

Σύνοψη

Οι σύγχρονες τάσεις στο πεδίο των δικτύων υπολογιστών επιβάλλουν τη χρήση εικονικών μηχανών για την προσομοίωση πραγματικών δικτυακών τοπολογιών. Η εξέλιξη αυτή προσφέρει τη δυνατότητα εξοικονόμησης πόρων, καθώς και την ευελιξία της διαχείρισης των εξομοιούμενων μηχανών μέσω γραφικών περιβαλλόντων αλλά και προγραμματιστικών εργαλείων. Το υλικό που απαρτίζει τα σύγχρονα υπολογιστικά συστήματα μπορεί να υποστηρίξει εικονικές μηχανές με επιδόσεις συγκρίσιμες με αυτές που έχουν οι διακριτές δικτυακές συσκευές, όπως δρομολογητές (routers) και μεταγωγείς (switches). Στο πλαίσιο αυτό, η παρούσα διπλωματική εργασία αφορά στην ανάπτυξη μιας εφαρμογής για τη διαχείριση εικονικοποιημένων (virtualized) τοπολογιών δικτύων, οι οποίες αποτελούνται από δρομολογητές (routers), αλλά και εικονικές μηχανές που εξομοιώνουν τη λειτουργία προσωπικών υπολογιστών (PCs). Η εφαρμογή υλοποιείται αποκλειστικά με χρήση λογισμικού ανοιχτού κώδικα και είναι σχεδιασμένη για χρήση σε διανομές Linux. Συγκεκριμένα, η επικοινωνία με τον hypervisor (KVM) γίνεται μέσω του εργαλείου libvirt (API), που προσφέρει τη διασύνδεση με πολλές γλώσσες προγραμματισμού. Στόχος είναι ένα εύχρηστο γραφικό περιβάλλον που φιλοξενείται σε έναν φυλλομετρητή (browser) και ανταποκρίνεται στις ανάγκες εξομοιώσεων που πραγματοποιούνται στα πλαίσια εργαστηρίων δικτύων, τόσο σε εκπαιδευτικό όσο και σε ερευνητικό επίπεδο.

Abstract

Modern trends in the field of computer networks require the use of virtual machines to simulate real network topologies. This development enables the saving of resources, as well as the flexibility of managing the simulated machines through graphical environments and programming tools. The hardware used in modern computer systems can support virtual machines with performance comparable to discrete network devices, such as routers and switches. In this context, this diploma thesis concerns the development of an application for the management of virtualized network topologies, which consist of routers and virtual machines that simulate the operation of personal computers (PCs). The application is implemented exclusively using open source software and is designed for use in Linux distributions. Specifically, communication with the hypervisor (KVM) is done through libvirt (API), which offers an interface for many programming languages. The goal is an easy-to-use graphical interface hosted in a browser that responds to the needs of simulations performed within network laboratories at both educational and research levels.

Περιεχόμενα

Κεφάλαιο 1 Εισαγωγή	13
1.1 Πρόλογος.....	13
1.2 Αντικείμενο της Διπλωματικής Εργασίας.....	14
1.3 Δομή της Διπλωματικής Εργασίας	14
Κεφάλαιο 2 Σχετικές Τεχνολογίες και Εργαλεία	17
2.1 Εικονικοποίηση	17
2.1.1 Εικονικές Μηχανές	17
2.1.2 Hypervisors.....	18
2.1.3 KVM	18
2.2 Τεχνολογία λογισμικού	18
2.2.1 Συστήματα ελέγχου εκδόσεων λογισμικού	18
2.2.2 Αρχιτεκτονική Πελάτη – Διακομιστή	19
2.2.3 Τεχνολογίες Frontend	19
2.2.3.1 JavaScript	20
2.2.3.2 JQuery	20
2.2.4 Τεχνολογίες backend.....	20
2.2.4.1 Python	20
Flask	20
Werkzeug	21
2.2.4.2 libvirt	21
2.2.5 Άλλες τεχνολογίες	23
2.2.5.1 Jinja	23
2.3 Λειτουργικά Συστήματα και Πυρήνες	23
2.3.1 Linux	23
2.3.2 FreeBSD	24
2.3.2.1 BSDRP	24
Κεφάλαιο 3 Εικονικοποιημένα Δίκτυα και Δικτυακές Διεπαφές	27
3.1 Εικονικοποιημένα Δίκτυα.....	27
3.1.1 Γενικά Στοιχεία	27
3.1.2 Επιλογές Διασύνδεσης	27
3.1.2.1 Σύνδεση σε Γέφυρα	27
3.1.2.2 Προώθηση.....	28
3.1.3 Ποιότητα Υπηρεσίας	29
3.1.4 Portgroups.....	29
3.1.5 Στατικές Διαδρομές	29
3.1.6 VLANs	31
3.2 Εικονικοποιημένες και Εξομοιούμενες Δικτυακές Διεπαφές.....	32
3.3 Παραδείγματα Χρήσης Δικτύων	33
Κεφάλαιο 4 Παραγωγή & Παραμετροποίηση της Βασικής Εικόνας του BSDRP.....	39
4.1 Δημιουργία της εικόνας BSDRP από τον πηγαίο κώδικα	39
4.2 Παραμετροποίηση της εικόνας BSDRP	41
4.2.1 Τροποποίηση αρχείων του λειτουργικού συστήματος.....	42
4.2.2 Εγκατάσταση του λογισμικού	45

4.2.2.1	Εγκατάσταση των εξαρτήσεων λογισμικού pkg	45
4.2.2.2	Εγκατάσταση και προσαρμογή του λογισμικού rpxterm.js	45
Κεφάλαιο 5 Εγκατάσταση και Χρήση της Εφαρμογής		49
5.1	Απαιτήσεις	49
5.2	Γραφικό Περιβάλλον	50
5.2.1	Μενού Διαχείρισης Εικονικών Μηχανών	50
5.2.2	Μενού Δημιουργίας Εικονικών Μηχανών	50
5.3	Εγκατάσταση	51
5.4	Σενάριο Χρήσης OSPF.....	53
5.5	Σενάριο Χρήσης BGP	62
Κεφάλαιο 6 Συμπεράσματα και Προοπτικές		69
6.1	Συμπεράσματα	69
6.2	Προοπτικές.....	69
Παράρτημα Α Πηγαίος κώδικας της εφαρμογής		71
Παράρτημα Β Προκαθορισμένες διευθύνσεις MAC & IP των εικονικοποιημένων δικτύων		125
Βιβλιογραφία		127

Ευρετήριο Πινάκων

Πίνακας 1 Σύνοψη διευθύνσεων IP των εικονικών μηχανών για το σενάριο OSPF	59
Πίνακας 2 Σύνοψη διευθύνσεων IP των εικονικών μηχανών για το σενάριο BGP	68
Πίνακας 3: Προκαθορισμένες διευθύνσεις MAC & IP	125

Ευρετήριο Εικόνων

Εικόνα 1 Αποτύπωση του libvirt σε ένα τυπικό λειτουργικό σύστημα	22
Εικόνα 2 Η μορφή ενός εικονικού μηχανήματος στο γραφικό περιβάλλον, ανάλογα με την κατάσταση του (αριστερά εν λειτουργία, δεξιά απενεργοποιημένο).	50
Εικόνα 3 Μενού δημιουργίας εικονικών μηχανών	51
Εικόνα 4 Τοπολογία OSPF	53
Εικόνα 5 Επιλογή αριθμού δρομολογητών & προσωπικών υπολογιστών	54
Εικόνα 6 Επιλογή ονόματος & αριθμού διεπαφών για την κάθε εικονική μηχανή	55
Εικόνα 7 Παραμετροποίηση των διεπαφών των εικονικών μηχανών	56
Εικόνα 8 Σελίδα διαχείρισης εικονικών μηχανών	57
Εικόνα 9 Προσαρμογή των καρτών	58
Εικόνα 10 Παραμετροποίηση του δρομολογητή R1	60
Εικόνα 11 Προβολή border routers	61
Εικόνα 12 Γείτονες του δρομολογητή R1	61
Εικόνα 13 Traceroute από το PC1 στο PC2	62
Εικόνα 14 Τοπολογία BGP	62
Εικόνα 15 Παραμετροποίηση των διεπαφών των εικονικών μηχανών	63
Εικόνα 16 Παρουσίαση τοπολογίας στην εφαρμογή	64
Εικόνα 17 Παραμετροποίηση του δρομολογητή R1	65
Εικόνα 18 Παραμετροποίηση του δρομολογητή R2	66
Εικόνα 19 Παραμετροποίηση του δρομολογητή R3	67
Εικόνα 20 Η εντολή traceroute από το PC1 στο PC2	67

Κεφάλαιο 1

Εισαγωγή

1.1 Πρόλογος

Στα παραδοσιακά υπολογιστικά συστήματα, η δημιουργία δικτύων προϋπέθετε τη χρήση διακριτών δικτυακών συσκευών και τερματικών και η επικοινωνία μεταξύ αυτών γινόταν με χρήση φυσικών δικτυακών διεπαφών (καρτών δικτύου), και γινόταν είτε ενσύρματα είτε ασύρματα. Ωστόσο, στη διασύνδεση ενός πλήθους υπολογιστών απαιτούνταν διαφορετικοί τύποι καλωδίων, καρτών δικτύου, λειτουργικών συστημάτων, γεφυρών και πολλών άλλων στοιχείων δικτύωσης καθώς και διαφορετικοί τρόποι συνδεσμολογίας, ώστε να επιτευχθεί η κατάλληλη τοπολογία για κάθε περίπτωση.

Το πρόβλημα της σύνδεσης όλων αυτών των διαφορετικών στοιχείων και της δημιουργίας πολλαπλών συνδυασμών μεταξύ τους έρχεται να επιλύσει η εικονικοποίηση των διακριτών συσκευών και κατά συνέπεια των τοπολογιών δικτύων. Η υιοθέτηση της τεχνολογίας εικονικοποίησης έχει αυξηθεί σημαντικά τα τελευταία χρόνια. Ωστόσο, δεν πρόκειται για μια καινούργια ιδέα αλλά για μια τεχνολογία που υφίσταται εδώ και αρκετό καιρό και παρέχει έναν τρόπο διαχωρισμού του φυσικού υλικού (υπολογιστή) και του λογισμικού (λειτουργικό σύστημα και εφαρμογές), προσομοιώνοντας τη λειτουργία ενός φυσικού μηχανήματος που χρησιμοποιεί λογισμικό. Με αυτό τον τρόπο καθίσταται δυνατό σε ένα υπολογιστικό περιβάλλον να εκτελούνται πολλά ανεξάρτητα συστήματα ταυτόχρονα.

Σήμερα, η εικονικοποίηση χρησιμοποιείται στα κέντρα δεδομένων ώστε να επιτυγχάνεται ένα επίπεδο αφαίρεσης του φυσικού υλικού. Η τεχνολογία αυτή βοηθά στην κατανομή μεγάλων ποσοτήτων πόρων (CPU, κύρια μνήμη, δίσκοι και αποθηκευτικά μέσα) σε χρήστες ή πελάτες με τη μορφή εικονικών μηχανών και άλλων υπηρεσιών. Ακόμη, αξιοποιείται σε εμπορικά δίκτυα, μειώνοντας τον αριθμό των συνολικών φυσικών συσκευών σε έναν οργανισμό, αφού κάθε υπολογιστικό σύστημα μπορεί πλέον να φιλοξενεί πολλές εικονικές μηχανές και άρα πολλές εφαρμογές.

Σε ακαδημαϊκό επίπεδο, η εφαρμογή της εικονικοποίησης στη δημιουργία δικτυακών τοπολογιών έχει ανοίξει νέους δρόμους σε εκπαιδευτικό και ερευνητικό επίπεδο. Πιο συγκεκριμένα, οι εικονικοποιημένες τοπολογίες δικτύου προσφέρουν τη δυνατότητα υλοποίησης ποικίλων εξομοιώσεων που πραγματοποιούνται στα πλαίσια εργαστηρίων δικτύων από φοιτητές, καθηγητές και ερευνητές. Ακόμα, η εικονικοποίηση επιτρέπει τη δημιουργία ενός ασφαλούς δοκιμαστικού περιβάλλοντος στο οποίο μπορούν να λειτουργήσουν εύκολα και γρήγορα οι δικτυακές τοπολογίες δίχως να απαιτούνται οι φυσικές συσκευές και ο χρόνος εγκατάστασης και παραμετροποίησης

αυτών. Πέρα από την ασφάλεια των εικονικών τοπολογιών, μειώνονται σημαντικά και οι δαπάνες κατανάλωσης ενέργειας που θα αποδίδονταν για την λειτουργία τους. Υπό αυτό το πρίσμα, ο πειραματισμός επί των τοπολογιών καθίσταται για όλη την ακαδημαϊκή και ερευνητική κοινότητα άμεσος, εύκολος στη χρήση και δίχως χρηματική επιβάρυνση.

1.2 Αντικείμενο της Διπλωματικής Εργασίας

Η παρούσα διπλωματική αναπτύσσεται γύρω από το σχεδιασμό και την υλοποίηση μιας web εφαρμογής για τη διαχείριση δικτυακών τοπολογιών. Στο πλαίσιο της εφαρμογής, προσφέρεται στο χρήστη η δυνατότητα κατασκευής και παραμετροποίησης μιας εικονικοποιημένης τοπολογίας δικτύου μέσω ενός εύχρηστου και φιλικού προς το χρήστη περιβάλλοντος. Βασικός πυρήνας της υλοποίησης του λογισμικού είναι η αξιοποίηση των εργαλείων της εικονικοποίησης και συγκεκριμένα των δυνατοτήτων αυτοματοποίησης της δημιουργίας εικονικών μηχανών και εικονικοποιημένων δικτύων.

1.3 Δομή της Διπλωματικής Εργασίας

Η διπλωματική εργασία εκτυλίσσεται σε **έξι** κεφάλαια με βάση τη ροή της διαδικασίας που ακολουθήθηκε κατά την εκπόνησή της.

Κεφάλαιο 1 – Εισαγωγή

Στο πρώτο κεφάλαιο, γίνεται η εισαγωγή στο αντικείμενο της διπλωματικής εργασίας καθώς και γενική αναφορά στην τεχνολογία της εικονικοποίησης. Ακόμη, επισημαίνεται η συμβολή των εικονικών δικτυακών τοπολογιών σε ακαδημαϊκό και ερευνητικό επίπεδο, και στη συνέχεια πραγματοποιείται ο διαχωρισμός της εργασίας σε κεφάλαια, με συνοπτική περιγραφή καθενός από αυτά.

Κεφάλαιο 2 – Σχετικές Τεχνολογίες και Εργαλεία

Στο δεύτερο κεφάλαιο, αναλύονται και επεξηγούνται οι λεπτομέρειες και οι βασικές έννοιες γύρω από την υλοποίηση της εφαρμογής, τα εργαλεία καθώς και οι τεχνολογίες που χρησιμοποιούνται.

Κεφάλαιο 3 – Εικονικοποιημένα Δίκτυα και Δικτυακές Διεπαφές

Στο τρίτο κεφάλαιο, αναφέρονται οι επιλογές δικτύωσης που παρέχονται από το libvirt και το QEMU/KVM. Πιο συγκεκριμένα, παρουσιάζονται αναλυτικά οι διαφορετικοί τύποι δικτύων και διεπαφών, καθώς και παραδείγματα ορισμού δικτύων που υποστηρίζονται, όπως αυτά χρησιμοποιούνται για την υλοποίηση της εφαρμογής.

Κεφάλαιο 4 – Παραγωγή & Παραμετροποίηση της Βασικής Εικόνας του BSDRP

Στο τέταρτο κεφάλαιο, περιγράφονται αναλυτικά τα βήματα δημιουργίας της βασικής εικόνας του λειτουργικού συστήματος που χρησιμοποιείται στην εφαρμογή. Περιλαμβάνεται η εγκατάσταση όλων των εξαρτήσεων και προγραμμάτων, καθώς και η παραμετροποίηση της βασικής εικόνας ώστε να ανταποκρίνεται στις ανάγκες των εξομοιώσεων.

Κεφάλαιο 5 – Εγκατάσταση και Χρήση της Εφαρμογής

Στο πέμπτο κεφάλαιο, περιγράφεται αναλυτικά η διαδικασία εγκατάστασης των εξαρτήσεων και της εφαρμογής, γίνεται μία αναφορά στα στοιχεία που συνθέτουν το γραφικό περιβάλλον και παρουσιάζεται η χρήση της εφαρμογής μέσα από την εκτέλεση δύο σεναρίων.

Κεφάλαιο 6 – Συμπεράσματα και Προοπτικές

Στο έκτο κεφάλαιο, επισημαίνονται τα συμπεράσματα που προκύπτουν από τη διαδικασία υλοποίησης της εφαρμογής αλλά και τη χρήση της. Επίσης, γίνεται αναφορά σε ορισμένες μελλοντικές επεκτάσεις του λογισμικού.

Κεφάλαιο 2

Σχετικές Τεχνολογίες και Εργαλεία

2.1 Εικονικοποίηση

Εικονικοποίηση [1] είναι η τεχνολογία που δημιουργεί ένα επίπεδο αφαίρεσης πάνω από το υλικό ενός υπολογιστή και επιτρέπει την κατανομή υπολογιστικών πόρων με σκοπό τη δημιουργία εικονικών μηχανών. Κατά συνέπεια, δίνει τη δυνατότητα πιο αποδοτικής του υλικού ενός υπολογιστικού συστήματος, διότι μπορούν να λειτουργούν παράλληλα πολλά διαφορετικά λειτουργικά συστήματα και διεργασίες στο ίδιο υλικό.

2.1.1 Εικονικές Μηχανές

Εικονική μηχανή [2] είναι ένας υπολογιστής καθορισμένος από λογισμικό, ο οποίος λειτουργεί εντός ενός υπολογιστικού συστήματος. Η υπολογιστική ισχύς των σύγχρονων υπολογιστικών συστημάτων επιτρέπει την υποστήριξη εικονικών μηχανών, παρέχοντας ευελιξία στους χρήστες μέσω απλών στη χρήση προγραμμάτων για τη διαχείρισή τους. Ενδεικτικά, μερικές χρήσεις που έχουν σήμερα οι εικονικές μηχανές είναι οι εξής:

- Δοκιμή λογισμικού – Οι προγραμματιστές μπορούν να δοκιμάζουν το λογισμικό που δημιουργούν σε πολλά διαφορετικά λειτουργικά συστήματα.
- Λογισμικό σχεδιασμένο για άλλα λειτουργικά συστήματα – Σε πολλές περιπτώσεις, τα προγράμματα αναπτύσσονται για διάφορους λόγους για χρήση σε ένα μόνο λειτουργικό σύστημα.
- Παλαιότερες εκδόσεις λογισμικού – Προγράμματα που έχουν σχεδιαστεί για χρήση σε παλαιότερα λειτουργικά συστήματα πιθανόν να μην υποστηρίζονται από τις πιο σύγχρονες εκδόσεις. Στην περίπτωση αυτή, ο χρήστης μπορεί να εγκαταστήσει μια παλαιότερη έκδοση του λειτουργικού συστήματος σε μια εικονική μηχανή.

Ακόμη, η φιλοξενία εικονικών μηχανών προσφέρεται σήμερα ως υπηρεσία από τους παρόχους υπηρεσιών σύννεφου (cloud providers). Οι πάροχοι αυτοί προσφέρουν στους πελάτες τη δυνατότητα φιλοξενίας εικονικών μηχανών, οι οποίες έχουν ποικίλες χρήσεις, όπως εφαρμογές SaaS (Software-as-a-Service), δημιουργία και διαχείριση αντιγράφων ασφαλείας και υπηρεσίες φιλοξενίας ιστοτόπων ή ηλεκτρονικού ταχυδρομείου.

2.1.2 Hypervisors

Ο hypervisor [3] (ή Virtual Machine Monitor) είναι λογισμικό που μπορεί να δημιουργεί και να διαχειρίζεται εικονικές μηχανές. Μεταξύ των αρμοδιοτήτων του είναι η κατανομή υπολογιστικών πόρων, όπως χρήση κεντρικής μονάδας επεξεργασίας (CPU), κύριας μνήμης και μέσων αποθήκευσης, στις εικονικές μηχανές. Η διαχείριση αυτή επιτυγχάνεται μέσω λογισμικού που τρέχει στο επίπεδο του λειτουργικού συστήματος, όπως πχ. διαχειριστής μνήμης (memory manager), χρονοπρογραμματιστής διεργασιών (process scheduler), η στοίβα εισόδου – εξόδου (I/O stack) και προγράμματα οδήγησης (device drivers).

Οι hypervisors διακρίνονται σε Τύπου 1 και Τύπου 2. Αφενός, οι Τύπου 1 hypervisors (bare metal) αλληλεπιδρούν άμεσα με το υλικό για την υποστήριξη εικονικών μηχανών, χωρίς να χρειάζεται να ζητήσουν πόρους από ένα παραδοσιακό λειτουργικό σύστημα. Προσφέρουν κατά κανόνα καλύτερες επιδόσεις εικονικοποίησης και χρησιμοποιούνται σήμερα σε εξυπηρετητές κέντρων δεδομένων (data centers) και άλλων μεγάλων επιχειρήσεων. Αφετέρου, οι Τύπου 2 hypervisors (hosted) τρέχουν εντός ενός λειτουργικού συστήματος. Στην περίπτωση αυτή χρειάζεται να κατανέμονται οι υπολογιστικοί πόροι μεταξύ των φιλοξενούμενων (guests) με τρόπο ώστε να εξασφαλίζεται η απρόσκοπτη λειτουργία των διεργασιών που τρέχουν στο φιλοξενούν λειτουργικό σύστημα (host).

Στα πλαίσια της διπλωματικής εργασίας θα χρησιμοποιηθεί η μονάδα εικονικοποίησης KVM, η οποία επεκτείνει τη λειτουργικότητα του πυρήνα του λειτουργικού συστήματος, ώστε ο τελευταίος να λειτουργεί ως hypervisor.

2.1.3 KVM

Το KVM (Kernel-based Virtual Machine) [4] είναι μια μονάδα εικονικοποίησης του πυρήνα Linux, η οποία επιτρέπει στον πυρήνα να λειτουργεί ως hypervisor. Προϋποθέτει την ύπαρξη των επεκτάσεων εικονικοποίησης, όπως οι Intel VT-x και AMD-V. Οι τελευταίες δίνουν τη δυνατότητα στον hypervisor να αξιοποιεί περαιτέρω δυνατότητες του υλικού με σκοπό την άμεση βελτίωση των επιδόσεων των εικονικών μηχανών που φιλοξενούνται. Η τεχνική αυτή ονομάζεται εικονικοποίηση υποβοηθούμενη από το υλικό (hardware assisted virtualization). Το KVM υποστηρίζει μέσω hardware-assisted virtualization πολλά διαφορετικά φιλοξενούμενα λειτουργικά συστήματα, όπως είναι οι διανομές Linux, οι διανομές BSD, τα Windows, Haiku, Solaris, macOS και ReactOS. Επίσης, παρέχει υποστήριξη για παραεικονικοποίηση (paravirtualization) μέσω του Virtio API [5], το οποίο παρέχει εξομοιωμένες εκδόσεις φυσικών συσκευών και προγραμμάτων οδήγησης.

2.2 Τεχνολογία λογισμικού

2.2.1 Συστήματα ελέγχου εκδόσεων λογισμικού

Ο έλεγχος εκδόσεων (version control) [6] είναι η τεχνική παρακολούθησης και διαχείρισης των αλλαγών που υφίσταται ο πηγαίος κώδικας ενός λογισμικού. Τα συστήματα ελέγχου εκδόσεων λογισμικού χρησιμοποιούνται από ομάδες σχεδιασμού και ανάπτυξης λογισμικού και παρακολουθούν όλες τις αλλαγές που κάνουν οι προγραμματιστές. Με αυτό τον τρόπο, μπορούν εύκολα να εντοπιστούν προβλήματα και λάθη που δημιουργούνται και καθίσταται απλούστερη και ευκολότερη η διαδικασία διόρθωσης και συντήρησης μεγάλων έργων λογισμικού. Τα σύγχρονα

πακέτα λογισμικού version control, όπως είναι το Git, επιτρέπουν μεταξύ άλλων τη σύγκριση διαφορετικών εκδόσεων του λογισμικού, την παράλληλη ενασχόληση πολλών προγραμματιστών με τον ίδιο πηγαίο κώδικα χωρίς να δημιουργείται σύγχυση, την αυτοματοποίηση δοκιμών και σεναρίων ελέγχου (tests) αλλά και την αυτοματοποίηση των διαδικασιών της συνεχούς ολοκλήρωσης (continuous integration – CI) και της συνεχούς παράδοσης (continuous deployment – CD).

2.2.2 Αρχιτεκτονική Πελάτη – Διακομιστή

Οι υπηρεσίες που προσφέρονται μέσω διαδικτύου (web services) συνηθίζεται να βασίζονται στην αρχιτεκτονική τους στο μοντέλο πελάτη – διακομιστή [7]. Σύμφωνα με το μοντέλο αυτό, οι πελάτες (χρήστες ή προγράμματα σε ένα δίκτυο) υποβάλλουν αιτήματα σε έναν μικρό αριθμό εξυπηρετητών. Οι εξυπηρετητές είναι άλλα προγράμματα σε ένα δίκτυο που ανταποκρίνονται στα αιτήματα του πελάτη. Οι εξυπηρετητές που λειτουργούν σήμερα, προκειμένου να μπορούν να εξυπηρετήσουν τα αιτήματα των πελατών απρόσκοπτα, χρησιμοποιούν υλικό με αυξημένες υπολογιστικές δυνατότητες, ταχείες συνδέσεις δικτύου καθώς και μηχανισμούς εφεδρείας σε περίπτωση που τεθούν εκτός λειτουργίας.

Ένας πελάτης υποβάλλει ένα αίτημα στον εξυπηρετητή και ο εξυπηρετητής αποκρίνεται με ικανοποίηση του αιτήματος του πελάτη. Στο μοντέλο πελάτη – εξυπηρετητή, νέοι εξυπηρετητές μπορούν να προστεθούν σταδιακά με την αύξηση των αιτημάτων και γενικότερα με την αύξηση της ζήτησης. Δηλαδή, το μοντέλο αυτό είναι εύκολα επεκτάσιμο. Πολλοί πελάτες μπορούν να μοιραστούν τους πόρους που παρέχονται από έναν μόνο εξυπηρετητή. Αυτό εξαλείφει την ανάγκη κάθε πελάτη να έχει το δικό του «αντίγραφο» των πόρων. Κάθε υπηρεσία διαδικτύου έχει το δικό της συνδεδεμένο σύνολο πελατών και εξυπηρετητών.

Μερικοί τύποι εξυπηρετητών [8] είναι οι εξής:

- Εξυπηρετητής ιστού
- Εξυπηρετητής βάσης δεδομένων
- Εξυπηρετητής εφαρμογής
- Εξυπηρετητής αρχείων
- Εξυπηρετητής ηλεκτρονικής αλληλογραφίας

2.2.3 Τεχνολογίες Frontend

Το frontend της εφαρμογής είναι δομημένο γύρω από τις τεχνολογίες HTML και CSS, καθώς και τα πιο εξειδικευμένα εργαλεία (βιβλιοθήκες, μηχανές προτυποποίησης) που αναλύονται παρακάτω. Η HTML είναι η γλώσσα σήμανσης που χρησιμοποιείται στο διαδίκτυο. Χρησιμοποιείται για την αποτύπωση περιεχομένου (κείμενο, εικόνες, πίνακες, κ.ά.) σε μία σελίδα ιστού. Η CSS είναι η γλώσσα με την οποία περιγράφεται η παρουσίαση και η μορφοποίηση του περιεχομένου που έχει οργανωθεί σε HTML (ή XML).

2.2.3.1 JavaScript

Η JavaScript [9] είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού που συνδυάζει πολλά διαφορετικά χαρακτηριστικά. Είναι γλώσσα υψηλού επιπέδου, τελευταίας στιγμής μεταγλωττιζόμενη (compiled just-in-time) και υποστηρίζει προγραμματιστικά παραδείγματα (paradigms) όπως αυτά του συναρτησιακού (functional), προστακτικού (imperative), χειρισμού γεγονότων (event-driven) και αντικειμενοστρεφούς (object-oriented) προγραμματισμού. Συμμορφώνεται στο πρότυπο ECMAScript, ο ρόλος του οποίου είναι η διασφάλιση της συμβατότητας της γλώσσας JavaScript με όλους τους φυλλομετρητές ιστού. Σήμερα έχει υπολογιστεί ότι χρησιμοποιείται από το 97% των ιστοτόπων του διαδικτύου [10].

