελο του έργου. Το αρχείο αυτό αντιγράφεται στον φάκελο εξόδου, κατά τη διαδικασία χτισίματος, αυτόματα από το **dotnet**, καθώς έχει αναφερθεί στο **csproj** ως αρχείο του έργου. Επίσης, πρέπει να έχει αυστηρή δομή, καθώς και το πρόγραμμα το διαβάζει και το κάνει deserialize σε αντικείμενα της C#.

Καταρχάς, ορίζεται το path προς το αρχείο με το ιδιωτικό κλειδί SSH, που θα χρησιμοποιείται από την εφαρμογή.

```yaml
ssh_key: "~/.ssh/id_rsa"
```

Επίσης, είναι δυνατή η ρύθμιση της διάρκειας που θα επιτρέπεται στα ping που θα εκτελούν τα address pool όταν ελέγχουν εάν κάποια διεύθυνση χρησιμοποιείται ήδη. Εάν τα διαθέσιμα pool διευθύνσεων έχουν μεγάλο εύρος τιμών και οι διαθέσιμοι ESXi hosts είναι κοντά γεωγραφικά στην εφαρμογή, μικρότερο timeout στους ελέγχους ping μπορεί να βελτιώσει σημαντικά την ταχύτητα της εφαρμογής.

```yaml
ping_timeout_ms: 400
```

Ύστερα, ορίζονται τα pool διευθύνσεων, που θα χρησιμοποιούνται σε λίστα YAML. Όπως αναφέρθηκε στην ενότητα **4.1.3**, το κάθε pool απορρυθμίζεται από κάποιο μοναδικό ID, το οποίο χρησιμοποιείται για αναφορά σε αυτό.

```yaml
address_pools:
  - pool_id: 0
    zero_day: 147.102.39.41
    from: 147.102.39.30
    to: 147.102.39.40
    gateway: 147.102.39.1
    suffix: 26
  - pool_id: 1
    ...
```

Ορίζεται μία λίστα σε YAML, η οποία περιέχει πληροφορίες για τους διαθέσιμους ESXi hosts.

```yaml
hosts:
  - name: 'esxi1'
    address: '147.102.39.18'
    username: 'root'
    data_dir: '/vmfs/volumes/datastore1'
    address_pool: 0
    images:
      vyos: 'vyos'
  - name: 'esxi2'
```

```
    ...
```

Τέλος, παραμετροποιούνται και τα διαθέσιμα είδη δρομολογητών, προσδιορίζοντας, έτσι, τα ονόματα των ομάδων, στις οποίες θα υπάγονται, κατά τη χρήση του Ansible.

```
router_types:
  vyos: routers_vyos
  cisco: routers_cisco
  ...
```

6.3 Παραμετροποίηση του Ansible

Στην ενότητα αυτή, θα πραγματοποιηθεί η περιγραφή του πώς η εφαρμογή του server μπορεί να επεκταθεί, για την υποστήριξη περισσότερων λειτουργικών συστημάτων στους εικονικούς δρομολογητές που χρησιμοποιούνται, αλλά και για την υλοποίηση της συνδεσιμότητας των ιδιωτικών δικτύων, με μεθόδους διαφορετικές από τα GRE tunnels, που χρησιμοποιούνται από την αρχική υλοποίηση. Όπως αναφέρθηκε στο σχετικό κεφάλαιο, το δυναμικό inventory, που παράγεται από την εφαρμογή, αποτελείται από το αρχείο **hosts**, το οποίο περιέχει τη λίστα με τους δρομολογητές της τοπολογίας, ομαδοποιημένους ανά λειτουργικό σύστημα, καθώς και ένα αρχείο για κάθε δρομολογητή στον φάκελο **host_vars**. Τα αρχεία με τις παραμέτρους του κάθε δρομολογητή δεν είναι παραμετροποιήσιμα και παράγονται από την εφαρμογή με τέτοιο τρόπο, ώστε να παράγουν τις κατάλληλες μεταβλητές, που περιγράφουν πλήρως την επιθυμητή συνδεσιμότητα.

Αρχικά, για την αλλαγή της μεθόδου εγκατάστασης των απομακρυσμένων ζεύξεων, αρκεί η επεξεργασία του play, που αντιστοιχεί στην επιθυμητή ομάδα από hosts. Συγκεκριμένα, στο αρχείο **Ansible/playbook.yml**. Για παράδειγμα, για χρήση του πρωτοκόλλου IPIP αντί του πρωτοκόλλου GRE, μπορεί να εφαρμοστεί η παρακάτω προσαρμογή:

```
- name: "configuring interface tunnels"
  loop: "{{ tunnels }}"
  vyos_config:
    save: yes
    lines:
      - set interfaces tunnel "{{ item.interface }}" encapsulation ~~gre~~ ipip
      - set interfaces tunnel "{{ item.interface }}" local-ip "{{ inventory_hostname }}"
      - set interfaces tunnel "{{ item.interface }}" remote-ip "{{ item.remote }}"
      - set interfaces tunnel "{{ item.interface }}" address "{{ item.addr }}/32"
      - set protocols static interface-route "{{ item.route }}/32" next-hop-interface "{{
item.interface }}"
```

Και για την προσθήκη νέων λειτουργικών συστημάτων εικονικών δρομολογητών, απαιτούνται οι παρακάτω τροποποιήσεις. Αρχικά, η προσθήκη του είδους του δρομολογητή στο αρχείο **options.yml**.

```
router_types:
  vyos: routers_vyos
  quagga: routers_quagga
```

Με την παραπάνω ρύθμιση, το αρχείο hosts, θα ομαδοποιεί τώρα κατάλληλα τους δρομολογητές που λειτουργούν με Quagga. Οπότε, τώρα μένει μόνο η εισαγωγή ενός νέου play στο αρχείο **playbook.yml**, το οποίο να περιγράφει κατάλληλα την παραμετροποίηση του νέου δρομολογητή.

```
- name: "Initialization of VyOS routers"
  hosts: routers_vyos
  gather_facts: no
  tasks:
  - name: "setting router hostnames"
    ...

- name: "Initialization of Quagga routers"
  hosts: routers_quagga
  tasks:
  - name: "setting router hostnames"
    shell: hostname "{{ inventory_hostname }}"
```

6.4 Ανάπτυξη με Χρήση του Docker

Όπως προαναφέρθηκε, η διαδικασία χτισίματος της εφαρμογής έχει κάποιες απαιτήσεις (dependencies) από το περιβάλλον που επιχειρεί να κάνει το compile, οι οποίες ενδέχεται να μην υπάρχουν σε κάθε σύστημα. Επιπλέον, το περιβάλλον εκτέλεσης της εφαρμογής έχει, επίσης, απαιτήσεις εκτέλεσης (runtime dependencies), όπως την ύπαρξη του εκτελέσιμου **dotnet** και του **Ansible**. Για το λόγο αυτό, μαζί με τον κώδικα της εφαρμογής, παρέχεται και η δυνατότητα εκτέλεσης και χρήσης της με χρήση του **Docker**. Το Docker είναι πρόγραμμα, το οποίο παρέχει την δυνατότητα εκτέλεσης λογισμικού σε **containers**. Ένα container είναι στιγμιότυπο ενός **image**, το οποίο παράγεται με χρήση ενός αρχείου που αποκαλείται **Dockerfile**, το οποίο περιγράφει μια σειρά εντολών, που, αν εκτελεστούν στο σύνολο αρχείων κάποιου βασικού λειτουργικού συστήματος, παράγουν μία κατάσταση των αρχείων αυτών, που περιέχει όλα τα απαραίτητα αρχεία και προγράμματα για την εκτέλεση κάποιας εφαρμογής. Τα containers του Docker συνήθως συμπεριφέρονται ως εικονικές μηχανές, από την άποψη του ότι παρέχουν πλήρη απομόνωση του περιβάλλοντος εκτέλεσης της εφαρμογής μέσα στο container από το υπόλοιπο μηχάνημα. Στην πραγματικότητα, όμως, δεν είναι εικονικές μηχανές, λειτουργούν παρέχοντας απομόνωση στο επίπεδο του συστήματος αρχείων (file system). Έτσι, στα πλαίσια της εφαρμογής, παρέχεται το κατάλληλο Dockerfile, ώστε η εφαρμογή να μπορεί τόσο να χτίζεται όσο και να εκτελείται ομοιόμορφα σε κάθε σύστημα, με καμία άλλη απαίτηση, πέραν του Docker.

Για τη βελτιστοποίηση του χώρου που καταναλώνει το image της εφαρμογής, θα εκμεταλλευτεί μία στρατηγική, κατά την οποία χρησιμοποιείται προσωρινή εικόνα με τα build dependencies, η οποία, αφού εκτελέσει το χτίσιμο της εφαρμογής, αντιγράφει τα παραγόμενα αρχεία σε άλλη εικόνα, που περιέχει τα runtime dependencies, όπως το Ansible. Χρησιμοποιώντας την τακτική αυτή, αναπτύσσεται το παρακάτω Dockerfile:

```
##################################
#      Build Environment
##################################
FROM microsoft/aspnetcore-build:2.0 AS build-env
WORKDIR /app

# Copy csproj and restore as distinct layers
COPY Autonet/*.csproj ./
RUN dotnet restore

# Copy everything else
COPY Autonet/ ./

# Install npm depedencies and run gulp tasks to build wwwroot/
RUN npm install
RUN gulp build
RUN gulp min

# Build the .NET project
RUN dotnet publish -c Release -o out

##################################
#      Runtime Environment
##################################
FROM microsoft/aspnetcore:2.0

WORKDIR /app

# Install Ansible
RUN apt-get update && apt-get install --no-install-recommends -y gnupg dirmngr bash \
    && echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main" >>
/etc/apt/sources.list \
    && apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367 \
    && apt-get update \
    && apt-get install --no-install-recommends -y ansible \
    && rm -rf /var/lib/apt/lists/*

ENV ANSIBLE_HOST_KEY_CHECKING False

# Store SSH private key
ARG SSH_PRIVATE_KEY
ENV SSH_PRIVATE_KEY $SSH_PRIVATE_KEY
RUN mkdir -p /root/.ssh && echo "$SSH_PRIVATE_KEY" > /root/.ssh/id_rsa

# Copy the packaged web server
COPY --from=build-env /app/out .

CMD ["dotnet", "Autonet.dll"]
```

Οπότε, το χτίσιμο της εφαρμογής μπορεί να πραγματοποιηθεί εάν, στον ίδιο φάκελο με το παραπάνω Dockerfile, εκτελεστεί η παρακάτω εντολή **docker build**, η οποία θα δημιουργήσει ένα image με όνομα “autonet” βάσει του Dockerfile.

```
$ docker build –build-arg SSH_PRIVATE_KEY=”…” -t autonet .
```

Ύστερα, το image που δημιουργήθηκε μπορεί να χρησιμοποιηθεί για την έναρξη ενός νέου container μέσω της εντολής **docker run**, η οποία, επίσης, θα εκθέσει την θύρα 80 του container στην θύρα 8080 του host μηχανήματος για πρόσβαση στην εφαρμογή.

```
$ docker run -p 8080:80 netauto
```

6.5 Τεχνικό Χρέος

Ο όρος **Technical Debt** (αλλιώς design debt, ή code debt) είναι μία έννοια, στο πλαίσιο της τεχνικής ανάπτυξη ενός έργου λογισμικού, η οποία αναφέρεται στο επιπλέον κόστος, που απαιτείται για την αναδιάταξη, την διόρθωση ή την βελτιστοποίηση ορισμένων κομματιών λογισμικού, τα οποία συνήθως υλοποιήθηκαν εσκεμμένα με υποβέλτιστο τρόπο. Το technical debt, που παράγεται συνειδητά, δεν είναι πάντα κακόηθες ή επιβλαβές για το έργο και συνήθως δημιουργείται ως συνέπεια δόμησης προτεραιοτήτων του έργου. Παραθέτοντας και αναφορά στον Donald Knuth, νικητή βραβείου Turing του 1974, «η πρόωρη βελτιστοποίηση είναι η πηγή όλου του κακού στον προγραμματισμό» [17]. Οπότε, το technical debt δημιουργείται συνήθως για να υλοποιηθεί άμεσα κάποια λειτουργικότητα του έργου, χωρίς την επιβάρυνση της βελτιστοποίησης της υλοποίησης της λειτουργικότητας αυτής, καθώς και η διαδικασία αυτή μπορεί να είναι χρονοβόρα.

Στην περίπτωση της εργασίας αυτής, το technical debt, που θα αναφερθεί σε αυτή την ενότητα, προέκυψε από σχεδιαστικές επιλογές, κατά την ανάπτυξη του λογισμικού, που είχαν ως σκοπό είτε την απλούστευση της απεικόνισης κάποιας διαδικασίας ή τον διαχωρισμό της λειτουργικότητας της εφαρμογής από αυτήν των πιθανών σεναρίων χρήσης. Οι επιλογές αυτές βασίζονται στην άποψη ότι περεταίρω βελτιστοποίηση και προσοχή στη λεπτομέρεια θα απαιτούσαν όγκο λογισμικού, που θα επικάλυπτε την ουσία και τα κύρια σημεία της ανάπτυξης της εργασίας.

- **Φιλτράρισμα ονομάτων και διευθύνσεων στο σχεδιαστικό περιβάλλον του χρήστη.**
  Ο μόνος έλεγχος, που πραγματοποιείται αυτή τη στιγμή στο περιβάλλον αυτό, είναι ο έλεγχος για την εγκυρότητα διευθύνσεων IP, που εισάγει ο χρήστης, που ακολουθείται από κόκκινη ένδειξη γύρω από μη-έγκυρες διευθύνσεις. Επιβάλλεται να πραγματοποιούνται επιπλέον έλεγχοι για τους επιτρεπτούς χαρακτήρες στα ονόματα (hostnames) των δρομολογητών και για το αν κάποια διεύθυνση IP είναι ήδη δεσμευμένη. Οι έλεγχοι αυτοί πρέπει να επαναλαμβάνονται από τον server, ο οποίος θα επικοινωνεί λεπτομέρειες τυχόν σφαλμάτων στον χρήστη.

- **Επικοινωνία σφαλμάτων από τον server στον client.**
  Για να διατηρηθεί το κομμάτι του κώδικα που χειρίζεται τα αιτήματα του client λιτό και ευνόητο, πολλοί έλεγχοι σφαλμάτων ή τμήματα παραγωγής πληροφοριακών μηνυμάτων προς τον χρήστη έχουν παραληφθεί. Σε κάποιο σενάριο χρήσης, όπου ο χρήστης δεν γνωρίζει τις προδιαγραφές της εφαρμογής, όλα τα σενάρια σφάλματος πρέπει να χειρίζονται κατάλληλα και να ενημερώνουν κατάλληλα τον χρήστη.

- **Χρήση της μεταβλητής inventory_hostname στο playbook του Ansible.**
  Η μεταβλητή αυτή χρησιμοποιείται στο playbook για τον προσδιορισμό της τοπικής διεύθυνσης στα tunnel interfaces των δρομολογητών. Έχει γίνει, δηλαδή, η παραδοχή ότι το αρχείο **hosts** θα περιέχει μόνο διευθύνσεις IP, ενώ, στην πραγματικότητα, μπορεί να περιέχει και hostnames, μέσω ανάκτησης ονομάτων από DNS.