2.2.3.2 JQuery

Η JQuery [11] είναι μια βιβλιοθήκη της Javascript με αρκετές δυνατότητες, όπως διάσχιση αρχείων HTML και αλλαγή στοιχείων, χειρισμό γεγονότων, εφαρμογή κινούμενων γραφικών (animations) και υποστήριξη AJAX (Asynchronous Javascript and XML). Αποτελεί ένα ισχυρό εργαλείο για τους προγραμματιστές και σχεδιαστές ιστοτόπων και διαδικτυακών εφαρμογών, ενθαρρύνοντας τη δημιουργία επεκτάσεων και επιπέδων αφαίρεσης πάνω από τις εφαρμογές. Συγχρόνως, προωθεί τη δημιουργία δυναμικών εφαρμογών web και τη χρήση διασυνδέσεων API.

Αυτά τα στοιχεία της JQuery την καθιστούν ιδανική επιλογή, σε συνδυασμό με τις άλλες τεχνολογίες που αναφέρονται, για την ανάπτυξη μιας διαδραστικής διεπαφής χρήστη στο πλαίσιο της παρούσας εργασίας. Συγκεκριμένα, στην εφαρμογή αξιοποιούνται χαρακτηριστικά στοιχεία της JQuery (έκδοση 3.6.0) όπως αυτά που προαναφέρθηκαν, αλλά και της επέκτασης JQuery UI (έκδοση 1.12.1).

2.2.4 Τεχνολογίες backend

2.2.4.1 Python

Η Python [12] είναι μια υψηλού επιπέδου, διερμηνευόμενη, διαδραστική, φορητή, αντικειμενοστρεφής γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως σήμερα κυρίως λόγω της απλότητας στη σύνταξη, της αναγνωσιμότητας και των διεπαφών που υποστηρίζει. Συγκεκριμένα, έχουν αναπτυχθεί πολλές επεκτάσεις και διεπαφές γύρω από την Python μέσω των οποίων μπορεί να αλληλεπιδράσει μεταξύ των άλλων με λειτουργικά συστήματα, συστήματα διαχείρισης παραθύρων, άλλες γλώσσες προγραμματισμού και υλικό. Υποστηρίζει προγραμματιστικά υποδείγματα όπως αυτό του συναρτησιακού προγραμματισμού και του διαδικαστικού προγραμματισμού.

Η παρούσα διπλωματική εργασία βασίζεται στην Python τόσο για την αλληλεπίδραση με τις εικονικές μηχανές, όσο και για τη μοντελοποίηση του backend και των δεδομένων. Συγκεκριμένα, ο βασικός πυρήνας της εφαρμογής συντάσσεται με την έκδοση 3.8.6 της Python και τα εργαλεία Flask και libvirt-python, τα οποία αναλύονται παρακάτω.

Flask

Το Flask [13] είναι ένα WSGI (Web Server Gateway Interface) μικρο-πλαίσιο (microframework) της Python. Δημιουργήθηκε από την Pallets Projects και αποτελεί μία από τις πλέον διαδεδομένες λύσεις στο χώρο των εφαρμογών διαδικτύου. Το Flask στηρίζεται στα εργαλεία Jinja και Werkzeug, τα οποία αναλύονται παρακάτω. Με το πρόθεμα “micro” εννοείται ότι το συγκεκριμένο framework διατηρεί

μια απλή και επεκτάσιμη βάση, δίνοντας τη δυνατότητα στους προγραμματιστές να λαμβάνουν εκείνοι αποφάσεις για το ποια εργαλεία ή επεκτάσεις θα χρησιμοποιήσουν (μηχανές προτυποποίησης, συστήματα διαχείρισης βάσεων δεδομένων, εξυπηρετητές WSGI, REST APIs, άλλα frameworks) ανάλογα με την εφαρμογή.

Werkzeug

Το Werkzeug [14] είναι μια περιεκτική WSGI βιβλιοθήκη εφαρμογών ιστού της Python που δημιουργήθηκε από την Pallets Projects. Ο ρόλος της στην παρούσα εφαρμογή είναι συμπληρωματικός, καθώς ενδείκνυται η χρήση του μαζί με το Flask και τη Jinja για τη δημιουργία διαδικτυακών εφαρμογών βασισμένων σε Python, αλλά το πεδίο εφαρμογής του δεν περιορίζεται εκεί. Μπορεί να χρησιμοποιηθεί αυτούσιο (περιλαμβάνει έναν εξυπηρετητή WSGI) μαζί με μία μηχανή προτυποποίησης.

2.2.4.2 libvirt

Το libvirt [15] είναι ένα εργαλείο εικονικοποίησης που προσφέρει ένα φορητό API για τη γλώσσα C με σκοπό τη διαχείριση εικονικών μηχανών. Υποστηρίζει όλους τους ευρέως διαδεδομένους hypervisors όπως KVM, Xen, LXC, ESX, VMware, Hyper-V, Virtualbox και bhyve. Το API της C με τη σειρά του έχει συνδεθεί με πολλές άλλες γλώσσες προγραμματισμού, όπως είναι οι Python, Perl, PHP, GO, Java και OCaml. Το libvirt αποτελεί δωρεάν και ανοικτού κώδικα λογισμικό που δημιουργήθηκε από τη Red Hat και σήμερα βρίσκεται εφαρμογή σε λύσεις λογισμικού βασισμένες στο υπολογιστικό σύννεφο (cloud-based solutions) [16].

Το libvirt είναι δομημένο στην αρχιτεκτονική πελάτη – εξυπηρετητή. Στον εξυπηρετητή εκτελείται η διεργασία libvirtd, ο σκοπός της οποίας είναι να δέχεται αιτήματα από τους πελάτες και να τα προωθεί στον κατάλληλο hypervisor. Η επικοινωνία μεταξύ libvirt και των hypervisors πραγματοποιείται μέσω του οδηγού (driver) που προσδιορίζει τον εκάστοτε hypervisor σε μορφή URI (RFC 2396) [17], ακολουθούμενο προαιρετικά από τη μέθοδο επικοινωνίας (TCP, SSH, κ.ά.). Ενδεικτικά, για να πραγματοποιηθεί σύνδεση με τον QEMU/KVM εντός του ίδιου συστήματος, χρησιμοποιείται το URI

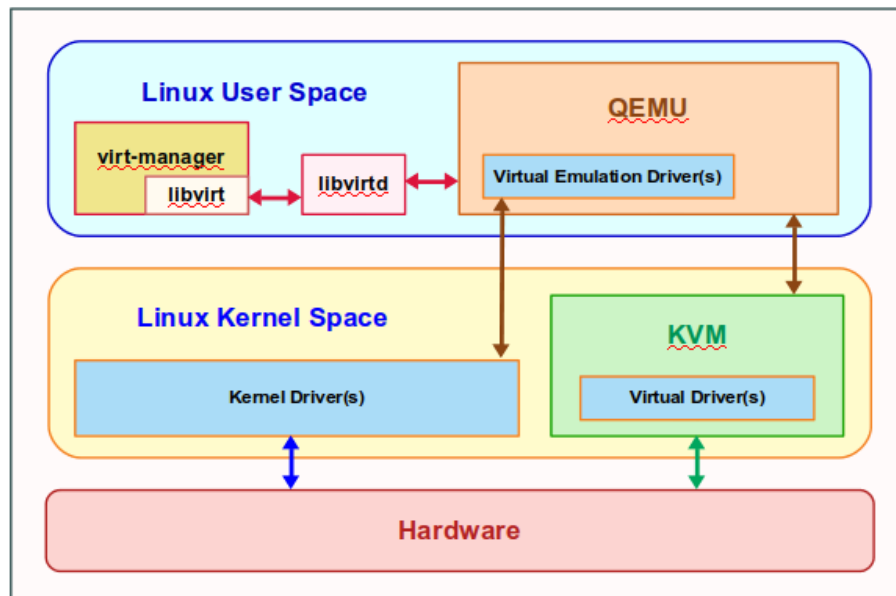
```
qemu:///system
```

Η συγκεκριμένη διεύθυνση αναφέρεται στη διεργασία που libvirt εκτελείται από το ίδιο το σύστημα, εν αντιθέσει με τη διεύθυνση session, η οποία αναφέρεται στη διεργασία που εκτελείται από το μη εξουσιοδοτημένο (unprivileged) χρήστη qemu [18]. Η πρώτη έχει πρόσβαση σε όλους τους υπολογιστικούς πόρους που είναι διαθέσιμοι από το λειτουργικό σύστημα, ενώ η δεύτερη δεν έχει πρόσβαση στους πόρους που ανήκουν αποκλειστικά στον υπερχρήστη (root). Άλλα παραδείγματα διευθύνσεων URI αποτελούν τα εξής:

```
xen+ssh://user@host
```

```
qemu+tcp://localhost:5000/default
```

Στο πρώτο παράδειγμα, γίνεται κρυπτογραφημένη σύνδεση στον hypervisor Xen που βρίσκεται στον εξυπηρετητή host μέσω SSH με όνομα χρήστη user. Στο δεύτερο παράδειγμα, γίνεται μη κρυπτογραφημένη σύνδεση στον QEMU πάνω από το πρωτόκολλο μεταφοράς TCP στο τοπικό σύστημα, στη θύρα 5000 στο default path. Στην Εικόνα 1 παρουσιάζεται συνοπτικά η αλληλεπίδραση μεταξύ libvirt και του λειτουργικού συστήματος κατά τη διαχείριση εικονικών μηχανών.



Εικόνα 1 Αποτύπωση του libvirt σε ένα τυπικό λειτουργικό σύστημα

Το libvirt αποτελεί τη βάση για εργαλεία χρήστη που έχουν αναπτυχθεί ώστε να διευκολύνουν τη διαχείριση εικονικών μηχανών και hypervisors. Μερικά από αυτά είναι τα εξής:

- `virsh`, το οποίο είναι λογισμικό γραμμής εντολών για τη διαχείριση εικονικών μηχανών, εικονικοποιημένων δικτύων, εικονικοποιημένων μονάδων δίσκου, τερματικών, αλλά και του ίδιου του hypervisor.
- `virt-clone`, το οποίο είναι ένα εργαλείο γραμμής εντολών για την δημιουργία κλώνων εικονικών μηχανών. Υποστηρίζει συγκεκριμένα συστήματα αρχείων των φιλοξενούμενων, όπως το EXT4.
- `virt-install`, το οποίο είναι ένα εργαλείο γραμμής εντολών που επιτρέπει την δημιουργία εικονικών μηχανών. Το εργαλείο αυτό παρέχει πολλές επιλογές παραμετροποίησης, ενώ υποστηρίζει την εκκίνηση των φιλοξενούμενων τόσο από οπτικά μέσα όσο και μέσω δικτύου (NFS, PXE, κ.ά.).
- `virt-manager`, το οποίο αποτελεί μία από τις πιο διαδεδομένες εφαρμογές γραφικού περιβάλλοντος για τη διαχείριση των εικονικών μηχανών. Υποστηρίζει τη δυνατότητα σύνδεσης σε καθορισμένες διευθύνσεις (URIs), όπως αυτές ορίζονται από το libvirt και έχουν ήδη αναφερθεί. Αυτό επιτρέπει τη διαχείριση εικονικών μηχανών από πολλούς διαφορετικούς hypervisors, τόσο σε τοπικό επίπεδο όσο και σε απομακρυσμένα υπολογιστικά συστήματα.

Στην παρούσα διπλωματική χρησιμοποιείται η έκδοση 7.1.0 του API του libvirt για τη γλώσσα Python (`libvirt-python`).

2.2.5 Άλλες τεχνολογίες

Σε αυτή την υποενότητα αναφέρονται εργαλεία που δεν ανήκουν σε κάποια από τις προαναφερθείσες κατηγορίες.

2.2.5.1 Jinja

Η Jinja [19] είναι μία ταχεία, εκφραστική και επεκτάσιμη μηχανή προτυποποίησης (templating engine) που χρησιμοποιείται σήμερα ευρέως ως συνδετικός κρίκος μεταξύ ενός τυπικού Flask backend και του frontend. Μπορεί να συνυπάρχει μαζί με κώδικα HTML στο ίδιο αρχείο, ενώ ειδικοί χαρακτήρες του προτύπου Jinja μεταφράζονται σε απλό κείμενο κατά την εκτέλεση (rendering). Κάθε πρότυπο μπορεί να περιέχει τα εξής:

- άλλα πρότυπα Jinja (υποστηρίζει κληρονομικότητα και ενσωμάτωση μέσω των εντολών `inherit` και `include` αντίστοιχα)
- μακροεντολές (macros) που ορίζονται εντός του προτύπου
- Πρότυπα HTML με δυνατότητα autoescaping
- υποστήριξη για AsyncIO (ασύγχρονη κλήση συναρτήσεων)
- υποστήριξη για διεθνοποίηση (internationalization/i18n) με τη χρήση της Babel
- λογική επιλογής και επανάληψης (if, for, κλπ.)

Η Jinja2 [20] (τρέχουσα έκδοση της Jinja) αποτελεί μια χρήσιμη μηχανή που επιτρέπει το διαχωρισμό του πηγαιού κώδικα Python από το frontend. Ακόμη, δίνει στους σχεδιαστές ισχυρά εργαλεία με τα οποία μπορούν να εύκολα να ενσωματώσουν λογική και μεταβλητές στα πρότυπα που δημιουργούν, χωρίς να περιορίζονται από τον κώδικα Python.

2.3 Λειτουργικά Συστήματα και Πυρήνες

2.3.1 Linux

Ο πυρήνας Linux [21] είναι ένας αρθρωτός, μονολιθικός πυρήνας λειτουργικού συστήματος. Δημιουργήθηκε το 1991 και αποτελεί ένα από τα πιο χαρακτηριστικά παραδείγματα ελεύθερου και ανοικτού κώδικα λογισμικού. Σήμερα συναντάται σε πολλούς διαφορετικούς τύπους υπολογιστικών συστημάτων και συσκευών, μεταξύ των οποίων είναι προσωπικοί υπολογιστές, κινητά τηλέφωνα, ενσωματωμένα συστήματα, εξυπηρετητές και υπερυπολογιστές. Κύρια χαρακτηριστικά του Linux είναι τα εξής:

- Είναι αρθρωτός πυρήνας [22], δηλαδή επιτρέπει στο χρήστη να προσθέτει περαιτέρω λειτουργικότητα φορτώνοντας υπομονάδες πυρήνα (φορτώσιμα αρθρώματα) κατά το δοκούν. Παραδείγματα τέτοιων αρθρωμάτων αποτελούν τα προγράμματα οδήγησης (για την υποστήριξη υλικού), τα αρθρώματα για συστήματα αρχείων, δαίμονες και κλήσεις συστήματος που δεν υποστηρίζονται από το βασικό πυρήνα [23].

- Είναι μονολιθικός πυρήνας [24], με την έννοια ότι λειτουργεί εξ ολοκλήρου στο χώρο πυρήνα (kernel space) και ενσωματώνει σε μία βασική διεργασία όλες τις επιμέρους υπομονάδες που τον απαρτίζουν. Αυτό συμβάλλει στην ελαχιστοποίηση του αποτυπώματος του πυρήνα στη μνήμη.

Η εφαρμογή που περιγράφεται παρακάτω έχει σχεδιαστεί και υλοποιηθεί στη διανομή Ubuntu (έκδοση 20.10), η οποία είναι μια δημοφιλής διανομή βασισμένη στον πυρήνα Linux. Το Ubuntu επιλέχθηκε διότι αφενός χρησιμοποιείται τόσο σε προσωπικούς υπολογιστές όσο και σε σύνθετα υπολογιστικά συστήματα (μεγάλες επιχειρήσεις, κέντρα δεδομένων, πανεπιστήμια, ερευνητικά κέντρα), και αφετέρου παρέχει σε τακτική βάση ενημερώσεις ασφαλείας και υποστήριξη νέων προγραμμάτων οδήγησης. Ακόμη, υποστηρίζεται από μια μεγάλη κοινότητα που αποτελείται από χρήστες προσωπικών υπολογιστών, επιστήμονες, ερευνητές, μηχανικούς, κέντρα ερευνών, δημόσιους φορείς αλλά και τεχνολογικούς κολοσσούς.

2.3.2 FreeBSD

Το FreeBSD [25] είναι ένα *nix (παρόμοιο με Unix) δωρεάν και ανοικτού κώδικα λειτουργικό σύστημα που προέκυψε από το BSD (Berkeley Software Distribution). Δημιουργήθηκε το 1993 και αποτελεί το πιο δημοφιλές της οικογένειας λειτουργικών συστημάτων BSD. Έχει αρκετές διαφορές με το Linux, μεταξύ των οποίων οι εξής:

- Το FreeBSD είναι ένα ολοκληρωμένο λειτουργικό σύστημα στο οποίο περιλαμβάνεται ο πυρήνας, τα προγράμματα οδήγησης για την υποστήριξη υλικού, η τεκμηρίωση και ο χώρος χρήστη (userland¹) δηλαδή εφαρμογές χρήστη, δαίμονες, βιβλιοθήκες και άλλες μονάδες συστήματος [26]. Αντιθέτως, ο όρος Linux αναφέρεται μόνο στον πυρήνα. Ως εκ τούτου, το λογισμικό του χώρου χρήστη που ενσωματώνεται σε κάθε διανομή αποτελεί επιλογή του δημιουργού της κάθε διανομής.
- Το FreeBSD διανέμεται κάτω από την ανεκτική (permissive) άδεια BSD, ενώ το Linux διατίθεται με την άδεια GPL. Με λίγα λόγια, η άδεια BSD είναι πιο αυστηρή, καθώς επιτρέπει την αναδιανομή αντιγράφων λογισμικού αποκλειστικά με χρήση της ίδιας άδειας, ενώ η άδεια GPL επιτρέπει την αναδιανομή με χρήση άλλων αδειών, μεταξύ των οποίων και αδειών κλειστού κώδικα.

2.3.2.1 BSDRP

Το BSDRP [27] (BSD Router Project) είναι μια διανομή του BSD που εστιάζει στη χρήση σε δίκτυα υπολογιστών. Συγκεκριμένα, στο λειτουργικό σύστημα περιλαμβάνονται ευρέως διαδεδομένες σουίτες λογισμικού για δικτύωση, όπως τα FRRouting [28] (απόγονος του Quagga) και BIRD[29] (BIRD Internet Routing Daemon). Τα λογισμικά αυτά υποστηρίζουν πολλά από τα πρωτόκολλα δικτύωσης που χρησιμοποιούνται σήμερα, περιλαμβανομένων των πρωτοκόλλων διανυσμάτων απόστασης (distance vector), κατάστασης ζεύξεων (link state) και διανυσμάτων διαδρομών (path vector). Ενδεικτικά αναφέρονται τα πρωτόκολλα RIP, OSPF, BGP, BABEL, EIGRP και IS-IS.

¹ Με τον όρο userland εννοείται γενικά οποιοδήποτε λογισμικό δεν ανήκει στο χώρο πυρήνα.

Η συγκεκριμένη διανομή έχει αξιολογηθεί και έχει επιλεγθεί για χρήση στη διπλωματική εργασία. Συγκεκριμένα, έχει τροποποιηθεί καταλλήλως, όπως παρουσιάζεται σε επόμενη ενότητα, ώστε να εξυπηρετήσει το ρόλο του δρομολογητή σε περιβάλλον εξομίωσης δικτύων υπολογιστών.

Κεφάλαιο 3

Εικονικοποιημένα Δίκτυα και Δικτυακές Διεπαφές

Ένας από τους κύριους λόγους που το libvirt API και το QEMU/KVM χρησιμοποιούνται στην παρούσα διπλωματική εργασία είναι η πληθώρα επιλογών που παρέχονται σχετικά με τη δικτύωση εικονικών μηχανών. Στην ενότητα αυτή παρουσιάζονται αναλυτικά οι δυνατότητες που προσφέρουν τα εργαλεία εικονικοποίησης για την περιγραφή των διαφορετικών τύπων δικτύων και δικτυακών διεπαφών που μπορούν να χρησιμοποιηθούν στην εξομίωση τοπολογιών δικτύων.

3.1 Εικονικοποιημένα Δίκτυα

3.1.1 Γενικά Στοιχεία

Τα δίκτυα που δημιουργούνται στο libvirt περιγράφονται σε XML [30]. Το στοιχείο ρίζας είναι πάντοτε το `network`, ενώ τα εμφωλευμένα σε αυτό στοιχεία προορίζονται για τον προσδιορισμό διαφόρων παραμέτρων του κάθε δικτύου. Κατ' αρχάς, στο δεύτερο επίπεδο εμφώλευσης ορίζονται παράμετροι όπως το όνομα του δικτύου, ο μοναδικός κωδικός ταυτοποίησης (UUID²), οι παράμετροι της γέφυρας (όνομα, STP, κ.ά.), η διεύθυνση MAC, η MTU³, οι παράμετροι DNS, καθώς και η διεύθυνση IP⁴.

3.1.2 Επιλογές Διασύνδεσης

Για τη διασύνδεση των φιλοξενούμενων παρέχονται μεταξύ των άλλων οι επιλογές σύνδεσης με προώθηση (forwarding) και σύνδεσης σε γέφυρα (χωρίς προώθηση).

3.1.2.1 Σύνδεση σε Γέφυρα

Η λύση αυτή στοχεύει στη σύνδεση όλων των φιλοξενούμενων σε μία γέφυρα. Με αυτό τον τρόπο επιτυγχάνεται επικοινωνία μεταξύ τους, αλλά και απομόνωση του τοπικού αυτού δικτύου από άλλα δίκτυα και από το δίκτυο του φιλοξενούντος. Η σύνδεση δηλώνεται με το στοιχείο `bridge` στο δεύτερο επίπεδο. Σε αυτό τον τύπο σύνδεσης μπορούν να καθοριστούν τα παρακάτω:

- Το πεδίο `name`, δηλαδή το όνομα της γέφυρας. Στο πεδίο αυτό δίνονται συνήθως ονόματα της μορφής `vbrbr[x]`, όπου `x` ο αριθμός της (εικονικοποιημένης) γέφυρας. Σε περίπτωση που το όνομα δεν οριστεί, ανατίθεται από το libvirt ένα όνομα με την παραπάνω μορφή.

² Προαιρετικό πεδίο. Αν δεν δοθεί, τότε ανατίθεται ένα τυχαίο UUID από το libvirt σε κάθε δίκτυο.

³ Maximum Transmission Unit

⁴ Χρησιμοποιείται ως προεπιλεγμένη πύλη για τις συσκευές που είναι συνδεδεμένες στο δίκτυο

- Το πεδίο `stp`, με επιλογές `on` ή `off`, για την ενεργοποίηση ή την απενεργοποίηση αντίστοιχα του πρωτοκόλλου STP.
- Το πεδίο `delay` για την εισαγωγή καθυστέρησης στην προώθηση πακέτων από τη γέφυρα.
- Το πεδίο `macTableManager`, με τιμές `kernel` (εκ προεπιλογής) και `libvirt`, για την ανάθεση στο λειτουργικό σύστημα ή στο `libvirt` αντίστοιχα της διαχείρισης του πίνακα διευθύνσεων MAC (MAC address table). Σε κάποιες περιπτώσεις παρατηρούνται αυξημένες επιδόσεις με τη χρήση του `libvirt` για το σκοπό αυτό.

3.1.2.2 Προώθηση

Μέσω της λειτουργίας προώθησης, το δίκτυο μπορεί να επικοινωνήσει με το δίκτυο του φιλοξενούντος. Η προώθηση μπορεί να γίνει με τους παρακάτω τρόπους, ανάλογα με την τιμή που ορίζεται στο πεδίο `forward`:

- `nat`, για μετάφραση διευθύνσεων με το πρωτόκολλο NAT. Στην περίπτωση αυτή, το τείχος προστασίας επιτρέπει την κίνηση μεταξύ των δύο δικτύων (φιλοξενούντος και φιλοξενούμενου), εκτός αν έχει οριστεί συγκεκριμένη συσκευή για την προώθηση, οπότε το τείχος προστασίας επιτρέπει την κίνηση μόνο στη συσκευή αυτή. Προαιρετικά, μπορεί να οριστεί προώθηση σε συγκεκριμένες θύρες (port forwarding) μέσω των στοιχείων `port` εντός του στοιχείου `nat`.
- `route`, για δρομολόγηση (από το δίκτυο του φιλοξενούμενου στο δίκτυο του φιλοξενούντος). Στη λειτουργία αυτή η κίνηση μεταξύ των δύο δικτύων δεν περιορίζεται, εκτός αν έχει οριστεί συγκεκριμένη συσκευή για την προώθηση, οπότε και το τείχος προστασίας επιτρέπει μόνο την κίνηση στη συσκευή αυτή.
- `open`, για προώθηση πακέτων χωρίς περιορισμούς από το τείχος προστασίας. Στην περίπτωση αυτή υποτίθεται ότι ο δρομολογητής του τοπικού δικτύου (του φιλοξενούμενου) έχει τις κατάλληλες εγγραφές για την προώθηση πακέτων προς το δίκτυο του φιλοξενούντος.
- `bridge`, είτε για τη χρήση μίας υπάρχουσας γέφυρας που έχει καθοριστεί στο δίκτυο του φιλοξενούντος, είτε για τη χρήση υπάρχουσας γέφυρας τύπου Open vSwitch, είτε για τη δικτύωση με απ' ευθείας διασυνδέσεις τύπου `macvtap`.
- `private`, για τη δικτύωση με `macvtap` κάθε εικονικής μηχανής ξεχωριστά στο δίκτυο του φιλοξενούντος. Στην περίπτωση αυτή, οι εικονικές μηχανές είναι απομονωμένες μεταξύ τους. Προαιρετικά, μπορεί να συνδυαστεί με το πρωτόκολλο 802.1Qbh (με ετικέτες VN), μέσω του οποίου ταυτοποιούνται οι εικονικοποιημένες ή εξομοιούμενες δικτυακές διεπαφές [31] (προϋποθέτει την υποστήριξη του πρωτοκόλλου από ένα εξωτερικό switch).
- `vepa`, για τη δρομολόγηση με το πρωτόκολλο 802.1Qbg (VEPA⁵). Το πρωτόκολλο αυτό χρησιμοποιείται για τον έλεγχο της διαδρομής των πακέτων μεταξύ `hypervisor` και φιλοξενούμενων εικονικών μηχανών. Συγκεκριμένα, η κίνηση μεταξύ εικονικών μηχανών περνάει υποχρεωτικά από ένα εξωτερικό switch που υποστηρίζει το πρωτόκολλο αυτό.

⁵ Virtual Ethernet Port Aggregator

- `passthrough`, όπου χρησιμοποιείται η απ' ευθείας σύνδεση `macvtap` με μία φυσική διεπαφή, η οποία καθορίζεται από το στοιχείο `interface` (στο τρίτο επίπεδο). Κάθε τέτοια εγγραφή δεσμεύει την εκάστοτε φυσική διεπαφή, οπότε η τελευταία δεν μπορεί να χρησιμοποιηθεί από άλλη εικονική μηχανή.
- `hostdev`, όπου η εικονική μηχανή συνδέεται απ' ευθείας με μία φυσική διεπαφή μέσω `PCI passthrough` (υποστηρίζονται διεπαφές `PCI` και `PCIe`). Λόγω περιορισμών στη σχεδίαση των προγραμμάτων οδήγησης δικτυακών διεπαφών `PCI`, υποστηρίζονται αποκλειστικά οι συσκευές με την προδιαγραφή `SR-IOV`⁶. Προαιρετικά μπορεί να συνδυαστεί με ετικέτες `VLAN` ή `VN`.

Στην πράξη, οι επιλογές `private`, `vnet`, `passthrough` και `hostdev` χρησιμοποιούνται ευρέως για την αξιοποίηση εξειδικευμένων λειτουργιών των φυσικών `switches`, οι οποίες είτε δεν έχουν υλοποιηθεί σε επίπεδο `hypervisor`, είτε υστερούν σε επιδόσεις σε σχέση με τα `switches`.

3.1.3 Ποιότητα Υπηρεσίας

Μία ακόμη λειτουργία που μπορεί να αξιοποιηθεί από τους φιλοξενούμενους είναι αυτή της ποιότητας υπηρεσίας (`Quality of Service – QoS`). Οι βασικές παράμετροι της λειτουργίας αυτής σχετίζονται με το εύρος ζώνης που κατανέμεται σε ένα δίκτυο. Συγκεκριμένα, στο στοιχείο `bandwidth` εμφωλεύονται τα `inbound` και `outbound`, τα οποία έχουν τις παραμέτρους `average`, `peak` και `burst` για τον προσδιορισμό του μέσου όρου, του μέγιστου και του εύρους ζώνης και του εύρους ζώνης απότομης μεταφοράς (`burst`) αντίστοιχα, όπως φαίνεται στο παρακάτω απόσπασμα.

```
<bandwidth>
  <inbound average='1000' peak='5000' burst='5120' />
  <outbound average='128' peak='256' burst='256' />
</bandwidth>
```

3.1.4 Portgroups

Μία ακόμη δυνατότητα που προσφέρεται κατά τη δημιουργία νέων εικονικοποιημένων δικτύων είναι ο καθορισμός `portgroups`. Αυτά χρησιμοποιούνται για την ομαδοποίηση δικτυακών διασυνδέσεων σε κλάσεις, ανάλογα με το επίπεδο ή τύπο υπηρεσίας που προσφέρεται. Τα βασικά στοιχεία ενός `portgroup` είναι το όνομα (`name`), το `virtualport` και προαιρετικά το εύρος ζώνης (`bandwidth`). Στην περίπτωση που η διεπαφή της εκάστοτε εικονικής μηχανής καθορίζει ένα `portgroup`, οι πληροφορίες αυτής της κλάσης θα συγχωνευτούν στη διαμόρφωση της διεπαφής. Κάθε `bandwidth` που καθορίζεται στο αρχείο `XML` της εικονικής μηχανής υπερισχύει έναντι οποιασδήποτε ρύθμισης στο επιλεγμένο `portgroup` ενώ εάν έχουν οριστεί `virtualports` στο `portgroup`, τότε αυτά θα συγχωνευθούν.

3.1.5 Στατικές Διαδρομές

Ο καθορισμός στατικών διαδρομών εντός των δικτύων είναι χρήσιμος στην περίπτωση που είναι επιθυμητή η δήλωση διαδρομών προς τα δίκτυα των φιλοξενούμενων, οι οποίες δεν έχουν

⁶ Single Root I/O Virtualization

γνωστοποιηθεί με κάποιον τρόπο στον hypervisor αλλά είναι προσβάσιμες μόνο από τους φιλοξενούμενους. Όπως φαίνεται στο παρακάτω παράδειγμα, είναι δυνατό να οριστεί μια εικονική διεπαφή δικτύου χωρίς διευθύνσεις IP, παραλείποντας εξ ολοκλήρου τα στοιχεία ip, dns και forward. Τέτοια δίκτυα είναι χρήσιμα για να παρέχουν στον hypervisor δυνατότητα διασύνδεσης σε δίκτυα τα οποία είναι προσβάσιμα μόνο μέσω φιλοξενούμενων και είναι γνωστά ως very private ή very isolated.

```
<network ipv6='yes'>
  <name>very_private</name>
  <bridge name='virbr1' stp='on' />
  <mac address='52:54:00:5D:C7:9E' />
</network>
```

Ένας φιλοξενούμενος με δυνατότητα σύνδεσης τόσο στο δίκτυο μόνο για φιλοξενούμενους όσο και σε άλλο δίκτυο που είναι άμεσα προσβάσιμο από τον hypervisor μπορεί να λειτουργήσει ως πύλη μεταξύ των δικτύων. Η πληροφορία για τη δρομολόγηση από το δίκτυο του hypervisor στο very isolated δίκτυο μπορεί να συμπεριληφθεί μέσω της στατικής διαδρομής.

Ένα τυπικό παράδειγμα δήλωσης στατικής διαδρομής, όπως φαίνεται παρακάτω, μπορεί να περιλαμβάνει τα εξής:

- διεύθυνση IP (που χρησιμοποιείται ως προεπιλεγμένη πύλη).
- DHCP pool του εξυπηρετητή DHCP για τη δυναμική απόδοση διευθύνσεων στις διεπαφές των φιλοξενούμενων.
- το στοιχείο route, για τη δήλωση της στατικής διαδρομής. Σε αυτό δηλώνονται οι εγγραφές address (διεύθυνση δικτύου προορισμού), prefix (πρόθεμα δικτύου), gateway (πύλη) και metric⁷ (μετρική).

```
<ip address="192.168.100.1" netmask="255.255.255.0">
  <dhcp>
    <range start="192.168.100.50" end="192.168.100.127" />
  </dhcp>
</ip>
<route address="192.168.200.0" prefix="28" gateway="192.168.100.200" />
<ip family="ipv6" address="2001:db8:ca2:2::1" prefix="64" />
<route family="ipv6" address="2001:db9:4:1::" prefix="64"
gateway="2001:db8:ca2:2::3" metric='2' />
```

⁷ Το metric υποδηλώνει τη σειρά προτίμησης κατά τον ορισμό διαδρομών. Αν στο παράδειγμα υπήρχε διαδρομή με το ίδιο ταίριασμα (δικτύου και προθέματος) με μικρότερο metric, τότε θα προτιμούνταν αυτή.

Στο παράδειγμα αυτό, αν υπήρχε στατική διαδρομή για το δίκτυο 2001:db9:4:1::/64 (δεύτερη εγγραφή route) με μικρότερο metric ή χωρίς δηλωμένο metric (το προεπιλεγμένο metric είναι 1), τότε θα προτιμούνταν αυτή.

3.1.6 VLANs

Το πρωτόκολλο 802.1Q (ετικέτες VLAN) μπορεί να χρησιμοποιηθεί αν στη δικτύωση των φιλοξενουμένων υποστηρίζεται διαφάνεια ετικετών VLAN. Οι διασυνδέσεις που υποστηρίζουν τη χρήση ετικετών VLAN διαφανών ως προς τους φιλοξενούμενους είναι μεταξύ άλλων οι bridge συνδέσεις σε Open vSwitch και οι hostdev συνδέσεις σε SR-IOV. Επισημαίνεται ότι οι επιλογές δικτύωσης που προσφέρονται από τον πυρήνα και το libvirt δεν υποστηρίζουν τις διαφανείς ετικέτες VLAN. Επίσης, η λειτουργία trunking με πολλαπλές ετικέτες VLAN υποστηρίζεται αποκλειστικά στις συνδέσεις Open vSwitch.

Στην πράξη, η λειτουργία ετικετών VLAN μπορεί να εφαρμοστεί σε δίκτυα με το στοιχείο `vlan` (στο δεύτερο επίπεδο εμφώλευσης). Όπως φαίνεται στο παρακάτω παράδειγμα, μπορούν να δηλωθούν πολλαπλές ετικέτες σε trunk, ενώ μπορεί επίσης να οριστεί ετικέτα VLAN σε portgroups.

```
<network>
  <name>ovs-net</name>
  <forward mode='bridge' />
  <bridge name='ovsbr0' />
  <virtualport type='openvswitch'>
    <parameters interfaceid='09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f' />
  </virtualport>
  <vlan trunk='yes'>
    <tag id='42' nativeMode='untagged' />
    <tag id='47' />
  </vlan>
  <portgroup name='dontpanic'>
    <vlan>
      <tag id='42' />
    </vlan>
  </portgroup>
</network>
```

3.2 Εικονικοποιημένες και Εξομοιούμενες Δικτυακές Διεπαφές

Στην ενότητα αυτή θα αναλυθεί η σύνδεση εικονικών μηχανών στο δίκτυο μέσω δικτυακών διεπαφών. Κατ' αρχάς, η κάθε διεπαφή καθορίζεται μέσω του στοιχείου `interface`, το οποίο είναι εμφωλευμένο εντός του `devices`. Η πρώτη εγγραφή που ορίζεται είναι ο τρόπος σύνδεσης της διεπαφής (`type`), που μπορεί να πάρει τις τιμές `network`, `direct`, `ethernet`, `bridge`, `hostdev`, `udr`, `vhostuser`, `user`, ανάλογα με τη λειτουργία που είναι επιθυμητή, ενώ συνήθως χρησιμοποιείται η λειτουργία `network`.

Στο επόμενο επίπεδο, ορίζονται οι παράμετροι της κάθε διεπαφής. Τα στοιχεία που περιέχουν την πληροφορία αυτή είναι τα εξής:

- `source`, που περιέχει τις εγγραφές για το δίκτυο στο οποίο θα συνδεθεί η διεπαφή (`network`), τη γέφυρα (`bridge`).
- `mac`, με την εγγραφή `address` για τον προσδιορισμό της διεύθυνσης MAC της διεπαφής.
- `target` (της μορφής⁸ `vnet[x]`, `vif[x]`, `macvtap[x]`, `macvlan[x]`), που περιέχει την εγγραφή `dev` για τη συσκευή τύπου TUN/TAP στην οποία συνδέεται η διεπαφή αναλόγως του τρόπου διασύνδεσης.
- `model`, δηλαδή το μοντέλο της διεπαφής (στην εγγραφή `type`).
- το ψευδώνυμο (`alias`), που χρησιμεύει συνήθως στη συνάθροιση (`aggregation`) διεπαφών.
- `address`, που περιέχει τις εγγραφές για τον τύπο (`type`) της σύνδεσης της εικονικής συσκευής της διεπαφής στην εικονική μηχανή (συνήθως `pci`), τα `domain`, `bus`, `slot` και `function` που σχετίζονται με την εσωτερική οργάνωση του εικονικοποιημένου ή εξομοιούμενου υλικού εντός του `libvirt`.

Σε ό,τι αφορά τα μοντέλα δικτυακών διεπαφών που υποστηρίζονται από το QEMU/KVM, η κυριότερη κατηγοριοποίηση σχετίζεται με την εξομοίωση ή την εικονικοποίηση αυτών. Οι εξομοιούμενες διεπαφές προκύπτουν από την προσομοίωση πραγματικών καρτών δικτύου και οι εικονικές μηχανές συμπεριφέρονται προς αυτές ως φυσικές συσκευές. Το πλεονέκτημα που προσφέρουν είναι η συμβατότητα με παλαιότερες εκδόσεις λειτουργικών συστημάτων, οι οποίες πιθανόν να μην διαθέτουν προγράμματα οδήγησης για εικονικοποιημένες διεπαφές δικτύου. Οι διεπαφές που υποστηρίζονται βασίζονται σε δημοφιλή `chipsets` κατασκευαστών όπως οι Intel, Realtek, AMD. Συγκεκριμένα, τα μοντέλα που διατίθενται είναι τα παρακάτω:

- `ne2k_isa/ne2k_pci` (Novell NE2000, 10Mb/s),
- `i82551/i82557b/i82559er` (Intel 8255x, 10/100Mb/s),
- `pcnet` (AMD, 10Mb/s),

⁸ Τα ονόματα αυτά είναι δεσμευμένα για χρήση από το `libvirt`.

- `rtl8139` (Realtek, 10/100Mb/s),
- `e1000` (Intel PRO/1000, 10/100/1000Mb/s).

Οι εικονικοποιημένες διεπαφές, σε αντίθεση με τις εξομοιούμενες, στηρίζονται στην παραεικονικοποίηση (paravirtualization) υλικού και όχι στην προσομοίωση πραγματικών συσκευών. Η διεπαφή που διατίθεται για χρήση στον QEMU/KVM είναι η `virtio`. Το πλεονέκτημα που προσφέρει μια τέτοια διεπαφή είναι κυρίως οι βελτιωμένες επιδόσεις λόγω της απουσίας του overhead που προκύπτει από την εξομοίωση μιας πραγματικής συσκευής. Πέρα από αυτό, η `virtio` υποστηρίζει μεγαλύτερο εύρος ζώνης⁹ από τις εξομοιούμενες κάρτες δικτύου, έως και 10Gb/s.

3.3 Παραδείγματα Χρήσης Δικτύων

Με βάση την παραπάνω ανάλυση, θα αναλυθούν στη συνέχεια μερικές ενδεικτικές περιπτώσεις εικονικοποιημένων δικτύων και διεπαφών που χρησιμοποιούνται στην εφαρμογή που αναπτύχθηκε στο πλαίσιο της εργασίας. Κατ' αρχάς, σε κάθε εικονική μηχανή χρησιμοποιείται μία διεπαφή που προορίζεται αποκλειστικά για την πρόσβαση στην κονσόλα, ώστε να είναι δυνατή η διαχείριση μέσα από ένα απομακρυσμένο τερματικό ακόμα και στην περίπτωση που γίνει κάποιο λάθος στην παραμετροποίηση των υπόλοιπων διεπαφών. Το εικονικοποιημένο δίκτυο που εξυπηρετεί τη διεπαφή διαχείρισης δημιουργείται αυτομάτως τη στιγμή που ορίζεται (on demand) και καταστρέφεται όταν διαγράφεται η εκάστοτε εικονική μηχανή. Τα χαρακτηριστικά του δικτύου αυτού συνοψίζονται στα εξής:

- Παρέχεται απομόνωση μεταξύ όλων των δικτύων διαχείρισης των εικονικών μηχανών, καθώς κάθε διεπαφή διαχείρισης συνδέεται αποκλειστικά σε μία γέφυρα.
- Κάθε εικονική μηχανή έχει μία διεύθυνση IP για απομακρυσμένη πρόσβαση, η οποία βρίσκεται σε ένα απομονωμένο υποδίκτυο.
- Κάθε δίκτυο διαχείρισης έχει μοναδικά στοιχεία (όνομα, UUID, διεύθυνση MAC, προεπιλεγμένη πύλη).
- Κάθε εικονική μηχανή λαμβάνει διεύθυνση IP για τη διεπαφή διαχείρισης μέσω του DHCP server που εξυπηρετεί στη διεύθυνση που έχει οριστεί στο πεδίο `address` του στοιχείου `ip`.
- Κάθε εικονική μηχανή λαμβάνει δυναμικά `hostname`, το οποίο έχει οριστεί μέσω του μηχανισμού αντιστοίχισης αφενός της διεύθυνσης MAC με τη διεύθυνση IP και αφετέρου της διεύθυνσης IP με το `hostname`.

Τα δίκτυα αυτά έχουν μελετηθεί ώστε να ανταποκρίνονται στο πρότυπο του δικτύου Host-Only που προσφέρεται στο Virtualbox, το οποίο παρέχει απ' ευθείας επικοινωνία μεταξύ του host και του εκάστοτε guest, ενώ παράλληλα παρέχει απομόνωση από άλλα εικονικοποιημένα ή φυσικά δίκτυα στα οποία έχει πρόσβαση ο hypervisor.

⁹ Διαπραγματευμένο – Negotiated.

Η παραμετροποίηση (διευθύνσεις MAC και IP) όλων των δικτύων που παράγονται από την εφαρμογή περιγράφεται αναλυτικά στο Παράρτημα Β. Παρακάτω παρατίθεται ενδεικτικά το εικονικοποιημένο δίκτυο που εξυπηρετεί την απομακρυσμένη διαχείριση του προσωπικού υπολογιστή PC1.

```
<network>
  <name>mgmtpc1</name>
  <uuid>6ac5acf9-940b-41fc-87a7-1ae02acccc41</uuid>
  <forward mode="nat"/>
  <bridge stp="on" delay="0" name="virbr125"/>
  <mac mac="52:54:00:41:cc:00"/>
  <dns>
    <host ip="172.21.2.1">
      <hostname>PC1</hostname>
    </host>
  </dns>
  <ip netmask="255.255.255.0" address="172.21.2.200">
    <dhcp>
      <host mac="52:54:00:41:cc:01" ip="172.21.1.1"/>
    </dhcp>
  </ip>
</network>
```

Ένας άλλος τύπος δικτύωσης που χρησιμοποιείται, η χρήση του οποίου είναι αρκετά διαδεδομένη σε προσομοιώσεις δικτύων, είναι η εσωτερική δικτύωση (internal networking¹⁰). Τα εσωτερικά δίκτυα επιτρέπουν την κίνηση μεταξύ των φιλοξενούμενων (οι οποίοι βρίσκονται στο ίδιο υποδίκτυο), χωρίς όμως να έχει οριστεί προώθηση από το δίκτυο αυτό προς το δίκτυο του φιλοξενούντος. Στην πράξη, επομένως, δεν υπάρχει επικοινωνία με κανένα άλλο δίκτυο, εκτός αν υπάρχουν διαδρομές από κάποιον φιλοξενούμενο προς άλλα δίκτυα. Στην περίπτωση αυτή, ο φιλοξενούμενος αυτός, εφόσον δύναται να προωθεί πακέτα από μία διεπαφή του προς μία άλλη, μπορεί να θεωρηθεί ως δρομολογητής για τα δίκτυα που δεν είναι προσβάσιμα από το εσωτερικό δίκτυο προς ένα άλλο δίκτυο. Παρακάτω, παρατίθεται ένα απόσπασμα του ορισμού του εικονικοποιημένου δικτύου LAN1 σε μορφή XML, όπως χρησιμοποιείται στην εφαρμογή.

```
<network>
  <name>LAN1</name>
  <uuid>6ac5acf9-940b-41fc-87a7-1ae02af8cc01</uuid>
  <bridge name='virbr9' stp='on' delay='0'/>
```

¹⁰ Η ορολογία αυτή συναντάται στο Virtualbox.

```

<mac address='52:54:00:01:4d:00' />

<dns>
  <host ip='10.10.1.1'>
    <hostname>R1</hostname>
  </host>
  <host ip='10.10.1.2'>
    <hostname>R2</hostname>
  </host>
  ...
  <host ip='10.10.1.24'>
    <hostname>R24</hostname>
  </host>
</dns>
<ip address='10.10.1.200' netmask='255.255.255.0'>
  <dhcp>
    <range start='10.10.1.49' end='10.10.1.99' />
    <host mac='52:54:00:01:4d:01' ip='10.10.1.1' />
    <host mac='52:54:00:01:4d:02' ip='10.10.1.2' />
    ...
    <host mac='52:54:00:01:4d:18' ip='10.10.1.24' />
  </dhcp>
</ip>
</network>

```

Στη συγκεκριμένη περίπτωση, η εικονικοποιημένη γέφυρα που χρησιμοποιείται από το δίκτυο LAN1 είναι η `virbr9`, ενώ δεν περιέχεται το στοιχείο `forward`, υποδεικνύοντας την απουσία προώθησης και άρα την επικοινωνία αποκλειστικά μεταξύ των φιλοξενούμενων, όπως αναλύθηκε παραπάνω. Επίσης, αναφορικά με τις εγγραφές του DHCP server, φαίνεται η αντιστοίχιση συγκεκριμένων διευθύνσεων MAC σε διευθύνσεις IP, η οποία γίνεται εντός του στοιχείου `dhcp`. Τέλος, η δυναμική ανάθεση `hostname` στους φιλοξενούμενους γίνεται εντός του στοιχείου `dns`, όπου οι εικονικές μηχανές με τις συγκεκριμένες διευθύνσεις IP μπορούν να λάβουν δυναμικά το εκάστοτε `hostname` που αναγράφεται στην παραμετροποίηση. Με όμοιο τρόπο ορίζονται όλα τα εσωτερικά δίκτυα της εφαρμογής, τα οποία περιγράφονται στο Παράρτημα Β.

Όμοια παραμετροποίηση παρουσιάζουν τα δίκτυα NAT (NAT networks¹¹), καθώς έχουν την ίδια μορφή με τα εσωτερικά δίκτυα, με τη διαφορά ότι περιέχουν επίσης την επιλογή προώθησης. Στα

¹¹ Η ορολογία αυτή συναντάται στο Virtualbox.

δίκτυα αυτά, όπως είναι γνωστό, οι φιλοξενούμενοι συνδέονται σε μία κοινή γέφυρα, οπότε υπάρχει επικοινωνία μεταξύ τους. Επιπλέον, η κίνηση προωθείται μέσω NAT στο δίκτυο του φιλοξενούντος, ενώ σημειώνεται ότι η κίνηση από το δίκτυο του φιλοξενούντος προς το δίκτυο των φιλοξενούμενων δεν επιτρέπεται, εκτός αν έχει οριστεί προώθηση θυρών. Παρακάτω παρουσιάζεται η παραμετροποίηση σε μορφή XML ενός τέτοιου δικτύου, όπως χρησιμοποιείται στην εφαρμογή.

```
<network>
  <name>network1</name>
  <uuid>6ac5acf9-940b-41fc-87a7-1ae02af8ccc1</uuid>
  <forward mode='nat' />
  <bridge name='virbr0' stp='on' delay='0' />
  <mac address='52:54:00:c1:4d:00' />
  <dns>
    <host ip='10.0.1.1'>
      <hostname>R1</hostname>
    </host>
    <host ip='10.0.1.2'>
      <hostname>R2</hostname>
    </host>
    ...
    <host ip='10.0.1.24'>
      <hostname>R24</hostname>
    </host>
  </dns>
  <ip address='10.0.1.200' netmask='255.255.255.0'>
    <dhcp>
      <range start='10.0.1.49' end='10.0.1.99' />
      <host mac='52:54:00:c1:4d:01' ip='10.0.1.1' />
      <host mac='52:54:00:c1:4d:02' ip='10.0.1.2' />
      ...
      <host mac='52:54:00:c1:4d:18' ip='10.0.1.24' />
    </dhcp>
  </ip>
</network>
```


Κεφάλαιο 4

Παραγωγή & Παραμετροποίηση της Βασικής Εικόνας του BSDRP

4.1 Δημιουργία της εικόνας BSDRP από τον πηγαίο κώδικα

Στην ενότητα αυτή περιγράφονται τα βήματα για τη ρύθμιση της εικόνας (image) του BSDRP που θα χρησιμοποιηθεί ως δρομολογητής στην εφαρμογή.

Η εικόνα του BSDRP μπορεί να μεταφορτωθεί από τη σελίδα του project (bsdrp.net). Λόγω των περιορισμών στη χρήση του διαχειριστή λογισμικού του FreeBSD (pkg), προτιμήθηκε η μεταγλώττιση του BSDRP από τον πηγαίο κώδικα. Η μεταγλώττιση μπορεί αποκλειστικά σε FreeBSD, επομένως αρχικά θα μεταφορτωθεί μια εικόνα μορφής iso του FreeBSD (χρησιμοποιείται η έκδοση 12.2). Στη συνέχεια, υποθέτοντας ότι η εικόνα βρίσκεται στον κατάλογο των λήψεων, μπορεί να δημιουργηθεί μια εικονική μηχανή από αυτή την εικόνα με την εντολή

```
virt-install \
--name FreeBSD \
--ram 3072 \
--disk path=/var/lib/libvirt/images/FreeBSD.qcow2,size=32 \
--network network=default,model='e1000' \
--vcpus 2 \
--os-type linux \
--os-variant generic \
--console pty,target_type=serial \
--cdrom ~/Downloads/FreeBSD-12.2-RELEASE-amd64-dvd1.iso
```

Η παραπάνω εντολή αναθέτει στην εικονική μηχανή μία μονάδα εικονικού δίσκου (μορφής qcow2) μεγέθους 32GB, δύο πυρήνες CPU και 3GB κύριας μνήμης (RAM), καθώς και μια δικτυακή διεπαφή τύπου NAT που συνδέεται στο εικονικοποιημένο δίκτυο με όνομα default. Το δίκτυο αυτό έχει οριστεί εκ προεπιλογής από το libvirt.

Με την ολοκλήρωση της ανάθεσης του εικονικού δίσκου, η εικονική μηχανή τίθεται αυτόματα σε λειτουργία. Αμέσως μετά εμφανίζεται το παράθυρο της εικονικής μηχανής, και συγκεκριμένα η εγκατάσταση του FreeBSD. Ακολουθώντας τα βήματα, το λειτουργικό σύστημα εγκαθίσταται στον εικονικό δίσκο που έχει δημιουργηθεί προηγουμένως. Μετά την πρώτη εκκίνηση του λειτουργικού συστήματος, η εικονική μηχανή είναι πλέον έτοιμη για χρήση. Αρχικά πραγματοποιείται ενημέρωση του καταλόγου αποθηκών λογισμικού του pkg, με την εντολή

```
pkg update
```

Στη συνέχεια, θα γίνει η εγκατάσταση του εργαλείου git με την εντολή

```
pkg install git
```

Για την εκτέλεση των σεναρίων (scripts) του BSDRP, είναι απαραίτητη η κλωνοποίηση (clone) της αποθήκης (repository) από το GitHub. Μετά την εγκατάσταση του git, το repository κλωνοποιείται με την εντολή

```
git clone https://github.com/occochard/BSDRP
```

Αφού τεθεί ο κατάλογος με όνομα BSDRP ως τρέχων κατάλογος, θα τροποποιηθούν τα αρχεία παραμετροποίησης της εικόνας που θα δημιουργηθεί. Η πρώτη τροποποίηση αφορά στο μέγεθος της εικόνας, όπου αντί 2GB θα επεκταθεί στα 3GB. Για την αλλαγή αυτή, στο αρχείο BSDRP/make.conf τροποποιείται η γραμμή

```
DISK_SIZE=2000
```

Η γραμμή αυτή αντικαθίσταται με

```
DISK_SIZE=3000
```

Η δεύτερη τροποποίηση αφορά στο μέγεθος των διαμερισμάτων (partitions) του δίσκου που προσαρτώνται στους καταλόγους /etc και /tmp/var. Στα διαμερίσματα αυτά αποθηκεύονται προσωρινά τα αρχεία που μεταφορτώνονται από το pkg, όμως εκ προεπιλογής έχουν μικρό μέγεθος (περίπου 10MB και 30MB αντίστοιχα). Το μέγεθος μπορεί να αυξηθεί σε 100MB και 300MB αντίστοιχως, τροποποιώντας το αρχείο BSDRP/BSDRP.nano και αντικαθιστώντας τις γραμμές

```
NANO_RAM_ETCSIZE=20480
```

```
NANO_RAM_TMPVARSIZE=62500
```

Μετά την τροποποίηση, οι γραμμές αυτές έχουν τις τιμές

```
NANO_RAM_ETCSIZE=204800
```

```
NANO_RAM_TMPVARSIZE=625000
```

Σημειώνεται ότι οι αριθμοί αυτοί αντιπροσωπεύουν το μέγεθος του κάθε διαμερίσματος σε τομείς των 512 bytes. Αφού πραγματοποιηθεί η αλλαγή αυτή, μπορεί να δημιουργηθεί η εικόνα του BSDRP. Η εντολή που θα χρησιμοποιηθεί είναι η παρακάτω.

```
./make.sh -w -c serial
```

Η επιλογή `-c serial` χρησιμοποιείται ώστε να δημιουργηθεί εικόνα στην οποία η επικοινωνία μπορεί να γίνεται μέσω της σειριακής κονσόλας. Αυτό είναι ιδιαίτερα χρήσιμο, καθώς το `libvirt` προσφέρει τη δυνατότητα απ' ευθείας επικοινωνίας με εικονικές μηχανές μέσω του εργαλείου `virsh`, και συγκεκριμένα με τη βοήθεια της εντολής `virsh console <domain-name>`. Επίσης, προσδιορίζεται η επιλογή `-w` ώστε να μην εμφανιστούν στο τερματικό τα μηνύματα που σχετίζονται με τη μεταγλώττιση του `world` του FreeBSD (`buildworld`), δηλαδή του συνόλου του λειτουργικού συστήματος, εκτός του πυρήνα. Σημειώνεται ότι εκ προεπιλογής το `script` θα δημιουργήσει μία εικόνα συμβατή με την αρχιτεκτονική `x86-64` (λειτουργικό σύστημα 64 bit). Η διαδικασία της μεταγλώττισης του BSDRP διαρκεί προσεγγιστικά μία ώρα και σαράντα λεπτά, δεδομένων των υπολογιστικών πόρων που έχουν ανατεθεί στη συγκεκριμένη περίπτωση στην εικονική μηχανή. Η κατανομή περισσότερων υπολογιστικών πόρων (πυρήνων CPU, κύριας μνήμης), εφόσον είναι διαθέσιμα, μπορεί να βελτιώσει το χρόνο που απαιτείται για τη δημιουργία της εικόνας.

Μετά τη ολοκλήρωση της διαδικασίας δημιουργίας της εικόνας, προτείνεται η παραμετροποίηση ορισμένων μερών του λειτουργικού συστήματος ώστε να ανταποκρίνεται στις ανάγκες της παρούσας εφαρμογής.

4.2 Παραμετροποίηση της εικόνας BSDRP

Η εικόνα BSDRP που προέκυψε από τον πηγαίο κώδικα του BSDRP project, αφού μεταφερθεί στον host, μπορεί να υποστεί αλλαγές. Κατ' αρχάς, θα δημιουργηθεί μία εικονική μηχανή, η μονάδα δίσκου της οποίας θα είναι η εικόνα BSDRP. Για να επιτευχθεί αυτό, θα πρέπει να μετατραπεί από `img` (RAW) σε `qcow2`. Θέτοντας τον κατάλογο του τερματικού σε αυτόν που περιέχει την εικόνα BSDRP, η μετατροπή μπορεί να γίνει με την εντολή

```
qemu-img convert -f raw -O qcow2 BSDRP-1.98-full-amd64-serial.img BSDRP-1.98-full-amd64-serial.qcow2
```

Η εικόνα που θα προκύψει μπορεί να μεταφερθεί εντός του καταλόγου της εφαρμογής (`labonline`), στον κατάλογο `images` και να μετονομαστεί για λόγους απλότητας σε `BSDRP.qcow2`. Στη συνέχεια, η εικονική μηχανή θα εκκινηθεί με χρήση του εργαλείου `virt-install`, και συγκεκριμένα με την εντολή

```
virt-install \
--name BSDRP \
--disk images/BSDRP.qcow2 \
--import \
--network network=default,model='e1000' \
--vcpus 1 \
--ram 512 \
--os-type linux \
--os-variant freebsd12.0 \
--console pty,target_type=serial
```

Αναλυτικά, ως όνομα της εικονικής μηχανής εισάγεται το BSDRP με την επιλογή `--name`. Ο δίσκος που θα χρησιμοποιηθεί είναι αυτός που έχει δημιουργηθεί από το εργαλείο `qemu-img`. Συγκεκριμένα, η επιλογή `--import` προσδιορίζει την εισαγωγή υπάρχουσας μονάδας δίσκου αντί της δημιουργίας μίας νέας. Για την παραμετροποίηση της εικόνας δεν χρειάζονται περισσότερες από μία δικτυακές διεπαφές. Μέσω της επιλογής `--network` προσδιορίζεται η κατάλληλη διεπαφή. Στη συνέχεια, στην εικονική μηχανή αποδίδονται ένας επεξεργαστής και 512MB κύριας μνήμης, με τη χρήση των επιλογών `--vcpu` και `--ram` αντίστοιχα. Τέλος, επιλέγονται ο τύπος του λειτουργικού συστήματος και η έκδοσή (variant) αυτού, καθώς και ο τύπος της κονσόλας που θα χρησιμοποιηθεί για την επικοινωνία με την εικονική μηχανή. Με τη συγκεκριμένη κονσόλα (PTY – Pseudo TTY), μπορεί να επιτευχθεί επικοινωνία μέσω του εργαλείου `virsh console` από ένα απλό τερματικό.