- **Αρχικοποίηση δρομολογητών με χρήση του Ansible.**
  Όπως λειτουργεί αυτή τη στιγμή η διαδικασία αρχικοποίησης, η δημόσια διαχειριστική διεύθυνση των δρομολογητών αποδίδεται σε αυτούς μέσω SSH. Αυτό

κάνει το λογισμικό του server να μην είναι εύκολα επεκτάσιμο, καθώς, στην τωρινή του κατάσταση, η προσθήκη νέου είδους δρομολογητή απαιτεί παρέμβαση στον κώδικα της εφαρμογής. Η προσέγγιση αυτή επιλέχθηκε, ώστε να συμπεριληφθεί, στα πλαίσια της εφαρμογής, η διαδικασία πρόσβασης και παραμετροποίησης μέσω SSH, σε έναν δρομολογητή, που δεν υποστηρίζει δυνατότητες εκτέλεσης. Η εφαρμογή ιδανικά θα επεκτεινόταν, ώστε και η αρχικοποίηση να πραγματοποιείται με χρήση του Ansible, επιτρέποντας, έτσι, επέκταση χωρίς παρέμβαση στον κώδικα, καθώς θα απαιτείται μόνο ανάπτυξη νέων playbook.

- **Χρήση software locks.**
  Πολλοί κοινοί πόροι της εφαρμογής, όπως οι δεσμευμένες διευθύνσεις των pool διευθύνσεων και η χρήση των zero-day διευθύνσεων, προστατεύονται από συγκρούσεις με χρήση software locks. Αυτό αφήνει την εφαρμογή ευάλωτη κατά την εκτέλεση πολλαπλών στιγμιότυπων του κώδικα, τα οποία μοιράζονται ESXi hosts ή pools διευθύνσεων, καθώς και τα software locks δεν θα εκτείνονται μεταξύ τους. Για τέτοιες υλοποιήσεις, θα χρειαστεί να ερευνηθεί κάποιος τρόπος, ώστε τα κλειδώματα, για την αποφυγή συγκρούσεων, να πραγματοποιούνται σε χαμηλότερο επίπεδο.

Κεφάλαιο 7

Συμπεράσματα

7.1 Συμπεράσματα

Ολοκληρώνοντας την υλοποίηση της εργασίας, προέκυψαν συμπεράσματα ως προς τις δυνατότητες του software-defined networking (SDN), καθώς και για τις εφαρμογές του. Κάνοντας χρήση της εργασίας αυτής, αναπτύσσονται εικονικά ιδιωτικά δίκτυα (VPN), χωρίς μέριμνα ή προϋπολογισμό του υποκείμενου υλικού (hardware) που θα τα φιλοξενήσει. Αυτό επιτρέπει τη δημιουργία τέτοιων δικτύων και χειροκίνητα αλλά και προγραμματιστικά, απαλείφοντας, έτσι, τη χρονοβόρα και περίπλοκη διαδικασία εγκατάστασης και παραμετροποίησης μίας τοπολογίας σε εικονικούς δρομολογητές. Δοκιμάζοντας, επίσης, την εφαρμογή ως χρήστης από το γραφικό περιβάλλον, οδηγείσαι στο συμπέρασμα ότι ο σχεδιασμός περίπλοκων τοπολογιών ήταν στην πράξη εύκολος και προσιτός από τον μέσο χρήστη. Με την προσθήκη δρομολογητών, αλλά και ζεύξεων μεταξύ τους, να είναι δυνατή με μερικά μόνο κλικ, η περιπλοκότητα της παραμετροποίησης και εγκατάστασης ενός εικονικού δικτύου απαλείφεται σχεδόν πλήρως. Η ιδιότητα αυτή επιφέρει ακαδημαϊκή αξία στην εργασία, επιτρέποντας την εκπαίδευση και τον πειραματισμό πάνω σε περίπλοκες δικτυακές τοπολογίες, αντιμετωπίζοντας τα σχετικά προβλήματα υψηλού επιπέδου και όχι αυτά του υλικού ή των φυσικών περιορισμών.

7.2 Μελλοντικές Επεκτάσεις

Κατά την υλοποίηση της εργασίας, παραλείφθηκαν τμήματα, επιλογές και δυνατότητες που θεωρούνται εκτός εύρους στα πλαίσιά της. Η υλοποίησή τους βασίζεται σε σχεδιαστικές επιλογές, οι οποίες εξαρτώνται άμεσα από το περιβάλλον και τα σενάρια χρήσης της εφαρμογής. Για το λόγο αυτό, οι δυνατότητες αυτές, μαζί με άλλες ιδέες, θα περιγραφούν συνοπτικά σε αυτή την ενότητα ως πιθανές επεκτάσεις για το μέλλον της εφαρμογής.

- **Επικάλυψη του τεχνικού χρέους.**
  Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, η επέκταση της εφαρμογής για χρήση σε παραγωγικό περιβάλλον οφείλει να ολοκληρώσει τις τεχνικές εκκρεμότητες που παραλείφθηκαν στα πλαίσια της ανάπτυξης, με σκοπό να διατηρηθεί η έμφαση στο θεωρητικό και όχι στο κατασκευαστικό κομμάτι. Οι επεκτάσεις στα πλαίσια του τεχνικού χρέους περιέχουν σχεδιάστηκες επιλογές, όπως ο έλεγχος πρόσβασης με λογαριασμούς χρηστών.

- **Πρόσβαση στις εικονικές μηχανές μέσω της ιστοσελίδας.**
  Λογικό επόμενο βήμα στην επέκταση της λειτουργικότητας και των δυνατοτήτων της εφαρμογής είναι η δυνατότητα του χρήστη για πρόσβαση στις δημιουργημένες εικονικές μηχανές, μέσα από το παράθυρο του web browser. Αυτό μπορεί να επιτευχθεί μέσω δυναμικού συνδέσμου, που θα εκτελεί την εφαρμογή PuTTY στο περιβάλλον του χρήστη ή με χρήση κάποιας βιβλιοθήκης όπως η WebSSH2 [21] που μετατρέπει μία ιστοσελίδα σε τερματικό μέσω SSH.

- **Έρευνα για βελτιστοποίηση χρησιμοποίησης πόρων.**
  Η εφαρμογή αυτή επιτρέπει την ανάπτυξη ιδιωτικών δικτύων, που από την οπτική του χρήστη δεν εξαρτώνται από το υποκείμενο υλικό. Εκμεταλλευόμενοι αυτού και αντιμετωπίζοντας το υλικό αφαιρετικά (abstract), μπορεί πλέον να ερευνηθεί ο βέλτιστος καταμερισμός των διαθέσιμων πόρων ως ένα πρόβλημα υψηλού επιπέδου.

- **Παροχή εικονικής υπηρεσίας VPN.**
  Η κύρια λειτουργία της εφαρμογής είναι η αυτοματοποίηση της εγκατάστασης εικονικών δικτυακών τοπολογιών. Μία ωφέλιμη επέκταση της λειτουργικότητας αυτής είναι η εισαγωγή παραμετροποίησης τερματικών χρηστών, είτε με τη μορφή νένω εικονικών μηχανών ή υπάρχων λειτουργικών συστημάτων. Η εφαρμογή έτσι έχει τη δυνατότητα να χρησιμοποιείται για την παροχή εικονικών ιδιωτικών δικτύων ως NFV.

Βιβλιογραφία

[1] YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) https://tools.ietf.org/html/rfc6020

[2] Point-to-Point Tunneling Protocol (PPTP) – Request for Comments: 2637 - http://www.rfc-editor.org/rfc/pdfrfc/rfc2637.txt.pdf

[3] T. Nguyen and X. Lujan, "Method and apparatus for integrating tunnelling protocols with standard routing protocols" https://patents.google.com/patent/US20020016926A1/en

[4] Xen Dom0 https://wiki.xenproject.org/wiki/Dom0

[5] A. Blenk, A. Barsta, M. Reisslein and W. Kellerer, "Survey on Network Virtualization Hypervisors for Software Defined Networking" https://arxiv.org/abs/1506.07275

[6] A. Velte and T. Velte, "Microsoft Virtualization with Hyper-V" https://dl.acm.org/citation.cfm?id=1593830

[7] G. Alessandro and A. Santana, "Data Center Virtualization Fundamentals" https://books.google.gr/books?hl=en&lr=&id=6ZagQNuqWh4C&oi=fnd&pg=PR9&dq=cisco+nexus+1000v&ots=bpzCvnisMY&sig=K7XIQr49Le4OBwsVEZ5DU1hd6hc&redir_esc=y#v=onepage&q&f=false

[8] S. Burke, "VMware Continues Virtualization Market Romp" https://www.crn.com/news/virtualization/232900547/vmware-continues-virtualization-market-romp.htm

[9] M. Foley, "It's a Unix system, I know this!" https://blogs.vmware.com/vsphere/2013/06/its-a-unix-system-i-know-this.html

[10] HP VSR1000 Virtual Services Router Series - Overview https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c03952700

[11] J. Gold, "The 10 most powerful companies in enterprise networking" https://www.networkworld.com/article/3211410/lan-wan/the-10-most-powerful-companies-in-enterprise-networking.html

[12] Cisco Cloud Services Router 1000v Data Sheet https://www.cisco.com/c/en/us/products/collateral/routers/cloud-services-router-1000v-series/datasheet-c78-733443.html

[13] User Guide - VyOS Wiki https://wiki.vyos.net/wiki/User_Guide

[14] D. Hall, "Ansible Configuration Management" https://books.google.gr/books?hl=en&lr=&id=ETQmAgAAQBAJ&oi=fnd&pg=PT5&dq=ansible&ots=CJtVCk8XvH&sig=YT64J5nJwrRlB0E9QVpi8t7E60c&redir_esc=y#v=onepage&q=ansible&f=false

[15] Chef (software) https://en.wikipedia.org/wiki/Chef_(software)

[16] All Modules - Ansible Documentation https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html

[17] D. Knuth "Computer Programming as an Art" https://dl.acm.org/citation.cfm?id=1283929

[18] "A YANG Data Model for Network Topologies" https://tools.ietf.org/html/rfc8345

[19] J. Wettinger, "Unified Invocation of Scripts and Services for Provisioning, Deployment, and Management of Cloud Applications Base on TOSCA" http://www.iaas.uni-stuttgart.de/RUS-data/INPROC-2014-22%20-%20Unified%20Invocation%20of%20Scripts%20and%20Services%20for%20Provisioning,%20Deployment,%20and%20Management%20of%20Cloud%20Applications%20Based%20on%20TOSCA.pdf

[20] vSphere 6.5 Command-Line Documentation https://pubs.vmware.com/vsphere-6-5/index.jsp?topic=%2Fcom.vmware.vsphere.scripting.doc%2FGUID-7F7C5D15-9599-4423-821D-7B1FE87B3A96.html

[21] B.Church, WebSSH2 https://github.com/billchurch/WebSSH2

[22] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S.Azodolmolky and S. Uhlig "Software-Defined Networking: A Comprehensive Survey"

[23] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee "Network Function Virtualization: Challenges and Opportunities for Innovations"

[24] Juniper Networks "A 6-Step Guide to Getting Started With Network Automation"

[25] A. Funk "Virtual Private Network" http://www.infosecwriters.com/Papers/AFunk_VPN.pdf

Appendix A

Πηγαίος Κώδικας της Εφαρμογής

```
// Scripts/dataPort.ts
import Designer from './designer/designer';
import Line from './designer/line';
import Router from './models/router';
import Options from './options';
import * as yaml from 'js-yaml';
function exportObj(routers: Router[]): any
{
    let obj = {
        tosca_definitions_version: 'tosca_simple_yaml_1_0',
        description: 'Automatically generated topology',
        topology_template: {
            node_templates: {
                router: {
                    type: Options.TOSCA_TYPE,
                    capabilities: {
                        host: {
                            num_cpus: 1,
                            disk_size: '10 GB',
                            mem_size: '256 MB'
                        },
                        os: {
                            properties: {
                                architecture: 'x86_64'
                            }
                        }
                    }
                }
            }
        }
    };
    routers.forEach(r =>
    {
        let rObj = {
            type: Options.TOSCA_TYPE,
            properties: {
                router_id: r.RouterId,
                hostname: r.getLabel()
            },
            tunnels: []
        };
        r.getInterfaces().forEach(i =>
        {
            rObj.tunnels.push({
                addr: i.toString(),
                route: r.getRemoteAddressOnTunnel(i).toString(),
                remote_id: r.getRemoteRouterOnTunnel(i).RouterId
            });
        });
        obj.topology_template[r.getLabel()] = rObj;
    });
    return obj
}
function importObj(designer: Designer, topology: any)
{
    designer.reset();
    let routers: Router[] = [];
    let randInt = max => Math.floor(Math.random() * max);
    let routerWithId = id => routers.filter(r => r.RouterId == id)[0];
    /*
```

```
     * Loop over the json object creating the Router instances
     */
    Object.keys(topology.topology_template).forEach(key =>
    {
        if (key == "node_templates")
            return;
        let rObj = topology.topology_template[key];
        let router = new Router(rObj.properties.router_id);
        router.setLabel(rObj.properties.hostname);
        router.position.set(
            randInt(designer.width - router.width),
            randInt(designer.height - router.height)
        );
        routers.push(router);
    });
    /*
     * Loop over the router's tunnels, connecting them to neighbors
     */
    routers.forEach(router =>
    {
        designer.add(router);
        topology.topology_template[router.getLabel()].tunnels.forEach(t =>
        {
            /*
             * Check if this router already has this tunnel due
             * to its neighbor being looked up first
             */
            if (router.getNeighbors().some(r => r.RouterId == t.remote_id))
                return;
            let other = routerWithId(t.remote_id);
            /*
             * Apply the appropriate address both to this router and to its neighbor.
             */
            let line = new Line(router, other);
            router.getInterfaceOn(line).trySetIP(t.addr);
            other.getInterfaceOn(line).trySetIP(t.route);
            designer.add(line);
        });
    });
    designer.autoScale();
}
export function exportJson(routers: Router[]): any
{
    return JSON.stringify(exportObj(routers), null, 4);
}
export function exportTosca(routers: Router[]): string
{
    return yaml.dump(exportObj(routers));
}
export function importJson(designer: Designer, topology: string)
{
    importObj(designer, JSON.parse(topology));
}
export function importTosca(designer: Designer, topology: string)
{
    importObj(designer, yaml.load(topology));
}
// Scripts/options.ts
let options = {
    SELECTION_COLOR: 0xf44242,
    LINE_WIDTH: 8,
    ZOOM_SPEED: 0.16,
    FONT_STYLE: new PIXI.TextStyle({
        fontFamily: "Arial",
        fontSize: 40,
        fill: "black"
```

```
    }),
    ADDR_FONT_STYLE: new PIXI.TextStyle({
        fontFamily: "Arial",
        fontSize: 24,
        fill: "black"
    }),
    MIN_ZOOM: 1,
    MAX_ZOOM: .3,
    TOSCA_TYPE: 'tosca.nodes.Router',
    LOG_CHECK_INTERVAL: 400
};
export default options;
// Scripts/app.ts
import { IPv4 } from 'ip-num/IPv4'
import Designer from './designer/designer';
import { exportTosca, exportJson, importTosca, importJson } from './dataPort';
import * as hljs from 'highlight.js';
let designer: Designer;
function init()
{
    /*
     * Resize canvas to fill the Bootstrap column
     */
    let canvas = $('#designer-canvas');
    canvas.width(canvas.parent().width());
    canvas.height(window.innerHeight * .96);
    designer = new Designer(<HTMLCanvasElement>canvas.get(0));
    designer.load();
}
export function getDesigner(): Designer
{
    return designer;
}
$(document).ready(function ()
{
    init();
    /*
     * Bind sidebar designer control buttons
     */
    $('#add-router-btn').click(() => designer.createRouter());
    $('#auto-scale-btn').click(() => designer.autoScale());
    $('.delete-btn').click(() => designer.deleteSelected());
    $('#export-tosca-btn').click(() =>
    {
        let routers = designer.getNodes();
        let tosca = exportTosca(routers);
        let json = exportJson(routers);
        $('#export-tosca').html(tosca);
        $('#export-json').html(json);
        hljs.highlightBlock($('#export-tosca').get(0));
        hljs.highlightBlock($('#export-json').get(0));
        $('#export-modal').modal('show');
    });
    $('#import-tosca-btn').click(() =>
    {
        importTosca(designer, <string>$('#import-tosca').val());
    });
    $('#import-json-btn').click(() =>
    {
        importJson(designer, <string>$('#import-json').val());
    });
    /*
     * Bind properties input sanitization
     */
    let ipAddr = new RegExp("^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-
9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$");
```

```
    let hostname = new RegExp("^(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]*[a-zA-Z0-
9])\.)*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])$");
    $('#hostname').keypress(function (e)
    {
        var str = String.fromCharCode(!e.charCode ? e.which : e.charCode);
        if (hostname.test($(this).val() + str))
        {
            return true;
        }
        e.preventDefault();
        return false;
    });
    /*
     * Bind create button
     */
    $('#create-btn').click(() =>
    {
        let routers = designer.getNodes();
        $.ajax({
            url: '/api/create',
            method: 'POST',
            data: exportJson(routers),
            contentType: "application/json",
            success: function (resp, xhr, err)
            {
                let topologyId = resp;
                window.location.href = '/topology/' + topologyId;
            },
            error: function (x1, x2, x3)
            {
                console.log(x1);
                console.log(x2);
                console.log(x3);
            }
        });
    });
});
// Scripts/topology.ts
import Options from './options';
var logLine = 0;
var topologyId = () => $('#topology-id').val();
function refreshLog()
{
    $.get({
        url: '/api/logs/' + topologyId(),
        success: function(log)
        {
            while (logLine < log.length)
            {
                let outputWindow = $('#output');
                let line = log[logLine];
                if (line.text)
                {
                    let logLineSpan =
                        $('<pre class="log-line">')
                            .html(line.text.trim())
                            .css({
                                color: line.color
                            });
                    // Append the line
                    outputWindow.append(logLineSpan);
                    // Scroll to the bottom of the page
                    $("html, body").scrollTop($(document).height() - $(window).height());
                }
                logLine++;
            }
```

```
        }
    });
}
function refreshStatus()
{
    $.get({
        url: '/api/topology/' + topologyId(),
        success: function (routers)
        {
            let routersTable = $('#router-status-table');
            routersTable.empty();
            routers.forEach(r =>
            {
                let status = r.online ? "green" : "red";
                routersTable.append(
                    $('<tr style="font-size: x-large">')
                        .append($('<td>').html('<i class="fas fa-power-
off"></i>').addClass(status))
                        .append($('<td colspan="2">').html(r.ip))
                );
                r.tunnels.forEach(t =>
                {
                    routersTable.append(
                        $('<tr>')
                            .append($('<td>'))
                            .append($('<td>').html(t.interface))
                            .append($('<td>').html(t.addr))
                    );
                });
            });
        }
    });
}
function deleteTopology()
{
    $.post({
        url: '/api/delete/' + topologyId()
    });
}
$(document).ready(function ()
{
    // Will run at page load
    setInterval(refreshLog, Options.LOG_CHECK_INTERVAL);
    setInterval(refreshStatus, Options.LOG_CHECK_INTERVAL);
    $('#delete-topology-btn').click(deleteTopology);
});
// Scripts/models/selectable.ts
export default interface Selectable
{
    select(): void;
    unselect(): void;
    remove(): void;
}
// Scripts/models/draggable.ts
export default class Draggable extends PIXI.Container
{
    private dragging: boolean = false;
    private moved: boolean = false;
    constructor(obj: PIXI.DisplayObject)
    {
        super();
        /*
         * Events
         */
        obj.on('mousedown', this.onDragStart.bind(this))
            .on('mouseup', this.onDragEnd.bind(this))
```

```
            .on('mouseupoutside', this.onDragEnd.bind(this))
            .on('mousemove', this.onDragMove.bind(this));
    }
    private onDragStart(evt)
    {
        if (evt.data.button == 0)
        {
            this.dragging = true;
            this.moved = false;
            this.alpha = 1;
            let pointerX: number = evt.data.getLocalPosition(this).x;
            let pointerY: number = evt.data.getLocalPosition(this).y;
            this.pivot.set(pointerX, pointerY);
            this.position.x += pointerX;
            this.position.y += pointerY;
            evt.stopPropagation();
            evt.data.originalEvent.stopPropagation();
            evt.data.originalEvent.preventDefault();
        }
    }
    private onDragEnd()
    {
        if (this.dragging)
        {
            this.dragging = false;
            this.alpha = 1;
            this.position.x -= this.pivot.x;
            this.position.y -= this.pivot.y;
            this.pivot.set(0, 0);
            if (!this.moved)
            {
                this.onPress();
            }
        }
    }
    private onDragMove(evt)
    {
        if (this.dragging)
        {
            let pointerX: number = evt.data.getLocalPosition(this.parent).x;
            let pointerY: number = evt.data.getLocalPosition(this.parent).y;
            this.position.set(pointerX, pointerY);
            this.moved = true;
            this.onMove();
        }
    }
    protected onMove()
    {
    }
    protected onPress()
    {
        console.log('press');
    }
}
// Scripts/models/component.ts
import Selectable from './selectable';
import Draggable from './draggable';
import Line from '../designer/line';
import { getDesigner } from '../app';
import Options from '../options';
/**  */
export default class Component extends Draggable implements Selectable
{
    private sprite: PIXI.Sprite;
    private outline: PIXI.Graphics;
    private label: PIXI.Text;
```

```
private connectors: Line[];
constructor(resourceName: string)
{
    /*
     * Sprite
     */
    let sprite = new PIXI.Sprite(PIXI.loader.resources[resourceName].texture);
    sprite.interactive = true;
    sprite.buttonMode = true;
    super(sprite);
    this.sprite = sprite;
    this.addChild(sprite);
    /*
     * Label
     */
    this.label = new PIXI.Text('', Options.FONT_STYLE);
    this.setLabel(resourceName);
    this.addChild(this.label);
    this.outline = new PIXI.Graphics();
    this.addChild(this.outline);
    this.connectors = [];
}
public setLabel(text: string)
{
    this.label.text = text;
    this.label.pivot.set(this.label.width / 2, 0);
    this.label.position.set(this.sprite.width / 2, this.sprite.height);
}
public center(): PIXI.Point
{
    return new PIXI.Point(
        this.x - this.pivot.x + this.sprite.width / 2,
        this.y - this.pivot.y + this.sprite.height / 2
    );
}
public isConnectedTo(cmp: Component): boolean
{
    return this.connectors.some(c => c.other(this) == cmp);
}
protected onMove()
{
    this.connectors.forEach(c => c.update());
}
protected onPress()
{
    getDesigner().onSelect(this);
}
public select()
{
    this.outline.beginFill(Options.SELECTION_COLOR, 0);
    this.outline.lineStyle(4, Options.SELECTION_COLOR);
    this.outline.drawRect(0, 0, this.sprite.width, this.sprite.height);
}
public unselect()
{
    this.outline.clear();
}
public addConnector(line: Line)
{
    this.connectors.push(line);
}
public removeConnector(line: Line)
{
    let index = this.connectors.indexOf(line);
    if (index != -1)
    {
```

```
            this.connectors.splice(index, 1);
        }
    }
    public remove()
    {
        while (this.connectors.length > 0)
        {
            this.connectors[0].remove();
        }
        getDesigner().remove(this);
    }
    public getLabel(): string
    {
        return this.label.text;
    }
    public update()
    {
        this.connectors.forEach(c => c.update());
    }
}
// Scripts/models/interface.ts
import { IPv4 } from 'ip-num/IPv4'
let ipv4Assignment = new IPv4("10.0.0.2");
export default class Interface
{
    private ip: IPv4;
    constructor()
    {
        this.ip = ipv4Assignment;
        ipv4Assignment = ipv4Assignment.nextIPNumber();
    }
    public get IP(): IPv4
    {
        return this.ip;
    }
    public trySetIP(value: string)
    {
        try
        {
            this.ip = new IPv4(value);
            return true;
        }
        catch
        {
            return false;
        }
    }
    public toString()
    {
        return this.ip.toString();
    }
}
// Scripts/models/router.ts
import Interface from './interface';
import Component from './component';
import Line from '../designer/line';
let routerCount = 1;
interface InterfaceMap
{
    interface: Interface,
    line: Line
}
export default class Router extends Component
{
    private interfaces: InterfaceMap[];
    private routerId: Number;
```

```
constructor(id?: number)
{
    super('router');
    if (id)
    {
        this.routerId = id;
    }
    else
    {
        this.routerId = (routerCount++);
    }
    this.setLabel('router' + this.routerId);
    this.interfaces = [];
}
public get RouterId(): Number
{
    return this.routerId;
}
public addConnector(line: Line)
{
    super.addConnector(line);
    this.interfaces.push({
        line: line,
        interface: new Interface()
    });
}
public removeConnector(line: Line)
{
    super.removeConnector(line);
    let index = -1;
    this.interfaces.forEach(i => {
        if (i.line == line)
            index = this.interfaces.indexOf(i);
    });
    if (index != -1) {
        this.interfaces.splice(index, 1);
    }
}
public get hostname()
{
    return super.getLabel();
}
public set hostname(value: string)
{
    super.setLabel(value);
}
public getInterfaces(): Interface[]
{
    return this.interfaces.map(i => i.interface);
}
public getInterfaceOn(line: Line): Interface
{
    let match = this.interfaces.filter(l => l.line == line);
    if (match.length == 0)
        return null;
    return match[0].interface;
}
public getLineOn(intf: Interface): Line
{
    let match = this.interfaces.filter(l => l.interface == intf);
    if (match.length == 0)
        return null;
    return match[0].line;
}
public getRemoteRouterOnTunnel(intf: Interface): Router
{
```

```
        let line = this.getLineOn(intf);
        if (!line)
            return null;
        return <Router>line.other(this);;
    }
    public getRemoteAddressOnTunnel(intf: Interface): Interface
    {
        let line = this.getLineOn(intf);
        if (!line)
            return null;
        let other = this.getRemoteRouterOnTunnel(intf);
        if (!other)
            return null;
        return other.getInterfaceOn(line);
    }
    public getNeighbors(): Router[]
    {
        return this.interfaces.map(i => this.getRemoteRouterOnTunnel(i.interface));
    }
}
// Scripts/designer/line.ts
import Selectable from '../models/selectable';
import Component from '../models/component';
import Router from '../models/router';
import { getDesigner } from '../app';
import Options from '../options';
/**
 * This class represents a line connecting two components
 * in the designer window.
 */
export default class Line extends PIXI.Graphics implements Selectable
{
    private start: Component;
    private end: Component;
    private color: number = 0x000000;
    constructor(start: Component, end: Component)
    {
        super();
        start.addConnector(this);
        end.addConnector(this);
        this.start = start;
        this.end = end;
        this.update();
        this.on('mousedown', (e) =>
        {
            getDesigner().onSelect(this);
            e.stopPropagation();
            e.data.originalEvent.stopPropagation();
            e.data.originalEvent.preventDefault();
        });
    }
    /** Re-renders he line on the canvas. To be called after each use input event. */
    public update()
    {
        this.clear();
        this.removeChildren();
        this.interactive = true;
        this.buttonMode = true;
        /*
         * Draw line
         */
        this.beginFill(this.color);
        this.lineStyle(Options.LINE_WIDTH, this.color);
        let s = this.start.center();
        let e = this.end.center();
        this.moveTo(s.x, s.y);
```

```
        this.lineTo(e.x, e.y);
        this.endFill();
        /*
         * Create hitbox polygon
         */
        let points = rectanglePoints(s, e, Options.LINE_WIDTH * 2);
        this.hitArea = new PIXI.Polygon([
            points.sLeft,
            points.sRight,
            points.eRight,
            points.eLeft
        ]);
        /*
         * Draw interface IPs
         */
        let offset = 100;
        let unit = new PIXI.Point(e.x - s.x, e.y - s.y);
        let msr = Math.sqrt(unit.x * unit.x + unit.y * unit.y);
        unit.set(unit.x / msr, unit.y / msr);
        let startIf = (<Router>this.start).getInterfaceOn(this);
        if (startIf != null)
        {
            let startIP = new PIXI.Text(startIf.toString(), Options.ADDR_FONT_STYLE);
            startIP.position.set(s.x + offset * unit.x, s.y + offset * unit.y);
            this.addChild(startIP);
        }
        let endIf = (<Router>this.end).getInterfaceOn(this);