4.2.1 Τροποποίηση αρχείων του λειτουργικού συστήματος

Με την εκκίνηση του φιλοξενούμενου, εμφανίζεται η προτροπή εισόδου (login prompt). Το BSDRP παρέχει εκ προεπιλογής τη δυνατότητα ταυτοποίησης του υπερχρήστη (root) χωρίς συνθηματικό (password). Αφού πραγματοποιηθεί η είσοδος, μπορεί να αποδοθεί διεύθυνση IP στη διεπαφή `em0` με την εντολή

```
dhclient em0
```

Η απόδοση της IP μπορεί να επαληθευθεί με τον έλεγχο της εξόδου της εντολής

```
ifconfig em0
```

Αμέσως μετά, σειρά έχει η προσάρτηση του καταλόγου / (κατάλογος root) με τη δυνατότητα εγγραφής¹². Αυτό επιτυγχάνεται με την εντολή

```
mount -uo rw /
```

Τα δικαιώματα εγγραφής στον κατάλογο αυτό είναι απαραίτητα για τις αλλαγές που περιγράφονται παρακάτω. Επίσης, η εντολή προσάρτησης (mount) είναι απαραίτητη κάθε φορά που γίνεται εκκίνηση της εικονικής μηχανής, δεδομένου ότι χρειάζεται να τροποποιηθούν αρχεία ή καταλόγοι του λειτουργικού συστήματος.

Για την αποφυγή εκτύπωσης σφαλμάτων tty κάθε 30 δευτερόλεπτα κατά την επικοινωνία μέσω PTY, προτείνεται η τροποποίηση του αρχείου `/etc/ttys`. Στο αρχείο αυτό, αφαιρούνται οι γραμμές

tttyu1	"/usr/libexec/getty 3wire"	vt100	on secure
tttyu2	"/usr/libexec/getty 3wire"	vt100	onifconsole secure
tttyu3	"/usr/libexec/getty 3wire"	vt100	onifconsole secure

Για την επιτάχυνση της διαδικασίας εκκίνησης, προτείνεται η αλλαγή του χρόνου αναμονής στην αρχική οθόνη επιλογής του τρόπου λειτουργίας κατά την εκκίνηση του λειτουργικού συστήματος. Για το σκοπό αυτό, μπορεί να τροποποιηθεί το αρχείο `/boot/loader.conf` και να εγγραφεί σε αυτό η γραμμή

¹² Σημειώνεται ότι εκ προεπιλογής το BSDRP δεν επιτρέπει την εγγραφή στον κατάλογο ρίζας. Τυχόν αλλαγές μπορούν να εγγραφούν στον κατάλογο μόνο με την εντολή `config save (alias wr)` ώστε να είναι διαθέσιμες μετά την επανεκκίνηση ή τον τερματισμό του συστήματος.

```
autoboot_delay="1"
```

Για την αυτόματη απόδοση διευθύνσεων IP από τον εκάστοτε εξυπηρετητή DHCP κατά την εκκίνηση του λειτουργικού συστήματος, προτείνεται η τροποποίηση του αρχείου `/etc/rc.conf`. Στο αρχείο αυτό προστίθενται οι γραμμές

```
ifconfig_em0="DHCP"
ifconfig_em1="DHCP"
ifconfig_em2="DHCP"
ifconfig_em3="DHCP"
ifconfig_vtnet0="DHCP"
```

Επισημαίνεται ότι οι εικονικοποιημένες δικτυακές διεπαφές που αποδίδονται από την εφαρμογή σε κάθε εικονική μηχανή είναι το πολύ τέσσερις τύπου `e1000`¹³ (με όνομα διεπαφής της μορφής `em[x]` εντός του λειτουργικού συστήματος), καθώς και μία για σκοπούς διαχείρισης, τύπου `virtio`¹⁴ (εμφανίζεται ως `vtnet0`).

Στο ίδιο αρχείο, είναι απαραίτητη η διαγραφή της εγγραφής με όνομα `hostname`. Για τους σκοπούς της εφαρμογής, το `hostname` καθορίζεται στο λειτουργικό σύστημα τη στιγμή ανάθεσης της διεύθυνσης IP από τον εκάστοτε εξυπηρετητή DHCP.

Επιπλέον τροποποιήσεις χρειάζονται για τη βελτίωση της χρηστικότητας της κονσόλας που προσφέρεται εντός του φυλλομετρητή κατά τη λειτουργία της εφαρμογής. Συγκεκριμένα, είναι απαραίτητη η αφαίρεση της γραμμής που ξεκινάει με το χαρακτήρα `"` στο αρχείο `/root/.exrc`. Το αρχείο αυτό περιέχει την παραμετροποίηση του επεξεργαστή αρχείων `vi`. Μετά την αλλαγή αυτή, χρειάζεται να προσπελαστεί εκ νέου το αρχείο αυτό από το λειτουργικό σύστημα, με την εντολή

```
source /root/.exrc
```

Επιπλέον, πρέπει να προσαρμοστούν τα δικαιώματα χρήσης του αρχείου αυτού. Συγκεκριμένα, το αρχείο χρειάζεται να έχει δικαιώματα `-rw-----` δηλαδή εγγραφής και ανάγνωσης μόνο από τον ιδιοκτήτη. Επομένως, εκτελείται η εντολή

```
chmod 600 /root/.exrc
```

Στο BSDRP, ιδιαίτερα χρήσιμα για τη χρηστικότητα και την ταχύτητα εκτέλεσης εντολών και σεναρίων είναι τα ψευδώνυμα (`aliases`) που προσφέρονται από το ίδιο το λειτουργικό σύστημα. Πιο αναλυτικά, το αρχείο παραμετροποίησης του φλοιού (`shell`) `CSH`, προσφέρει εκ προεπιλογής ορισμένα χρήσιμα `aliases`, όπως:

- `cli`, αντί της εντολής `ntysh` (για την είσοδο στο περιβάλλον του FRR).
- `reload` ή `reboot`, αντί της εντολής `system reboot` (για την επανεκκίνηση).
- `halt`, αντί της εντολής `system halt` (για τον τερματισμό λειτουργίας).

¹³ Intel PRO/1000

¹⁴ VirtIO – Εικονικοποιημένη δικτυακή διεπαφή.

- `wr`, αντί της εντολής `config save` (για την μόνιμη αποθήκευση των τροποποιημένων αρχείων στο δίσκο).
- `please`, για την εκτέλεση της πιο πρόσφατης εντολής με δικαιώματα υπερχρήστη.

Για να επιτραπεί η χρήση των `aliases` στο περιβάλλον του γραμμής εντολών εντός του φυλλομετρητή, απαιτείται η αντιγραφή τους από το αρχείο `/root/.cshrc` στο αρχείο `/.cshrc` (στον κατάλογο ρίζας). Οι γραμμές προς αντιγραφή παρατίθενται παρακάτω.

```
alias ls      ls -G
alias cli     vtysh
alias include grep
alias reload  'system reboot'
alias halt    'system halt'
alias reboot  'system reboot'
alias wr      'config save'
alias tmux    tmux -u
alias please  'sudo \!-1'

#Ugly patch because birdc is compiled for using /usr/local/var/run
alias birdc   'birdc -s /var/run/birdctl'
```

Για τη χρήση των βελών πλοήγησης κατά τη χρήση εργαλείων όπως το `less`, χρειάζεται η αλλαγή του αρχείου `/.cshrc`. Συγκεκριμένα, στις εγγραφές `setenv` είναι απαραίτητη η προσθήκη της γραμμής

```
setenv      TERM      cons25
```

Επίσης στο αρχείο `/root/.cshrc` χρειάζεται η διαγραφή των γραμμών

```
printf "\033k`hostname -s`\033\\"ws title with hostname
# Load command complete file
source ~/.complete
```

Στο ίδιο αρχείο, προτείνεται η προσθήκη της γραμμής

```
setenv      TERM      xterm
```

Για να είναι δυνατή η χρήση των βελών πλοήγησης εντός του `vi` στο τερματικό εντός του φυλλομετρητή, είναι απαραίτητη η τροποποίηση του αρχείου `/.exrc`. Στο αρχείο αυτό χρειάζεται η προσθήκη της γραμμής

```
set term=cons25
```


4.2.2 Εγκατάσταση του λογισμικού

Πριν την εγκατάσταση νέου λογισμικού, αποτελεί καλή πρακτική η ενημέρωση του καταλόγου του εκάστοτε διαχειριστή λογισμικού. Στη συγκεκριμένη περίπτωση, για την ενημέρωση του καταλόγου αποθηκών λογισμικού του `pkg` καθώς και τη λήψη των τελευταίων εκδόσεων των εγκατεστημένων εφαρμογών, εκτελούνται οι εντολές

```
pkg update
pkg upgrade
```

4.2.2.1 Εγκατάσταση των εξαρτήσεων λογισμικού `pkg`

Τα εργαλεία που θα χρειαστούν στην εικόνα του BSDRP είναι τα `pip` και `git`, τα οποία μπορούν να εγκατασταθούν με την εντολή

```
pkg install py37-pip git
```

Το `pip` είναι υπεύθυνο για την εγκατάσταση και τη διαχείριση των εκδόσεων των εξαρτήσεων Python και θα χρησιμοποιηθεί για την εγκατάσταση των εξαρτήσεων του `pyxterm.js`. Σημειώνεται επίσης ότι η εγκατάσταση του `Git` είναι προαιρετική, καθώς θα χρησιμοποιηθεί μόνο για τη λήψη του πακέτου `pyxterm.js` που θα αναλυθεί παρακάτω. Η λήψη του πακέτου αυτού μπορεί να γίνει εναλλακτικά με κάποιο από τα εργαλεία `wget` και `curl`¹⁵.

Στη συνέχεια, ακολουθεί η εγκατάσταση του `pyxterm.js`.

4.2.2.2 Εγκατάσταση και προσαρμογή του λογισμικού `pyxterm.js`

Το `pyxterm.js` [32] αποτελεί δωρεάν και ανοικτού κώδικα λογισμικό¹⁶, σκοπός του οποίου είναι η λειτουργία ενός ανεξάρτητου PTY τερματικού εντός ενός φυλλομετρητή. Μέσα από το τερματικό αυτό μπορούν να εκτελεστούν εντολές στον υπολογιστή που είναι εγκατεστημένο το λογισμικό. Εν ολίγοις, το `pyxterm.js` αποτελείται από πηγαίο κώδικα Python και Flask (μαζί με τη βιβλιοθήκη `flask-socketio`) στο backend και HTML με στοιχεία από τη βιβλιοθήκη `Xterm.js` της TypeScript στο Frontend.

Η εγκατάστασή του στην εικονική μηχανή BSDRP γίνεται στον κατάλογο `/etc`¹⁷. Αρχικά, για τη λήψη του μέσω `Git` εκτελείται η εντολή

```
git clone https://github.com/cs01/pyxtermjs
```

Εναλλακτικά, μπορεί να γίνει λήψη του πηγαίου κώδικα με μία από τις εντολές

```
wget https://github.com/cs01/pyxtermjs/archive/refs/heads/master.zip
curl -OL https://github.com/cs01/pyxtermjs/archive/refs/heads/master.zip
```

¹⁵ Στην περίπτωση που χρησιμοποιηθούν τα εργαλεία `wget` ή `curl`, πρέπει να γίνει αποσυμπίεση των αρχείων που μεταφορτώνονται.

¹⁶ Διανέμεται μέσω της άδειας MIT.

¹⁷ Σε περίπτωση που εγκατασταθεί σε άλλο κατάλογο, θα πρέπει να τροποποιηθούν αναλόγως οι αναφορές στα αρχεία εντός του repository.

Στο repository περιλαμβάνεται το αρχείο `requirements.txt`, στο οποίο περιέχονται όλες οι εξαρτήσεις (dependencies) της εφαρμογής αυτής. Οι εξαρτήσεις αυτές μπορούν να εγκατασταθούν με την εντολή

```
pip install -r requirements.txt
```

Αφού εγκατασταθούν οι εξαρτήσεις, μπορεί να χρησιμοποιηθεί το `pip` για την εγκατάσταση του `waitress`, του εξυπηρετητή που θα προσφέρει στο φυλλομετρητή την εφαρμογή `pyxterm.js`. Η εγκατάσταση πραγματοποιείται με την εντολή

```
pip install waitress
```

Όπως προαναφέρθηκε, για την εκτέλεση του `pyxterm.js` θα χρησιμοποιηθεί ο εξυπηρετητής `waitress` αντί του εξυπηρετητή που προσφέρει εκ προεπιλογής το `Flask`, καθώς ο τελευταίος ενδείκνυται για χρήση σε περιβάλλοντα ανάπτυξης λογισμικού. Για το λόγο αυτό, προτείνεται να δημιουργηθεί το αρχείο `/etc/pyxtermjs/pyxtermjs/waitress.py` με περιεχόμενα τα παρακάτω.

```
#!/usr/local/bin/python3

from waitress import serve
import app

serve(app.app, host='0.0.0.0', port=80)
```

Συνοπτικά, στο script αυτό καλείται η συνάρτηση `serve()` του `Waitress`, με ορίσματα το όνομα της εφαρμογής, τον `host`¹⁸ και τη θύρα στην οποία θα εξυπηρετεί τα αιτήματα HTTP. Προτείνεται η χρήση της θύρας 80 (προεπιλεγμένη θύρα για την εξυπηρέτηση αιτημάτων HTTP).

Αμέσως μετά, θα γίνει η εγκατάσταση του `rc.d`¹⁹ script που θα εκτελεί τον εξυπηρετητή `waitress` που έχει οριστεί παραπάνω κατά την εκκίνηση του λειτουργικού συστήματος. Για να γίνει αυτό, σε πρώτη φάση θα δημιουργηθεί το αρχείο `/usr/local/etc/rc.d/ pyxtermjs`, στο οποίο θα εγγραφεί το παρακάτω script.

```
#!/bin/sh

# PROVIDE: pyxtermjs
. /etc/rc.subr

name=pyxtermjs
rcvar=pyxtermjs_enable

command="/etc/pyxtermjs/pyxtermjs/waitress_server.py"
```

¹⁸ Επιλέγεται η διεύθυνση `0.0.0.0` ώστε η εφαρμογή να είναι προσβάσιμη από το δίκτυο.

¹⁹ Το `rc.d` είναι ο πλέον ενδεδειγμένος τρόπος διαχείρισης υπηρεσιών που εκτελούνται κατά την εκκίνηση του λειτουργικού συστήματος.

```
command_interpreter="/usr/local/bin/python3.7"

load_rc_config $name

: ${pyxtermjs_enable:=NO}

run_rc_command "$1"
```

Στο προαναφερθέν αρχείο χρειάζονται δικαιώματα εκτέλεσης προκειμένου να μπορεί να εκτελεστεί. Αυτό επιτυγχάνεται με την εντολή

```
chmod +x /usr/local/etc/rc.d/pyxtermjs
```

Στο αρχείο `/etc/rc.conf` χρειάζεται να υπάρχει μία εγγραφή για την ενεργοποίηση του σεναρίου. Επομένως, προστίθεται στο αρχείο η γραμμή

```
pyxtermjs_enable="YES"
```

Επίσης, η υπηρεσία με όνομα `pyxtermjs` ενεργοποιείται (ώστε να εκτελείται στην εκκίνηση) με την εντολή

```
service pyxtermjs enable
```

Η εικονική μηχανή είναι πλέον έτοιμη για χρήση στο περιβάλλον της εφαρμογής και η εικονική μηχανή που είχε δημιουργηθεί μπορεί πλέον να τερματιστεί και να διαγραφεί με τις εντολές (στο τερματικό του host)

```
virsh destroy BSDRP
virsh undefine BSDRP
```

Έχοντας ολοκληρώσει τη διαδικασία τροποποίησης της βασικής εικόνας, μπορεί να παραχθεί ο συνδεδεμένος κλώνος²⁰ (linked clone) της εικόνας αυτής. Για την κλωνοποίηση, αξιοποιείται το εργαλείο `qemu-img` και συγκεκριμένα η λειτουργία `create`, μέσω της οποίας παράγονται κλώνοι εικονικοποιημένων μονάδων δίσκου. Συγκεκριμένα, η παραγωγή της νέας εικόνας, με όνομα `BSDRP_linked.qcow2`, γίνεται με την εντολή

```
qemu-img create -f qcow2 -F qcow2 -b BSDRP.qcow2 BSDRP_linked.qcow2
```

²⁰ Σκοπός του συνδεδεμένου κλώνου είναι η δημιουργία μικρού μεγέθους εικόνων, οι οποίες εξαρτώνται από τη βασική εικόνα. Στις εικόνες αυτές καταγράφονται μόνο τυχόν αλλαγές και αποκλίσεις από τη βασική εικόνα.

Κεφάλαιο 5

Εγκατάσταση και Χρήση της Εφαρμογής

5.1 Απαιτήσεις

Με βάση τη σκοπιμότητα της εφαρμογής, προέκυψαν ορισμένες βασικές απαιτήσεις αναφορικά με τις δυνατότητες, τη λειτουργικότητα και τα στοιχεία του γραφικού περιβάλλοντος του λογισμικού που αναπτύσσεται στο πλαίσιο της εργασίας αυτής. Συγκεκριμένα, για την υλοποίηση της εφαρμογής λαμβάνονται υπ' όψιν οι παρακάτω απαιτήσεις:

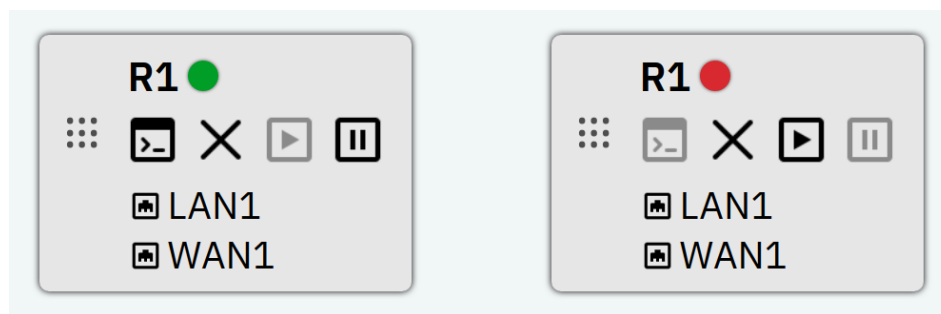
- Ο χρήστης θα έχει πρόσβαση σε ένα γραφικό περιβάλλον μέσω φυλλομετρητή. Στο περιβάλλον αυτό θα μπορεί να κατασκευάζει και να διαχειρίζεται μαζικά εικονικές μηχανές.
- Ο χρήστης θα έχει τη δυνατότητα δημιουργίας δύο τύπων (router ή PC) εικονικών μηχανών.
- Ο χρήστης θα έχει τη δυνατότητα επιλογής των αριθμών των διεπαφών και των τύπων αυτών για κάθε εικονική μηχανή που δημιουργεί.
- Ο χρήστης θα μπορεί να τροποποιήσει τα προεπιλεγμένα ονόματα των εικονικών μηχανών που επιθυμεί να δημιουργήσει.
- Ο χρήστης θα μπορεί να προσαρμόσει τη θέση της κάθε εικονικής μηχανής στο γραφικό περιβάλλον, ώστε να έχει καλύτερη εποπτεία της εκάστοτε τοπολογίας.
- Ο χρήστης θα έχει πρόσβαση στην κάθε εικονική μηχανή ως υπερχρήστης (root), ώστε να μπορεί να την παραμετροποιήσει κατάλληλα.
- Ο χρήστης θα μπορεί να διαγράφει οποιαδήποτε εικονική μηχανή, χωρίς να επηρεάζεται η λειτουργία των άλλων. Επίσης, θα υπάρχει η δυνατότητα ταυτόχρονης διαγραφής όλων των εικονικών μηχανών.
- Μέσω του γραφικού περιβάλλοντος θα υπάρχει ανά πάσα στιγμή εποπτεία της κατάστασης όλων των εικονικών μηχανών, δηλαδή θα φαίνεται ποιες λειτουργούν και ποιες όχι.
- Ο χρήστης θα μπορεί να προβάλλει τα DHCP leases των εικονικών μηχανημάτων.

5.2 Γραφικό Περιβάλλον

Το γραφικό περιβάλλον της εφαρμογής χωρίζεται σε δύο μέρη: το μενού διαχείρισης και το μενού δημιουργίας εικονικών μηχανών. Με βάση τις απαιτήσεις που αναλύθηκαν παραπάνω, παρουσιάζονται οι προδιαγραφές του γραφικού περιβάλλοντος και των συστατικών του στοιχείων.

5.2.1 Μενού Διαχείρισης Εικονικών Μηχανών

Στη γραφική διεπαφή διαχείρισης, εμφανίζονται σε μορφή κάρτας όλα τα εικονικά μηχανήματα, καθώς και οι δικτυακές διεπαφές που έχουν οριστεί για κάθε ένα από αυτά. Για κάθε μηχανήμα υπάρχει ένδειξη κατάστασης (πράσινο αν λειτουργεί και κόκκινο αν δεν λειτουργεί), οι επιλογές εκκίνησης, τερματισμού λειτουργίας, διαγραφής και ανοίγματος του τερματικού σε νέα καρτέλα του περιηγητή²¹.



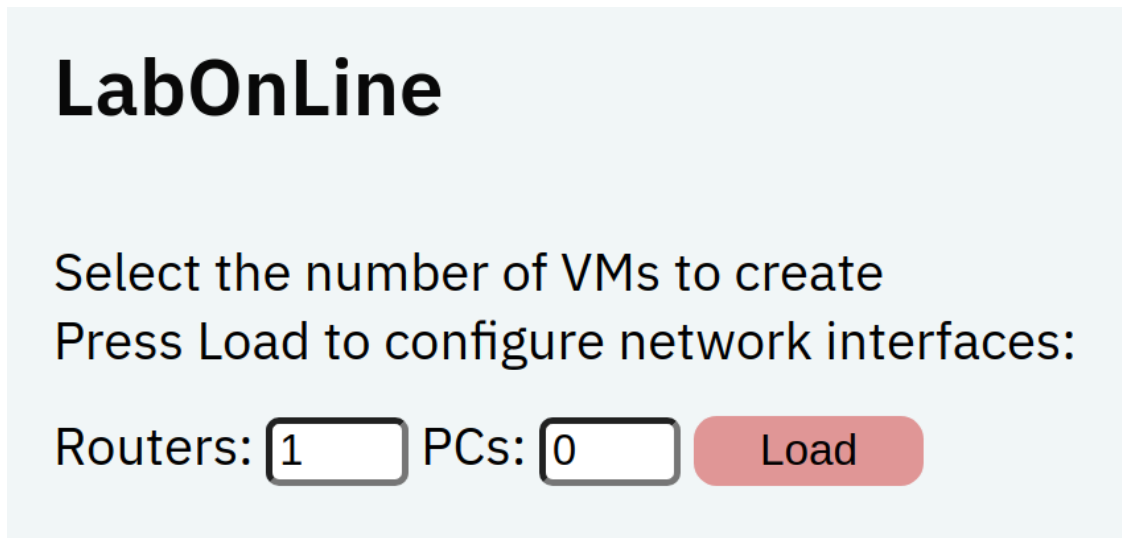
Εικόνα 2 Η μορφή ενός εικονικού μηχανήματος στο γραφικό περιβάλλον, ανάλογα με την κατάστασή του (αριστερά εν λειτουργία, δεξιά απενεργοποιημένο).

Κάθε κάρτα μπορεί να μετακινηθεί σε οποιοδήποτε σημείο πέρα από το προκαθορισμένο. Επίσης, οι εικονικές μηχανές που βρίσκονται στο ίδιο δίκτυο αντιστοιχίζονται σε κάρτες που ενώνονται μεταξύ τους με ένα ευθύγραμμο τμήμα. Το ευθύγραμμο τμήμα προσαρμόζεται αυτόματα ανάλογα με τη θέση που έχει επιλέξει ο χρήστης για κάθε εικονική μηχανή. Τέλος, εφόσον έχουν οριστεί εικονικές μηχανές, εμφανίζονται στο μενού πλήκτρα για την εμφάνιση των μισθώσεων DHCP (DHCP leases), την επαναφορά των θέσεων των καρτών, καθώς και τη διαγραφή όλων των εικονικών μηχανών. Το μενού διαχείρισης εμφανίζεται αν έχουν οριστεί εικονικές μηχανές, επομένως στην αρχική κατάσταση της εφαρμογής αποκρύπτεται από το γραφικό περιβάλλον.

5.2.2 Μενού Δημιουργίας Εικονικών Μηχανών

Στο μενού αυτό, μπορούν να επιλεγθούν ο αριθμός των δρομολογητών και προσωπικών υπολογιστών που θα δημιουργήσει ο χρήστης, ο αριθμός των δικτυακών διεπαφών για κάθε εικονική μηχανή, καθώς και ο τύπος της κάθε διεπαφής. Οι επιλογές που δίνονται για τον τύπο της κάθε εικονικής μηχανής παρουσιάζονται αναλυτικά στο Παράρτημα Β. Οι εικονικές μηχανές δημιουργούνται με το πάτημα του πλήκτρου Submit. Τέλος, το μενού δημιουργίας εικονικών μηχανών είναι πάντα προσβάσιμο πατώντας στον τίτλο της σελίδας (LabOnline).

²¹ Τα πλήκτρα ενεργοποιούνται δυναμικά, ανάλογα με την κατάσταση της εκάστοτε μηχανής (τα πλήκτρα τερματικού και τερματισμού λειτουργίας είναι απενεργοποιημένο όταν η μηχανή δεν λειτουργεί, ενώ το πλήκτρο εκκίνησης είναι απενεργοποιημένο όταν η μηχανή λειτουργεί).



Εικόνα 3 Μενού δημιουργίας εικονικών μηχανών

5.3 Εγκατάσταση

Για την εγκατάσταση της εφαρμογής, αρχικά απαιτείται η παρουσία virtualization extensions (Intel VT-x ή AMD-V), όπως αναλύθηκε παραπάνω. Η ύπαρξη αυτών μπορεί να ελεγχθεί με την εντολή

```
cat /proc/cpuinfo
```

Αν στην έξοδο της εντολής αυτής υπάρχει η συμβολοσειρά “vmx” ή “svm”, τότε ο υπολογιστής υποστηρίζει τις επεκτάσεις Intel VT-x ή AMD-V αντίστοιχα. Όπως προαναφέρθηκε, η παρουσία των εν λόγω επεκτάσεων κρίνεται αναγκαία, καθώς η εφαρμογή αξιοποιεί τις δυνατότητες του KVM.

Ακολουθεί η εγκατάσταση των εξαρτήσεων (dependencies), δηλαδή των εφαρμογών και εργαλείων που χρησιμοποιούνται έμμεσα ή άμεσα από την εφαρμογή. Οι εξαρτήσεις του συστήματος εγκαθίστανται εξ ολοκλήρου από τον διαχειριστή λογισμικού apt του Ubuntu, και είναι οι εξής:

- Το Python pip (package installer), το οποίο διαχειρίζεται τις εξαρτήσεις λογισμικού Python.
- Το virtualenv της Python, το οποίο δημιουργεί τοπικά περιβάλλοντα ανάπτυξης και εκτέλεσης εφαρμογών, εξασφαλίζοντας απομόνωση από το καθολικό (global) περιβάλλον Python, αλλά και μεταξύ των τοπικών περιβαλλόντων.
- Τα libvirt-daemon-system, libvirt-clients, libvirt-dev, virt-viewer και virt-manager, που αφορούν στη διαχείριση των εικονικών μηχανών μέσω του εργαλείου libvirt.
- Τα ebtables και bridge-utils, που αφορούν στη δημιουργία και τη διαχείριση της κίνησης μεταξύ γεφυρών (bridges).
- Το qemu-kvm, το οποίο εγκαθιστά τον KVM στο σύστημα.