        if (endIf != null)
        {
            let endIP = new PIXI.Text(endIf.toString(), Options.ADDR_FONT_STYLE);
            endIP.position.set(e.x - offset * unit.x, e.y - offset * unit.y);
            this.addChild(endIP);
        }
    }
    /**
     * Returns the other end of this line.
     * @param cmp
     */
    public other(cmp: Component): Component
    {
        if (cmp == this.start)
            return this.end;
        else if (cmp == this.end)
            return this.start;
        else
            return null;
    }
    /** Toggle this line as selected, updating its color. */
    public select()
    {
        this.color = Options.SELECTION_COLOR;
        this.update();
    }
    /** Toggle this line as unselected, updating its color. */
    public unselect()
    {
        this.color = 0x000000;
        this.update();
    }
    /** Delete this line from the designer and both of its end components. */
    public remove()
    {
        this.start.removeConnector(this);
        this.end.removeConnector(this);
        getDesigner().remove(this);
    }
```

```
    public points(): Router[]
    {
        return [ <Router>this.start, <Router>this.end ];
    }
}
/**
 * Generate a rectangle on the screen out of this line's start, end and width.
 * Necessary for detecting user input on the line itself in the pixi.js library.
 * @param start
 * @param end
 * @param lineWidth
 */
function rectanglePoints(start, end, lineWidth) : any
{
    let Point = PIXI.Point;
    var half = lineWidth / 2,
        sLeft = {},
        sRight = {},
        eRight = {},
        eLeft = {},
        sX = start.x,
        sY = start.y,
        eX = end.x,
        eY = end.y;
    if ((sX < eX && sY < eY) || // topLeft to bottRight
        (sX > eX && sY > eY)) { // bottRight to topLeft
        sLeft = new Point(sX - half, sY + half);
        sRight = new Point(sX + half, sY - half);
        eRight = new Point(eX + half, eY - half);
        eLeft = new Point(eX - half, eY + half);
    } else if ((sX > eX && sY < eY) || // topRight to bottLeft
        (sX < eX && sY > eY)) { // bottLeft to topRight
        sLeft = new Point(sX - half, sY - half);
        sRight = new Point(sX + half, sY + half);
        eRight = new Point(eX + half, eY + half);
        eLeft = new Point(eX - half, eY - half);
    } else if (sX === eX &&
        (sY > eY || sY < eY)) { // vertical
        sLeft = new Point(sX - half, sY);
        sRight = new Point(sX + half, sY);
        eRight = new Point(eX + half, eY);
        eLeft = new Point(eX - half, eY);
    } else if (sY === eY &&
        (sX < eX || sX > eX)) { // horizontal
        sLeft = new Point(sX, sY + half);
        sRight = new Point(sX, sY - half);
        eRight = new Point(eX, eY - half);
        eLeft = new Point(eX, eY + half);
    }
    return <any>{
        sLeft: sLeft,
        sRight: sRight,
        eRight: eRight,
        eLeft: eLeft
    };
}
// Scripts/designer/designer.ts
import Router from '../models/router';
import Component from '../models/component';
import Selectable from '../models/selectable';
import { KeyboardKey, getWheelDelta } from './input';
import Line from './line'
import Options from '../options';
import * as properties from './properties';
/**
 * This class represents the base of the topology designer,
```

```
 * holding the pixi.js application as well as handling input.
 */
export default class Designer
{
    private app: PIXI.Application;
    private container: PIXI.Container;
    private selected: Selectable;
    private dragging: boolean = false;
    private moved: boolean = false;
    constructor(canvas: HTMLCanvasElement)
    {
        this.app = new PIXI.Application({
            width: canvas.clientWidth,
            height: canvas.clientHeight,
            view: canvas,
            backgroundColor: 0xFFFFFF,
            resolution: 1,
            antialias: true
        });
        /*
        * Place the entire scene in a container
        * So that we can drag it around
        */
        this.reset();
        /*
        * Bind viewport events
        */
        $(canvas).bind('mousewheel DOMMouseScroll', this.onScroll.bind(this));
        $(canvas).bind('contextmenu', (e) => e.preventDefault());
        $(canvas)
            .mousedown((e) =>
            {
                this.moved = false;
                if (e.which == 1)
                    this.dragging = true;
            })
            .mouseup(() =>
            {
                this.dragging = false;
                if (!this.moved)
                {
                    this.unselect();
                }
            })
            .mousemove((e: any) =>
            {
                if (this.dragging)
                {
                    this.container.x += e.originalEvent.movementX /
this.app.stage.scale.x;
                    this.container.y += e.originalEvent.movementY /
this.app.stage.scale.y;
                    this.moved = true;
                }
            });
        /*
         * Bind keys
         */
        let delKey = new KeyboardKey(46);
        delKey.press = (e) => {
            if (this.selected)
            {
                this.selected.remove();
                this.selected = null;
            }
        }
```

```
        this.unselect();
    }
    /** Reset the topology by deleting all routers. */
    public reset()
    {
        if (this.container)
        {
            this.app.stage.removeChild(this.container);
        }
        this.container = new PIXI.Container();
        this.app.stage.addChild(this.container);
    }
    /** Initialize the designer, loading any necessary assets. */
    public load()
    {
        PIXI.loader.add('router', 'images/network/router.png')
            .load(this.setup.bind(this));
    }
    /**
     * Function to be called when the user clicks on a component in the designer.
     * @param cmp
     */
    public onSelect(cmp: Selectable)
    {
        if (this.selected
        && this.selected instanceof Component && cmp instanceof Component
        && !(<Component>this.selected).isConnectedTo(<Component>cmp)
        && this.selected !== cmp)
        {
            let line = new Line(<Component>this.selected, <Component>cmp);
            this.add(line);
            this.selected.unselect();
        }
        else if (this.selected)
        {
            this.selected.unselect();
        }
        this.select(cmp);
    }
    /**
     * Toggle the selection in the designer and update the property editor on the
sidebar.
     * @param cmp
     */
    private select(cmp: Selectable)
    {
        cmp.select();
        this.selected = cmp;
        if (cmp instanceof Router)
        {
            properties.editRouter(<Router>cmp);
        }
        else if (cmp instanceof Line)
        {
            properties.editLink(<Line>cmp);
        }
        else
        {
            properties.hide();
        }
    }
    /** Unselect the currently selected component and hide the property editor on the
sidebar. */
    public unselect()
    {
        if (this.selected)
```

```
            this.selected.unselect();
        this.selected = null;
        properties.hide();
    }
    /** Function that creates an basic initial topology on screen. */
    private setup()
    {
        let r1 = new Router();
        let r2 = new Router();
        let r3 = new Router();
        r2.position.set(310, 256);
        r3.position.set(512, 50);
        let line = new Line(r1, r2);
        this.add(r1, r2, line, r3);
    }
    /**
     * Add objects to the viewport's container.
     * @param children
     */
    public add(...children: PIXI.DisplayObject[])
    {
        children.forEach(c => this.container.addChild(c));
        this.container.children.sort(function (a, b)
        {
            if (a instanceof Component) return 1;
            else return 0;
        });
    }
    /**
     * Remove an object from the viewport's container.
     * @param obj
     */
    public remove(obj: PIXI.DisplayObject)
    {
        this.container.removeChild(obj);
    }
    /** Create a new router at the center of the current viewport */
    public createRouter()
    {
        let router = new Router();
        let width = this.app.screen.width / this.app.stage.scale.x;
        let height = this.app.screen.height / this.app.stage.scale.y;
        router.position.set(
            (width / 2) - this.container.x - this.app.stage.x,
            (height / 2) - this.container.y - this.app.stage.y
        );
        this.add(router);
        this.unselect();
        this.onSelect(router);
    }
    /** Automatically position and scale the screen  */
    public autoScale()
    {
        let focusObj = this.container.children[0];
        this.app.stage.position.set(0, 0);
        this.container.position.set(-focusObj.x, -focusObj.y);
        if (this.container.width <= this.app.screen.width)
        {
            this.app.stage.scale.set(1, 1);
        }
        else
        {
            let scale = Math.max(Options.MAX_ZOOM, this.app.screen.width /
this.container.width);
            this.app.stage.scale.set(scale, scale);
        }
```

```
    }
    /** Mouse wheel callback for zooming */
    private onScroll(evt: WheelEvent)
    {
        evt.stopPropagation();
        evt.preventDefault();
        let stage = this.app.stage;
        let renderer = this.app.renderer;
        let delta = getWheelDelta(evt);
        let zoomFactor = 1 + Options.ZOOM_SPEED;
        if (delta < 0)
        {
            zoomFactor = 1 / zoomFactor;
        }
        // Check if at zoom limits
        if ((delta < 0 && stage.scale.x * zoomFactor < Options.MAX_ZOOM)
            || (delta >= 0 && stage.scale.x * zoomFactor > Options.MIN_ZOOM))
        {
            return;
        }
        // Zoom
        stage.scale.x *= zoomFactor;
        stage.scale.y *= zoomFactor;
        // Center on cursor
        var mouse_loc = renderer.plugins.interaction.eventData.data.global;
        stage.x -= (mouse_loc.x - stage.x) * (zoomFactor - 1);
        stage.y -= (mouse_loc.y - stage.y) * (zoomFactor - 1);
        stage.x = Math.min(0, stage.x);
        stage.y = Math.min(0, stage.y);
    }
    /** Deletes the currently selected component */
    public deleteSelected()
    {
        if (this.selected)
        {
            this.selected.remove();
            this.unselect();
        }
    }
    /** Returns true if the designer canvas is the element under focus in the browser
window. */
    public hasFocus(): boolean
    {
        return document.getElementById('#designer-canvas') === document.activeElement;
    }
    /** Returns an array of all the routers currently in the designer. */
    public getNodes(): Router[]
    {
        return this.container.children.filter(c => c instanceof Router).map(r =>
<Router>r);
    }
    /** Gets the width of the designer in the page. */
    public get width(): number
    {
        return this.app.view.clientWidth;
    }
    /** Gets the height of the designer in the page. */
    public get height(): number
    {
        return this.app.view.clientHeight;
    }
}
// Scripts/designer/properties.ts
import Component from '../models/component';
import Router from '../models/router';
import Interface from '../models/interface';
```

```
import Line from '../designer/line';
let routerProperties = () => $('#router-properties');
let routerInterfaces = () => $('#router-interfaces');
let routerHostname = () => $('#hostname');
let linkProperties = () => $('#link-properties');
let linkDetails = () =>
[
    {
        hostname: $('#edit-if1 > label'),
            addr: $('#edit-if1 > input')
    },
    {
        hostname: $('#edit-if2 > label'),
            addr: $('#edit-if2 > input')
    }
];
/** Hide the currently opened properties window. */
export function hide()
{
    routerProperties().hide();
    linkProperties().hide();
}
/**
 * Update the properties editor window to be displaying and editing the given router.
 * @param router
 */
export function editRouter(router: Router)
{
    hide();
    routerHostname()
        .val(router.hostname)
        .unbind()
        .keyup(() =>
        {
            router.hostname = <string>routerHostname().val();
        });
    routerInterfaces().empty();
    router.getInterfaces().forEach(i =>
    {
        let div = $('<div class="form-group">');
        let inp = $('<input type="text" class="form-control">').val(i.toString());
        bindInputEvent(router, i, inp);
        routerInterfaces().append(
            div.append(inp)
        );
    });
    routerProperties().show();
}
/**
 * Update the properties editor window to be displaying and editing the given link.
 * @param line
 */
export function editLink(line: Line)
{
    hide();
    let cmps = line.points();
    for (let i = 0; i <= 1; i++)
    {
        let router = cmps[i];
        let intf = router.getInterfaceOn(line);
        let label = linkDetails()[i].hostname;
        let inp = linkDetails()[i].addr;
        label.html(router.getLabel());
        inp.val(intf.toString());
        bindInputEvent(router, intf, inp);
    }
```

```
    linkProperties().show();
}
/**
 * Binds an input event to a given HTML text box, which will attempt to edit the given
 * interface on the given router on each update, changing the highlight color of the text
 * box to red when its value is invalid.
 * @param router
 * @param intf
 * @param inp
 */
function bindInputEvent(router: Router, intf: Interface, inp: JQuery<HTMLElement>)
{
    inp.keyup(() =>
    {
        if (intf.trySetIP(<string>inp.val()))
        {
            inp.removeClass('error');
        }
        else
        {
            inp.addClass('error');
        }
        router.update();
    });
}
// Scripts/designer/input.ts
import { getDesigner } from '../app';
/**
 * Wrapper function getting the wheel scroll from a mouse event in
 * methods compatible with different browsers.
 * @param e
 */
export function getWheelDelta(e) : number
{
    if (e.wheelDelta)
    {
        return e.wheelDelta;
    }
    if (e.originalEvent.detail)
    {
        return e.originalEvent.detail * -40;
    }
    if (e.originalEvent && e.originalEvent.wheelDelta)
    {
        return e.originalEvent.wheelDelta;
    }
}
/**
 * Wrapper class for a keyboard key, used for binding events to it.
 */
export class KeyboardKey
{
    code: number;
    private isDown: boolean = false;
    private isUp: boolean = true;
    press: (e: KeyboardEvent) => void;
    release: (e: KeyboardEvent) => void;
    constructor(keyCode: number)
    {
        this.code = keyCode;
        window.addEventListener(
            "keydown",
            event =>
            {
                if (event.keyCode === this.code && getDesigner().hasFocus())
                {
```

```
                    if (this.isUp && this.press) this.press(event);
                    this.isDown = true;
                    this.isUp = false;
                }
            },
            false
        );
        window.addEventListener(
            "keyup",
            event => {
                if (event.keyCode === this.code && getDesigner().hasFocus()) {
                    if (this.isDown && this.release) this.release(event)
                    this.isDown = false;
                    this.isUp = true;
                }
            },
            false
        );
    }
}
// gulpfile.js
/// <binding BeforeBuild='compile' Clean='clean' />
"use strict";
var gulp = require("gulp"),
    concat = require("gulp-concat"),
    cssmin = require("gulp-cssmin"),
    uglify = require("gulp-uglify"),
    merge = require("merge-stream"),
    del = require("del"),
    tsify = require("tsify"),
    less = require("gulp-less"),
    browserify = require('browserify'),
    babel = require('gulp-babel'),
    source = require('vinyl-source-stream'),
    buffer = require('vinyl-buffer'),
    bundleconfig = require("./bundleconfig.json"),
    tsConfig = require('./tsconfig.json');
var regex = {
    css: /\.css$/,
    html: /\.(html|htm)$/,
    js: /\.js$/,
    less: /\.less/
};
/*
 * Full build task to run when without visual studio
 */
gulp.task("build", build);
gulp.task("clean", clean);
gulp.task("min:js", minJs);
gulp.task("min:css", minCss);
gulp.task("min", ["min:js", "min:css"]);
gulp.task("compile:less", compileLess);
gulp.task("compile:typescript", compileTypescript);
gulp.task("compile", ["compile:less", "compile:typescript"]);
function build()
{
    return clean()
        .then(compile)
        .then(min);
}
/*
 * Minifications
 */
function min()
{
    return merge([
```

```
        minJs(),
        minCss()
    ]);
}
function minJs()
{
    var tasks = getBundles(regex.js).map(function (bundle)
    {
        return gulp.src(bundle.inputFiles, { base: "." })
            .pipe(concat(bundle.outputFileName))
            .pipe(uglify())
            .pipe(gulp.dest("."));
    });
    return merge(tasks);
}
function minCss()
{
    var tasks = getBundles(regex.css).map(function (bundle)
    {
        return gulp.src(bundle.inputFiles, { base: "." })
            .pipe(concat(bundle.outputFileName))
            .pipe(cssmin())
            .pipe(gulp.dest("."));
    });
    return merge(tasks);
}
function clean()
{
    var files = bundleconfig.map(function (bundle)
    {
        return bundle.outputFileName;
    });
    return del(files);
}
/*
 * Compilations
 */
function compile()
{
    return merge([
        compileLess(),
        compileTypescript(),
    ]);
}
function compileLess()
{
    return gulp.src('./Styles/**/*.less')
                .pipe(less())
                .pipe(gulp.dest('./wwwroot/css/'));
}
function compileTypescript()
{
    return merge([
        bundleTypescript("./Scripts/app.ts", "app.js"),
        bundleTypescript("./Scripts/topology.ts", "topology.js")
    ]);
}
function bundleTypescript(entrypoint, bundleName)
{
    return browserify()
        .add(entrypoint)                                // Add the entrypoint
        .plugin("tsify", tsConfig.compilerOptions)    // Load the TypeScript plugin
        .bundle()                                       // Compile
        .pipe(source(bundleName))                       // Get the bundle
        .pipe(buffer())                                   // Transform it to a gulp pipe
stream
```

```
        .pipe(babel({                                   // Fix incompatible TS syntax
            presets: ['env']                           // with the older ES2015 standards
        }))
        .pipe(gulp.dest("./wwwroot/js/"));              // Write the bundle
}
gulp.task("watch", function () {
    getBundles(regex.js).forEach(function (bundle) {
        gulp.watch(bundle.inputFiles, ["min:js"]);
    });
    getBundles(regex.css).forEach(function (bundle) {
        gulp.watch(bundle.inputFiles, ["min:css"]);
    });
    getBundles(regex.html).forEach(function (bundle) {
        gulp.watch(bundle.inputFiles, ["min:html"]);
    });
    getBundles(regex.less).forEach(function (bundle) {
        gulp.watch(bundle.inputFiles, ["compile:less"]);
    });
});
function getBundles(regexPattern)
{
    return bundleconfig.filter(function (bundle) {
        return regexPattern.test(bundle.outputFileName);
    });
}
// Styles/designer.less
body {
}
#designer-canvas {
    display: block;
    border: 1px solid rgba(0, 0, 0, .5);
    border-radius: 4px;
}
.sidebar {
    border-radius: 12px;
    background-color: rgba(200, 200, 200, 0.25);
    padding: 1em 2em 1em 2em;
    margin-top: 1em;
    >h1,
    >h2 {
        text-transform: uppercase;
    }
}
.properties {
    display: none;
}
input {
    &.error {
        border-color: red;
    }
    &.error:focus {
        border-color: black;
        box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075), 0 0 8px rgba(255, 0, 0, 0.6);
    }
}
#export-modal {
    .modal-dialog {
        overflow-y: initial !important
    }
    .tab-content {
        height: 600px;
        overflow-y: auto;
    }
}
#import-modal {
    .export-tab {
```

```
        margin: 1em;
    }
}
// Styles/topology.less
.log-container {
    background-color: #303030;
    margin-bottom: 2em;
    padding: 1em;
    .log-line {
        display: block;
        margin: 0;
        overflow-wrap: normal;
    }
}
.green {
    color: green;
}
.red {
    color: red;
}
.topology-sidebar {
    position: fixed;
    display: block;
}
#router-status-table {
    width: 100%;
}
<!-- Views/_ViewImports.cshtml -->
@using Autonet
@using Autonet.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
<!-- Views/_ViewStart.cshtml -->
@{
    Layout = "_Layout";
}
<!-- Views/Home/Topology.cshtml -->
@{
    ViewData["Title"] = "Topology";
    ViewData["script"] = "~/js/topology.js";
    ViewData["script_min"] = "~/js/topology.min.js";
}
<input type="hidden" value="@ViewBag.TopologyId" id="topology-id" />
<div class="container-fluid">
    <div class="row h-100">
        <div class="col-md-3">
            <div class="topology-sidebar">
                <h2>Topology @ViewBag.TopologyId</h2>
                <table id="router-status-table" class="table table-hover table-
striped"></table>
                <button id="delete-topology-btn" class="btn btn-outline-danger center-
block">Delete</button>
            </div>
        </div>
        <div class="col-md-9">
            <div class="log-container">
                <div id="output">
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Views/Home/Index.cshtml -->
@{
    ViewData["Title"] = "Home Page";
    ViewData["script"] = "~/js/app.js";
    ViewData["script_min"] = "~/js/app.min.js";
```

```
}
<!-- Views/Home/Index.cshtml -->
<div class="container-fluid body-content">
    <div class="row">
        <div class="col-md-2">
            <div class="container-fluid">
                <div class="sidebar sidebar-left">
                    <h2>Toolbar</h2>
                    <hr />
                    <button id="add-router-btn" class="btn btn-block btn-lg btn-outline-
primary">
                        <i class="far fa-fw fa-plus-square"></i> Router
                    </button>
                    <button id="auto-scale-btn" class="btn btn-block btn-lg btn-outline-
primary">
                        <i class="fas fa-fw fa-desktop"></i> Auto Scale
                    </button>
                    <hr />
                    <button id="export-tosca-btn" class="btn btn-block btn-lg btn-
outline-primary">
                        <i class="fas fa-download"></i> Export
                    </button>
                    <button id="import-tosca-btn" class="btn btn-block btn-lg btn-
outline-primary"
                            data-toggle="modal" data-target="#import-modal">
                        <i class="fas fa-upload"></i> Import
                    </button>
                    <hr />
                    <button id="create-btn" class="btn btn-block btn-lg btn-primary">
                        <i class="fas fa-rocket"></i> Deploy
                    </button>
                </div>
            </div>
        </div>
        <div class="col-md-8">
            <canvas id="designer-canvas"></canvas>
        </div>
        <div class="col-md-2">
            <div class="container-fluid">
                <div id="router-properties" class="sidebar sidebar-right properties">
                    <h2>Router</h2>
                    <hr />
                    <div class="form-group">
                        <label for="hostname">Hostname</label>
                        <input type="text" class="form-control" id="hostname" />
                    </div>
                    <div class="form-group">
                        <label for="router-type">Operating System</label>
                        <select class="form-control" id="router-type">
                            <option value="vyos">VyOS</option>
                        </select>
                    </div>
                    <hr />
                    <h4>Interfaces</h4>
                    <div id="router-interfaces">
                    </div>
                    <hr />
                    <div class="form-group">
                        <button class="btn btn-block btn-lg btn-outline-danger delete-
btn">
                            <i class="fas fa-times"></i> Delete
                        </button>
                    </div>
                </div>
                <div id="link-properties" class="sidebar sidebar-right properties">
                    <h2>Link</h2>
```

```
                    <hr />
                    <div class="form-group" id="edit-if1">
                        <label for="if-ip1" id="if-host1">Hostname1</label>
                        <input type="text" class="form-control" id="if-ip1" />
                    </div>
                    <div class="form-group" id="edit-if2">
                        <label for="if-ip2" id="if-host2">Hostname1</label>
                        <input type="text" class="form-control" id="if-ip2" />
                    </div>
                    <hr />
                    <div class="form-group">
                        <button class="btn btn-block btn-lg btn-outline-danger delete-
btn">
                            <i class="fas fa-times"></i> Delete
                        </button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Views/Shared/Error.cshtml -->
@model ErrorViewModel
@{
    ViewData["Title"] = "Error";
}
<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>
@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}
<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more detailed
information about the error that occurred.
</p>
<p>
    <strong>Development environment should not be enabled in deployed
applications</strong>, as it can result in sensitive information from exceptions being
displayed to end users. For local debugging, development environment can be enabled by
setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to
<strong>Development</strong>, and restarting the application.
</p>
<!-- Views/Shared/_Layout.cshtml -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Autonet</title>
    <environment include="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
        <link rel="stylesheet" href="~/lib/fontawesome/css/fontawesome-all.css" />
        <link rel="stylesheet"
              asp-href-include="~/css/**/*.css"
              asp-href-exclude="~/css/*.min.css"
              asp-append-version="true" />
    </environment>
    <environment exclude="Development">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.1.1/css/bootstrap.min.css"
              asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
```

```
            asp-fallback-test-class="sr-only" asp-fallback-test-property="position"
asp-fallback-test-value="absolute" />
        <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.13/css/all.css"
            asp-fallback-href="~/lib/fontawesome/css/fontawesome-all.min.css"
            asp-fallback-test-class="fa" asp-fallback-test-property="position" asp-
fallback-test-value="absolute" />
        <link rel="stylesheet" href="~/css/bundle.min.css"/>
    </environment>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/9.12.0/styles/github.min.css"
/>
</head>
<body>
    @await Html.PartialAsync("~/Views/Shared/Modals.cshtml")
    @RenderBody()
    <environment include="Development">
        <script src="~/lib/jquery/dist/jquery.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.js"></script>
        <script src="~/lib/pixi/pixi.js"></script>
        <script src="@ViewData["script"]" asp-append-version="true"></script>
    </environment>
    <environment exclude="Development">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"
            asp-fallback-src="~/lib/jquery/dist/jquery.min.js"
            asp-fallback-test="window.jQuery">
        </script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.1.1/js/bootstrap.bundle.js"
            asp-fallback-src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.fn &&
window.jQuery.fn.modal">
        </script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/pixi.js/4.7.1/pixi.min.js"
            asp-fallback-src="~/lib/pixi/pixi.min.js"
            asp-fallback-test="PIXI.Container">
        </script>
        <script src="@ViewData["script_min"]" asp-append-version="true"></script>
    </environment>
    @RenderSection("Scripts", required: false)
</body>
</html>
<!-- Views/Shared/Modals.cshtml -->
<div class="modal" id="export-modal">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <!-- Modal Header -->
            <div class="modal-header">
                <h4 class="modal-title">Export Topology TOSCA</h4>
                <button type="button" class="close" data-dismiss="modal">&times;</button>
            </div>
            <!-- Modal body -->
            <div class="modal-body">
                <ul class="nav nav-pills nav-justified">
                    <li class="nav-item"><a class="nav-link active" data-toggle="pill"
href="#export-tosca-tab">YAML</a></li>
                    <li class="nav-item"><a class="nav-link" data-toggle="pill"
href="#export-json-tab">JSON</a></li>
                </ul>
                <div class="tab-content">
                    <div id="export-tosca-tab" class="tab-pane container active">
                        <pre class="export-tab"><code class="yaml" id="export-
tosca"></code></pre>
                    </div>
                    <div id="export-json-tab" class="tab-pane container">
```

```
                        <pre class="export-tab"><code class="json" id="export-
json"></code></pre>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<div class="modal" id="import-modal">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <!-- Modal Header -->
            <div class="modal-header">
                <h4 class="modal-title">Import Topology TOSCA</h4>
                <button type="button" class="close" data-dismiss="modal">&times;</button>
            </div>
            <!-- Modal body -->
            <div class="modal-body">
                <ul class="nav nav-pills nav-justified">
                    <li class="nav-item"><a class="nav-link active" data-toggle="pill"
href="#import-tosca-tab">YAML</a></li>
                    <li class="nav-item"><a class="nav-link" data-toggle="pill"
href="#import-json-tab">JSON</a></li>
                </ul>
                <div class="tab-content">
                    <div id="import-tosca-tab" class="tab-pane container text-center
active">
                        <textarea class="export-tab form-control" rows="10" id="import-
tosca"></textarea>
                        <button id="import-tosca-btn" type="button" class="btn btn-
outline-success" data-dismiss="modal">Import</button>
                    </div>
                    <div id="import-json-tab" class="tab-pane container text-center">
                        <textarea class="export-tab form-control" rows="10" id="import-
json"></textarea>
                        <button id="import-json-btn" type="button" class="btn btn-
outline-success" data-dismiss="modal">Import</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Views/Shared/_ValidationScriptsPartial.cshtml -->
<environment include="Development">
    <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment exclude="Development">
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js"
            asp-fallback-src="~/lib/jquery-validation/dist/jquery.validate.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator"
            crossorigin="anonymous"
            integrity="sha384-
Fnqn3nxp3506LP/7Y3j/25BlWeA3PXTyT1l78LjECcPaKCV12TsZP7yyMxOe/G/k">
    </script>
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.6/jquery.validate.
unobtrusive.min.js"
            asp-fallback-src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator &&
window.jQuery.validator.unobtrusive"
            crossorigin="anonymous"
```

```
            integrity="sha384-
JrXK+k53HACyavUKOsL+NkmSesD2P+73eDMrbTtTk0h4RmOF8hF8apPlkp26JlyH">
    </script>
</environment>
// Extensions.cs
using Ansible;
using Autonet.Service;
using Renci.SshNet;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using System.Diagnostics;
namespace Autonet
{
    /// <summary>
    /// Helper extension methods.
    /// </summary>
    public static class Extensions
    {
        /// <summary>
        /// Wrapper for running SSH commands asynchronously.
        /// </summary>
        /// <param name="client"></param>
        /// <param name="command"></param>
        /// <returns></returns>
        public static Task<string> RunCommandAsync(this SshClient client, string command)
        {
            SshCommand cmd = client.CreateCommand(command);
            return Task.Factory.FromAsync(cmd.BeginExecute(), cmd.EndExecute);
        }
        /// <summary>
        /// Wrapper for running SSH commands asynchronously.
        /// </summary>
        /// <param name="client"></param>
        /// <param name="format"></param>
        /// <param name="args"></param>
        /// <returns></returns>
        public static Task<string> RunCommandAsync(this SshClient client, string format,
params object[] args)
        {
            return client.RunCommandAsync(string.Format(format, args));
        }
        /// <summary>
        /// Gets the IP address that sequentially follows
        /// the one given by incrementing by one.
        /// </summary>
        /// <param name="ipAddress"></param>
        /// <returns></returns>
        public static IPAddress Next(this IPAddress ipAddress)
        {
            byte[] nextAddress = BitConverter.GetBytes(ipAddress.ToUint() + 1);
            return IPAddress.Parse(string.Join(".", nextAddress.Reverse()));
        }
        /// <summary>
        /// Converts an IP address to an unsigned integer.
        /// </summary>
        /// <param name="ipAddress"></param>
        /// <returns></returns>
        public static uint ToUint(this IPAddress ipAddress)
        {
            byte[] addressBytes = ipAddress.GetAddressBytes().Reverse().ToArray();
            return BitConverter.ToUInt32(addressBytes, 0);
        }
        /// <summary>
```

```
        /// Clone an IP address object to avoid shared references.
        /// </summary>
        /// <param name="ipAddress"></param>
        /// <returns></returns>
        public static IPAddress Clone(this IPAddress ipAddress)
        {
            return IPAddress.Parse(ipAddress.ToString());
        }
        /// <summary>
        /// Shortcut to enqueuing multiple elements at once in a Queue<T>.
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="queue"></param>
        /// <param name="collection"></param>
        public static void Enqueue<T>(this Queue<T> queue, IEnumerable<T> collection)
        {
            foreach (T item in collection)
                queue.Enqueue(item);
        }
        /// <summary>
        /// Wrapper extension method for executing an Ansible command using the ansible-
net
        /// library asynchronously.
        /// </summary>
        /// <param name="cmd"></param>
        /// <param name="logger"></param>
        /// <returns></returns>
        public static async Task ExecuteAsync(this Playbook cmd, Logger logger)
        {
            ProcessStartInfo startInfo = new ProcessStartInfo
            {
                WorkingDirectory = cmd.WorkingDirectory,
                FileName = cmd.CommandName,
                Arguments = cmd.Name,
                CreateNoWindow = true,
                UseShellExecute = false,
                RedirectStandardOutput = true,
                RedirectStandardError = true,
            };
            Process commandProcess = new Process { StartInfo = startInfo };
            commandProcess.OutputDataReceived += (sender, args) =>
logger?.Log(args.Data);
            commandProcess.Start();
            commandProcess.BeginOutputReadLine();
            try
            {
                string error = await commandProcess.StandardError.ReadToEndAsync();
                logger.Error(error);
            }
            catch { }
            commandProcess.WaitForExit();
        }
    }
}
// Program.cs
using Autonet.Service;
using Autonet.Models;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System.Linq;
using System;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
```

```
using YamlDotNet.Serialization;
using YamlDotNet.Serialization.NamingConventions;
namespace Autonet
{
    public class Program
    {
        /// <summary>
        /// Globally visible class used to hold the application's options
        /// deserialized from the options.yml file.
        /// </summary>
        public static Options Options;
        /// <summary>
        /// The entrypoint of the application.
        /// Loads he options file and runs the ASP.NET MVC application.
        /// </summary>
        /// <param name="args"></param>
        public static void Main(string[] args)
        {
            Options = new DeserializerBuilder()
                .WithNamingConvention(new CamelCaseNamingConvention())
                .Build()
                .Deserialize<Options>(new StreamReader("options.yml"));
            TopologyController.Initialize(Options.LoadSshKey().Result);
            Cleanup().Wait();
            BuildWebHost(args).Run();
        }
        static async Task Cleanup()
        {
            foreach (EsxiHost host in Options.Hosts)
            {
                EsxiClient client = await Options.GetEsxiClient(host.Name);
                await client.Connect();
                foreach (VirtualMachine vm in await client.ListVirtualMachines())
                {
                    if (IsAutonetVM(vm))
                    {
                        await client.RemoveVm(vm.ID);
                    }
                }
                await client.Disconnect();
            }
        }
        /// <summary>
        /// Returns true if the given vm is part of a topology generated by this tool.
        /// Used to avoid removing unecessary VMs during cleanup.
        /// </summary>
        /// <param name="vm"></param>
        /// <returns></returns>
        static bool IsAutonetVM(VirtualMachine vm)
        {
            return vm.File.Trim().StartsWith("vms");
        }
        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .Build();
    }
}