Η εγκατάσταση των παραπάνω εξαρτήσεων συστήματος μπορεί να γίνει με την εντολή

```
sudo apt install python3-pip python3-virtualenv libvirt-daemon-system
libvirt-clients libvirt-dev virt-viewer virt-manager ebttables bridge-utils
qemu-kvm
```

Μετά την εγκατάσταση των εξαρτήσεων, σειρά έχει η προσθήκη του χρήστη στις ομάδες (group) `kvm`, `libvirt` και `libvirt-qemu` προκειμένου να διευκολυνθεί η διαχείριση εικονικών μηχανών, εικονικοποιημένων δικτύων και μονάδων δίσκου από τον τρέχοντα χρήστη. Σημειώνεται ότι η ομάδα αυτή δημιουργήθηκε από την εξάρτηση `qemu-kvm`. Με την παρακάτω εντολή ο τρέχων χρήστης προστίθεται στην ομάδα.

```
sudo adduser $USER kvm libvirt libvirt-qemu
```

Αμέσως μετά, απαιτείται επανεκκίνηση του συστήματος.

Στη συνέχεια, αφού οριστεί ως τρέχων κατάλογος ο κατάλογος της εφαρμογής (`labonline`), μπορεί να δημιουργηθεί το εικονικό περιβάλλον της Python με την εντολή

```
python3 -m venv .env
```

Η εντολή αυτή δημιουργεί το εικονικό περιβάλλον με την ονομασία `.env` και ένα νέο ομώνυμο κατάλογο εντός του τρέχοντος καταλόγου. Ο κατάλογος αυτός περιέχει εκτελέσιμα αρχεία της Python, αρχεία παραμετροποίησης και βιβλιοθήκες που αξιοποιούνται από την εφαρμογή. Το περιβάλλον που έχει δημιουργηθεί ενεργοποιείται με την εντολή

```
source .env/bin/activate
```

Το σύστημα είναι πλέον έτοιμο για την εγκατάσταση της εφαρμογής.

```
pip3 install -e .
```

Με την παραπάνω εντολή γίνεται η εγκατάσταση της εφαρμογής, η οποία περιλαμβάνει την λήψη των εξαρτήσεων Python, δηλαδή των εργαλείων και βιβλιοθηκών που χρησιμοποιούνται από την εφαρμογή, μεταξύ των οποίων είναι τα πακέτα `libvirt-python`, `Flask`, `Jinja2`, `Werkzeug`, `lxml`, `simplejson` και `waitress`. Μερικά από αυτά αναλύονται περαιτέρω σε προηγούμενη ενότητα. Η πλήρης λίστα με τις εξαρτήσεις και τις εκδόσεις που χρησιμοποιούνται βρίσκεται στο αρχείο `requirements_dev.txt`. Οι εξαρτήσεις αυτές εγκαθίστανται εντός του καταλόγου `.env` και είναι διαθέσιμες μόνο εντός του τρέχοντος εικονικού περιβάλλοντος Python.

Μετά την εγκατάσταση, ακολουθεί η εκτέλεση του σεναρίου κελύφους (shell script) με όνομα `network_init.sh`. Για την εκτέλεση, χρειάζονται τα ανάλογα δικαιώματα στο χρήστη. Εφόσον αυτά δεν υπάρχουν, μπορούν να αποκτηθούν με την εκτέλεση της εντολής:

```
chmod +x network_init.sh
```

Το αρχείο μπορεί έπειτα να εκτελεστεί με την ακόλουθη εντολή:

```
./network_init.sh
```

Το εν λόγω script δημιουργεί τα εικονικοποιημένα δίκτυα που χρησιμοποιούνται από την εφαρμογή, τα οποία είναι ορισμένα μέσω αρχείων XML που βρίσκονται στον κατάλογο `net_xml`. Κατ' αρχάς, κάθε δίκτυο δημιουργείται, μέσω της εντολής `virsh net-define <network-name>` και μετά ενεργοποιείται, μέσω της εντολής `virsh net-start <network-name>`. Επίσης, διαγράφεται το

δίκτυο `default`, το οποίο είναι ορισμένο εκ προεπιλογής, ώστε να αποφευχθούν τυχόν προβλήματα μεταξύ αυτού και των νέων δικτύων που ορίζονται από το `script`.

Σημειώνεται ότι τα δίκτυα που έχουν οριστεί με το `script` αυτό μπορούν να διαγραφούν με την εκτέλεση του σεναρίου `network_rm.sh`.

Με τη διαδικασία αυτή το σύστημα είναι πλέον έτοιμο για την εκτέλεση της εφαρμογής.

Η εφαρμογή υποστηρίζεται από το `Waitress`, έναν εξυπηρετητή ιστού της `Python` που συνιστάται για χρήση σε περιβάλλον παραγωγής (`production`). Ο εξυπηρετητής εκτελείται μέσω της παρακάτω εντολής και εξυπηρετεί αιτήματα `HTTP` στη θύρα `5000` εκ προεπιλογής.

```
python3 waitress_server.py
```

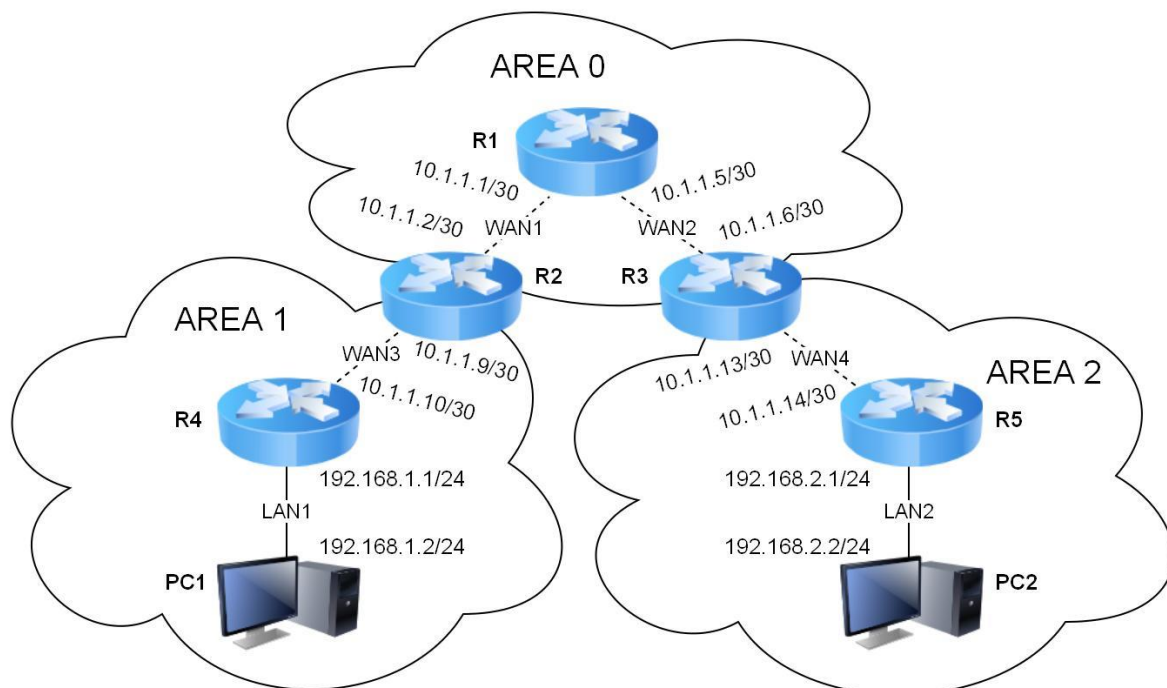
Ο χρήστης μπορεί να έχει πρόσβαση στην εφαρμογή πληκτρολογώντας στον περιηγητή τη διεύθυνση

```
localhost:5000
```

Η εφαρμογή είναι προσβάσιμη επίσης από απομακρυσμένα μηχανήματα, αφού είναι ρυθμισμένη ώστε να αποκρίνεται σε αιτήματα από περιηγητές που φιλοξενούνται εντός του εκάστοτε τοπικού δικτύου.

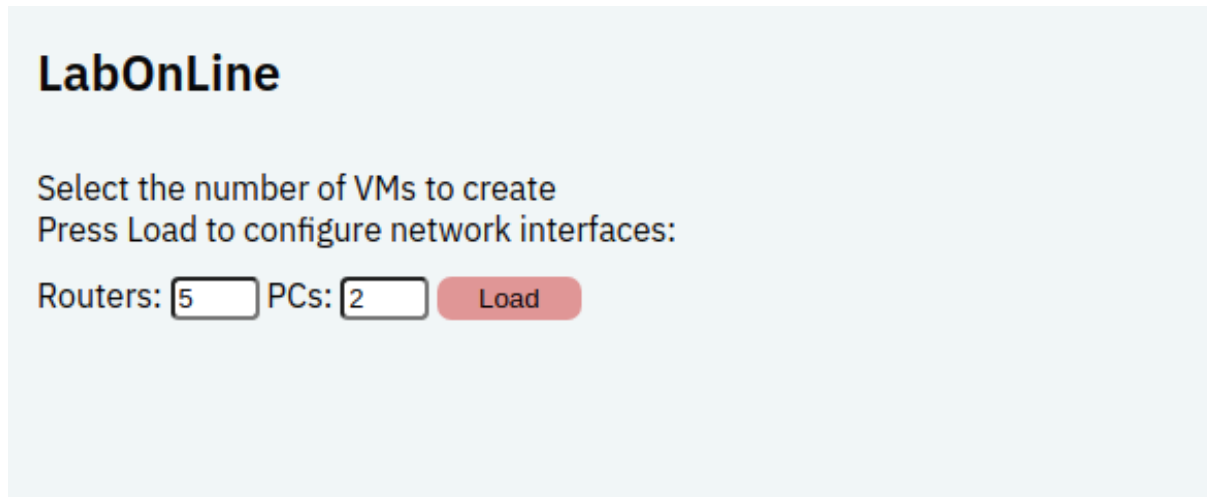
5.4 Σενάριο Χρήσης OSPF

Ένα σενάριο στο οποίο μπορεί να ανταποκριθεί η εφαρμογή είναι αυτό μίας εργαστηριακής άσκησης. Ενδεικτικά, θα παρουσιαστεί μία τοπολογία για τη δοκιμή του πρωτοκόλλου `OSPF` σε ένα δίκτυο με πολλές περιοχές `OSPF`. Στο σενάριο αυτό θα χρησιμοποιηθούν πέντε δρομολογητές (`R1`, `R2`, `R3`, `R4`, `R5`) και δύο προσωπικοί υπολογιστές (`PC1`, `PC2`). Στην Εικόνα 4 παρουσιάζεται η τοπολογία που θα υλοποιηθεί με την εφαρμογή.



Εικόνα 4 Τοπολογία OSPF

Σε πρώτη φάση, θα οριστούν οι επτά εικονικές μηχανές και οι δικτυακές διεπαφές τους, όπως φαίνονται στο διάγραμμα της τοπολογίας. Στην Εικόνα 5 επιλέγεται ο αριθμός των εικονικών μηχανών που θα δημιουργηθούν.



The screenshot shows the 'LabOnline' interface. It has a light blue background. At the top, the text 'LabOnline' is displayed in a large, bold, black font. Below this, the instructions 'Select the number of VMs to create' and 'Press Load to configure network interfaces:' are shown in a smaller black font. At the bottom, there are two input fields: 'Routers:' followed by a text box containing the number '5', and 'PCs:' followed by a text box containing the number '2'. To the right of these fields is a red button with the white text 'Load'.

Εικόνα 5 Επιλογή αριθμού δρομολογητών & προσωπικών υπολογιστών

Έπειτα, καθορίζονται το όνομα της κάθε εικονικής μηχανής²² και ο αριθμός των διεπαφών, όπως φαίνεται στην Εικόνα 6. Στη συγκεκριμένη περίπτωση, θα χρησιμοποιηθούν τα προεπιλεγμένα ονόματα για όλες τις εικονικές μηχανές, καθώς ανταποκρίνονται στην τοπολογία που πρόκειται να κατασκευαστεί.

²² Ως προεπιλεγμένα ονόματα για την κάθε εικονική μηχανή ορίζονται τα $R[x]$ και $PC[x]$ για δρομολογητές και προσωπικούς υπολογιστές αντιστοίχως. Ο εκάστοτε αριθμός x που εμφανίζεται είναι ο μικρότερος διαθέσιμος για κάθε τύπο εικονικής μηχανής.

LabOnLine

Select the number of VMs to create
Press Load to configure network interfaces:

Routers: 5 PCs: 2

R1	Number of interfaces: 2
R2	Number of interfaces: 2
R3	Number of interfaces: 2
R4	Number of interfaces: 2
R5	Number of interfaces: 2
PC1	Number of interfaces: 1
PC2	Number of interfaces: 1

Load interface selection

Εικόνα 6 Επιλογή ονόματος & αριθμού διεπαφών για την κάθε εικονική μηχανή

Στο επόμενο στάδιο, προσδιορίζεται ο τύπος διασύνδεσης της κάθε διεπαφής. Στο συγκεκριμένο σενάριο χρησιμοποιούνται τα εσωτερικά (internal) δίκτυα LAN1, LAN2 και WAN1, WAN2, WAN3, WAN4. Η παραμετροποίηση παρουσιάζεται συνολικά στην Εικόνα 7.

LabOnLine

Select the number of VMs to create
Press Load to configure network interfaces:

Routers: 5 PCs: 2

R1
Number of interfaces: 2
Interface 1: WAN1
Interface 2: WAN2

R2
Number of interfaces: 2
Interface 1: WAN1
Interface 2: WAN3

R3
Number of interfaces: 2
Interface 1: WAN2
Interface 2: WAN4

R4
Number of interfaces: 2
Interface 1: WAN3
Interface 2: LAN1

R5
Number of interfaces: 2
Interface 1: WAN4
Interface 2: LAN2

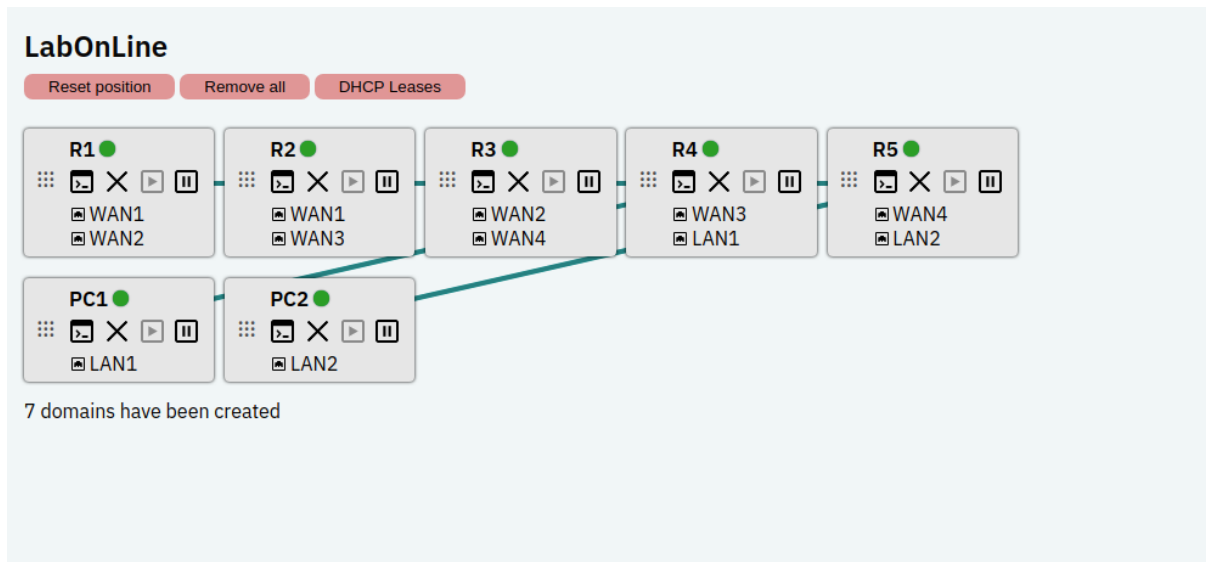
PC1
Number of interfaces: 1
Interface 1: LAN1

PC2
Number of interfaces: 1
Interface 1: LAN2

Submit

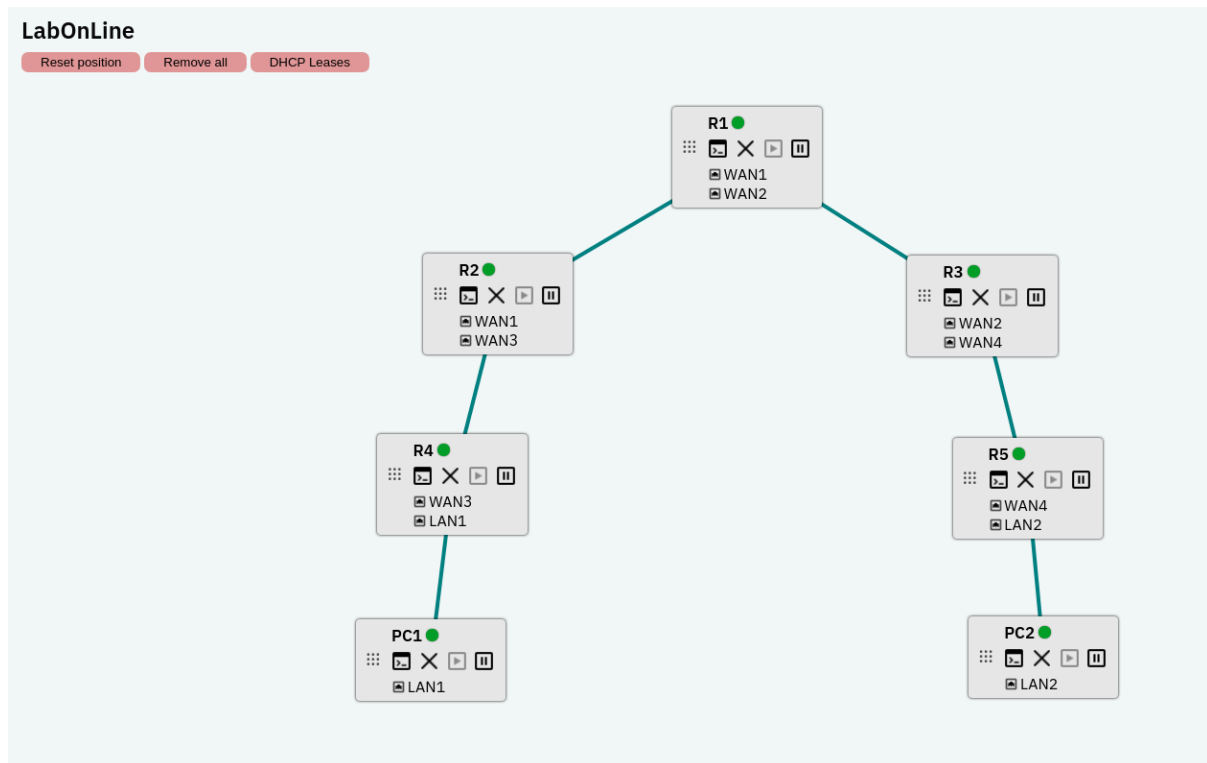
Εικόνα 7 Παραμετροποίηση των διεπαφών των εικονικών μηχανών

Με το πάτημα του πλήκτρου Submit, δημιουργούνται οι εικονικές μηχανές με τις διεπαφές που καθορίστηκαν και κάθε μία παρουσιάζεται με τη μορφή κάρτας, όπως φαίνεται στην Εικόνα 8.



Εικόνα 8 Σελίδα διαχείρισης εικονικών μηχανών

Στη σελίδα διαχείρισης μπορούν να γίνουν όλες οι ενέργειες που έχουν περιγραφεί στην παράγραφο 5.2, όπως άνοιγμα της κονσόλας, εκκίνηση, τερματισμός λειτουργίας, διαγραφή εικονικών μηχανών, μετακίνηση των καρτών μέσα στη σελίδα του φυλλομετρητή και προβολή των DHCP leases. Στην αρχική κατάσταση της σελίδας διαχείρισης, η διασύνδεση μεταξύ των εικονικών μηχανών είναι λιγότερο εμφανής, όμως για τη βελτίωση της εμπειρίας χρήσης, η θέση των καρτών μπορεί να προσαρμοστεί κατάλληλα ώστε να ανταποκρίνεται στο διάγραμμα. Με τον τρόπο αυτό, καθίσταται πιο εύκολη η παραμετροποίηση της κάθε εικονικής μηχανής και η εποπτεία της τοπολογίας συνολικά. Η δικτυακή τοπολογία μετά την προσαρμογή της θέσης των καρτών φαίνεται στην Εικόνα 9. Η διασύνδεση μεταξύ των εικονικών μηχανών είναι πλέον πιο εμφανής και έτσι είναι πιο εύκολη η διαχείρισή τους.



Εικόνα 9 Προσαρμογή των καρτών

Αναφορικά με την τοπολογία που έχει επιλεγεί, οι διευθύνσεις IP μπορούν να τροποποιηθούν με τις κατάλληλες εντολές εντός του περιβάλλοντος του FRR σε κάθε εικονική μηχανή. Στον παρακάτω πίνακα συνοψίζονται οι διευθύνσεις IP όλων των εικονικών μηχανών του σεναρίου.

	R1	R2	R3	R4	R5	PC1	PC2
WAN1	10.1.1.1/30	10.1.1.2/30	-	-	-	-	-
WAN2	10.1.1.5/30	-	10.1.1.6/30	-	-	-	-
WAN3	-	10.1.1.9/30	-	10.1.1.10/30	-	-	-
WAN4	-	-	10.1.1.13/30	-	10.1.1.14/30	-	-
LAN1	-	-	-	192.168.1.1/24	-	192.168.1.2/24	-
LAN2	-	-	-	-	192.168.2.1/24	-	192.168.2.2/24
Loopback	172.22.22.1/32	172.22.22.2/32	172.22.22.3/32	172.22.22.4/32	172.22.22.5/32	-	-

Πίνακας 1 Σύνοψη διευθύνσεων IP των εικονικών μηχανών για το σενάριο OSPF

Ενδεικτικά, παρουσιάζεται η συνολική παραμετροποίηση του δρομολογητή R1 στο FRR, όπως φαίνεται στην Εικόνα 10, με τη βοήθεια της εντολής

```
sh run
```

```
LabOnLine status: connected
root@r1:~ # cli

Hello, this is FRRouting (version 7.5.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1# sh run
Building configuration...

Current configuration:
!
frr version 7.5.1
frr defaults traditional
hostname r1
!
interface em0
 ip address 10.1.1.1/30
!
interface em1
 ip address 10.1.1.5/30
!
interface lo0
 ip address 172.22.22.1/32
!
router ospf
 network 10.1.1.0/30 area 0
 network 10.1.1.4/30 area 0
!
line vty
!
end
r1#
```

Εικόνα 10 Παραμετροποίηση του δρομολογητή R1

Στη σύνοψη αυτή παρουσιάζονται οι διευθύνσεις IP που έχουν οριστεί για τις δύο διεπαφές (WAN1 και WAN2) καθώς και η διεύθυνση loopback της εικονικής μηχανής. Επίσης, τα δύο αυτά δίκτυα (10.1.1.0/30 και 10.1.1.4/30 αντιστοίχως) έχουν τεθεί στην περιοχή 0 του OSPF. Αντίστοιχη διαδικασία ακολουθείται για όλους τους δρομολογητές.

Αναφορικά με τα PCs, η παραμετροποίηση είναι πιο απλή, καθώς απαιτείται μόνο ο ορισμός της διεύθυνσης IP και ο ορισμός προεπιλεγμένης πύλης. Υπενθυμίζεται ότι στη συγκεκριμένη περίπτωση υπάρχει ήδη μία προεπιλεγμένη πύλη, επομένως θα πρέπει πρώτα να διαγραφεί με την εντολή

```
route delete default
```


Στη συνέχεια, μπορεί να οριστεί η νέα πύλη. Για παράδειγμα, στο PC1 θα οριστεί η πύλη με την εντολή

```
route add default 192.168.1.1
```

Στην Εικόνα 11 φαίνονται οι border routers αφού έχουν παραμετροποιηθεί επιτυχώς όλοι οι δρομολογητές, ενώ στην Εικόνα 12 παρουσιάζονται οι γείτονες του R1, οι οποίοι είναι οι R2 και R3.

```
LabOnLine status: connected
root@r1:~ # cli

Hello, this is FRRouting (version 7.5.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1# sh ip ospf border-routers
===== OSPF router routing table =====
R    172.22.22.2          [10] area: 0.0.0.0, ABR
                        via 10.1.1.2, em0
R    172.22.22.3          [10] area: 0.0.0.0, ABR
                        via 10.1.1.6, em1

r1#
```

Εικόνα 11 Προβολή border routers

```
LabOnLine status: connected
root@r1:~ # cli

Hello, this is FRRouting (version 7.5.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1# sh ip ospf neighbor

Neighbor ID    Pri State           Dead Time Address      Interface      RXmtL RqstL DBsmL
172.22.22.2    1 Full/Backup      36.403s 10.1.1.2      em0:10.1.1.1   0      0      0
172.22.22.3    1 Full/Backup      35.720s 10.1.1.6      em1:10.1.1.5   0      0      0

r1#
```

Εικόνα 12 Γείτονες του δρομολογητή R1

Τέλος, η επιτυχής λειτουργία του δικτύου μπορεί να επαληθευθεί με το εργαλείο traceroute. Στην Εικόνα 13 παρατίθεται το αποτέλεσμα του traceroute από το PC1 (192.168.1.2) στο PC2 (192.168.2.2). Τα βήματα μέχρι τον τελικό προορισμό είναι τα εξής:

- R4 (192.168.1.1)
- R2 (10.1.1.9)
- R1 (10.1.1.1)

- R3 (10.1.1.6)
- R5 (10.1.1.14)
- PC2 (192.168.2.2)

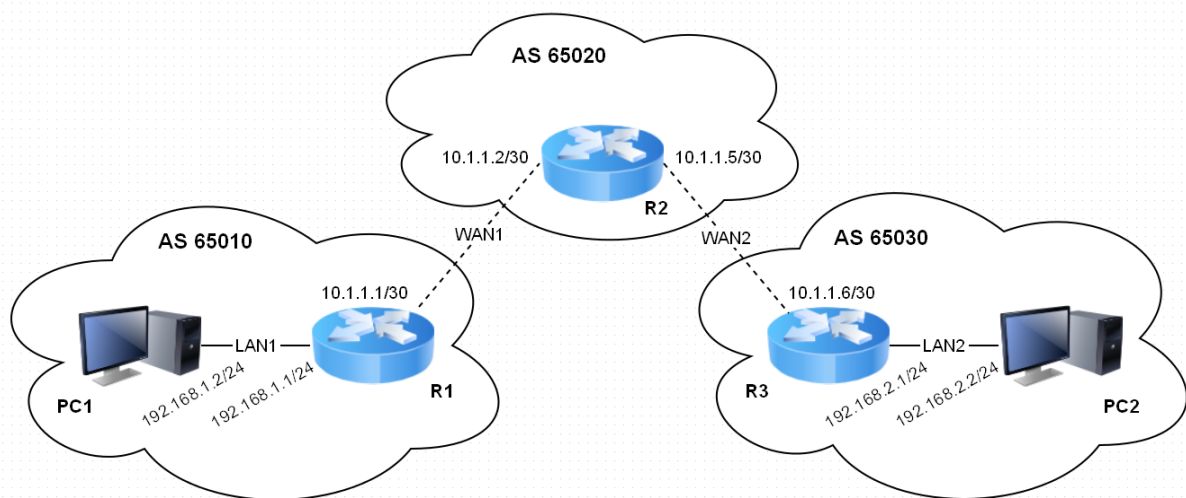
```

LabOnLine status: connected
root@pc1:~ # traceroute 192.168.2.2
traceroute to 192.168.2.2 (192.168.2.2), 64 hops max, 40 byte packets
 1 192.168.1.1 (192.168.1.1) 0.395 ms 0.738 ms 0.682 ms
 2 10.1.1.9 (10.1.1.9) 1.329 ms 1.366 ms 1.258 ms
 3 10.1.1.1 (10.1.1.1) 1.873 ms 2.262 ms 2.032 ms
 4 10.1.1.6 (10.1.1.6) 2.623 ms 2.837 ms 2.544 ms
 5 10.1.1.14 (10.1.1.14) 2.893 ms 3.656 ms 2.288 ms
 6 192.168.2.2 (192.168.2.2) 2.916 ms 5.092 ms 3.050 ms
root@pc1:~ #
    
```

Εικόνα 13 Traceroute από το PC1 στο PC2

5.5 Σενάριο Χρήσης BGP

Παρόμοιο σενάριο μπορεί να διαρθρωθεί με τη χρήση του πρωτοκόλλου BGP (Border Gateway Protocol). Στην Εικόνα 14 παρουσιάζεται μία τοπολογία με τρεις δρομολογητές (R1, R2, R3) και δύο προσωπικούς υπολογιστές (PC1, PC2). Συγκεκριμένα, η δικτύωση στη συγκεκριμένη περίπτωση αφορά στη δρομολόγηση μεταξύ διαφορετικών διαχειριστικών περιοχών (Autonomous Systems). Για το σκοπό αυτό χρησιμοποιείται το πρωτόκολλο BGP και ειδικότερα το eBGP (external BGP).