// Startup.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
```

```
using Microsoft.Extensions.DependencyInjection;
using Newtonsoft.Json;
namespace Autonet
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }
        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            services
            .AddMvc()
            .AddJsonOptions(opt =>
            {
                opt.SerializerSettings.Converters.Add(new IPAddressConverter());
                opt.SerializerSettings.NullValueHandling = NullValueHandling.Ignore;
            });
        }
        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseBrowserLink();
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }
            app.UseStaticFiles();
            app.UseMvc();
        }
    }
}
// Service/VMClient.cs
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Renci.SshNet;
using Renci.SshNet.Common;
namespace Autonet.Service
{
    /// <summary>
    /// This class is a wrapper of the basic fields, properties and methods
    /// used to open and maintain an SSH connection as well as to send command through
    /// it while logging the output before return it.
    /// </summary>
    public abstract class VMClient
    {
        protected SshClient client;
        public Logger Logger { get; set; }
        public bool IsConnected => client.IsConnected;
        protected VMClient(string host, string username, string privateKey)
        {
```

```
            PrivateKeyFile keyFile = new PrivateKeyFile(new
MemoryStream(Encoding.ASCII.GetBytes(privateKey)));
            client = new SshClient(host, username, keyFile);
            client.ConnectionInfo.Timeout = TimeSpan.FromSeconds(2);
            Logger = new ConsoleLogger();
        }
        public virtual async Task Connect()
        {
            await Task.Factory.StartNew(() => client.Connect());
        }
        public virtual async Task Disconnect()
        {
            try
            {
                await Task.Factory.StartNew(() => client.Disconnect(), new
CancellationTokenSource(500).Token);
            }
            catch (SshOperationTimeoutException) { }
            catch (Exception ex) { Logger.Error(ex.ToString()); }
        }
        public virtual async Task<bool> TryConnect()
        {
            try
            {
                await Connect();
            }
            catch (SocketException) { return false; }
            catch (SshOperationTimeoutException) { return false; }
            return true;
        }
        protected virtual async Task<string> RunCommand(string command)
        {
            string stdout = await client.RunCommandAsync(command);
            Logger?.Command("$ " + command);
            Logger?.Log(stdout);
            return stdout;
        }
        protected Task<string> RunCommand(string format, params object[] args)
        {
            return RunCommand(string.Format(format, args));
        }
    }
}
// Service/Topology.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using Autonet.Models;
using System.Collections;
namespace Autonet.Service
{
    /// <summary>
    /// This class is meant to be a serializable object that holds a private network.
    /// It holds the routers of the network as well as a table matching these routers
    /// to their assigned ESXi host machine.
    /// It also holds a Logger object to which output and debug messages regarding
    /// the initialization process of this network are written.
    /// </summary>
    [JsonObject(MemberSerialization.OptIn)]
    public class Topology : IEnumerable<Router>
    {
        [JsonProperty("id")]
```

```
        public readonly int ID;
        [JsonProperty("topology")]
        List<Router> routers = new List<Router>();
        Dictionary<Router, EsxiClient> routerEsxiMatch = new Dictionary<Router,
EsxiClient>();
        public Logger Logger { get; set; } = new OutputLogger();
        public volatile bool Deleted;
        /// <summary>
        /// Gets a router in this topology by its hostname.
        /// </summary>
        /// <param name="routerHostname"></param>
        /// <returns></returns>
        public Router this[string routerHostname]
        {
            get { return routers.FirstOrDefault(r => r.Hostname == routerHostname); }
        }
        /// <summary>
        /// Gets a router in this topology by its router id.
        /// </summary>
        /// <param name="routerId"></param>
        /// <returns></returns>
        public Router this[int routerId]
        {
            get { return routers.FirstOrDefault(r => r.RouterId == routerId); }
        }
        public Topology(int id)
        {
            ID = id;
        }
        /// <summary>
        /// Add a router to this topology together with the ESXi machine that will host
it.
        /// </summary>
        /// <param name="router"></param>
        /// <param name="esxi"></param>
        public void Add(Router router, EsxiClient esxi)
        {
            routers.Add(router);
            routerEsxiMatch.Add(router, esxi);
        }
        IEnumerator IEnumerable.GetEnumerator()
        {
            return ((IEnumerable<Router>)routers).GetEnumerator();
        }
        public IEnumerator<Router> GetEnumerator()
        {
            return ((IEnumerable<Router>)routers).GetEnumerator();
        }
        /// <summary>
        /// Get a list of the routers in this topology.
        /// </summary>
        /// <returns></returns>
        public List<Router> GetRouters()
        {
            return routers.ToList();
        }
        /// <summary>
        /// Get the ESXi host machine that has been assigned to a given router.
        /// </summary>
        /// <param name="router"></param>
        /// <returns></returns>
        public EsxiClient GetEsxiClientForRouter(Router router)
        {
            return routerEsxiMatch[router];
        }
    }
```

```
}
// Service/AnsibleInventory.cs
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Ansible;
using YamlDotNet.Serialization;
using Autonet.Models;
namespace Autonet.Service
{
    /// <summary>
    /// This class is used to generate a dynamic Ansible inventory
    /// for the configuration of a given topology.
    /// </summary>
    public class AnsibleInventory
    {
        const string PLAYBOOK_FILE = "playbook.yml";
        string targetDir;
        Dictionary<string, List<Router>> hostGroups;
        public AnsibleInventory(int topologyId)
        {
            hostGroups = new Dictionary<string, List<Router>>();
            targetDir = Path.Combine(Options.AnsibleTargetConfigurationDir, "playbook_" +
topologyId);
        }
        /// <summary>
        /// Add a host to the inventory.
        /// <para>
        /// This method will add its public IP to the list of hosts, as well as its
        /// tunnel and routing variables to the host_vars directory of the inventory.
        /// </para>
        /// </summary>
        /// <param name="type"></param>
        /// <param name="host"></param>
        public void AddHost(string type, Router host)
        {
            if (!hostGroups.ContainsKey(type))
                hostGroups.Add(type, new List<Router>());
            hostGroups[type].Add(host);
        }
        /// <summary>
        /// Export the inventory, along with the playbook, to a generated temporary
        /// path that should not conflict with that of other topologies.
        /// </summary>
        /// <returns>The path to the exported playbook directory.</returns>
        public async Task<string> Export()
        {
            string sourceDir = Options.AnsibleBaseConfigurationDir;
            /*
             * Copy the base Ansible files from the project onto a temp directory
             */
            CopyDirectoryRecursive(sourceDir, targetDir);
            /*
             * Generate and write the hosts file
             */
            string hostsFile = Path.Combine(targetDir, "inventory/hosts");
            await File.WriteAllTextAsync(hostsFile, GetHostsFile());
            /*
             * Generate and writ the host_vars files
             */
            string hostVarsDir = Path.Combine(targetDir, "inventory/host_vars");
            Directory.CreateDirectory(hostVarsDir);
            foreach (Router host in GetHosts())
```

```
            {
                string hostVarsFile = Path.Combine(hostVarsDir, host.PublicIP.ToString()
+ ".yml");
                string routerVarsYaml = new Serializer().Serialize(host);
                await File.WriteAllTextAsync(hostVarsFile, routerVarsYaml);
            }
            return targetDir;
        }
        /// <summary>
        /// Delete the temporary playbook directory that has been exported by this
instance.
        /// </summary>
        public void Delete()
        {
            try
            {
                // The OS sometimes doesn't like this.
                Directory.Delete(targetDir, true);
            }
            catch { }
        }
        public List<Router> GetHosts()
        {
            List<Router> routers = new List<Router>();
            foreach (KeyValuePair<string, List<Router>> group in hostGroups)
            {
                routers.AddRange(group.Value);
            }
            return routers;
        }
        /// <summary>
        /// Get the command that will execute the playbook at the exported directory.
        /// </summary>
        /// <returns></returns>
        public Playbook GetPlaybookCommand()
        {
            Playbook playbookCmd = new Playbook("playbook.yml");
            playbookCmd.WorkingDirectory = targetDir;
            return playbookCmd;
        }
        /// <summary>
        /// Generate the hosts ini file for the inventory.
        /// Will contain all the hosts currently held by this class,
        /// placed under their respective groups.
        /// </summary>
        /// <returns></returns>
        string GetHostsFile()
        {
            StringBuilder hostsFile = new StringBuilder();
            foreach (KeyValuePair<string, List<Router>> group in hostGroups)
            {
                string groupName = Program.Options.GetRouterTypeAnsibleGroup(group.Key);
                hostsFile.AppendFormat("[{0}]\n", groupName);
                foreach (Router host in group.Value)
                {
                    hostsFile.AppendLine(host.PublicIP.ToString());
                }
                hostsFile.AppendLine();
            }
            return hostsFile.ToString();
        }
        static void CopyDirectoryRecursive(string source, string target)
        {
            CopyDirectoryRecursive(new DirectoryInfo(source), new DirectoryInfo(target));
        }
        static void CopyDirectoryRecursive(DirectoryInfo source, DirectoryInfo target)
```

```
        {
            foreach (DirectoryInfo dir in source.GetDirectories())
                CopyDirectoryRecursive(dir, target.CreateSubdirectory(dir.Name));
            foreach (FileInfo file in source.GetFiles())
                file.CopyTo(Path.Combine(target.FullName, file.Name), true);
        }
    }
}
// Service/VyosClient.cs
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Renci.SshNet;
using Renci.SshNet.Common;
namespace Autonet.Service
{
    /// <summary>
    /// This class is meant to hold an active SSH connection to a VyOS
    /// router and to send commands to it.
    ///
    /// This implementation was tricky, as the VyOS environment doesn't have
    /// proper SSH exec capabilities, meaning that input and output towards it
    /// needs to be implemented through streams.
    /// </summary>
    public class VyosClient : VMClient
    {
        const string INTERFACE = "eth0";
        ShellStream shell;
        public VyosClient(string host, string username, string privateKey)
            : base(host, username, privateKey)
        {
        }
        public override async Task Connect()
        {
            await base.Connect();
            Dictionary<TerminalModes, uint> terms = new Dictionary<TerminalModes,
uint>();
            terms.Add(TerminalModes.ECHO, 53);
            shell = client.CreateShellStream("xterm", 80, 24, 800, 600, 1024, terms);
            await ReadOutput("");
        }
        public override async Task Disconnect()
        {
            try
            {
                shell.Dispose();
            }
            catch (SshOperationTimeoutException) { }
            await base.Disconnect();
        }
        /// <summary>
        /// Enters VyOS configuration mode.
        /// </summary>
        /// <returns></returns>
        async Task EnterConfiguration()
        {
            await RunCommand("configure");
        }
        /// <summary>
        /// Commits the current configuration and saves it on the disk.
        /// Does everything on a single command because changing the router's
```

```
        /// address may cause the SSH connection to be dropped.
        /// </summary>
        /// <returns></returns>
        async Task SaveConfiguration()
        {
            await RunCommand("commit; save; exit");
        }
        /// <summary>
        /// Run a command on the router by writing it on the input stream followed by a
newline feed.
        /// </summary>
        /// <param name="command"></param>
        /// <returns></returns>
        protected override async Task<string> RunCommand(string command)
        {
            Logger?.Command("$ " + command);
            byte[] buff = Encoding.ASCII.GetBytes(command + "\n");
            await shell.WriteAsync(buff, 0, buff.Length, new
CancellationTokenSource(500).Token);
            await shell.FlushAsync(new CancellationTokenSource(400).Token);
            return await ReadOutput(command);
        }
        /// <summary>
        /// Read any output left on the shell's output stream.
        /// The implementation is a bit odd since the ssh library used
        /// doesn't seem to work properly when trying to read a binary
        /// stream from the shell.
        /// </summary>
        /// <param name="command"></param>
        /// <returns></returns>
        async Task<string> ReadOutput(string command)
        {
            StringBuilder output = new StringBuilder();
            while (true)
            {
                string line = shell.ReadLine(TimeSpan.FromMilliseconds(400));
                if (line == null)
                    break;
                if (!line.Contains(command) && !line.Contains("[edit]"))
                {
                    output.AppendLine(line);
                }
            }
            Logger?.Log(output.ToString());
            return output.ToString();
        }
        /// <summary>
        /// Configure an address on this router's primary interface.
        /// </summary>
        /// <param name="ipAddress"></param>
        /// <param name="suffix"></param>
        /// <param name="gateway"></param>
        /// <returns></returns>
        public async Task ConfigureAddress(IPAddress ipAddress, int suffix, IPAddress
gateway)
        {
            await EnterConfiguration();
            /*
             * Clear the interface
             */
            await RunCommand("delete interfaces ethernet {0}", INTERFACE);
            /*
             * Set default gateway
             */
            await RunCommand("set system gateway-address {0}", gateway.ToString());
            /*
```

```
             * Set the the interface address
             */
            await RunCommand("set interfaces ethernet {0} address {1}/{2}", INTERFACE,
ipAddress.ToString(), suffix);
            await RunCommand("show interfaces ethernet eth0");
            await SaveConfiguration();
        }
        /// <summary>
        /// Configure an address on this router's primary interface.
        /// </summary>
        /// <param name="ipAddress"></param>
        /// <param name="suffix"></param>
        /// <param name="gateway"></param>
        /// <returns></returns>
        public Task ConfigureAddress(string ipAddress, int suffix, string gateway)
        {
            return ConfigureAddress(IPAddress.Parse(ipAddress), suffix,
IPAddress.Parse(gateway));
        }
    }
}
// Service/TopologyController.cs
using System;
using System.Net;
using System.Net.NetworkInformation;
using System.Threading;
using System.Collections.Generic;
using System.Collections.Concurrent;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using YamlDotNet.Serialization;
using Ansible;
using Autonet.Models;
namespace Autonet.Service
{
    /// <summary>
    /// This static globally-accessible class is responsible for the general network
orchestration.
    /// Provides public methods for assigning addresses to virtual machines as well as
setting them up
    /// on their assigned ESXi hosts.
    /// Also holds the topologies in a thread-safe data type, so as to safely be accessed
from the
    /// web server's API handler methods concurrently.
    /// </summary>
    public static class TopologyController
    {
        const int SSH_ATTEMPT_INTERVAL = 4;
        static int idPool;
        static ConcurrentDictionary<int, Topology> topologies;
        static string sshKey;
        public static void Initialize(string sshPrivateKey)
        {
            idPool = 0;
            topologies = new ConcurrentDictionary<int, Topology>();
            sshKey = sshPrivateKey;
        }
        /// <summary>
        /// Parses the JSON object passed by the front-end of the application and
generate the routers
        /// contained in the model as well as the tunnels/links between them.
        /// During this process it also assigns each router to an ESXi host, as well as a
public IP
        /// address from the specific host's pool.
```

```
        /// </summary>
        /// <param name="topologyJson"></param>
        /// <returns></returns>
        public static async Task<Topology> Deserialize(JObject topologyJson)
        {
            int id = Interlocked.Increment(ref idPool);
            Topology topology = new Topology(id);
            /*
             * Deserialize routers and their tunnels, while assigning addresses from the
pool.
             */
            foreach (KeyValuePair<string, JToken> routerObj in
topologyJson.Value<JObject>("topology_template"))
            {
                if (routerObj.Key == "node_templates")
                {
                    // Exclude the list of templates
                    continue;
                }
                Router router = new Router()
                {
                    Hostname = routerObj.Value["properties"].Value<string>("hostname"),
                    RouterId = routerObj.Value["properties"].Value<int>("router_id"),
                    Tunnels =
JsonConvert.DeserializeObject<List<Tunnel>>(routerObj.Value["tunnels"].ToString())
                };
                EsxiClient esxi = await PickHost(router);
                IPAddress assignedAddress = await
esxi.GetAddressPool().TryAssignAddress();
                if (assignedAddress == null)
                {
                    return null;
                }
                router.PublicIP = assignedAddress;
                topology.Add(router, esxi);
            }
            /*
             * Match the remote router of the tunnels with their assigned addresses
             */
            foreach (Router router in topology)
            {
                foreach (Tunnel tunnel in router.Tunnels)
                {
                    tunnel.RemoteRouter = topology[tunnel.RemoteId];
                }
            }
            /*
             * Construct the static routes for each router
             */
            foreach (Router router in topology)
            {
                IStaticRouteProvider routeProvider = new BreadthFirstRouteProvider();
                router.StaticRoutes = routeProvider.GetRoutesFor(router);
            }
            return topology;
        }
        /// <summary>
        /// Parses the JSON object passed by the front-end of the application,
        /// deserializes it into a topology of routers and tunnel interfaces
        /// and deploys the topology on the assigned ESXi hosts.
        /// </summary>
        /// <param name="topologyJson"></param>
        /// <returns></returns>
        public static async Task<int> Create(JObject topologyJson)
        {
            Topology topology = await Deserialize(topologyJson);
```

```
    if (topology == null)
        return -1;
    if (!topologies.TryAdd(topology.ID, topology))
    {
        return -1;
    }
    Task.Factory.StartNew(() =>
    {
        try
        {
            Create(topology).Wait();
        }
        catch (Exception ex)
        {
            topology.Logger.Error(ex.ToString());
        }
    });
    return topology.ID;
}
static async Task Create(Topology topology)
{
    topology.Logger.Log("Creating topology: " + topology.ID);
    /*
     * Set up VMs and configure their assigned addresses
     */
    foreach (Router router in topology)
    {
        EsxiClient esxi = topology.GetEsxiClientForRouter(router);
        esxi.Logger = topology.Logger;
        if (topology.Deleted) return;
        try
        {
            await esxi.Connect();
            VyosClient vyos = await InitializeRouter(esxi, router, topology);
            topology.Logger.Log("Initialized VyOS router: {0}", router.PublicIP);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
            topology.Logger.Error(ex.ToString());
        }
        finally
        {
            await esxi.Disconnect();
        }
    }
    /*
     * Generate the Ansible inventory
     */
    AnsibleInventory inventory = new AnsibleInventory(topology.ID);
    topology.Logger.Log("Generating Ansible inventory");
    foreach (Router router in topology)
    {
        inventory.AddHost("vyos", router);
    }
    string ansiblePlaybookPath = await inventory.Export();
    topology.Logger.Log("Deployed playbook directory: {0}", ansiblePlaybookPath);
    topology.Logger.Log("Running Ansible playbook on the generated inventory");
    Playbook playbook = inventory.GetPlaybookCommand();
    await playbook.ExecuteAsync(topology.Logger);
    topology.Logger.Log("Clearing up Ansible inventory");
    inventory.Delete();
    topology.Logger.Log("Topology {0} successfully initialized", topology.ID);
}
/// <summary>
```

```
        /// Fully initializes a router VM while locking its zero-day configuration
address.
        /// Releases said lock after configuring the router to use its publicly assigned
        /// IP address.
        /// </summary>
        /// <param name="esxi"></param>
        /// <param name="router"></param>
        /// <param name="logger"></param>
        /// <returns></returns>
        static async Task<VyosClient> InitializeRouter(EsxiClient esxi, Router router,
Topology topology)
        {
            AddressPool addressPool = esxi.GetAddressPool();
            /*
             * Lock the ZeroDay address to avoid multiple initialization processes
             * using it simultaneously
             */
            await addressPool.LockZeroDay();
            try
            {
                /*
                 * Register and power-on the VM
                 */
                topology.Logger.Log("Registering VM: " + router.Hostname);
                topology.Logger.Log("Assigned IP: " + router.PublicIP.ToString());
                esxi.Logger = logger;
                int vmId = await esxi.RegisterVm(router.Hostname);
                router.VmID = vmId;
                VyosClient vyosClient = new VyosClient(addressPool.ZeroDay.ToString(),
"vyos", sshKey);
                vyosClient.Logger = topology.Logger;
                /*
                 * Wait for the VM to come online
                 */
                topology.Logger.Log("Waiting for the VM to boot...");
                while (!await vyosClient.TryConnect())
                {
                    if (topology.Deleted) return null;
                    await Task.Delay(TimeSpan.FromSeconds(SSH_ATTEMPT_INTERVAL));
                }
                /*
                 * Configure the VM's new address
                 */
                topology.Logger.Log("Configuring address {0} on the new VM",
router.PublicIP);
                await vyosClient.ConfigureAddress(router.PublicIP, addressPool.Suffix,
addressPool.Gateway);
                /*
                 * Release the address on the pool
                 * thus subjecting it to ping checks like the others
                 */
                esxi.GetAddressPool().ReleaseAddress(router.PublicIP);
                router.Online = true;
                topology.Logger.Log("Done. Disconnecting...");
                Task.Factory.StartNew(() => vyosClient.Disconnect().Wait(), new
CancellationTokenSource(500).Token);
                return new VyosClient(router.PublicIP.ToString(), "vyos", sshKey);
            }
            finally
            {
                addressPool.ReleaseZeroDay();
            }
        }
        /// <summary>
        /// Get a topology with the given id if it exists in the controller.
        /// </summary>
```

```
        /// <param name="topologyId"></param>
        /// <returns></returns>
        public static Topology GetTopology(int topologyId)
        {
            Topology topology;
            topologies.TryGetValue(topologyId, out topology);
            return topology;
        }
        public static bool Delete(int topologyId)
        {
            Topology topology;
            if (!topologies.TryGetValue(topologyId, out topology))
            {
                return false;
            }
            Task.Factory.StartNew(async () =>
            {
                try
                {
                    await Delete(topology);
                }
                catch (Exception ex)
                {
                    topology.Logger.Error(ex.ToString());
                }
            });
            return true;
        }
        static async Task<bool> Delete(Topology topology)
        {
            topology.Logger.Log("Deleting topology {0}", topology.ID);
            topology.Deleted = true;
            /*
             * Group routers with their ESXi host to power them all off at once
             * without connecting over SSH multiple times.
             */
            Dictionary<EsxiClient, List<Router>> esxiRouterGroups = new
Dictionary<EsxiClient, List<Router>>();
            foreach (Router router in topology)
            {
                EsxiClient esxi = topology.GetEsxiClientForRouter(router);
                esxi.Logger = topology.Logger;
                if (!esxiRouterGroups.ContainsKey(esxi))
                    esxiRouterGroups.Add(esxi, new List<Router>());
                esxiRouterGroups[esxi].Add(router);
            }
            /*
             * Power-off all of the routers on each ESXi host.
             */
            foreach (KeyValuePair<EsxiClient, List<Router>> group in esxiRouterGroups)
            {
                EsxiClient client = group.Key;
                if (!client.IsConnected)
                    await client.Connect();
                foreach (Router router in group.Value)
                {
                    topology.Logger.Log("Deleting router {0} on ESXi host {1}",
router.Hostname, client.Address);
                    await client.RemoveVm(router.VmID);
                    router.Online = false;
                }
                await client.Disconnect();
            }
            return true;
        }
        /// <summary>
```

```
        /// Will go through the resource allocation among available ESXi hosts
        /// to pick one to deploy the given router.
        /// </summary>
        static async Task<EsxiClient> PickHost(Router router)
        {
            return await Program.Options.GetEsxiClient("esxi1");
        }
    }
}
// Service/EsxiClient.cs
using Autonet.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
namespace Autonet.Service
{
    /// <summary>
    /// This class is meant to hold an active SSH connection to an ESXi host.
    /// </summary>
    public class EsxiClient : VMClient
    {
        EsxiHost hostInfo;
        public string Address => hostInfo.Address;
        public EsxiClient(EsxiHost hostInfo, string privateKey) : base(hostInfo.Address,
hostInfo.Username, privateKey)
        {
            this.hostInfo = hostInfo;
        }
        /// <summary>
        /// Gets a list of the currently registered virtual machines on this ESXi host.
        /// </summary>
        /// <returns></returns>
        public async Task<List<VirtualMachine>> ListVirtualMachines()
        {
            List<VirtualMachine> list = new List<VirtualMachine>();
            string output = await RunCommand("vim-cmd vmsvc/getallvms");
            output
                .Split("\n")
                .Skip(1)
                .ToList()
                .ForEach(line =>
                {
                    string[] p = line
                        .Split(null)
                        .Where(x => !string.IsNullOrWhiteSpace(x))
                        .ToArray();
                    if (p.Length == 0)
                        return;
                    list.Add(new VirtualMachine()
                    {
                        ID = int.Parse(p[0]),
                        Name = p[1],
                        File = p[3],
                        GuestOS = p[4],
                        Version = p[5]
                    });
                });
            return list;
        }
        /// <summary>
        /// Get the info for a specific virtual machine given its id.
        /// </summary>
        /// <param name="vmId"></param>
        /// <returns></returns>
```

```
        public Task<VirtualMachine> GetVirtualMachine(int vmId)
        {
            return ListVirtualMachines().ContinueWith(vms => vms.Result.FirstOrDefault(vm
=> vm.ID == vmId));
        }
        /// <summary>
        /// Register and power on a new virtual machine on this ESXi host.
        /// </summary>
        /// <param name="name">The name to assign the newly created VM on the
ESXi.</param>
        /// <returns></returns>
        public async Task<int> RegisterVm(string name)
        {
            string imageDir = hostInfo.GetImageDir("vyos");
            string vmxDir = hostInfo.GetPath(string.Format("vms/{0}-{1}", name,
DateTimeOffset.UtcNow.ToUnixTimeSeconds()));
            string vmx = Path.Combine(vmxDir, "vyos.vmx").Replace('\\', '/');
            /*
             * Create the destination folder
             */
            await RunCommand("mkdir -p \"{0}\"", vmxDir);
            /*
             * Copy the VMX file
             */
            await RunCommand("cp '{0}/vyos.vmx' '{1}'", imageDir, vmx);
            /*
             * Clone the virtual disk
             */
            await RunCommand("vmkfstools -d thin -i '{0}/vyos.vmdk' '{1}/vyos.vmdk'",
imageDir, vmxDir);
            /*
             * Register a new VM at the new dir
             */
            string register = await RunCommand("vim-cmd solo/registervm '{0}' '{1}'",
vmx, name);
            /*
             * Power-On the new VM
             */
            int vmId = int.Parse(register.Trim());
            await RunCommand("vim-cmd vmsvc/power.on {0}", vmId);
            return vmId;
        }
        public async Task RemoveVm(int vmId)
        {
            /*
             * Retrieve the VMs info to locate its directory in the datastore
             */
            VirtualMachine vmInfo = await GetVirtualMachine(vmId);
            string vmDir = Path.GetDirectoryName(hostInfo.GetPath(vmInfo.File));
            /*
             * Power-off the VM
             */
            await RunCommand("vim-cmd vmsvc/power.off {0}", vmId);
            /*
             * Unregister it from the ESXi
             */
            await RunCommand("vim-cmd vmsvc/unregister {0}", vmId);
            /*
             * Delete its directory on the datastore
             */
            await RunCommand("rm -rf {0}", vmDir);
        }
        /// <summary>
        /// Shortcut to retrieving the address pool assigned to this host in the options.
        /// </summary>
        /// <returns></returns>
```

```
        public AddressPool GetAddressPool()
        {
            return Program.Options.GetAddressPool(hostInfo.AddressPoolId);
        }
    }
}
// Service/Routing/IStaticRouteProvider.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Autonet.Models;
namespace Autonet.Service
{
    interface IStaticRouteProvider
    {
        /// <summary>
        /// Given a router and exploring its neighbors through its tunnels, this method
        /// will construct a static route for every other reachable router in the graph.
        /// </summary>
        /// <param name="router"></param>
        /// <returns></returns>
        List<StaticRoute> GetRoutesFor(Router router);
    }
}
// Service/Routing/BreadthFirstRouteProvider.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Autonet.Models;
namespace Autonet.Service
{
    /// <summary>
    /// This class generates a routing table using Breadth-First graph traversal.
    /// </summary>
    public class BreadthFirstRouteProvider : IStaticRouteProvider
    {
        public List<StaticRoute> GetRoutesFor(Router router)
        {
            List<StaticRoute> routes = new List<StaticRoute>();
            HashSet<string> visited = router.GetNeighbors()
                                            .Select(r => r.PublicIP.ToString())
                                            .ToHashSet();
            foreach (Tunnel tunnel in router.Tunnels)
            {
                Router neighbor = tunnel.RemoteRouter;
                Queue<Router> queue = new Queue<Router>();
                queue.Enqueue(neighbor);
                while (queue.Count > 0)
                {
                    Router r = queue.Dequeue();
                    foreach (Tunnel intf in r.Tunnels)
                    {
                        StaticRoute route = new StaticRoute(intf.Addr, tunnel.Route);
                        if (!router.HasInterfaceAddress(route.Dest))
                        {
                            routes.Add(route);
                        }
                    }
                    visited.Add(r.PublicIP.ToString());
                    queue.Enqueue(from n in r.GetNeighbors()
                                  where !visited.Contains(n.PublicIP.ToString())
                                  select n);
                }
            }
```

```
            return routes;
        }
    }
}
// Service/Logging/OutputLogger.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
namespace Autonet.Service
{
    /// <summary>
    /// Is a serializable thread-safe class meant to hold log messages that are logged
through it.
    /// </summary>
    [JsonObject(MemberSerialization.OptIn)]
    public class OutputLogger : Logger
    {
        [JsonProperty]
        List<LogMessage> log = new List<LogMessage>();
        object _lock = new object();
        void Add(LogMessage msg)
        {
            if (string.IsNullOrWhiteSpace(msg.Text))
                return;
#if DEBUG
            Console.WriteLine(msg.Text);
#endif
            lock (_lock)
            {
                log.Add(msg);
            }
        }
        public override void Log(string line)
        {
            Add(new LogMessage(0xFFFFFF, line));
        }
        public override void Error(string line)
        {
            Add(new LogMessage(0xFF0000, line));
        }
        public override void Command(string line)
        {
            Add(new LogMessage(0x00FF00, line));
        }
        public IEnumerable<LogMessage> GetLogs()
        {
            lock (_lock)
            {
                return log.ToList();
            }
        }
    }
}
// Service/Logging/ConsoleLogger.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace Autonet.Service
{
    /// <summary>
    /// Represents a simple logger that writes everything on the console.
    /// </summary>
    public class ConsoleLogger : Logger
```

```
    {
        public override void Log(string line)
        {
            Console.WriteLine(line);
        }
    }
}
// Service/Logging/LogMessage.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
namespace Autonet.Service
{
    /// <summary>
    /// Represents a serializable line of output with a specific color
    /// for better visualization.
    /// </summary>
    [JsonObject(MemberSerialization.OptIn)]
    public struct LogMessage
    {
        public readonly int Color;
        [JsonProperty("text")]
        public readonly string Text;
        [JsonProperty("color")]
        public string ColorHex
        {
            get { return "#" + Color.ToString("X6"); }
        }
        public LogMessage(int color, string text)
        {
            Color = color;
            Text = text;
        }
    }
}
// Service/Logging/Logger.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace Autonet.Service
{
    /// <summary>
    /// The abstract implementation of a loger used throughout the application.
    /// </summary>
    public abstract class Logger
    {
        public abstract void Log(string line);
        public void Log(string format, params object[] args) => Log(string.Format(format,
args));
        public virtual void Error(string line) => Log(line);
        public void Error(string format, params object[] args) =>
Error(string.Format(format, args));
        public virtual void Command(string line) => Log(line);
        public void Command(string format, params object[] args) =>
Command(string.Format(format, args));
    }
}
// Options/Options.cs
using Autonet.Service;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
```

```
using System.Threading.Tasks;
using YamlDotNet.Serialization;
namespace Autonet
{
    /// <summary>
    /// Class used to hold the options of the application.
    /// </summary>
    public class Options
    {
        /*
         * Configuration properties
         */
        string sshKeyPath;
        [YamlMember(Alias = "ssh_key", ApplyNamingConventions = false)]
        public string SshKeyPath {
            get
            {
                return sshKeyPath.Replace("~",
Environment.GetFolderPath(Environment.SpecialFolder.UserProfile));
            }
            set { sshKeyPath = value; }