Εικόνα 14 Τοπολογία BGP

Κατ' αρχάς, ορίζονται μέσω της εφαρμογής οι συσκευές του σεναρίου με τις κατάλληλες δικτυακές διεπαφές. Στην περίπτωση αυτή, θα χρησιμοποιηθούν τα προκαθορισμένα ονόματα εικονικών μηχανών, όπως φαίνεται στην Εικόνα 15.

LabOnLine

Select the number of VMs to create
Press Load to configure network interfaces:

Routers: 3 PCs: 2

R1
Number of interfaces: 2
Interface 1:
Interface 2:

R2
Number of interfaces: 2
Interface 1:
Interface 2:

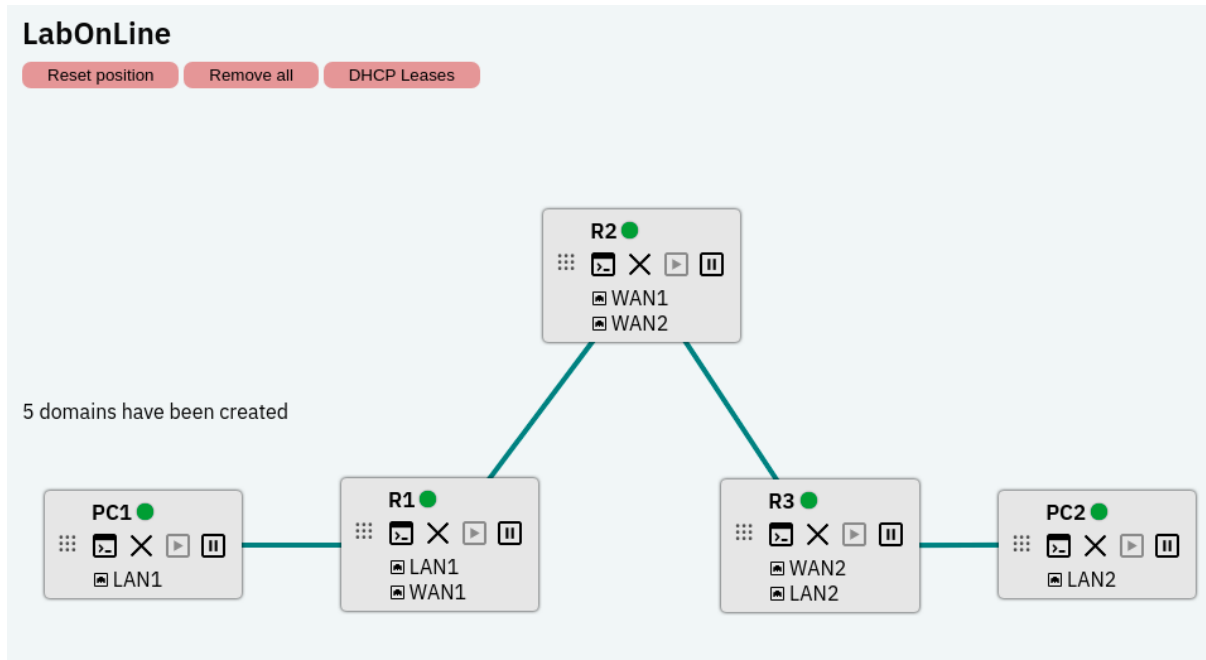
R3
Number of interfaces: 2
Interface 1:
Interface 2:

PC1
Number of interfaces: 1
Interface 1:

PC2
Number of interfaces: 1
Interface 1:

Εικόνα 15 Παραμετροποίηση των διεπαφών των εικονικών μηχανών

Αφού κατασκευαστούν και ενεργοποιηθούν όλες οι εικονικές μηχανές, παρουσιάζονται στη σελίδα του φυλλομετρητή. Αφού προσαρμοστεί η θέση τους ώστε να ανταποκρίνεται στις θέσεις των συσκευών που φαίνονται στην τοπολογία, μπορεί να γίνει η παραμετροποίηση κάθε μίας από αυτές.



Εικόνα 16 Παρουσίαση τοπολογίας στην εφαρμογή

Παρακάτω παρατίθενται οι παραμετροποιήσεις των δρομολογητών (R1, R2, R3). Ειδικότερα, σε κάθε δρομολογητή φαίνονται τα αποτελέσματα των εντολών

```
do show running-config
do show ip bgp
do show ip bgp summary
```

Η πρώτη εντολή παρουσιάζει συνοπτικά τη συνολική παραμετροποίηση που έχει γίνει σε κάθε συσκευή, η δεύτερη εντολή συνοψίζει τα προθέματα δικτύων που έχει μάθει ο κάθε δρομολογητής μέσω του πρωτοκόλλου, καθώς και τις διαδρομές προς κάθε ένα από αυτά, και η τρίτη εντολή παραθέτει πληροφορίες για τους δρομολογητές – γείτονες, όπως το AS στο οποίο βρίσκονται, πόσα προθέματα έχουν αναγγελθεί, κ.ά..

LabOnLine status: connected

```
Current configuration:
!
frr version 7.5.1
frr defaults traditional
hostname r1
!
interface em0
 ip address 192.168.1.1/24
!
interface em1
 ip address 10.1.1.1/30
!
router bgp 65010
 no bgp ebgp-requires-policy
 neighbor 10.1.1.2 remote-as 65020
!
 address-family ipv4 unicast
  network 192.168.1.0/24
 exit-address-family
!
line vty
!
end
r1(config-router)# do sh ip bgp
BGP table version is 2, local router ID is 192.168.1.1, vrf id 0
Default local pref 100, local AS 65010
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
                i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network        Next Hop           Metric LocPrf Weight Path
*> 192.168.1.0/24  0.0.0.0                0         32768 i
*> 192.168.2.0/24  10.1.1.2                0         65020 65030 i

Displayed 2 routes and 2 total paths
r1(config-router)# do sh ip bgp sum

IPv4 Unicast Summary:
BGP router identifier 192.168.1.1, local AS number 65010 vrf-id 0
BGP table version 2
RIB entries 3, using 576 bytes of memory
Peers 1, using 14 KiB of memory

Neighbor      V      AS   MsgRcvd   MsgSent   TblVer   InQ  OutQ  Up/Down State/PfxRcd  PfxSnt
10.1.1.2      4      65020     12       12        0     0    0 00:07:01          1         2

Total number of neighbors 1
r1(config-router)#
```

Εικόνα 17 Παραμετροποίηση του δρομολογητή R1

```

LabOnLine status: connected
Building configuration...

Current configuration:
!
frr version 7.5.1
frr defaults traditional
hostname r2
!
interface em0
 ip address 10.1.1.2/30
!
interface em1
 ip address 10.1.1.5/30
!
router bgp 65020
 no bgp ebgp-requires-policy
 neighbor 10.1.1.1 remote-as 65010
 neighbor 10.1.1.6 remote-as 65030
!
line vty
!
end
r2(config-router)# do sh ip bgp
BGP table version is 2, local router ID is 172.22.2.1, vrf id 0
Default local pref 100, local AS 65020
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop              Metric LocPrf Weight Path
*> 192.168.1.0/24    10.1.1.1                  0             0 65010 i
*> 192.168.2.0/24    10.1.1.6                  0             0 65030 i

Displayed 2 routes and 2 total paths
r2(config-router)# do sh ip bgp sum

IPv4 Unicast Summary:
BGP router identifier 172.22.2.1, local AS number 65020 vrf-id 0
BGP table version 2
RIB entries 3, using 576 bytes of memory
Peers 2, using 29 KiB of memory

Neighbor      V      AS   MsgRcvd   MsgSent   TblVer   InQ  OutQ  Up/Down  State/PfxRcd  PfxSnt
10.1.1.1      4      65010      10       10         0     0    0 00:05:13      1           2
10.1.1.6      4      65030       7        8         0     0    0 00:02:26      1           2

Total number of neighbors 2
r2(config-router)#

```

Εικόνα 18 Παραμετροποίηση του δρομολογητή R2

```

LabOnLine status: connected
Current configuration:
!
frr version 7.5.1
frr defaults traditional
hostname r3
!
interface em0
 ip address 10.1.1.6/30
!
interface em1
 ip address 192.168.2.1/24
!
router bgp 65030
 no bgp ebgp-requires-policy
 neighbor 10.1.1.5 remote-as 65020
!
 address-family ipv4 unicast
  network 192.168.2.0/24
 exit-address-family
!
line vty
!
end
r3(config-router)# do sh ip bgp
BGP table version is 2, local router ID is 192.168.2.1, vrf id 0
Default local pref 100, local AS 65030
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop           Metric LocPrf Weight Path
*> 192.168.1.0/24    10.1.1.5              0 65020 65010 i
*> 192.168.2.0/24    0.0.0.0                0      32768 i

Displayed 2 routes and 2 total paths
r3(config-router)# do sh ip bgp sum

IPv4 Unicast Summary:
BGP router identifier 192.168.2.1, local AS number 65030 vrf-id 0
BGP table version 2
RIB entries 3, using 576 bytes of memory
Peers 1, using 14 KiB of memory

Neighbor      V      AS    MsgRcvd   MsgSent   TblVer   InQ  OutQ  Up/Down State/PfxRcd  PfxSnt
10.1.1.5      4      65020       8         8         0     0    0 00:03:49      1         2

Total number of neighbors 1
r3(config-router)#

```

Εικόνα 19 Παραμετροποίηση του δρομολογητή R3

Με τη σωστή παραμετροποίηση όλων των δρομολογητών, αλλά και των προσωπικών υπολογιστών που συμμετέχουν στην τοπολογία, αναμένεται ότι αφενός όλοι οι δρομολογητές έχουν γνώση για τα δύο προθέματα δικτύων που έχουν αναγγελθεί από τους R1 και R3, και αφετέρου ότι οι PC1 και PC2 μπορούν να επικοινωνήσουν μεταξύ τους. Το τελευταίο επιβεβαιώνεται με την εντολή traceroute που φαίνεται στην Εικόνα 20.

```

LabOnLine status: connected
root@pc1:~ # traceroute 192.168.2.2
traceroute to 192.168.2.2 (192.168.2.2), 64 hops max, 40 byte packets
 1  192.168.1.1 (192.168.1.1)  2.035 ms  0.993 ms  1.801 ms
 2  10.1.1.2 (10.1.1.2)  2.079 ms  2.104 ms  2.182 ms
 3  10.1.1.6 (10.1.1.6)  3.074 ms  3.106 ms  2.907 ms
 4  192.168.2.2 (192.168.2.2)  4.900 ms  3.647 ms  3.065 ms
root@pc1:~ #

```

Εικόνα 20 Η εντολή traceroute από το PC1 στο PC2



WAN1	10.1.1.1/30	10.1.1.2/30	-	-	-
WAN2	-	10.1.1.5/30	10.1.1.6/30	-	-
LAN1	192.168.1.1/24	-	-	192.168.1.2/24	-
LAN2	-	-	192.168.2.1/24	-	192.168.2.2/24

Πίνακας 2 Σύνοψη διευθύνσεων IP των εικονικών μηχανών για το σενάριο BGP

Κεφάλαιο 6

Συμπεράσματα και Προοπτικές

6.1 Συμπεράσματα

Με την ολοκλήρωση και δοκιμή της εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, προέκυψαν ορισμένα συμπεράσματα αναφορικά με τη χρήση προγραμματιστικών εργαλείων εικονικοποίησης για την εξομοίωση δικτυακών τοπολογιών. Ένα από τα σημαντικά πλεονεκτήματα που προκύπτουν με τη χρήση της εφαρμογής είναι αφενός η γρήγορη κατασκευή εικονικών μηχανών με πολλαπλές δικτυακές διεπαφές και αφετέρου η εποπτεία που προσφέρεται μέσω του γραφικού περιβάλλοντος στην εκάστοτε τοπολογία. Καθίσταται σαφές ότι η γραφική διεπαφή της εφαρμογής έχει αναπτυχθεί με μέριμνα για την εξοικονόμηση χρόνου, σε σύγκριση με την παραδοσιακή προσέγγιση της δημιουργίας δικτυακών τοπολογιών με χρήση λογισμικών εικονικοποίησης όπως το Virtualbox.

Σημαντικό ρόλο στη βελτίωση της εμπειρίας χρήστη διαδραματίζει η δυνατότητα πρόσβασης στην κονσόλα των εικονικών μηχανών ώστε να γίνονται οι απαραίτητες αλλαγές στην παραμετροποίηση. Επίσης, όπως προαναφέρθηκε, υπογραμμίζεται η σημασία της εποπτείας που παρέχεται στο χρήστη μέσω της προσαρμογής της θέσης των εικονικών μηχανών στο γραφικό περιβάλλον ώστε να υπάρχει αντιστοιχία με διαγράμματα εκφωνήσεων ενός εργαστηριακού μαθήματος.

Τα παραπάνω έχουν ιδιαίτερη εκπαιδευτική σημασία, αφού διευκολύνουν τη διερεύνηση σεναρίων δικτύων ποικίλων μεγεθών και δυνατοτήτων, προάγοντας τον πειραματισμό σε ένα γραφικό περιβάλλον και αίροντας τους περιορισμούς που συνδέονται με τη χρήση υπαρχόντων εργαλείων.

6.2 Προοπτικές

Η εφαρμογή που αναπτύχθηκε είναι λειτουργική σε πολλά σενάρια χρήσης, όπως αυτό που αναλύθηκε στην προηγούμενη ενότητα, αλλά και πολλά άλλα που περιλαμβάνουν διαφορετικά πρωτόκολλα δικτύωσης. Παρ' όλα αυτά, παρακάτω προτείνονται ορισμένες μελλοντικές επεκτάσεις της εφαρμογής που στοχεύουν στην προσθήκη περαιτέρω λειτουργικότητας. Στον κλάδο της τεχνολογίας λογισμικού, οι προοπτικές μίας εφαρμογής επηρεάζονται από τη συντηρησιμότητα του πηγαίου κώδικα (maintainability), το τεχνικό χρέος (technical debt), την ποιότητα του κώδικα και άλλες παραμέτρους που συμβάλλουν στο πόσο εύκολο ή δύσκολο είναι να επεκταθεί στο μέλλον ένα έργο λογισμικού. Μερικές επεκτάσεις που μπορούν να γίνουν στην εφαρμογή είναι οι εξής:

- Παραλληλοποίηση της εφαρμογής ώστε να μπορεί να εξυπηρετήσει πολλαπλούς χρήστες σε διαφορετικές συνεδρίες (sessions). Στόχος είναι η ταυτόχρονη πρόσβαση πολλών πελατών στον εξυπηρετητή που τρέχει η εφαρμογή. Κάθε χρήστης μπορεί να φτιάχνει δικτυακές τοπολογίες χρησιμοποιώντας την εφαρμογή, χωρίς να βλέπει τις τοπολογίες των άλλων χρηστών. Η επέκταση αυτή προϋποθέτει την εγκατάσταση του λογισμικού σε έναν server που μπορεί να υποστηρίξει μεγάλο αριθμό αιτημάτων και εικονικών μηχανών. Η εφαρμογή μπορεί επίσης να διατεθεί για πρόσβαση τόσο εντός ενός τοπικού δικτύου εργαστηρίου, όσο και μέσω internet σαν ιδιωτικό cloud, δεδομένου ότι υπάρχουν οι κατάλληλες υποδομές διαπίστευσης χρηστών.
- Εισαγωγή διαφορετικών βασικών εικόνων λειτουργικών συστημάτων από το φυλλομετρητή. Στην υπάρχουσα υλοποίηση, ο χρήστης μπορεί να προσθέσει μία διαφορετική εικόνα αλλάζοντας το αρχείο BSDRP.qcow2 (και παράγοντας κάθε φορά το αντίστοιχο linked clone) που βρίσκεται στον κατάλογο images. Αντίστοιχη λειτουργικότητα μπορεί να προσφέρεται στο γραφικό περιβάλλον της εφαρμογής.
- Δημιουργία REST²³ API γύρω από την εφαρμογή. Η προσθήκη αυτή επικεντρώνεται στην αποστολή αιτημάτων προς τον server και την απόκριση του τελευταίου με έναν προκαθορισμένο τρόπο, σε μορφή JSON. Αυτός ο τρόπος επικοινωνίας επιτρέπει επίσης ευκολότερες συναλλαγές μεταξύ μηχανών (machine to machine).
- Απομακρυσμένη πρόσβαση στην κονσόλα των εικονικών μηχανών. Με τον τρόπο που είναι δομημένη η εφαρμογή, το γραφικό περιβάλλον είναι προσβάσιμο από άλλους υπολογιστές εντός του τοπικού δικτύου του hypervisor, με εξαίρεση το τερματικό των εικονικών μηχανών. Αυτό συμβαίνει διότι τα τερματικά φιλοξενούνται από τις ίδιες τις εικονικές μηχανές, στις διευθύνσεις IP των οποίων έχει πρόσβαση μόνο ο host. Με τις κατάλληλες τροποποιήσεις (PAT²⁴ ή/και VPN²⁵), οι εικονικές μηχανές μπορούν να είναι προσβάσιμες από τους χρήστες του τοπικού δικτύου.
- Προβολή επιπλέον πληροφοριών για τις εικονικές μηχανές εντός του γραφικού περιβάλλοντος. Πέρα από την κατάσταση της κάθε εικονικής μηχανής (αν λειτουργεί ή όχι), μπορούν να παρουσιαστούν επιπλέον στοιχεία στο γραφικό περιβάλλον, όπως η κίνηση των εικονικοποιημένων δικτύων, η χρήση υπολογιστικών πόρων (CPU, RAM, κ.ά.), οι τρέχουσες διευθύνσεις IP και άλλα στατιστικά στοιχεία που αφορούν τις εικονικές μηχανές.

²³ REpresentational State Transfer.

²⁴ Port Address Translation.

²⁵ Virtual Private Network.

Παράρτημα Α

Πηγαίος κώδικας της εφαρμογής

app.py

```
''' Main Flask routing script '''

import os
from datetime import timedelta
import json
import re

from flask import Flask, send_from_directory, render_template, url_for, request, redirect, session
import libvirt

from libvirt_domain import create_router, create_pc, simpleHash, start_domain, shutdown_domain,
remove_domain, domain_status, dhcp_leases, cleanup

active_r = []
active_pc = []
active_net_r = []
active_net_pc = []
active_netconf_r = []
active_netconf_pc = []
status_r = []
status_pc = []

app = Flask(__name__)
app.config['SECRET_KEY'] = 'eiaj38dx09'
app.config['SESSION_COOKIE_HTTPONLY'] = True
app.config['SESSION_COOKIE_SAMESITE'] = 'Lax'
app.permanent_session_lifetime = timedelta(days = 2)

@app.route('/favicon.ico')
def favicon():
    ''' Returns favicon '''

    return send_from_directory(os.path.join(app.root_path, 'static'), 'favicon.ico')
```

```
@app.route('/', methods=['POST', 'GET'])
def index():
    ''' Handles index.html '''

    session.permanent = True
    session['current_page'] = request.endpoint

    active_net_r = []
    active_net_pc = []
    active_netconf_r = []
    active_netconf_pc = []
    if 'active_r' not in session and 'active_pc' not in session:
        print('NO SESSION')
        session['active_r'] = []
        session['active_pc'] = []
        session['active_net_r'] = []
        session['active_net_pc'] = []
        session['active_netconf_r'] = []
        session['active_netconf_pc'] = []
        session['status_r'] = []
        session['status_pc'] = []
    active_net_r = session['active_net_r']
    active_net_pc = session['active_net_pc']
    print(session['active_r'], session['active_pc'])
    print(session['active_net_r'], session['active_net_pc'])
    allDomainsStatus()
    print(session['status_r'], session['status_pc'])
    if request.method == 'POST':
        num_r = int(request.form['num_r'])
        num_pc = int(request.form['num_pc'])
        session['num_r'] = num_r
        session['num_pc'] = num_pc
        net_r = []
        net_pc = []
        name_r = []
        name_pc = []
```

```

net_conf_data = request.form.to_dict(flat = False)

j = 0
if num_r != 0:
    name_r = request.form.getlist('name_r')
    net_r = request.form.getlist('net_r')
    netconf_r = [[] for i in net_r]
    for i in range(len(net_r)):
        for elem in range(int(net_r[i])):
            netconf_r[i].append(net_conf_data['interface_type'][j])
            j = j + 1
    session['active_r'].extend(name_r)
    session['active_net_r'].extend(net_r)
    session['netconf_r'] = netconf_r
    session['active_netconf_r'].extend(netconf_r)
if num_pc != 0:
    name_pc = request.form.getlist('name_pc')
    net_pc = request.form.getlist('net_pc')
    netconf_pc = [[] for i in net_pc]
    for i in range(len(net_pc)):
        for elem in range(int(net_pc[i])):
            netconf_pc[i].append(net_conf_data['interface_type'][j])
            j = j + 1
    session['active_pc'].extend(name_pc)
    session['active_net_pc'].extend(net_pc)
    session['netconf_pc'] = netconf_pc
    session['active_netconf_pc'].extend(netconf_pc)

print(session['active_netconf_r'], session['active_netconf_pc'])

j = 0
if num_r != 0:
    for key, val in enumerate(netconf_r):
        try:
            create_router(name_r[j], netconf_r[key])
            j = j + 1

```

```

        except libvirt.libvirtError:
            print('Domain has not been created')
            return redirect(url_for('index'))

    k = j
    if num_pc != 0:
        for key, val in enumerate(netconf_pc):
            try:
                create_pc(name_pc[j - k], netconf_pc[key])
                j = j + 1
            except libvirt.libvirtError:
                print('Domain has not been created')
                return redirect(url_for('index'))

        return redirect(url_for('created'))
    else:
        return render_template('index.html', \
                                active_r = json.dumps(session['active_r']), active_pc = \
                                json.dumps(session['active_pc']), \
                                active_net_r = session['active_net_r'], active_net_pc = session['active_net_pc'], \
                                active_net_r_json = json.dumps(session['active_net_r']), active_net_pc_json = \
                                json.dumps(session['active_net_pc']), \
                                active_netconf_r = session['active_netconf_r'], active_netconf_pc = \
                                session['active_netconf_pc'], \
                                active_netconf_r_json = json.dumps(session['active_netconf_r']), active_netconf_pc_json \
                                = json.dumps(session['active_netconf_pc']), \
                                status_r = json.dumps(session['status_r']), status_pc = \
                                json.dumps(session['status_pc']))

@app.route('/created', methods=['POST', 'GET'])
def created():
    ''' Handles created.html '''

    session['current_page'] = request.endpoint

    number_r = session['num_r']
    number_pc = session['num_pc']
    session['num_r'] = '0'
    session['num_pc'] = '0'

    allDomainsStatus()

```

```

print(session['active_r'], session['active_pc'])
print(session['active_net_r'], session['active_net_pc'])
print(session['active_netconf_r'], session['active_netconf_pc'])
print(session['status_r'], session['status_pc'])

if session['active_r'] == [] and session['active_pc'] == []:
    return(redirect(url_for('index')))
return render_template('created.html', \
    number_r = number_r, number_pc = number_pc, \
    active_r = json.dumps(session['active_r']), active_pc = json.dumps(session['active_pc']), \
    active_net_r = session['active_net_r'], active_net_pc = session['active_net_pc'], \
    active_net_r_json = json.dumps(session['active_net_r']), active_net_pc_json = \
    json.dumps(session['active_net_pc']), \
    active_netconf_r = session['active_netconf_r'], active_netconf_pc = \
    session['active_netconf_pc'], \
    active_netconf_r_json = json.dumps(session['active_netconf_r']), active_netconf_pc_json = \
    json.dumps(session['active_netconf_pc']), \
    status_r = session['status_r'], status_pc = session['status_pc'])

@app.route('/domain_start', methods=['POST', 'GET'])
def domain_start():
    ''' Starts selected domain '''

    domain = request.args.get('domain')
    start_domain(domain)

    if domain in session['active_r']:
        domain_index = session['active_r'].index(domain)
        session['status_r'][domain_index] = '1'
    if domain in session['active_pc']:
        domain_index = session['active_pc'].index(domain)
        session['status_pc'][domain_index] = '1'
    return redirect(url_for(session['current_page']))

@app.route('/domain_shutdown', methods=['POST', 'GET'])
def domain_shutdown():
    ''' Shuts down selected domain '''

    domain = request.args.get('domain')

```

```

shutdown_domain(domain)

if domain in session['active_r']:
    domain_index = session['active_r'].index(domain)
    session['status_r'][domain_index] = '5'
if domain in session['active_pc']:
    domain_index = session['active_pc'].index(domain)
    session['status_pc'][domain_index] = '5'
return redirect(url_for(session['current_page']))

@app.route('/domain_remove', methods=['POST', 'GET'])
def domain_remove():
    ''' Removes selected domain '''

    domain = request.args.get('domain')
    remove_domain(domain)
    with open('domains_xml/domains.txt', 'r') as fin:
        lines = fin.readlines()
    with open('domains_xml/domains.txt', 'w') as fout:
        for line in lines:
            if domain not in line:
                fout.write(line)
    if domain in session['active_r']:
        domain_index = session['active_r'].index(domain)
        if domain in session['active_r']:
            del session['active_r'][domain_index]
            del session['active_net_r'][domain_index]
            del session['active_netconf_r'][domain_index]
            del session['status_r'][domain_index]
    if domain in session['active_pc']:
        domain_index = session['active_pc'].index(domain)
        if domain in session['active_pc']:
            del session['active_pc'][domain_index]
            del session['active_net_pc'][domain_index]
            del session['active_netconf_pc'][domain_index]
            del session['status_pc'][domain_index]
    return redirect(url_for(session['current_page']))

```



```
@app.route('/domains_cleanup', methods=['POST', 'GET'])
def domains_cleanup():
    ''' Removes all domains '''

    cleanup()
    session.clear()

    active_r.clear()
    active_pc.clear()
    active_net_r.clear()
    active_net_pc.clear()
    active_netconf_r.clear()
    active_netconf_pc.clear()
    status_r.clear()
    status_pc.clear()

    return redirect(url_for('index'))

@app.route('/xterm/<domain>', methods=['POST', 'GET'])
def xterm(domain):
    ''' Opens console for selected domain '''

    second_octet = '21'
    if domain in session['active_r']:
        second_octet = '22'
    if domain.startswith('R') or domain.startswith('PC'):
        domain_number = re.sub('[PCR]', '', domain)
    else:
        domain_number = simpleHash(domain)
    with open('domains_xml/domains.txt') as file:
        if domain in file.read():
            print('Exists')
            xterm_url = 'http://172.' + second_octet + '.' + str(domain_number) + '.1'
            return redirect(xterm_url)
        else:
            return render_template('404.html')
```

```
@app.route('/leases', methods=['GET', 'POST'])
def leases():
    ''' Lists info on DHCP leases '''

    active_net_leases = dhcp_leases()
    return render_template('leases.html', active_net_leases = active_net_leases)

def allDomainsStatus():
    ''' Returns the status of all defined domains '''

    session['status_r'] = []
    session['status_pc'] = []

    for i in session['active_r']:
        session['status_r'].extend(str(domain_status(i)))
    for i in session['active_pc']:
        session['status_pc'].extend(str(domain_status(i)))

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

libvirt_domain.py

```
''' This script contains functions which handle connection to qemu,  
domain initialization, startup, shutdown, status report, removal,  
and a function for printing DHCP leases. '''
```

```
from shutil import copyfile
```

```
from lxml import etree
```

```
import re
```

```
import hashlib
```

```
import os
```

```
import libvirt
```

```
import time
```

```
def simpleHash(s):
```

```
    ''' Returns the hash of a string '''
```

```
    result = int(hashlib.md5(s.encode('utf-8')).hexdigest(), 16)
```

```
    result = int(str(result)[:2])
```

```
    while (result > 99 or result < 49):
```

```
        if result > 99:
```

```
            result = result//2 + result//3
```

```
        if result < 49:
```

```
            result = result + result//2
```

```
    print(result)
```

```
    return result
```

```
def init_conn():
```

```
    ''' Initializes the connection to qemu '''
```

```
    try:
```

```
        conn = libvirt.open('qemu:///system')
```

```
        return conn
```

```
    except libvirt.libvirtError:
```

```
        print('Failed to connect to the hypervisor')
```

```
        return
```

```
def create_router(name_r, netconf_r: list):
    ''' Creates a router '''