        }
        public List<EsxiHost> Hosts { get; set; }
        [YamlMember(Alias = "address_pools", ApplyNamingConventions = false)]
        public AddressPool[] AddressPools { get; set; }
        [YamlMember(Alias = "router_types", ApplyNamingConventions = false)]
        public Dictionary<string, string> RouterTypes { get; set; }
        [YamlMember(Alias = "ping_timeout_ms", ApplyNamingConventions = false)]
        public int PingTimeoutMs { get; set; }
        /*
         * Global configuration constants
         */
        public static string AnsibleBaseConfigurationDir =>
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Ansible");
        public static string AnsibleTargetConfigurationDir =>
Path.Combine(Path.GetTempPath(), "autonet");
        /*
         * Shortcuts
         */
        public async Task<string> LoadSshKey()
        {
            StreamReader reader = new StreamReader(SshKeyPath);
            return await reader.ReadToEndAsync();
        }
        public async Task<EsxiClient> GetEsxiClient(string name)
        {
            EsxiHost host = Hosts.First(h => h.Name == name);
            return new EsxiClient(host, await LoadSshKey());
        }
        public AddressPool GetAddressPool(int addressPoolId)
        {
            return AddressPools.FirstOrDefault(pool => pool.PoolId == addressPoolId);
        }
        public string GetRouterTypeAnsibleGroup(string routerType)
        {
            return RouterTypes.ContainsKey(routerType) ? RouterTypes[routerType] : "all";
        }
    }
}
// Options/IPAddressTypeConverters.cs
using Newtonsoft.Json;
using System;
using System.Net;
using YamlDotNet.Core;
using YamlDotNet.Core.Events;
using YamlDotNet.Serialization;
```

```
namespace Autonet
{
    /// <summary>
    /// Helper class used to convert the IPAddress type when serializing YAML.
    /// </summary>
    public class IPAddressTypeConverter : IYamlTypeConverter
    {
        public bool Accepts(Type type)
        {
            return type == typeof(IPAddress);
        }
        public object ReadYaml(IParser parser, Type type)
        {
            Scalar scalar = (Scalar)parser.Current;
            return IPAddress.Parse(scalar.Value);
        }
        public void WriteYaml(IEmitter emitter, object value, Type type)
        {
            IPAddress ip = (IPAddress)value;
            emitter.Emit(new Scalar(ip.ToString()));
        }
    }
    /// <summary>
    /// Helper class used to convert the IPAddress type when serializing JSON.
    /// </summary>
    public class IPAddressConverter : JsonConverter
    {
        public override bool CanConvert(Type objectType)
        {
            return (objectType == typeof(IPAddress));
        }
        public override void WriteJson(JsonWriter writer, object value, JsonSerializer
serializer)
        {
            writer.WriteValue(value.ToString());
        }
        public override object ReadJson(JsonReader reader, Type objectType, object
existingValue, JsonSerializer serializer)
        {
            return IPAddress.Parse((string)reader.Value);
        }
    }
}
// Options/EsxiHost.cs
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using YamlDotNet.Serialization;
namespace Autonet
{
    /// <summary>
    /// This class holds the connection configuration for an ESXi host to be deserialized
    /// from the YAML configuration file.
    /// </summary>
    public class EsxiHost
    {
        public string Name { get; set; }
        public string Address { get; set; }
        public string Username { get; set; }
        [YamlMember(Alias = "data_dir", ApplyNamingConventions = false)]
        public string DataDir { get; set; }
        public Dictionary<string, string> Images { get; set; }
        [YamlMember(Alias = "address_pool", ApplyNamingConventions = false)]
        public int AddressPoolId { get; set; }
```

```
        public string GetImageDir(string vm)
        {
            return GetPath(Images[vm]);
        }
        public string GetPath(string path)
        {
            return Path.Combine(DataDir, path).Replace('\\', '/');
        }
    }
}
// Options/AddressPool.cs
using System;
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Net;
using YamlDotNet.Serialization;
using System.Net.NetworkInformation;
namespace Autonet
{
    /// <summary>
    /// This class represents as address pool to be deserialized
    /// from the YAML configuration file.
    /// </summary>
    public class AddressPool
    {
        SemaphoreSlim assignLock = new SemaphoreSlim(1);
        SemaphoreSlim zerodayLock = new SemaphoreSlim(1);
        readonly ConcurrentDictionary<string, object> dirtyAddresses = new
ConcurrentDictionary<string, object>();
        [YamlMember(Alias = "pool_id", ApplyNamingConventions = false)]
        public int PoolId { get; set; }
        [YamlMember(Alias = "zero_day", ApplyNamingConventions = false)]
        public IPAddress ZeroDay { get; private set; }
        public IPAddress From { get; private set; }
        public IPAddress To { get; private set; }
        public IPAddress Gateway { get; private set; }
        public int Suffix { get; private set; }
        public bool InPool(IPAddress ipAddress)
        {
            return ipAddress.ToUint() >= From.ToUint()
                && ipAddress.ToUint() <= To.ToUint();
        }
        public async Task<IPAddress> TryAssignAddress()
        {
            /*
             * Hold a lock while checking pool adresses
             * to avoid conflicts
             */
            await assignLock.WaitAsync();
            try
            {
                IPAddress current = From.Clone();
                while (InPool(current))
                {
                    /*
                     * Pick an address from the pool
                     * and advance the pool forward under the lock
                     */
                    if (!dirtyAddresses.ContainsKey(current.ToString())
                        && !await AddressInUse(current))
                    {
                        // If the address is not in use return it.
                        dirtyAddresses.TryAdd(current.ToString(), null);
```

```
                        return current.Clone();
                    }
                    current = current.Next();
                }
            }
            finally
            {
                assignLock.Release();
            }
            /*
            * This pool has reached the end.
            * Return null
            */
            return null;
        }
        public Task LockZeroDay()
        {
            return zerodayLock.WaitAsync();
        }
        public void ReleaseZeroDay()
        {
            zerodayLock.Release();
        }
        public void ReleaseAddress(IPAddress addr)
        {
            object tmp;
            dirtyAddresses.TryRemove(addr.ToString(), out tmp);
        }
        /// <summary>
        /// Performs a ping with a timeout to check if an address is in use
        /// by another device.
        /// </summary>
        /// <param name="addr"></param>
        /// <returns></returns>
        static async Task<bool> AddressInUse(IPAddress addr)
        {
            Ping ping = new Ping();
            try
            {
                PingReply reply = await ping.SendPingAsync(addr,
Program.Options.PingTimeoutMs);
                return reply.Status == IPStatus.Success;
            }
            catch (PingException)
            {
                return false;
            }
            finally
            {
                if (ping != null)
                {
                    ping.Dispose();
                }
            }
        }
    }
}
// Models/HttpError.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace Autonet.Models
{
    public class HttpError : JsonResult
```

```
    {
        public HttpError(int statusCode, string message)
            : base(new
            {
                status = statusCode,
                message
            })
        {
        }
    }
}
// Models/VirtualMachine.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
namespace Autonet.Models
{
    /// <summary>
    /// This class represents a virtual machine on an ESXi host.
    /// </summary>
    public class VirtualMachine
    {
        [JsonProperty("vmid")]
        public int ID;
        [JsonProperty("name")]
        public string Name;
        [JsonProperty("file")]
        public string File;
        [JsonProperty("guest_os")]
        public string GuestOS;
        [JsonProperty("version")]
        public string Version;
    }
}
// Models/ErrorViewModel.cs
using System;
namespace Autonet.Models
{
    public class ErrorViewModel
    {
        public string RequestId { get; set; }
        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
// Models/Topology/Tunnel.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
using YamlDotNet.Serialization;
using YamlDotNet.Serialization.NamingConventions;
namespace Autonet.Models
{
    /// <summary>
    /// This class represents a tunnel interface.
    /// Deserializable from the JSON-encoded topology sent by the client.
    /// Serializable into YAML to generate an Ansible inventory.
    /// </summary>
    [JsonObject(MemberSerialization.OptIn)]
    public class Tunnel
    {
        /// <summary>
        /// The interface name to assign to this tunnel when configuring the router.
```

```
        /// </summary>
        [YamlMember(Alias = "interface")]
        [JsonProperty("interface")]
        public string Interface { get; set; }
        /// <summary>
        /// The address to assign to the tunnel interface on the router.
        /// </summary>
        [YamlMember(Alias = "addr")]
        [JsonProperty("addr")]
        public string Addr { get; set; }
        /// <summary>
        /// The address of the remote VM on this tunnel, for which this VM should have a
static route for.
        /// </summary>
        [YamlMember(Alias = "route")]
        [JsonProperty("route")]
        public string Route { get; set; }
        /// <summary>
        /// The unique ID of the remote router as it was generated by the designer.
        /// </summary>
        [YamlIgnore]
        [JsonProperty("remote_id")]
        public int RemoteId { get; set; }
        /// <summary>
        /// Returns the public IP of the remote VM, to which the tunnel will be set
towards.
        /// </summary>
        [YamlMember(Alias = "remote")]
        public string Remote { get { return RemoteRouter.PublicIP.ToString(); } }
        /// <summary>
        /// The remote VM, to which the tunnel will be set towards.
        /// </summary>
        [YamlIgnore]
        public Router RemoteRouter { get; set; }
    }
}
// Models/Topology/Router.cs
using System;
using System.Net;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
using YamlDotNet.Serialization;
using YamlDotNet.Serialization.NamingConventions;
namespace Autonet.Models
{
    /// <summary>
    /// This class represents a router along with the descriptions of its tunnel
interfaces.
    /// </summary>
    public class Router
    {
        [YamlMember(Alias = "hostname", ApplyNamingConventions = false)]
        [JsonProperty("hostname")]
        public string Hostname { get; set; }
        [JsonProperty("ip")]
        public IPAddress PublicIP;
        public int RouterId;
        [JsonProperty("online")]
        public volatile bool Online;
        [JsonProperty("vm_id")]
        public int VmID;
        List<Tunnel> tunnels;
        /// <summary>
        /// Wrapper property for editting the router's tunnels.
```

```
        /// The setter will sequentially name the interfaces tunX.
        /// </summary>
        [YamlMember(Alias = "tunnels", ApplyNamingConventions = false)]
        [JsonProperty("tunnels")]
        public IEnumerable<Tunnel> Tunnels
        {
            get { return tunnels; }
            set
            {
                tunnels = new List<Tunnel>();
                int interfaceNum = 0;
                foreach (Tunnel tunnel in value)
                {
                    tunnel.Interface = "tun" + (interfaceNum++);
                    tunnels.Add(tunnel);
                }
            }
        }
        [YamlMember(Alias = "static_routes", ApplyNamingConventions = false)]
        public IEnumerable<StaticRoute> StaticRoutes { get; set; }
        /// <summary>
        /// Returns true if the router has the given address on
        /// any of its tunnel interfaces.
        /// </summary>
        /// <param name="ipAddr"></param>
        /// <returns></returns>
        public bool HasInterfaceAddress(string ipAddr)
        {
            return tunnels != null
                && tunnels.Exists(t => t.Addr == ipAddr);
        }
        /// <summary>
        /// Returns an array of this router's neighbors.
        /// </summary>
        /// <returns></returns>
        public Router[] GetNeighbors()
        {
            return Tunnels.Select(t => t.RemoteRouter).ToArray();
        }
    }
}
// Models/Topology/StaticRoute.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using YamlDotNet.Serialization;
using YamlDotNet.Serialization.NamingConventions;
namespace Autonet.Models
{
    /// <summary>
    /// This class represents a static route in a routing table.
    /// </summary>
    public class StaticRoute
    {
        [YamlMember(Alias = "dest")]
        public string Dest { get; private set; }
        [YamlMember(Alias = "next_hop", ApplyNamingConventions = false)]
        public string NextHop { get; private set; }
        public StaticRoute(string dest, string nextHop)
        {
            Dest = dest;
            NextHop = nextHop;
        }
    }
}
```

```
// Controllers/HomeController.cs
using Microsoft.AspNetCore.Mvc;
namespace Autonet.Controllers
{
    /// <summary>
    /// This class is the basic controller for the application to
    /// serve the HTML pages.
    /// </summary>
    public class HomeController : Controller
    {
        /// <summary>
        /// Used to serve the home page.
        /// </summary>
        /// <returns></returns>
        [HttpGet("")]
        public IActionResult Index()
        {
            return View();
        }
        /// <summary>
        /// Used to serve the monitoring page for a specific topology.
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGet("topology/{id}")]
        public IActionResult Topology(int id)
        {
            ViewBag.TopologyId = id;
            return View();
        }
    }
}
// Controllers/ApiController.cs
using System;
using System.Net;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using Autonet.Models;
using Autonet.Service;
namespace Autonet.Controllers
{
    /// <summary>
    /// This class serves as the controller for the application's API.
    /// </summary>
    [Route("api")]
    public class ApiController : Controller
    {
        /// <summary>
        /// HTTP endpoint for creating a new topology.
        /// </summary>
        /// <param name="topologyJson"></param>
        /// <returns></returns>
        [HttpPost("create")]
        public IActionResult Create([FromBody]JObject topologyJson)
        {
            int topologyId = TopologyController.Create(topologyJson).Result;
            if (topologyId == -1)
            {
                int status = (int)HttpStatusCode.NotAcceptable;
                HttpContext.Response.StatusCode = status;
                return new HttpError(status, "Not enough addresses in the pool to assign
to this topology");
```

```
            }
            return Ok(topologyId);
        }
        /// <summary>
        /// HTTP endpoint for deleting a topology.
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpPost("delete/{id}")]
        public IActionResult Delete(int id)
        {
            bool found = TopologyController.Delete(id);
            return found ? (IActionResult)Ok() : NotFound();
        }
        /// <summary>
        /// HTTP endpoint for retrieving information regarding the topology with a given
id.
        /// This information currently includes details on the routers along with their
status.
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGet("topology/{id}")]
        public IEnumerable<Router> Topology(int id)
        {
            Topology topology = TopologyController.GetTopology(id);
            if (topology == null)
            {
                HttpContext.Response.StatusCode = (int)HttpStatusCode.NotFound;
                return null;
            }
            return topology.GetRouters();
        }
        /// <summary>
        /// HTTP endpoint used to retrieve the current command output and other
diagnostic messages
        /// for the initialization process of a topology with a given id.
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGet("logs/{id}")]
        public IEnumerable<LogMessage> Output(int id)
        {
            Topology topology = TopologyController.GetTopology(id);
            if (topology == null
                || !(topology.Logger is OutputLogger))
            {
                HttpContext.Response.StatusCode = (int)HttpStatusCode.NotFound;
                return null;
            }
            OutputLogger logger = topology.Logger as OutputLogger;
            return logger.GetLogs();
        }
    }
}
// options.yml
ssh_key: "~/.ssh/id_rsa"
ping_timeout_ms: 400
hosts:
- name: 'esxi1'
  address: '147.102.39.18'
  username: 'root'
  data_dir: '/vmfs/volumes/datastore1'
  address_pool: 0
  images:
    vyos: 'vyos'
```

```
address_pools:
  - pool_id: 0
    zero_day: 147.102.39.41
    from: 147.102.39.30
    to: 147.102.39.40
    gateway: 147.102.39.1
    suffix: 26
router_types:
  vyos: routers_vyos
// Ansible/playbook.yml
- name: "Initialization of VyOS routers"
  hosts: routers_vyos
  gather_facts: no
  tasks:
  - name: "setting router hostnames"
    vyos_system:
      host_name: "{{ hostname }}"
  - name: "clearing any existing tunnels"
    loop: "{{ tunnels }}"
    ignore_errors: yes
    vyos_config:
      lines:
        - delete interfaces tunnel "{{ item.interface }}"
  - name: "configuring interface tunnels"
    loop: "{{ tunnels }}"
    vyos_config:
      save: yes
      lines:
        - set interfaces tunnel "{{ item.interface }}" encapsulation gre
        - set interfaces tunnel "{{ item.interface }}" local-ip "{{ inventory_hostname
}}"
        - set interfaces tunnel "{{ item.interface }}" remote-ip "{{ item.remote }}"
        - set interfaces tunnel "{{ item.interface }}" address "{{ item.addr }}/32"
        - set protocols static interface-route "{{ item.route }}/32" next-hop-interface
"{{ item.interface }}"
  - name: "configuring static routes"
    loop: "{{ static_routes }}"
    vyos_static_route:
      prefix: "{{ item.dest }}"
      mask: 32
      next_hop: "{{ item.next_hop }}"
// Ansible/inventory/group_vars/routers_vyos.yml
# Specify the user to login as
ansible_user: vyos
# Necessary to support the shell that VyOS runs
ansible_connection: network_cli
ansible_network_os: vyos
// Ansible/ansible.cfg
[defaults]
inventory = inventory
private_key_file = ~/.ssh/id_rsa
host_key_checking = False
```