    # Find existing routers
    with open('domains_xml/domains.txt', 'r') as f:
        max_r = 0
        lines = f.read().splitlines()
        for domain in lines:
            dom_number = re.sub('[PCR]', '', domain)
            if domain.startswith('R'):
                max_r = int(dom_number)
            if name_r == domain:
                print('Domain already exists')
                return

    dom_number = re.sub('[PCR]', '', name_r)

    j = max_r + 1
    if name_r.startswith('R'):
        j = int(dom_number)
    else:
        j = simpleHash(name_r)

    dom_name = name_r
    print('\nDomain', dom_name, 'will be created')

    # Print domain disk location
    abs_path = os.path.dirname(__file__)
    img_dest = os.path.join(abs_path, 'images/' + dom_name + '.qcow2')

    # Create domain disk from template
    linked_dest = os.path.join(abs_path, 'images/BSDRP_linked.qcow2')
    copyfile(linked_dest, img_dest)

    # Create network for management interface
    # Use sample domain XML
    xml_file = 'net_xml/sample_nat.xml'
```

```
# Get tree root in network XML file
tree = etree.parse(xml_file)
root = tree.getroot()

# Set name in new network XML file
attr_name = root.find('name')
attr_name.text = 'mgmt' + dom_name.lower()

# Set UUID in new network XML file
b_hex = int('20', 16)
uuid_last_mgmt = j + b_hex
uuid_last_mgmt = '{:02x}'.format(uuid_last_mgmt)
uuid = root.find('./uuid')
uuid.text = '6ac5acf9-940b-41fc-87a7-1ae02acccc' + uuid_last_mgmt

# Set bridge name in new network XML file
bridge_name = root.find('./bridge')
bridge_name.set('name', 'virbr' + str(j+100))

# Set network MAC address in new network XML file
mac_add = root.find('./mac')
mac_add.set('mac', '52:54:00:' + uuid_last_mgmt + ':cc:00')

# Set host IP in new network XML file
host_ip = root.find('./dns/host')
host_ip.set('ip', '172.22.' + str(j) + '.1')

# Set hostname in new network XML file
hostname = root.find('./dns/host/hostname')
hostname.text = dom_name

# Set DHCP IP in new network XML file
ip = root.find('./ip')
ip.set('address', '172.22.' + str(j) + '.200')

# Set host MAC and IP in new network XML file
```

```
host = root.find('./ip/dhcp/host')
host.set('mac', '52:54:00:' + uuid_last_mgmt + ':cc:01')
host.set('ip', '172.22.' + str(j) + '.1')

# Create XML for new network
xml_dest = 'net_xml/mgmt' + dom_name.lower() + '.xml'
tree.write(xml_dest)
xml_open = open(xml_dest)
xmlconfig = xml_open.read()

# Create network from new XML file
nat_network = init_conn().networkDefineXML(xmlconfig)

# Set as autostart and start network
nat_network.setAutostart(True)
nat_network.create()
print('Network mgmt' + dom_name.lower(), 'has been created')

# Create domain
# Use sample domain XML
xml_file = 'domains_xml/sample_domain.xml'

# Get tree root in domain XML file
parser = etree.XMLParser(remove_blank_text=True)
tree = etree.parse(xml_file, parser)
root = tree.getroot()

# Set name in new domain XML file
attr_name = root.find('name')
attr_name.text = dom_name

# Set image file in new domain XML file
source = root.find('./devices/disk/source')
source.set('file', img_dest)

# Set last octet of MAC address
k = '{:02x}'.format(j)
```

```
# Create new NIC in XML if applicable
for i, val in enumerate(netconf_r):
    devices = root.find('.devices')
    current_interface = val
    regex_match = re.match('NAT Network', current_interface)

    if regex_match: # NAT Network
        net_number = current_interface.split(regex_match[0], )
        net_number = int(net_number[1])
        interface = etree.Element('interface')
        interface.set('type', 'network')
        mac = etree.SubElement(interface, 'mac')
        mac.set('address', '52:54:00:c' + str(net_number) + ':4d:' + k)
        source = etree.SubElement(interface, 'source')
        source.set('network', 'network' + str(net_number))
        source.set('bridge', 'virbr' + str(net_number-1))
        model = etree.SubElement(interface, 'model')
        model.set('type', 'e1000')
        alias = etree.SubElement(interface, 'alias')
        alias.set('name', 'net' + str(i))
        address = etree.SubElement(interface, 'address')
        address.set('type', 'pci')
        address.set('domain', '0x0000')
        address.set('bus', '0x00')
        address.set('slot', '0x0' + str(i+2))
        address.set('function', '0x0')
        devices.append(interface)

elif current_interface == 'NAT':

    # Create network for NAT interface
    xml_file = 'net_xml/sample_nat.xml'
    tree = etree.parse(xml_file)
    root = tree.getroot()
    attr_name = root.find('name')
    attr_name.text = 'nat' + dom_name.lower()
```

```

b_hex = int('a0', 16)
uuid_last_nat = j + b_hex
uuid_last_nat = '{:02x}'.format(uuid_last_nat)
uuid = root.find('./uuid')
uuid.text = '6ac5acf9-940b-41fc-87a7-1ae02adddd' + uuid_last_nat
bridge_name = root.find('./bridge')
bridge_name.set('name', 'virbr' + str(j+18))
mac_add = root.find('./mac')
mac_add.set('mac', '52:54:00:' + uuid_last_nat + ':dd:00')
host_ip = root.find('./dns/host')
host_ip.set('ip', '192.168.' + str(j) + '.1')
hostname = root.find('./dns/host/hostname')
hostname.text = dom_name
ip = root.find('./ip')
ip.set('address', '192.168.' + str(j) + '.200')
host = root.find('./ip/dhcp/host')
host.set('mac', '52:54:00:' + uuid_last_nat + ':dd:01')
host.set('ip', '192.168.' + str(j) + '.1')
xml_dest = 'net_xml/mgmt' + dom_name.lower() + '.xml'
tree.write(xml_dest)
xml_open = open(xml_dest)
xmlconfig = xml_open.read()
nat_network = init_conn().networkDefineXML(xmlconfig)
nat_network.setAutostart(True)
nat_network.create()

# Create NAT interface
interface = etree.Element('interface')
interface.set('type', 'network')
mac = etree.SubElement(interface, 'mac')
mac.set('address', '52:54:00:' + uuid_last_nat + ':dd:' + k)
source = etree.SubElement(interface, 'source')
source.set('network', 'network' + str(i+1))
source.set('bridge', 'virbr' + str(i))
model = etree.SubElement(interface, 'model')
model.set('type', 'e1000')
alias = etree.SubElement(interface, 'alias')

```



```
alias.set('name', 'net' + str(i))
address = etree.SubElement(interface, 'address')
address.set('type', 'pci')
address.set('domain', '0x0000')
address.set('bus', '0x00')
address.set('slot', '0x0' + str(i+2))
address.set('function', '0x0')
devices.append(interface)

elif current_interface == 'hostonly':
    interface = etree.Element('interface')
    interface.set('type', 'bridge')
    mac = etree.SubElement(interface, 'mac')
    mac.set('address', '52:54:00:d1:4d:' + k)
    source = etree.SubElement(interface, 'source')
    source.set('bridge', 'virbr4')
    model = etree.SubElement(interface, 'model')
    model.set('type', 'e1000')
    alias = etree.SubElement(interface, 'alias')
    alias.set('name', 'hostonly')
    address = etree.SubElement(interface, 'address')
    address.set('type', 'pci')
    address.set('domain', '0x0000')
    address.set('bus', '0x00')
    address.set('slot', '0x0' + str(i+2))
    address.set('function', '0x0')
    devices.append(interface)

elif current_interface == 'bridge':
    interface = etree.Element('interface')
    interface.set('type', 'bridge')
    mac = etree.SubElement(interface, 'mac')
    mac.set('address', '52:54:00:e1:4d:' + k)
    source = etree.SubElement(interface, 'source')
    source.set('bridge', 'virbr8')
    model = etree.SubElement(interface, 'model')
    model.set('type', 'e1000')
```

```

alias = etree.SubElement(interface, 'alias')
alias.set('name', 'net' + str(i))

address = etree.SubElement(interface, 'address')
address.set('type', 'pci')
address.set('domain', '0x0000')
address.set('bus', '0x00')
address.set('slot', '0x0' + str(i+2))
address.set('function', '0x0')

devices.append(interface)

else:
    regex_match = re.match('LAN|WAN', current_interface)
    if regex_match:
        # if LAN 0, if WAN 5
        int_type = 0
        if regex_match[0] == 'WAN':
            int_type = 10

        net_number = current_interface.split(regex_match[0], )
        net_number = int(net_number[1])

        interface = etree.Element('interface')
        interface.set('type', 'bridge')

        net_number_hex = int(hex(net_number), 16)
        net_number_hex = '{:01x}'.format(net_number_hex)

        mac = etree.SubElement(interface, 'mac')
        mac.set('address', '52:54:00:' + str(int_type)[0] + net_number_hex + ':4d:' + k)

        source = etree.SubElement(interface, 'source')
        source.set('bridge', 'virbr' + str(int_type + net_number + 8))

        model = etree.SubElement(interface, 'model')
        model.set('type', 'e1000')

        alias = etree.SubElement(interface, 'alias')
        alias.set('name', 'net' + str(i))

        address = etree.SubElement(interface, 'address')
        address.set('type', 'pci')
        address.set('domain', '0x0000')
        address.set('bus', '0x00')
        address.set('slot', '0x0' + str(i+2))
        address.set('function', '0x0')

```

```

        devices.append(interface)

    i = i + 1

# Define management interface
interface = etree.Element('interface')
interface.set('type', 'network')
mac = etree.SubElement(interface, 'mac')
mac.set('address', '52:54:00:' + uuid_last_mgmt + ':cc:01')
source = etree.SubElement(interface, 'source')
source.set('network', 'mgmt' + dom_name.lower())
source.set('bridge', 'virbr' + str(j+100))
model = etree.SubElement(interface, 'model')
model.set('type', 'virtio')
alias = etree.SubElement(interface, 'alias')
alias.set('name', 'net' + str(i))
address = etree.SubElement(interface, 'address')
address.set('type', 'pci')
address.set('domain', '0x0000')
address.set('bus', '0x00')
address.set('slot', '0x0' + str(i+2))
address.set('function', '0x0')
devices.append(interface)

# Create XML for new domain
xml_dest = 'domains_xml/' + dom_name + '.xml'
tree.write(xml_dest, pretty_print=True)

# Create domain from new XML file
with open(xml_dest, 'r') as xmlconfig:
    dom = init_conn().defineXML(xmlconfig.read())

# Start domain
dom.create()
print('Guest', dom.name(), 'has booted\n')

# Append new domain to domains.txt

```

```

with open('domains_xml/domains.txt', 'a') as f:
    f.write(dom_name + '\n')

def create_pc(name_pc, netconf_pc: list):
    ''' Creates a PC '''

    # Find existing PCs
    with open('domains_xml/domains.txt', 'r') as f:
        max_pc = 0
        lines = f.read().splitlines()
        for domain in lines:
            dom_number = re.sub('[PCR]', '', domain)
            if domain.startswith('PC'):
                max_pc = int(dom_number)
            if name_pc == domain:
                print('Domain already exists')
                return

    dom_number = re.sub('[PCR]', '', name_pc)

    j = max_pc + 1
    if name_pc.startswith('PC'):
        j = int(dom_number)
    else:
        j = simpleHash(name_pc)

    dom_name = name_pc
    print('\nDomain', dom_name, 'will be created')

    # Print domain disk location
    abs_path = os.path.dirname(__file__)
    img_dest = os.path.join(abs_path, 'images/' + dom_name + '.qcow2')

    # Create domain disk from template
    linked_dest = os.path.join(abs_path, 'images/BSDRP_PC_linked.qcow2')
    copyfile(linked_dest, img_dest)

```

```
# Create network for management interface

# Use sample domain XML

xml_file = 'net_xml/sample_nat.xml'

# Get tree root in network XML file

tree = etree.parse(xml_file)

root = tree.getroot()

# Set name in new network XML file

attr_name = root.find('name')

attr_name.text = 'mgmt' + dom_name.lower()

# Set UUID in new network XML file

b_hex = int('40', 16)

uuid_last_mgmt = j + b_hex

uuid_last_mgmt = '{:02x}'.format(uuid_last_mgmt)

uuid = root.find('./uuid')

uuid.text = '6ac5acf9-940b-41fc-87a7-1ae02acccc' + uuid_last_mgmt

# Set bridge name in new network XML file

bridge_name = root.find('./bridge')

bridge_name.set('name', 'virbr' + str(j+100+24))

# Set network MAC address in new network XML file

mac_add = root.find('./mac')

mac_add.set('mac', '52:54:00:' + uuid_last_mgmt + ':cc:00')

# Set host IP in new network XML file

host_ip = root.find('./dns/host')

host_ip.set('ip', '172.21.' + str(j) + '.1')

# Set hostname in new network XML file

hostname = root.find('./dns/host/hostname')

hostname.text = dom_name

# Set DHCP IP in new network XML file

ip = root.find('./ip')
```

```
ip.set('address', '172.21.' + str(j) + '.200')

# Set host MAC and IP in new network XML file
host = root.find('./ip/dhcp/host')
host.set('mac', '52:54:00:' + uuid_last_mgmt + ':cc:01')
host.set('ip', '172.21.' + str(j) + '.1')

# Create XML for new network
xml_dest = 'net_xml/mgmt' + dom_name.lower() + '.xml'
tree.write(xml_dest)
xml_open = open(xml_dest)
xmlconfig = xml_open.read()

# Create network from new XML file
nat_network = init_conn().networkDefineXML(xmlconfig)

# Set as autostart and start network
nat_network.setAutostart(True)
nat_network.create()
print('Network mgmt' + dom_name.lower(), 'has been created')

# Create domain
# Use sample domain XML
xml_file = 'domains_xml/sample_domain.xml'

# Get tree root in domain XML file
parser = etree.XMLParser(remove_blank_text=True)
tree = etree.parse(xml_file, parser)
root = tree.getroot()

# Set name in new domain XML file
attr_name = root.find('name')
attr_name.text = dom_name

# Set image file in new domain XML file
source = root.find('./devices/disk/source')
source.set('file', img_dest)
```

```
# Set last octet of MAC address

k = '{:02x}'.format(j)

# Create new NIC in XML if applicable
for i, val in enumerate(netconf_pc):
    devices = root.find('.devices')
    current_interface = val
    regex_match = re.match('NAT Network', current_interface)

    if regex_match: # NAT Network
        net_number = current_interface.split(regex_match[0], )
        net_number = int(net_number[1])
        interface = etree.Element('interface')
        interface.set('type', 'network')
        mac = etree.SubElement(interface, 'mac')
        mac.set('address', '52:54:00:c' + str(net_number) + ':5d:' + k)
        source = etree.SubElement(interface, 'source')
        source.set('network', 'network' + str(net_number))
        source.set('bridge', 'virbr' + str(net_number-1))
        model = etree.SubElement(interface, 'model')
        model.set('type', 'e1000')
        alias = etree.SubElement(interface, 'alias')
        alias.set('name', 'net' + str(i))
        address = etree.SubElement(interface, 'address')
        address.set('type', 'pci')
        address.set('domain', '0x0000')
        address.set('bus', '0x00')
        address.set('slot', '0x0' + str(i+2))
        address.set('function', '0x0')
        devices.append(interface)

    elif current_interface == 'NAT':

        # Create network for NAT interface
        xml_file = 'net_xml/sample_nat.xml'
        tree = etree.parse(xml_file)
```

```

root = tree.getroot()
attr_name = root.find('name')
attr_name.text = 'nat' + dom_name.lower()
b_hex = int('a0', 16)
uuid_last_nat = j + b_hex
uuid_last_nat = '{:02x}'.format(uuid_last_nat)
uuid = root.find('./uuid')
uuid.text = '6ac5acf9-940b-41fc-87a7-1ae02adddd' + uuid_last_nat
bridge_name = root.find('./bridge')
bridge_name.set('name', 'virbr' + str(j+18))
mac_add = root.find('./mac')
mac_add.set('mac', '52:54:00:' + uuid_last_nat + ':dd:00')
host_ip = root.find('./dns/host')
host_ip.set('ip', '192.168.' + str(j) + '.1')
hostname = root.find('./dns/host/hostname')
hostname.text = dom_name
ip = root.find('./ip')
ip.set('address', '192.168.' + str(j) + '.200')
host = root.find('./ip/dhcp/host')
host.set('mac', '52:54:00:' + uuid_last_nat + ':dd:01')
host.set('ip', '192.168.' + str(j) + '.1')
xml_dest = 'net_xml/mgmt' + dom_name.lower() + '.xml'
tree.write(xml_dest)
xml_open = open(xml_dest)
xmlconfig = xml_open.read()
nat_network = init_conn().networkDefineXML(xmlconfig)
nat_network.setAutostart(True)
nat_network.create()

# Create NAT interface
interface = etree.Element('interface')
interface.set('type', 'network')
mac = etree.SubElement(interface, 'mac')
mac.set('address', '52:54:00:' + uuid_last_nat + ':dd:' + k)
source = etree.SubElement(interface, 'source')
source.set('network', 'network' + str(i+1))
source.set('bridge', 'virbr' + str(i))

```



```

        model = etree.SubElement(interface, 'model')
        model.set('type', 'e1000')

        alias = etree.SubElement(interface, 'alias')
        alias.set('name', 'net' + str(i))

        address = etree.SubElement(interface, 'address')
        address.set('type', 'pci')
        address.set('domain', '0x0000')
        address.set('bus', '0x00')
        address.set('slot', '0x0' + str(i+2))
        address.set('function', '0x0')

        devices.append(interface)

elif current_interface == 'hostonly':
    interface = etree.Element('interface')
    interface.set('type', 'bridge')

    mac = etree.SubElement(interface, 'mac')
    mac.set('address', '52:54:00:d1:5d:' + k)

    source = etree.SubElement(interface, 'source')
    source.set('bridge', 'virbr4')

    model = etree.SubElement(interface, 'model')
    model.set('type', 'e1000')

    alias = etree.SubElement(interface, 'alias')
    alias.set('name', 'hostonly')

    address = etree.SubElement(interface, 'address')
    address.set('type', 'pci')
    address.set('domain', '0x0000')
    address.set('bus', '0x00')
    address.set('slot', '0x0' + str(i+2))
    address.set('function', '0x0')

    devices.append(interface)

elif current_interface == 'bridge':
    interface = etree.Element('interface')
    interface.set('type', 'bridge')

    mac = etree.SubElement(interface, 'mac')
    mac.set('address', '52:54:00:e1:4d:' + k)

    source = etree.SubElement(interface, 'source')

```

```

source.set('bridge', 'virbr8')

model = etree.SubElement(interface, 'model')
model.set('type', 'e1000')

alias = etree.SubElement(interface, 'alias')
alias.set('name', 'net' + str(i))

address = etree.SubElement(interface, 'address')
address.set('type', 'pci')
address.set('domain', '0x0000')
address.set('bus', '0x00')
address.set('slot', '0x0' + str(i+2))
address.set('function', '0x0')

devices.append(interface)

else:
    regex_match = re.match('LAN|WAN', current_interface)
    if regex_match:
        # if LAN 0, if WAN 5
        int_type = 0
        if regex_match[0] == 'WAN':
            int_type = 10

        net_number = current_interface.split(regex_match[0], )
        net_number = int(net_number[1])
        interface = etree.Element('interface')
        interface.set('type', 'bridge')
        net_number_hex = int(hex(net_number), 16)
        net_number_hex = '{:01x}'.format(net_number_hex)
        mac = etree.SubElement(interface, 'mac')
        mac.set('address', '52:54:00:' + str(int_type)[0] + net_number_hex + ':5d:' + k)
        source = etree.SubElement(interface, 'source')
        source.set('bridge', 'virbr' + str(int_type + net_number + 8))
        model = etree.SubElement(interface, 'model')
        model.set('type', 'e1000')
        alias = etree.SubElement(interface, 'alias')
        alias.set('name', 'net' + str(i))
        address = etree.SubElement(interface, 'address')
        address.set('type', 'pci')
        address.set('domain', '0x0000')

```

```

        address.set('bus', '0x00')
        address.set('slot', '0x0' + str(i+2))
        address.set('function', '0x0')
        devices.append(interface)

    i = i + 1

# Define management interface
interface = etree.Element('interface')
interface.set('type', 'network')
mac = etree.SubElement(interface, 'mac')
mac.set('address', '52:54:00:' + uuid_last_mgmt + ':cc:01')
source = etree.SubElement(interface, 'source')
source.set('network', 'mgmt' + dom_name.lower())
source.set('bridge', 'virbr' + str(j+100+24))
model = etree.SubElement(interface, 'model')
model.set('type', 'virtio')
alias = etree.SubElement(interface, 'alias')
alias.set('name', 'net' + str(i))
address = etree.SubElement(interface, 'address')
address.set('type', 'pci')
address.set('domain', '0x0000')
address.set('bus', '0x00')
address.set('slot', '0x0' + str(i+2))
address.set('function', '0x0')
devices.append(interface)

# Create XML for new domain
xml_dest = 'domains_xml/' + dom_name + '.xml'
tree.write(xml_dest, pretty_print=True)

# Create domain from new XML file
with open(xml_dest, 'r') as xmlconfig:
    dom = init_conn().defineXML(xmlconfig.read())

# Start domain
dom.create()

```

```
print('Guest', dom.name(), 'has booted\n')

# Append new domain to domains.txt
with open('domains_xml/domains.txt', 'a') as f:
    f.write(dom_name + '\n')

def start_domain(domain: str):
    ''' Starts selected domain '''

    # Check if domain exists
    try:
        dom = init_conn().lookupByName(domain)
    except libvirt.libvirtError:
        print('Domain not found')
        return

    # Check if domain is shutdown
    domain_state = dom.info()[0]
    if domain_state == libvirt.VIR_DOMAIN_SHUTOFF:
        try:
            dom.create()
            print('Domain', domain, 'has booted')
        except libvirt.libvirtError:
            print('Could not start domain')
            return
    elif domain_state == libvirt.VIR_DOMAIN_RUNNING:
        print('Domain is running')

def shutdown_domain(domain: str):
    ''' Shuts down selected domain '''

    # Check if domain exists
    try:
        dom = init_conn().lookupByName(domain)
    except libvirt.libvirtError:
        print('Domain not found')
        return
```

```
# Check if domain is running
domain_state = dom.info()[0]
if domain_state == libvirt.VIR_DOMAIN_RUNNING:
    try:
        dom.destroy()
        print('Domain', domain, 'has been shutdown')
    except libvirt.libvirtError:
        print('Could not shutdown domain')
    return
elif domain_state == libvirt.VIR_DOMAIN_SHUTOFF:
    print('Domain is not running')

def remove_domain(domain: str):
    ''' Removes selected domain and management network '''

    # Check if domain exists
    try:
        dom = init_conn().lookupByName(domain)
    except libvirt.libvirtError:
        print('Domain not found')
    return

    # Check if domain is running
    domain_state = dom.info()[0]
    if domain_state == libvirt.VIR_DOMAIN_RUNNING:
        try:
            dom.destroy()
            print('Domain', domain, 'has been shutdown')
        except libvirt.libvirtError:
            print('Could not shutdown domain')
        return
    try:
        dom.undefine()
        abs_path = os.path.dirname(__file__)
        xml_dest = os.path.join(abs_path, 'domains_xml/' + domain + '.xml')
        os.remove(xml_dest)
```

```
img_dest = os.path.join(abs_path, 'images/' + domain + '.qcow2')
os.remove(img_dest)
print('Domain', domain, 'has been removed')
except libvirt.libvirtError:
    print('Could not remove domain')

# Remove management network
try:
    network = init_conn().networkLookupByName('mgmt' + domain.lower())
    network.destroy()
    network.undefine()
    abs_path = os.path.dirname(__file__)
    xml_dest = os.path.join(abs_path, 'net_xml/mgmt' + domain.lower() + '.xml')
    os.remove(xml_dest)
    print('Network mgmt' + domain.lower(), 'has been undefined')
except libvirt.libvirtError:
    print('Could not remove network')

def domain_status(domain: str):
    ''' Returns domain status '''

    # Check if domain exists
    try:
        dom = init_conn().lookupByName(domain)
    except libvirt.libvirtError:
        print('Domain not found')
        return

    # Return domain status
    domain_state = dom.info()[0]
    return domain_state

def dhcp_leases():
    ''' Returns DHCP leases '''

    networks = init_conn().listNetworks()
    networks.sort()
```

```

active_net_leases = {}

for network in networks:
    net = init_conn().networkLookupByName(network)
    leases = net.DHCPLeases()
    net_leases = []
    if leases != []:
        for lease in leases:
            expiry = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(lease['expirytime']))
            mac = lease['mac']
            addr_prefix = str(lease['ipaddr']) + '/' + str(lease['prefix'])
            hostname = lease['hostname']
            dhcp_entry = expiry, mac, addr_prefix, hostname
            net_leases.append(dhcp_entry)
        active_net_leases.update({network: net_leases})
    return(active_net_leases)

def cleanup():
    ''' Removes all domains and management networks '''

    # Open domains.txt file (read)
    with open('domains_xml/domains.txt', 'r') as f:
        lines = f.read().splitlines()

    if lines != []:
        for domain in lines:
            print('Removing', str(domain) + '...')
            try:
                dom = init_conn().lookupByName(str(domain))
                try:
                    dom.destroy()
                except libvirt.libvirtError:
                    print('Domain', str(domain), 'is not running')
                dom.undefine()
            except libvirt.libvirtError:
                print('Domain', str(domain), 'does not exist')

    # Remove management network

```

```
try:

    network = init_conn().networkLookupByName('mgmt' + str(domain.lower()))

    network.destroy()

    network.undefine()

    print('Removing network mgmt' + str(domain.lower()) + '...')


except libvirt.libvirtError:

    print('Could not remove network')


# Remove network XML
abs_path = os.path.dirname(__file__)
xml_dest = os.path.join(abs_path, 'net_xml/mgmt' + str(domain.lower()) + '.xml')
os.remove(xml_dest)


# Remove domain XML and image
xml_dest = os.path.join(abs_path, 'domains_xml/' + str(domain) + '.xml')
os.remove(xml_dest)

img_dest = os.path.join(abs_path, 'images/' + str(domain) + '.qcow2')
os.remove(img_dest)


else:

    print('No defined domains')


# Remove contents of domains.txt
with open('domains_xml/domains.txt', 'w+') as f:

    pass
```


setup.py

```
from distutils.core import setup

with open('requirements_dev.txt') as f:
    requirements = f.read().splitlines()

setup(
    name = 'LabOnLine',
    install_requires = requirements
)
```

waitress_server.py

```
#!/usr/local/bin/python3
```

```
from waitress import serve
```

```
import app
```

```
serve(app.app, host='0.0.0.0', port=5000)
```

sample_domain.xml

```
<domain type='kvm'>
  <name>BSDRP</name>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://freebsd.org/freebsd/12.0"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>524288</memory>
  <currentMemory unit='KiB'>524288</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-i440fx-groovy'>hvm</type>
    <boot dev='hd' />
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off' />
  </features>
  <cpu mode='host-model' check='partial' />
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup' />
    <timer name='pit' tickpolicy='delay' />
    <timer name='hpet' present='no' />
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no' />
    <suspend-to-disk enabled='no' />
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' />
    </disk>
  </devices>
</domain>
```

```

<source file='./images/BSDRP_linked.qcow2' />
<target dev='vda' bus='virtio' />
<address type='pci' domain='0x0000' bus='0x00' slot='0x09' function='0x0' />
</disk>
<controller type='usb' index='0' model='ich9-ehci1'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x7' />
</controller>
<controller type='usb' index='0' model='ich9-uhci1'>
  <master startport='0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0' multifunction='on' />
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
  <master startport='2' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x1' />
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
  <master startport='4' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x2' />
</controller>
<controller type='pci' index='0' model='pci-root' />
<controller type='virtio-serial' index='0'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0b' function='0x0' />
</controller>
<serial type='pty'>
  <target type='isa-serial' port='0'>
    <model name='isa-serial' />
  </target>
</serial>
<console type='pty'>
  <target type='serial' port='0' />
</console>
<channel type='spicevmc'>
  <target type='virtio' name='com.redhat.spice.0' />
  <address type='virtio-serial' controller='0' bus='0' port='1' />
</channel>
<input type='mouse' bus='ps2' />
<input type='keyboard' bus='ps2' />

```

```
<graphics type='spice' autoport='yes'>
  <listen type='address' />
  <image compression='off' />
</graphics>
<sound model='ich6'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0c' function='0x0' />
</sound>
<video>
  <model type='qxl' ram='65536' vram='65536' vgamem='16384' heads='1' primary='yes' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0d' function='0x0' />
</video>
<redirdev bus='usb' type='spicevmc'>
  <address type='usb' bus='0' port='1' />
</redirdev>
<redirdev bus='usb' type='spicevmc'>
  <address type='usb' bus='0' port='2' />
</redirdev>
<memballoon model='virtio'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0e' function='0x0' />
</memballoon>
</devices>
</domain>
```

sample_nat.xml

```
<network>
  <name></name>
  <uuid></uuid>
  <forward mode='nat' />
  <bridge stp='on' delay='0' />
  <mac />
  <dns>
    <host>
      <hostname></hostname>
    </host>
  </dns>
  <ip netmask='255.255.255.0'>
    <dhcp>
      <host />
    </dhcp>
  </ip>
</network>
```

base.html

```
{% include 'head.html' %}

{% block head %}{% endblock %}

<h2><a href="{{ url_for('index') }}">LabOnLine</a></h2>

{% if session['active_r'] != [] or session['active_pc'] != [] %}

    <button type="submit" id="resetPosition" onclick="resetPosition()">Reset position</button>

    <form action="{{ url_for('domains_cleanup') }}">

        <button type="submit" id="removeAll" onclick="deleteCookies()">Remove all</button>

    </form>

    <form action="{{ url_for('leases') }}">

        <button type="submit" id="dhcp_button">DHCP Leases</button>

    </form><br /><br />

{% endif %}

{% block body %}{% endblock %}

{% block devices %}

<div id="routers">

    {% for p in range(session['active_r']|length): %}

        {% set dom = session['active_r'][p]|string %}

        <div id="{{ dom }}" class="draggable">

            <div id="router_name">

                <h4>{{ (session['active_r'][p]|string) }}</h4>

                <div id="circle_{{ session['active_r'][p]|string }}"></div>

            </div>

            <div id="router_specs">

                <script>

                    $(document).ready(function(){

                        var domain = "{{ dom }}";

                        var p = "{{ p }}";

                        var status_r = {{ status_r|safe }};

                        if (status_r[p] == '5') {

                            $("#shutdown_" + domain).attr("disabled", "disabled");

                        }

                    });

                </script>

            </div>

        </div>

    {% endfor %}

</div>

{% endblock %}
```

```

        $("#console_" + domain).attr("disabled", "disabled");
        $("#circle_" + domain).attr("style", "background-color: #d7292f")
    }
    if (status_r[p] == '1') {
        $("#start_" + domain).attr("disabled", "disabled");
        $("#circle_" + domain).attr("style", "background-color: #009e26");
    }
});
</script>

<form target="_blank" action="{{ url_for('xterm', domain = dom) }}" method="POST">
    <button type="submit" class="control" id="console_{{ dom }}" value="Console">
        
    </button>
</form>

<form action="{{ url_for('domain_remove', domain = dom) }}" method="POST">
    <button type="submit" class="control">
        
    </button>
</form>

<form action="{{ url_for('domain_start', domain = dom) }}" method="POST">
    <button type="submit" class="control" id="start_{{ dom }}">
        
    </button>
</form>

<form action="{{ url_for('domain_shutdown', domain = dom) }}" method="POST">
    <button type="submit" class="control" id="shutdown_{{ dom }}">
        
    </button>
</form>

{% for i in range((active_net_r[p]|int): %)
    <div class="connections">
        

        <p1>{{ session['active_netconf_r'][p][i] }}</p1>
    </div>

```



```

        {% endfor %}

    </div>

</div>

{% endfor %}

</div>
<div id="pcs">
    {% for p in range(session['active_pc']|length): %}
        {% set dom = session['active_pc'][p]|string %}
        <div id="{{ dom }}" class="draggable">
            <div id="pc_name">
                <h4>{{ (session['active_pc'][p])|string }}</h4>
                <div id="circle_{{ session['active_pc'][p]|string }}"></div>
            </div>
            <div id="pc_specs">
                <script>
                    $(document).ready(function(){
                        var domain = "{{ dom }}";
                        var p = "{{ p }}";
                        var status_pc = {{ status_pc|safe }};
                        if (status_pc[p] == '5') {
                            $("#shutdown_" + domain).attr("disabled", "disabled");
                            $("#console_" + domain).attr("disabled", "disabled");
                            $("#circle_" + domain).attr("style", "background-color: #d7292f");
                        }
                        if (status_pc[p] == '1') {
                            $("#start_" + domain).attr("disabled", "disabled");
                            $("#circle_" + domain).attr("style", "background-color: #009e26");
                        }
                    });
                </script>
                <form target="_blank" action="{{ url_for('xterm', domain = dom) }}" method="POST">
                    <button type="submit" class="control" id="console_{{ dom }}" value="Console">
                        
                    </button>
                </form>
                <form action="{{ url_for('domain_remove', domain = dom) }}" method="POST">

```

```

        <button type="submit" class="control">
            

        </button>

    </form>

    <form action="{{ url_for('domain_start', domain = dom) }}" method="POST">
        <button type="submit" class="control" id="start_{{ dom }}">
            

        </button>

    </form>

    <form action="{{ url_for('domain_shutdown', domain = dom) }}" method="POST">
        <button type="submit" class="control" id="shutdown_{{ dom }}">
            

        </button>

    </form>

    {% for i in range((active_net_pc[p])|int): %}
        <div class="connections">
            

            <p1>{{ session['active_netconf_pc'][p][i] }}</p1>

        </div>

    {% endfor %}

</div>

</div>

{% endfor %}

</div>

{% endblock %}

{% block createnew %}{% endblock %}

<script>
    $(".draggable").draggable({
        drag: function() {
            document.querySelectorAll("div[id^=line_]").forEach(e => e.remove());
            areConnected();
        },

```

```

        stop: function(event, ui) {
            var deviceId = this.id;
            document.cookie = deviceId + '=' + JSON.stringify(ui.position);
        }
    });

    function getCookie(cname) {
        var name = cname + '=';
        var ca = document.cookie.split(';');
        for(var i=0; i<ca.length; i++) {
            var c = ca[i].trim();
            if (c.indexOf(name)==0) return c.substring(name.length,c.length);
        }
    }

    $(document).ready(function() {
        var active_r_json = [{ session['active_r']|safe }];
        var active_pc_json = [{ session['active_pc']|safe }];
        const initial_position = {"top":0,"left":0};

        for (var i = 0; i < active_r_json.length; i++) {
            var exists = getCookie(active_r_json[i])
            if (!exists) {
                document.cookie = active_r_json[i] + '=' + JSON.stringify(initial_position);
            } else {
                $("##" + active_r_json[i]).css({ top: JSON.parse(getCookie(active_r_json[i])).top,
                left: JSON.parse(getCookie(active_r_json[i])).left });
            }
        }

        for (var i = 0; i < active_pc_json.length; i++) {
            var exists = getCookie(active_pc_json[i])

            if (!exists) {
                document.cookie = active_pc_json[i] + '=' + JSON.stringify(initial_position);
            } else {
                $("##" + active_pc_json[i]).css({ top: JSON.parse(getCookie(active_pc_json[i])).top,
                left: JSON.parse(getCookie(active_pc_json[i])).left });
            }
        }
    }

```

```

    }
    areConnected();
  });

function areConnected() {
  var active_r_json = {{ session['active_r']|safe }};
  var active_pc_json = {{ session['active_pc']|safe }};
  var active_netconf_r_json = {{ active_netconf_r_json|safe }};
  var active_netconf_pc_json = {{ active_netconf_pc_json|safe }};
  var allDevices = active_r_json.concat(active_pc_json);
  var allInterfaces = active_netconf_r_json.concat(active_netconf_pc_json);

  var devices = new Array;
  var interfaces = new Array;
  for (i = 0; i < allDevices.length; i++) {
    for (j = i + 1; j < allDevices.length; j++) {
      devices.push([allDevices[i], allDevices[j]]);
      interfaces.push(allInterfaces[i].concat(allInterfaces[j]));
    }
  }

  for (const[i, interface] of interfaces.entries()) {
    var setOf = new Set(interface);
    if (setOf.size !== interface.length && interface !== 'NAT' && ((active_r_json.length !==
0 && (devices[i][0][0] !== 'P' || devices[i][1][0] !== 'P')) || (active_r_json.length === 0 &&
(devices[i][0][0] == 'P' || devices[i][1][0] == 'P')))) {
      var div1 = document.getElementById(devices[i][0]);
      var div2 = document.getElementById(devices[i][1]);
      connect(div1, div2);
    }
  }
}

function getOffset(el) {
  var rect = el.getBoundingClientRect();
  return {
    left: rect.left + window.pageXOffset,
    top: rect.top + window.pageYOffset,
  }
}

```

```

        width: rect.width || el.offsetWidth,
        height: rect.height || el.offsetHeight
    };
}

function connect(div1, div2) {
    var off1 = getOffset(div1);
    var off2 = getOffset(div2);

    // bottom right
    var x1 = off1.left + off1.width/2;
    var y1 = off1.top + off1.height/2;

    // top right
    var x2 = off2.left + off2.width/2;
    var y2 = off2.top + off2.height/2;

    var length = Math.sqrt(((x2-x1) * (x2-x1)) + ((y2-y1) * (y2-y1)));

    // center
    var cx = ((x1 + x2)/2) - (length/2);
    var cy = ((y1 + y2)/2) - (4/2);

    var angle = Math.atan2((y1-y2),(x1-x2))*(180/Math.PI);

    var line = "<div id='line_" + div1.id + "_" + div2.id + "' style='"
        + "left: " + cx + "px; top: " + cy + "px; width: "
        + length + "px; transform: rotate(" + angle + "deg);' />";

    $(div1).after(line);
}

function resetPosition() {
    var routers = [{ session['active_r']|safe }];
    var pcs = [{ session['active_pc']|safe }];
    const initial_position = {"top":0,"left":0};

    for (var i = 0; i < routers.length; i++) {
        document.cookie = routers[i] + '=' + JSON.stringify(initial_position);
    }

    for (var i = 0; i < pcs.length; i++) {
        document.cookie = pcs[i] + '=' + JSON.stringify(initial_position);
    }
}

```

```
        window.location.reload();
    }

    function deleteCookies() {
        var allCookies = document.cookie.split(';');
        for (var i = 0; i < allCookies.length; i++) {
            document.cookie = allCookies[i] + ";;expires=" + new Date(0).toUTCString();
        }
    }
</script>
```

index.html

```
{% extends 'base.html' %}

{% block head %}
<title>Home | LabOnLine</title>
{% endblock %}

{% block createnew %}
<div id="createnew">
    <form method="POST">
        <p1>Select the number of VMs to create</p1>
        <br />
        <p1>Press Load to configure network interfaces:</p1>
        <div id="new_vm_numbers">
            <p1>Routers:</p1>
            <input name="num_r" id="num_r" class="input_number" type="number" value=1 min=0 max=24>
            <p1 id="number_r" hidden="hidden"></p1>
            <p1>PCs:</p1>
            <input name="num_pc" id="num_pc" class="input_number" type="number" value=1 min=0 max=24>
            <p1 id="number_pc" hidden="hidden"></p1>
            <button type="button" id="load">Load</button>
        </div>
        <div id="routers_new"></div>
        <div id="pcs_new"></div>
        <button type="button" id="load_interface" hidden="hidden">Load interface
selection</button><br />
        <input type="submit" hidden="hidden" id="submit" value="Submit">
    </form><br />
</div>

<script>
$("#load").click(function(){
    $("#load").remove();
    $("#load_interface").removeAttr("hidden");

    $("#num_r").attr("hidden", "hidden");
    num_r = $("#num_r").val();
```

```

$("#number_r").removeAttr("hidden");
$("#number_r").append(num_r);
var last_number_r = new Array;
var domains_r = [{ session['active_r']|safe }];
var domains_r_only = new Array;
for (var i=0; i<domains_r.length; i++) {
    if (domains_r[i].startsWith('R')) {
        domains_r_only.push(parseInt(domains_r[i].match(/\d+/)));
    }
}
domains_r_only.sort();
for (var i=1; i<=domains_r_only.length+num_r; i++){
    if (domains_r_only.indexOf(i) == -1) {
        if (last_number_r.length <= num_r) {
            last_number_r.push(i);
        } else {
            break;
        }
    }
}

for (var i=0; i<num_r; i++) {
    $("#routers_new").append('<div id="router_new"><div id="router_name">'
        + '<h4 id="name_r' + (i+1) + '" hidden="hidden"></h4>'
        + '<input name="name_r" id="input_name_r' + (i+1)
        + '" class="input_name" type="text" value="R' + (last_number_r[i])
        + '"></input></div><div id="r' + (i+1) + '" _specs"><p1>Number of interfaces: </p1>'
        + '<p1 id="number_int_r' + (i+1) + '" hidden="hidden"></p1><input name='
        + '"net_r" id="net_r' + (i+1) + '" class="input_number" type=number'
        + ' value=1 min=1 max=4 /></div></div><br />');
}

$("#num_pc").attr("hidden", "hidden");
num_pc = $("#num_pc").val();
$("#number_pc").removeAttr("hidden");
$("#number_pc").append(num_pc);
var last_number_pc = new Array;

```



```

var domains_pc = {{ session['active_pc']|safe }};
var domains_pc_only = new Array;
for (var i=0; i<domains_pc.length; i++) {
    if (domains_pc[i].startsWith('PC')) {
        domains_pc_only.push(parseInt(domains_pc[i].match(/\d+/)));
    }
}
domains_pc_only.sort();
for (var i=1; i<=domains_pc_only.length+num_pc; i++){
    if (domains_pc_only.indexOf(i) == -1) {
        if (last_number_pc.length <= num_pc) {
            last_number_pc.push(i);
        } else {
            break;
        }
    }
}
for (var i=0; i<num_pc; i++) {
    $("#pcs_new").append('<div id="pc_new"><div id="pc_name">'
        + '<h4 id="name_pc" + (i+1) + '" hidden="hidden"></h4>'
        + '<input name="name_pc" id="input_name_pc" + (i+1)
        + '" class="input_name" type="text" value="PC" + (last_number_pc[i])
        + '"></input></div><div id="pc" + (i+1) + "_specs"><p1>Number of interfaces: </p1>'
        + '<p1 id="number_int_pc" + (i+1) + '" hidden="hidden"></p1><input name='
        + '"net_pc" id="net_pc" + (i+1) + '" class="input_number" type=number'
        + ' value=1 min=1 max=4 /></div></div><br />');
}
});
$(document).on("click", "#load_interface", function(){
    $("#load_interface").remove();
    $("#submit").removeAttr("hidden");
    var interface_select = '</p1><select name="interface_type" id="interface_type">'
        + '<option value="LAN1" selected="selected">LAN1</option>'
        + '<option value="LAN2">LAN2</option>'
        + '<option value="LAN3">LAN3</option>'
        + '<option value="LAN4">LAN4</option>'
        + '<option value="LAN5">LAN5</option>'

```

```

+ '<option value="LAN6">LAN6</option>'
+ '<option value="LAN7">LAN7</option>'
+ '<option value="LAN8">LAN8</option>'
+ '<option value="LAN9">LAN9</option>'
+ '<option value="LAN10">LAN10</option>'
+ '<option value="WAN1">WAN1</option>'
+ '<option value="WAN2">WAN2</option>'
+ '<option value="WAN3">WAN3</option>'
+ '<option value="WAN4">WAN4</option>'
+ '<option value="WAN5">WAN5</option>'
+ '<option value="WAN6">WAN6</option>'
+ '<option value="WAN7">WAN7</option>'
+ '<option value="WAN8">WAN8</option>'
+ '<option value="WAN9">WAN9</option>'
+ '<option value="WAN10">WAN10</option>'
+ '<option value="NAT">NAT</option>'
+ '<option value="hostonly">Host-Only</option>'
+ '<option value="bridge">Bridge</option>'
+ '<option value="NAT Network 1">NAT Network 1</option>'
+ '<option value="NAT Network 2">NAT Network 2</option>'
+ '<option value="NAT Network 3">NAT Network 3</option>'
+ '<option value="NAT Network 4">NAT Network 4</option></select>';

var name_r = new Array();
var net_arr_r = new Array();
for (i=0; i<num_r; i++) {
    name_r[i] = $("#input_name_r" + (i+1)).val();
    net_arr_r[i] = parseInt($("#net_r" + (i+1)).val());
    $("#input_name_r" + (i+1)).attr("hidden", "hidden");
    $("#name_r" + (i+1)).append(name_r[i]);
    $("#name_r" + (i+1)).removeAttr("hidden");
    $("#net_r" + (i+1)).attr("hidden", "hidden");
    $("#number_int_r" + (i+1)).append(net_arr_r[i]);
    $("#number_int_r" + (i+1)).removeAttr("hidden");
    for (j=0; j<net_arr_r[i]; j++) {
        $("#r" + (i+1) + "_specs").append("<br><p1>Interface " + (j+1) + ": " +
interface_select);
    }
}

```

```

    }

    var name_pc = new Array();
    var net_arr_pc = new Array();
    for (i=0; i<num_pc; i++) {
        name_pc[i] = $("#input_name_pc" + (i+1)).val();
        net_arr_pc[i] = parseInt($("#net_pc" + (i+1)).val());
        $("#input_name_pc" + (i+1)).attr("hidden", "hidden");
        $("#name_pc" + (i+1)).append(name_pc[i]);
        $("#name_pc" + (i+1)).removeAttr("hidden");
        $("#net_pc" + (i+1)).attr("hidden", "hidden");
        $("#number_int_pc" + (i+1)).append(net_arr_pc[i]);
        $("#number_int_pc" + (i+1)).removeAttr("hidden");
        for (j=0; j<net_arr_pc[i]; j++) {
            $("#pc" + (i+1) + "_specs").append("<br><p1>Interface " + (j+1) + ": " +
interface_select);
        }
    }
});
</script>
{% endblock %}

```

created.html

```
{% extends 'base.html' %}

{% block head %}
<title>Created | LabOnLine</title>
{% endblock %}

{% block body %}

{% set total = number_r|int + number_pc|int %}
{% if total == 0 %}

    <p1></p1>

{% elif total == 1 %}

    <p1>{{ total }} domain has been created<br /><br /></p1>

{% else %}

    <p1>{{ total }} domains have been created<br /><br /></p1>

{% endif %}

{% endblock %}
```

leases.html

```
{% include 'head.html' %}

{% block body %}

<h2><a href="{{ url_for('index') }}">LabOnLine</a></h2><br />
<h4>DHCP Leases:</h4>
<div id="dhcp_leases">
    <table style="border: none;">
        {% for network, leases in active_net_leases.items(): %}
            <tr><td>{{ network }}</td></tr>
            {% for lease in leases: %}
                <tr>
                    {% for i in lease: %}
                        <td>{{ i }} &nbsp;</td>
                    {% endfor %}
                </tr>
            {% endfor %}
        {% endfor %}
    </table>
</div>

<script>
    setInterval('windowReload()', 3000);
    function windowReload() {
        $('#dhcp_leases').load(location.href + ' #dhcp_leases');
    }
</script>

{% endblock %}
```

network_init.sh

```
#!/usr/bin/env bash

virsh net-destroy default
virsh net-undefine default

for file in net_xml/*; do
    virsh net-define $file
done

for i in $(seq 1 4); do
    virsh net-autostart network${i}
    virsh net-start network${i}
done

for i in $(seq 1 10); do
    virsh net-autostart LAN${i}
    virsh net-start LAN${i}
done

for i in $(seq 1 10); do
    virsh net-autostart WAN${i}
    virsh net-start WAN${i}
done

virsh net-autostart hostonly
virsh net-start hostonly
virsh net-autostart bridge
virsh net-start bridge
```

network_rm.sh

```
#!/usr/bin/env bash
```

```
for i in $(seq 1 4); do
    virsh net-destroy network${i}
    virsh net-undefine network${i}
done
```

```
for i in $(seq 1 10); do
    virsh net-destroy LAN${i}
    virsh net-undefine LAN${i}
done
```

```
for i in $(seq 1 10); do
    virsh net-destroy WAN${i}
    virsh net-undefine WAN${i}
done
```

```
virsh net-destroy hostonly
virsh net-undefine hostonly
virsh net-destroy bridge
virsh net-undefine bridge
```


Παράρτημα Β

Προκαθορισμένες διευθύνσεις MAC & IP των εικονικοποιημένων δικτύων

Στον παρακάτω πίνακα παρουσιάζονται συνοπτικά τα εικονικοποιημένα δίκτυα που έχουν δημιουργηθεί για τους σκοπούς της εφαρμογής, καθώς και η αντιστοίχιση διευθύνσεων MAC και IP σε κάθε εικονική μηχανή που δημιουργείται από αυτήν.

Δίκτυο	Διεύθυνση MAC	Διεύθυνση IP
LAN[1-10]	52 : 54 : 00 : 0[[1-10].hex] : [4 5]d : [xy.hex]	10 . 10 . [1-10] . [xy]
WAN[1-10]	52 : 54 : 00 : 1[[1-10].hex] : [4 5]d : [xy.hex]	10 . 11 . [1-10] . [xy]
Management (NAT)	52 : 54 : 00 : [[20 40].hex+xy.hex] : cc : 01	172 . [21 22] . [xy] . 1
NAT Network [1-4]	52 : 54 : 00 : c[1-4] : [4 5]d : [xy.hex]	10 . 0 . [1-4] . [xy]
Host-Only	52 : 54 : 00 : d1 : 4d : [xy]	172 . 16 . 1 . [xy]

Πίνακας 3: Προκαθορισμένες διευθύνσεις MAC & IP

Παρατίθενται ορισμένα στοιχεία για την επεξήγηση του Πίνακας 3:

- Το πρώτο μισό των διευθύνσεων MAC (52:54:00) έχει επιλεχθεί καθώς αντιστοιχεί σε κάρτες δικτύου εικονικών μηχανών KVM.
- Οι αριθμοί [1-10] αναφέρονται στον αριθμό του εκάστοτε δικτύου (LAN ή WAN).
- Οι αριθμοί [1-4] αναφέρονται στον αριθμό του εκάστοτε δικτύου (NAT Network).
- Οι αριθμοί [4|5] αντιστοιχούν σε δρομολογητή ή προσωπικό υπολογιστή αντίστοιχα.
- Οι αριθμοί [20|40] αντιστοιχούν σε δρομολογητή ή προσωπικό υπολογιστή αντίστοιχα.
- Ως .hex νοείται η αναπαράσταση του εκάστοτε αριθμού σε δεκαεξαδικό σύστημα.
- Η πράξη της πρόσθεσης (+) εκτελείται μεταξύ δεκαεξαδικών αριθμών.
- Με xy αναπαρίσταται ο αριθμός²⁶ του εκάστοτε δρομολογητή ή προσωπικού υπολογιστή (πχ. R1, R21, PC5).

²⁶ Επισημαίνεται ότι σε κάθε εικονική μηχανή ανατίθεται ένας αριθμός της μορφής xy, για λόγους διευθυνοδότησης μοναδικών MAC & IP, όπως έχει ήδη αναλυθεί. Αν η εικονική μηχανή δεν ακολουθεί την ονοματολογία Rxy/PCxy, τότε ο αριθμός xy υπολογίζεται από το δοθέν όνομα.

Για παράδειγμα, η εικονική μηχανή με όνομα R3 και διεπαφή στα δίκτυα LAN2 και WAN1 θα έχει τις εξής διευθύνσεις:

- Για το LAN2, MAC 52:54:00:02:4d:03 και IP 10.10.2.3
- Για το WAN1, MAC 52:54:00:11:4d:03 και IP 10.11.1.3
- Για το management, MAC 52:54:00:23:cc:01 και IP 172.21.3.1

Βιβλιογραφία

- [1] IBM. (n.d.-b). *Virtualization*. Retrieved March 29, 2021, from <https://www.ibm.com/cloud/learn/virtualization-a-complete-guide>
- [2] Cloudflare. (n.d.). *What Is a Virtual Machine?* Retrieved March 29, 2021, from <https://www.cloudflare.com/learning/cloud/what-is-a-virtual-machine/>
- [3] Red Hat. (n.d.). *What is a hypervisor?* Retrieved March 29, 2021, from <https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor>
- [4] Kernel-based Virtual Machine. (2021, March 11). In Wikipedia. https://en.wikipedia.org/w/index.php?title=Kernel-based_Virtual_Machine&oldid=1011573862
- [5] IBM. (n.d.). *Virtio: An I/O virtualization framework for Linux*. IBM Developer. Retrieved March 29, 2021, from <https://developer.ibm.com/technologies/linux/articles/l-virtio/>
- [6] Atlassian. (n.d.). *What is version control | Atlassian Git Tutorial*. Retrieved March 29, 2021, from <https://www.atlassian.com/git/tutorials/what-is-version-control>
- [7] Greenlaw, R., & Hepp, E. M. (2003). Internet, Overview. *Encyclopedia of Information Systems*, 667–681. <https://doi.org/10.1016/b0-12-227240-4/00096-4>
- [8] Server (computing). (2021, March 18). In Wikipedia. [https://en.wikipedia.org/w/index.php?title=Server_\(computing\)&oldid=1012771846](https://en.wikipedia.org/w/index.php?title=Server_(computing)&oldid=1012771846)
- [9] JavaScript. (2021, March 24). In Wikipedia. <https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=1014012714>
- [10] W3Techs. (2021, March 30). *Usage Statistics of JavaScript as Client-side Programming Language on Websites, March 2021*. <https://w3techs.com/technologies/details/cp-javascript/>
- [11] JS Foundation. (n.d.). *jQuery API Documentation*. JQuery. Retrieved March 30, 2021, from <https://api.jquery.com>
- [12] Python Software Foundation. (2021, April 15). *General Python FAQ — Python 3.9.4 documentation*. Python. <https://docs.python.org/3/faq/general.html>
- [13] *Welcome to Flask — Flask Documentation (1.1.x)*. (n.d.). Pallets Projects. Retrieved March 22, 2021, from <https://flask.palletsprojects.com/en/1.1.x/>
- [14] *Werkzeug — Werkzeug Documentation (1.0.x)*. (n.d.). Pallets Projects. Retrieved April 14, 2021, from <https://werkzeug.palletsprojects.com/en/1.0.x/>
- [15] *libvirt/libvirt*. (n.d.). GitHub. Retrieved March 29, 2021, from <https://github.com/libvirt/libvirt>
- [16] libvirt. (2021, February 14). In Wikipedia. <https://en.wikipedia.org/w/index.php?title=Libvirt&oldid=1006729468>
- [17] Berners-Lee, T., Fielding, R., Irvine, U. C., & Masinter, L., Xerox Corporation (1998, August). *Uniform Resource Identifiers (URI): Generic Syntax*. IETF. <https://www.ietf.org/rfc/rfc2396.txt>
- [18] Robinson, C. (2016, January 11). *qemu:///system vs qemu:///session*. Cole Robinson. <https://blog.wikichoon.com/2016/01/qemusystem-vs-qemusession.html>
- [19] Python Package Index. (n.d.). *Jinja2*. PyPI. Retrieved March 29, 2021, from <https://pypi.org/project/Jinja2/>
- [20] *pallets/jinja*. (n.d.). GitHub. Retrieved March 29, 2021, from <https://github.com/pallets/jinja>
- [21] Linux kernel. (2021, March 29). In Wikipedia. https://en.wikipedia.org/w/index.php?title=Linux_kernel&oldid=1014882193

- [22] Hauck, T. (2020, October 23). *Modular and monolithic kernels in Redhat Linux*. DistributedNetworks. <https://www.distributednetworks.com/configure-linux-components/module2/modular-monolithic-kernels.php>
- [23] Loadable kernel module. (2021, March 16). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Loadable_kernel_module&oldid=1012384891
- [24] Bitterling, P. (2010). *Operating System Kernels*. CiteSeerX, 2. http://www.mi.fu-berlin.de/inf/groups/ag-tech/teaching/2008-09_WS/S_19565_Proseminar_Technische_Informatik/bitterling09operating.pdf
- [25] FreeBSD. (2021, March 27). In *Wikipedia*. <https://en.wikipedia.org/w/index.php?title=FreeBSD&oldid=1014559924>
- [26] User space. (2021, February 11). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=User_space&oldid=1006101300
- [27] Cochard-Labbé, O. (n.d.). *ocochard/BSDRP*. GitHub. Retrieved November 2, 2020, from <https://github.com/ocochard/BSDRP>
- [28] *Overview — FRR latest documentation*. (n.d.). FRRouting. Retrieved March 30, 2021, from <http://docs.frrouting.org/en/latest/overview.html>
- [29] Filip, O. (n.d.). *The BIRD Internet Routing Daemon Project*. CZ.NIC. Retrieved March 30, 2021, from <https://bird.network.cz>
- [30] Red Hat. (n.d.). *libvirt: Network XML format*. Retrieved April 11, 2021, from <https://libvirt.org/formatnetwork.html>
- [31] Onisick, J. (n.d.). *Access Layer Network Virtualization: VN-Tag and VEPA*. Define The Cloud. Retrieved April 13, 2021, from <https://www.definethecloud.net/access-layer-network-virtualization-vn-tag-and-vepa/>
- [32] Smith, C. (n.d.). *cs01/pyxtermjs*. GitHub. Retrieved April 2, 2021, from <https://github.com/cs01/pyxtermjs>